CS417 Parallel Processing
Assignment 02: Matrix Multiplication Optimization
Due : 23h59 Friday 27th Octobre, 2023.

FCSE
GIKI

Fall 2023

1 Objectives

(a) Test matrix multiplication implementations.

(b) Test the use of pthreads, openmp.

(c) Test ability to write clean readable code.

(d) Test ability do error handling in code.

(e) Test ability to follow written instructions.

You will be using the C programming language, the pthread library, and OpenMP to do this assignment.

2 Matrix Multiplication

Matrix multiplication is a very common operation in machine learning (ML) and high performance computing (HPC), therefore a lot of research has been done to perform it in an optimal manner. The basic operation can be summarized as:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{np} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1p} \\ c_{21} & c_{22} & \cdots & c_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mp} \end{bmatrix}$$

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in} + b_{nj} = \sum_{k=1}^{n} a_{ik}b_{kj}$$

You have been given a basic matrix mutiplication implementation in the file `matmul_dynamic.c`. This program takes an input file as a command line argument and reads the matrices **A** and **B** from the file, multiplies them, and calculates the product matrix **C**. The actual multiplication happens inside the `matmul()` function. A sample input file (`matmul_input.txt`) has been provided; it's format is the same as the ones you've been using in your labs.

Your job, in this assignment, is to create further (optimized) versions of `matmul()` function.

While doing your tasks, **you'll also be preparing a document** for each section of this assignment. For each section you'll explain, what optimization you did, how did you think it'll improve matrix multiplication, what was the actual time taken by multiplication for 10x10 and 100x100 matrices.

2.1 Task 0. Time `matmul()`.

Note a how much time does the matrix multiplication via `matmul()` takes on your system for 10x10 matrices, 100x100 matrices.

2.2 Task 1. Create `matmul2()` (a multithreaded pthread version).

Now create a multithreaded function matmul2() using the pthreaded library.

In your document explain how it would improve performance and measure the timings for multiplying 10x10 and 100x100 matrices with 2, 4, and 8 threads.

2.3 Task 2. Create `matmul3()` (a multithreaded OpenMP version).

Now create a multithreaded function matmul3() using OpenMP.

In your document explain how it would improve performance and measure the timings for multiplying 10x10 and 100x100 matrices with 2, 4, and 8 threads.

2.4 Task 3. Create `matmul4()` (a tiled version).

Now create a singlethreaded function matmul4() using which does matrix multiplication of 10x10 and 100x100 matrices in a tiled manner.

In your document explain how it would improve performance and measure the timings on your PC using different tile sizes. See which tile size gives the best performance on your PC.

2.5 Task 4. Create `matmul5()` (a multithreaded tiled version).

Now create a multithreaded function matmul5() using OpenMP (or Pthreads) which multiplies the matrices in a tiled manner.

In your document explain how it would improve performance and measure the timings for multiplying 10x10 and 100x100 matrices with 2, 4, and 8 threads. Use the tile size which gave best performance in Task 3.

Try to maximize the parallelism in your implementation.

3 Error Checking and Clean Code instructions

Your programs should guard against invalid user input and in case of an invalid input (e.g. input file missing, or matrices dimensions not suitable for multiplication), it should print a helpful error message. **Your code should cater for non-square matrices.**

You would lose marks if your program crashes during use.

It is the programmer's responsibility to free any system resources i.e. memory , file descriptors, etc., they have acquired from the system. Your program should always free system resources once it is done using them.

Code should be properly indented, readable and commented.

You should always your compile your code using the `-Wall` option to check for warnings.

When displaying their valid output on the screen your programs should write to `stdout` and when displaying error messages, they should write to `stderr`.

4 Submission Instructions

1. Submission will be on MS Teams.
2. You submission should consist of two files: a .c file containing code and a MS Word file containing the report. In the report, add a section for each task. It should contain the code for that task, an explanation of how you think it should have affected performance, and the observed time taken to do multiplication of matrices of different sizes and how would you explain why it took this much time.
3. You can name your submission as u2020xxx_a2.c, u2020xxx_a2.docx where u2020xxx is your registration number.
4. Missing submission deadline on MS Teams will cost you 40 marks. Submissions received more than 24 hours after submission deadline will get a 0.

5 Rubric

This is an individual assignment. Any form of collaboration, cheating, plagiarism will get you a 0. Giving your code to somebody else, even if it is for their understanding only, is not allowed.

Most of this stuff you've done in lab so it shouldn't be very difficult.

You may be called for a viva; if you are unable to explain **any line** of your submitted code, you'll get a 0 even if it's working perfectly (we live in the age of chatgpt!).

Category	Marks
Task 0 working properly	5 marks
Task 1 working properly	10 marks
Task 2 working properly	10 marks
Task 3 working properly	15 marks
Task 4 working properly	20 marks
Report	40 marks
Missing deadline	-40 marks
Not following instructions	upto -100 marks
Max marks	100 marks