SIT315 Programming Paradigms

Module2 Concurrent Programming

TaskM2.T1P: Parallel Matrix Multiplication

Overview of the task

To fulfill the requirements of this task, you will need to demonstrate your skills to use multi-threading in C/C++ to speed up sequential program. In this task, we use matrix multiplication as the example problem. See here (https://en.wikipedia.org/wiki/Matrix_multiplication)

Submission Details

Please make sure to provide the following:

- Source code of the sequential matrix multiplication program,
- Document outlining your program parallelisation and decomposition,
- Source code of the parallel program,
- Evaluation of your program on different input sizes and number of threads,
- Source code of the OpenMP version, and
- Updated evaluation document comparing OpenMP vs the other two programs.

Instructions

1. Implement a simple matrix multiplication program in C or C++. Matrix multiplication of C = A X B where A, B and C are matrices of size N X N (N rows and N columns) and both A and B are initialised with random values. At the end of the program, You should writ your output to a file. How matrix multiplication works? Please see here (https://en.wikipedia.org/wiki/Matrix_multiplication). Simply to calculate Ci,j (row i, column j), you multiply every element in matrix A row i by every element in matrix B column j. This is simply a three nested loops.

2. At the end of the program, please print the execution time - time taken to calculate the matrix multiplication - not including initialisation of matrices or writing results to file.

3. Once you have completed and tested the program, please review your code and develop a roadmap to parallelise your code. You should start with decomposition of the program/problem into sub-tasks - i.e. partioning data/tasks. Document your list of sub-tasks or activities you plan to do in parallel vs activities that need to be in sequence.

4. Implement your parallel algorithm in C or C++ using pthread library.

5. Evaluate the performance of your program (using execution time as a metric), to assess the speed up achieved. Reflect on different sizes of the input matrices and also number of threads you used in your program - vary from 2 to MAX number of threads. Compare the results with the sequential program.

6. Modify your sequential program to use OpenMP to achieve parallelism

7. Evaluate the performance of the OpenMP implementation vs pthread implementation vs the sequential program. Discuss your findings.

8. Submit your task as detailed on the submission details section above to OnTrack.