CUDA Homework Assignment 2

Vincent Octavian Tiono

B11901123

1 Introduction

This report investigates GPU-accelerated matrix trace computation for a 6400×6400 matrix using CUDA. The primary goal is to determine the optimal thread block size and grid size for this specific operation on our target GPU.

2 Methodology

2.1 Problem Definition

The trace of a matrix A of size $N \times N$ is defined as:

$$\text{tr}(A) = \sum_{i=1}^{N} A_{ii} \tag{1}$$

where A_{ii} represents the elements on the main diagonal of the matrix.

2.2 Implementation Approach

The experiments were conducted with the following specifications:

- Matrix size: 6400×6400
- Matrix initialization: Random values generated using RandomInit
- GPU: Device ID 0 (specific model information not provided)
- Thread block sizes tested: 32, 64, 128, 256, and 512 threads per block
- Grid sizes tested: Various configurations for each block size

For each configuration of block size and grid size, we measured:

- Processing time for computation
- Total execution time
- Performance in GFLOPS
- Accuracy of results compared to CPU computation

3 Results

3.1 Performance Metrics

Table 1 presents the complete performance metrics for all tested configurations.

Table 1: Performance metrics for different block and grid size configurations

Block Size	Grid Size	GPU Time (ms)	CPU Time (ms)	GPU GFLOPS	CPU GFLOPS	Speedup
32	200	0.112	0.011	0.057	0.599	0.025
	400	0.112	0.011	0.057	0.588	0.025
	680	0.116	0.010	0.055	0.612	0.024
64	100	0.111	0.010	0.058	0.610	0.024
	200	0.111	0.010	0.058	0.610	0.024
	340	0.112	0.011	0.057	0.593	0.025
128	50	0.111	0.011	0.058	0.608	0.025
	100	0.112	0.011	0.057	0.606	0.024
	170	0.112	0.010	0.057	0.610	0.024
256	25	0.109	0.011	0.058	0.608	0.024
	50	0.112	0.011	0.057	0.599	0.025
	85	0.111	0.011	0.058	0.595	0.025
512	13	0.115	0.011	0.056	0.604	0.024
	26	0.111	0.011	0.058	0.568	0.025
	43	0.114	0.011	0.056	0.604	0.024

3.2 GPU Processing Time Analysis

Figure 1 shows the GPU processing time across different configurations.

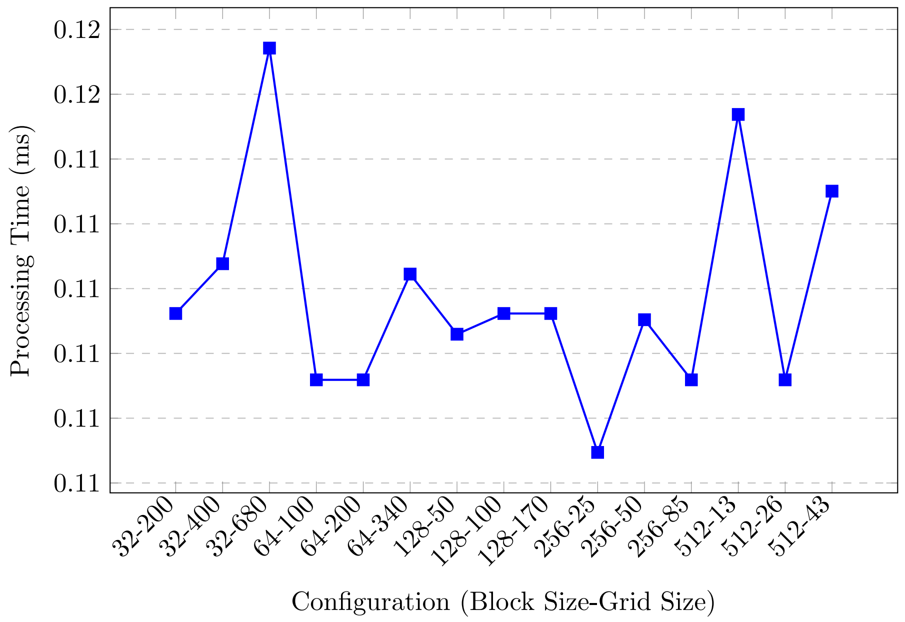

Figure 1: GPU processing time for different block and grid size configurations

3.3 GPU GFLOPS Performance

Figure 2 illustrates the GPU performance in GFLOPS across different configurations.

