UNIVERSIDAD COMPLUTENSE MADRID

HWB. Spark Programming

Ignacio M. Llorente, v1.1 - 18 November 2020

Abstract

The objective in this homework is to develop practical skills in parallel data processing for computational and data science. The focus is on scaling data-intensive computations using functional parallel programming over distributed architectures. In these exercises we will practice Spark programming with special emphasis on data parallel processing.

Guidelines

- The **datasets** needed to do the exercises are available for download from Google Classroom.
- **AWS**
  - **First you should have followed the Guide "First Access to AWS"**. It is assumed you already have an AWS account and a key pair, and you are familiar with the AWS EC2 environment.

If you are using AWS, we strongly recommend you use the same instance type for all the experiments (**m4.xlarge**) so you can compare the performance results achieved with the different programming models and platforms

- **Spark**
  - Programs should be developed in Python.
  - Install a local version of Spark, by following the Guide "Install Spark Cluster in Local Mode", to develop your programs (exercises 1-5) on a single AWS VM (or your local computer). As Spark's local mode is fully compatible with the cluster mode; programs written and tested locally can be run on a cluster with just a few additional steps.
- **Submission**

Any computing experiment that cannot be replicated cannot be considered as a valid submission

- Upload on **Google Classroom** the files specified in each assignment.
- The grade on this assignment is **10% (100 points) of the final grade.**
- There are **no late days.**

Table of Contents

1. Distributed Grep (20 points)

Develop a Spark version of the grep tool to search words in very large documents. The output should be the lines that contain a given *word*.

You should use as an input file the input text used in the word count example described in class (eBook of Moby Dick). Your script will be tested using the following linux command.

```
$ spark-submit P1_spark.py word
```

Submission

- `P1_spark.py`: Spark script

2. Count URL Access Frequency (20 points)

Develop a Spark script to find the frequency of each URL in a web server log. The output should be the URLs and their frequency.

You should use as input file the sample Apache log file `access_log` (downloaded from http://www.monitorware.com/en/logsamples/apache.php). Your script will be tested using the following linux command.

```
$ spark-submit P2_spark.py
```

Submission

- `P2_spark.py`: Spark script

3. Stock Summary (20 points)

Write a Spark script to calculate the average daily stock price at close of Alphabet Inc. (GOOG) per year since 2009. The output should be the year and the average price.

You should use as input file the daily historical data `GOOGLE.csv` (downloaded from Yahoo Finance https://finance.yahoo.com/quote/GOOG/history?ltr=1). You have to preprocess the `GOOGLE.csv` file to remove the head line. Your script will be tested using the following linux command.

```
$ spark-submit P3_spark.py
```

You may need to use **DataFrames** to simplify the processing of the data. A DataFrame is a Dataset organized into named columns. It is conceptually equivalent to a table in a relational database or a data frame in R/Python, but with richer optimizations under the hood.

Submission

- `P3_spark.py`: Spark script

4. Movie Rating Data (20 points)

GroupLens Research has collected and made available rating data sets from the MovieLens web site (http://movielens.org). The data sets were collected over various periods of time, depending on the size of the set. Before using these data sets, please review their README files for the usage licenses and other details. You can use as input file the small version of the dataset `ml-latest-small.zip` (downloaded from https://grouplens.org/datasets/movielens/).

Develop a Spark version of a job to show movies with an average rating in the ranges:

Range 1: 1 or lower
Range 2: 2 or lower (but higher than 1)
Range 3: 3 or lower (but higher than 2)
Range 4: 4 or lower (but higher than 3)
Range 5: 5 or lower (but higher than 4)

Your script will be tested using the following linux command.

```
$ spark-submit P4_spark.py
```

You may need to use **DataFrames** to simplify the processing of the data.

Submission

- `P4_spark.py`: Spark script

5. Meteorite Landing (20 points)

The NASA's Open Data Portal hosts a comprehensive data set from The Meteoritical Society that contains information on all of the known meteorite landings. The Table `Meteorite_Landings.csv` (downloaded from https://data.nasa.gov/Space-Science/Meteorite-Landings/gh4g-9sfh) consists of 34,513 meteorites and includes fields like the type of meteorite, the mass and the year. Develop a Spark version of the job to calculate the average mass per type of meteorite.

You have to preprocess the `Meteorite_Landings.csv` file to remove the head line and do some data cleansing work. Your script will be tested using the following linux command.

```
$ spark-submit P5_spark.py
```

You may need to use **DataFrames** to simplify the processing of the data.

Submission

- `P5.py`: Spark script