Ravisutha Sakrepatna Srinivasamurthy

Gaussian Elimination

ECE 6730

Introduction

This project is about parallelizing Gaussian elimination method. Along with that various performance analysis is performed. The **RMS error** in the parallel calculations was found to be **zero**.

Performance Analysis

1. Timings for different dimensional matrices are collected and speedup, efficiencies are calculated.

Number of processors → Matrix size ↓	1	4	16	32	64
1000	22.4406	2.6094	0.4747	0.2437	0.1968
2000	183.6851	21.3252	3.7914	1.8179	1.1065
3000	623.9677	202.1143	13.0486	6.3823	3.1556

Table 1. Timing table

2. The formula for computing speedup is as shown below. Tables 2 contains the speedup for the corresponding execution times.
   Observation
   - As the number of processors increases, the speedup also increases.

$$speedup = \frac{sequential\ execution\ time}{parallel\ execution\ time}$$

Number of processors → Matrix size ↓	1	4	16	32	64
1000	1	8.5999	47.2732	92.0829	114.0274
2000	1	8.6135	48.4478	101.0425	166.0055
3000	1	3.0872	47.8187	97.7653	197.7335

Table 2. Speedup Table

3. Efficiency is given by

$$efficiency = speedup / processors$$

Efficiency is calculated in table 3. For all three matrices, the efficiency is calculated and plot of efficiency vs number of processors is given in fig 3.

- Observation
  - Efficiency initially reduces as number of processors increases.
  - But later, efficiency increases as the number of nodes increases (observe row 3000). This is because the overhead caused due to communication will fade out as the number of node increases.

Number of processors → Matrix size ↓	1	4	16	32	64
1000	1	2.15	2.9546	2.8776	1.7817
2000	1	2.1534	3.0280	3.1576	2.5938
3000	1	0.7718	2.9887	3.0552	3.0896

