Computer Architecture

Project 4

Objective: Become familiar with pthreads and its programming model to evaluate the performance of matrix multiplication (gemm).

In this project, you will be required to use your computer to write 'C' program for executing matrix multiplication by creating multiple threads. POSIX threads (Pthreads) are widely used by the programmers for exploiting in-built multithreading paradigm. In this project, you will learn how to use this library to execute matrix multiplication.

Note: We are using your computer processor to run the multithreaded program and we will NOT be using gem5 for this project. We will use gcc as the compiler and NOT alpha cross compiler.

Getting started with Pthreads:

1. Here is a link to get you started on the basics of pthreads.
    a. https://www.cs.cmu.edu/afs/cs/academic/class/15492-f07/www/pthreads.html
2. There are two programs given with this project.
    a. The first deals with creating/spawning one child thread from a parent thread, printing something and then joining with the parent.
    b. The second program deals with creating three children threads from the parent execution thread, printing something and merging with the parent after completion
    c. You can compile and execute the pthreads programs by using the following commands:

```
user@ubuntu:~/$ gcc  multithread_p1.c -o mp1 -lpthreads
user@ubuntu:~/$ ./mp1
Before Thread
Printing CSE420 from Thread
After Thread
gfg@ubuntu:~/$
```

    d. Similarly you can compile and execute the second program and analyze the output.

Problem 1 [3 points]: Writing your own Matrix Multiplication benchmark

Given below is a simple matrix multiplication function:

```
void* mygemm(void* arg)
{
    // matrix multiplication
    for (int i =0; i < N; i++)
        for (int j = 0; j < N; j++)
            for (int k = 0; k < N; k++)
                matC[i][j] += matA[i][k] * matB[k][j];
}
```

Task: This simple matrix multiplication computes the product of two matrices and stores it into the third matrix. All the matrices are initialized as global variables.

a. Write a complete program to generate two matrices matA and matB with random values and call the mygemm function to compute the product of matA and mat B and store it in matC.
b. Please record the time in seconds of the execution of mygemm function.
   HINT: You can use time_t variables and clock() function in C. Start timer before calling mygemm function and have another timer to find the difference of start time and end time of the timer. You can multiply the difference by 1000 and then divide it by CLOCKS_PER_SEC, a library function, to get the time in milliseconds.
c. Try for different matrices (matA, matB and matC) sizes, 128x128, 256x256, 512x512 and 1024x1024.

Problem 2 [4 points]: Pthreads version

You can reuse lines from the two sample programs given.

Task: Develop a pthreads version of the program from Problem 1.

For each execution create matrices of size 128x128, 256x256, 512x512 and 1024x1024. Your program should be flexible to create 2, 4, 8 and 16 threads. Based on the threads created, each thread should receive a fraction of matA and whole of matB to compute a fraction of matC. For example, for matrix size of 128x128 and 2 threads, each thread should hold matA of size 64x128 and matB of size 128x128. The resulting matrix is a 64x128 per thread (matC), which can be joined after the mygemm function.

Note: Please be aware of the row ids of matA while dividing the matrix per thread.

Problem 3 [3 points]: Performance evaluation

You should have a clock calculating the milliseconds of each run. The total runs is given in the following table:

Matrix size	1 thread (Problem 1 execution)	2 threads	4 threads	8 threads	16 threads
	Runtime				
128x128					
256x256					
512x512					
1024x1024					

1. **Graph the above table, with number of threads as X axis and runtime as Y axis. You can provide four different graphs for each matrix size.**
2. **Please provide your processor configuration. The make, model, L1 size, L2 size, L3 size (if available) etc.**
3. **Comment (not more than 8 sentences) on the trends obtained from the plot.**

Submission Instruction

1. Submit one PDF file with all the graphs, the table, processor configuration on which the program was executed and your comment on the plot.
2. Submit one C file, with Problem 1 function and the Problem 2 function(s). It is up to you to decide on the design of the function. You can have one function with threads number as a variable or have 4 different functions for each matrix size. Matrix sizes can also be hardcoded or can be given as user input.
3. Submit one script that compiles and runs your entire configuration. This script can be a bash script or any other script of your convenience.
4. Zip all the three files into one file and submit.
5. In case where you are doing project in a group, only one group member should make submission through Canvas. All the team members of the group will receive the same score.

Rubric

1. **Problem 1 [3 points]**
   a. Correct program that compiles and runs with mygemm function.
2. **Problem 2 [4 points]**
   a. Correct function(s) to create multiple threads [2 points]
   b. Correct matrix size configuration and generation. [2 points]
3. **Problem 3 [3 points]**
   a. 4 graphs [2 points]
   b. 1 table [1 points]