Scalability Analysis: SIMD Implementation

Vectorization Impact & Precision Costs

Progetto AMSC

November 27, 2025

Performance Overview: SIMD (Float)

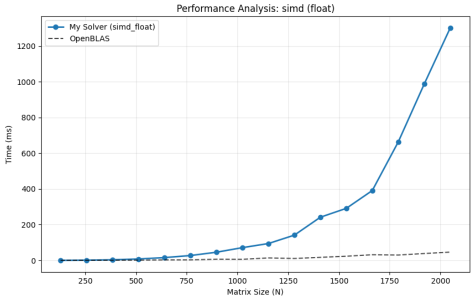

Figure: **Execution Time (ms)**

Drastic reduction in time compared to Naive (47s $\rightarrow$ 1s).

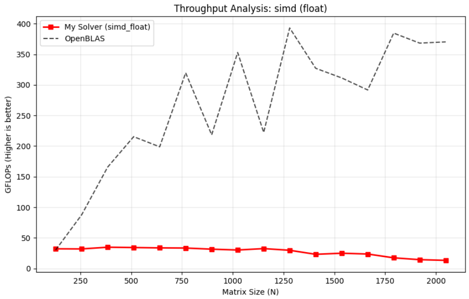

Figure: **Throughput (GFLOPS)**

Throughput drops as N increases (Memory Bandwidth bottleneck).

Quantitative Analysis: Float vs Double

Impact of Precision on SIMD Performance

Size (N)	Float	Double	Δ Overhead
128^3	0.14 ms	0.37 ms	+158%
512^3	8.08 ms	22.48 ms	+178%
1024^3	78.01 ms	170.75 ms	+118%
2048^3	**1.07 s**	**3.24 s**	**+202% ($\approx$ 3x)**

Insight: The Cost of Precision

Unlike the Naive method, precision matters significantly here.

Why is Double 3x slower?

- **Vector Capacity:** An AVX register holds 8 floats but only 4 doubles. Theoretical throughput is halved immediately.
- **Bandwidth Saturation:** Doubles require 2x memory bandwidth, saturating the bus faster.

Improvement vs Naive: SIMD Float at $N = 2048$ is $\approx$ **44x Faster** (1s vs 47s).

Why is SIMD fast (but not optimal)?

SIMD (Single Instruction, Multiple Data) leverages CPU vector registers (SSE/AVX) to process multiple elements per cycle.

- **Data Parallelism (The Win):** Instead of 1 multiplication per instruction, we perform 4 (Double) or 8 (Float) simultaneously.
- **The New Bottleneck (Memory):** As shown in the GFLOPS graph, performance drops for large N (30 $\rightarrow$ 16 GFLOPS). The CPU is now faster than the RAM. We are **Memory Bound**.
- **Next Step:** To reach OpenBLAS levels, we must reuse data in Cache (L1/L2) using **Tiling/Blocking**.

Vectorization Concept

Single Operation (Scalar):
`A[0] * B[0]`

SIMD Operation (Vector):
`[A0, A1, A2, A3] * [B0, B1, B2, B3]`

1 Cycle, 4 Results