# COE 502 / 661 Parallel Processing Architectures

# Parallel Programming Assignment

## Objectives:

- Developing, writing, and debugging parallel programs
- Using OpenMP tools
- Parallel program performance evaluation

## Problem Statement

This assignment is about the parallelization of sequential algorithms and assessing the performance of parallel execution. The following sequential algorithms should be parallelized:

1. Dense matrix-matrix multiplication
2. Gaussian Elimination
3. Gauss-Seidel equation solver (red-black)
4. Quick sort

## Tools:

The programs should be developed and tested on the HPC cluster, using the installed OpenMP tools.

## Groups:

Two students can form a group. Make sure to write the names of the students involved in your group on the project report.

## Coding and Documentation:

Develop the code with the following aspects in mind:

- Correctness: the code should work properly.
- Scalability and performance: the code should run on multiple cores. Vary the number of threads and measure the speedup of parallel execution.
- Documentation: the code should be well written and documented.

## Report Document:

The project report must contain sections highlighting the following:

■ **Program Design**

Four parallel programs should be developed. Specify the parallel algorithm, the decomposition and assignment of data to the various threads. Indicate the communication and synchronization requirement between the various threads, and whether the code is scalable to a larger number of threads.

■ **Program Output and Performance**

Run all programs using large inputs files. Read the input matrices from text files and write the output to output files. Repeat the parallel execution using different number of threads. Use 1, 2, 4, 8, 16, and 32 threads. Verify the correctness of the parallel program against its sequential version (after ensuring the correctness of the sequential version). The best way to achieve this is to have a well-defined output and compare the results against the well defined output. Alternatively, you may generate random input, but for a pre-determined output. For example, you may generate random set of equations for a specific solution vector. Similarly, you may generate a random array for sorting. Use large input, such as 1000 by 1000 matrices, and sort arrays of $10^6$ elements or larger.

For each program, run the sequential code and repeat the execution of the parallel code varying the number of threads from 1 to 32. Compute and record the Wall execution time. Collect data in tables and draw charts showing the execution time versus the number of threads and the input size. Also compute and show the speedup of computation. Use a profiling tool to show the execution time of each thread and to fine tune the OMP code.

- **Discussion**

  Discuss all the various inputs and configurations that were handled. Discuss the speedup of the computation as the number of threads increases. Also discuss the effect of increasing the input size on the execution time.

  Compare the execution time of the sequential version of the program against the parallel versions to find the speedup and the overhead of parallel execution. Discuss the difference in the execution time when you run the sequential version and the parallel version using a single thread. Do not include the I/O time.

  Discuss whether input and output can be parallelized or should be executed sequentially. Measure the I/O time separately, especially when reading input and writing output to text files, and comment whether this time is substantial when compared to the execution time of the algorithm.

- **Teamwork**

  Group members are required to divide the work equally among themselves, so that everyone is involved in program development, and debugging. Show clearly the division of work among the group members using a Chart.

## Submission Guidelines:

All submissions will be done through WebCT.

Submit one zip file containing the source and binary code of all programs as well as the report document. Make sure that all programs are well documented.

## Grading Policy:

The grade will be divided according to the following components:

- Correctness of code: program produces correct results
- Efficiency and scalability of parallel code: achieving speedup in the parallel versions
- Documentation of code: program is well documented
- Team Work: participation and contribution to the project
- Report document: report is well written and results are well reported and discussed

## Late Policy:

The project should be submitted on the due date by midnight. Late projects are accepted, but will be penalized 5% for each late day and for a maximum of 2 late days (or 10%). Projects submitted after 2 late days will not be accepted.