Big data processing and analytics

June 20, 2024

Student ID ____

First Name ____

Last Name ____

The exam is **open book**

Part I

Answer the following questions. There is only one right answer for each question.

1. (2 points) Consider the following Spark Streaming application.

```
import …
public class SparkDriver {
    public static void main(String[] args) throws InterruptedException {
            SparkConf conf = new SparkConf().setAppName("Spark Streaming - Question");
            JavaStreamingContext jssc = new JavaStreamingContext(conf, Durations.seconds(10));
            // Define a DStream associated with the TPC socket localhost:9999
            JavaDStream<String> inputDStream = jssc.socketTextStream("localhost", 9999);

            // Part A
            JavaDStream<Integer> resADStream = inputDStream
                            .map(value -> Integer.valueOf(value))
                            .filter(value -> value<15)
                            .window(Durations.seconds(30), Durations.seconds(10))
                            .reduce((v1,v2) -> Math.max(v1,v2))
                            .filter(value -> value>10);
            // Print the result on standard output
            resADStream.print();

            // Part B
            JavaDStream<Integer> resBDStream = inputDStream
                            .map(value -> Integer.valueOf(value))
                            .reduce((v1,v2) -> Math.max(v1,v2))
                            .filter(value -> value<15)
                            .window(Durations.seconds(30), Durations.seconds(10))
                            .reduce ((v1,v2) -> Math.max(v1,v2))
                            .filter(value -> value>10);
            // Print the result on standard output
            resBDStream.print();

            // Part C
            JavaDStream<Integer> resCDStream = inputDStream
                            .map(value -> Integer.valueOf(value))
                            .filter(value -> value<15)
                            .reduce((v1,v2) -> Math.max(v1,v2))
```

```
                                        .window(Durations.seconds(30), Durations.seconds(10))
                                        .reduce ((v1,v2) -> Math.max(v1,v2))
                                        .filter(value -> value>10);

                    // Print the result on standard output
                    resCDStream.print();

                    // Start the computation
                    jssc.start();
                    jssc.awaitTerminationOrTimeout(120000);
                    jssc.close();
          }
     }
```

Which one of the following statements is true?

a)

a) Independently of the content of **inputDStream**, **resADStream** and **resBDStream** contain always the same integer values, while **resCDStream** may contain different integer values with respect to **resADStream** and **resBDStream**.

b) Independently of the content of **inputDStream**, **resADStream** and **resCDStream** contain always the same integer values, while **resBDStream** may contain different integer values with respect to **resADStream** and **resCDStream**.

c) Independently of the content of **inputDStream**, **resBDStream** and **resCDStream** contain always the same integer values, while **resADStream** may contain different integer values with respect to **resBDStream** and **resCDStream**.

d) Independently of the content of **inputDStream**, **resADStream**, **resBDStream**, and **resCDStream** contain always the same integer values.

2. (2 points) Consider the input HDFS folder myFolder that contains the following three files:

- ProfilesItaly.txt
  - The text file ProfilesItaly.txt contains the following four lines:
    Luca,Rome
    Luca,Rome
    Carmen,Naples
    Luca,Turin
- ProfilesFrance.txt
  - the text file ProfilesFrance.txt contains the following line:
    Claire,Paris
- ProfilesSpain.txt
  - the text file ProfilesSpain.txt contains the following two lines:
    Carmen,Barcelona
    Laura,Barcelona

Suppose that you are using a Hadoop cluster that can potentially run up to 3 instances of the mapper class in parallel. Suppose the HDFS block size is 1024MB. Suppose to execute a MapReduce application for Hadoop that analyzes the content of myFolder. Suppose the map phase emits, overall, the following key-value pairs (the key part is a name while the value part is always 1):

(Luca, 1)
(Luca, 1)
(Carmen, 1)
(Luca,1)
(Claire,1)
(Carmen,1)
(Laura,1)

Suppose the number of instances of the reducer class is set to 4 and suppose the reduce method of the reducer class sums the values associated with each key and emits one pair (name, sum values) for each key. Specifically, suppose the following pair is emitted, overall, by the reduce phase:

(Luca, 3)
(Carmen, 2)
(Claire, 1)
(Laura, 1)

Suppose to run the above application once. Which of the following statements is **true**?

a) Among the 4 instances of the reducer class:
- The reduce method of the first instance of the reducer class is invoked 2 times.
- The reduce method of the second instance of the reducer class is invoked 2 times.
- The reduce method of the third instance of the reducer class is never invoked.
- The reduce method of the fourth instance of the reducer class is never invoked.

b) Among the 4 instances of the reducer class:
- The reduce method of the first instance of the reducer class is invoked 3 times.
- The reduce method of the second instance of the reducer class is invoked 2 times.
- The reduce method of the third instance of the reducer class is invoked 1 time.
- The reduce method of the fourth instance of the reducer class is invoked 1 time.

c) Among the 4 instances of the reducer class:
- The reduce method of the first instance of the reducer class is invoked 4 times.
- The reduce method of the second instance of the reducer class is invoked 3 times.
- The reduce method of the third instance of the reducer class is never invoked.
- The reduce method of the fourth instance of the reducer class is never invoked.

d) Among the 4 instances of the reducer class:
- The reduce method of the first instance of the reducer class is invoked 7 times.
- The reduce method of the second instance of the reducer class is never invoked.

- The reduce method of the third instance of the reducer class is never invoked.
- The reduce method of the fourth instance of the reducer class is never invoked.

Part II

The managers of PoliJob, an international job portal company, asked you to develop two applications:

- a MapReduce program (Exercise 1)
- a Spark-based program (Exercise 2)

to address the analyses they are interested in. The applications can read data from three input files: JobPostings.txt, JobOffers.txt, and JobContracts.txt.

JobPostings.txt

- It is a textual file containing information about the job postings, i.e., positions for jobs that companies are looking for. There is one line for each job posting. The JobID uniquely identifies the job postings. Multiple job postings with the same position (title) can be present. This file is extremely large and you cannot suppose its content can be stored in one in-memory Java variable.
- Each line has the following format.
    - *JobID,Title,Country*
- *Example data.*
    - *1,Software Engineer,IT*
    - *2,Data Scientist,FR*
    - *3,Software Engineer,ES*
    - *4,Software Engineer,IT*
    - *5,Data Scientist,IT*
- Example: The Job 1 (JobID) offers the position (title) of Software Engineer in Italy (IT).

JobOffers.txt

- It is a textual file containing information about the job offers, i.e., job offers that were proposed to candidates applying for a Job posting (JobID). There is one line for each offer. OfferID uniquely identifies the offers. The status indicates whether the offer has been accepted or not. If an offer is accepted, then you might have a job contract (i.e., there can be from 0 to 1 job contract for each offer in JobContracts.txt). This file is extremely large and you cannot suppose its content can be stored in one in-memory Java variable.
- Each line has the following format.
    - *OfferID,JobID,Salary,Status,SSN*
- Example data.

- 101,1,97000,Accepted,800-11-2222
- 102,2,120000,Rejected,800-11-2223
- 103,3,120000,Accepted,800-12-2222
- 104,5,120000,Rejected,801-21-3222
- 105,5,120000,Rejected,800-41-2232
- 106,4,120000,Rejected,800-14-2422
- 107,3,120000,Rejected,800-17-2252
- 114,5,120000,Accepted,800-51-2222
- 115,1,120000,Accepted,800-51-2622
- 116,5,120000,Accepted,800-14-2262
- 117,5,120000,Accepted,800-14-2262

- Example: The offer 101 has been proposed to a candidate for job 1 (JobID) with a salary of 97 thousand euros, and the candidate accepted the offer. 800-11-2222 (SSN) is the social security number of the candidate to whom the offer has been made.