Table 3. Efficiency table

4. Plots of timings, speedup and efficiency is as follows

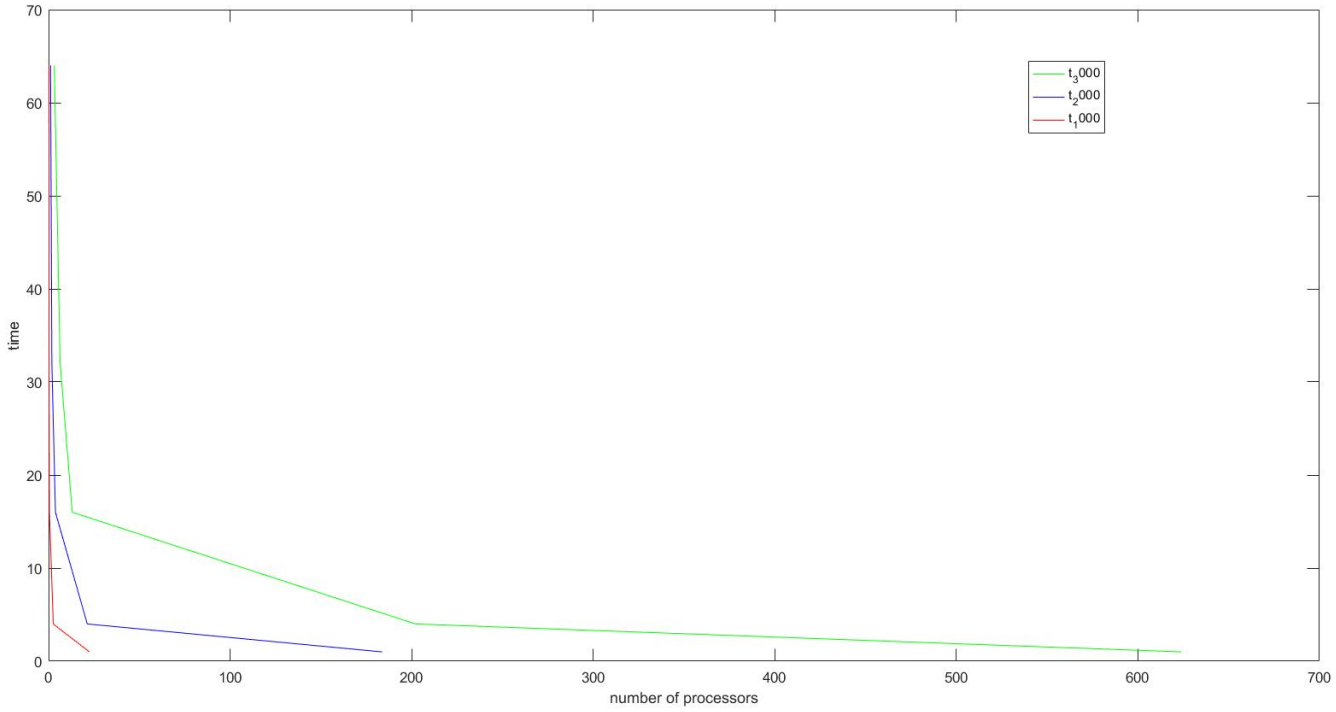

Fig1. Timing diagram

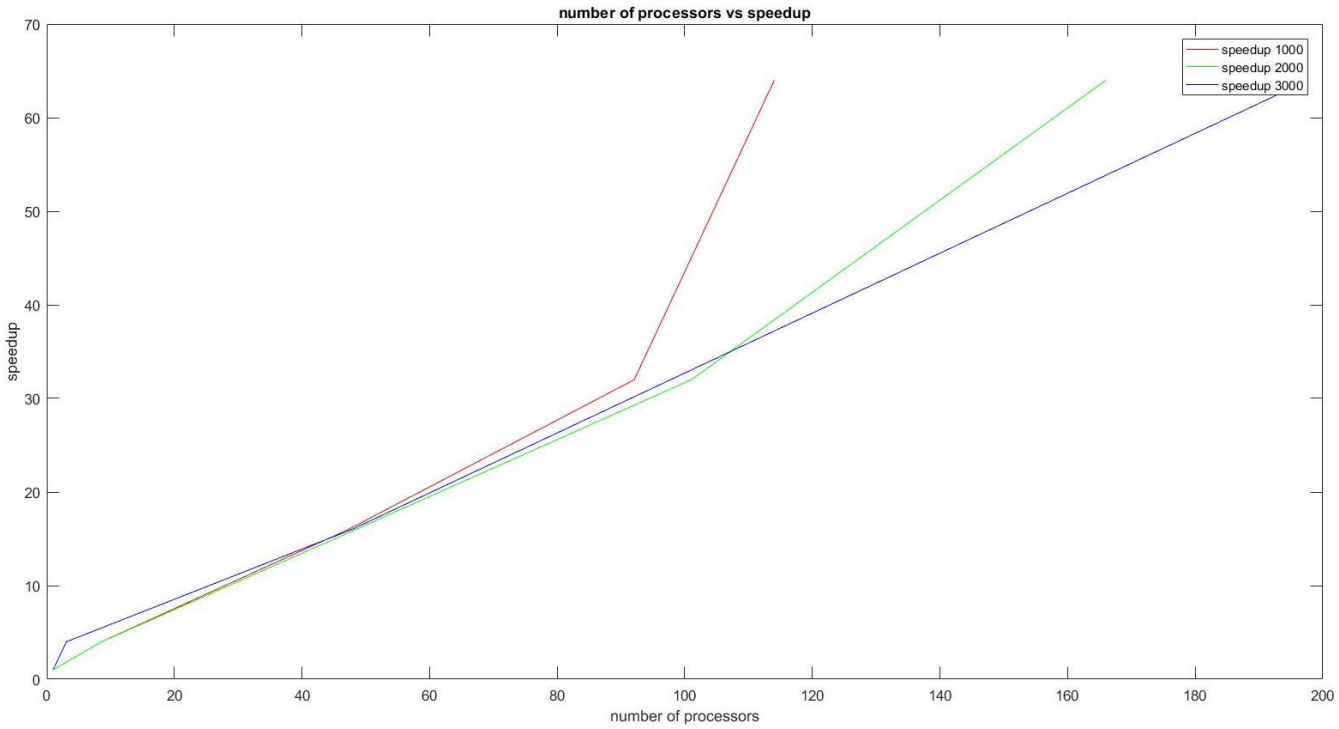

Fig 2. Speedup vs number of processors

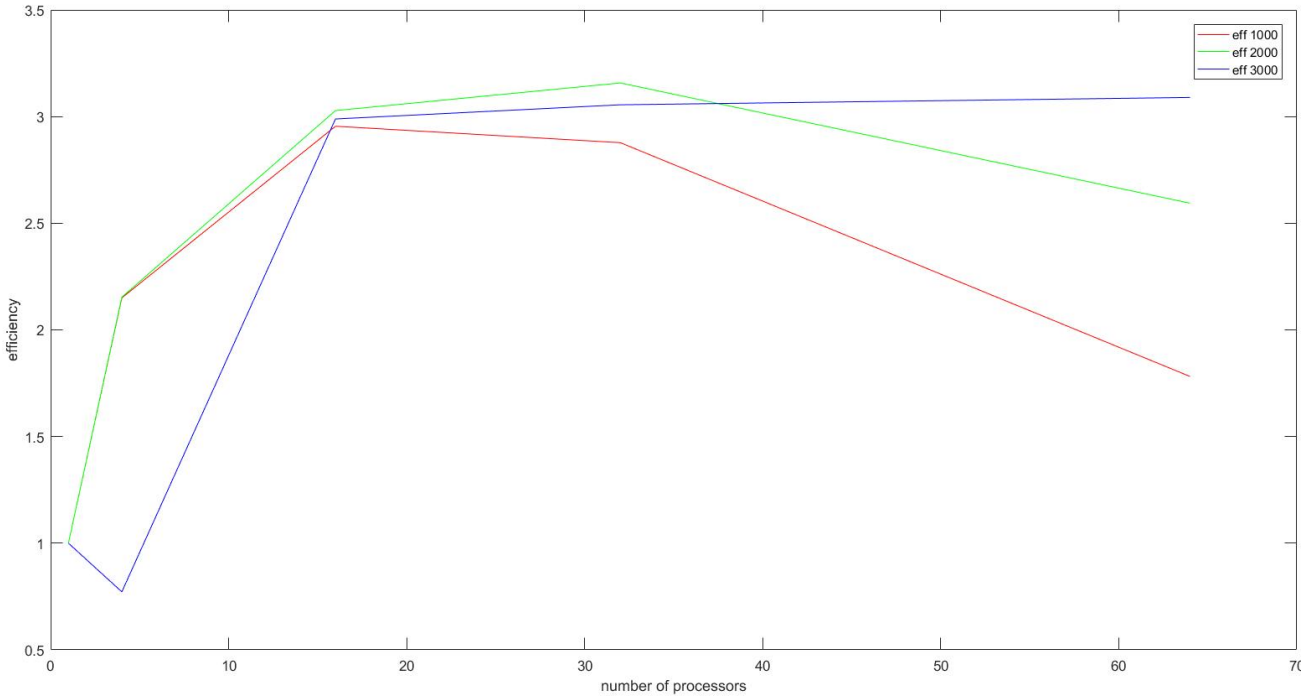

Fig 3. Efficiency vs number of processors

5. Execution Steps
   - ✓ Generating input file
     - Folder: SEQ/FILE
     - Compile make_graph: make
     - Compile print_graph: make print_graph.exe
     - Execution: ./make_graph.exe -r 10 -c 10 -p file.dat
   - ✓ Sequential Execution
     - Folder: SEQ/GUASSIAN
     - Execution: ./guassian.exe file.dat
   - ✓ Parallel Execution
     - Folder: PARALLEL/
     - Execution: qsub rsakrep.hw4.pbs
6. Obtained Results
   - ✓ PARALLEL/RESULT_1000
   - ✓ PARALLEL/RESULT_2000
   - ✓ PARALLEL/RESULT_3000

Note:

- Equation considered in the program is Ax = b.
  - A = system matrix/coefficient matrix
  - x = unknown
  - b = output vector
- File are named guassian instead of Gaussian