CCT College Dublin

Module Title:	Concurrent Systems
Assessment Title:	Application of Concurrency to Common Tasks
Lecturer Name:	Sam Weiss
Student Full Name/ID:	Heber Junio dos Anjos Mota – 2020317
Assessment Due Date:	08/04/2023
Date of Submission:	08/04/2023

Declaration

By submitting this assessment, I confirm that I have read the CCT policy on Academic Misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source. I declare it to be my own work and that all material from third parties has been appropriately referenced. I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution.

Application of Concurrency to Common Tasks

Tradeoffs For Each Task

In this project we were encouraged to develop a program that could calculate different types of operations via a *data.csv* file previously provided. Furthermore, we should also be able to make the program make these operations in concurrency with which other by using the threads methodology in order to better understand its functioning. Below, the observations made in regards to each task operated – although, overall – they present very similar traits:

- **Standard Deviation:** The standard deviation is a metric for the set's dispersion, or how uniformly distributed the data is within the set. The standard deviation measures the variation of the values from the set's mean; the closer it is to zero, the less dispersed the set's data are. By using concurrency programming to make these calculations, we are able to get a better optimized processing of values, consequently contributing speed and performance especially when it comes to bigger sets of data. Given that, it demands more complexity of coding and attention to detail once they will be working closely together.

- **Merge Sort:** Merge Sort divides the array into two subarrays, each containing half of the original array's items. Then, this process is repeatedly done to the two subarrays until the final picture is assembled. Using threads for this purpose can also be extremely beneficial to potentialize speed of this process from data gathered through the CSV file. Again, the managements of such tasks need to be done thoroughly, choosing enough threads to assure perfect functioning and avoid errors that could appear.

- **Matrix Multiplication:** In the Matrix Multiplication, the components that integrate the lines of the first matrix and entries of the second matrix are used to multiply two or more matrices since each matrix is made up of a collection of elements or entries grouped in lines and columns. In this case, we divided a set of 200 values from the CSV file into two halves and multiplied them to obtain their respective results combined. The code complexity for this calculation is bigger and therefore it is vital that the threads are working in accordance to manage the computation properly. When working with bigger sets, the threads would be perfect to handle such complexity and manage CPU and memory resources. A challenge could be to determine the number of threads to act on the workload once too many threads could slow down the process rather than speed it up. Error mitigation and complexity are other factors to consider.

Overall Tradeoffs Single & Concurrent Program For Reading Data From **.csv** File

A Concurrent Program consists of a set of instructions that are executed sequentially, and the execution of these instructions is called a threads. This type of program specifies two or more sequential threads that can be run concurrently as parallel processes. Speaking of Concurrent programming as a whole, it is important to point the benefits and advantages of using this technology, which is listed below:

Benefits

Optimized Performance & Resource Utilization: By using threads to run different tasks simultaneously and taking advantage of multiple processors or cores, we can obtain better memory and disk utilization, for example: in a program that plays videos, a thread can process the image, another one can process sound, which makes the overall processing time shorter and efficient.

Enhance Scalability & Responsiviness: Concurrent programming makes it possible for Java programs to manage huge numbers of users or requests more effectively as for the processing of large data sets from a CSV file, or it can ensure that your application responds to user input by removing time-consuming operations from the primary UI thread – which increases the application's ability to scale and deliver results quicker.

Smart & Simplified Programming: Threads make programming easier – depending on the complexity of your program – by letting you create concurrent code without worrying about managing numerous processes all at once.

Disadvantages

Bigger Complexity & Risk of Bugs: Concurrent programming can be difficult to implement because it takes a lot of work on the part of the programmer to monitor and manage the changes or accesses that these threads make throughout the program. Conditional Synchronization is a widely used technique for thread control to tackle such issue. Multithreading, therefore, can be more complex to code, maintain and debug (dealing with race conditions, deadlocks and livelocks) than the single ones.

Memory Overhead: Java threads require some memory overhead, so by using too many of them – that access shared data or resources – a sychronization overhead could happen leading it to result in increased memory usage which negatively impacts performance.

Careful Designing & Planning: On the creation and execution of the threads, it can be hard to understand what's happening because things are happening at the same time and can interfere with each other, so careful planning is needed to prevent errors and ensure a correct operation and interaction of threads.

References

- Pankaj, P. (2022). Multithreading in Java - Everything You MUST Know. [online] digitalocean.com. Available at: https://www.digitalocean.com/community/tutorials/multithreading-in-java [Accessed 20 Mar. 2023].

- schools, w. (2021). Java Threads. [online] w3schools.com. Available at: https://www.w3schools.com/java/java_threads.asp [Accessed 20 Mar. 2023].

- Jones, B. (2018). Threads in Java, Debugging Threads in Eclipse. [online] youtube.com. Available at: https://www.youtube.com/watch?v=QyZvEfaXdzQ [Accessed 26 Mar. 2023].

- Tutor, J. (2021). read text file using threads java. [online] youtube.com. Available at: https://www.youtube.com/watch?v=UABwCn76Eeg [Accessed 20 Mar. 2023].

- Faria, C. (2011). Programação Concorrente: Vantagens e Desvantagens [Concurrent Programming: Advantages and Disadvantages]. WordPress. https://cassiofaria.files.wordpress.com/2011/06/programao-concorrente-vantangens-desvantagens.pdf [Accessed 29 Mar. 2023].

- Freedom, D. (2022). CALCULATE FOR STANDARD DEVIATION USING JAVA. [online] youtube.com. Available at: https://www.youtube.com/watch?v=DIjUx82Dlbc [Accessed 23 Mar. 2023].

- Data, D. (2021). Java Program to Calculate Standard Deviation. [online] youtube.com. Available at: https://www.youtube.com/watch?v=nWrO9S4_ZaM [Accessed 23 Mar. 2023].

- Concur-Distrib-Programming, C. (2021). Concur-Distrib-Programming. [online] github.com. Available at: https://github.com/deeppatel2899/Concur-Distrib-Programming/blob/main/Parallel_matrix_multiplication.java [Accessed 27 Mar. 2023].

- T. (2021). Matrix Multiplication in Java Practical. [online] youtube.com. Available at: https://www.youtube.com/watch?v=7kZDlUTct9k [Accessed 27 Mar. 2023].

- John, C. (2022). Merge Sort Algorithm in Java - Full Tutorial. [online] youtube.com. Available at: https://www.youtube.com/watch?v=bOk35XmHPKs&t=2s [Accessed 28 Mar. 2023].

- Baeldung. (2022). Merge Sort in Java. [online] baeldung.com. Available at: https://www.baeldung.com/java-merge-sort [Accessed 28 Mar. 2023].

- GPT, C. (2023, March 28). Advantages and Disadvantages of Concurrent Programming.. Chat. https://chat.openai.com/chat [Accessed 28 Mar. 2023].