CUDA Homework Assignment 2

Hao-Tse , Hsiao

B12902100

Abstract. In Homework Assignment 2, we are required to find the trace of a matrix of real numbers using parallel reduction. We compute the trace using two methods: first, by executing the calculation on a CPU, and second, by utilizing a GPU with parallel reduction. Finally, we determine the optimal block and grid size configurations to maximize computational efficiency.

1 Task Description

We begin by initializing an $N \times N$ matrix A with random real values in the range $[-1, 1]$. The trace of the matrix, defined as $\text{trace}(A) = \sum_{i=0}^{N-1} A_{ii}$, is computed using both CPU and GPU implementations. We use parallel reduction on the GPU to efficiently accumulate the diagonal elements.

2 Task Result

After performing the matrix computation on both the CPU and GPU, we compiled the results in the following table. The numbers in parentheses indicate the number of threads per block and the number of blocks per grid, respectively.Each value in the table represents the execution time. The relative difference between GPU and CPU computed results is on the order of 10^{-8}, indicating a high degree of numerical accuracy.

Table 1. Execution Time Comparison

Device	Total Running Time (ms)	Speedup (relative to CPU)
CPU	0.063424	1
GPU(32 , 128)	27.138496	0.002335
GPU(32 , 256)	28.970367	0.002178
GPU(64 , 128)	27.897985	0.002273
GPU(64 , 256)	27.222015	0.002330
GPU(128 , 128)	29.142145	0.002196
GPU(128 , 256)	28.853632	0.002235

As shown in the results, the optimal performance was achieved with CPU , and the optimal performance on GPU is a block size of 32, 128 blocks per grid.

3 Discussion

In this experiment, the CPU outperforms the GPU due to the relatively small matrix size ($N = 6400$). The GPU incurs significant overhead in memory transfer and kernel launch, which diminishes its expected advantage in parallel computation. However, for larger matrices, the GPU is anticipated to provide substantial speedup.

Also , a block size that is too large may reduce parallelization efficiency, while a block size that is too small could create excessive thread overhead, reducing computational performance. These findings indicate that while GPU acceleration may not be effective for small matrix sizes, it holds significant advantages for large-scale computations with sufficient parallelism.

4 Reference

1. Introduction to CUDA Parallel Programming — Homework Assignment 2