Big Data: Architectures and Data Analytics

July 16, 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 following driver of a Spark application.

```
package …
import ….
public class SparkDriver {
    public static void main(String[] args) {
        SparkConf conf=new SparkConf().setAppName("Exam");
        JavaSparkContext sc = new JavaSparkContext(conf);


        /* Analyze errors */
        JavaRDD<String> errorsRDD = sc.textFile("errors.txt");
        Long numErrors = errorsRDD.count();
        System.out.println(numErrors);


        /* Analyze warnings */
        JavaRDD<String> warningsRDD = sc.textFile("warnings.txt");
        JavaRDD<String> selWarningsRDD = warningsRDD.filter(w -> w.startsWith("warning 1"));

        Long numWarnings = warningsRDD.count();
        Long numSelectedWarnings = selWarningsRDD.count();
        System.out.println("Num. warnings: "+numWarnings);
        System.out.println("Num. selected warnings: "+numSelectedWarnings);


        sc.close();
    }
}
```

Which one of the following statements is true?

a) Caching *errorsRDD* can improve the efficiency of the application (in terms of execution time)

b) Caching *warningsRDD* can improve the efficiency of the application (in terms of execution time)

c) Caching *selWarningsRDD* can improve the efficiency of the application (in terms of execution time)

d) Caching the RDDs of this Spark application does not improve its efficiency (in terms of execution time)

2. (2 points) Consider an input HDFS folder *inputFold* containing the files log1.txt and log2.txt. The size of log1.txt is 2048MB and the size of log2.txt is 512MB. Suppose that you are using a Hadoop cluster that can potentially run up to 5 instances of the mapper in parallel and suppose to execute the word count application, based on MapReduce, by specifying *inputFold* as input folder. Which of the following values is a proper HDFS block size if you want to "force" Hadoop to run 5 instances of the mapper in parallel when you execute the word count application by specifying *inputFold* as input folder?

   a) Block size: 2048MB

   b) Block size: 1536MB

   c) Block size: 1024MB

   d) Block size: 512MB

Part II

PoliCar is an international automobile company characterized by several production plants around the world. In each production plant, robots are used to produce some components of the PoliCar's automobiles. PoliCar computes a set of statistics to identify critical robots. Specifically, PoliCar stores the faults of its robots and the associated information in the following input data sets/files.

- Robots.txt
    - Robots.txt is a text file containing the list of robots managed by PoliCar. Robots.txt contains one line for each robot of PoliCar. The number of managed robots is more than 10,000.
    - Each line of Robots.txt has the following format
        - RID,PlantID,MAC

          where *RID* is the robot identifier, PlantID is the identifier of the production plant in which the robot is installed, and *MAC* is its MAC address.

        - For example, the following line

          R5,PID5,02:42:3e:aa:a4:d8

          means that robot ***R5*** is installed in the production plant ***PID5*** and the MAC address of ***R5*** is ***02:42:3e:aa:a4:d8***.

- Faults.txt
    - Faults.txt is a text file containing the historical information about the faults of the robots. A new line is inserted in Faults.txt every time a fault of a robot occurs. The number of managed robots is more than 10,000 and Faults.txt contains the historical data about the faults of the last 15 years.
    - Each line of Faults.txt has the following format
        - RID,FaultTypeCode,FaultDuration,Date,Time

          where *RID* is the identifier of the robot with a fault, *Date* is the date of the fault, *Time* is the time of the fault, *FaultTypeCode* is the code of the type of fault, and *FaultDuration* is the number of minutes for which the robot was unavailable due to the fault.

        - For example, the following line

          R5,FCode122,20,2017/05/02,06:40:51

          means that robot R5 has been affected by the fault type FCode122 at ***06:40:51*** (hour=06, minute=40, second=51) of ***May 2, 2017*** and ***R5*** was unavailable for ***20*** minutes due to that fault.

Exercise 1 – MapReduce and Hadoop (8 points)

The managers of PoliCar are interested in selecting the robots that were affected by a *FCode100* fault (FaultTypeCode equal to "FCode100") at least one time in January 2015 and at least one time in February 2015.

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

A. *RIDs of the robots with FCode100 Faults in January and February 2015*. Considering only the faults associated with fault type code "FCode100", the application must select the identifiers (RIDs) of the robots that were affected in **both months** by the **FCode100** fault: at least one time by an FCode100 fault in **January 2015 AND** at least one time by an FCode100 fault in **February 2015**. Store the result of the analysis in a HDFS folder. The output contains one RID per line.

The arguments of the Hadoop application are (i) the path of the input file Faults.txt and (ii) the name of the output folder. Note that Faults.txt contains the faults of all types occurred in the last 15 years, but the analysis we are interested in is limited to a specific fault type and time period.

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 PoliCar are interested in (A) performing some analyses about the number of faults per production plant in the first 6 months of year 2015 (i.e., from January 2015 to June 2015) and (B) identifying faulting robots in the first 6 months of year 2015.

The managers of PoliCar asked you to develop one single application to address all the analyses they are interested in. The application has the following arguments: the path of the input file Robots.txt, the path of the input file Faults.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) *Production plant faults in the first semester of year 2015*. Considering only the faults related to the first 6 months of year 2015 (from January 2015 to June 2015), the application must select the production plants (PlantIDs) associated with at least 180 faults during the first 6 months of year 2015 and store in the first HDFS output folder the information “PlantID,Number of faults in the first 6 months of year 2015” for the selected production plants (one production plant per line).

B. (11 points) *RIDs of “Faulting robots in the first semester of 2015”*. Considering only the faults related to the first 6 months of year 2015 (from January 2015 to June 2015), the application must select the identifiers (RIDs) of the robots classified as “faulting robot in the first semester of 2015”. A robot is classified as a “faulting robot in the first semester of 2015” if (i) it was affected by **at least 5 faults in each of the first 6 months of year 2015** and (ii) **at least one fault of that robot in the first semester of year 2015 has a FaultDuration greater than 120 minutes** (i.e., the maximum value of FaultDuration for that robot from January 2015 to June 2015 is greater than 120 minutes). The application stores in the second HDFS output folder the information about the identifiers (RIDs) of the selected robots (one RID per line).

Big Data: Architectures and Data Analytics

July 16, 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 instances of the reducer of the first job
        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(_________________________________);

                // Job 2 - Number instances of the reducer 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);
        }
}
```