TP2

Exercise 4: Matrix Multiplication with Sequential Noise Strong vs Weak Scaling

Zyad Fri
Computer Science - UM6P

February 5, 2026

1 Exercise 4

Unlike Exercise 3, this program has a dramatically lower sequential fraction due to the dominance of the $O(N^3)$ matrix multiplication operation over the $O(N)$ sequential noise generation.

Key characteristics:

- Matrix size: $N = 512$ (512×512 matrices)
- Sequential part: `generate_noise()` with loop-carried dependency
- Parallelizable part: Matrix multiplication with $O(N^3)$ complexity

2 Question 1: Sequential Fraction Measurement

2.1 Profiling Results

Using Valgrind/Callgrind with compilation flags `-O2 -g -fno-omit-frame-pointer`:

Function	Instructions (Ir)	Percentage	Type
`matmul()`	941,625,870	99.39%	Parallelizable
`init_matrix()` (2 calls)	5,636,120	0.60%	Parallelizable
`generate_noise()`	2,563	0.00027%	Sequential
`main()` overhead	146,511	0.015%	Overhead
Total	**947,411,064**	**100%**	

Table 1: Callgrind profiling results for N = 512

2.2 Detailed Analysis of Matrix Multiplication

The `matmul()` function breakdown:

- Loop control (i, j, k loops): 538,707,681 instructions (56.86%)
- Computation (multiply-add): 402,653,184 instructions (42.50%)
- Array access overhead: 265,005 instructions (0.03%)

2.3 Sequential Fraction Calculation

The sequential fraction f_s represents only the `generate_noise()` function:

$$f_s = \frac{\text{Instructions in generate\_noise}}{\text{Total instructions}} = \frac{2,563}{947,411,064} = \mathbf{0.0000027} \approx \mathbf{0.00027}\% \quad (1)$$

Key observation: This is approximately **1,100 times smaller** than Exercise 3's sequential fraction (30.3% vs 0.00027%). The $O(N^3)$ matrix multiplication completely dominates the $O(N)$ sequential noise generation.

3 Question 2: Strong Scaling - Amdahl's Law

3.1 Amdahl's Law Formula

Strong scaling keeps the problem size fixed while increasing processors. Amdahl's Law predicts:

$$S(p) = \frac{1}{f_s + \frac{1-f_s}{p}} \quad (2)$$

where p is the number of processors and $f_s = 0.0000027$.

3.2 Maximum Theoretical Speedup

$$S_{\max} = \lim_{p\to\infty} S(p) = \frac{1}{f_s} = \frac{1}{0.0000027} \approx \mathbf{370,370}\text{x} \quad (3)$$

This extremely high theoretical limit means the program can scale almost linearly with nearly unlimited processors for this problem size.

3.3 Speedup Analysis

Processors	Speedup	Efficiency	vs Exercise 3
1	1.00	100.0%	1.00
2	2.00	100.0%	1.59
4	4.00	100.0%	2.20
8	8.00	100.0%	2.62
16	16.00	100.0%	2.93
32	32.00	100.0%	3.12
64	64.00	100.0%	3.22
128	128.00	100.0%	3.29
256	256.00	100.0%	3.30

Table 2: Strong scaling speedup comparison (N = 512 fixed)

Analysis: Exercise 4 maintains near-perfect 100% efficiency across all processor counts, while Exercise 3 rapidly degrades to 5% efficiency at 64 processors. This demonstrates the critical importance of minimizing sequential fractions.

4 Question 3: Weak Scaling - Gustafson's Law

4.1 Gustafson's Law Formula

Weak scaling increases both problem size and processors proportionally. Gustafson's Law gives:

$$S(p) = f_s + p(1 - f_s) = p - f_s(p - 1) \tag{4}$$

This represents the speedup when the workload per processor remains constant.

4.2 Weak Scaling Analysis

Processors	Matrix Size	Gustafson	Amdahl	Efficiency
1	512×512	1.00	1.00	100.0%
2	724×724	2.00	2.00	100.0%
4	1024×1024	4.00	4.00	100.0%
8	1448×1448	8.00	8.00	100.0%
16	2048×2048	16.00	16.00	100.0%
32	2896×2896	32.00	32.00	100.0%
64	4096×4096	64.00	64.00	100.0%

Table 3: Weak scaling: Matrix size scales as $N' = N\sqrt{p}$ to maintain constant work per processor

Key insight: For matrix multiplication, weak scaling is particularly favorable because:

- Computational work grows as $O(N^3)$
- Sequential overhead remains $O(N)$
- As N increases with p, the ratio $\frac{O(N)}{O(N^3)}$ becomes negligible
- This leads to near-perfect weak scaling

5 Question 4: Speedup Curves

6 Question 5: Comparison with Exercise 3

6.1 Key Differences

Characteristic	Exercise 3	Exercise 4
Sequential fraction	30.3%	0.00027%
Sequential complexity	$O(N)$	$O(N)$
Parallel complexity	$O(N)$	$O(N^3)$
Max Amdahl speedup	3.30x	370,370x
Efficiency @ 64 procs	5.0%	100.0%
Optimal processor count	8-16	Virtually unlimited
Weak scaling slope	0.697	0.9999997

