Big Data: Architectures and Data Analytics

February 21, 2022

Student ID _____

First Name _____

Last Name _____

The exam is **closed book** and lasts **2 hours**

Part I

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

Report your answers in the following boxes.

	Question 1	Question 2
Answer		

1. (2 points) Consider the input HDFS folder *myFolder* that contains the following two files:

   - NamesItaly.txt
     - the text file NamesItaly.txt contains the following three lines (size: 33 bytes)
       Luis,Turin
       Luca,Rome
       Paolo,Turin

   - NamesFrance.txt
     - the text file NamesFrance.txt contains the following two lines (size: 22 bytes)
       Paolo,Nice
       Luis,Paris

   Suppose that you are using a Hadoop cluster that can potentially run up to 4 instances of the mapper class in parallel. Suppose the HDFS block size is 128MB. Suppose to execute a MapReduce application for Hadoop that analyzes the content of myFolder. Suppose that among the (overall) 5 input lines/input pairs, the map phase selects only the lines starting with "P" and emits, overall, the following 2 key-value pairs (the key part is a name while the value part is always 1):

("Paolo", 1)
("Paolo", 1)

Suppose the number of instances of the reducer class is set to 3 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. Suppose the following pair is, overall, emitted by the reduce phase:

("Paolo", 2)

Considering all the instances of the mapper class, overall, how many times is the ***map method*** invoked?

a) 2
b) 3
c) 4
d) 5

2. (2 points) Consider the following Spark Streaming applications.

```
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
            // Map input strings to integers, define windows, compute the minimum value, apply a filter
            JavaDStream<Integer> resADStream = inputDStream
                            .map(value -> Integer.valueOf(value))
                            .window(Durations.seconds(30), Durations.seconds(10))
                            .reduce((v1,v2) -> Math.min(v1,v2))
                            .filter(value -> value>10);

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

            // Part B
            // Map input strings to integers, apply a filter, compute the minimum value, define windows,
            // and compute the minimum again
            JavaDStream<Integer> resBDStream = inputDStream
                            .map(value -> Integer.valueOf(value))
                            .filter(value -> value>10)
                            .reduce((v1,v2) -> Math.min(v1,v2))
                            .window(Durations.seconds(30), Durations.seconds(10))
                            .reduce((v1,v2) -> Math.min(v1,v2));

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

```
            // Part C
            // Define windows, map input strings to integers, apply a filter,
            // and compute the minimum value
            JavaDStream<Integer> resCDStream = inputDStream
                                .window(Durations.seconds(30), Durations.seconds(10))
                                .map(value -> Integer.valueOf(value))
                                .filter(value -> value>10)
                                .reduce((v1,v2) -> Math.min(v1,v2));

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

Part II

PoliOnline is an international company that sells items online. To improve the number of sales and revenue of PoliOnline, a set of statistics about the managed items and customers are computed based on the following input data sets/files.

- ItemsCatalog.txt
  - ItemsCatalog.txt is a textual file containing the information about the items that are sold by PoliOnline. There is one line for each item and the total number of items is greater than 1,000,000. This file is large and you cannot suppose the content of ItemsCatalog.txt can be stored in one in-memory Java variable.
  - Each line of ItemsCatalog.txt has the following format
    - ItemID,Name,Category,FirstTimeInCatalog

      where *ItemID* is the item unique identifier, *Name* is the name of *ItemID*, *Category* is its category (i.e., the item category), and *FirstTimeInCatalog* is the first timestamp in which *ItemID* was included in the catalog of PoliOnline. The format of FirstTimeInCatalog is "YYYY/MM/DD-HH:MM:SS".

    - For example, the following line

      ID1,t-shirt-winter,Clothing,2001/03/01-12:00:00

      means that the item with ItemID ***ID1*** is characterized by the name ***t-shirt-winter***, it belongs to the ***Clothing*** category, and it was included in the PoliOnline's catalog on March 1, 2001, at 12:00:00.

- Customers.txt
  - Customers.txt is a textual file containing the information about the customers who are registered on the web site of PoliOnline. There is one line for each customer and the total number of customers is greater than 10,000,000. This file is large and you cannot suppose the content of Customers.txt can be stored in one in-memory Java variable.
  - Each line of Customers.txt has the following format
    - Username,Name,Surname,DateOfBirth

      where *Username* is the customer unique identifier, *Name* and *Surname* are his/her name and surname, respectively, and DateOfBirth is his/her date of birth. The DateOfBirth format is "YYYY/MM/DD".

    - For example, the following line

      User20,Paolo,Garza,1976/03/01

      means that the name and surname of customer ***User20*** are ***Paolo*** and ***Garza***, respectively, and that the customer was born on March 1, 1976.

