Big Data: Architectures and Data Analytics

June 30, 2017

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 file
HumidityValuesA.txt	16 bytes	51.45 9.55 9.15
HumidityValuesB.txt	18 bytes	40.53 10.99 52.99

Suppose that you are using a Hadoop cluster that can potentially run up to 5 mappers in parallel and suppose that the HDFS block size is 1024MB.

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("2017/06/30 - 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 top1;

        protected void setup(Context context) {
                top1 = null;
        }

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

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

                if (top1 == null || val.doubleValue() > top1) {
                        top1 = val;
                }
        }

        protected void cleanup(Context context) throws IOException, InterruptedException {
                // emit the content of top1
                context.write(new DoubleWritable(top1), NullWritable.get());
        }
}
```

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

a) One file containing 9.15

b) One file containing 52.99

c) Two files

- One containing the value 51.45
- One containing the value 40.53

d) Two files

- One containing the value 51.45
- One containing the value 52.99

2. (2 points) Consider the HDFS files logs.txt and logs2.txt. The size of logs.txt is 2000MB and the size of log2.txt is 1072MB. Suppose that you are using a Hadoop cluster that can potentially run up to 100 mappers in parallel and suppose to execute a (map-only) MapReduce-based program that receives as input the folder containing logs.txt and logs2.txt and selects the rows of the two files containing the words “WARNING” or “ERROR”. How many mappers are instantiated by Hadoop if the HDFS block size is 1024MB?

   a) 2 mappers

   b) 3 mappers

   c) 4 mappers

   d) 12 mappers

Part II

PoliWeather is an environmental company that monitors weather data for performing short- and long-term analyses. Specifically, PoliWeather is focused on temperature analyses and the analyses of interest are based on the following data sets/files.

- Temperatures.txt
  - Temperatures.txt is a text file containing the historical information about the temperatures on several cities around the world.
  - The sampling rate is 10 minutes (i.e., every 10 minutes the system collects the temperatures of the cities under analyses and a new line for each city is inserted in Temperatures.txt)
  - Each line of the input file has the following format
    - date,hour:minute,city,temperature

      where *city* is a city name and *temperature* is the observed temperature in *city* at time *date,hour:minute.*

    - For example, the line

      2016/06/20,16:10,Turin,20.5

      means that the observed temperature in ***Turin*** on ***June 20, 2016*** at ***16:10*** was ***20.5°C***

Exercise 1 – MapReduce and Hadoop (8 points)

The managers of PoliWeather are interested in performing long-term analyses of temperatures over the world. Specifically, they are interested in selecting, only for year 2015, the months associated with significant variations of the temperature values.

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

A. *Monthly min-max temperature variations*. Only for the historical data of year 2015, the application must compute for each pair (city,month) the maximum and the minimum temperature of the city during the month and also the *monthly percentage temperature variation* (i.e., (maximum temperature–minimum temperature)/ minimum temperature). Store the results, in an HDFS folder, only for those pairs (city,month) characterized by a *monthly percentage temperature variation* greater than 10%. The output contains one line for each of the selected pairs (city,month), and the format of each output line is as follows

   city_month\tmaximum-temp,minimum-temp,monthly-percentage-temp-variation

   e.g., Turin_07 35,25,40

The name of the output folder is one argument of the application. The other argument is the path of the input file Temperatures.txt.

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

Exercise 2 – Spark and RDDs (19 points)

The managers of PoliWeather are interested in selecting the maximum temperature for each pair (city, date) by considering only the historical data of year 2015. This result is exploited to draw a chart that is used to analyze the behavior of the maximum temperature of the cities under analysis.

PoliWeather is also interested in identifying the stocks that are frequently characterized by a “negative weekly trend”. Specifically, given a week of the year and a city, the weekly trend of the city temperature in that week is classified as a “negative weekly trend” if the difference between the maximum city temperature of the first day of the week and the maximum city temperature of the last day of the week is greater than 0. The application must select those cities that are characterized by more than *NW* “negative weekly trends” (i.e., more than *NW* weeks with a negative trend for each of the selected cites). *NW* is an integer number and is a parameter of the application. The analysis is based on the historical data stored in Temperatures.txt, considering only year 2015.

Note that you can easily compare dates by representing them as strings based on the format “year/month/day”. For instance, the string “2015/05/19” precedes the string “2015/05/20” (and date 2015/05/19 precedes date 2015/05/20).

Suppose that someone has already implemented the following static method.

- *public static Integer weekNumber(String date)* of the *DateTool* class.
    - The parameter of this method is a string representing a date. The returned value is an integer value that uniquely identifies the week of the year to which the provided date belongs.
    - For example, the invocation

      Integer weekNum=DateTool.*weekNumber*("*2015/06/20*");

      stores 25 in the variable weekNum because 2015/06/20 is part of the 25th week of year 2015.

The managers of PoliWeather asked you to develop an application to address the analyses they are interested in. The application has four arguments/parameters: the file Temperatures.txt, the value of NW, 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. *Maximum temperature per pair (city, date).* The application selects from Temperatures.txt only the historical temperature values observed during year 2015 and then computes, for each pair (city, date), the maximum temperature. The application stores in the first HDFS output folder the information "(*city_date,maximum temperature)*" for each pair (city, date). The results are stored in ascending order by considering the fields city, date (i.e., the results must be sorted by city; if the city is the same, then the date is considered).

B. *Select cities that are frequently characterized by a "negative weekly trend"*. The application must count, for each city, the number of weeks with a *"negative weekly trend",* based on the definition reported above, by considering only year 2015. Finally, the application stores in the second HDFS output folder only the cities characterized by at least *NW* "negative weekly trends" (i.e., at least *NW* weeks with a negative trend). The output file contains one city per line.