Big data processing and analytics

February 15, 2023

Student ID _____

First Name _____

Last Name _____

The exam is **open book**

Part I

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

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

```
package it.polito.bigdata.spark.exam;
import ….;
public class SparkDriver {

    public static void main(String[] args) {

        // Create a configuration object and set the name of the application
        SparkConf conf = new SparkConf().setAppName("Spark Code");

        // Create a Spark Context object
        JavaSparkContext sc = new JavaSparkContext(conf);

        JavaRDD<String> HumRDD = sc.textFile("Humidity.txt");

        JavaRDD<String> TempRDD = sc.textFile("Temperature.txt");

        // Computes the number of lines of Humidity.txt
        long numLinesHumidity = HumRDD.count();

        // Computes the number of lines of Temperature.txt
        long numLinesTemperature = TempRDD.count();

        // Print on the standard output the difference between the
        // number of lines of the two files
        System.out.println(numLinesHumidity - numLinesTemperature);

        // Create an RDD that contains the intersection of HumRDD and TempRDD
        JavaRDD<String> IntersectionRDD = HumRDD.intersection(TempRDD);

        // Print on the standard output the size of IntersectionRDD
        System.out.println(“Size intersection: ”+ IntersectionRDD.count());

        // Store the content of IntersectionRDD in the output folder
        IntersectionRDD.saveAsTextFile("outputFolder/");
```

```
        // Close the Spark context
        sc.close();
    }
}
```

Suppose the input files Humidity.txt and Temperature.txt are read from HDFS. Suppose this Spark application is executed only 1 time. Which one of the following statements is true?

a) This application reads the content of Humidity.txt 1 time and the content of Temperature.txt 1 time.

b) This application reads the content of Humidity.txt 2 times and the content of Temperature.txt 2 times.

c) This application reads the content of Humidity.txt 3 times and the content of Temperature.txt 3 times.

d) This application reads the content of Humidity.txt 4 times and the content of Temperature.txt 4 times.

2. (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))
                .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))
                .filter(value -> value<10)
                .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
        resBDStream.print();

        // Part C
        JavaDStream<Integer> resCDStream = inputDStream
                        .map(value -> Integer.valueOf(value))
                        .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) 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.

Part II

HouseWaterConsumption (HWC) is an international company that collects data about the water consumption of houses around the world. HWC computes a set of statistics about the monitored houses. The analyses are performed by considering the following input data sets/files.

- Houses.txt
  - The big file Houses.txt is a text file containing the list of houses monitored by HWC. Each line of Houses.txt is associated with one house and contains its profile. The number of monitored houses is more than 400,000,000.
  - Each line of Houses.txt has the following format
    - HID,City,Country,YearBuilt

      where *HID* is the house identifier, *City* and *Country* are the city and country in which the house is located, respectively, and *YearBuilt* is the year in which the house was constructed.

    - For example, the following line

      H3402,Turin,Italy,1965

      means that the house identified by ***H3402*** is located in the city of ***Turin*** (***Italy***) and it was built in the year 1965.

- MonthlyWaterConsumption.txt
  - The big file MonthlyWaterConsumption.txt contains the information about the monthly water consumption of the houses under analysis in the last 40 years. For each month of the last 40 years in which a house had water consumption greater than 0 a line was stored in MonthlyWaterConsumption.txt. The months with no consumption for a house are not stored in MonthlyWaterConsumption.txt.
  - Each line of MonthlyWaterConsumption.txt has the following format
    - HID,Month,M3

      where *M3* is the water consumption in cube meter of house *HID* during the month *Month*.

      Each line of MonthlyWaterConsumption.txt is uniquely identified by the primary key (HID,Month). Hence, each combination (HID,Month) occurs at most one time in MonthlyWaterConsumption.txt.

      The format of Month is YYYY/MM.

    - For example, the following line

      H3402,2022/12,10

      means that the water consumption of the house with HID ***H3402*** was ***10*** m^3 on ***December 2022***.

Exercise 1 – MapReduce and Hadoop (8 points)

Exercise 1.1

The managers of HWC are interested in performing some statistics.

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

1. *Countries with an average number of houses per city greater than 10000 houses.* Compute the average number of houses per city in each country and select the countries with an average number of houses per city greater than 10000 houses. The output HDFS folder contains one line for each selected country and the format is "country, average number of houses per city in that country".

   Toy example

   For the sake of simplicity, suppose that we have a small toy input file with only the following houses:

   - *H1,Turin,Italy,1965*
   - *H2,Turin,Italy,2020*
   - *H3,Turin,Italy,2004*
   - *H11,Milan,Italy,2019*
   - *H12,Milan,Italy,2019*
   - *H21,Barcelona,Spain,2004*
   - *H22,Barcelona,Spain,2014*
   - *H31,Madrid,Spain,2004*
   - *H41,Malaga,Spain,1965*
   - *H42,Malaga,Spain,1978*
   - *H43,Malaga,Spain,2001*

   Given this input, the output will be empty because the average number of houses per city in Italy is 2.5 and the average number of houses per city in Spain is 2.

   Note that in the actual input file there are millions of houses.

