CS 698L Semester 2019–2020-I: Assignment 3

6th September 2019

Due Your assignment is due by Sep 15 2019 11:59 PM IST.

General Policies

- You should do this assignment ALONE.
- Do not plagiarize or turn in solutions from other sources. You will be PENALIZED if caught.
- We MAY check your submission(s) with plagiarism checkers.

Submission

- Write your programs in C++.
- Submission will be through Canvas. Submit a compressed file with name "<roll-no>.tar.gz". The compressed file should have the following structure.

```
roll-no
--report.pdf
--<problem1-dir>
----source-files
--<problem2-dir>
----source-files
```

  The "report.pdf" file should include your name and roll number, and results and explanations that have been asked for in the following problems. In addition to the results, also include the exact compilation instructions for each programming problem, for example, `gcc -O3 problem1.cpp -pthread`.

  You are encouraged to use the LaTeX typesetting system for generating the PDF file.

- For late submissions, email your submission to the instructor.
- Submitting your assignments late will mean losing points automatically. You will lose 10% for each day that you miss, for up to three days.

Evaluation

- Please write your code such that the EXACT output format (if specified) is respected.
- We will compile and evaluate your implementations on a Unix-like system, for example, recent Debian-based distributions.
- The evaluators are not expected to fix compilation issues.
- We will evaluate the implementations with our OWN inputs and test cases, so remember to test thoroughly.

Problem 1

[50 points]

Perform loop transformations to improve the performance of the following code snippet for *sequential execution on one core* (i.e., no multithreading). You may use any transformation we have studied, for e.g., loop permutation and loop tiling, but NO ARRAY PADDING OR LAYOUT TRANSFORMATION OF ARRAYS ARE ALLOWED.

```
#define N 4096
#define Niter 10;

double x[N], y[N], z[N], A[N][N];

for (int t = 0; t < Niter; t++) {
  for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
      y[j] = y[j] + A[i][j]*x[i];
      z[j] = z[j] + A[j][i]*x[i];
    }
  }
}
```

For this problem, you are free to use any desktop in the CSE department (or any other reasonable machine). A good start is to check the processor architecture and cache parameters for your workstation to better understand the impact on performance. Include the system description (for e.g., levels of private caches, L1/L2 cache size, processor frequency) in your report.

We will provide a template code to help with the computation. Modify the optimized function in the template code and perform evaluations. Present speedup results with the GNU gcc compiler (gcc -O3) in a tabular format. Submit a report explaining the main reason for the poor performance of the provided reference version, your proposed optimizations (with justifications) to improve performance, and the improvements achieved.

Problem 2

[50 points]

This is similar to Problem 1.

```
#define N 1024

double A[N][N], B[N][N], C[N][N];

for (int i = 0; i < N; i++) {
  for (int j = 0; j < N; j++) {
    for (int k = 0; k < i+1; k++) {
      C[i][j] += A[k][i] * B[j][k];
    }
  }
}
```

Problem 3

[50 points]

Write a Pthreads program to perform parallel matrix multiplication (*ijk* form). This is more of a warm up exercise with Pthreads, so we will assume a naïve O(n^3) algorithm. Populate your matrices with all ones.

(a) Assume the matrices are of dimensions 4096×4096. Report speedup numbers for 2, 4, 8, and 16 threads over the sequential implementation for the same matrix dimensions. This is strong scaling (problem size remains same but you increase compute resources).

(b) For this problem, we will use varying matrix dimensions. Report speedup numbers for

- 1 thread with matrix dimensions 256×256
- 2 threads with matrix dimensions 512×512
- 4 threads with matrix dimensions 512×512
- 8 threads with matrix dimensions 1024×1024
- 16 threads with matrix dimensions 1024×1024
- 16 threads with matrix dimensions 2048×2048
- 16 threads with matrix dimensions 4096×4096

This will check for weak scaling (kind of), where the problem size increases along with compute resources.

Your report should include speedup plots, x-axis should be the number of threads, and y-axis should be the speedup compared to the sequential version.

Problem 4

[50 points]

Write a Pthreads program to implement a producer-consumer system with condition variables. The system will have one producer thread and two consumer threads. The producer thread reads characters one by one from a string stored in a file named "input.txt", and then writes these characters into a circular buffer. A consumer thread reads four units of data from the circular buffer and prints them in the *same order* to stdout. The following constraints should be respected.

- Assume a buffer of size 8 characters.
- The producer thread must not overwrite the circular buffer until the data has not been read by a consumer thread.
- The consumer threads must not pick up tasks until there is something present in the shared data structure.
- Do not explicitly control the schedule of consumer threads, it is fine to have nondeterminism because of synchronization.

Test your code to avoid data races and deadlock conditions.