CUDA Homework Assignment 1

Vincent Octavian Tiono

B11901123

1 Introduction

This report analyzes the performance of matrix reciprocal sum operation defined as $C(i,j) = \frac{1}{A(i,j)} + \frac{1}{B(i,j)}$. The experiment uses two input $N \times N$ matrices with random values between 0.0 and 1.0, where $N = 6400$. The primary goal is to determine the optimal thread block size for this specific operation on our target GPU.

2 Methodology

I tested the matrix operation with square thread block configurations of sizes 8×8, 16×16, 24×24, 28×28, and 32×32. For each configuration, I measured:

- GPU compute time
- GPU GFlops performance
- Total GPU time
- Resulting error compared to CPU calculation

All experiments were performed on a single GPU (device ID: 0).

3 Results

3.1 Performance Measurements

Table 1: Performance comparison of different block sizes

Block Size	Number of Blocks	GPU Compute Time (ms)	GPU GFlops	Total GPU Time (ms)	CPU Time (ms)
8×8	640,000	12.95	9.49	52.01	305.29
16×16	160,000	11.67	10.53	50.67	305.72
24×24	71,289	13.34	9.21	52.32	305.32
28×28	52,441	13.64	9.01	52.80	305.24
32×32	40,000	13.53	9.08	52.58	305.46

3.2 Speedup Analysis

The speedup of GPU over CPU computation was calculated based on the processing time:

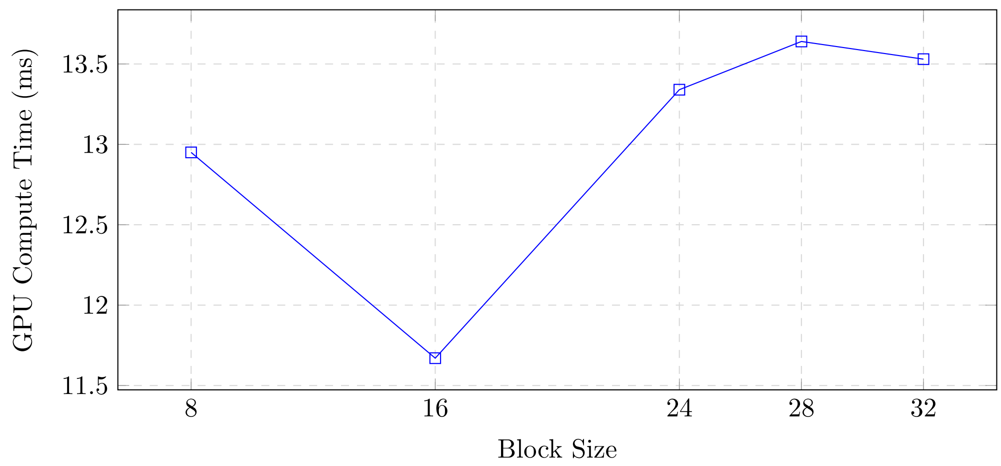

Figure 1: GPU compute time for different block sizes

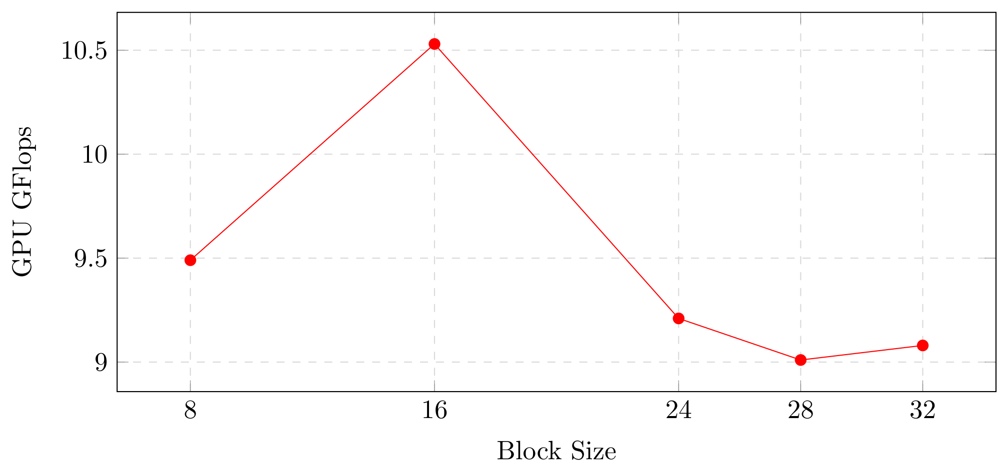

Figure 2: GPU GFlops for different block sizes

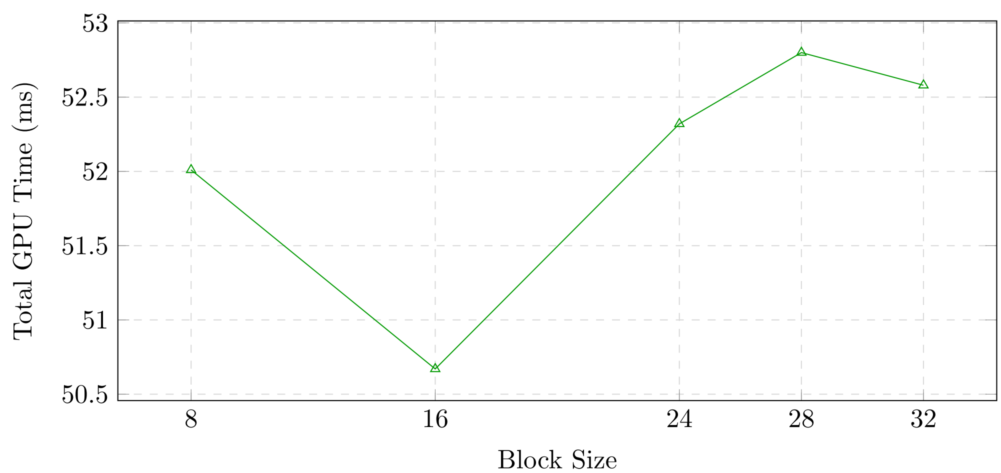

Figure 3: Total GPU time for different block sizes

Table 2: GPU vs. CPU performance speedup

Block Size	Speedup (CPU time / GPU total time)
8×8	5.87
16×16	6.03
24×24	5.84
28×28	5.78
32×32	5.81

4 Discussion

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

1. **Optimal Block Size**: The $\mathbf{16 \times 16}$ block size configuration consistently outperformed all other tested configurations:
   - Lowest GPU compute time at 11.67 ms
   - Highest GPU GFlops at 10.53
   - Lowest total GPU time at 50.67 ms
   - Best speedup over CPU at 6.03x
2. **Number of Blocks vs. Performance**: While the 8×8 configuration created the largest number of blocks (640,000), it did not yield the best performance. Similarly, the 32×32 configuration with the fewest blocks (40,000) also didn't perform optimally.

The performance advantage of the 16×16 configuration can be attributed to:

- **Warp Alignment**: This size (256 threads per block) aligns well with the GPU warp size (32), allowing for efficient thread scheduling.
- **Occupancy Balance**: It provides a good balance between having too many small blocks (8×8) which increases scheduling overhead, and too few large blocks (32×32) which may reduce parallelism.

5 Conclusion

For the matrix reciprocal sum operation with matrix size $N = 6400$, the optimal thread block size is $\mathbf{16 \times 16}$. This configuration provides the best performance in terms of computation time, GFlops, and overall efficiency. The GPU implementation with this block size achieves approximately a 6x speedup over the CPU implementation, demonstrating the significant benefits of parallel computing for this operation.