THE UNIVERSITY OF CHICAGO
Department of Computer Science
MPCS 51025 — Autumn 2013

Lab Assignment #4

Multiple Threads Matrix Multiplication

Due Tuesday, October 29, 2013 by 5:00 PM

October 21, 2013

1 PURPOSE & RATIONALE

The function of this lab is to let students gain specific experience with basic thread programming in C++. The goal of this lab is to provide students with the foundations necessary to understand the dynamics of multithreading that will be fundamental in many of the components in the UCEE architecture.

2 PRIMARY RESOURCES

- You should refer to relevant sections of the man pages for assistance for this lab, in addition to materials in the assigned chapters from the primary texts for this week (per the syllabus).
- You should ssh into the cluster to perform all lab activities.
- Make sure you have read this week's assigned reading.

3 ASSIGNMENT

You are to multiply two matrices together, specifically, Matrix A and Matrix B found in the labs directory. Both of these files have exactly 300 lines each. You will use the following algorithm to do produce the product Matrix C:

$$c_{i,j} = \sum_{k=1}^{n} a_{i,k} b_{k,j} \tag{1}$$

Specifically, on each iteration, a row of A multiplies a column of B, such that:

$$c_{p,k} = c_{p,k} + (a_{p,p-1} b_{p-1,k}) \tag{2}$$

For this lab, you are to produce two solutions:

1. A single threaded version of your algorithm (one main thread) that multiplies Matrix A and Matrix B and derives Matrix C. You are to time that version (`man 1 time`).
2. A multithreaded version of your algoritm that does the same thing, but as specified in section 3.1 below.

3.1 THREADING

As there are 300 rows in each matrix (A and B above), you will create 6 threads (in addition to the main thread) and these 6 threads will each multiply 50 rows each of matrix A and matrix B yielding a portion of product C. The first thread will therefore produce the first 50 rows (0 – 49) of the product matrix. The second thread will multiply the next 50 rows, namely, rows 50 – 99, etc. The sixth thread will of course work on rows 250 – 299 (0 offset) of the product matrix C. As part of your solution, you should make sure that each thread calls `std::thread::yield()` after each 10 rows that it processes.

3.2 ALGORITHM

There are several sources on the web (via Google search) to find out how to basically code the matrix multiplcation portion of this lab. Here is some sample code that multiples two 3×3 matrices (A and B) into a product matrix C (we are assuming of course that A and B actually have data):

```
int i,j,k;
for (i=1; i<=3; i++)
{
  for (j=1; j<=3; j++)
```

```
      {
        sum = 0;
        for (k=1; k<=3; k++)
        sum = sum + A[i][k]*B[k][j];
      }
      C[i][j] = sum;
    }
```

If you are wondering why we do not have to worry about collisions in our product matrix (and why we are not using mutexes, etc.), the answer is (a) we haven't covered mutexes and (b) you don't need them. The reason why you don't need them is that matrix multiplication is one of those embarrasingly parallel activities that serve our initial purposes perfectly. Unless you miscode the algorithm, you cannot possibly collide with another write.

You are to time (`man time`) your multithreaded version and explain the results you encounter. Turn in a compressed tar file containing your source code, `README.txt`, and `RESULTS.txt` as described in the Lab Submissions Requirements document on Piazza.