

Analysis of COVID-19 Data using Apache Spark

Design Document for the Apache Spark project of the course "Middleware Technologies for Distributed Systems".

Authors:

Accordi Gianmarco

Buratti Roberto

Motta Dennis



POLITECNICO
MILANO 1863

Dipartimento di Elettronica, Informazione e Bioingegneria

Politecnico di Milano

Italy

24/02/2021

Contents

1	Introduction	2
2	Project Analysis and Assumption	2
2.1	Queries	2
2.2	Datasets	2
2.3	Assumptions	2
3	Design	2
3.1	ECDC dataset steps	3
3.2	Simulation dataset steps	3
4	Implementation	4
4.1	Spark cache	4
5	Testing	5

1 Introduction

The purpose of this report is to explain the work done by this group on the project "Analysis of COVID-19 Data using Apache Spark" for the "Middleware Technologies for Distributed Systems" course of "Politecnico di Milano". The analysis of the task to be solved has been analyzed and taken from the projects' pdf provided by the professors [2].

2 Project Analysis and Assumption

The **goal of this project** is to use Apache Spark to analyze COVID-19 datasets to study the evolution of the situation worldwide.

2.1 Queries

In practice 3 queries must be computed:

- Seven days moving average of new reported cases, for each country and for each day.
- Percentage increase (with respect to the day before) of the seven days moving average, for each country and for each day.
- Top 10 countries with the highest percentage increase of the seven days moving average, for each day.

2.2 Datasets

We implemented a structure that allows to easily adapt to any dataset provided with very few changes to the code. Thanks to this we easily implemented the analysis on two datasets:

- The ECDC dataset[1].
- The simulation dataset computed from the "Simple Model for Virus Spreading" that we developed (this model will be referred later as "project 4").

2.3 Assumptions

In the ECDC dataset that provides weekly reports we assumed that the weekly increment is evenly spread across the days of the week.

3 Design

We decided to apply a simple pipeline design since each query requires the result of the previous one. The design is shown in Figure 1.

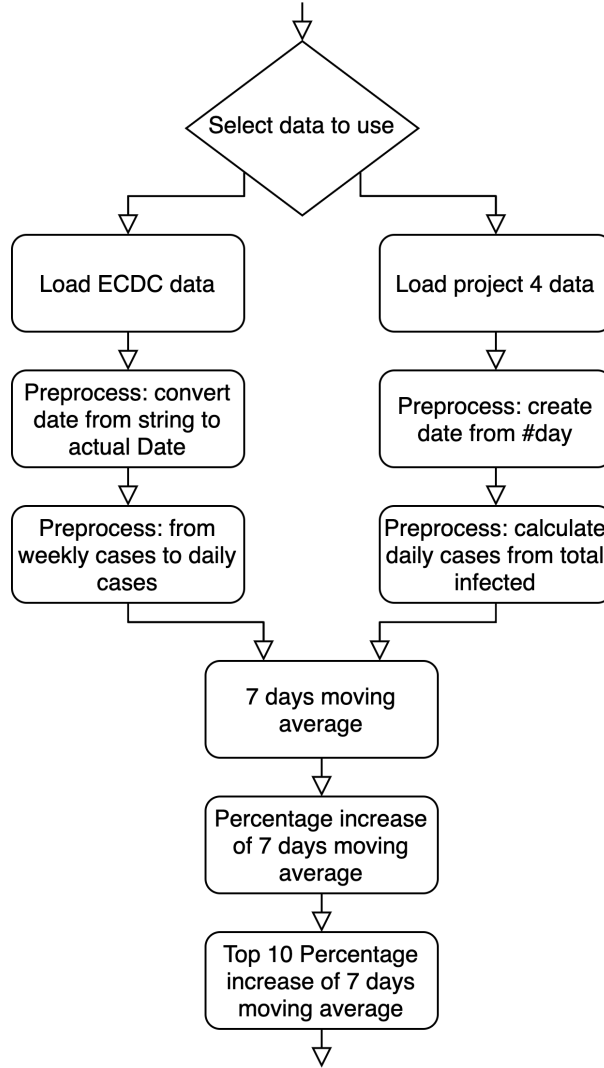


Figure 1: The steps for the analysis.

3.1 ECDC dataset steps

In details these are the steps needed to perform the analysis on the ECDC dataset:

1. **Load ECDC data:** the single csv file provided by ECDC is loaded into Spark.
2. **Preprocess: convert date from string to actual Date:** the format used in the csv for the date (DD/MM/YYYY) is converted to the standard Spark DateType.
3. **Preprocess: from weekly cases to daily cases:** the ECDC provides only weekly reports, so we assumed that the weekly increment is evenly spread across the days of the week. In practice we created a new column in the dataset where for each report of the ECDC data we created 7 rows, one for each day of the week, and then with a division we computed the evenly spread daily cases.

