On identifying the compensation for vaguely followed trips

Data Intensive Systems Course, Utrecht University, 2023-2024

Instructor: Prof. Yannis Velegrakis

DEADLINE: June 30th, 2024 **Number of Persons: max 3**

Situation Description:

You have been hired to work for a network provider. The company operates several servers. A server executes a **service task**. A service task is a task that involves some local processing by the server, but the server may also ask some other servers to execute some part of its task. This is known as **subtask**. For instance, a service to purchase a book may require an authentication service (i.e., a request for an id check and the receipt of a response), followed by a credit card check and a response. To execute a service task the server receives a service request by a user or by another server and returns to the requestor a service response.

The moment a response is provided by the server to the requestor is always after the moment the service request was made. The duration of a service task is the difference between the moment a request has been made to the server and the time a response has returned. We do not consider network delays. This means that the moment a request is sent, the receiving server receives it immediately, and when a response is sent back to the originating requestor, the requestor receives it at the same exact moment. Due to this, we can model a request or a response by listing the sender and the recipient of the request alongside the time the request/response was made. For example: *< S0, S1, t0, Request, 1821 >* or *< S1, S0, t6, Response, 1821 >* Requests that are originally made by the user, can be modeled by considering the origin to be null. For instance, < null, S0, t10, Request, 1821 >. The number 1821 is a unique id assigned to the original request and is used to link together different requests (will be explained below).

The duration of a service task is the amount of time from the moment request is received until the moment a response is sent back. It includes the time that the server needs to process the request locally plus any additional time needed for the execution of any possible subtasks. Thus, the duration can be computed by subtracting from the time a response is sent the time the request for the service task was received.

Consider the following example. Server S1 receives from a user a request to buy a book X at time t0. This request is given the process id number 1821. Server S1 realizes that it needs to do a credit check, so at time t1 it makes the requests for credit card check to the credit card service provider which is the server S2 (clearly t1>t0). While waiting for server S2 to respond, the server S1 at time t3 (where t3>t1) receives another request from a user to buy a book Y, which is given the process id 1978. To do this process, server S1 sends a request to the id checking server S3 at time t4>t3. Then while waiting for an answer from server S3, at time t5, with t5>t4, the response from server S2 is received, in which case the response to the purchase of the book X is sent back to the requestor at time t6 (with t6>t5). At some point in time t7, which is after t6, the response from the server S3 is also received. To deal with this multiplexing of services, and be able to know for every service task why it is made, the task id of the original request made by the user is carried alongside the requests and the responses. This is called the **process identifier**. All the service tasks that have the same identifier constitute a process.

When a request or a response is sent between two servers, this information is recorded by been encoded into a record. The record has the form:

<FromServer, ToServer, time, action (**Request** or **Response**), processId >

These records are stored in a log file in a server where we have access. For example, the log entries in the log file for the scenario described above is the following:

< null, S1, 0, Request, 1821 >

< S1, S2, 1, Request, 1821 >

< null, S1, 3, Request, 1978 >

< S1, S3, 46, Request, 1978 >

< S2, S1, 51, Response, 1821 >

< S1, null, 62, Response, 1821 >

< S3, S1, 71, Response, 1978 >

< S1, null, 88, Response, 1978>

Its server is responsible for one specific task. In the example above for instance, server S1 is responsible for buying a book, S2 for doing the credit card check and S3 for doing the id Check. Normally one may have expected that a server takes exactly the same time to perform the task it is designed to do. This is not the case. The time for the tasks may experience big variations because the server may call different subtasks. For instance, in the example above, the server S1 performs a book purchase in both times but in one case it performs an id check while another time it performs a credit card check. Even in the case that a task is implemented by executed exactly the same sub-tasks, for instance imagine the purchase of a book done by a creditcard check as happened in the process 1821 above, it may be the case that there are small variations. This is because the server may experience different loading time (i.e., CPU being busy with other tasks). Another form of small variations is a situation where you have 2 or more servers that perform similar task, for instance, there may be creditCardCheckVISA and creditCardCheckMastercard. Those two services do very similar tasks. We know they are similar because the names are similar.

Goal

The log file is essential a description of processes that have taken place (described as a sequence of Requests and Responses). The task we need to do is two-fold.

1. First we need to identify different small variations of the a processes recorded in the log file and when we identify processes that are similar, replace them with one specific representation (it is up to you to find how to identify them and what that unique representation will be. The specific representation is basically a process that somehow can be a good representer of those processes that are getting eliminated.
2. Identify groups of processes that look similar (but are not very similar to be considered the same as it happened in the #1 above.

We assume that you are given the log file, and you are expected to create a framework to achieve the two points above. You will need to provide a very detailed description of your approach on how you solve the problem, develop a program that implements your idea as you described it, and perform all the necessary evaluation steps that indicate the effectiveness and efficiency of the framework.

