Assignment 2 Report - Andrew Chan

All experiments were conducted on an NVIDIA A100 GPU. Square matrices with dimensions m = k = n = 10,000 were used, with elements stored as single precision floating point values. Kernel execution time was measuing using the CUDA events, and performance was reported in GFLOPs using the following formula: GFLOPs = $2 * m * k * n / (\text{execution time in seconds} * 10^9)$

Block sizes of 8x8, 16x16, and 32x32 threads were evaluated for all kernels.

Task 1: Global memory matrix multiplication

In task 1, a naive matrix multiplication kernal was implemented in which each thread computes a single element of the output matrix. Threads directly ccessed global memory for all of the reads of matrices A and B, and no shared memory was being utilized.

Results:

```
Block 8x8 -> Time: 1488.677 ms, GFLOPs: 1343.48
Block 16x16 -> Time: 835.938 ms, GFLOPs: 2392.52
Block 32x32 -> Time: 667.423 ms, GFLOPs: 2996.60
```

The best performance for Task 1 was achieved using a 32x32 block size.

Task 2: Shared memory tiled matrix multiplication

Task 2 improves upon the normal method by using shared memory tiling. Each thread block loads a tile of matrix A and a tile of matrix B into shared memory. These tiles are reused by all threads in the block to compute partial dot products which helped significantly reduce redundant global memory access.

Results:

```
[Task2] Block 8x8 -> Time: 945.764 ms, GFLOPs: 2114.69
[Task2] Block 16x16 -> Time: 560.660 ms, GFLOPs: 3567.22
[Task2] Block 32x32 -> Time: 537.012 ms, GFLOPs: 3724.31
```

The best performance for Task 2 was achieved using a 32x32 block size.

Perormance comparison and discussion

The shared memory tiled implementation in Task 2 outperforms the global memory implementation in Task 1 for all of the block sizes that were evaluated. Depending on the block size, Task 2 achieves between a 1.24x and 1.57x higher throughput. This performance improvement is due to increased arithmetic intensity since each element loaded from the global memry is reused multiple times by threads within a block which reuces global memory traffic. Larger block sizes allow greater reuse which explains why 32x32 blocks give the highest performance in both tasks.

Conclusion

This assignment demonstrated the impact of memory hierarchy on GPU performance. While the naive global memory matrix multiplication give us a good baseline, the use of shared memory tiling significantly improved the performance by reducing redundant memory access. The results also confirm that shared memory optimizations are faster and important for achieving high throughput on modern GPUs.