Sami Farhat

Amine Bouras

STUDENT ID: 40097236

STUDENT ID: 40128006

**LAB 1**

**Merge Sort using multithreading**

Operating systems

COEN 346

Section YJ-X

Winter 2021

Concordia University

12th February 2021

**Theory & Objectives**

A Program under execution is called a process. However, if we are looking at a basic unit of execution, we would call that a thread. A program might have multiple process and a process might have multiple threads. The use of multithreading is to compute in parallel efficiently multiple tasks at the same time. This would allow a program to be executed way faster than it should be . In theory, a thread will not wait for another one to finish, all the threads will basically execute at the same time. However , thread A and B could execute at the same time and we could arrange thread C to wait for both to finish. If we were to give an example, if we executed a code that would take 1 second to execute 10 times, the whole code would take around 10 seconds to execute. However, if we implement the code using thread, all the iterations of that code would run at the same time, making the execution time 1 second. As an other example, if we were to use the multithreading principle to code a process that would calculate a really complex equation, it could be done more efficiently since we could make every thread do a separate task instead of a singular thread doing all the work in sequence. If were to look for an example a little more complicated, we could look at how an algorithm that analyse images use multithreading . This algorithm will separate that image in different parts and apply filtering to the image by parts, every thread applying to a specific part.

In this lab, the objective is to familiarize ourselves with the concept of multithreading and to familiarize ourselves with how to use that concept in codes and how to control those threads. It is required to code a merge sort to sort efficiently the values given.

**Task & Procedure and Results**

In order to fulfill this laboratory , we needed to implement a merge sort using multithreading. In short, a merge sort will separate an array in two halves recursively , and then compare every value between those two halves to sort them in a resulting array. A merge sort is efficient because it will always perform in a log N time, however it requires a huge amount of space since we must copy everything back to a result array. We have coded this sort with python using a sorting function and halving function. The latter will , using threads , recursively halves the function . Then the first, will ,as long as the index is lower than the length of the array, compare the two indexes in the halves and then append it to the result array. We have also added a global counter in order to count the number of threads that are being declared, with every one of them outputting a start message and an ending message. We have used the ThreadPoolExecutor, which is used to execute a task using pooled threads and which is also useful when it comes to controlling threads. So , in short, the threads will halve the arrays until they can be compared efficiently and then be appended in a result array .

Our experience with threading has been interesting. We have learned a new way to code and a new way to implement our code in a manner that is faster. The fact that a thread will automatically start even though the previous one hasn’t finished yet is a game changer. It was hard to begin with since it was a new concept for both of us. It took a lot of tries to understand how to call the functions and how to keep track of the threads.

**Conclusion**

In conclusion, this lab was a success since we were able to achieve all the manipulations asked and we got to familiarise ourselves with the different ways threads are used. The output that we got is similar in form and has the same answers as the one provided as a solution. We were able to understand the theory behind threading and to see its application in a real-world context.