Georgia State University

CSc 4320/6320 Operating Systems

Fall 2020

Programming Assignment 2

Due Time: 11:59 PM, September 27

Object:
To understand and experiment with thread creation using Pthread library in Linux.

Problem Statement:
Matrix multiplication is one of the most fundamental math operations in computer science. As shown in math textbooks, the definition of matrix multiplication is given bellow:

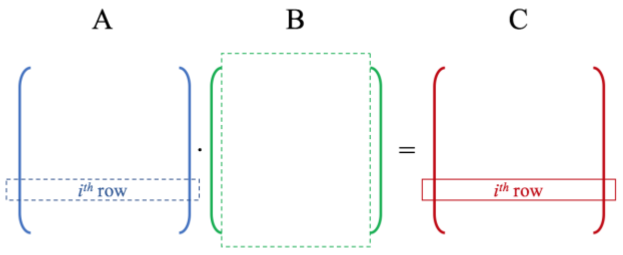
Given two maxtrices $A \in R_{m\times n}$, $B \in R_{n\times k}$, their multiplication $A\cdot B = C \in R_{m\times k}$, and

$$C_{ij} = (A\cdot B)_{ij} = \sum_{l=1}^{n} A_{il} B_{lj} \quad .$$

From the definition above, you can see that, for each element in *C*, it requires *n* times of multiplication and *n-1* times of addition to compute its value. The total number of computations is on the order of O($m\times k\times n$) or O(n^3) if *A* and *B* are *nxn* square matrices. Therefore, the standard matrix multiplication serial algorithm takes O(n^3) time to complete.

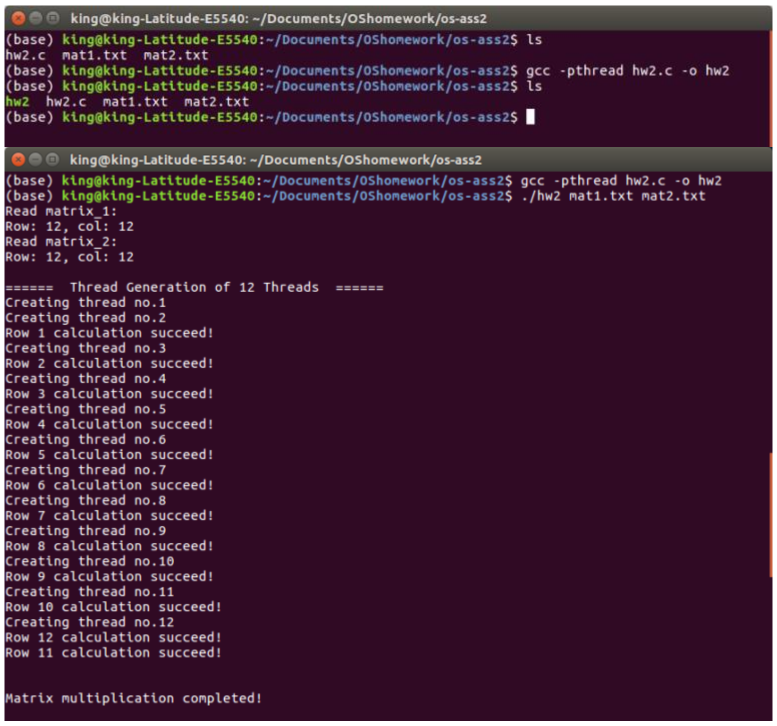
This computation overhead becomes significant as the dimension of a matrix grows. To largely boost up the matrix computation, parallel algorithms are good alternatives to conduct matrix multiplication. This is realized by distributing the computation load equally across multiple CPU cores and combining the intermediate results to get the final result.

This assignment asks you to implement a parallel algorithm through multi-threading to compute the result of the multiplication of two 12-by-12 matrices. The algorithm will distribute the computation loads equally into 12 threads. Specifically, create one thread for computing one row of the resulting matrix as shown in the figure below. In the i^{th} thread, compute the i^{th} row of *C* using the dashed boxes in *A* and *B*.



