Big data processing and analytics

June 23, 2026

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 application.

```
RDD_A = sc.textFile("A.txt")


# Select a subset of lines
RDD_B = RDD_A.filter(lambda line: line.split(',')[1] == 'Polito')


# Count the number of elements and print it
v2 = RDD_B.count()

print(v2)


# Create an RDD of integers and cache it
RDD_C = RDD_B.map(lambda line: int(line.split(',')[0]))

RDD_D = RDD_C.cache()


# Compute max and min, and their difference
min_Value = RDD_D.reduce(lambda v1, v2: min(v1, v2))

max_Value = RDD_D.reduce(lambda v1, v2: max(v1, v2))


diff_value = max_Value - min_Value


# Print diff_value

print(diff_value)
```

Suppose the input file A.txt is read from HDFS, and this Spark application is executed only **once**. Suppose also that the content of RDD_C is small enough to be completely stored in memory when cached.

Which one of the following statements is **true**?

a) The file A.txt is read from HDFS 1 time.
b) The file A.txt is read from HDFS 2 times.
c) The file A.txt is read from HDFS 3 times.
d) The file A.txt is read from HDFS 4 times.

2. (2 points) Consider the following MapReduce application for Hadoop.

DriverBigData.java

```
/* Driver class */
package it.polito.bigdata.hadoop;
import ….;


/* Driver class */
public class DriverBigData extends Configured implements Tool {
  @Override
public int run(String[] args) throws Exception {
    int exitCode;

    Configuration conf = this.getConf();

    // Define a new job
    Job job = Job.getInstance(conf);

    // Assign a name to the job
    job.setJobName("Exam question");

    // Set path of the input file/folder for this job
    FileInputFormat.addInputPath(job, new Path("inputFolder/"));

    // Set path of the output folder for this job
    FileOutputFormat.setOutputPath(job, new Path("outputFolder/"));

    // Specify the class of the Driver for this job
    job.setJarByClass(DriverBigData.class);

    // Set job input format
    job.setInputFormatClass(TextInputFormat.class);

    // Set job output format
    job.setOutputFormatClass(TextOutputFormat.class);

    // Set map class
    job.setMapperClass(MapperBigData.class);
```

```
    // Set map output key and value classes
    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(NullWritable.class);



    // Set reduce class
    job.setReducerClass(ReducerBigData.class);

    // Set reduce output key and value classes
    job.setOutputKeyClass(IntWritable.class);
    job.setOutputValueClass(NullWritable.class);

    // Set the number of reducers to 2
    job.setNumReduceTasks(2);



    // Execute the job and wait for completion
    if (job.waitForCompletion(true)==true)
     exitCode=0;
    else
     exitCode=1;

    return exitCode;
  }

  /* Main of the driver */
  public static void main(String args[]) throws Exception {
     int res = ToolRunner.run(new Configuration(), new DriverBigData(), args);
     System.exit(res);
   }
}
```

MapperBigData.java

```
/* Mapper class */
package it.polito.bigdata.hadoop;
import …;

class MapperBigData extends
    Mapper<LongWritable, // Input key type
                    Text, // Input value type
                    Text, // Output key type
                    NullWritable> { // Output value type

    protected void map(LongWritable key, // Input key type
                    Text value, // Input value type
```

```
                Context context) throws IOException, InterruptedException {

                // Emit the pair (value, NullWritable)
                context.write(new Text(value), NullWritable.get());
    }
}
```

ReducerBigData.java

```
/* Reducer class */
package it.polito.bigdata.hadoop;
import …;

class ReducerBigData extends
        Reducer<Text,    // Input key type
                NullWritable,   // Input value type
                IntWritable,    // Output key type
                NullWritable> { // Output value type

    // Define count
    int count;

    protected void setup(Context context) {
        // Initialize count
        count = 0;
    }


    protected void reduce(Text key, // Input key type
                Iterable<NullWritable> values, // Input value type
                Context context) throws IOException, InterruptedException {
        // Define and initialize occurrences
        int occurrences = 0;

        // Iterate over the set of values and increment occurrences
        for (NullWritable value: values) {
                occurrences++;
        }

        // Increment count if occurrences is greater than 2
        if (occurrences>2)
                count++;
    }

    protected void cleanup(Context context) throws IOException, InterruptedException {
        // Emit the pair (count, NullWritable))
        context.write(new IntWritable(count), NullWritable.get());
```

```
    }
}
```

Suppose that inputFolder contains the files Cities1.txt, Cities2.txt, and Cities3.txt. Suppose the HDFS block size is 512 MB.

Content of Cities1.txt, Cities2.txt, and Cities3.txt:

Filename	Content
Cities1.txt	Beijing Cairo Turin
Cities2.txt	Chongqing Cairo Turin
Cities3.txt	Istanbul Milan Cairo Turin

Suppose we run the above MapReduce application (note that the input folder is set to inputFolder/).

What is a ***potential*** output that could be generated by running the MapReduce application reported above?

a) The content of the output folder is as follows.
-rw-r--r-- 1 paolo paolo 2 set 3 14:00 part-r-00000
-rw-r--r-- 1 paolo paolo 2 set 3 14:00 part-r-00001
-rw-r--r-- 1 paolo paolo 0 set 3 14:00 _SUCCESS

The content of the part files is as follows.

Filename (number of lines)	Content
part-r-00000 (1 line)	2
part-r-00001 (1 line)	0

b) The content of the output folder is as follows.
-rw-r--r-- 1 paolo paolo 2 set 3 14:00 part-r-00000
-rw-r--r-- 1 paolo paolo 2 set 3 14:00 part-r-00001
-rw-r--r-- 1 paolo paolo 0 set 3 14:00 _SUCCESS

The content of the part files is as follows.

