
CS 301

High-Performance Computing

Lab Report-02

Matrix Multiplication

Performance Analysis

Atik Vohra (202301447)
Soham Mevada (202301484)

Contents

1	Introduction	3
2	Hardware Details	3
2.1	LAB207 PCs	3
2.2	HPC Cluster	3
3	Methodology	4
4	Matrix Multiplication	4
4.1	Problem Size vs Time	4
4.1.1	Local System	4
4.1.2	HPC Cluster	4
5	Transpose Method	7
6	Block Matrix Multiplication	7
7	Observations	7
8	Conclusion	8

1 Introduction

This report analyzes the performance of matrix multiplication algorithms on local and HPC cluster systems. We compare different loop orderings, transpose-based optimization, and block-based multiplication.

Performance is evaluated using execution time for problem sizes ranging from 2^1 to 2^{12} .

2 Hardware Details

2.1 LAB207 PCs

- Architecture: x86_64
- CPU(s): 12
- Threads per core: 2
- Cores per socket: 6
- Processor: Intel Core i5-12500
- L1 Cache: 288K
- L2 Cache: 7.5M
- L3 Cache: 18M

2.2 HPC Cluster

- Architecture: x86_64
- CPU(s): 16
- Threads per core: 1
- Cores per socket: 8
- Sockets: 2
- Processor: Intel Xeon E5-2640 v3
- L1 Cache: 32K
- L2 Cache: 256K
- L3 Cache: 20480K

3 Methodology

Matrix multiplication is implemented using:

- Six loop orderings
- Transpose optimization
- Block multiplication

Execution time is measured and converted to logarithmic scale:

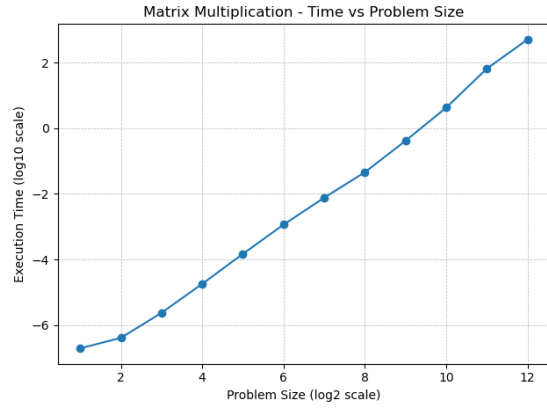
$$C[i][j] = \sum_{k=0}^{n-1} A[i][k] \times B[k][j] \quad (1)$$

