Big Data: Architectures and Data Analytics

July 4, 2022

Student ID ____

First Name ____

Last Name ____

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

Part I

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

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

```
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);

                // Read a first input file
                JavaRDD<String> Temp1RDD = sc.textFile("Temperature1.txt");

                // Print on the standard output the number of elements of Temp1RDD
                System.out.println(Temp1RDD.count());

                // Read a second input file
                JavaRDD<String> Temp2RDD = sc.textFile("Temperature2.txt");

                // Print on the standard output the number of elements of Temp2RDD
                System.out.println(Temp2RDD.count());

                // Create an RDD that contains the union of Temp1RDD and
                // Temp2RDD
                JavaRDD<String> UnionRDD = Temp1RDD.union(Temp2RDD);

                // Computes the number of lines of UnionRDD
                long numLinesUnion = UnionRDD.count();

                // Print on the standard output the value of numLinesUnion
                System.out.println(numLinesUnion);

                // Store the content of UnionRDD in the output folder
                UnionRDD.saveAsTextFile("outputFolder/");
                sc.close();
        }
}
```

Suppose the input files Temperature1.txt and Temperature2.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 Temperature1.txt 1 time and the content of Temperature2.txt 1 time.
b) This application reads the content of Temperature1.txt 2 times and the content of Temperature2.txt 2 times.
c) This application reads the content of Temperature1.txt 3 times and the content of Temperature2.txt 3 times.
d) This application reads the content of Temperature1.txt 4 times and the content of Temperature2.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
            // 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** 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

PoliDataCenters is a company that manages several data centers located around the world. The staff of PoliDataCenters logs data about the managed servers, the available patches for each operating system, and the applied patches. The analyses of interest are based on the following input data sets/files.

- Servers.txt
  - Servers.txt is a text file containing the list of servers managed by PoliDataCenters. Each line of Servers.txt is associated with one server. PoliDataCenters manages millions of servers.
  - Each line of Servers.txt has the following format
    - SID,OperatingSystem,Model

      where *SID* is the unique identifier of the server while *OperatingSystem* is its operating system and *Model* is its model.

    - For example, the following line

      S10,Ubuntu6,SunUltraServer1

      means that the server identified by the SID ***S10*** has the operating system ***Ubuntu6*** and is a ***SunUltraServer1*** server.

- Patches.txt
  - Patches.txt is a textual file containing the information about the patches of more than 100 operating systems. There are millions of patches in this file.
  - Each line of Patches.txt has the following format
    - PID,ReleaseDate,OperatingSystem

      where *PID* is the patch identifier, *ReleaseDate* is the date on which the patch was released, and *OperatingSystem* is the operating system for which the patch was released.
    - For example, the following line

      PIDW10_22,2022/01/18,Ubuntu6

      means that the patch with id ***PIDW10_22***, which is a patch for ***Ubuntu6***, was released on ***January 18, 2022***.

- AppliedPatches.txt
  - AppliedPatches.txt is a textual file containing the information about which patches were applied on each server in the last 30 years.
  - Each line of AppliedPatches.txt has the following format
    - PID,SID, ApplicationDate

      where *PID* is the identifier of the applied patch, *SID* is the identifier of the server on which the patch PID was applied, and *ApplicationDate* is the date associated with the application of PID on SID.
      Each line of AppliedPatches.txt is uniquely identified by the primary key (PID,SID). Each patch can be applied to many servers and each server can be associated with many patches but the same combination (PID,SID) occurs at most one time in AppliedPatches.txt.
      Moreover, the data are correct and hence each server was patched only with patches associated with its operating system.
    - For example, the following line

      PIDW10_22,S10,2022/02/21

      means that the patch with id ***PIDW10_***22 was applied on the server with SID ***S10*** on ***February 21, 2022***.

Exercise 1 – MapReduce and Hadoop (8 points)

Exercise 1.1

The managers of PoliDataCenters 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. *Operating system with the maximum number of patches released in the last year.* The application considers only the patches released in the last year (ReleaseDate from July 4, 2021 to July 3, 2022) and selects the operating system with the highest number of released patches in the last year. If there is more than one operating system associated with the maximum number of released patches in the last year, the application selects the first operating system in alphabetical order. Store the selected operating system in the output HDFS folder.

Suppose that the input is Patches.txt and has been already set. Suppose that also the name of the output folder has been already set.

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

Answer the following two questions to specify the number of jobs (one or two) 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 PoliDataCenters asked you to develop one single application to address all the analyses they are interested in. The application has five arguments: the input files Servers.txt, Patches.txt, and AppliedPatches.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. *Patches of the operating system Ubuntu 2 that were applied on many servers on the same date they were released.* The first part of this application considers the patches released for the operating system Ubuntu 2 and selects the patches that were applied on at least 100 servers on the same date they were released (i.e., consider only the cases with ApplicationDate equal to ReleaseDate). Store the identifiers (PIDs) of the selected patches in the first HDFS output folder (one PID per line).

2. *Number of months of the year 2021 without applied patches for each server.* The second part of this application computes for each server the number of months of the year 2021 in which no patches have been applied. Store in the second HDFS output folder the SID and the number of months of the year 2021 in which no patches have been applied for each server (one pair (SID, number of months of the year 2021 in which no patches have been applied to SID) per output line). The servers with 0 months of the year 2021 without applied patches are also stored in the second output folder.

   Examples Point 2

   - For the sake of clarity, suppose that there are only three servers, which are associated with the following identifiers: SID1, SID2, and SID3. The following table reports the number of patches applied to each server in the twelve months of the year 2021.

	Months of the year 2021											
SID	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**
SID1	1	2	**0**	5	2	**0**	7	5	1	3	10	**0**
SID2	4	5	1	20	3	1	6	7	10	4	4	4
SID3	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

The following output is generated for this toy example

- SID1,3
- SID2,0
- SID3,12

- **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.