COSC 3P93
Project – Step 2

Due date: October 10th, 2025 at 23:59 (11:59pm)
Delivery method: the student needs to delivery the project only through D2L (BrightSpace).
Delivery contents: document with answers and [Java, C, C++] codes if applicable (see Submission instructions).
Attention: check the Late Submission Policy.

Project Overview

This project is intended for students to apply design patterns and strategies for parallel systems in a sequential solution. The patterns and strategies applied in the project for the design and implementation are covered in classes and course reading materials. The project's final output is a performance comparison between a sequential program and its respective parallel design. For both programs, students will have to write the codes themselves and define performance analysis parameters and metrics for the comparative analysis. It is strongly recommended that the implementation language in this project be either C or C++. Most likely, students will need to familiarize themselves with Linux/Unix systems and bash command line to compile, debug, and run their code. For each project step, the students are expected to properly employ the concepts learned in class and report them in the document submitted together with their code.

Step 2 - Description

The second step of the project consists of implementing a sequential version of an algorithm in the assigned project topic (subtopic). More important than the implementation itself, this step requires a complete description of the algorithm (pseudocode) and a discussion of the possible performance improvement opportunities through a parallel construct of the same program.

Step 2 - Details

The group will need to identify an algorithm in the topic that was selected and assigned to it. The algorithm must be designed to run sequentially and be fully implemented in either C or C++. Students must note that designing and implementing a vanilla version of the chosen algorithm for solving a simple problem fulfils the requirements of this project step. However, the oversimplified implementation will render practically no parallel performance improvement in later project steps. **Thus, identifying a complex problem to apply the assigned algorithm(s) is crucial for this project overall.**

> ***For example***, a group has been assigned a shortest path algorithm, which has a redesign goal to create a parallel all-pairs shortest path. The group chooses to apply and implement the algorithm to a vehicular path recommendation problem. The group identifies or generates a trace of vehicle trips in an urban centre; the trace contains 100 source-destination trips on a $1km^2$ map. The complexity of the problem (number of trips plus length of each trip) is very low. Because of the little complexity of the problem, there are minimal design options to convert the sequential algorithm into parallel; also, the achieved parallel performance gain is almost negligible. The group will have difficulty applying and justifying parallel design patterns in its final project report, possibly standing on partial final project marks.

Please contact the instructor or course TA for feedback about project design and implementation decisions – to know if the complexity conforms with the project expectations.
A complete and detailed description of the algorithm, or pseudocode, is essential at this step. The description needs to emphasize the algorithmic aspects of the code instead of its general purpose/intent. For instance, students should not discuss the benefits of using a bio-inspired convolutional optimization method as a meta-heuristic for a Hard problem. Instead, the students should detail the parts of the code that may be considered a bottleneck in the code, may be implemented only sequentially or may lead to a future parallelization potential.
The description document should objectively describe the following:

A The design decisions and logic of the code in algorithmic aspects.

B An analysis of performance particularities: range of parameters and respective consequences to execution time.

C A discussion about bottlenecks in the code and areas/regions/blocks of code where parallelism can be applied.

Also, for implementing the code, the students should avoid using specialized libraries at any cost. In other words, students should fully implement the code themselves. The reason behind this approach is to avoid using any optimization (parallel execution optimizations) that such a specialized library may bring. Ultimately, in case the use of the library cannot be avoided, students will have to fully describe the functions and proprieties of the library used in their code. This description should include implementation details of the library (to show the hidden optimizations it may contain).

Marking Scheme

Marks will be awarded for completeness and demonstration of understanding of the material. It is vital that students fully show their knowledge when providing solutions concisely. Quality and conciseness of solutions are considered when awarding marks. Every code added to the originals should be well commented on and explicitly indicated in the Java files; lack of clarity may lead students to lose marks, so keep it simple and clear.

Submission Material

The submission for this project step consists of two parts:

5.1. The ***Description document (PDF)***. A description document must be provided, explaining implementation decisions, justifications, issues in the code, and challenges. The comments added in the code should be a good starting point for generating this document. Students should also describe how to compile and run their code as part of their explanation. There is no need to worry about the formatting of this document; students must follow the provided Latex template.
   - ***Latex template - a must for writing your project***. For writing the description document, use the Latex template enclosed in this project (update it accordingly!). Students do not need to install Latex software on their computers. They can write it through Overleaf on your browser (it is a free tool). Just upload the latex template to an Overleaf project; it should compile/render the tex file gracefully.

5.2. The **C/C++ code**. It consists of the C/C++ code of project implementation, which needs to be well organized with instructive, complete comments. The code should compile and should run properly. Please do not forget to provide instructions on compiling the project's code in the description document. As recommended, make the project compilable and runnable from the command line for marking purposes, so define all setup steps, such as environment variable setups, to make the code run in any "foreign" environment. The compilation and execution of the code should not rely on any IDE.
   - ***Compilation***. Provide the command for compiling the source code from the command line.
   - ***Running***. Provide the command for running the compiled code from the command line.

Submission

Submission is to be a PDF (description document), and text code files. All submissions should be performed electronically through D2L (BrightSpace).

Students must guarantee that their code is legible, clear, and succinct. Remember that any questionable implementation decisions or copying from any source might negatively impact marks. If students still have questions about which file types are acceptable, please inquire prior to submission. Note that it is not the markers' fault if they cannot mark an project due to submitting an unreasonable or uncommon file format.

The submission consists of two files and must submitted through D2L:

- ***Description Document in a PDF file***.
- ***Code in a Zip file***.

*** Do not forget to include the names and student IDs of group members.**
**** Only one student needs to submit the files on behalf of a group.**

Late Assignment Policy

There is no late submission policy. No late submission is allowed.

Plagiarism

Students are expected respect academic integrity and deliver evaluation materials that are only produced by themselves. Any copy of content, text or code, from other students, books, web, or any other source is not tolerated. Similarity checking tools (Turnitin) will be used on submitted materials. If there is any indication that an activity contains any part copied from any source, a case will be open and brought to a plagiarism committee's attention. In case plagiarism is determined, the activity will be cancelled, and the author(s) will be subject to the university regulations.
For further information on this sensitive subject, please refer to the document below:
`https://brocku.ca/academic-integrity/`