CS 426
Project 2 Report
Conjugate Gradient Solver Performance

Cahit Ediz Civan
22003206

1. Introduction

This report summarizes the implementation and performance evaluation of a serial and parallel Conjugate-Gradient (CG) solver for dense, symmetric positive-definite linear systems . Two **strong-scaling** experiments were conducted—fixing the problem size at 4096 and 8192, while varying the number of processes —and one **weak-scaling** experiment with a fixed local row count . Timing measurements were embedded in each run's output file and later aggregated for analysis.

2. Implementation Overview

- **Input Generator (`generate_input`)**: Creates a random symmetric, diagonally dominant matrix (off-diagonal entries ∈ [0,1), diagonals = row-sum + 1) and a length- vector with entries ∈ [0,1), writing them in the specified text format.
- **Serial CG (`serial.c`)**: Implements Algorithm 1 in C, using `clock_gettime(CLOCK_MONOTONIC)` for timing. Outputs elapsed time and solution to `s_output.txt`.
- **Parallel CG (`parallel.c`)**: Implements Algorithm 3 with MPI:
    1. Master (rank 0) reads data and distributes blocks of and to workers.
    2. Workers (ranks 1…) perform CG iterations, using nonblocking `MPI_Irecv`/`MPI_Send` for peer-to-peer gather of the search vector **p**, and `MPI_Allreduce` for global dot-products.
    3. Master collects the solution and elapsed time, writing to `p_output.txt`.

Compilation is via the provided **Makefile**:

```
make  # builds generate_input, serial, parallel
```

3. Experimental Setup

- **Hardware**: ORFOZ cluster (55 cores/node).

3.1 Strong-Scaling

- **Problem sizes**: N = 4096 and 8192.
- **Process counts**: 1, 16, 32, 64, 128, 256.

3.2 Weak-Scaling

- **Local rows per process**: 32.
- **Total size**: N = process count * 32.
- **Process counts**: 16, 32, 64, 128, 256.

4. Results

4.1 Strong Scaling

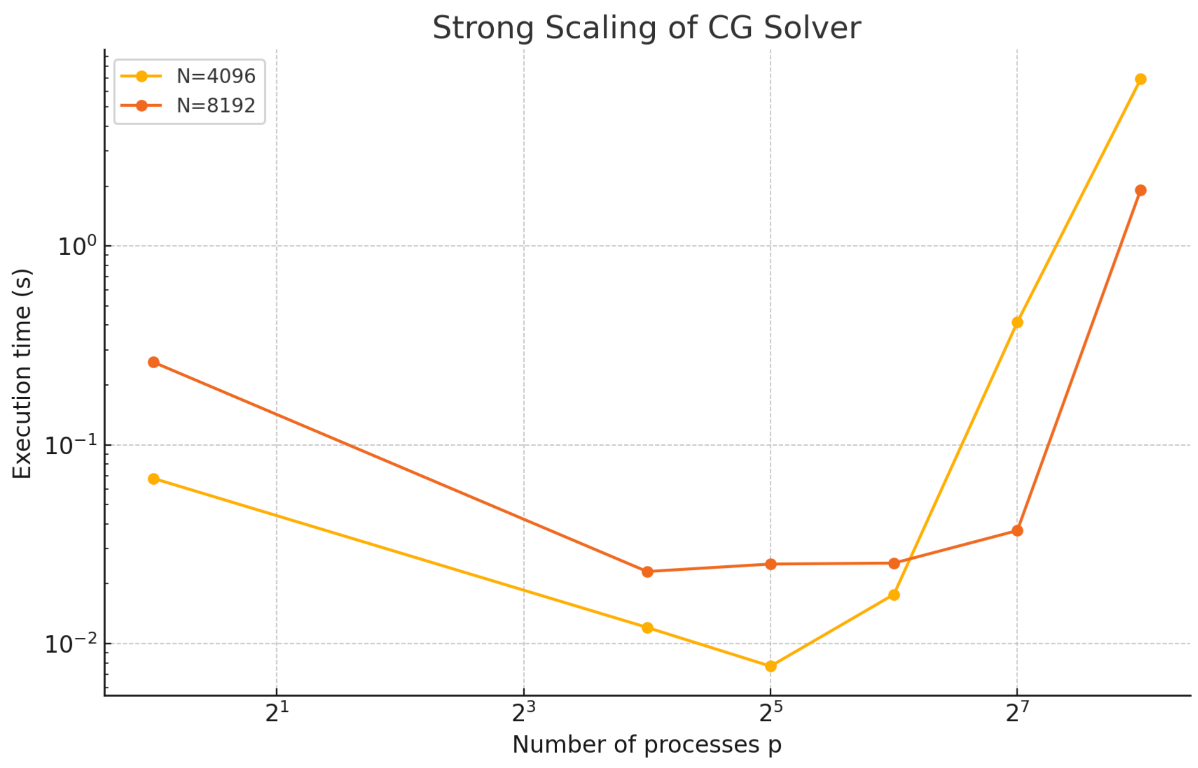

Figure 1. Execution time vs. number of processes

N	p	Time (s)
4096	1	0.067750
4096	16	0.012066
4096	32	0.007702
4096	64	0.017691
4096	128	0.413424
4096	256	6.910741
8192	1	0.260220
8192	16	0.023055
8192	32	0.025142
8192	64	0.025416
8192	128	0.037043
8192	256	1.918395

