Shared Memory Programming Assignment

ECE 666: Homework #1

Assigned: January 29, 2021
Due: 2/5/2021 at 11:59 PM.

Please turn in your assignment on Brightspace.

Your task in this assignment is to write two **Pthreads** programs. You may assume that the functions specified below will be invoked by sequential code, that the programs will be run on an 8-core machine. Your system will be evaluated using the saguaro (saguaroX.ecn.purdue.edu where X = [1-9]) machines. You have been given course accounts on those machines. Your program should include standard unicore and multicore optimizations. Your program may not have any data races (e.g., shared counters that are unprotected by locks).

The compilation will be performed with basic gcc optimizations (-O1) but nothing further. (In practice, this is a non-issue, as gcc optimizations have little impact on this code.) You must submit three files for this assignment: 2 for the source codes specified below, and 1 report (in PDF format) that explains the algorithms that you used, operation of your program, challenges in achieving high performance, and the performance results with 1, 2, 4, and 8 processors for at least 3 reasonably large problem sizes. You should not provide a main function, though you will need to use one for testing.

Collaboration Policy

This is an individual assignment. You may not collaborate with any other student or any other party on either part of this assignment, nor may you draw from any solutions or sources previously written by others (including web-based resources
other than Pthreads reference guides). You may use your course notes and textbook. Good luck!

1. Write a C function called `matmult` with the following prototype and functionality. You may assume that the 3 matrices passed in (by pointer) have no overlapping addresses, and that the dimension of the matrix is a power-of-2 greater than 32 but less than 10000. You may create any helper functions that you need in your submitted file, as well as any structure definitions or static variables. Submit this program as a file called hw1-mm.c.

```
void matmult(int N, int *a, int *b, int *c) {
     int i, j, k;
```

```
        for (j=0; j<N; j++) {
              for (i=0; i<N; i++) {
                    c[i*N + j] = 0;
              }
        }
        for (k=0; k<N; k++) {
              for (j=0; j<N; j++) {
                    for (i=0; i<N; i++) {
                          c[i*N + j] += a[i*N + k] * b[k*N + j];
                    }
              }
        }
  }
```

2. Write a parallel in-place sorting function with prototype `void sort(int N, int *arr)`. The first argument represents the size of the array to be sorted and the second argument points to the array. You may use any sorting algorithm that you choose. You may assume that N is a power-of-2 at least 1024 but no bigger than 100 million. Submit this as a C source file called hw1-sort.c .