Problem set 4, OpenMP/Pthreads

TDT4200, Fall 2013

The assignments int this course are mandatory, and the work need be your own. You should NOT collaborate with any one on this assignment. The code will be checked using MOSS (`http://theory.stanford.edu/~aiken/moss/`) for similarities. If plagerism and/or collaborations are suspected, you may fail this course and/or be reported to the administration.

Deadline: 17.10.2013 at 22.00 Contact course staff if you cannot meet the deadline.

Evaluation: Pass/Fail

Delivery: Use It's Learning. Deliver exactly two files:

- *yourusername_ps4.pdf*, with answers to the theory questions
- *yourusername_code_ps4.*{*zip* |*tar.gz* |*tar*} containing your modified versions of the files:
  - *kmeans_openmp.c*
  - *kmean_pthread.c*
  - *gemm_openmp.c*
  - *gemm_pthread.c*

General notes: Code must compile and run on `clustis3.idi.ntnu.no`. You should only make changes to the files indicated. Do not add additional files or thrid party code/libraries.

Part 1, Theory

Problem 1, Multithreading

a) What is the difference between a process and a thread?

b) What is the difference between Pthreads and OpenMP?

c) Why is it not possible to take full advantage of a cluster like *clustis* with just OpenMP or Pthreads?

Problem 2, OpenMP Schedules

a) What is the purpose of the OpenMP schedule clause.

b) Mention and briefly describe the different schedule types.

Problem 3, Reductions & Races

a) What is a *reduction*?

b) What is a *race condition*?

c) The program *add.c* computes the sum of all the elements in an array in parallel. Explain why the result might be incorrect.

d) Change the code so that the result will always be correct, without degrading the performance.

Note: this will require changing less than 5 lines of code. Include (just the) changed code in your theory answers, not as a separate code file.

Problem 4, Deadlocks

The program in *loop.c* performs a parallel search for a number in a list. A while loop is used to end the search as soon as the number is found. Explain why this program might potentially deadlock.

Note: The program will not deadlock on every execution. You might have to run it several times before you see the deadlock.

Part 2, Code

Problem 1, Matrix multiplication

The programs *gemm_openmp.c* and *gemm_pthreads.c* both perform serial matrix multiplication.

a) Parallelize *gemm_openmp.c* as much as possible using OpenMP.

b) Parallelize *gemm_pthreads.c* as much as possible using Pthreads.

c) Pick a reasonable matrix size, and time both programs using 1 to 8 threads. Discuss your results.

Problem 2, k-means

The programs *kmeans_openmp.c* and *kmeans_pthreads.c* both implement a serial version of the k-means algorithm.

a) Parallelize *kmeans_openmp.c* as much as possible using OpenMP.

b) Parallelize *kmeans_pthreads.c* as much as possible using Pthreads.

c) Pick a reasonable problem size, and time both programs using 1 to 8 threads. Discuss your results.