COSC 3P93
Assignment 1

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

Introduction [10]

1.1. (i) What is Shared-memory Architecture? (ii) What are its advantages and disadvantages? (iii) Can we have multiple processes of the same program, sharing data in a Shared-memory System? Explain. **[5]**
1.2. (i) What is a Distributed-memory Architecture? (ii) What are its advantages and disadvantages? (iii) Can we have a multithreaded program sharing data among its threads in a Distributed-memory System? Explain. **[5]**

Parallel Hardware [28]

2.1. (i) What Is NUMA Memory Architecture? (ii) Is NUMA Architecture better than UMA Architectures? **[5]**
2.2. List and explain the two techniques to maintain Cache Coherence. **[5]**
2.3. (i) What is a Hypercube Connection? (ii) How does a Hypercube topology improve when compared against a 2D Grid Mesh and a Fully Connected Network? **[5]**
2.4. (i) Define and explain the Bisection and Diameter of an N-node Hypercube? (ii) Why are these two metrics important? **[5]**
2.5. Given the provided two codes **sharing_work.cpp** and **sharing_work2.cpp**, you must identify the common both undergo that compromise their performance. **[8]**
(i) For **sharing_work.cpp**, compile it using the following command (in Linux shell): "`g++ -O0 -Wall -std=c++0x -pthread sharing_work.cpp -o sharing_work`". You can run it by invoking (in Linux shell): "`./sharing_work`". Read its code, and you will notice that uncommenting one line (45) and compiling it again will improve its parallel performance drastically. Why? Explain.
(ii) For **sharing_work2.cpp**, compile it using the following command (in Linux shell): "`g++ -Wall -std=c++0x -pthread sharing_work2.cpp -o sharing_work2`". You can execute it by invoking (in Linux shell) "`./sharing_work2 4 1`" and "`./sharing_work2 4 8`". Why does the second run executes so much faster? Explain.

Parallel Software [22]

3.1. (i) What is Data-parallel Computation? (ii) What is Task-parallel Computation? (iii) Can we have programs that are both Data- and Task-parallel? Explain. **[5]**
3.2. Is user process locking required to control the order of access to guarantee sequential consistency? **[5]**
3.3. Why are Locks important in parallel algorithms? Define and explain the major problem that locks bring to parallel programs. **[6]**
3.4. How can one ensure Mutual Exclusion without Locks? **[6]**

Threads [16]

4.1. Assume a producer-consumer thread-based program. How producers and consumers can communicate/interact with each other? Why are queues (FIFOs) one of the best data structures for organizing their communication? **[8]**
4.2. In the context of threads, explain (i) condition variables and (ii) read-write locks. (iii) Why are they useful? (iv) Write two short scriplets, and explain/describe them, to exemplify the use of condition variables and read-write locks. **[8]**

OpenMP [24]

5.1. What are OpenMP (i) pragmas, (ii) directives, and (iii) clauses? **[8]**

5.2. Explain what the following **pragma** calls are making and justify their construct (the need for them): **[8]**

a)
```
result = 0.0;
#pragma omp parallel num_threads(thread_count)
  calculate( a,b,n, &result );
  adjust( &result );
```

b)
```
if (my_rank % 2 == 0 ) {
  #pragma omp critical (odd)
    a += local_sum;
} else {
  #pragma omp critical (even)
    b += local_sum;
}
```

c)
```
sum = 0.0;
#pragma omp parallel num_threads(thread_count)
for ( i = 0; i < n; i++ )
  #pragma omp for
  for ( j = 0; j < k; j++ )
    sum[i] += filter( a, b, j, n );
```

5.3. Explain the following loop schedule policies in detail: (i) default schedule, (ii) static schedule (different chunk sizes), (iii) dynamic schedule, (iv) guided schedule (different chunk sizes), and (v) runtime schedule. **[8]**

Marking Scheme

Marks will be awarded for completeness and demonstration of understanding of the material. It is crucial 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. Thus, students are recommended to keep it simple and clear.

Submission Material

The submission for this assignment will consist of a single part:

7.1. The ***Answer document (PDF)***. The answers to the assignment questions should all be included in this document. Also, include the description of the source Java/C/C++ code, especially the compilation and execution instructions. 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 assignment***. For writing the description file, use the Latex template enclosed in this assignment (update it accordingly!). Students do not need to install Latex software on their computers. Students can write it through Overleaf on their browser (it is a free tool). Just upload the latex template to an Overleaf project; it should compile/render the tex file gracefully.

7.2. The **C/C++/Java code respective to the assignment questions**. The code must be documented (commented) properly. The code should compile and run properly. The compilation and execution of students' codes 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***.

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/`