Distributed Systems Assignment 3: MapReduce

Tom De Schepper, Glenn Daneels, and Steven Latré

2018-2019

1 Introduction

This document discusses the third assignment for the practical part of the Distributed Systems course. The goal of this assignment is to understand the principle and functionality of MapReduce and to apply it in a small-scale project. The assignment will need to be completed individually. In this document we will provide you with a detailed list of the features and requirements that are needed.

2 Goal

The goal of the this assignment is the construction of an application that uses the MapReduce idea to utlize a (relatively) large amount of data. In particular, the application allows a user to look for a number of statistics and information related to the stops of DeLijn. For instance, a user will need to be able to see how many stops are placed in a certain municipality (commune/city) or the number of stops per citizen of that municipality. Per data analysis a Jupyter Notebook needs to be provided, while the Spark framework is used to gather or calculate the requested information. It is very important to highlight and explain the different steps you made to fulfill each requested functionality and the logical process behind them.

3 Requirements

This section lists the different features and components that will need to be implemented in the assignment.

3.1 Data

In order for a user to request certain information, the required data for this needs to be present. In contrast to the previous assignments, you do not need to use web API calls to, amongs others, De Lijn. We provide you with three data files:

- *stops.txt* : an outprint of all the stops of De Lijn (obtained from the API).
- *citizens.txt* : an overview of the number of citizens per municipality (commune/city). This file is structered per region (e.g., Flanders), province (e.g., Antwerp), and arrondissement (e.g., Antwerp)

- *flemish_districts.txt* : an overview of the districs per municipality. This is needed as the stops of De Lijn are sometimes located per districts, while the populations are listed per (larger) municipality.
- *zipcodes.csv*: an overview of all zipcodes mapped to the town name and its geocoordinates

3.2 Functionalities

You will have to perform the following analysis on the data:

1. Number of stops per citizen in general
2. Number of stops per town
3. Sorted overview of number of stops per town per citizen
4. Number of stops per within a radius of a certain position (lat, long)
5. Overview of the distances of every stop to the location (lat, long) of the town in which the stops are located
6. Return the closest stop to a given position (lat, long)
7. Number of stops per citizen of all towns that are within a radius R of a certain position (lat, long)
8. Overview of the towns where X% of the stops is within a R radius of the location (lat, long) of the town

3.3 Notebooks

Per analysis a Jupyter Notebook needs to be provided, while the Spark framework is used to gather or calculate the requested information. Thus, the final solution should contain eight Jupyter notebooks and their respective names should be $notebook\_\{numberAssigment\}.ipynb$. Per notebook each logical step, required to obtain the requested analysis, needs to be extensively documented. Note that a web or other graphical application is not required, as this application really focuses on the logical process and utilizing correctly the MapReduce/Spark functionality.

3.4 Efficiency

As the goal of MapReduce/Spark is to process large amount of data, efficiency is thus very important. Therefor you need to think very carefully (and explain) how you perform each step in your process and always take efficiency in to account. Questions you should ask yourself are, among others, how you parse and pre-process your data, how you structure the data in Spark, how to limit the necessary actions you have to perform, which spark functionality is the most suited, etc.

4 Tools

You will use two tools to implement this assignment. Below you will find information on these tools:

- PySpark: Spark is a fast and powerful framework that provides an API to perform massive distributed processing over resilient sets of data. To support Spark in Python the PySpark tool was released.
- Jupyter Notebook: a popular application that enables you to edit, run and share Python code into a web view. It allows you to modify and re-execute parts of your code in a very flexible way. For this assignment, **we only allow Jupyter Python 2 Notebooks**.

You will find numerous tutorials on how to install/use PySpark by using a Jupyter Notebook.

5 Minimal requirements

There are minimum requirements that your solution should comply to, in order to pass for this part of the course. If these requirements are not fulfilled, it is not possible to pass:

- Per analysis a separate Jupiter Notebook needs to be present.
- It needs to be able to run your solution (e.g., open the different Notebooks, execute the different blocks of code, ... Therefore, make sure that your solution is complete upon submission.
- Make sure to use the Spark functionality to calculate the desired statistics and to performs the analysis listed above.
- Per Notebook your code needs to be split up in different blocks that contain the code required per logical step in the analysis process.
- As the logical process is a key part of this assignment, **all steps** need to be extensively documented.

Remark: if your solution complies to these minimum requirements, this does not mean that you will certainly pass this assignment.

6 Deadline and submission

The deadline for this is assignment is Sunday 6th of January 2019 at 23:59. Your solution has to be submitted through Blackboard.