Lab Session 3

Submission deadline: Feb 22, 11:59pm

Please submit your lab results through CatCourses, including Makefile, code and a short report (up to one page).You may find the OpenMP tutorial from Lawrence Livermore National Lab useful (https://computing.llnl.gov/tutorials/openMP/).

1. Producer consumer

Parallelize the "prod_cons.c" program (see lab3/prod_cons.c at CatCourses). This is a well-known pattern called the producer consumer pattern: One thread produces values that another thread consumes. This pattern is often used with a stream of produced values to implement "pipeline parallelism".

Use OpenMP directives to parallel prod_cons.c. Report the execution time.

Note: the key to parallel this program with OpenMP is to employ pairwise synchronization between threads.

2. Prefix sum

Parallelize the prefix sum algorithm with OpenMP based on the OpenMP lectures. Apply the techniques you learned from the lectures to remove flow dependency.

Test the execution correctness of your code by using different input arrays and different number of threads.

2.1 Report the execution time of your code for an input array with 4,096 random integers and 4 threads.

2.2 For 2.1, report the execution time for each of the 4 steps (see the lecture slides for the 4 steps).

Note: You may want to apply what you learned from the first lab to collect the execution time.

3. Linked list

Consider the "linked.c" program (see lab3/linked.c at CatCourses). This program traverses a linked list computing a sequence of Fibonacci numbers at each node.

3.1 Parallelize this program using OpenMP tasks.

3.2 Parallelize this program using loop worksharing constructs.

Change the list size (i.e., N) and the number of threads, and report the execution times.