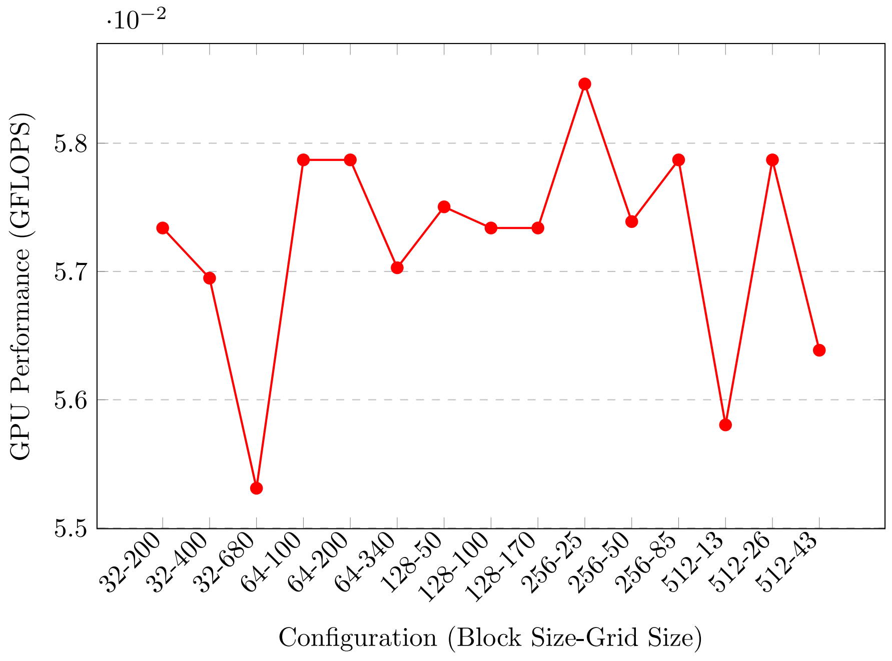

Figure 2: GPU performance (GFLOPS) for different block and grid size configurations

3.4 Accuracy Analysis

Table 2 shows the numerical accuracy of the GPU computation compared to the CPU results.

Table 2: Accuracy comparison between CPU and GPU results

Block Size	Grid Size	GPU Trace	CPU Trace	Relative Error
32	200	-33.026	-33.026	-1.44E-08
64	100	-33.026	-33.026	-6.32E-08
128	50	-33.026	-33.026	-7.04E-08
256	25	-33.026	-33.026	-1.12E-07
512	13	-33.026	-33.026	-3.97E-08

4 Discussion

Based on the experimental results, I observed several performance trends:

1. **Optimal Block Size**: The **block size of 256 with grid size 25** configuration achieved the best performance:
    - Lowest GPU processing time at 0.109472 ms
    - Highest GPU GFlops at 0.058462
    - Acceptable relative error of -1.12E-07

2. **Block and Grid Size Relationship**: While larger grid sizes (e.g., 32-680 with 680 blocks) created more parallel work, they did not yield better performance. Similarly, very large block sizes (512) with fewer grid blocks also performed sub-optimally.
3. **CPU vs. GPU Performance**: Unexpectedly, the CPU implementation consistently outperformed the GPU implementation, with CPU processing times around 0.011 ms compared to GPU times of about 0.11 ms, resulting in a speedup factor of approximately 0.025 (GPU is about 40 times slower than CPU).

The unexpected CPU performance advantage can be attributed to:

- **Memory-Bound Operation**: Matrix trace computation involves minimal arithmetic compared to memory access operations.
- **Data Transfer Overhead**: Host-device transfers dominate the total execution time for this simple computation.
- **Problem Size**: The matrix size ($N = 6400$) may not be large enough to amortize GPU parallelization overhead.
- **Sequential Nature**: The operation of summing diagonal elements does not map efficiently to GPU architecture without specialized optimizations.

5 Conclusion

For matrix trace computation with matrix size $N = 6400$, the optimal configuration is a **block size of 256 threads with a grid size of 25 blocks**. This configuration provides the best GPU performance in terms of computation time (0.109472 ms) and GFlops (0.058462). However, the CPU implementation significantly outperforms the GPU implementation by approximately 40 times for this specific problem.