Introduction to Spark

Master M2 – Université Grenoble Alpes & Grenoble INP

2022

In this lab, we are going to write our first Spark programs. To this end, we are going to conduct some analysis on a dataset of significant size. The programs are going to be run on a single node. Note however that the same code would be used to run on multiple nodes.

1 About the lab

This lab is not graded. The computations proposed in the lab are some examples of simple data processing that can be done with Spark. You are strongly encouraged to try running other analysis on the provided dataset to continue practicing.

Do not hesitate to ask questions during the lab session. If you have questions after the session, you can always send them by email to Thomas Ropars (thomas.ropars@univ-grenoble-alpes.fr).

2 Before starting

Before starting solving this lab, you need a working installation of Spark. Please refer to the following document for instructions regarding the installation: `https://tropars.github.io/downloads/lectures/LSDM/LSDM-install-spark.pdf`.

For this lab, we provide you with some initial source code that can be downloaded at `https://tropars.github.io/downloads/lectures/LSDM/LSDM-lab-intro-spark.tar.gz`.

3 The dataset

The dataset we will study in this lab comes from the project *Climatological Database for the World's Oceans*. Detailed information about this project can be found at `https://en.wikipedia.org/wiki/CLIWOC`.

In a few words, this dataset has been created starting from the logbook of ships that were navigating the oceans during the 18th and 19th century. These logbooks were maintained by the crew of the ships and contain a lot of information regarding the weather conditions during that period.

Each observation reported in the dataset includes many entries including the date of the observation, the location, the air temperature, the wind speed, etc. A detailed description of all included fields is included in the provided archive (file `CLIWOC_desc.html`).

The dataset is to be downloaded at the following address: `https://cloud.univ-grenoble-alpes.fr/index.php/s/5zjJaf5BStr4zw2`. The file extracted from the archive is to be stored in the directory `data` of the provided source code.

4 Work on the dataset

We are going to use Spark to analyze the data included in this dataset. Instructions on how to use Spark are provided in Appendix A.

4.1 Programming language

For this lab, you can either use Scala or Python as programming language. Using Scala has the advantage that in general performance will be better since Spark executes inside a JVM.

Note that to use Scala, you will have to use the machines from the lab room (or your own laptop with Linux). The installation based on Docker does not have support to compile Scala code.

4.2 Provided code

The provided project contains an initial Spark code in Python (file `code/navigation.py`) and in Scala (file `code/Navigation/src/main/scala/navigation.scala`). This code already works and does the following

- It creates a RDD out of the input dataset
- It displays 5 of the nationalities that produced reports in the dataset (column "Nationality")
- Note that the provided code assumes that the dataset has been stored in the directory `code/data`.

Start by running this code and try to understand how it works.

A complete documentation of the Spark API for manipulating RDDs is available online:

- For Scala: `https://spark.apache.org/docs/latest/api/scala/org/apache/spark/rdd/RDD.html`
- For Python: `https://spark.apache.org/docs/latest/api/python/reference/pyspark.html#rdd-apis`

4.3 A few comments

Before starting doing the exercises, here are a few comments to have in mind while you are programming with Spark:

- While you are debugging a program, it can be good to run with a single executor thread (`local[1]`) to avoid very large and messy log traces in case of error.
- Once your program works, you can try executing it with more executor threads to observe the impact on performance. (Elapsed time can be measured in the way described here in Python: `https://stackoverflow.com/a/25823885`, and here in Scala: `https://stackoverflow.com/a/37731494`)
- Some of the questions below can be solved in different ways. Do not hesitate to implement multiple solutions and compare their performance.
- Caching RDDs can have a significant impact on the performance of Spark (see `https://spark.apache.org/docs/latest/rdd-programming-guide.html#rdd-operations`). You can also try to evaluate this impact during your tests.
- Note that in the dataset, an entry with no value is represented by the string "NA" (stands for *Not Available*)
- You can access the Spark Web UI by opening the Url `http://localhost:4040/` (or `http://127.0.0.1:4040/`). Take the time during the lab to observe the information accessible through this interface (Graph of tasks, execution time, etc.)
  - The Web UI is only accessible when a Spark application is running. To allow you to simply connect to the interface, the provided example of Spark code prevents the application from terminating immediately after the computation are done: you are required to press `Enter` to terminate the program.

