Big Data: Architectures and Data Analytics

January 22, 2018

Student ID _____

First Name _____

Last Name _____

The exam is **open 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 HDFS folder "inputData" containing the following two files:

Filename	Size	Content of the files	HDFS Blocks	
			Block ID	Content of the block
Prices1.txt	24 bytes	21.55 52.55 43.55 10.55	B1	21.55 52.55
			B2	43.55 10.55
Prices2.txt	12 bytes	60.55 60.55	B3	60.55 60.55

Suppose that you are using a Hadoop cluster that can potentially run up to 4 mappers in parallel and suppose that the HDFS block size is 12 bytes.

Suppose that the following MapReduce program is executed by providing the folder "inputData" as input folder and the folder "results" as output folder.

```
/* Driver */
import … ;
public class DriverBigData extends Configured implements Tool {
        @Override
        public int run(String[] args) throws Exception {
                Configuration conf = this.getConf();
                Job job = Job.getInstance(conf);
                job.setJobName("2018/01/22 - Theory");

                FileInputFormat.addInputPath(job, new Path(args[0]));
                FileOutputFormat.setOutputPath(job, new Path(args[1]));

                job.setJarByClass(DriverBigData.class);

                job.setInputFormatClass(TextInputFormat.class);
                job.setOutputFormatClass(TextOutputFormat.class);

                job.setMapperClass(MapperBigData.class);
                job.setMapOutputKeyClass(DoubleWritable.class);
                job.setMapOutputValueClass(NullWritable.class);
```

```
                job.setNumReduceTasks(0);

                if (job.waitForCompletion(true) == true)
                        return 0;
                else
                        return 1;
        }

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

/* Mapper */

```
import …;

class MapperBigData extends Mapper<LongWritable, Text, DoubleWritable, NullWritable> {
        Double maxPrice, minPrice;

        protected void setup(Context context) {
                maxPrice = null;
                minPrice = null;
        }

        protected void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {

                Double price = new Double(value.toString());

                if (maxPrice == null || price.doubleValue() > maxPrice) {
                        maxPrice = price;
                }

                if (minPrice == null || price.doubleValue() < minPrice) {
                        minPrice = price;
                }


        }

        protected void cleanup(Context context) throws IOException, InterruptedException {
                // emit the maximum difference
                Double maxDiff = maxPrice-minPrice;
                context.write(new DoubleWritable(maxDiff), NullWritable.get());
        }
}
```

What is the output generated by the execution of the application reported above?

a) The output folder contains two files

- One file that contains the following line

  42.00

- One file that contains the following line

  0.00

b) The output folder contains three files

- One file that contains the following line

  31.00

- One file that contains the following line

  33.00

- One file that contains the following line

  0.00

c) The output folder contains only one file

- The output file contains the following line

  50.00

d) The output folder contains only one file

- One file that contains the following three lines

  31.00

  33.00

  0.00

2. (2 points) Consider the HDFS folder MyLogs, which contains two files: logs1.txt and logs2.txt. The size of logs1.txt is 500MB and the size of logs2.txt is 524MB. Suppose that you are using a Hadoop cluster that can potentially run up to 10 mappers in parallel and suppose to execute a MapReduce-based program that selects the rows of the files in MyLogs containing the word “POLITO”. Which of the following values is a proper HDFS block size if you want to “force” Hadoop to run exactly 3 mappers in parallel when you execute the application by specifying the folder MyLogs as input?

   a) Block size: 1024MB

   b) Block size: 512MB

   c) Block size: 256MB

   d) Block size: 128MB

Part II

BigDataBooks is a large e-commerce company selling books. The management of BigDataBooks is interested in analyzing their data to improve the quality of their books.

The analyses are based on the following data sets/files.

- Books.txt
  - Books.txt is a large textual file containing the catalog of available books
  - The file contains one single line for each book
  - Each line of the file has the following format
    - bid,title,genre,publisher,year_publication

      where, *bid* is the book identifier, *title* is its title, *genre* is the genre (e.g., Adventure, Romance, Mystery) of the book, *publisher* is the name of its publisher, and *year_publication* is the year when it has been published. Each book is associated with one single genre.

    - For example, the line

      B1020,The Body in the Library,Crime,Dodd and Company,1942

      means that the title of book ***B1020*** is "***The Body in the Library***", its genre is "***Crime",*** its publisher is "***Dodd and Company***", and it has been published in 1942.

- Purchases.txt
  - Purchases.txt is a textual file containing the list of purchases of the last 5 years
  - Every time a customer buys a book a new line is appended at the end of Purchases.txt, i.e., each line of the file contains the information about one single purchase
  - Each line of the file has the following format
    - customerid,bid,timestamp,price

      where *customerid* is a customer identifier, *bid* is the book identifier, timestamp is the time at which *customerid* bought/purchased *bid* and *price* is the cost of the purchase.

    - For example, the line

      customer1,B1020,20170502_13:10,19.99

      means that ***customer1*** bought ***B1020*** on ***May 2, 2017*** at ***13:10*** and the price of the purchase was ***19.99***€

Exercise 1 – MapReduce and Hadoop (9 points)

The managers of BigDataBooks are interested in selecting the most purchased book in year 2016.

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

A. *Most purchased book*. Considering only the subset of purchases of year 2016, the application must select the bid of the most purchased book in year 2016. The most purchased book in year 2016 is the one with the highest number of purchases in year 2016, based on the data reported in Purchases.txt. In case of tie (i.e., two or more books associated with the highest number of purchases in 2016), the application selects as the most purchased book the first one according to the alphabetical order (e.g., if books "A101" and "B203" are both characterized by the highest number of purchases in year 2016, "A101" is selected). Store the bid of the most purchased book in year 2016 in a HDFS folder. The output file contains one single line with the bid of the most selected book and the number of purchases in 2016 for that book. The line of the output file has the following format
   - Bid\t*number_of_purchases_in_2016*

The name of the output folder is one argument of the application. The other argument is the path of the input file Purchases.txt. Note that Purchases.txt contains the data of the last 5 years but the analysis is focused only on year 2016 (i.e., from January 1, 2016 to December 31, 2016).

Fill in the provided template for the Driver of this exercise. Use your papers for the other parts (Mapper and Reducer).

Exercise 2 – Spark and RDDs (18 points)

The managers of BigDataBooks are interested in performing an analysis about the maximum and minimum price at which each book has been sold in the last year (2017), by considering only the books that have been purchased at least one time in 2017. Specifically, the managers of BigDataBooks are interested in selecting the subset of books that are characterized by a significant difference (greater than 15€) between the maximum and the minimum price at which they have been sold in 2017 and understand the reasons of this anomalous behavior.

The managers of BigDataBooks are also interested in performing some analyses about the genres of the unsold books. Specifically, they are interested in analyzing the books that have never been purchased in the last year (2017) and computing the percentage of never purchased books for each genre.

The managers of BigDataBooks asked you to develop an application to address all the analyses they are interested in. The application has four arguments/parameters: the files Books.txt and Purchases.txt and two output folders (associated with the outputs of the following points A and B, respectively).

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

A. (8 points) *Books with anomalous price behavior in 2017.* The application must select the books with an anomalous price behavior in 2017. By considering only the purchases related to year 2017, a book is characterized by an anomalous price behavior if the difference between the maximum price and the minimum price associated to the purchases of that book in 2017 is greater than 15€. The application stores in the first HDFS output folder the information "bid,maximum price,minimum price" for all the books with an anomalous price behavior in 2017 (one line for each of the selected books).

B. (10 points) *Percentage of never purchases books in the last year for each genre.* The application must select the books that have never been purchased in year 2017 and compute the percentage of never purchased books for each genre in year 2017. The application stores in the second HDFS output folder the information "genre,percentage of never purchased books in 2017 for that genre" for all genres (one line for each genre). The output must contain also the genres for which the percentage of never purchased books in 2017 is equal to 0. **Note: To solve this part of the exam, you can assume that the number of books that have never been purchased in year 2017 is small and can be stored in a local Java variable, whereas the complete catalog of books cannot be stored in a local variable.**

Big Data: Architectures and Data Analytics

January 22, 2018

Student ID _____

First Name _____

Last Name _____

Use the following template for the Driver of Exercise 1

Fill in the missing parts. You can strikethrough the second job if you do not need it.

