Problem 2 Report: Parallel Performance of Matrix Multiplication

Joshua BRIONNE - 50241647

🧰 Compilation & Execution

To avoid installing Java manually, this project includes a `Dockerfile` for a consistent Java development environment.

🐳 Using Docker (Recommended)

1. Install Docker by following these instructions.
2. Build and run the project using:

```
make run
./tester.sh <NameOfTheJavaFile>

#  If Java is Already Installed
# You can directly compile and run the Java file on your machine:

javac <NameOfTheJavaFile>.java
java <NameOfTheJavaFile>
```

🏗️ 1. Environment

All tests were executed inside a Docker container using the following image:

```
FROM openjdk:21-slim
```

Property	**Value**
Java Version	21
OS Name	Linux
OS Version	6.8.0-52-generic
Architecture	amd64
Available processors (cores)	8
Max memory (MB)	3936

📊 2. Benchmark Results

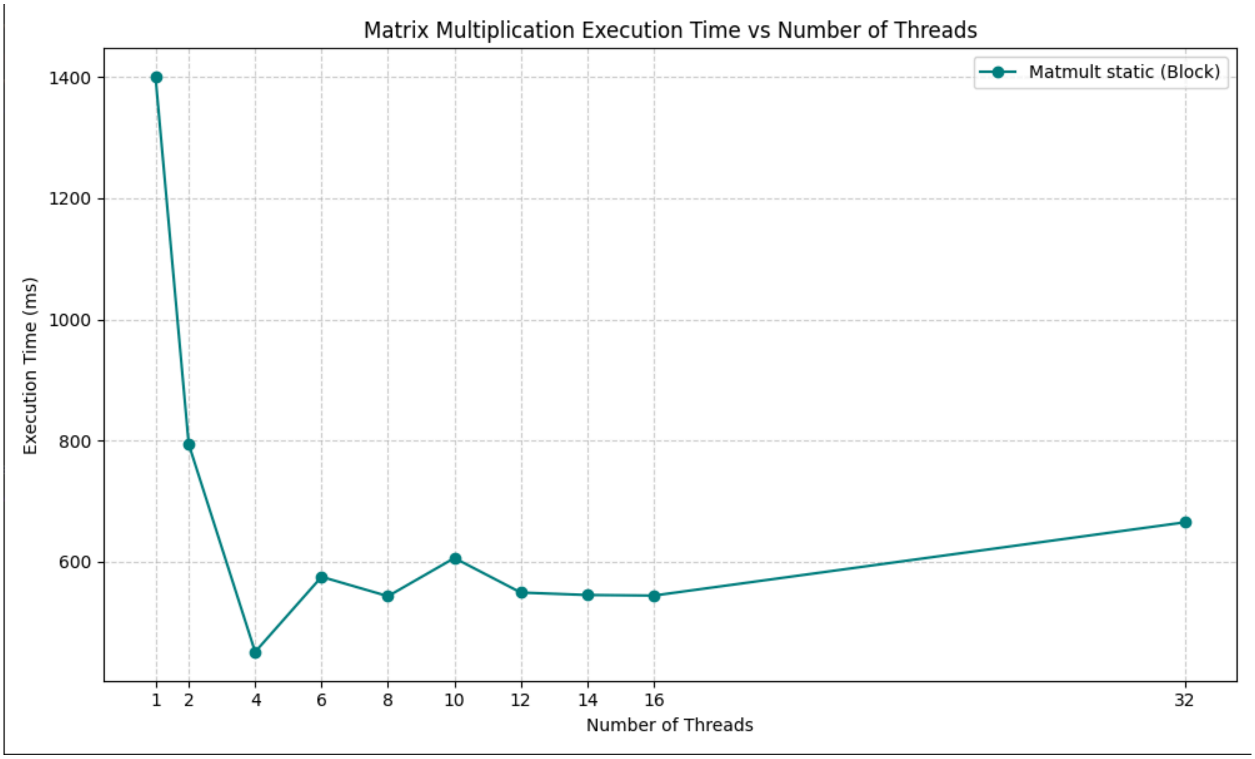
🔁 Serial Execution

Threads	**1**	**2**	**4**	**6**	**8**	**10**	**12**	**14**	**16**	**32**
Default serial (ms)	2016	2206	2128	1985	2081	1964	2150	2071	2180	1991

> The serial implementation doesn’t benefit from multiple threads, so the time remains roughly constant.