Suppose that the input is Houses.txt and has been already set. Suppose that also the name of the output folder has been already 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 (**Exercises 1.2 and 1.3)** 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"

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 (19 points)

The managers of HWC asked you to develop one single application to address all the analyses they are interested in. The application has four arguments: the input files Houses.txt and MonthlyWaterConsumption.txt, and two output folders "outPart1/" and "outPart2/", which are associated with the outputs of the following points 1 and 2, respectively.

Specifically, design a single application, based on Spark, and write the corresponding code, to address the following two points:

1. *Houses with increasing consumption in at least three trimesters of the year 2022 compared to the corresponding trimesters of the year 2021.* The first part of this application compares the water consumption of each house in each of the four trimesters of the year 2021 with the water consumption of each house in each of the four trimesters of the year 2022. A house is selected if there are at least 3 trimesters in the year 2022 with water consumption greater than the corresponding trimesters in the year 2021. Store the identifiers of the selected houses and the cities in which they are located in the first HDFS output folder (one HID, City per output line).

   Note that in MonthlyWaterConsumption.txt there is at least one line per trimester for each house considering all houses monitored by HWC and all trimesters of the last 40 years (i.e., all houses have a consumption greater than 0 for all the trimesters of the 40 years considered in this application).

   Example Part 1

   Suppose there are, among the others, the following two houses:

   - *H1,Turin,Italy,1965*
   - *H40,Malaga,Spain,1965*

And suppose these are the water consumption in the trimesters of the year 2021 and 2022 for these two example houses:

	I Trimester 2021 (1-3/2021)	II Trimester 2021 (4-6/2021)	III Trimester 2021 (7-9/2021)	IV Trimester 2021 (10-12/2021)
House H1	30 m3	30 m3	30 m3	30 m3

	I Trimester 2022 (1-3/2022)	II Trimester 2022 (4-6/2022)	III Trimester 2022 (7-9/2022)	IV Trimester 2022 (10-12/2022)
House H1	40 m3	35 m3	25 m3	32 m3

	I Trimester 2021 (1-3/2021)	II Trimester 2021 (4-6/2021)	III Trimester 2021 (7-9/2021)	IV Trimester 2021 (10-12/2021)
House H40	130 m3	120 m3	100 m3	115 m3

	I Trimester 2022 (1-3/2022)	II Trimester 2022 (4-6/2022)	III Trimester 2022 (7-9/2022)	IV Trimester 2022 (10-12/2022)
House H40	160 m3	118 m3	125 m3	110 m3

Given this example data

- H1 is selected, and “H1,Turin” is stored in the first output folder, because there are three trimesters of the year 2022 associated with a consumption greater than the consumption of the corresponding trimesters of the year 2021 for H1 (I, II, and IV trimesters).
- H40 is discarded because there are only two trimesters of the year 2022 associated with a consumption greater than the consumption of the corresponding trimesters of the year 2021 for H40 (I and III trimesters).

2. *Cities with a few houses for which there is at least one annual consumption decrease.* This second part of the application selects the cities for which at most 2 of their houses **(between 0 and 2)** are characterized by at least one annual water consumption decrease. A house is characterized by at least one annual water consumption decrease if there is at least one year in which its annual water consumption is lower than the annual consumption of the previous year. Store the selected cities in the second output folder (one city per output line).

   We remind that there is at least one line per trimester for each house considering all houses monitored by HWC and all trimesters of the last 40 years. It means that all houses have an annual consumption greater than 0 for all the 40 years considered in this application.

   Example Part 2

   For the sake of simplicity, suppose that we have only the following houses:

- *H1,Turin,Italy,1981*
- *H11,Turin,Italy,2019*
- *H12,Turin,Italy,2009*
- *H21,Barcelona,Spain,2004*
- *H22,Barcelona,Spain,2014*
- *H31,Barcelona,Spain,2004*
- *H41,Barcelona,Spain,1990*
- *H51,Nince,France,2001*

Suppose that, only for this toy example, there are only 5 years of data and the following are the annual consumption of these houses in each of those five years:

City	House	2018	2019	2020	2021	2022	At least one annual decrease
Turin	H1	300 m^3	200 m^3	300 m^3	350 m^3	360 m^3	Yes
	H11	300 m^3	300 m^3	250 m^3	260 m^3	160 m^3	Yes
	H12	300 m^3	180 m^3	170 m^3	150 m^3	300 m^3	Yes
Barcelona	H21	300 m^3	280 m^3	310 m^3	290 m^3	280 m^3	Yes
	H22	300 m^3	301 m^3	400 m^3	405 m^3	410 m^3	No
	H31	400 m^3	300 m^3	301 m^3	310 m^3	430 m^3	Yes
	H41	350 m^3	350 m^3	360 m^3	405 m^3	410 m^3	No
Nice	H51	300 m^3	400 m^3	400 m^3	410 m^3	450 m^3	No

Given this small example, the cities Barcelona (2 houses with at least one annual water consumption decrease) and Nice (0 houses with at least one annual water consumption decrease) are selected. Turin is discarded because it has more than two houses (3 houses) with at least one annual water consumption decrease.

Note that in the actual input file there are millions of houses.

- You do not need to report imports. Focus on the content of the main method.
- Suppose both JavaSparkContext ***sc*** and SparkSession ***ss*** have been already set.
- 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"