4 Matrix Multiplication

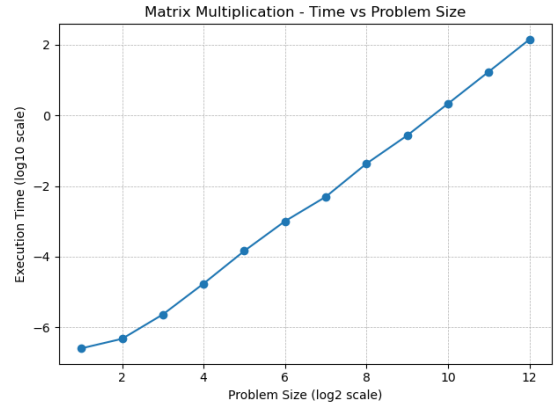
4.1 Problem Size vs Time

4.1.1 Local System

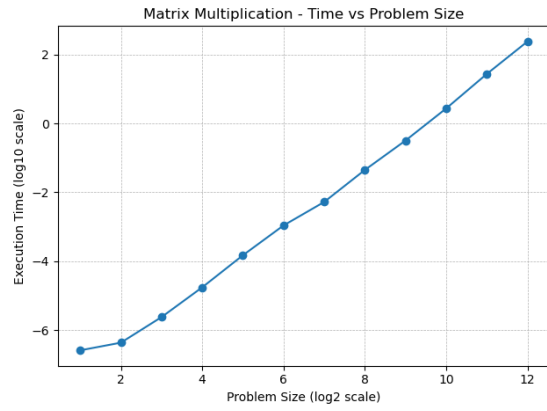
4.1.2 HPC Cluster



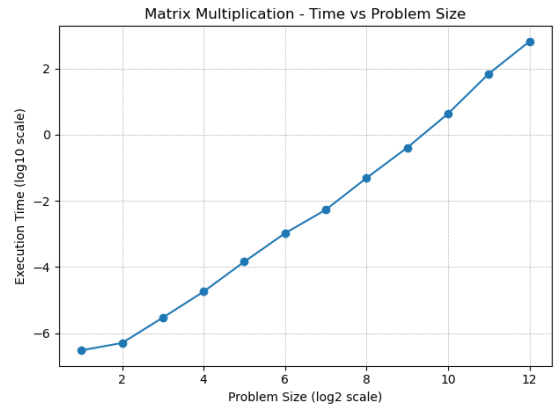
(a) IJK



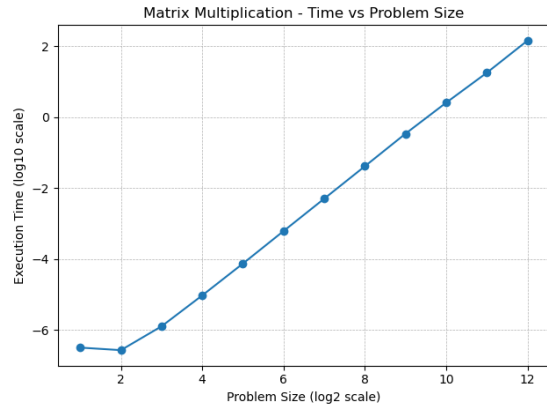
(b) IKJ



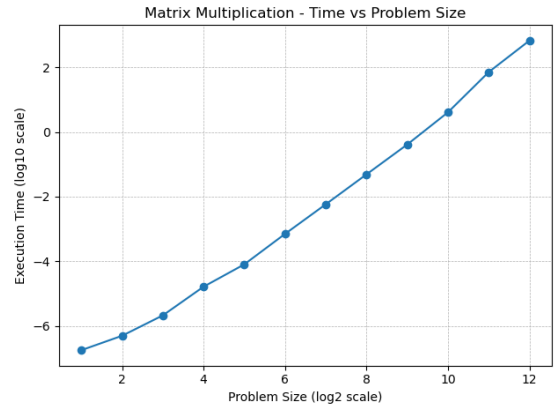
(c) JIK



(d) JKI

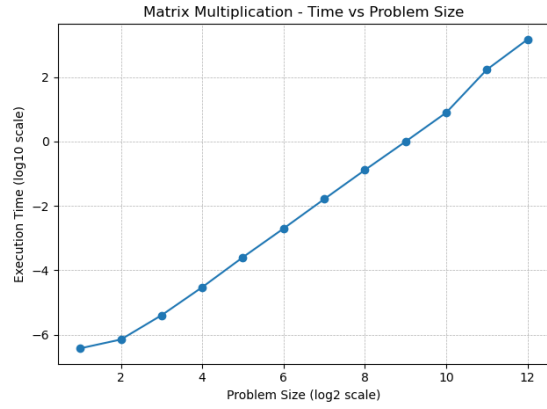


(e) KIJ

