1 Ubuntu Virtual Machine Specifications

- **Host Machine**: Windows 11
- **Virtualization Software**: VirtualBox
- **Base System**:
  - ✓ **Device Name**: DESKTOP-LRMR5I7
  - ✓ **Processor**: Intel® Core™ i5-8350U CPU @ 1.70GHz (4 Cores allocated)
  - ✓ **Installed RAM**: 8GB (7.86GB usable)
- **Virtual Machine (Ubuntu LTS) Configuration**:
  - ✓ **Allocated CPU Cores**: 4
  - ✓ **Allocated RAM**: 4GB
  - ✓ **Disk Space**: 25 GB
  - ✓ **Graphics Acceleration**: *VBoxSVGA*

2 Result Summary

1. Thread Scaling

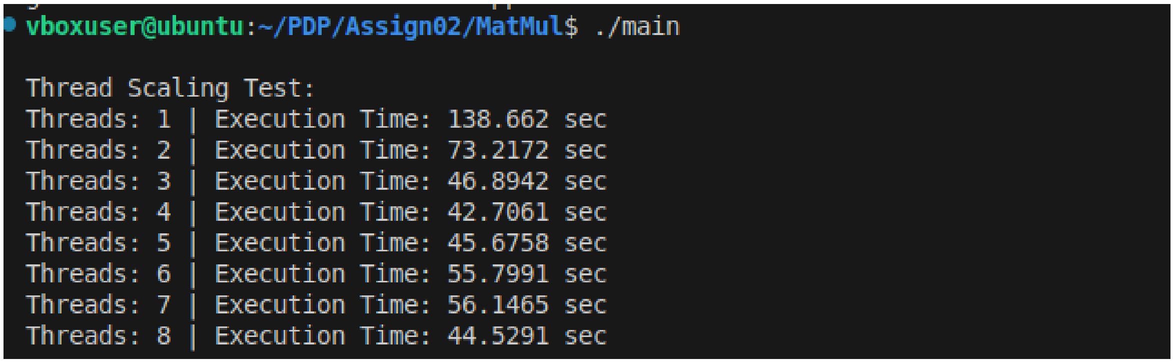
- Execution time **decreases** with more threads up to the allocated CPU cores.
- Beyond this, slight overhead increases occur but drop again at multiples of core counts.

2. Matrix Scaling

- Execution time **grows exponentially** with matrix size.
- Larger matrices face memory bandwidth and cache limitations, impacting performance

3 Tread Scaling Performance

- The execution time **decreases** as the number of threads increases, up to a certain point, likely corresponding to the number of allocated CPU cores.
- Beyond this point, the execution time slightly **increases** due to factors such as thread scheduling overhead, memory bandwidth limitations, or context switching.
- However, at specific multiples of the allocated cores, execution time decreases again, suggesting that the workload is better distributed across **logical cores** .



```
vboxuser@ubuntu:~/PDP/Assign02/MatMul$ ./main

Thread Scaling Test:
Threads: 1 | Execution Time: 138.662 sec
Threads: 2 | Execution Time: 73.2172 sec
Threads: 3 | Execution Time: 46.8942 sec
Threads: 4 | Execution Time: 42.7061 sec
Threads: 5 | Execution Time: 45.6758 sec
Threads: 6 | Execution Time: 55.7991 sec
Threads: 7 | Execution Time: 56.1465 sec
Threads: 8 | Execution Time: 44.5291 sec
```