Observations:

- Both curves exhibit steep speedup from 1 to 32 processes.
- For size 4096, the fastest time is 0.0077 s at 32 processes. Beyond 32 processes, communication overhead dominates, giving 0.413 s at 128 processes and 6.91 s at 256 processes.
- For size 8192, scaling holds up to 64 processes, with a minimum of 0.0254 s. At 128 processes it rises to 0.0370 s and at 256 processes to 1.92 s.

4.2 Weak Scaling

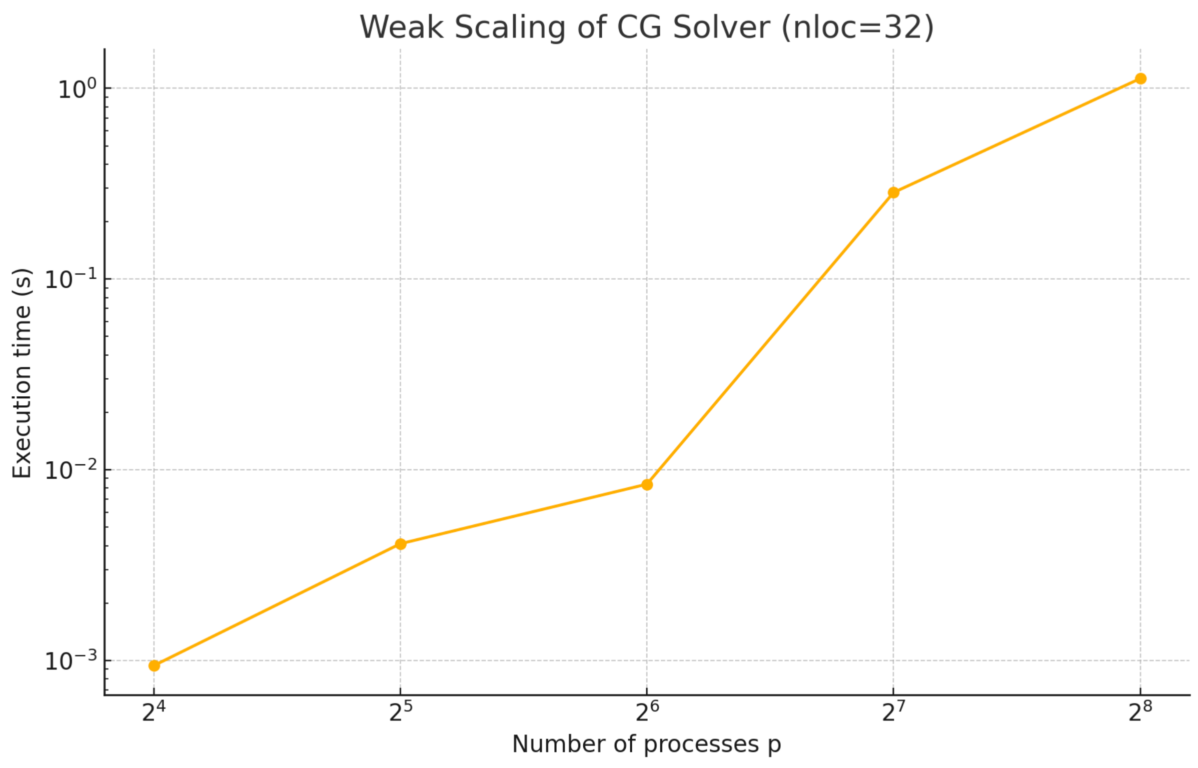

Figure 2. Execution time vs. number of processes for fixed local size 32.

Local rows	**p**	**Time (s)**
32	**16**	**0.000938**
32	**32**	**0.004090**
32	**64**	**0.008399**
32	**128**	**0.284359**
32	**256**	**1.126034**

Observations:

- Ideal weak scaling (almost) holds up to 64 processes (time remains under 0.01 s).

- At 128 processes the time increases to 0.284 s; at 256 processes it further rises to 1.126 s, reflecting growing communication overhead.

5. Discussion

- Peak efficiency occurs at 32 processes for the 4096-sized problem and at 64 processes for the 8192-sized problem, where each rank’s compute/communication ratio is most favorable.
- Diminishing returns beyond those points as each rank’s local block shrinks.

- For the weak scaling experiment, up to 64 processors the time remains similar (as they are very small we can say they are almost constant). Beyond that, the communication overhead increases significantly.

6. Conclusion

The MPI‑parallel CG solver delivers significant speedup for moderate core counts but is constrained by communication costs at scale.