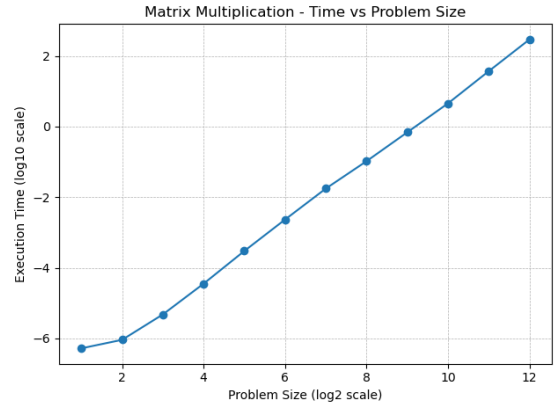


(f) KJI

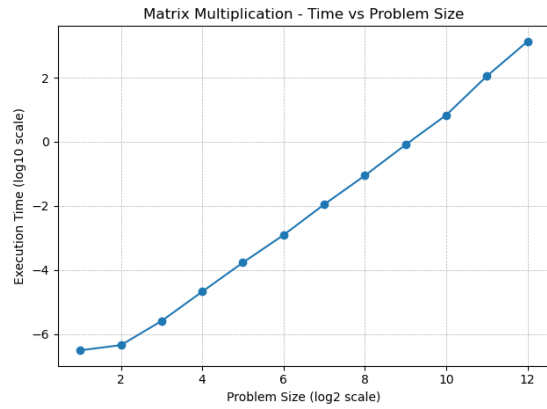
Figure 1: Execution Time vs Problem Size (Local System)



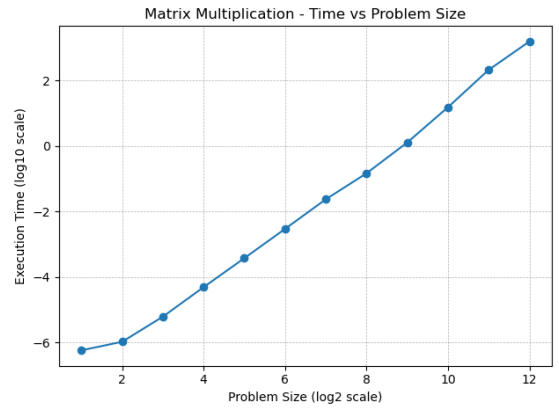
(a) IJK



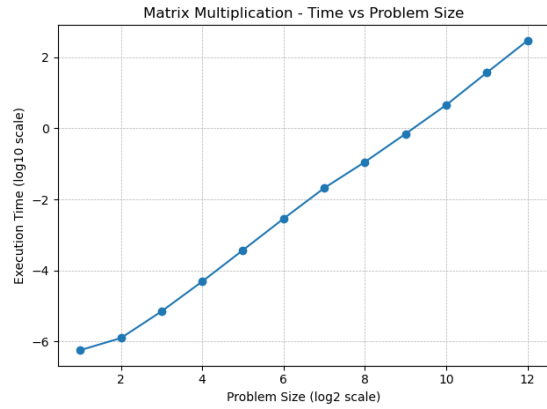
(b) IKJ



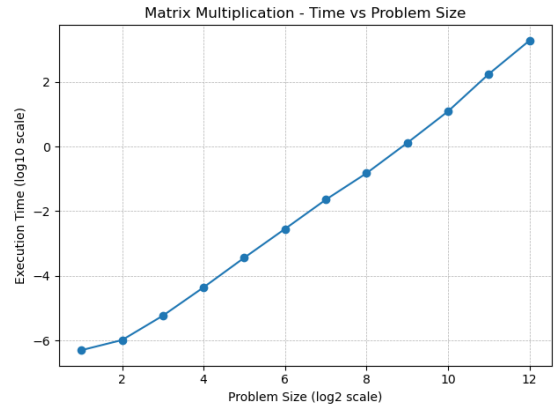
(c) JIK



(d) JKI



(e) KIJ



(f) KJI

Figure 2: Execution Time vs Problem Size (HPC Cluster)