```
import ….
/* Driver class. */
public class DriverBigData extends Configured implements Tool {
        public int run(String[] args) throws Exception {
        Path inputPath = new Path(args[0]);
        Path outputDir = new Path(args[1]);
        Configuration conf = this.getConf();

        // First job
        Job job1 = Job.getInstance(conf);
        job1.setJobName("Exercise 1 - Job 1");
        // Job 1 - Input path
        FileInputFormat.addInputPath(job,_____);

        // Job 1 - Output path
        FileOutputFormat.setOutputPath(job,_____);

        // Job 1 - Driver class
        job1.setJarByClass(DriverBigData.class);

        // Job1 - Input format
        job1.setInputFormatClass(_____);

        // Job1 - Output format
        job1.setOutputFormatClass(_____);

        // Job 1 -  Mapper class
        job1.setMapperClass(Mapper1BigData.class);
        // Job 1 – Mapper: Output key and output value: data types/classes
        job1.setMapOutputKeyClass(_____);

        job1.setMapOutputValueClass(_____);

        // Job 1 -  Reducer class
        job.setReducerClass(Reducer1BigData.class);

        // Job 1 – Reducer: Output key and output value: data types/classes
        job1.setOutputKeyClass(_____);

        job1.setOutputValueClass(_____);

        // Job 1 - Number of reducers
        job1.setNumReduceTasks( 0[ _ ] or 1[ _ ] or >=1[ _ ] ); /* Select only one of the three options */
```

```
        // Execute the first job and wait for completion
        if (job1.waitForCompletion(true)==true)
        {
                // Second job
                Job job2 = Job.getInstance(conf);
                job2.setJobName("Exercise 1 - Job 2");
                // Set path of the input folder of the second job
                FileInputFormat.addInputPath(job2,_____);

                // Set path of the output folder for the second job
                FileOutputFormat.setOutputPath(job2,_____);

                // Class of the Driver for this job
                job2.setJarByClass(DriverBigData.class);

                // Set input format
                job2.setInputFormatClass(_____);

                // Set output format
                job2.setOutputFormatClass(_____);

                // Set map class
                job2.setMapperClass(Mapper2BigData.class);

                // Set map output key and value classes
                job2.setMapOutputKeyClass(_____);

                job2.setMapOutputValueClass(_____);

                // Set reduce class
                job2.setReducerClass(Reducer2BigData.class);

                // Set reduce output key and value classes
                job2.setOutputKeyClass(_____);

                job2.setOutputValueClass(_____);

                // Set number of reducers of the second job
                job2.setNumReduceTasks( 0[ _ ] or 1[ _ ] or >=1[ _ ] ); /*Select only one of the three
                                                                        options*/

                // Execute the job and wait for completion
                if (job2.waitForCompletion(true)==true)
                        exitCode=0;
                else
                        exitCode=1;
           }
           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);
        }
}
```