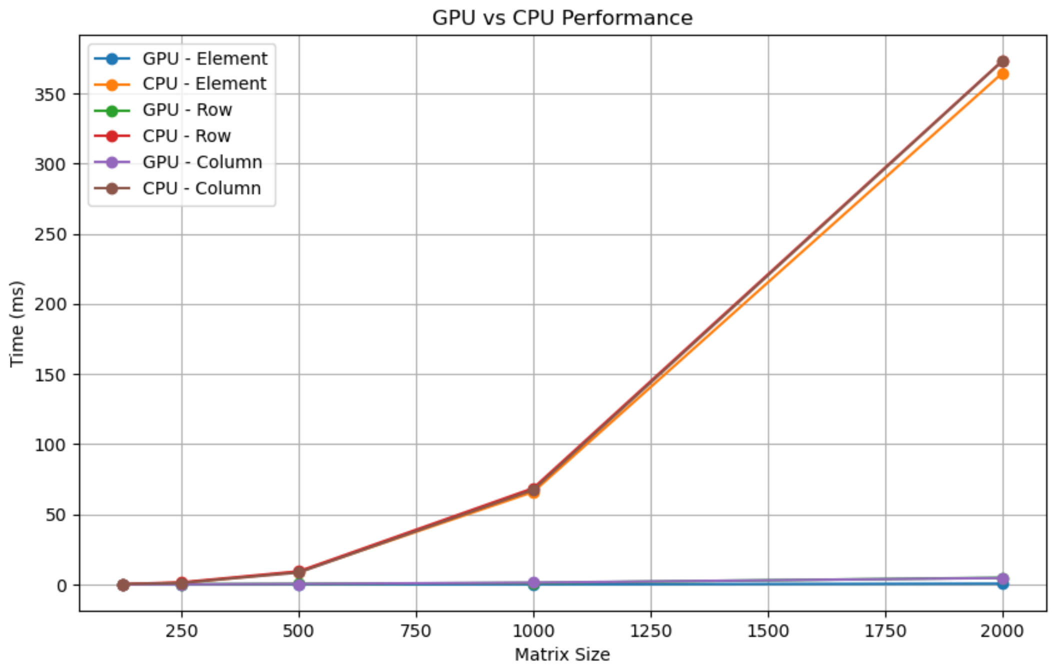
Machine Problem 2

Abdellah Ghassel (20230384)

“I do hereby verify that this machine problem submission is my own work and contains my own original ideas, concepts, and designs. No portion of this report or code has been copied in whole or in part from another source, with the possible exception of properly referenced material".



Problem Breakdown

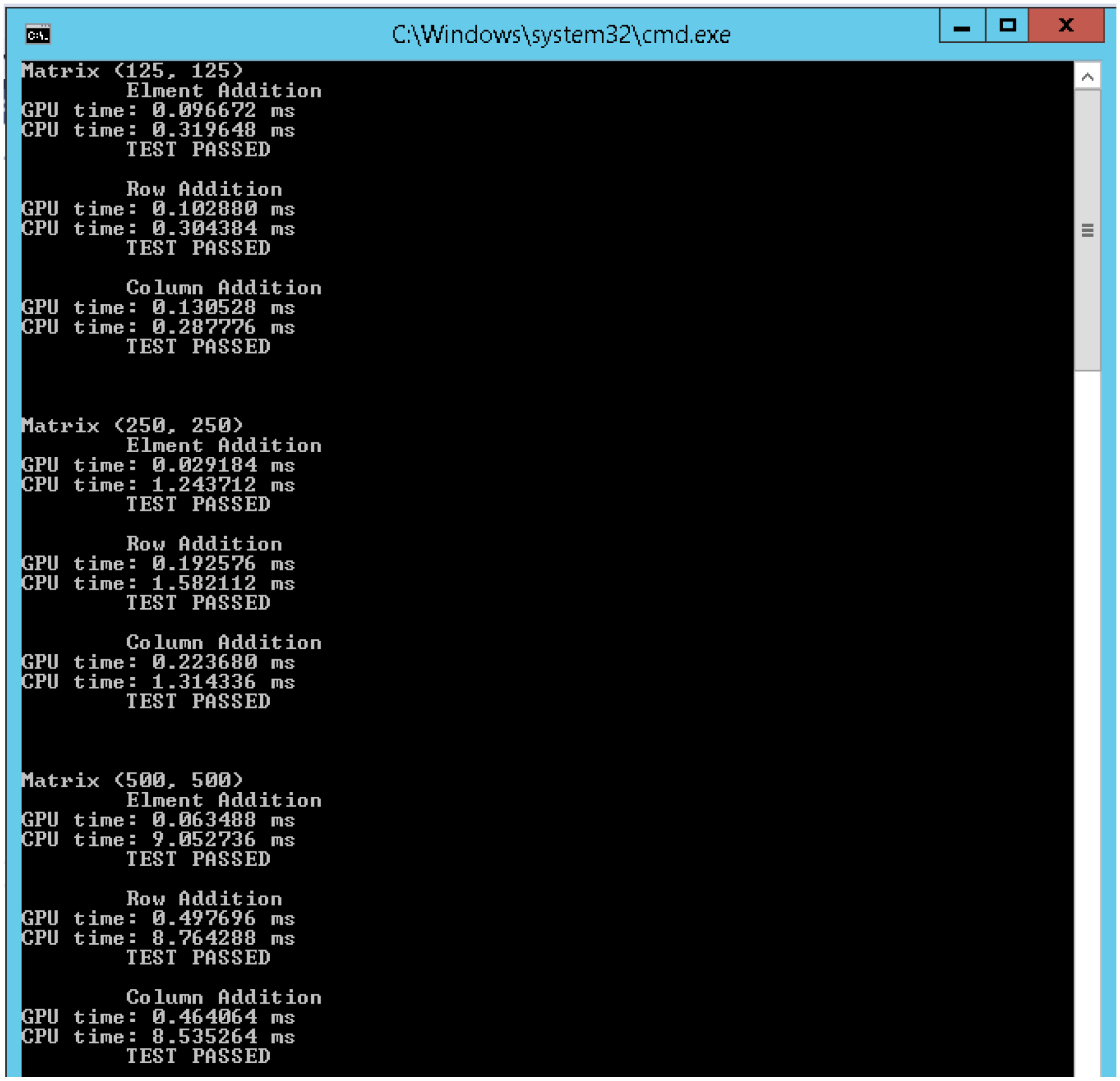
This is a CUDA program to add two matrices together in three different ways: element-wise, row-wise, and column-wise. Its main goal is to compare the performance of the GPU and CPU for each addition method.

