Distributed architectures for big data processing and analytics

June 26, 2023

Student ID _____

First Name _____

Last Name _____

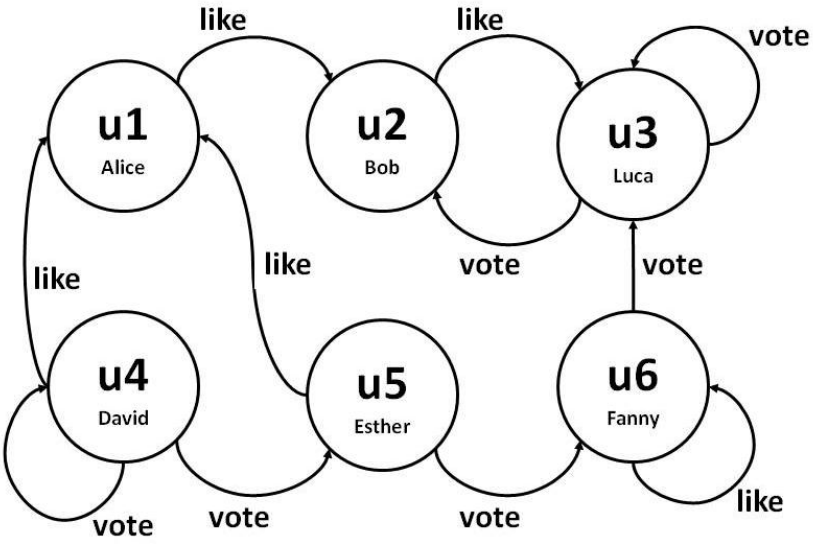
The exam lasts **2 hours**

Part I

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

1. (2 points) Consider the following graph and suppose g is its instantiation in GraphFrame.

   - Schema of the vertexes: ["id", "name"]
   - Schema of the edges: ["src", "dst", "relationship"]



