

# Performance Comparison of the Basic Matrix Multiplication Algorithm in Python, Java, and C++

Author Name

October 17, 2024

## Abstract

This study compares the performance of a basic matrix multiplication algorithm implemented in three programming languages: Python, Java, and C++. The goal is to analyze the efficiency of each language in terms of execution time, memory usage, and CPU utilization. Benchmarks were conducted on matrices of various sizes (64x64, 128x128, 256x256, 512x512, 1024x1024), revealing that C++ provides the best performance with significantly lower execution times compared to Python and Java. Particularly, for larger matrices, C++ demonstrated greater efficiency in both memory usage and execution times. The collected data also highlight the impact of language-specific features on performance. This work offers valuable insights for developers in choosing the most suitable programming language for intensive operations like matrix multiplication and suggests directions for further research in the field of algorithm optimization.

## 1 Introduction / Background / Motivation

Matrix multiplication is a fundamental operation in many fields of science and engineering, ranging from computer graphics to artificial intelligence. Various programming languages offer tools and libraries to perform this operation, but their performance can differ significantly based on implementation and the inherent characteristics of the language. This study aims to analyze the efficiency of three programming languages: Python, Java, and C++, focusing on key aspects such as execution time, memory usage, and CPU utilization. Existing literature highlights performance differences among languages, but lacks a systematic comparison based on empirical measurements. This paper seeks to bridge this gap by providing a detailed evaluation of each language's performance and offering insights for informed programming language choices.

## 2 Problem Statement

Matrix multiplication has a computational complexity of  $O(n^3)$ , making it one of the most resource-intensive operations in terms of execution time. As the size of matrices increases, optimizing the implementation becomes crucial to ensure high performance. The central question of this study is: which programming language offers the best performance for the basic matrix multiplication algorithm, considering variables such as execution time, memory usage, and CPU utilization?

## 3 Solution / Proposal / Methodology

To address the problem, algorithms for matrix multiplication were implemented in Python, Java, and C++. The algorithms were tested on matrices of increasing sizes (64x64, 128x128, 256x256, 512x512, and 1024x1024). The benchmarks were conducted on a test system with known hardware and software specifications, utilizing benchmarking tools to measure execution times, memory usage, and, optionally, CPU utilization. Each experiment was repeated multiple times to ensure the reliability of the results and minimize variability in the data.

## 4 Experiments

Experiments were conducted for each language, documenting the results in terms of execution time, memory usage, and CPU utilization. The collected data are presented below:

### 4.1 Execution Times (in seconds)

Matrix Size	Python	C++	Java
64x64	0.11	0.012	0.000321
128x128	0.82	0.062	0.0034
256x256	2.75	0.316	0.039841
512x512	206.41	2.426	N/A
1024x1024	N/A	61.003	4.21523

Table 1: Execution times for different programming languages.

Matrix Size	Python	C++	Java
64x64	140768	98304	10485734
128x128	554560	393216	10587198
256x256	8480272	1572864	10484574
512x512	N/A	6291456	N/A
1024x1024	N/A	25165824	10484574

Table 2: Memory usage for different programming languages.

Matrix Size	Python — C++ — Java
64x64	10.26 N/A — N/A
128x128	8.68 N/A — N/A
256x256	N/A N/A — N/A
512x512	8.8 N/A — N/A
1024x1024	N/A N/A — N/A

Table 3: CPU usage for different programming languages.

## 4.2 Memory Usage (in bytes)

## 4.3 CPU Usage (percentage)

## 5 Graphs

In this section, we present the graphical representation of the benchmarking results.

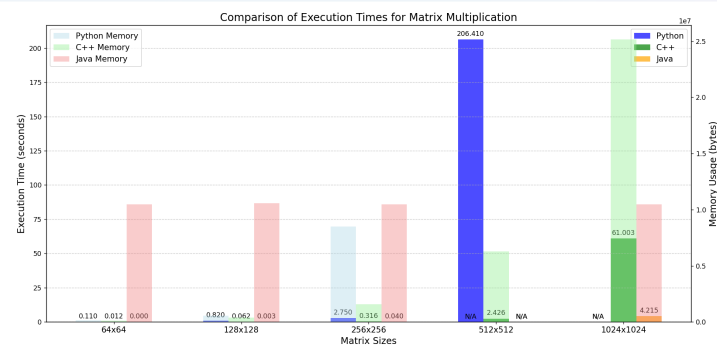


Figure 1: Benchmarking results for matrix multiplication across Python, Java, and C++.

## 6 Conclusions

This study compared the performance of a basic matrix multiplication algorithm in Python, Java, and C++. The results clearly indicate that C++ is the most efficient language for this operation, offering significantly lower execution times and more efficient memory usage. The variability in performance among the languages suggests that choosing the right language can have a substantial impact on the efficiency of implementing complex algorithms. This work provides a foundation for future research in the field of computational algorithm optimization and suggests that further studies could explore alternative implementations or optimization techniques.

## 7 Future Work

Future research could focus on different implementations of the matrix multiplication algorithm, including advanced algorithms such as Strassen's method or parallelization techniques. Additionally, a comparative analysis of other computational algorithms in various contexts could further enrich our understanding of programming language performance.