COSC 3P93
Project – Step 3

Due date: December 6th, 2022 at 23:59 (11:59pm)
Delivery method: the student needs to delivery the assignment only through Sakai.
Delivery contents: document with answers and [Java, C, C++] codes if applicable (see Submission instructions).
Attention: check the Late Assignment Policy.

Project Overview

This project is intended for students to apply design patterns, as well as strategies, for parallel systems in a sequential solution. The patterns and strategies applied in the project for the design and implementation are the ones covered in classes and course reading materials. The final output of the project is a performance comparison between a sequential program and its respective parallel design. For both programs, students will have the write the codes themselves, as well as 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++. Mostly likely, student will need to familiarize with Linux/Unix systems, as well as bash command line, to compile, debug, and run their code. For each of the project steps, the students are expected to employ the concepts learned in class properly and report them in the document submitted together with their code.

Step 3 - Description

This step 3 requires the implementation of the parallel version the code generated in Step 2. The students must document and report the design decisions, describing the algorithm. The students also need to conduct a performance analyses, such as asymptotic analysis, speedup, efficiency, and cost. The analyses must make use of a set of different performance metrics where the execution of the programs are conditioned by a series of parameters predefined by the students - problem size and system size.

Step 3 - Specifics

For the implementation in this step, the students will have to employ both OpenMP and OpenMPI programming interface extensions. Thus, there will be two parallel versions of the same solution. Consequently, the implementation of this step will need to be, necessarily, in either C or C++. For proper performance analysis, if the code generated in step 2 is not in C or C++, students will need convert their implementation to either C or C++.
All the design decisions must be reported and justified, being allied to the directives provided by each of the programming interface extensions, OpenMP and OpenMPI. Please note that, for the implementation of the code, the students should avoid at any cost the use of specialized libraries. In other words, students should fully implement the code themselves. As a result, all the parallel optimization will come from only the re-designs that the students have proposed and implemented in the code for this Step 3.
Expected elements in the submitted documents and codes:

- **3.1.** Legitimate **design reasoning** for parallelization of sequential code – parallel design justifications in each of the parts of the code (pseudo-code);
- **3.2.** Proper utilization and justification of **OpenMP directives** and clauses;
- **3.3.** Proper utilization and justification of **OpenMPI directives** and clauses;
- **3.4.** Consistent execution of both codes (both parallel versions of the code) – **reproducibility**;

3.5. Consistent **performance analysis** on reasonable metrics: Speedup, Efficiency, Cost, Parallel Overhead, etc.;

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.

All content files should be organized in a ZIP file for submission through D2L.

Late Submission Policy

A penalty of 25% will be applied to late submissions. Late submission are accepted until the Late Submission Date, three days after the Submission Due Date. No excuses are accepted for missing deadlines. However, deadline extensions may be granted under extenuating circumstances, such as medical or physical conditions; please note that granting the extension is at the instructor's discretion.

Plagiarism

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

`https://brocku.ca/academic-integrity/`