CSE 2431 **LAB 4** **AUTUMN 2015**

Using Threads for Parallelism

Electronic Submission: Mon. 11/30 11:59 pm
Written Report: Tuesday 12/1 (TTh section) or
Wednesday 12/2 (WF sections) at start of class

Goal: Getting experience in decomposing the solution in such way that multiple threads improve performance, i.e. decrease time to perform a given task, by taking advantage of multiple processors in the system. Note: This lab is not being written to take advantage of memory hierarchy.

Introduction: In this project, you will design programming solutions for matrix multiplication that take advantage of multiple threads executing in parallel.
Let matrix A be n*m matrix, and let B be m*p matrix. Then A*B results in matrix C n*p. Code below provides the algorithm for multiplication A*B=C.

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

Assignment: In your program matrix A should be 2400 x 2000 and matrix B should be 2000 x 500, i.e. n=2400, m=2000 and p=500.

Initialize members of matrix A, as follows a[i][j] = j-i+2, for i= 0, 1, 2 … n-1, and j=0, 1, 2, …, m-1.

Also, initialize members of matrix B, as follows b[i][j] = i-j+1, for i= 0, 1, 2 … m-1, and j=0, 1, 2, …, p-1.

1. Your program should first do multiplication of those two matrixes according to the algorithm given above. Let matrix C1 gets result of this multiplication, and you should display time needed to perform calculation. Do not include time needed for initialization of matrixes A and B.

2. At this point, you should have developed algorithm for matrix multiplication that should provide possibility that 2 or more threads can be executed (i.e. be running) in parallel. Try to develop algorithm that works for n threads, where n = 2, 3, 4, …

3. Next, your program should do multiplication according to the algorithm that decomposes the given algorithm such that 2 threads could work in parallel. Your main thread should create 2 threads that will do calculation and each thread terminates when it done with its task. Let matrix C gets results of this calculation and you should display time needed to perform the calculation.

4. Next, your program should do multiplication according to the algorithm that decomposes the given algorithm such that 3 threads could work in parallel. Your main thread should create 3 threads that will do calculation and each thread terminates when it done with its task. Let matrix C gets results of this calculation and you should display time needed to perform the calculation.

5. Repeat calculation with 4 threads, then with 5 threads, and finally with 6 threads similarly as described in Step 3 or Step 4 above.

Our Linux servers have 2 processors, each with 2 cores, with multithreading so you should see improvements in time. Also, compare each matrix C with matrix C1, and indicate if there are any differences; if your solutions are correct there will be no differences. Leave print statements used to check matrices in place, although commented out, so the TA will be able to check if matrices appear to be incorrect. The print statements should output a selection of 20-30 matrix elements.

Output: For a max of 6 threads, your display could look like this, with about the same running times: (Run#1 was from 3:44 pm Sat. 10/31 on Stdlinux, Run #2 was from 12:32 pm Sun. 11/1 on Stdlinux.) You should run the demo to compare the results to the results from your program.

Run#1:

Threads	Seconds
1	3.38
2	1.59
No error	
3	1.03
No error	
4	0.77
No error	
5	0.81
No error	
6	0.86
No error	

Run#2:

Threads	Seconds
1	3.67
2	1.63
No error	
3	1.04
No error	
4	0.87
No error	
5	0.99
No error	
6	0.93
No error	

Instructions: You may work with a partner on this project***, document: each partner's contribution and time spent as you are working on the lab***. Please complete your solution in our CSE Linux environment. The TA will compile and test your solution under this environment and using this compiler.

Compilation: Use the following to compile your code in <file.c> and obtain executable in <exec_file>:

gcc -O1 -Wall -o <exec_file> <file.c> –lpthread

Submission: Submit your < file.c> using the submit command. At the beginning of <file.c>, you need to state your full name, your email address, the same info for your partner, how to compile and run your program. Include conmments throughout code.

Submit Directories: TTh 11:10 Section: c2431ad
WF 9:35 Section: c2431ae
WF 12:45 Section: c2431ac

A late penalty of 10% per hour will be assessed, for instance, if a lab is submitted at 12:00 am or 12:59 am there is a 10% penalty. Any program does not compile will receive a zero. It is your responsibility to leave yourself enough time to ensure that you code can be compiled, run, and tested well before the due date.

Please turn in the following at the **start** of lecture following the electronic submission:

1. A 2-3 page typewritten report discussing:
    a) any design and implementation decisions made,
    b) a diagram illustrating how the matrix multiplication was partitioned for multiple threads,
    c) an explanation of how you determined that the matrix multiplication gave the correct result for each run with a different number of threads,
    d) a detailed explanation of each partner's contributions and the time each partner spent on the lab. ***This information is required for the lab to be graded.*** It's possible you may be asked questioned about this.
    e) test cases including an explanation of the results were what you expected and why or why not?
2. A printout of your completed code.

If you would like to work with a partner on this lab you will need to sign up during class by Thursday/Friday November 12th/13th. *Both partners must be present and sign up.*

Additional Notes/Clarifications:

1. You should be writing **one** program that will accept a number x from the command line where x represents the number of threads. x is a number between 1 and 6 inclusive; the program will run for 1 thread up to x threads. So for instance, if x=3, then the product of the matrices will be calculated using 1 thread, again with 2 threads, and then again with 3 threads.

2. When you are calculating the product of the matrices with some number of threads, say x, where x>1, the resulting matrix, C, should be compared with the matrix, C1, calculated using only one thread for the multiplication. If the matrices are the same, output 'No error'. Otherwise print an error statement. (Of course if the program is working correctly they will be the same and this is the expectation to receive full credit on this lab.)

3. The print statements you use to output elements of the matrix should output 20 elements where each element is from a unique row and column. You may comment out these print statements before you submit the lab but leave them in the code in case the TA wants to check values.

4. **BONUS OPPORTUNITY #1(**Possibility of 10% bonus points assuming that the rest of Lab 4 is working correctly, the bonus part is working correctly, and hardcopy report has been submitted). Modify the existing code so it will work for 7 threads. Do not add a special case to your code for 7 threads, instead you must modify or generalize the existing code so that it will also work for this case. If you are interested in including this in your implementation you must first figure out why 7 threads is different than using 1 to 6 threads. Note: The demo does not work for 7 threads. ***Do not discuss this on Piazza. Note: Each thread should be doing the same amount of work.***

5. **BONUS OPPORTUNITY #2**(Possibility of 5% bonus points assuming that the rest of Lab 4 is working correctly and the hardcopy report is submitted). Early submission: Electronic by Monday, November 23rd 11:59, Hardcopy by Tuesday November 24th , 11:10 am.

6. Hardcopy reports must be handed in at the ***start*** of the next class. ***Due to nearing the end of the semester, no late reports will be accepted.***