Distributed architectures for big data processing and analytics

July 18, 2022

Student ID _____

First Name _____

Last Name _____

The exam lasts **90 minutes**

Part I

Answer to the following questions. There is only one right answer for each question.

1. (2 points) Consider the following Spark application.

```
# Read a first input file
temp1RDD = sc.textFile("Temperature1.txt")

# Print on the standard output the number of elements of temp1RDD
print(temp1RDD.count())

# Read a second input file
temp2RDD = sc.textFile("Temperature2.txt")

# Print on the standard output the number of elements of temp2RDD
print(temp2RDD.count())

# Create an RDD that contains the union of temp1RDD and temp2RDD
unionRDD = temp1RDD.union(temp2RDD)

# Print on the standard output the number of distinct elements of unionRDD
print(unionRDD.distinct().count());
```

Suppose the input files Temperature1.txt and Temperature2.txt are read from HDFS. Suppose this Spark application is executed only 1 time. Which one of the following statements is true?

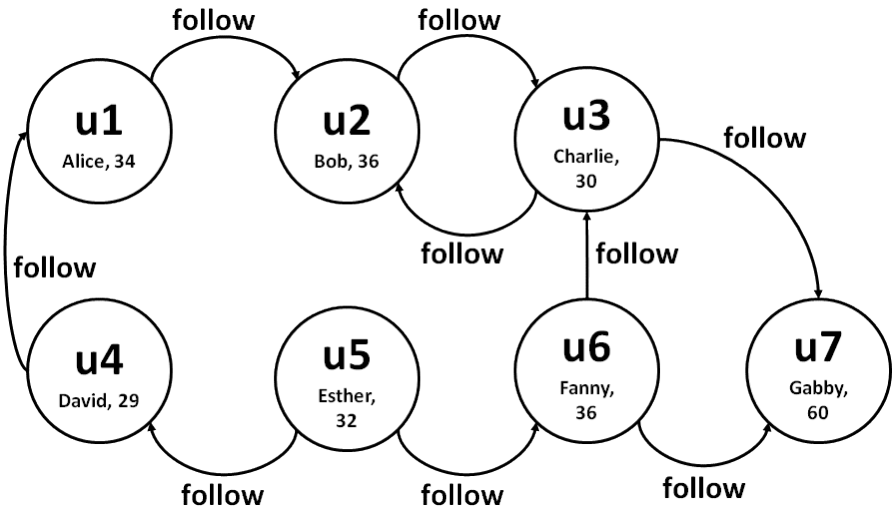
a) This application reads the content of Temperature1.txt 1 time and the content of Temperature2.txt 1 time.

b) This application reads the content of Temperature1.txt 2 times and the content of Temperature2.txt 2 times.

c) This application reads the content of Temperature1.txt 3 times and the content of Temperature2.txt 3 times.

d) This application reads the content of Temperature1.txt 4 times and the content of Temperature2.txt 4 times.

2. (2 points) Consider the following graph and suppose g is its instantiation in GraphFrame.



Suppose the following command is executed on g:

motifs = g.find("(v1)-[]->(v2); (v2)-[]->(v3); !(v1)-[]->(v3)")

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

a) One of the rows stored into the Dataframe motifs is

v1	v2	v3
[u5, Esther, 32]	[u4, David, 29]	[u1, Alice, 34]

b) One of the rows stored into the Dataframe motifs is

v1	v2	v3
[u6, Fanny, 36]	[u3, Charlie, 30]	[u7, Gabby, 60]

c) One of the rows stored into the Dataframe motifs is

v1	v2	v3
[u2, Bob, 36]	[u3, Charlie, 30]	[u2, Bob, 36]

d) One of the rows stored into the Dataframe motifs is

v1	v2	v3
[u3, Charlie, 30]	[u2, Bob, 36]	[u3, Charlie, 30]

Part II

PoliBikes is an international company characterized by several production plants around the world. In each production plant, robots are used to produce some components of the PoliBikes' bicycles. PoliBikes computes a set of statistics about its production plants and robots. The analyses are performed by considering the following input data sets/files.

- ProdPlants.txt
  - ProdPlants.txt is a text file containing the list of production plants of PoliBikes. Each line of ProductionPlants.txt is associated with one production plant. The number of production plants is more than 1000. In some cities, there is more than one production plant.
  - Each line of ProductionPlants.txt has the following format
    - PlantID,City,Country

      where *PlantID* is the production plant identifier while *City* and Country are the city and the country in which the production plant is located, respectively.

    - For example, the following line

      PID1,Turin,Italy

      means that the production plant ***PID1*** is located in ***Turin***, ***Italy***

- Robots.txt
  - Robots.txt is a text file containing the list of robots managed by PoliBikes. One line for each robot of PoliBikes is stored in Robots.txt. The number of managed robots is more than 15000.
  - Each line of Robots.txt has the following format
    - RID,PlantID,IP

      where *RID* is the robot identifier, *PlantID* is the identifier of the production plant in which the robot is installed, and *IP* is its IP address. Each robot is used in one single production plant.
    - For example, the following line

      R15,PID1,130.192.20.21

      means that robot ***R15*** is installed in the production plant ***PID1*** and the IP address associated with that robot is ***130.192.20.21***.