Steps and Requirements:

1. Download the incomplete version of *hw2.c* from iCollege
2. The two 12-by-12 input matrices are stored in separate text files, *mat1* and *mat2*, which can be downloaded from iCollege.
3. Use the names of the text files as program arguments on command line. Then you can access the name of these files through argv[1] and argv[2] in your main() function. As a result, you can open and read the matrices from the files.
4. Create **12** threads to finish the matrix multiplication. Each thread computes one row of the resulting matrix.
5. When moving matrix data around, pass the pointer of matrixes to a function instead of passing the values of the entire matrix. This will improve time and memory efficiency. For example, you should use "void fun(int (*mat)[12][12], ...);" or "void fun(int (*mat)[12], ...);" rather than "void fun(int mat[12][12], ...);" Penalty will be applied if this is not considered in the code.
6. Each thread should print a message stating whether or not its computation has finished (see sample screenshot below).
7. A message stating that the final result has been generated should be printed to the screen (see sample screenshot below).
8. Finally, print matrix 1, matrix2 and matrix3 to the screen (see sample screenshot below).
9. Compile the C source file using **"gcc -pthread -o hw2 hw2.c"**.
10. Use **"./hw2 mat1.txt mat2.txt"** to run the program.
11. Take screenshots of the program output in terminal.

