BigData Individual assignment 1

Ivan Anikin

October 2024

1 Introduction

Matrix multiplication is a fundamental operation in many fields of scientific computing, ML, and data analysis. Efficiency of matrix multiplication can vary significantly depending on the programming language, algorithm, and hardware environment used. In this assignment, I compare the performance of matrix multiplication across three programming languages—Python, C, and Java—using both intuitive approaches and optimized algorithms.

2 Performance comparison

Metric	Python	C	Java
Execution time	0.001007 s	0.000000 s	0.000000 s
Memory usage	15.41 MB	4.44 MB	24 MB
CPU usage	0.093750 s	0.031250 s	0.062000 s

Matrix size: 16x16

Metric	Python	C	Java
Execution time	0.481757 s	0.015628 s	0.032000 s
Memory usage	17.48 MB	4.964 MB	25 MB
CPU usage	0.578125 s	0.046875 s	0.062000 s

Matrix size: 128x128

Metric	Python	C	Java
Execution time	306.193860 s	18.636116 s	2.299000 s
Memory usage	140.46 MB	29.764 MB	25 MB
CPU usage	306.843750 s	17.859375 s	2.406000 s

Matrix size: 1024x1024

3 Performance analysis

In terms of memory usage for 16*16 matrix, C is the most efficient, using only 4.44 MB compared to Python's 15.41 MB and Java's 24 MB. For CPU usage, Python exhibits the highest combined User + System CPU time (0.093750 seconds), followed by Java at 0.062000 seconds, while C remains the most efficient at 0.031250 seconds.

As the matrix size increases to 128x128, the performance gap between the languages becomes more evident. C remains the fastest, with an execution time of only 0.015628 seconds, while Java takes 0.032 seconds, and Python lags behind at 0.481757 seconds.

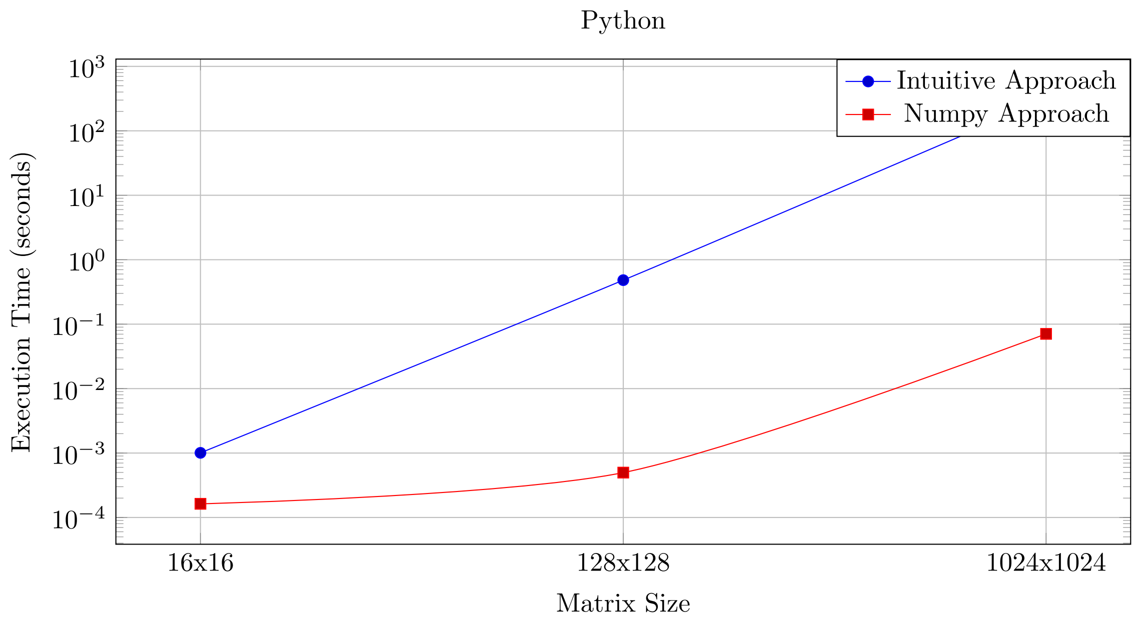
In terms of memory usage at 1024 matrix size, Python consumes the most (140.46 MB), while C uses 29.764 MB, and Java stays at 25 MB. Java also demonstrates superior CPU efficiency with 2.406000 seconds, compared to C's 17.859375 seconds and Python's 306.843750 seconds.