5 Transpose Method

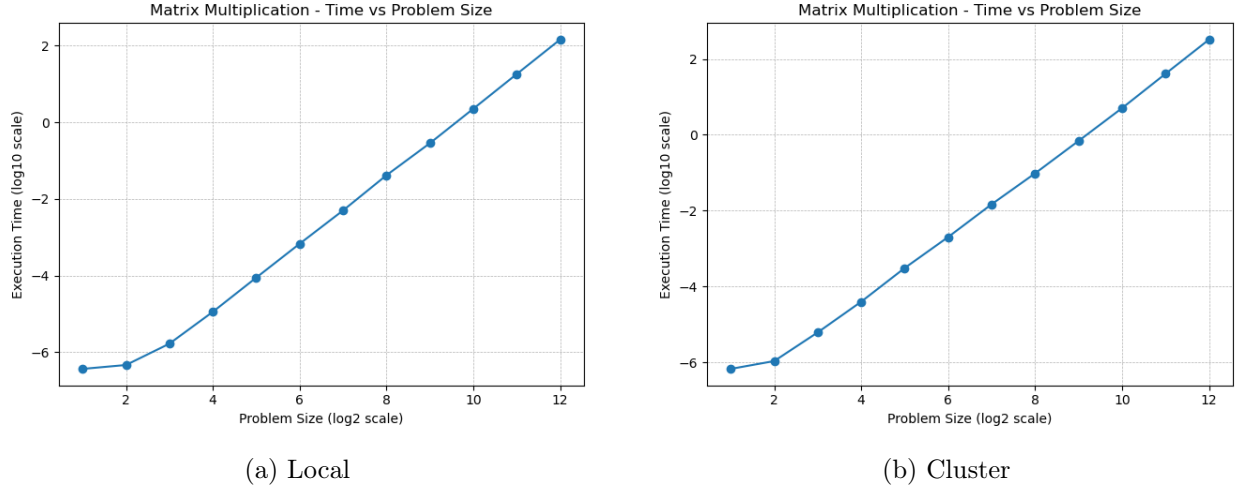


Figure 3: Transpose Method Performance

6 Block Matrix Multiplication

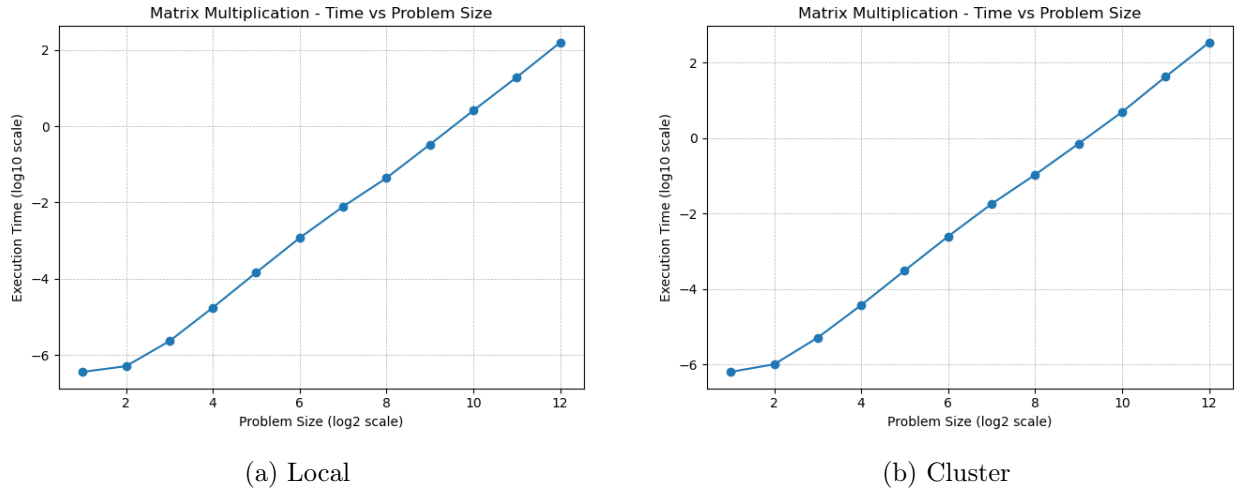


Figure 4: Block Method Performance

7 Observations

- Loop ordering strongly affects cache performance.
- IKJ provide best locality and performance.

- Transpose improves memory access.
- Blocking enhances cache reuse, and block size can be tuned to fit size of the working CPU and memory.

8 Conclusion

Memory access patterns play a crucial role in matrix multiplication performance. Loop optimization, transposition, and blocking significantly improve runtime, especially on HPC systems.