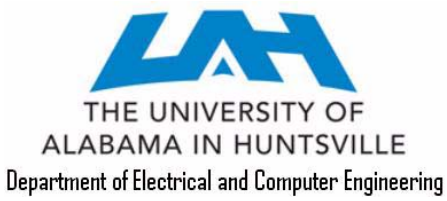

CPE 412/512
Fall Semester 2015

Homework Assignment Number 4

1. Students are to compile and execute on the Alabama Supercomputer's dmc system (dmc.asc.edu) the matrix/matrix multiplication program (*mm_mult_serial.cpp)* that is provided to them as part of this assignment. This reference code first generates a pseudorandom data set using input dimension parameters of *l*, *m*, and *n*, where the first matrix to be multiplied is of size *l* x *m* and the second matrix is of size *m* x *n*. These dimensions are entered on the linux command line. This code can be obtained by coping it from the shared directory on the dmc system using the command

   cp /home/shared/wells/hw4/mm_mult_serial.cpp .

   or by downloading it directly from the UAH CPE 512/412 Canvas™ site.

2. Next student should create a multi-threaded POSIX threads (pthreads) Single-Program-Multiple-Data (SPMD) matrix/matrix multiplication program (*mm_mult_pthread.cpp)* by expanding the serial matrix/matrix multiplication program (*mm_mult_serial.cpp)* that was provided in this assignment in a manner that allows for thread-based general purpose data parallelism. This program should divide the data set as evenly as possible among the threads using the row decomposition method that was outlined in class. The multi-threaded parallel code should generate the same pseudorandom data set as the original serial program where the first matrix to be multiplied is of size *l* x *m* and the second matrix is of size *m* x *n*. Both matrices should be generated and stored in the process local/thread global memory space of the host process and the time needed to generate and output these matrices should not be included as part of the timing data. The time to output this matrix to the screen, though, should not be included in this timing. The data type used in the parallel version should also be the same as the serial version which is of type ***float***. The program should be written in a general manner that allows the setting of the number of threads at run time using the first command line argument. The second, third, and fourth command line arguments should be the data size dimensions (*l*, *m*, and *n*, respectively, and in the case where only one of these is given then the program should make all three dimensions equal to the value of this second argument. Students should illustrate the correctness of this program for the 2,4,6, and 8 thread cases using a small but representative data set that demonstrates that the parallel version of the program functions correctly for at least one small case where the dimensions *l*, *m*, and *n* each have distinct/different values. Students should also verify the case where only all three dimensions are the same and only one is given on the command line. ***Students should include the resulting input/output as part of this homework write-up.***

3. After the small scale verification runs for the multi-threaded version of this program have been verified as discussed in Part 2 of this assignment, students are to perform detailed time measurements on both versions of the code for various data sizes using the gnu compiler These timing measurements should record the total execution time but should exclude the time it takes to generate and echo the input matrices as well as the time it takes to output the resultant matrix to the screen. It would be prudent for students to comment out the screen I/O statements during these runs. To allow for extended execution times and non-shared access to the CPU cores the dmc's batch queuing system must be employed. A separate handout contains information on how to access the batch queuing system. It is strongly suggested that students place all the run time commands into a single script and to execute this script using the *run_script* command in a manner where a total of 8 processing cores are reserved for the duration of the entire set of time measurements (even though most of the routines will use less than the 8 processing cores this will ensure that the same set of processing cores are used throughout the timing process). A separate script file is included (*mm_pthread_gnu.scr*) which is present on the Canvas™ site or can be obtained by coping it from the shared directory on the dmc system using the commands

cp /home/shared/wells/hw4/mm_pthread_gnu.scr .

The commands contained in this script assumes that only the data size and run time measurement information is supplied from the program and all other I/O are commented out.

The script can be invoked using the *run_script* command, students are to create separate executables for both the serial and parallel versions. If the serial source file is named *mm_mult_serial.cpp* and the parallel source file is named *mm_mult_pthread.cpp* then the following commands can be used from the linux command line to generate the two resulting executables.

To generate the gnu compiler executables for both the serial and parallel source code files enter the following commands:

g++ mm_mult_serial.cpp -o mm_mult_serial_gnu -lm -O3

g++ mm_mult_pthread.cpp -o mm_mult_pthread_gnu -lm -O3 -lpthread

Note that the same *g++* command is used to generate both executables.

The batch queuing system should be used to execute the script. Students should use the **class** queue, and reserve **8** processing cores (all other parameters should be given their defaults with the system to be executed on being specified as the **dmc**). These scripts will systematically execute each base serial and multi-threaded versions of the program multiple times by sweeping through data sizes of 200 to 5000 in increments of 200, where the data size is equal to the square dimension of the matrices are (i.e. $l = m = n$ = data size). The script is designed to execute the serial case and the parallel pthread representation for 2, 4, 6, and 8 threads process implementations where the first element on the command line is equal to the number of threads that are to be executed and the second is the square dimension size of the matrices involved in the matrix/matrix multiplication operation. They produce separate outputs for each implementation. After this data has been successfully generated students are to analyze the performance data that is generated as part of their homework report.

a. For the serial and multithreaded generated output, graph the runtime characteristics versus data size. Create five separate graphs. How do these implementations behave as the data size is increased? What is the algorithmic order of these implementations?

b. On the same graph plot the Relative Speedup versus the data size for each of the 2, 4, 6, and 8 thread implementations. Also create a graph that shows the Relative Efficiency versus the data size for each of the 2, 4, 6 and 8 thread implementations.

Students are to upload a single document file on the HW4 Assignment on the UAH Course Canvas™ site on or before its due date. Embedded in the final document submission should be the source code of the pthread program that was created by the student. *Due date is Wednesday October 28, 2015 by 11:59 PM.*