exercises corresponding to chapter 8

The exercises below are ordered chronologically, using a double numbering. The first element in this double numbering refers to the lecture. The lecture notes should suffice to find the answers. Programs we have seen in class are available on the class web site.

26.1) The program "get_next.c" is an illustration of the producer/consumer model. The retrieval of next data can be interlaced with the processing of the current data, using multithreading.

(a) Describe how to add an extra processing stage of the data. In the setting of the program "get_next.c", this extra stage consists in adding the numbers of the buffer.

(b) The retrieval, viewing, and adding stages form a 3-stage pipeline. Illustrate the working of this pipeline for 7 instances using a space-time diagram.

26.2) Modify the program "get_next.c" using pthreads to implement the 3-stage pipeline as described in exercise 26.1. While the computer is waiting for the user to press enter to continue, the next sequence of number is generated, and the previous sequence in added. Illustrate the proper working of your program showing the output of a run on copper.

27.1) As observed in class, the program "workers.c" to illustrate the manager/worker model with pthreads to find prime numbers is not very efficient. Modify this program to implement the sieve of Erathosthenes (see §5.3.3 in the book, also your expertise with exercise 16.5 should help you here).

27.2) Use the "workers.c" program as a template to implement the embarrassingly parallel computation of π modifying the program "compute_pi.c" of Lecture 13, using pthreads. Follow the MPI program closely in the sense that every thread has its own variable to store the approximation for the integral.

27.3) Modify the program of exercise 27.2 so that every thread updates the approximation for π in a critical section. Give the output of a run with the program to show a critical section is really needed (i.e.: without it the results are incorrect). Then also print the output of a correct run, using a critical section.

28.1) Consider a 4-by-4 lower triangular matrix L with ones on its diagonal. Denote its inverse by L^{-1}. Describe the algorithm to compute L^{-1} in pseudo code. Draw the data dependency graph between the elements of L^{-1}, labeling the nodes in the graph in the order of computation. Generalize your data dependency analysis to compute L^{-1} for general n-by-n lower triangular matrices L with ones on their diagonal. What does your analysis reveal for a potential speedup using $p = n$ processors?

28.2) Implement the parallel algorithm you designed in exercise 28.1 using pthreads. Show the output of a run on a 4-by-4 matrix to illustrate the correctness of your algorithm, computing $L^{-1}L$ and LL^{-1} as a final check.

28.3) Implement the parallel algorithm you designed in exercise 28.1 using OpenMP. Show the output of a run on a 4-by-4 matrix to illustrate the correctness of your algorithm, computing $L^{-1}L$ and LL^{-1} as a final check.

29.1) The prime tester program we have seen in class suffers also from the same problems as the program "workers.c" (see exercise 27.1). Use OpenMP to implement the sieve of Erathosthenes. Do you need a critical section? Explain and justify your answer.

29.2) One advantage of our supercomputer copper is the huge amount of memory at our disposal. Experiment with your code of exercise 29.1 to achieve a speedup using more processors and a larger sieve.

29.3) Prime testing is a very important application. Conduct a literature study and summarize in a couple of paragraphs the state of the art in this field. In particular, for a computer as copper, using the best algorithms and programs available, what is the size of the numbers one could factor?

Homework collection is every Friday, at 1PM. Homework (unlike projects) must be completed individually. Feel free to contact me with questions about these assignments.