Suppose the following commands are executed on g:

```
gfiltered=g.filterEdges(" relationship='vote' ")

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

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

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

v1	v2	v3
[u3, Luca]	[u2, Bob]	[u1, Alice]

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

v1	v2	v3
[u6, Fanny]	[u3, Luca]	[u3, Luca]

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

v1	v2	v3
[u5, Esther]	[u6, Fanny]	[u6, Fanny]

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

v1	v2	v3
[u3, Luca]	[u2, Bob]	[u3, Luca]

2. (2 points) Consider the input HDFS folder myFolder that contains the following three files:
   - ProfilesItaly.txt
     - The text file ProfilesItaly.txt contains the following four lines:
       Luca,Rome
       Luca,Rome
       Carmen,Naples
       Luca,Turin
   - ProfilesFrance.txt
     - The text file ProfilesFrance.txt contains the following two lines:
       Danilo,Paris
       Carmen,Paris
   - ProfilesSpain.txt
     - The text file ProfilesSpain.txt contains the following two lines:
       Carmen,Barcelona
       Pablo,Barcelona

Suppose you are using a Hadoop cluster that can run up to 5 instances of the mapper class in parallel. Suppose the HDFS block size is 1024MB. Suppose to execute a MapReduce application for Hadoop that analyzes the content of myFolder. Suppose the

map phase emits, overall, the following key-value pairs (the key part is a name while the value part is always 1):

(Luca, 1)
(Luca, 1)
(Carmen, 1)
(Luca,1)
(Danilo,1)
(Carmen,1)
(Carmen,1)
(Pablo,1)

Suppose the number of instances of the reducer class is set to 3 and suppose the reduce method of the reducer class sums the values associated with each key and emits one pair (name, sum values) for each key for which the sum is greater than 2. Specifically, suppose the following pairs are overall emitted by the reduce phase:

(Luca, 3)
(Carmen, 3)

Considering all the reducer class instances, how many times is the **reduce method** invoked?

a) 2
b) 3
c) 4
d) 8

Part II

PoliMeeting is an international company that manages online business meetings around the world. Statistics about the organized meetings and users are computed based on the following input data files, which have been collected in the company's latest 10 years of activity.

- Customers.txt
  - Customers.txt is a textual file containing information about the customers of PoliMeeting. There is one line for each customer and the total number of customers is greater than 200,000,000. This file is large and you cannot suppose the content of Customers.txt can be stored in one in-memory Java/Python variable.
  - Each line of Customers.txt has the following format
    - CID,Name,Surname, PricingPlan

where *CID* is the customer's unique identifier, *Name* and *Surname* are his/her name and surname, respectively, and *PricingPlan* is the type of pricing plan (e.g., free, business, premium) he/she subscribed to.

- For example, the following line

  Cust1000,Mario,Rossi,Business

  means that the name and surname of the user with identifier ***Cust1000*** are ***Mario*** and ***Rossi***, respectively, and the customer has subscribed to a ***Business*** pricing plan.

- Meetings.txt
  - Meetings.txt is a textual file containing information about the meetings managed by PoliMeeting. There is one line for each meeting. The total number of meetings stored in Meetings.txt is greater than 1,000,000,000. This file is large and you cannot suppose the content of Meetings.txt can be stored in one in-memory Java/Python variable.
  - Each line of Meetings.txt has the following format
    - MID,Title,StartTime,Duration,OrganizerCID

      where *MID* is the meeting's unique identifier, *Title* is the title of the meeting, *StartTime* is the start time of the meeting, *Duration* is its duration in minutes, and *OrganizerCID* is the identifier of the customer who organized the meeting. StartTime is a timestamp in the format YYYY/MM/DD-HH:MM. *Duration* is an integer.
    - For example, the following line

      MID1034,Polito project kick-off,2023/02/07-20:40,60,Cust1000

      means that the meeting with MID ***MID1034*** was organized by ***Cust1000*** and is titled ***"Polito project kick-off"***. It is scheduled for ***July 2, 2023***, at ***20:40*** and lasts ***60*** minutes.

- Invitations.txt
  - Invitations.txt is a textual file containing information about invitations to meetings. A new line is inserted in Invitations.txt every time someone is invited to a meeting. Invitations.txt includes the historical data about the latest 10 years. This file is big and you cannot suppose the content of Invitations.txt can be stored in one in-memory Java/Python variable.
  - Each line of Invitations.txt has the following format
    - MID,CID,Accepted

      where *MID* is the identifier of the meeting to which customer *CID* has been invited. *Accepted* can assume two values: "Yes" (when the customer accepted the invitation) or "No" (when the customer did not accept the invitation).

    - For example, the following line

      MID1034,Cust23,Yes

means that ***Cust23*** has been invited to the meeting ***MID1034***, and he/she has accepted to participate.

Note that the same customer can be invited to many meetings, and each meeting can have many invited Customers. However, each combination (MID, CID) occurs at most one time in Invitations.txt.

- Participations.txt
  - Participations.txt is a textual file containing information about who participated in the organized meetings. A new line is inserted in Participations.txt every time someone joins (participates in) a meeting. Participations.txt includes the historical data about the latest 10 years. This file is big and you cannot suppose the content of Participations.txt can be stored in one in-memory Java/Python variable.
  - Each line of Participations.txt has the following format
    - MID,CID,JoinTimestamp,LeaveTimestamp
      where MID is the identifier of the meeting that customer *CID* joined at *JoinTimestamp*. *LeaveTimestamp* is the timestamp at which *CID* left the meeting *MID*. The format of the timestamps *JoinTimestamp* and *LeaveTimestamp* is YYYY/MM/DD-HH:MM:SS.

    - For example, the following line
      MID1034,Cust23,2023/02/07-20:40:10, 2023/02/07-20:50:02

      means that ***Cust23*** joined the meeting ***MID1034*** on ***July 2, 2023***, at ***20:40:10*** and left on ***July 2, 2023***, at ***20:50:02***.

    Note that the same customer can participate in many meetings, and each meeting can have many participants. Moreover, **the same customer can join and leave each meeting several times** (a new line associated with a different JoinTimestamp is inserted every time a customer joins or rejoins the same meeting). Each triplet (MID, CID, JoinTimestamp) occurs at most one time in Participations.txt.

Exercise 1 – MapReduce and Hadoop (8 points)

Exercise 1.1

The managers of PoliMeeting are interested in performing some statistics.

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

1. *Organizer with the highest average meeting duration.* For each customer, compute the average duration of the meetings he/she organized and select the identifier (OrganizerCID) of the customer with the highest average meeting duration. If many customers are associated with the highest average meeting duration, select the one with the first OrganizerCID according to the alphabetical order. Store the OrganizerCID of the selected customer in the HDFS output folder.

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

- 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
    - do not report the get and set methods. Suppose they are "automatically defined"

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 PoliMeeting asked you to develop one single application to address all the analyses they are interested in. The application has six arguments: the input files Customers.txt, Meetings.txt, Invitations.txt, and Participations.txt, and two output folders "outPart1/" and "outPart2/", which are associated with the outputs of 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 two points:

1. *Pricing plans with a high percentage of long meetings.* For each pricing plan, compute the percentage of *long* meetings its subscribers organized. A meeting is classified as *long* if it lasts more than 60 minutes (duration>60 minutes). Select only the pricing plans with at least 90% of long meetings. Store the pricing plans with at least 90% of long meetings in the first HDFS output folder (one pricing plan per output line).

2. *The customer(s) with the most "mismatches".* The second part of this application considers only the meetings scheduled before January 1, 2023 (StartTime<*2023/01/01-00:00*). Moreover, this second part considers only the customers invited to at least one meeting scheduled before January 1, 2023 (i.e., the customers occurring in Invitations.txt associated with at least one meeting scheduled before January 1, 2023). For each of these customers, compute (i) how many times he/she accepted an invitation for a meeting (Accept="Yes") but then he/she did not participate in that meeting and (ii) how many times he/she declined an invitation for a meeting (Accept="No") but then he/she participated in that meeting. Store in the second HFDS output folder the customer(s) with the highest number of mismatches. The number of mismatches for each customer is the sum of the two values computed before ((i)+(ii)). The format of the output lines is

   CID, the number of times CID accepted an invitation but then did not participate in the meeting, the number of times CID declined an invitation but then participated in the meeting

   If multiple customers are associated with the highest number of mismatches, store all these customers in the second output folder (one selected customer per output line).

Example Part 2

For the sake of simplicity, suppose there are only the following four customers:

- *Cust1*
- *Cust2*
- *Cust3*
- *Cust4*

The following table reports the statistics about the acceptance/decline of the invitations and participations of each customer in the meetings scheduled before January 1, 2023.

CID	Number of invitations	Accepted but did not participate	Declined but participated	Accepted and participated	Declined and did not participate	Number of mismatches
Cust1	10	3	2	4	1	5
Cust2	30	3	0	15	12	3
Cust3	15	0	5	9	1	5
Cust4	10	2	1	7	0	3

Given this small example, the second part of this application selects *Cust1* and *Cust3* and stores in the second output folder the following two lines:

Cust1, 3, 2
Cust3, 0, 5

Remember that there are more than 200,000,000 customers and more than 1,000,000,000 meetings in the actual files.

- You do not need to report the imports.
- Suppose both SparkContext sc and SparkSession ss have already been set.