# Analyzing and debugging the performance of a Spark application

Master M2 – Université Grenoble Alpes & Grenoble INP

2018–2019

This assignment is about analyzing and improving the performance of a Spark application. The most important aspect of this assignment is the rigor of the performance analysis you are conducting.

We have discussed in class how to plan and conduct experiments for performance evaluation. You will apply the presented methods to do this practical work using a Spark application you developed or have been given.

## 1 Important information

- During the semester, you have to submit one lab that is going to be used for your *lab grade* in this course. You can choose to submit this lab. Only one of the two proposed labs should be submitted.
- The assignment is to be done by groups of at most **2** students.
- The assignment must be implemented using Apache Spark. You can choose to program in Scala or in Python

### 1.1 Collaboration and plagiarism

You are encouraged to discuss ideas and problems related to this project with the other students. You can also look for additional resources on the Internet. However, we consider plagiarism very seriously. Hence, if any part of your final submission reflects influences from external sources, you must cite these external sources in your report. Also, any part of your design, your implementation, and your report should come from you and not from other students/sources. We will run tests to detect similarities between projects. In case of plagiarism, your submission will not be graded and appropriate actions will be taken.

## 2 Description

You must study and try to improve the performance of a Spark application. All your attempts and results must be described in a lab book that will be graded.

## 2.1 Choice of a Spark application

You can choose one of these three applications:

- your solution to the "Introduction to Spark" lab session, which used the *Climatological Database for the World's Oceans* dataset;
- the code you wrote for "Analyzing Data with Spark" lab, using the traces from Google; or
- the one we provide to you, described in Section 3.

The last one is written in Scala, for the other two it depends on the choice you made when you wrote it, either Python or Scala.

## 2.2 Lab book contents

The lab book will contain everything involved in your performance evaluation, including:

- the Spark code you will use for your experiments;
- any code (for example scripts) you will develop to conduct the experiments;
- modifications of the original code you do to improve performance;
- the performance results of your experiments and how you analyze them.

Don't add the input data given to the applications in the experiments, but instead a detailed description of where it comes from and how it was obtained.

## 2.3 Experiments to run

As you have seen during this course you have two main approaches to improve performance:

- modify the code: to change memory management (use persist or cache methods for example) or how data are processed (narrow and wide operations for instance);
- change execution contexts: number of cores used, amount of memory for the JVM, number of Spark workers, etc.

This is not an exhaustive list. You can, and should, try other approaches.

## 2.4 Your submission

Your submission must be **one** PDF file for your lab book and all its information. Its name must be the last name of the two students involved in the project: `Name1_Name2_labbook.pdf`.

**Grading:** The following criteria will be taken into account for grading your work:

- the quality of the lab book produced;
- the rigor of the analysis conducted;
- the number of experiments;
- the performance analysis done.

## 3 The ParseTrace application

This application was used by researchers in the field of High-Performance Computing to characterize the accesses to the parallel file system of the Santos Dumont supercomputer, located in Brazil. Processing nodes generate one trace file per day, and each trace file has multiple measurements of I/O activity obtained during the day and identified by a timestamp.

The application receives as input a collection of trace files belonging to the same day and aggregate their information. It also uses information obtained from the resource manager of the machine about jobs that were executed in that day. The output is global I/O bandwidth and number of nodes and applications accessing the system at each timestamp.

Download the file `http://francielizanon.github.io/teaching/TP/sdumont.tgz`. The package includes:

- the *slurm-info/* folder that contains one file per day regarding the execution of jobs;
- the *file-system/* folder with the traces, organized per day;
- the *ParseTrace/* folder with the source code, including a *build.sbt* file that can be used with sbt to generate the .jar.

When running the application with spark-submit, you need to pass a folder containing trace files as the first argument[1] and one of the files of *slurm-info/* as the second argument. It is important to use traces and a slurm file that are about the same day.

Despite the fact the input dataset is not very large, this application may take hours to aggregate traces for a single day. We suggest you work with only a subset of the files at first.

[1] The application will read **all** traces in the folder.