Basic Matrix Multiplication in Different Languages

Luis Perera Pérez
Grado en Ciencia e Ingeniería de Datos

Academic year 2024/25

Abstract

In this paper, we compare the performance of matrix multiplication implemented in Python, Java, and C. The experiments were conducted using different matrix sizes (from 10×10 to 1000×1000), and the execution times were measured. The results show that C outperforms both Python and Java significantly for large matrices, highlighting the advantages of lower-level languages for computationally intensive tasks. This study provides insights into the trade-offs between ease of development and execution efficiency in these programming languages.

1 Introduction

Matrix multiplication plays a fundamental role in various computer applications, such as scientific computing and machine learning. As the size of data sets increases, it becomes essential to optimize this operation in different programming languages. This article benchmarks basic matrix multiplication in Python, Java, and C, three languages commonly used in both academic and industrial environments.

2 Problem Statement

The multiplication of two matrices of size $n \times n$ has a time complexity of $O(n^3)$. As matrix size increases, this operation becomes computationally expensive, making language efficiency crucial for applications involving large data sets. This paper addresses the problem of how different programming languages—Python, Java, and C—handle matrix multiplication, both in terms of execution speed and memory usage, and which language provides the best performance as matrix sizes increase.

3 Methodology

The matrix multiplication algorithm was implemented in Python, Java, and C using a standard approach with $O(n^3)$ complexity. The code was separated into production and test sections to ensure accurate benchmarking. To measure performance, we used 'pytest-benchmark' in Python, 'JMH' in Java, and 'perf' in C, running each experiment multiple times to account for variance. And we tested three matrix sizes: 10×10, 100×100, and 1000×1000.

4 Experiments

We conducted our experiments on matrix sizes of 10×10, 100×100, and 1000×1000. The results are summarized in Table 1.

Matrix Size	Python	Java	C
10×10	0.18 ms	0.007 ms	0.001 ms
100×100	0.436 s	2.00 ms	0.54 ms
1000×1000	189.93 s	5.23 s	2.31 s

Table 1: Execution times for matrix multiplication across different programming languages.

As shown in Table 1, C consistently outperformed both Java and Python, particularly for larger matrix sizes. While Python's performance significantly deteriorated with increasing matrix size, C maintained a clear advantage due to its low-level memory management and optimization capabilities.

5 Analysis

- **Python:** Exhibits the highest execution time, especially for larger matrices, due to being an interpreted language with higher computational overhead.
- **Java:** Performs better than Python, thanks to the optimizations of the JVM.
- **C:** Demonstrates the fastest execution times, highlighting the efficiency of low-level memory management in C.

6 Conclusion

This paper compared the performance of matrix multiplication across Python, Java, and C, focusing on execution time for different matrix sizes. The results

clearly indicate that C provides superior performance, especially for large matrix sizes, due to its lower-level memory management and optimization potential. Python, while being the slowest, offers simplicity and ease of development, making it more suitable for smaller-scale tasks where performance is less critical. Java sits between the two, benefiting from Just-In-Time compilation to achieve reasonable performance while maintaining cross-platform compatibility. These findings can assist developers in choosing the right tool for the right task, balancing performance with ease of development.

7 Future Work

Future work could explore alternative implementations of the matrix multiplication algorithm, such as Strassen's algorithm, to improve performance further. Additionally, testing the use of optimized libraries in Python (such as 'NumPy') or parallelization in C and Java could yield interesting insights. Expanding the benchmarks to include other programming languages, such as Rust or Go, could provide a more comprehensive comparison of performance across modern languages.

GitHub Repository

The implemented code for matrix multiplication in Python, Java and C is available in the following GitHub repository: https://github.com/LuisPereraPerez/BigData.