The program iterates through different matrix sizes, initializes the matrices A, B, and C, and calls the hostFunc() function with each addition method.

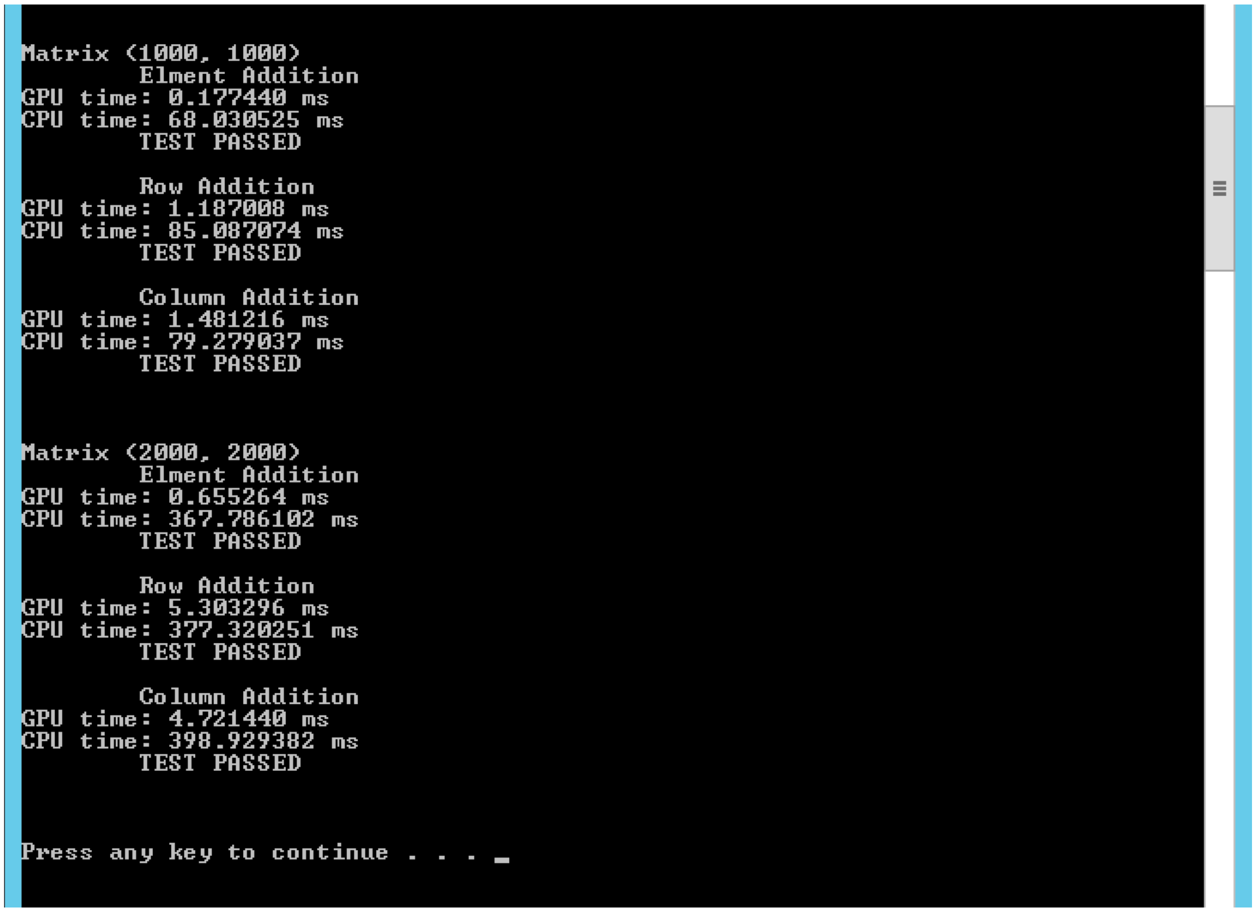
hostFunc() initializes the matrices with random values, allocates memory on the GPU, and measures the time taken for each GPU addition method. It then compares this time with the time taken for the CPU to perform the same addition. This function also checks for errors between the GPU and CPU results and prints whether the test passed or failed based on the error tolerance.

