CSE 30341: Home Work Project 2

Assigned: Feb 6

Due: Fri, Feb 22, 10:40AM

Late submissions will not be accepted

Group effort

In this project, we will parallelize a matrix program using pthreads library. Consider a C program matrix.c (available online on expsys-svr4.cse.nd.edu at /home/cse30341-sp08/OS/hwp02/). This program performs an operation that is very similar to matrix multiplication: the key difference is that the values are updated in place.

```
1. for (row = 0; row < INDEX; row++)
2.   for (col = (INDEX-1); col >= 0; col--) {
3.     result = 0;
4.     for (cur = 0; cur < INDEX; cur++)
5.       result += (*mat1)[row][cur] * (*mat2)[cur][col];
6.     (*mat2) [row][col] = result;
7.   }
```

Parallize this program using pthreads library (to run faster on multiprocessor machines). You will run your experiments in expsys-svr4.cse.nd.edu machine. You will report the time it took to run the program using 1 thread (as assigned), 2, 3, 4, 5 and 6 threads. Repeat your experiments to draw statistically valid conclusions (make sure that two of you are not running experiments at the same time). You should check to make sure that the matrix operation results are the same (between the single threaded program and your multi-threaded program i.e, compare the execution of correctly executing programs). You could save the output from your program using ‘a.out > original.txt’. You can compare two outputs using the diff command as ‘diff original.txt mine.txt’.

You may turn in your program and any raw data that you deem to be of interest in the AFS drop box. These files may be used to verify your graphs. However, for the most part, we will only grade your project based on the written report. Turn in a hard copy of the report. Feel free to discuss your results with your colleagues.

Helpful pthreads calls:

1. `pthread_create()`
2. `pthread_join ()`
3. Mutex
    a. `pthread_mutex_init( );`
    b. `pthread_mutex_destroy( );`
    c. `pthread_mutex_lock( );`
    d. `pthread_mutex_unlock( );`
    e. `pthread_mutex_trylock( );`
4. Semaphores:
    a. `sem_init();`
    b. `sem_destroy();`
    c. `sem_post();`
    d. `sem_wait();`
    e. `sem_trywait();`
    f. `sem_getvalue();`

Useful tutorials:

- https://computing.llnl.gov/tutorials/pthreads/
- http://randu.org/tutorials/threads/