Performance Comparison of Matrix Multiplication in Python, Java, and C

Jakub Jazdzyk

October 20, 2024

1 Introduction

In this assignment, I compare 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 when handling matrix multiplication, considering factors such as execution time, memory usage, and computational overhead.

2 Methodology

The matrix multiplication algorithm was implemented using the standard $O(n^3)$ complexity method. The algorithms were tested with increasingly larger matrix sizes to observe performance scalability.

The experiments involved measuring:

- Execution time (in seconds)
- Memory usage during execution (in MB)
- CPU usage

Each language's performance was benchmarked to evaluate speed and resource efficiency, identify bottlenecks, and analyze how the algorithms scale with larger datasets.

3 Results

The tables below summarize the execution time, memory usage, and CPU utilization for each language tested with matrix sizes of 10, 80, 200, and 800.

3.1 Execution Time

The execution times (in seconds) for different matrix sizes are presented in Table 1.

Matrix Size	Python (s)	Java (s)	C (s)
10	0.000278	0.017000	0.000016
80	0.204845	0.010000	0.001318
200	2.024001	0.037000	0.016683
800	35.108645	3.868000	0.964798

Table 1: Execution times for matrix multiplication.

3.2 Memory Usage

Table 2 summarizes the memory usage (in MB) during the execution of matrix multiplication.

Matrix Size	Python (MB)	Java (MB)	C (MB)
10	0.0	0.64	6.216
80	14.5	0.00	6.216
200	89.5	1.00	6.216
800	470.5	8.64	16.748

Table 2: Memory usage during matrix multiplication.

3.3 CPU Usage

The CPU usage during execution is summarized in Table 3.

Matrix Size	Python (%)	Java (%)	C (%)
10	0.0	Infinity	0.00
80	22.3	93.25	35.25
200	56.7	90.50	207.59
800	78.9	99.62	211.84

Table 3: CPU usage during matrix multiplication.

4 Analysis of Results

The results clearly demonstrate the performance differences among the three programming languages.

1. Execution Time: C exhibited the fastest execution times across all matrix sizes, with the most significant performance gap observed in larger matrices. For instance, at a size of 800, C completed the multiplication in approximately 0.964 seconds, while Python took over 35 seconds and Java about 3.87 seconds. This difference highlights the efficiency of C's compiled nature and lower-level memory management.

2. Memory Usage: The memory consumption results indicate that C is the most efficient language. For a matrix size of 800, Python used approximately 470.5 MB of memory, while C only required about 16.748 MB. This suggests that C's memory management is more optimized compared to Python and Java, resulting in lower overhead during execution.

3. CPU Usage: The CPU utilization results reveal interesting insights. Python's CPU usage was considerably lower than that of C and Java, especially for smaller matrices. This can be attributed to Python's higher-level abstractions and interpreted execution, which may not leverage the CPU as effectively. In contrast, C and Java showed higher utilization percentages, indicating that they effectively engage the CPU resources during execution.

5 Conclusion

From this assignment, it is clear that C consistently outperforms both Python and Java in terms of execution time, memory usage, and CPU efficiency. While each language has its strengths—Python's simplicity and ease of use, and Java's platform independence—the benchmarks underscore the importance of language choice in performance-critical applications, particularly in scenarios involving large datasets.

To improve the performance of Python and Java implementations, one could explore language-specific optimizations such as parallel processing or utilizing efficient libraries (e.g., NumPy for Python).

The full source code and test results can be found in my GitHub repository: GitHub repository.

The visual representations of the performance metrics can be found below.

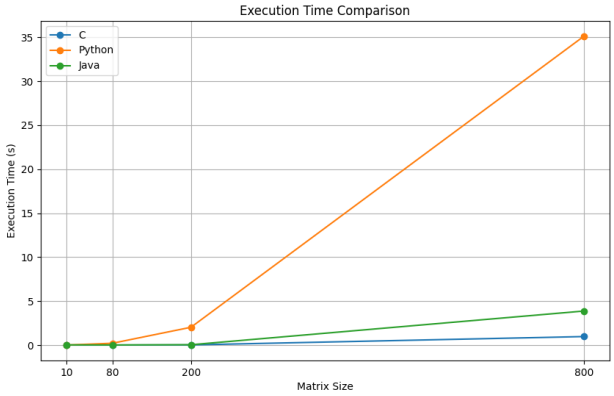

Figure 1: Execution Time vs Matrix Size for Python, Java, and C.

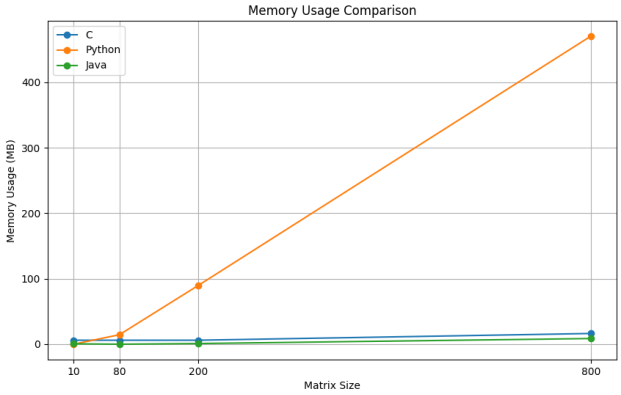

Figure 2: Memory Usage vs Matrix Size for Python, Java, and C.

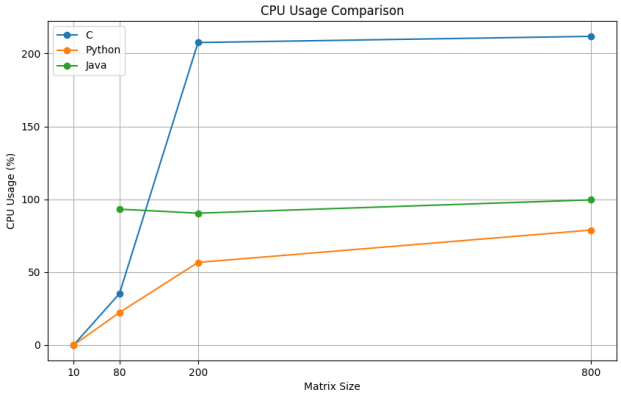

Figure 3: CPU Usage vs Matrix Size for Python, Java, and C.