Big Data Management
2nd Programming Assignment

Lecturer:
Christos Tryfonopoulos

Deadline: Sunday 03/06/2018 until 23.59
Assignment presentation: last lecture

IMPORTANT NOTES:

1. Assignment is done in teams of two. All submitted files MUST contain the names of the team members, their ids (AM), and username/emails in English.
2. After your assignment is complete you must submit it through the Assignments tool of e-class. Only one of the team members should submit the assignment.
3. Assignments sent by email or after the deadline will not be accepted for consideration. Notice that the deadline is firm, and after that the system will not allow you to upload your solution.
4. Please deliver all relevant files (source code, report, dataset, output, etc.) in a single compressed archive named after the surnames of both team members (e.g., Foka_Grivas.zip).
5. Plagiarism will not be tolerated, and all such cases will be immediately excluded from the course and appropriate action will be taken to notify all involved MSc stakeholders.

Counting your common Facebook friends using MapReduce

In Facebook, friendship is a bidirectional relationship; if A is a friend of B, then B is also a friend of A. In its current implementation when person A visits the Facebook page of person B, the system returns the list of all the friends of B. Let us assume that, in the light of the latest privacy scandal, when person A visits the Facebook page of person B Facebook has decided to list only common friends of A and B. Your task for this assignment is to implement the aforementioned functionality using MapReduce. To do so you need to download and setup the Hadoop MapReduce framework in your machine to help you write and debug your code. Notice that you may install/setup Hadoop for a single node cluster either in Local (Standalone) Mode (this is the default; Hadoop runs in a non-distributed mode as a single Java process, making it ideal for development and debugging) or in Pseudo-Distributed Mode (Hadoop runs on a single-node in a pseudo-distributed mode, where each Hadoop daemon runs in a separate Java process).

You may find all the necessary installation instructions for a single node cluster at: http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/SingleCluster.html

Apart from the lecture slides about MapReduce and Hadoop and you may also find more information on Hadoop at the following resources:

[1] http://developer.yahoo.com/hadoop/tutorial/
[2] https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html
[3] http://www.cloudera.com/wp-content/uploads/2010/01/4-ProgrammingWithHadoop.pdf
[4] http://hadoop.apache.org/common/docs/current/api/index.html

This programming assignment is compulsory and should be implemented in teams of two preferably in Java or Pig Latin. The recommended operating system for Hadoop is GNU/Linux, although you may also set it up -with some extra effort to build the Hadoop core- in Windows (see https://wiki.apache.org/hadoop/Hadoop2OnWindows for more information).

Required functionality

Let us assume that our resources include the friendship graph as a single file; in this file every person in the social network (e.g., PersonA in the following example) is at the beginning of each file line followed by his/her friends. The friends list is delimited using the space character as follows:

```
PersonA PersonB PersonC PersonD ...
...
```

1. (a) Given the friendship file, provide the pseudocode for the Map and Reduce functions (alongside with the Combiner -if needed) to calculate the number of friends each person in the social network has. Together with you pseudo code provide a running example and an execution schematic that explains the rationale behind the provided pseudocode.

   (b) Implement you solution in Hadoop using either native Java code or Pig Latin. The output file should contain (i) the name of the person, (ii) followed by a comma, and (iii) the number of friends this person has in the network. The persons in the output file should be in ascending lexicographic order. Below you may find an example of the expected output:

   ```
   PersonA,23
   PersonB,8
   ...
   ```

2. (a) Given the friendship file, provide the pseudocode for the Map and Reduce functions (alongside with the Combiner -if needed) to find the common friends between all pairs of persons in the social network. Together with you pseudo code provide a running example and an execution schematic that explains the rationale behind the provided pseudocode.

(b) Implement you solution in Hadoop using either native Java code or Pig Latin. The output file should contain (i) the names of two friends in ascending lexicographic order delimited with comma, (ii) followed by a space or tab, and (iii) the common friends of these two persons in ascending lexicographic order, delimited with commas. If two persons do not share any other common friends the corresponding filed will be empty in the output file. Below you may find an example of the expected output (notice that the pair of persons/friends in the first line of the example has three common friends, while the ones in the second line have no common friends):

```
PersonA,PersonB PersonC,PersonD,PersonE
PersonB,PersonF
...
```

To get a grasp of the input and output format and to test and debug your code some sample input/output files will be uploaded at the appropriate eClass module. Use the smallest input/output files for testing and debugging your code and the larger ones for the actual run and report you results for each file.

Deliverables, grading and bonus

A complete assignment is one that successfully and correctly addresses the aforementioned issues; implementations and reports that cover the requested issues only in part are graded accordingly. You may also implement extra characteristics for a maximum 20% bonus.

The assignment deliverables include (but may not be limited to):

- source files (including any necessary libraries)
- executable/jar (if any)
- the output files produced by your code for each of the provided input files
- a written report that will contain
    - instructions for setting up/running your code
    - details about your implementation that are worth noting
    - your comments on the output files you generated (did you get what expected? where you able to run the complete graph? if not why?)