Figure 1: Thread Scaling Test Console Output

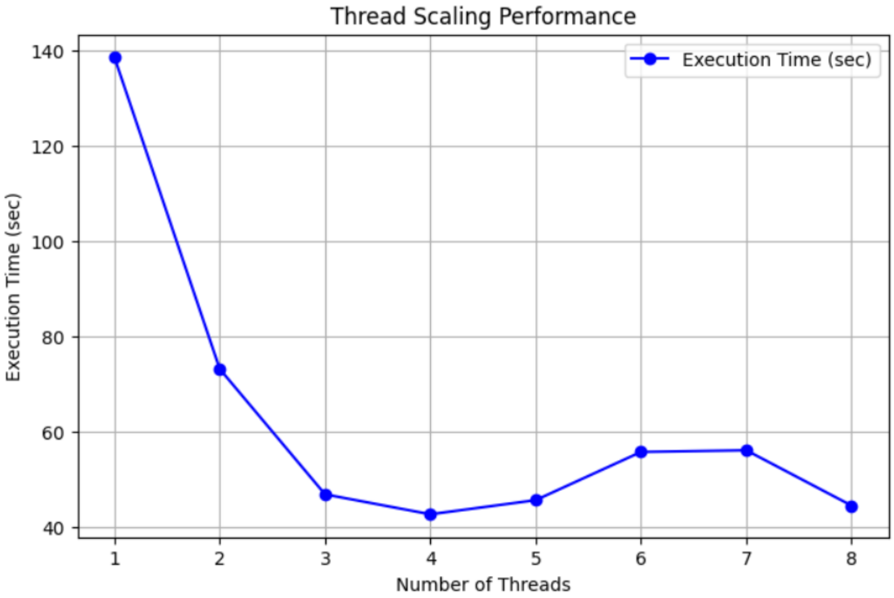
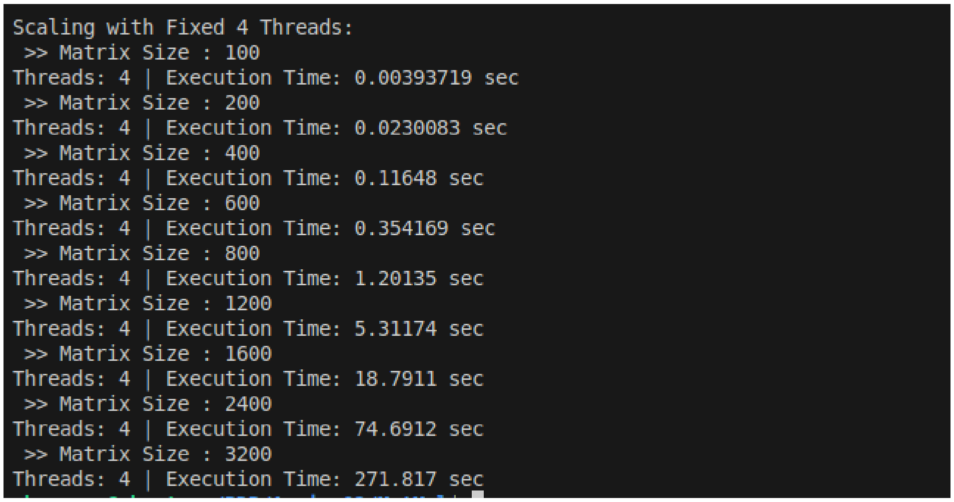


Figure 2: Graph of Thread Scaling Test

4 Matrix Scaling Performance (On 4 Threads)

- As the matrix size increases, execution time **grows significantly**, following an approximately exponential trend.
- The smaller matrix sizes execute in negligible time, but as matrix size increases, computation demands rise sharply, leading to a drastic increase in execution time.
- This behavior aligns with the computational complexity of matrix multiplication, which is typically **O(n^3)** for naive implementations.
- The rapid increase in execution time for larger matrices suggests memory bandwidth and cache limitations start to impact performance.



```
Scaling with Fixed 4 Threads:
 >> Matrix Size : 100
Threads: 4 | Execution Time: 0.00393719 sec
 >> Matrix Size : 200
Threads: 4 | Execution Time: 0.0230083 sec
 >> Matrix Size : 400
Threads: 4 | Execution Time: 0.11648 sec
 >> Matrix Size : 600
Threads: 4 | Execution Time: 0.354169 sec
 >> Matrix Size : 800
Threads: 4 | Execution Time: 1.20135 sec
 >> Matrix Size : 1200
Threads: 4 | Execution Time: 5.31174 sec
 >> Matrix Size : 1600
Threads: 4 | Execution Time: 18.7911 sec
 >> Matrix Size : 2400
Threads: 4 | Execution Time: 74.6912 sec
 >> Matrix Size : 3200
Threads: 4 | Execution Time: 271.817 sec
```

Figure 3: Matrix Scaling Test Console Output

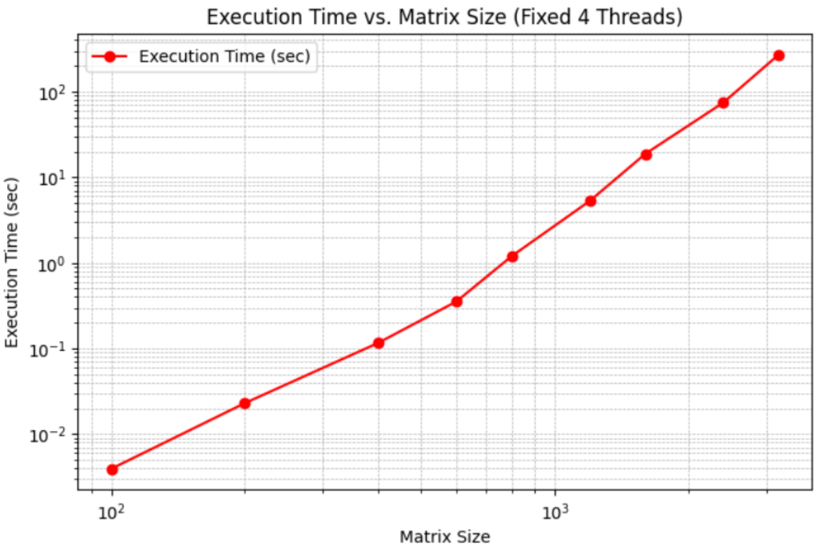


Figure 4: Log Scale Graph for Matrix Scale Testing