Multi-Threading Programming and Inter-Process Communication

M.Sc. course on "Technologies for Big Data Analysis" - Assignment 1

CHRISTOS BALAKTSIS (1234) and VASILEIOS PAPASTERGIOS (1234), Aristotle University, Greece

1 Introduction

The current document is a technical report for the first programming assignment in the M.Sc. course on *Technologies for Big Data Analysis*, offered by the *DWS M.Sc Program*[1] of the Aristotle University of Thessaloniki, Greece. The course is taught by Professor Apostolos Papadopoulos [2]. The authors attended the course during their first year of Ph.D. studies at the Institution.

The assignment contains 4 sub-problems and is part of a series, comprising 3 programming assignments on the following topics:

Assignment 1 Multi-threading Programming and Inter-Process Communication
Assignment 2 The Map-Reduce Programming Paradigm
Assignment 3 Big Data Analytics with Scala and Apache Spark

In this document we focus on Assignment 1 and its 4 sub-problems. We refer to them as *problems* in the rest of the document for simplicity. The source code of our solution has been made available at `https://github.com/Bilpapster/big-data-playground`.

Roadmap. The rest of our work is structured as follows. We devote one section for each one of the 4 problems. That means problems 1, 2, 3 and 4 are presented in sections 2, 3, 4 and 5 respectively. For each problem, we first provide the problem statement, as given by the assignment. Next, we thoroughly present the reasoning and/or methodology we have adopted to approach the problem and devise a solution. Wherever applicable, we also provide insights about the source code implementation we have developed. For problems 2 and 4, we complete the respective sections with a discussion about alternatives or improvements the solution could accept, in order to successfully support more complex requirements. Finally, we conclude our work in section 6.

2 Problem 1: Concurrent Array-Vector Multiplication

We discuss here the first problem of the assignment. The main target of the assignment is using multi-threading programming in Java programming language to concurrently perform an algebraic operation.

2.1 Problem Statement

It is known that in Linear Algebra we can multiply a matrix with a vector from the right-hand side, provided that the number of columns in the matrix equals the number of rows in the vector. For instance, given a matrix $\mathbf{A}$ with dimensions $n \times m$ and a vector v with dimensions $m \times 1$, then the product $\mathbf{A} * \mathbf{v}$ is an $n \times 1$ vector, which results from the implementation of the well-known method of multiplying a matrix with a vector. An example is given following: **TODO**

[1] https://dws.csd.auth.gr/
[2] https://datalab-old.csd.auth.gr/~apostol/

Authors' Contact Information: Christos Balaktsis (1234), balaktsis@csd.auth.gr; Vasileios Papastergios (1234), papster@csd.auth.gr, Aristotle University, Thessaloniki, Greece.

Provided that we are capable of using k threads, where k is a power of 2 and the matrix has dimensions $n \times m$, where n is also a power of 2 and $n > k$, design a solution that computes the product $\mathbf{A} * \mathbf{v}$ using k threads with the most efficient way. Your solution has to initialize both the matrix $\mathbf{A}$ and the vector $\mathbf{v}$ with random numbers in the range $[0, 10]$.

3 Problem 2: Race against a pandemic

The second problem asks for leveraging multi-threading programming in Java programming language, in order to simulate a pandemic spread.

3.1 Problem Statement

In this problem, you have to simulate a (simplified) pandemic spread, taking into account new infections, hospitalizations and recoveries occurring concurrently. In particular, suppose that a thread named `DISEASE` produces periodically (e.g., every 1 second) a random number of new infected people in the range $(0, k]$ that are in need of hospitalization in ICU[3]. The Health Care System has a limited number of ICU beds, let e (e.g., 20). At the same time, a thread named `HOSPITAL` periodically (e.g., every 5 seconds) treats a random number of infected patients in the range $(0, h]$, where $h < k$. When a patient is treated the ICU bed used for their treatment is no longer occupied; thus available for use by another patient.

Develop a solution that simulates the above behavior. Your solution should also keep track of the total number of treated patients, as well as the ones that were not able to find a spot in ICU. The simulation must complete its execution after a predefined number of steps (or time period), which will be given as program argument, alongside with all aforementioned parameters (periods, k, h, e). You should test your solution against different values of the parameters, in order to verify that it correctly operates in all possible cases. How would you convert your solution if, instead of a single hospital, there were three of them, with a single, shared queue of patients?

4 Problem 3: Key-value server store

We present here the third problem of the assignment. The main focus of the problem is the inter-process communication, leveraging TCP sockets in Java programming language.

4.1 Problem Statement

Develop a client-server application that works as follows.

Server. The server starts its operation by initializing an empty Hash Table. The Hash Table stores key-value pairs and has size of 2^{20}. Next, the server opens a TCP port, the number of which is passed as a program argument (e.g., 8888, 9999, etc.). The server awaits for clients to connect.

Client. The client starts its operation by trying to connect with the server, in the specified TCP port. Again, the port is passed as a program argument.

Communication protocol. After the successful connection between server and client, the client can send messages to the server, requesting some action from a predefined list of available actions. In particular, the client can perform the following operations:

Operation 0 Terminates the connection between client and server. Operation code: `0`.

Operation 1 Inserts a key-value pair in the Hash Table stored in the server-side. Operation code: (`1, K, V`), where `K` and `V` is the key and value to be inserted respectively.

Operation 2 Deletes a key-value pair from the Hash Table kept in the server-side. Operation code: (`2, K`), where `K` is the key to be deleted, jointly with the stored value that is associated with it.

[3] ICU: Intensive Care Unit

Operation 3 Searches for a key and retrieves its value (if exists). Operation code: (`3, K`), where `K` is the key to search for.

5 Problem 4: Multi-server producer-consumer interaction

This is the section for the fourth problem.

5.1 Problem Statement

6 Conclusion

This is the section for the conclusion.