Programming language

This is left totally to you.

Number of persons

The project is for 3 persons max.

Input Dataset

There is no specific data that is given to you. But you can create one (this is called synthetic Dataset Generation) in a way that you test your program. You will need to device a program that generates the necessary datasets that will allow you to stress test your work. You need to create a large dataset (do not create toy examples by hand. The toy examples will be only when you develop the code, but then when you need to evaluate the program you need serious examples.).

Output - Visualization

To Be defined

Delivery

(Draft) You need to deliver the code of the program you developed, the dataset you used, instructions on how the program runs and a report in which you describe the solution you have devised and the results of the experiments you have performed to prove the effectiveness and efficiency of your solution. To do that, you need to create a folder in your one-drive and share it (read and write permissions) with the instructor (i.velegrakis@uu.nl) and **send the instructor a mail with the link.** (Note that often just sharing is not enough so you need to also send the link). The

folder should be called DIS22_XX_YY_ZZ, where XX, YY, and ZZ are the last names of the participants in the project. It should contain:

1. A pdf document called **UUUU.pdf** where UUUU is the name of your team and should be structured as described in the section "Report Structure" below.
2. A directory called **src** in which you will place the code of your program. Include a README.txt file with instructions on how the program runs.
3. A directory called **data** in which you will place the data you have used in your experiments.
4. A directory called **results** in which you put the output of the runs of your program. The format of the file is up to you. The name of each file should start with the name of the dataset file that was used. For example, if the input file is movies.csv then the output file should be called movies_XXXXX where the XXXXXX is any informative string of your choice.

Presentation

On the last week of the lectures (see the lecture schedule for the exact day) there will be a presentation in which a representative of every team will make a 5 min (maximum) presentation of the solution the group is developing. At least one slide is required for each of the following topics.

1. Project/group name and members (**with photos of the members** so that we know who is with whom)
2. Solution
3. Datasets + Experiments (to be done or have been done)

The presentation should be in line with the final project that will be delivered. Changes are of course allowed but one cannot have a dramatically new solution.

Evaluation Criteria

The project is evaluated according to the following evaluation criteria:

1. Novelty & sophistication of the idea as well as how well it solves the problem.
2. Technical Depth: Detailed description of the approach and its challenging choices. How well data intensive technologies (Spark, Map-Reduce, NoSQL) have been exploited.
3. Presentation: Clarity and Completeness of the report & the presentation
4. Experimental Evaluation
   4.1. The dataset(s) that have been used in the evaluation
   4.2. The evaluation tests that have been made (what has been tested and how)
   4.3. The comments on the results of the evaluation

Structure of the report

The final report should be written in Latex, using the following template which is available on github: http://velgias.github.io/tmp/template.zip. It should contain the following sections:

1. **Introduction** (maximum 1 page) in which you introduce the problem you are solving, its importance and the main highlights of your solution (1 paragraph) and the results of your experiments (1 paragraph). Provide a motivation for this work. (Why you think that such a study is important? (you already have an application so it is clear that is important, but maybe you can think additional applications to make the statement stronger). And why it is challenging (i.e., not trivial) to perform this processing? What were the hard/challenging parts in developing a solution?) Note that a "hard/challenging" part should be generic and not personal to the authors. They should apply to everyone and are challenging due to the nature of the problem at hand. They should not be challenging just because of the capabilities of the author. For example, if the solution is developed in python and the programmer does not know python, then clearly the difficult is only for the specific author and not for everyone.
2. **Related work** and technologies (maximum 1 page): Any information you think is important for the reader to know but IS NOT your own work. For example, you could describe there what Spark is, what map reduce is,

etc. Do not waste space getting into details that everyone else knows already from the lectures or other online sources. Keep it to the basic and to the minimum.

3. **Solution**. In this section you describe in detail what your solution is (or what your solutions are, in case of more than one). The more detailed you are in this section the better the section is. Imagine that you give your report to someone else, and you ask her/him to implement your solution. Will that person be able to do it by looking only at what is written in the document? If yes, then the document is successful. Also explain the reasoning behind every choice you are making. You are free to include some pseudocode because it makes it much easier for people to understand what the text is saying.
4. **Experimental evaluation**. This section contains a detailed description of all the experiments you have done to understand how well your solution works. How does it compare to some baseline? The more things you are testing, the more it helps to understand the performance of the solution, and the better the report is. Since the company refuses to share its datasets, the experiments will be on synthetic data. The size of the section is up to you, since it depends on the complexity of the solution you are proposing and the details you would like to study. Make sure that you also provide a description of the datasets you used as input.
5. **Conclusion**. A recap of what you did in your work (the main highlights). Maximum half a page.