There are three kernel functions that handle matrix addition on the GPU, each corresponding to one of the three addition methods: matrixAddElementKernel(), matrixAddRowKernel(), and matrixAddColKernel().

There are also three wrapper functions that configure the launch parameters for each kernel and execute them: getMatrixElementAddition(), getMatrixRowAddition(), and getMatrixColAddition().



```
Matrix (125, 125)
        Elment Addition
GPU time: 0.096672 ms
CPU time: 0.319648 ms
        TEST PASSED

        Row Addition
GPU time: 0.102880 ms
CPU time: 0.304384 ms
        TEST PASSED

        Column Addition
GPU time: 0.130528 ms
CPU time: 0.287776 ms
        TEST PASSED


Matrix (250, 250)
        Elment Addition
GPU time: 0.029184 ms
CPU time: 1.243712 ms
        TEST PASSED

        Row Addition
GPU time: 0.192576 ms
CPU time: 1.582112 ms
        TEST PASSED

        Column Addition
GPU time: 0.223680 ms
CPU time: 1.314336 ms
        TEST PASSED


Matrix (500, 500)
        Elment Addition
GPU time: 0.063488 ms
CPU time: 9.052736 ms
        TEST PASSED

        Row Addition
GPU time: 0.497696 ms
CPU time: 8.764288 ms
        TEST PASSED

        Column Addition
GPU time: 0.464064 ms
CPU time: 8.535264 ms
        TEST PASSED
```



```
Matrix (1000, 1000)
        Elment Addition
GPU time: 0.177440 ms
CPU time: 68.030525 ms
        TEST PASSED

        Row Addition
GPU time: 1.187008 ms
CPU time: 85.087074 ms
        TEST PASSED

        Column Addition
GPU time: 1.481216 ms
CPU time: 79.279037 ms
        TEST PASSED


Matrix (2000, 2000)
        Elment Addition
GPU time: 0.655264 ms
CPU time: 367.786102 ms
        TEST PASSED

        Row Addition
GPU time: 5.303296 ms
CPU time: 377.320251 ms
        TEST PASSED

        Column Addition
GPU time: 4.721440 ms
CPU time: 398.929382 ms
        TEST PASSED


Press any key to continue . . .
```

Analysis

1. GPU performance is consistently better than CPU performance for all matrix sizes and addition methods (element-wise, row-wise, and column-wise). As the matrix size increases, the difference in performance between the GPU and CPU also becomes more significant.

2. As the matrix size increases, the time taken for both GPU and CPU computations increases. However, the rate of increase is much higher for CPU computations, indicating that the GPU scales better with larger matrix sizes.

3. Comparing the three different addition methods (element-wise, row-wise, and column-wise), we can see that the GPU and CPU have similar relative performance across these methods. For both GPU and CPU, row-wise and column-wise addition methods have a slightly higher computation time compared to the element-wise method. This is because row-wise and column-wise methods involve more complex memory access patterns, which can lead to a more overhead.

Code for Python Visualization Script:

Note: this data is different than the terminal screenshots above since the program was run again.

```
import matplotlib.pyplot as plt

# Data
matrix_sizes = [125, 250, 500, 1000, 2000]

gpu_times_element = [0.091392, 0.037088, 0.063904, 0.177088, 0.670720]
cpu_times_element = [0.314144, 1.182528, 9.074432, 66.057373, 364.438110]

gpu_times_row = [0.118976, 0.200448, 0.499424, 1.184864, 4.909344]
cpu_times_row = [0.315264, 1.629344, 9.461792, 68.714973, 373.171539]

gpu_times_col = [0.138880, 0.211648, 0.465568, 1.409856, 4.721728]
cpu_times_col = [0.301856, 1.203072, 8.594240, 67.487038, 372.913849]

# Plotting
plt.figure(figsize=(10, 6))
plt.plot(matrix_sizes, gpu_times_element, '-o', label='GPU - Element')
plt.plot(matrix_sizes, cpu_times_element, '-o', label='CPU - Element')
plt.plot(matrix_sizes, gpu_times_row, '-o', label='GPU - Row')
plt.plot(matrix_sizes, cpu_times_row, '-o', label='CPU - Row')
plt.plot(matrix_sizes, gpu_times_col, '-o', label='GPU - Column')
plt.plot(matrix_sizes, cpu_times_col, '-o', label='CPU - Column')

plt.xlabel('Matrix Size')
plt.ylabel('Time (ms)')
plt.title('GPU vs CPU Performance')
plt.legend()
plt.grid()
plt.show()
```