COMS3008A: Parallel Computing
Exercise 4

2022-8-18

Objectives

Students are able to

- Compile and run an OpenMP program written in C
- Use OpenMP `parallel` construct to parallelize a sequential program
- Use synchronization constructs including `critical`, `atomic`, and `barrier`, in OpenMP programs where applicable.
- Understand the problems of false sharing and race condition in shared memory programming, and apply proper techniques to eliminate such problems in OpenMP programs.

Instructions

1. Baseline codes are provided in folder `exercise_4_codes`. You may use any IDE for C/C++, such as CodeLite and Code::Blocks, or you may use commands from a terminal to compile and run OpenMP programs. In `exercise_4_codes` folder, a `Makefile` and a `run.sh` are provided for compilation and running of the programs respectively.
2. Submission is not required unless otherwise instructed.
3. **Note:** In the case of submission of codes for any type of assessments, your code must be able to compile and run from Linux command line (or Linux terminal), as that is the only way we are going to use to mark your codes.

Problems

1. Compile and run `parallel_region.c`. Change the number of threads in the parallel region by setting parallel construct clause, library function, or environment variable, respectively.

2. Write a sequential program for computing the number π using the method discussed in the class. Then parallelize the serial program using OpenMP parallel construct. Implement all the techniques discussed in the class. Evaluate the performances of the sequential and parallel codes respectively by measuring the elapsed time, and show the speedup of your parallel implementation.

3. Implement an OpenMP program that computes the histogram of a set of M^2 integers (range from 0 to 255) that are stored in an $M \times M$ two dimensional array. Answer the following questions.

   (a) What is the fine level task in this problem?

   (b) How would you assign the fine level tasks to the threads? Experiment with different assignment strategies in your parallel program.