- Purchases.txt
  - Purchases.txt is a textual file containing information about purchases/sales. A new line is inserted in Purchases.txt every time an item is bought by a customer (i.e., each line corresponds to one purchase). Purchases.txt contains the historical data about the last 20 years. This file is big and you cannot suppose the content of Purchases.txt can be stored in one in-memory Java variable.
  - Each line of Purchases.txt has the following format
    - SaleTimestamp,Username,ItemID,SalePrice

      where *SaleTimestamp* is the timestamp at which the customer *Username* bought the item identified by *ItemID*. *SalePrice* is the price at which *Username* bought *ItemID.*

    - For example, the following line

      2019/02/02-09:15:01,User20,ID1,50.99

      means that on ***February 2, 2019***, at ***09:15:01*** the item identified by ***ID1*** was bought by customer ***User20***, and ***User20*** bought that item for ***50.99*** euro. The format of SaleTimestamp is "YYYY/MM/DD-HH:MM:SS".

    Note that the same customer can buy the same item multiple times, in different timestamps.

Exercise 1 – MapReduce and Hadoop (8 points)

Exercise 1.1

The managers of PoliOnline are interested in performing some analyses about purchases.

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

1. *The first year with the maximum number of purchases.* The application considers all purchases and selects the year with the maximum number of purchases. If there is more than one year associated with maximum number of purchases, the first one in the temporal order is selected. Store the selected year and the associated number of purchases in the output HDFS folder (the pair (year, number of purchases in that year).

   Examples

   - *First example.* For the sake of simplicity only three years are considered in this example. Suppose that year 2019 is characterized by 100K purchases, the year 2020 by 120K purchases, and the year 2021 by 65K purchases. In this case, the year 2020 is selected and the pair (2020,120000) is stored in the output folder.

- *Second example.* For the sake of simplicity only three years are considered in this example. Suppose that year 2019 is characterized by 110K purchases, the year 2020 by 50K purchases, and the year 2021 by 110K purchases. In this case, the year 2019 is selected and the pair (2019,110000) is stored in the output folder. Also the year 2021 has 110K purchases but only the year 2019 is selected because 2019 precedes 2021.

Suppose that the input is Purchases.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 next 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"
- **Write your code on your papers.**

Answer the following two questions (Exercises 1.2 and 1.3) to specify the number of jobs (one or two) and the number of instances of each reducer.

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 PoliOnline are interested in performing some analyses about their items.

The managers of PoliOnline asked you to develop one single application to address all the analyses they are interested in. The application has five arguments: the input files ItemsCatalog.txt, Customers.txt, and Purchases.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. *Items purchased at least 10K times in 2020 and at least 10K times in 2021.* This first part of the application **considers all the items** included in the catalog and selects the subset of items bought at least 10000 times in the year 2020 and at least 10000 times in the year 2021. Only the subset of items that satisfy both conditions are selected. The identifiers of the selected items are stored in the first HDFS output folder (one ItemID per line).

2. *Items included in the catalog before the year 2020 with at least two months in the year 2020 each one with less than 10 distinct customers*. This second part of the application **considers only the items that were included in the catalog before the year 2020** (i.e., the items with FirstTimeInCatalog<'2020/01/01'). Considering only those items, an item is selected if it is characterized by at least two months in the year 2020 such that each of those months has less than 10 distinct customers who purchased that item. The identifiers and the categories of the selected items are stored in the second output folder (one pair (ItemID, Category) per output line).

   Note. The months with less than 10 distinct customers can be either consecutive or not consecutive.

   Examples Point 2

   For the sake of simplicity, the following table reports the number of distinct customers for each item in each month of the year 2020 for a small subset of items selected randomly from the big set of items managed by PoliOnline (the number of distinct customers in each month for each item is computed from the input files). This is only a toy example. Your solution must manage all the items sold by PoliOnline.

   Given the example values reported in the following table, the following four pairs will be selected and stored in the second output folder:
   - Item1, Sport
   - Item2, House
   - Item3, Sport
   - Item4, Sport

   Item5 and Item6 **are not selected**.

	Months of the year 2020											
ItemId,Category	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**
Item1, Sport	100	101	15	50	**5**	51	200	100	100	100	**2**	100
Item2, House	110	200	130	160	60	70	50	100	**5**	**0**	100	100
Item3, Sport	110	40	**1**	600	34	100	**6**	100	140	105	206	**9**
Item4, Sport	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Item5, Sport	55	60	45	104	102	**2**	100	60	150	104	400	100
Item6, House	100	130	300	25	100	100	11	140	55	63	104	44

Number of distinct customers for each pair “item, month of the year 2020”.

- **Write your code on your papers.**
- 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"