JobContracts.txt

- It is a textual file containing information about the job contracts signed by candidates and companies, i.e., after an offer has been accepted, a contract is typically signed (there are from 0 to 1 contract for each accepted offer). There is one line for each job contract, and ContractID uniquely identifies the contracts. This file is extremely large and you cannot suppose its content can be stored in one in-memory Java variable.
- Each line has the following format.
  - *<u>ContractID</u>,OfferID,ContractDate,ContractType*
- Example data.
  - *201,101,2023-01-15,Full-time*
  - *202,103,2023-02-01,Part-time*
  - *203,114,2023-03-10,Full-time*
  - *204,115,2023-04-20,Internship*
  - *205,116,2023-05-05,Full-time*
  - *206,117,2023-06-15,Contractor*
- Example: The contract 201 for the offer 101 has been signed on Jan 15, 2023, with type “Full-time”.

Exercise 1 – MapReduce and Hadoop (8 points)

Exercise 1.1

The managers of PoliJobs are interested in performing some analyses about jobs.

Design a single application based on MapReduce and Hadoop and write the corresponding Java code to address the following point:

1. *Jobs with no accepted offers and more than 10 high-salary rejected offers*. The application selects the job IDs of the job postings having no accepted offers and more than 10 rejected offers whose salary is higher than 100 thousand euros. Save the selected job IDs in the output HDFS folder, and the corresponding number of high-salary rejected offers (one Job ID per output line).
   Note: the list of offers by a specific job ID cannot be stored in a Java variable.

Suppose that the input folder contains JobOffers.txt has already been set. Suppose that the name of the output folder has also already been set.

- Write only the content of the Mapper and Reducer classes (map and reduce methods, setup and cleanup if needed). The content of the Driver must not be reported.
- Use the following two specific multiple-choice questions to specify the number of instances of the reducer class for each job.
- If your application is based on two jobs, specify which methods are associated with the first job and which are associated with the second job.
- If you need personalized classes, report for each of them:
  - the name of the class,
  - attributes/fields of the class (data type and name),
  - personalized methods (if any), e.g., the content of the toString() method if you override it,
  - do not report the get and set methods. Suppose they are "automatically defined".

Answer the following two questions to specify the number of jobs (one, two or three) and the number of instances of the reducer classes.

Exercise 1.2 - Number of instances of the reducer - Job 1

Select the number of instances of the reducer class of the first Job

(a) 0

(b) exactly 1

(c) any number >=1 (i.e., the reduce phase can be parallelized)

Exercise 1.3 - Number of instances of the reducer - Job 2

Select the number of instances of the reducer class of the second Job

(a) One single job is needed

(b) 0

(c) exactly 1

(d) any number >=1 (i.e., the reduce phase can be parallelized)

Exercise 2 – Spark and RDDs (19 points)

The managers of PoliJob asked you to develop a single Spark-based application based on RDDs or Spark SQL to address the following tasks. The application takes the paths of the three input files, and two output folders (associated with the outputs of the following points 1 and 2, respectively).

1. *Top 3 countries with the highest average salary*. For each country, compute the average salary, considering only accepted job offers. Then, select the top 3 countries with the highest value of average salary. The first HDFS output file must contain the identifiers of the selected countries (one country per output line) and their average salary (computed considering only accepted job offers).
   Note: Suppose there is at least an accepted job offer for each country.

2. Select the most popular job title in each country among job postings that resulted in job contracts (i.e., the job title with the highest number of job contracts in a given country). Store in the output folder the country names, their most common job title(s), and the contract count for that job title(s).

   Note: For each country, there might be many job titles associated with the highest number of job contracts for that country. Store all of them in the output folder (one of them per output line together with country and the contract count).

   Note: Suppose there is at least a contract for each country.

- You do not need to write Java imports. Focus on the content of the main method.
- Suppose both **JavaSparkContext sc** and **SparkSession ss** have already been set.
- **Only if you use Spark SQL**, suppose the first line of all files contains the header information/the name of the attributes. Suppose, instead, there are no header lines if you use RDDs.
- If you need personalized classes, report for each of them:
  - the name of the class,
  - attributes/fields of the class (data type and name),
  - personalized methods (if any), e.g., the content of the toString() method if you override it,
  - do not report the get and set methods. Suppose they are "automatically defined".