4.4 Exercises

Implement a Spark program that does the following:

1. When running the provided example code, you will observe that there might be several entries that are equivalent. More specifically, you will see two entries for "British" with the only difference that one has an extra white space in the name. Propose a new version of the computation that will consider these two entries as the same one.
2. Count the total number of observations included in the dataset (each line corresponds to one observation)
3. Count the number of years over which observations have been made (Column "Year" should be used)
4. Display the oldest and the newest year of observation

5. Display the years with the minimum and the maximum number of observations (and the corresponding number of observations)
6. Count the distinct departure places (column "VoyageFrom") using two methods (i.e., using the function *distinct()* or *reduceByKey()*) and compare the execution time.
7. Display the 10 most popular departure places
8. Display the 10 roads (defined by a pair "VoyageFrom" and "VoyageTo") the most often taken.
    - Here you can start by implementing a version where a pair "VoyageFrom"-"VoyageTo" A-B and a pair B-A correspond to different roads.
    - Implement then a second version where A-B and B-A are considered as the same road.
9. Compute the hottest month (defined by column "Month") on average over the years considering all temperatures (column "ProbTair") reported in the dataset.

As already mentioned, do not hesitate to run other analysis on the dataset to practice with Spark.

Also, observe what is happening through the Spark Web UI, as described above.

4.5 To go further: Dataframes

Dataframes have been introduced in Spark to make it simpler to manipulate structured data organized into named columns. This is the case for the data that are manipulated during this lab.

If you are curious about Dataframes, you can try solving the questions of this lab using Dataframes.

Here is a starting point for a description of Dataframes: `https://spark.apache.org/docs/latest/sql-programming-guide.html#datasets-and-dataframes`

A Using Apache Spark

Note that the installation based on Docker only allows running Spark applications coded in Python. If you want to use Scala, you have to follow the instructions for a native install on Linux.

A.1 Using the machines from the lab rooms (or a native installation on your machine)

A.1.1 Using Spark in Python

1. Assuming that you have a script called `navigation.py`, you can run it by simply executing in a terminal:

   ```
   python3 ./navigation.py
   ```

A.1.2 Using Spark in Scala

To use Spark in Scala

1. Insert your code in the file `Navigation/src/main/scala/navigation.scala` in the provided project tree.
2. Compile your project by running the command `sbt package` in the directory `Navigation`
3. To run your program, use the following command:

   ```
   spark-submit target/scala-2.12/navigation_2.12-1.0.jar
   ```

A.2 Using Spark installed with Docker

A.2.1 Using Spark in Python

1. Start the docker container by running in a terminal:

   ```
   docker run -v absolute_path_to_folder:/home/jovyan/work -it \
          --rm -p 8888:8888 -p 4040:4040 jupyter/pyspark-notebook
   ```

   - In the previous command, `absolute_path_to_folder` should be replaced by the actual path to the directory where you are going to store your Python scripts
   - **When you launch this command, logs are written into the terminal. The last displayed line is an `url` that you should copy into a web browser. This `url` allows you to connect to a Jupyter Notebook that gives access to Spark.**

2. In Jupyter Notebook, open a new terminal (New > Terminal)

3. In the new terminal, move into the directory storing your Python scripts (for this simply run: `cd work`)

4. Assuming that you have a script called `navigation.py`, you can run it by simply executing in the Notebook terminal:

   ```
   python3 ./navigation.py
   ```

A.2.2 A few additional comments

- The instructions provided above explain how to run the provided Python script. Of course, if you prefer you can create a new Notebook in the Jupyter instance started in the docker container.
- If you have problems running the docker command, you are also provided with a `docker-compose` file to automatize the start of the container (file `docker-compose.yml`). To use it, you should:
  - Install `docker-compose` (see `https://docs.docker.com/compose/install/`)
  - Run "`docker-compose up`" in the directory where `docker-compose.yml` is stored.

  Thanks to `@alvarogonjim` for proposing this solution.

A.3 Using Spark with Google Colab

A last solution is to do the lab with Google Colab. Please refer to this short introduction to start using Spark with Google Colab: `https://colab.research.google.com/drive/1-co8gEHx_EJLURFWfw0WZq1uik0uRfqC?usp=sharing`.