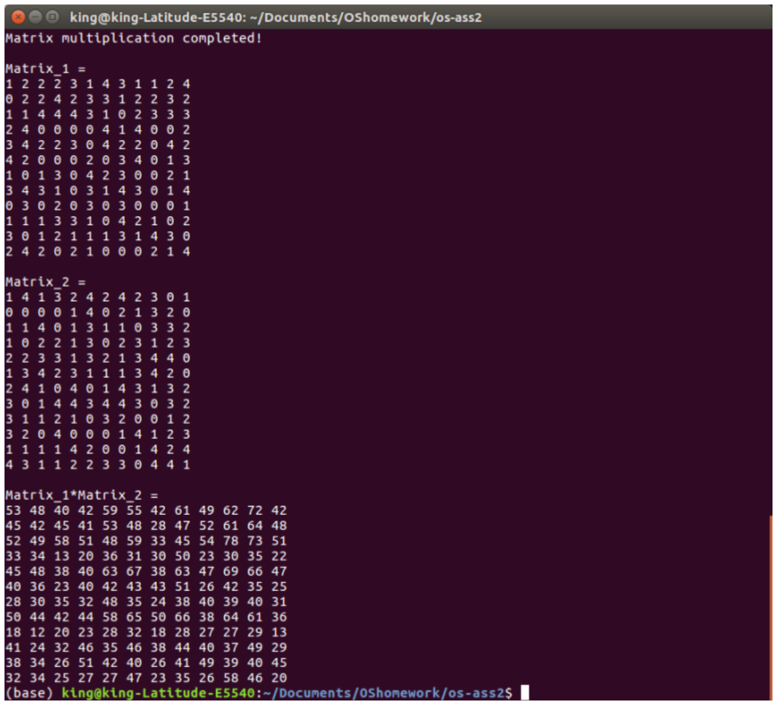


```
king@king-Latitude-E5540: ~/Documents/OShomework/os-ass2
(base) king@king-Latitude-E5540:~/Documents/OShomework/os-ass2$ ls
hw2.c  mat1.txt  mat2.txt
(base) king@king-Latitude-E5540:~/Documents/OShomework/os-ass2$ gcc -pthread hw2.c -o hw2
(base) king@king-Latitude-E5540:~/Documents/OShomework/os-ass2$ ls
hw2  hw2.c  mat1.txt  mat2.txt
(base) king@king-Latitude-E5540:~/Documents/OShomework/os-ass2$
```

```
king@king-Latitude-E5540: ~/Documents/OShomework/os-ass2
(base) king@king-Latitude-E5540:~/Documents/OShomework/os-ass2$ gcc -pthread hw2.c -o hw2
(base) king@king-Latitude-E5540:~/Documents/OShomework/os-ass2$ ./hw2 mat1.txt mat2.txt
Read matrix_1:
Row: 12, col: 12
Read matrix_2:
Row: 12, col: 12

======  Thread Generation of 12 Threads  ======
Creating thread no.1
Creating thread no.2
Row 1 calculation succeed!
Creating thread no.3
Row 2 calculation succeed!
Creating thread no.4
Row 3 calculation succeed!
Creating thread no.5
Row 4 calculation succeed!
Creating thread no.6
Row 5 calculation succeed!
Creating thread no.7
Row 6 calculation succeed!
Creating thread no.8
Row 7 calculation succeed!
Creating thread no.9
Row 8 calculation succeed!
Creating thread no.10
Row 9 calculation succeed!
Creating thread no.11
Row 10 calculation succeed!
Creating thread no.12
Row 12 calculation succeed!
Row 11 calculation succeed!


Matrix multiplication completed!
```



```
king@king-Latitude-E5540: ~/Documents/OShomework/os-ass2
Matrix multiplication completed!

Matrix_1 =
1 2 2 2 3 1 4 3 1 1 2 4
0 2 2 4 2 3 3 1 2 2 3 2
1 1 4 4 4 3 1 0 2 3 3 3
2 4 0 0 0 0 4 1 4 0 0 2
3 4 2 2 3 0 4 2 2 0 4 2
4 2 0 0 0 2 0 3 4 0 1 3
1 0 1 3 0 4 2 3 0 0 2 1
3 4 3 1 0 3 1 4 3 0 1 4
0 3 0 2 0 3 0 3 0 0 0 1
1 1 1 3 3 1 0 4 2 1 0 2
3 0 1 2 1 1 1 3 1 4 3 0
2 4 2 0 2 1 0 0 0 2 1 4

Matrix_2 =
1 4 1 3 2 4 2 4 2 3 0 1
0 0 0 0 1 4 0 2 1 3 2 0
1 1 4 0 1 3 1 1 0 3 3 2
1 0 2 2 1 3 0 2 3 1 2 3
2 2 3 3 1 3 2 1 3 4 4 0
1 3 4 2 3 1 1 1 3 4 2 0
2 4 1 0 4 0 1 4 3 1 3 2
3 0 1 4 4 3 4 4 3 0 3 2
3 1 1 2 1 0 3 2 0 0 1 2
3 2 0 4 0 0 0 1 4 1 2 3
1 1 1 1 4 2 0 0 1 4 2 4
4 3 1 1 2 2 3 3 0 4 4 1

Matrix_1*Matrix_2 =
53 48 40 42 59 55 42 61 49 62 72 42
45 42 45 41 53 48 28 47 52 61 64 48
52 49 58 51 48 59 33 45 54 78 73 51
33 34 13 20 36 31 30 50 23 30 35 22
45 48 38 40 63 67 38 63 47 69 66 47
40 36 23 40 42 43 43 51 26 42 35 25
28 30 35 32 48 35 24 38 40 39 40 31
50 44 42 44 58 65 50 66 38 64 61 36
18 12 20 23 28 32 18 28 27 27 29 13
41 24 32 46 35 46 38 44 40 37 49 29
38 34 26 51 42 40 26 41 49 39 40 45
32 34 25 27 27 47 23 35 26 58 46 20
(base) king@king-Latitude-E5540:~/Documents/OShomework/os-ass2$ 
```

Notes:

- Since the matrix data are shared between threads, it should be declared in the global data space before the main program. This would be true for any other data being shared among threads.
- Appropriate error checking of the command line is always a good practice for programmers. It is strongly encouraged because it also reduces grading error. (You can follow the examples in the textbook and the slides).
- Design your program in a clear and modular fashion.

Submission:

Submit the following to iCollege:

- The C source file *hw2.c*
- A report in PDF or Word that includes:
  o The screenshots of your program outputs
  o A high-level description of major components/functionality (including data structures) used in the program and how they interact with each other to achieve the assignment task.
  o A discussion on the pros and cons regarding efficiency and cost when
    - Using 1 thread for the whole job of matrix multiplication
    - Using 12 threads as in your program
    - Using 144 threads (one thread for one element of the final result)
  o A copy of your C source code from *hw2.c* file

Failure to follow the submission requirement will cause 10% deduction in the score.