Key Insights:

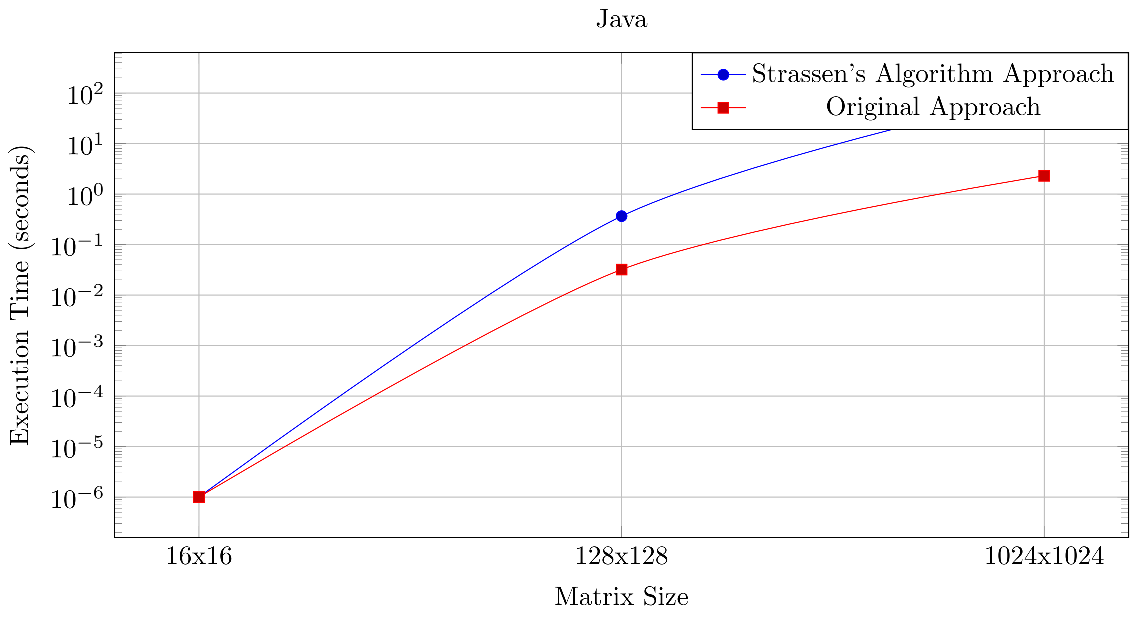
Execution Time: C performs exceptionally well on smaller matrices, but Java outperforms both Python and C significantly as matrix size increases, especially for larger matrices. Memory Usage: C uses the least memory across all matrix sizes, followed by Python and then Java. CPU Usage: Python consistently has the highest CPU usage, while C and Java are more efficient. Java particularly stands out in larger matrix sizes.

Based on these stats that I got by running the intuitive multiplication approach it seems, that the Python code is slowest while the C code is way faster. Both of them nevertheless seem to grow in cubic time regarding BigO. While Java seems closer to quadratic.

