Big Data: Architectures and Data Analytics

February 15, 2019

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 cache "mechanism" of Spark. Which one of the following statements is true?

   a) Caching an RDD is always useful

   b) PairRDDs cannot be cached

   c) At most one RDD per application can be cached

   d) Caching an RDD that is used at least two times in an application could improve the efficiency of the application (in terms of execution time).

2. (2 points) Consider the HDFS folder logsFolder, which contains four files: errors.txt, warning.txt, log1.txt, and log2.txt. The size of errors.txt is 1036MB, the size of warning.txt is 500MB, the size of log1.txt is 1024MB, and the size of log2.txt is 1024MB. Suppose that you are using a Hadoop cluster that can potentially run up to 12 mappers in parallel and suppose to execute a Map-only MapReduce-based program that selects only the lines of the input files in logsFolder containing the words "ERROR" or "WARNING". The HDFS block size is 512MB. How many parallel mappers are instantiated by Hadoop when you execute the application by specifying the folder logsFolder as input?

   a) 4

   b) 8

   c) 9

   d) 12

Part II

PoliCityQuality is a European company specialized in evaluating the quality of life in the European cities. To evaluate the quality of life, the managers of the PoliCityQuality use the information about the number and types of POIs (points of interest) of each city. Each city can have thousands of POIs.

The analyses are based on the following input data set/file.

- EuropeanPointsOfInterest.txt
  - EuropeanPointsOfInterest.txt is a text file containing the information about the POIs of all the European cities (i.e., millions of POIs).
  - Each line of EuropeanPointsOfInterest.txt contains the information about one point of interest and has the following format
    - ID_POI,city,country,main_category,specialized_category

      where *ID_POI* is the identifier of the represented POI, *city* and *country* are the city and the country associated with that POI, respectively, and main_category and specialized_category represent its main category and its specialized category, respectively.
    - For example, the following line

      POI204,Turin,Italy,Restaurant,FastFood

      means that POI ***POI204*** represents a ***fast food*** (specialized category) ***restaurant*** (main category).

Exercise 1 – MapReduce and Hadoop (8 points)

The managers of PoliCityQuality are interested in selecting the subset of Spanish cities with many "restaurant" POIs, few of which are fast foods.

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

A. *Spanish cities with high amount of "restaurant" POIs but few fast foods*. Considering only the Spanish cities and their POIs, the application must select the Spanish cities with (i) more than 200 restaurants POIs (main category="Restaurant") and (ii) less than 30 fast foods (specialized category="FastFood"). Note that the "FastFood" specialized category is a subcategory of the "Restaurant" main category. The output file contains one line for each of the selected Italian cities (one city name per line).

- The name of the output folder is an argument of the application. The other argument is the path of the input file EuropeanPointsOfInterest.txt. Note that EuropeanPointsOfInterest.txt contains the POIs of all European cities but the analysis is focused only on the Spanish cities.

Fill out the provided template for the Driver of this exercise. Use your sheets of paper for the other parts (Mapper and Reducer).

Exercise 2 – Spark and RDDs (19 points)

The managers of PoliCityQuality are interested in selecting the German cities with at least two pharmacies but without hospitals (i.e., those with POIs with specialized category="Pharmacy" but without POIs with specialized category="Hospital").

Moreover, they are interested in selecting the German cities with a number of universities less than the average number of universities per city in Germany.

The managers of PoliCityQuality asked you to develop one single application to address the two analyses they are interested in. The application has three arguments: the input file EuropeanPointsOfInterest.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 write the corresponding Java code, to address the following points:

A. (8 points) *German cities with at least two pharmacies but without hospitals*. Considering only the German cities, the application must select the German cities with at least two "pharmacy" POIs (specialized category="Pharmacy) but without "hospital" POIs (specialized category="Hospital"). The application stores in the first HDFS output folder the selected cities, one city per line.

B. (11 points) *German cities with "few" universities with respect to the other German cities.* Considering only the German cities, the application must select the German cities with a number of "university" POIs (specialized category="University") less than the average number of "university" POIs per city in Germany. The application stores in the second HDFS output folder the selected cities, one city per line.

   For instance, as a running example, suppose that in Germany there are only three cities: Borna, Berlin, and Munich. Suppose that Borna has 0 "university" POIs, Berlin has 10 "university" POIs, and Munich has 8 "university" POIs. Hence, the average number of "university" POIs per city in Germany is 6 and only Borna must be selected.

Big Data: Architectures and Data Analytics

February 15, 2019

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 exactly 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 exactly 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);
        }
}
```