3.2 Simulation dataset steps

In details these are the steps needed to perform the analysis on the simulation dataset created with the "Simple Model for Virus Spreading" that we developed:

1. **Load project 4 data:** the multiple csv files provided by the simulation are loaded into Spark. The simulation creates multiple files since the computation is parallelized across multiple processors.

2. **Preprocess: crate date from #day:** the simulation doesn't use a concept of date, but only keeps track of the number of days passed since starting. As a convention we decided to assume in the analysis 31/12/2019 as the first day of the simulation (31 December 2019 is the date of the first public message about a possible virus outbreak)[3].
3. **Preprocess: calculate daily cases from total infected:** the simulation provides for each day the total number of people infected in each area. To extract the daily cases from this information we just compute the difference with respect to the day before.

4 Implementation

To allow adaptability and maintainability we coupled the concept of various abstract actions (load, preprocess, apply query) with abstract Java classes. The details of this procedure can be seen in the UML diagram in Figure 2.

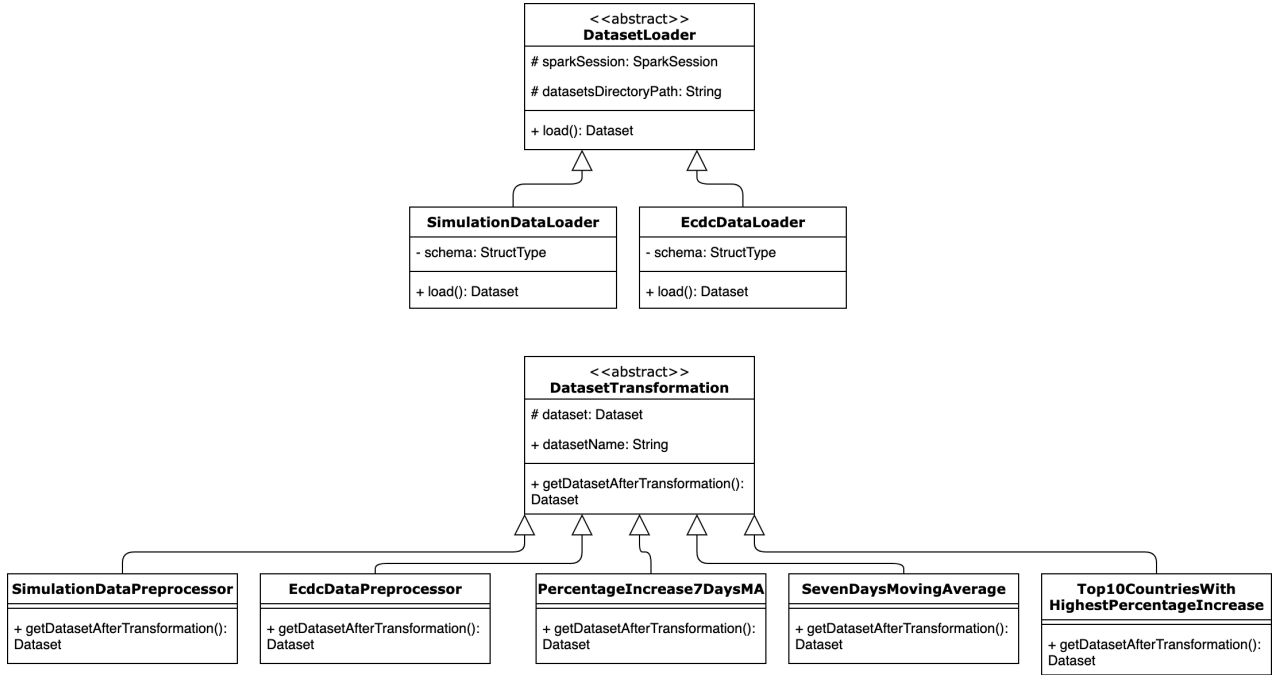


Figure 2: The most important classes in the implementation.