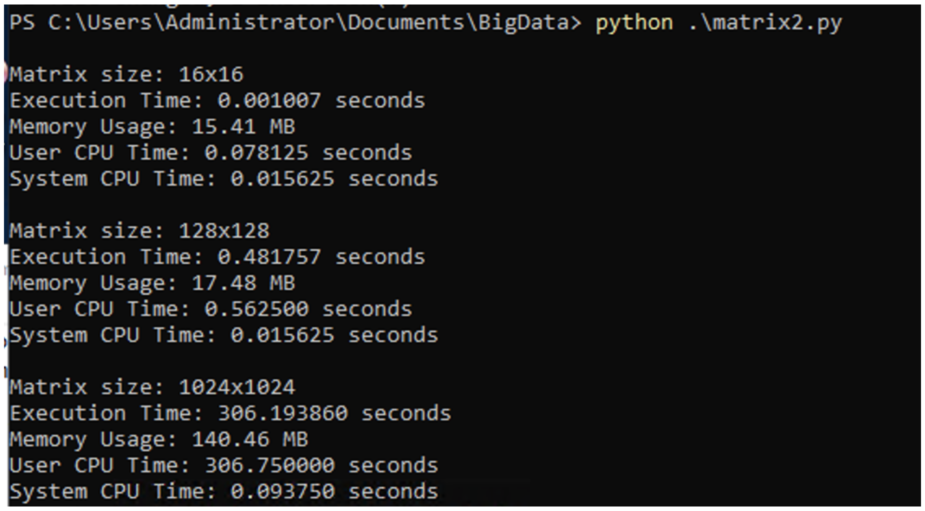
4 Multiplication time comparison charts



Python intuitive approach is slower than the Numpy implemented function taking using memory but at the same time moving the complexity from cubic to quadratic in BigO scale.

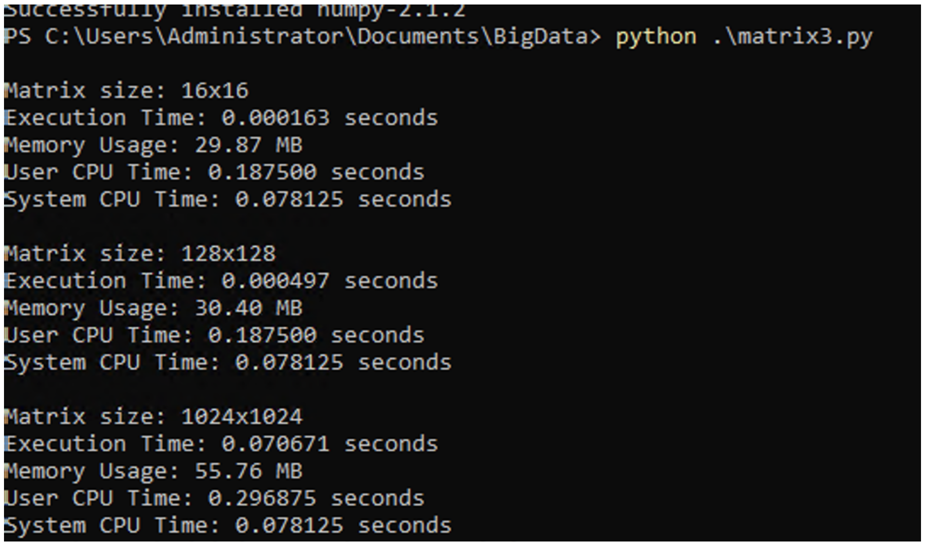


5 Console log output stats



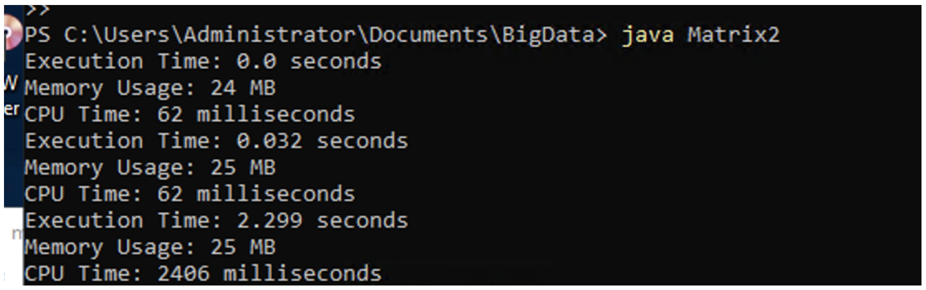
```
PS C:\Users\Administrator\Documents\BigData> python .\matrix2.py

Matrix size: 16x16
Execution Time: 0.001007 seconds
Memory Usage: 15.41 MB
User CPU Time: 0.078125 seconds
System CPU Time: 0.015625 seconds

Matrix size: 128x128
Execution Time: 0.481757 seconds
Memory Usage: 17.48 MB
User CPU Time: 0.562500 seconds
System CPU Time: 0.015625 seconds

Matrix size: 1024x1024
Execution Time: 306.193860 seconds
Memory Usage: 140.46 MB
User CPU Time: 306.750000 seconds
System CPU Time: 0.093750 seconds
```

