Parallelization of Gaussian Elimination

Yan Solihin *

1 Problem Description

Gaussian elimination is a method for solving a set of linear equations. In this problem, you will parallelize it at different levels of granularity, and compare parallelism performance between them. In this assignment, you are asked to parallelize the Gaussian Elimination code. The background materials that you may find useful can be found in Chapter 3, Section 3.8.

The code shows a system of linear equations of the following form: $Y = AX$, where A is a $N \times N$ matrix, and X and Y are $N \times 1$ matrices. The upper triangulation step is the most time consuming step. The diagonalization step is less time consuming, and the solution step is the least time consuming step. Focus on only parallelizing the upper triangulation step.

2 Assignments

1. **for pivot loop**. Your first assignment is to analyze the opportunities for parallelizing the Gaussian elimination code at the "for pivot" loop level.

2. **for i loop**. Your second assignment is to analyze the opportunities for parallelizing the Gaussian elimination code at the "for i" loop level.

3. **for j loop**. Your third assignment is to analyze the opportunities for parallelizing the Gaussian elimination code at the "for j" loop level.

Perform and report the following for each of the three assignments:

- Analyze the code. Show the Iteration Traversal Graph (ITG), Loop-carried Dependence Graph (LDG) of the code. Then, analyze the sources of parallelism in the code. State why the loop is parallel or not, and if it is parallel, show what type of parallelism it has.
- If you find loops that exhibit DOALL parallelism, parallelize it with an appropriate OpenMP directives. Then, run the program and verify your results against the correct output that is provided in the file called gauss.out.
- For all the loops you identify as parallel, vary the scheduling policy between static and dynamic, as well as various chunk sizes, and observe their execution time.
- For all the loops you identify as parallel, measure and report the wall-clock execution time of the program (both sequential and parallel versions with 2, 4, and 8 threads). Then, plot the speedup curve comparing the parallel execution with various number of threads with sequential execution.
- Write a short observation with regard to what loop parallelization is the most fruitful, what scheduling policy is gives the best speedup, and explain the reasons for your observation.