- OutOfOrders.txt
  - OutOfOrders.txt contains the historical information about the dates on which each robot was out of order. A new line is inserted in OutOfOrders.txt every time a robot is out of order. OutOfOrders.txt contains historical data about the last 20 years.

- Each line of OutOfOrders.txt has the following format
  - RID,Date,Cause
    where *RID* is the identifier of the robot that was out of order, *Date* is the date on which RID was out of order, and *Cause* is the reason why RID was out of order.
    Note. There is at most one line for each pair (RID,Date)
  - For example, the following line

    R15,2020/05/01,BrokenMotherboard

    means that robot ***R15*** was out of order on ***May 1, 2020***, due to a ***broken motherboard***.

Exercise 1 – MapReduce and Hadoop (8 points)

Exercise 1.1

The managers of PoliBikes are interested in performing some analyses about the date with the highest number of out-of-service robots due to broken motherboard problems.

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

1. *First date with the highest number of out-of-service robots due to broken motherboard problems.* This application considers only the lines of OutOfOrders.txt associated with a broken motherboard problem (i.e., Cause='BrokenMotherboard') and selects the date associated with the highest number of out-of-service robots associated with a broken motherboard problem. In case of a tie, the first date associated with the highest number of out-of-service robots is selected. Store in the output HDFS folder the selected date and the number of out-of-service robots associated with a broken motherboard problem on that date (i.e., Date\tNumber of out-of-service robots associated with a broken motherboard problem in that date).

Suppose that the input is OutOfOrders.txt and has been already set. Suppose that also the name of the output folder has been already set.

- **Write your code on your papers.**
- 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 (**Exercises 1.2 and 1.3)** 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

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 PoliBikes asked you to develop one single application to address all the analyses they are interested in. The application has five arguments: the input files ProdPlants.txt, Robots.txt, and OutOfOrders.txt, and two output folders "outPart1/" and "outPart2/", which are associated with the outputs of the following points 1 and 2, respectively. Specifically, design a single application, based on Spark RDDs or Spark DataFrames, and write the corresponding Python code, to address the following points:

1. *Robots with an increasing number of out-of-order events considering the years 2020 and 2021.* This first part of the application considers only the robots which were out of order on at least one date in the period from January 1, 2020, to December 31, 2021, and selects the robots with a number of distinct out-of-order dates in the year 2021 greater than the number of distinct out-of-order dates in the year 2020. Store the identifier and the city of the selected Robots in the first HDFS output folder (one pair (RID,City) per output line).

2. *Window(s) with the maximum number of out-of-order robots for each production plant.* For each production plant, this second part of the application selects the window(s) of three consecutive days (dates) with the maximum number of distinct robots that were out of order in at least one of the dates of the selected window. For each of the selected windows, store in the second HDFS output folder (one window per output line) the first date of the window, the identifier of the production plant (PlantID), and the number of distinct robots that were out of order in that window in that production plant.

   Hints for Point 2

   - Consider only the windows associated with at least one out-of-order event.
   - In case of a tie (i.e., more than one window associated with the maximum number of out-of-order robots for the same production plant), all the windows associated with the maximum value for the same production plant are stored in the second output folder.
   - The number of distinct robots installed in each production plant is small enough to be stored in a python list if needed.

- Suppose the function **previousDate(date, N)** is provided. Given a date in the format 'YYYY/MM/DD' and an integer number N, **previousDate(date, N)** returns date-N (again in the format 'YYYY/MM/DD).
  For instance, previousDate('2018/04/05', 2) returns the date '2018/04/03'.

Example Point 2

- *Toy example.* For the sake of simplicity, suppose that there are only two production plants: PID1 and PID2. Suppose that there are three robots in the production plant with PlantId PID1 (RIDs: R1, R2, and R3) and two robots in the production plant with PlantId PID2 (RIDs: R4 and R5).
  Suppose that, in this small running example, we consider only the data related to the time period from May 1, 2010, to May 5, 2010 and suppose the five robots were out of order as reported in the following table in that period. Each cell in the following table is associated with a pair (RID, date) and contains 1 if the robot in the row is out of order on the date in the column. 0 otherwise.

		Date				
PlantID	**RID**	**2010/05/01**	**2010/05/02**	**2010/05/03**	**2010/05/04**	**2010/05/05**
PID1	**R1**	0	1	0	0	0
	R2	0	0	1	1	0
	R3	0	0	0	1	1
PID2	**R4**	0	0	1	0	0
	R5	1	0	0	1	0

- The selected windows are as follow for this running example:
  - 2010/05/02,PID1,3
  - 2010/05/01,PID2,2
  - 2010/05/02,PID2,2
  - 2010/05/03,PID2,2

- **Write your code on your papers.**
- You do not need to write imports. Focus on the content of the main method.
- Suppose both **SparkContext sc** and **SparkSession ss** have been already set.
- Suppose the following variables have been already set:
  - prodPlantPath= ProdPlants.txt', robotsPath= Robots.txt', outOfOrderPath= OutOfOrders.txt', output1='outPart1/', output2='outPart2/'