Filename (number of lines)	Content
part-r-00000 (1 line)	6
part-r-00001 (1 line)	0

c) The content of the output folder is as follows.
-rw-r--r-- 1 paolo paolo 2 set 3 14:00 part-r-00000
-rw-r--r-- 1 paolo paolo 2 set 3 14:00 part-r-00001
-rw-r--r-- 1 paolo paolo 0 set 3 14:00 _SUCCESS

The content of the part files is as follows.

Filename (number of lines)	Content
part-r-00000 (1 line)	6
part-r-00001 (1 line)	6

d) The content of the output folder is as follows.
-rw-r--r-- 1 paolo paolo 2 set 3 14:00 part-r-00000
-rw-r--r-- 1 paolo paolo 2 set 3 14:00 part-r-00001
-rw-r--r-- 1 paolo paolo 0 set 3 14:00 _SUCCESS

The content of the part files is as follows.

Filename (number of lines)	Content
part-r-00000 (1 line)	3
part-r-00001 (1 line)	3

Part II

The managers of PoliEShop, an international e-commerce company, have asked you to develop some applications to support some analyses they are interested in. These analyses are based on the following input datasets/files.

- Customers.txt
  - This file contains the profile information for the customers registered on the platform. Each line represents a user. This file is large. You cannot assume its content can be stored in a single in-memory variable.
  - Each line in Customers.txt has the following format:
    - CID,DateOfBirth,Gender,Country
    - where
      - CID is the unique identifier of the user
      - DateOfBirth is the date of birth of the user (format: YYYY/MM/DD)
      - Gender is the gender of the user
      - Country is the country of residence of the user
    - For example, the following line

      C10,1976/03/01,F,Italy

      means that user C10 is an Italian female from Italy born on March 1, 1976.

- Products.txt
  - This file contains information about the products sold by the company. Each line represents a product. This file is large. You cannot assume its content can be stored in a single in-memory variable.
  - Each line in Products.txt has the following format:
    - PID,Name,Category
    - where
      - PID is the unique identifier of the product
      - Name is its name
      - Category is the product category (e.g., Electronics, Clothing, etc.)
    - For example, the following line

      PID34,PoliCap,Clothing

      means that product PID34 is called PoliCap and belongs to the clothing category.

- Purchases.txt
  - This is a textual file containing information about Purchases. Each line represents a purchase. This file is large. You cannot assume its content can be stored in a single in-memory variable.
  - Each line in Purchases.txt has the following format:

        - PurchID,CID,PID,PurchTimestamp
        - where
            - PurchID is the unique identifier of the purchase
            - CID is the identifier of the customer associated with the purchase
            - PID is the identifier of the product associated with the purchase
            - PurchTimestamp is the timestamp when the purchase occurred (timestamp format: YYYY/MM/DD:HH:MM)
        - For example, the following line

          Pur125,C10,PID34,2024/11/15:10:15

          means that customer C10 purchased product PID34 on November 15, 2024, at 10:15. The purchase is identified by the value Pur125.
    - Note that each user can purchase the same product multiple times, also at the same timestamp. Moreover, each user can buy different products.

Exercise 1 – MapReduce and Hadoop (8 points)

Exercise 1.1

The managers of PoliEShop are interested in performing some analyses about their data.

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

1. *Products without purchases in 2019, but the maximum number of purchases in 2020*. The application considers only the purchases in 2019 and 2020, and only the products with at least one purchase in 2020. Remind that each line of Purchases.txt represents one purchase. The application selects the product satisfying both the following constraints: (i) no purchases in 2019 and (ii) the maximum number of purchases in 2020. In case of many products satisfying both constraints, the first one according to the lexicographical order is selected. Store the PID of the selected product in the output folder, along with the number of its purchases in 2020.

   The single output line has the following format:
   PID,number of purchases in 2020 associated with product PID

Suppose that the input is Purchases.txt and it 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 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 (19 points)

The managers of PoliEShop 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. *French customer(s) with the maximum number of different products purchased in 2020.* The first part of this application considers only purchases made in 2020 by French customers. It selects the identifier of the French customer(s) who purchased the maximum number of different products in the year 2020 by considering only the purchases made by French customers in 2020. Suppose there is at least one

purchase made by French customers in 2020 (i.e., the maximum number of different products purchased in 2020 by French customers is greater than 0). The result is stored in the first output folder. In case of a tie, all French customers associated with the maximum value must be stored (each output line contains the identifier of one of the selected customers).

2. *French customers with the maximum number of purchases for at least one product*. The second part of this application considers only French customers and all their purchases, independently of the purchase timestamps. Only the products purchased at least once by at least one French customer are considered (i.e., products that have never been purchased by French customers must not be considered). This second part of the application selects the French customers who are associated with the maximum number of purchases for at least one product, considering only purchases made by French customers. The result is stored in the second output folder (the CID of one selected customer per output line). If a customer is associated with the maximum number of purchases for many products, that customer must be stored in the second output folder only once. The output format is as follows:

   CID

   Example Part 2
   Consider a toy example that contains only three customers (CID1, CID2, and CID3) and only two products: PID1, PID2.

   Suppose that:
   - CID1 purchased **PID1 10 times** and **PID2 35 times**
   - CID2 purchased **PID1 10 times** and PID2 3 times
   - CID3 purchased PID1 1 time and never purchased PID2

   For this toy running example, the maximum number of purchases for product PID1 is 10, while for product PID2 the maximum number of purchases is 35. Hence, CID1 and CID2 are selected because CID1 is associated with the maximum number of purchases for both products, while CID2 is associated with the maximum number of purchases for PID1.

   Therefore, for this running example, the content of the second output folder is:
   CID1
   CID2

- You do not need to write Java imports. Focus on the content of the main method.
- Suppose both **SparkContext 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.