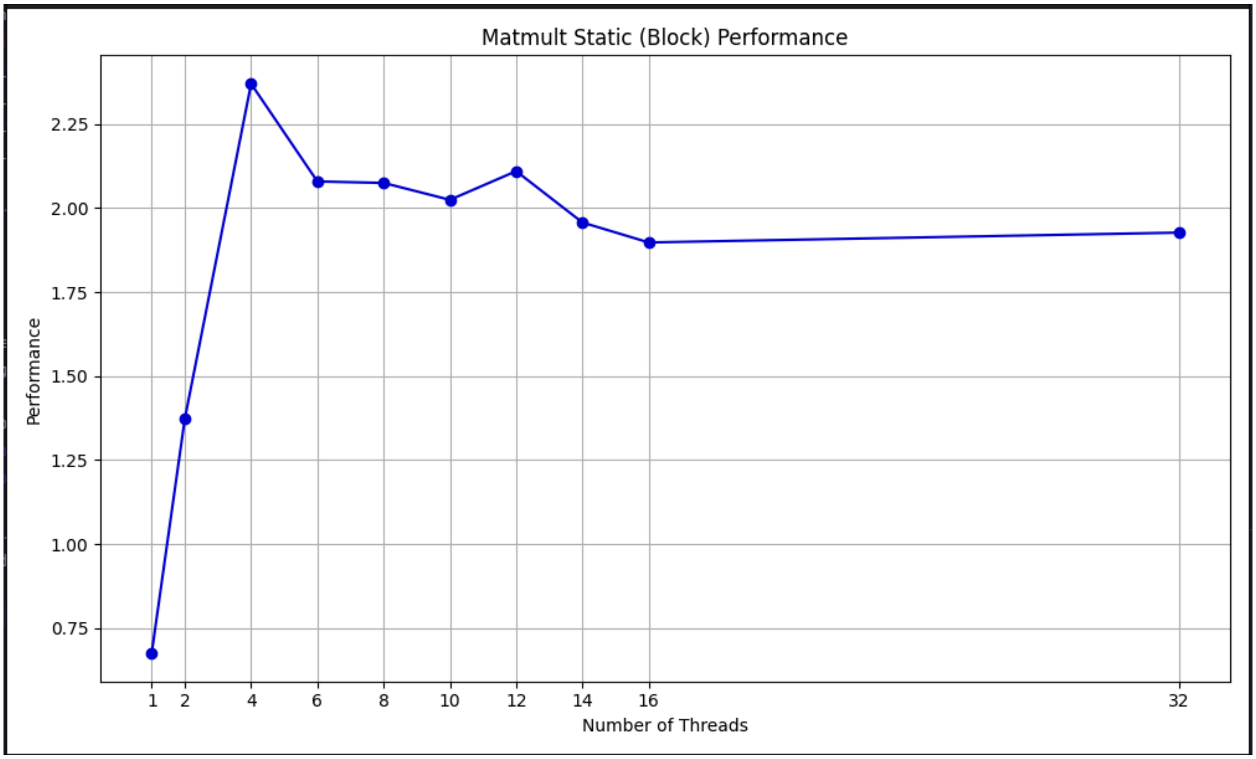
I used the Static block approach to multi thread this java code

🏎️ Parallel Execution Time (ms)



Threads	**1**	**2**	**4**	**6**	**8**	**10**	**12**	**14**	**16**	**32**
Matmult static (Block ms)	1400	794	451	575	543	606	549	545	544	665

⚡ Performance (1 / execution time in s)



Threads	**1**	**2**	**4**	**6**	**8**	**10**	**12**	**14**	**16**	**32**
Matmult static (Block)	0.68	1.37	2.37	2.08	2.07	2.02	2.11	1.96	1.90	1.93

🔍 3. Analysis & Interpretation

🖥️ CPU Information

Property	**Value**
Architecture	x86_64
CPU op-mode(s)	32-bit, 64-bit
CPU(s)	8
Core(s) per socket	4
Thread(s) per core	2
Socket(s)	1
Vendor ID	GenuineIntel
Model name	11th Gen Intel(R) Core(TM) i7-11370H
Base Frequency	3.30 GHz
Max Turbo Frequency	4.80 GHz
L1d Cache	192 KiB (4 instances)
L1i Cache	128 KiB (4 instances)
L2 Cache	5 MiB (4 instances)
L3 Cache	12 MiB (1 instance)
Hyperthreading	Enabled
Virtualization	VT-x
NUMA Node(s)	1
NUMA node0 CPU(s)	0-7

📊 5. Interpretation of Parallel Results

Analysis of Static Block Parallelization Based on CPU Architecture

The performance of the static block matrix multiplication was evaluated across different thread counts on a system equipped with an **11th Gen Intel® Core™ i7-11370H** processor. This CPU features **4 physical cores** and supports **hyperthreading**, allowing up to **8 logical threads**.

Results show a **significant performance improvement** as the number of threads increases from **1 to 4**, indicating efficient parallel utilization of the available physical cores. However, beyond 4 threads, the performance gain **diminishes**, and in some cases **slightly regresses**. This plateau corresponds to the use of **hyperthreads**, which share physical core resources and thus offer **limited performance benefits** for compute-intensive tasks.

When using more than 8 threads, **performance begins to degrade** due to **thread oversubscription**, leading to increased **overhead from context switching** and **resource contention**.

These findings demonstrate that **static block decomposition achieves optimal performance when the number of threads aligns with the number of physical cores**, and that hyperthreading and excessive parallelism can introduce **inefficiencies** for this particular workload.