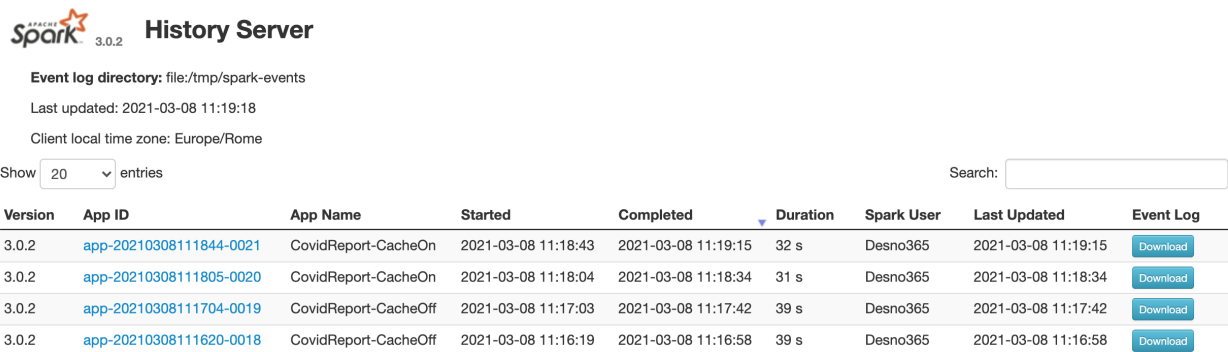
Then the class that handles all the procedure of the analysis is called CovidReport, and it implements the computation flow seen in Section 3, Figure 1.

4.1 Spark cache

The CovidReport class uses the cache feature of Spark to avoid the re-computation of previous transformations, resulting in a small speedup in the analysis. The cache can be turned on/off with an argument passed to the program.

In details the effects experienced on our test machine by enabling or disabling the cache are the following:

- Reduction of the execution time by 20% as can be seen in Figure 3.
- Big increase in the number of stages and tasks skipped as can be seen in Figure 4.



Apache Spark 3.0.2 History Server

Event log directory: file:/tmp/spark-events
 Last updated: 2021-03-08 11:19:18
 Client local time zone: Europe/Rome

Show 20 entries Search:

Version	App ID	App Name	Started	Completed	Duration	Spark User	Last Updated	Event Log
3.0.2	app-20210308111844-0021	CovidReport-CacheOn	2021-03-08 11:18:43	2021-03-08 11:19:15	32 s	Desno365	2021-03-08 11:19:15	Download
3.0.2	app-20210308111805-0020	CovidReport-CacheOn	2021-03-08 11:18:04	2021-03-08 11:18:34	31 s	Desno365	2021-03-08 11:18:34	Download
3.0.2	app-20210308111704-0019	CovidReport-CacheOff	2021-03-08 11:17:03	2021-03-08 11:17:42	39 s	Desno365	2021-03-08 11:17:42	Download
3.0.2	app-20210308111620-0018	CovidReport-CacheOff	2021-03-08 11:16:19	2021-03-08 11:16:58	39 s	Desno365	2021-03-08 11:16:58	Download

Figure 3: Execution times when running the analysis on both datasets on our test machine with cache enabled and with the "showResultsInTerminal" feature enabled.

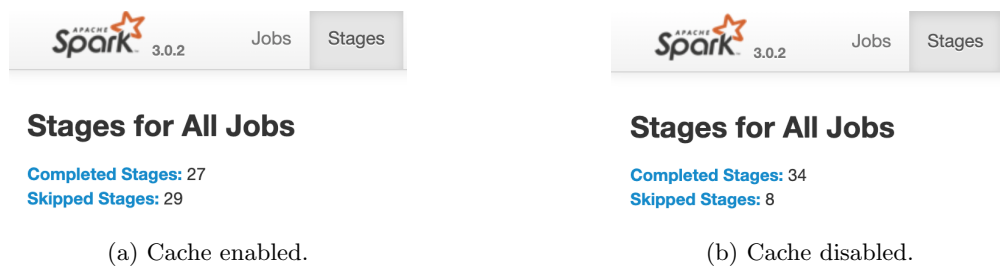


Figure 4: Comparison on the number of stages skipped when enabling or disabling the cache feature. When cache is enabled many more stages are skipped.

5 Testing

We created an automatic test using JUnit to check the correctness of the queries we wrote. We artificially created two small datasets, one with a similar structure to the one of ECDC and the other one with a similar structure to the one of the simulation. We then created the expected output for these two small datasets, so that when running the test it compares the expected output to the output obtained.

References

- [1] European Centre for Disease Prevention and Control. Historical data on the number of new reported covid-19 cases and deaths worldwide.
- [2] Luca Mottola and Alessandro Margara. Middleware technologies for distributed systems - exam projects 2020/2021, 2021.
- [3] Wikipedia. Timeline of the covid-19 pandemic in 2019.