CS4532 Concurrent Programming

Take Home Lab 3 and 4

Due – March 29 before 11:55 PM

Learning Outcomes

In this lab we will learn how to use "parallel for" and optimize a matrix-matrix multiplication to gain better performance. At the end of the lab you will be able to:

- use a "parallel for" statement to iterate through a loop body in parallel
- relate the performance of a given program to the architecture of the CPU
- improve the performance of the given program using a suitable optimization technique

Tasks

1. Write a sequential program to perform matrix-matrix multiplication
    - Dimensions of each matrix is $n \times n$ and elements/values of the matrix are of type Double.
    - Populate each matrix with randomly generated Double values.
    - Add necessary code to measure the time taken by the matrix-matrix multiplication. Do not measure the time to create matrixes and populate them with values.
    - The program should be written in C++ targeting the Linux environment.
2. Identify a suitable library for C++ and Linux environment that supports "parallel for" loops.
3. Write another C++ program to perform the matrix-matrix multiplication using a "parallel for" loop.
4. Run both sequential and parallel-for versions of your program while changing the matrix size n from 100 to 1,000 in steps of 100. [10 marks]
    - Record the time to perform matrix-matrix multiplication for each execution.
    - Collect sufficient number of samples for each configuration of n, while making sure your performance results are within an accuracy of ±5% and 95% confidence level.
    - Plot the matrix-matrix multiplication time against increasing n.
    - Plot speed up against increasing n.
5. Find out about the architecture of the CPU that you used for the evaluation. [5 marks]
    - Find out CPU model, no of cores, no of hardware-level threads, cache hierarchy, hardware optimizations, etc.
    - Can you justify the gained speed up knowing the architecture of the CPU?
    - Discuss your observations in detail while relating the observed behavior in graphs to the architecture of the CPU.
6. Find out at least two ways to optimize matrix-matrix multiplication.
    - Briefly describe a possible technique(s) while giving references.
7. Write another C++ program to perform the matrix-matrix multiplication using a "parallel for" loop and a suitable optimization techniques. [10 marks]
8. Run the new program and plot the results as described in Step 4.
9. Describe your observations in relation to the optimization technique(s) you selected and CPU architecture as in Step 5. [5 marks]

Notes

- Each team can have 2 students. You are free to select your lab buddy; however, it should not be the same buddy that you had for any of the previous labs.
- 25% of the marks (out of 40) for this lab is proportional to the extra speed up that you gain after optimizing the program in Step 7. Therefore, better the speed up more the marks you gain. Following formula will be used to calculate your 25%:

$$Min\ (Speedup\_after\_optimization - Speedup\_before\_optimization, 10)$$

- Student teams among the top 3 (overall) speedup values exceeding 20 will get 4 bonus points.
- Instructor will validate the speed up by running all 3 versions of your code. So make sure you give all the necessary instructions to complete and execute your programs.
- Make sure to comment your code.
- It is recommended that you clarify any concerns with the instructor before submitting the lab.

What to Submit

- Submit the following files as a single gzip file:
  - All source files
  - make file, if relevant
  - README.txt explaining how to compile and run your program. Also indicate any required libraries and how to install and link them to your program
  - A PDF file with graphs, images, and descriptions
- Name the file as **lab3_*<index no 1>*_*<index no 2>*gzip**. Replace <*index no x*> with your index number