CS 698L Semester 2019–2020-I: Assignment 4

4th October 2019

Due Your assignment is due by Oct 18 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
```

  Include your 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 problem1.cpp -fopenmp`.

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

- For this assignment, 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 template code. Modify the template code as instructed and perform evaluations. Compare performance of the different versions and report speedup results with the GNU gcc compiler.

- 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]**

Consider the following code snippet.

```
#define N (1<<12)
#define ITER 100
double X[N][N];
int i, j, k;
for (k=0; k<ITER; k++) {
  for (i=1; i<N; i++) {
    for (j=0; j<N-1; j++) {
      X[i][j+1] = X[i-1][j+1] + X[i][j+1];
    }
  }
}
```

Create a parallel version using OpenMP. Feel free to apply *valid* loop transformations.

Problem 2 **[50 points]**

Assume an array of type int and the size of the array is 2^{24}. Implement computing the sum of the elements of an array using the following strategies:

(a) OpenMP parallel for *without* reductions. You are allowed to use synchronization constructs like critical or atomic.

(b) an OpenMP parallel for *using* reductions,

(c) OpenMP tasks where each task will reduce a sub-array of size 1024.

Problem 3 **[50 points]**

Bubble sort is a naïve way to sort a list of integers. However, it is not very amenable to parallelization since there are dependences across iterations. An alternative is *odd-even* transposition sort that provides more opportunities for parallelization. Parallelize odd-even transposition sort with OpenMP.

Problem 4

[50 points]

Implement the following variant of the producer-consumer problem. The producer thread(s) read text from a collection of files, one per producer. They insert lines of text into a single shared deque (double-ended queue). The input to the program is a file that contains list of filename to read as input text. The consumer thread(s) read the lines of text from the shared deque and tokenize them — i.e., identify strings of characters separated by whitespace from the rest of the line. A consumer thread prints a token as soon as it finds one.

Your OpenMP program should accept two integer arguments, which specify the number of producer and consumer threads respectively. Use parallel regions to parallelize the producer and consumer threads. For this problem, you do not need to report any speedup results.