UNIVERSIDAD COMPLUTENSE MADRID

Assignment 2. Spark

Abstract

The objective in this assignment 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** to do the exercises are available for download from Google Classroom.
- Use either a VM in Google Cloud or a local computer with Spark in local mode (as in Lab 3).
- Spark programs should be written in Python.
- Upload the files specified on each exercise to the assignment on **Google Classroom**.
- Submit each file individually (**DO NOT submit a compressed file**).
- There are **no late days.**
- The grade on this assignment is **10% of the final grade.**

1. Distributed Grep (2 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*.

Use as input the text used in the word count example described in class (eBook of Moby Dick). Your script will be tested using the following command:

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

2. Count URL Access Frequency (2 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.

Use as input 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 command:

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

3. Stock Summary (2 points)

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

Use as input the daily historical data `GOOGLE.csv` (downloaded from Yahoo Finance https://finance.yahoo.com/quote/GOOG/history?ltr=1). Preprocess the `GOOGLE.csv` file to remove the header. Your script will be tested using the following command:

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

DataFrames may be used to simplify reading and processing data.

4. Movie Rating Data (2 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. A small version of the dataset, `ml-latest-small.zip` (downloaded from https://grouplens.org/datasets/movielens/), can be used.

Write a Spark script 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 command:

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

DataFrames may be used to simplify reading and processing data.

5. Meteorite Landing (2 points)

NASA's Open Data Portal hosts a comprehensive data set from The Meteoritical Society that contains information on all of the known meteorite landings. 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 script to calculate the average mass per type of meteorite.

Preprocess the `Meteorite_Landings.csv` file to remove the header and do some data cleansing work. Your script will be tested using the following command:

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

DataFrames may be used to simplify reading and processing data.

Submission

- `P1_spark.py`
- `P2_spark.py`
- `P3_spark.py`
- `P4_spark.py`
- `P5_spark.py`