Matrix Multiplication: A Comparative Microbenchmark in Python, Java, and C++

Martyna Chmielińska

October 23, 2025

Abstract

We implement and benchmark naive and cache-blocked matrix multiplication across Python, Java, and C++. We report mean, standard deviation, and best run for multiple matrix sizes and discuss profiling insights. The code, data, and this paper are available at: https://github.com/palantir00/individ assignment-matmul.

1 Introduction

This work compares programming languages and dataset sizes for a classic computational kernel: dense $N \times N$ matrix multiplication. We separate *production code* (algorithms) from *benchmark harnesses* and *tests*. We parameterize N, algorithm (naive vs. blocked), block size, and number of runs.

2 Methods

2.1 Algorithms

We implement two variants: (i) the naive triple-nested loop (IKJ ordering) and (ii) a cache-blocked variant with tunable block size.

2.2 Environments

Python 3.10+, Java 17 (Temurin), C++17 with CMake and Clang/GCC. All code paths use the same random seeds for fair inputs.

2.3 Benchmark Protocol

For each configuration we run R repetitions and compute mean, standard deviation, best runtime, and estimated GFLOP/s: $2N^3/t$. CSV outputs are written to `data/outputs/`.

3 Results

Table 1 summarizes representative results. Replace the numbers with your actual CSV values.

4 Profiling

Briefly describe bottlenecks (e.g., poor cache reuse in naive; reduced cache misses in blocked). Mention Python interpreter overhead vs. ahead-of-time compiled C++ and JIT-compiled Java.

Lang	Algo	N	Mean [ms]	Best [ms]	GFLOP/s
python	naive	256	—	—	—
python	blocked (64)	256	—	—	—
java	naive	256	—	—	—
java	blocked (64)	256	—	—	—
cpp	naive	256	—	—	—
cpp	blocked (64)	256	—	—	—

Table 1: Example summary; fill with your recorded metrics from CSV.

5 Discussion & Conclusion

Discuss how language/runtime and algorithmic blocking affect performance and stability across runs. Reflect on reproducibility and limitations (e.g., missing BLAS optimizations).

Reproducibility

All code, inputs, outputs, and this paper live in the repository: https://github.com/palantir00/individual-assignment-matmul. The folder contains: (a) this PDF, (b) `data/inputs` and `data/outputs`, (c) the full source code.