

**Fall 2005 CS 441/632/732 Parallel Computing
Homework-1**

Individual work only. 200 points. Due Sep 20, 2005.

1. Implement a barrier function using Pthreads condition variables and mutex primitives. Write a program to compute the sum of a vector of doubles of size 10,000,000 (use static allocation to create the vector and initialize the vector with random numbers). Create multiple threads and assign each thread a part of the initialization and summation (let the main thread also perform part of the initialization and summation). Use the barrier function to synchronize between all threads after the data is initialized and then start the local summation. Use mutex variables to perform the global summation and then call the barrier function again. Measure the total time spent between the two barrier function calls (get time after calling the barrier function). Also measure the time taken for the summation process. Use this to determine the average time taken by the barrier functions when number of threads involved in the barrier is varied from 1 through 4 (request # of processors = # of threads when submitting the job on Altix).
2. Implement matrix-matrix multiplication with 1-D and 2-D data distribution using (a) Pthread and (b) OpenMP primitives. Measure time taken for matrix sizes 5000x5000 and 10000x10000 for # of threads = 1, 2, 3, and 4 (request # of processors = # of threads when submitting job on Altix). For the 2-D data distribution use the following grid layouts: 1x1, 1x2, 1x3, 1x4, 2x1, 3x1, 4x1, and 2x2. Note that if develop the 2-D distribution then there is no need to develop a separate 1-D distribution, you can use Nx1 to test 1-D data distributions. Distribute the work load among multiple threads and use BLAS to perform local matrix-matrix multiplication within each thread. Use an efficient mapping of matrix blocks to threads to minimize data contention. Analyze the impact of the data distribution scheme and using Pthreads vs. OpenMP on the overall performance of matrix-matrix multiplication. To reduce the overall execution time of the program use threads to initialize the matrices but do not include the time spent in initialization when measuring the time taken for matrix-matrix multiplication. When using Pthreads use the barrier function developed in problem (1) to synchronize among different threads after the initialization, before starting the computations.

You can use Everest for you initial development, debugging, and testing and use Altix for the final submission and timing measurements. For both the problems plot the speedup curves for different number of threads.

Submission:

Email the source code along with any Makefile and scripts as a single tar file attachment to puri@cis.uab.edu with the subject "CS 441/632/732 Homework-1." Turn-in a printed report in class using the format provided at <http://www.cis.uab.edu/cs441/report.html>. After submission, do not make any changes to your source code on Altix, you will be asked to demonstrate your program on Altix and the timestamp of the files will be used to determine late submissions.

Grading:

Correct implementation of the programs	60 %
Test plan and verification of results	10 %
Performance Analysis	20 %
Lab report format/presentation	10 %