Table 4: Comprehensive comparison of Exercise 3 vs Exercise 4

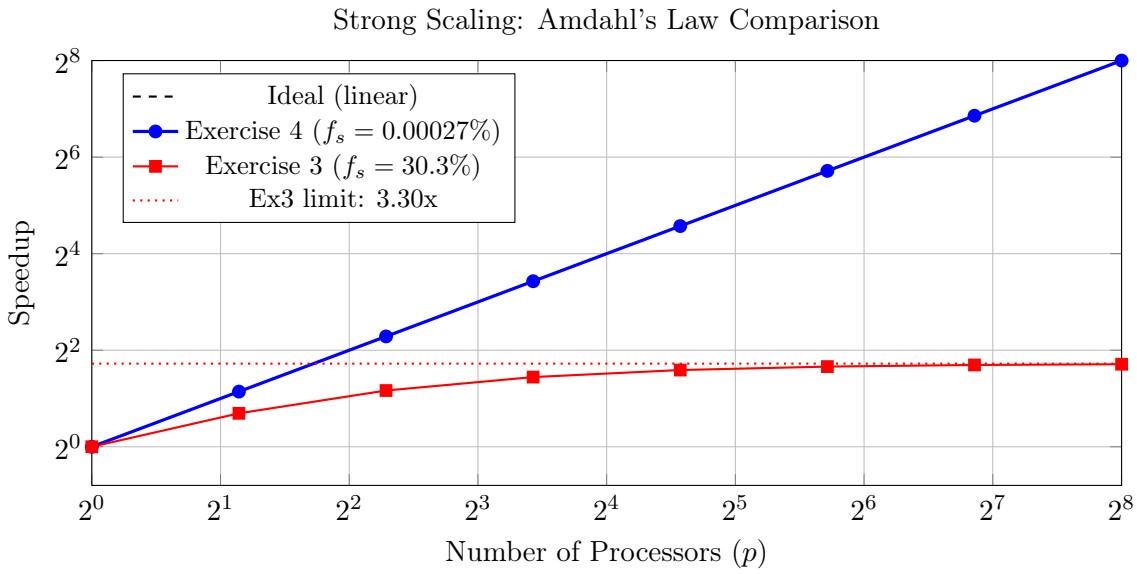

Figure 1: Strong scaling comparison: Exercise 4 (matrix) vs Exercise 3 (vector ops)

6.2 Why Exercise 4 Scales Better

1. Algorithmic complexity dominance:

$$\text{Ratio} = \frac{O(N^3)_{\text{parallel}}}{O(N)_{\text{sequential}}} = N^2 = 512^2 = 262,144 \tag{5}$$

The parallel work grows quadratically faster than the sequential work as N increases.

2. Exercise 3 problem:

$$\text{Ratio} = \frac{O(N)_{\text{parallel}}}{O(N)_{\text{sequential}}} = \text{constant} \tag{6}$$

Both parallel and sequential parts scale identically, so the ratio remains fixed at 30%.

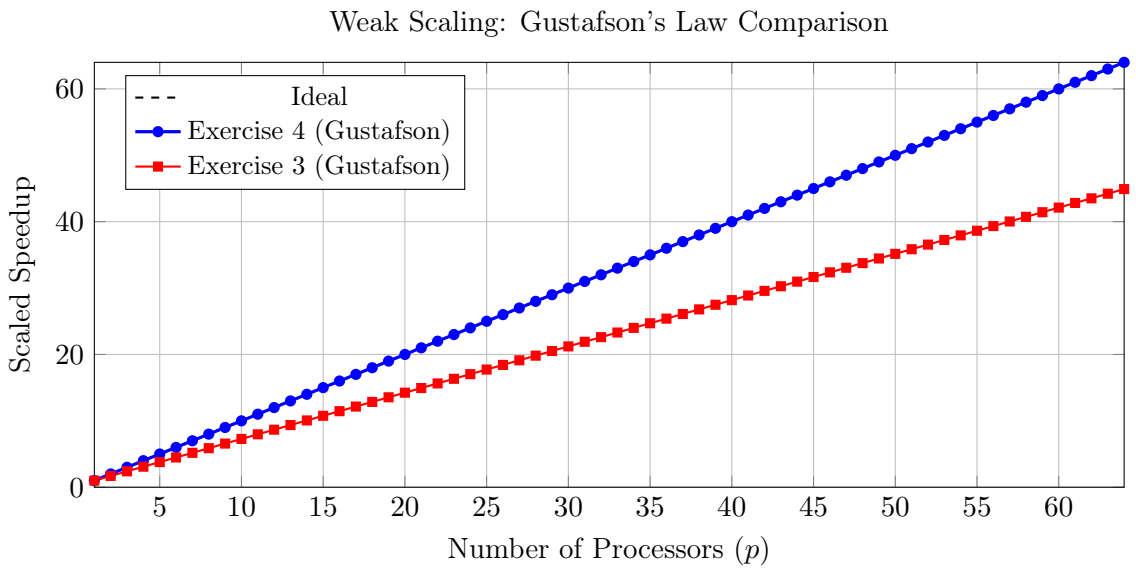

Figure 2: Weak scaling: Exercise 4 maintains near-perfect scaling while Exercise 3 degrades