Multiplication using intuitive basic approach



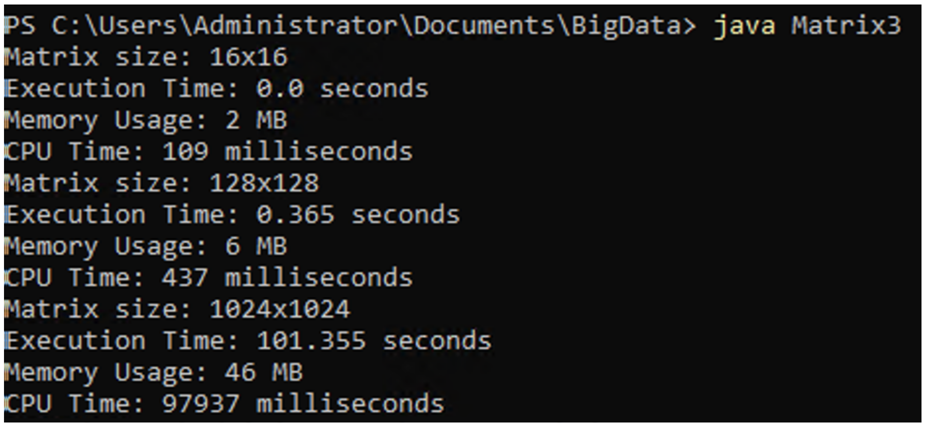
```
Successfully installed numpy-2.1.2
PS C:\Users\Administrator\Documents\BigData> python .\matrix3.py

Matrix size: 16x16
Execution Time: 0.000163 seconds
Memory Usage: 29.87 MB
User CPU Time: 0.187500 seconds
System CPU Time: 0.078125 seconds

Matrix size: 128x128
Execution Time: 0.000497 seconds
Memory Usage: 30.40 MB
User CPU Time: 0.187500 seconds
System CPU Time: 0.078125 seconds

Matrix size: 1024x1024
Execution Time: 0.070671 seconds
Memory Usage: 55.76 MB
User CPU Time: 0.296875 seconds
System CPU Time: 0.078125 seconds
```

Multiplication using Numpy.matmul



```
>>
PS C:\Users\Administrator\Documents\BigData> java Matrix2
Execution Time: 0.0 seconds
Memory Usage: 24 MB
CPU Time: 62 milliseconds
Execution Time: 0.032 seconds
Memory Usage: 25 MB
CPU Time: 62 milliseconds
Execution Time: 2.299 seconds
Memory Usage: 25 MB
CPU Time: 2406 milliseconds
```

Multiplication using intuitive basic approach



```
PS C:\Users\Administrator\Documents\BigData> java Matrix3
Matrix size: 16x16
Execution Time: 0.0 seconds
Memory Usage: 2 MB
CPU Time: 109 milliseconds
Matrix size: 128x128
Execution Time: 0.365 seconds
Memory Usage: 6 MB
CPU Time: 437 milliseconds
Matrix size: 1024x1024
Execution Time: 101.355 seconds
Memory Usage: 46 MB
CPU Time: 97937 milliseconds
```

Multiplication using Strassen's Algorithm