Technical Report #3

Matrix Multiplication with Tiling and Shared Memory

Alejandro Herce Bernal

October 2018

1. Introduction

Shared memory is a powerful feature for writing well optimized CUDA code. Access to shared memory is much faster than global memory access because it is located on chip. Because shared memory is shared by threads in a thread block, it provides a mechanism for threads to cooperate. Because it is on-chip, shared memory is much faster than local and global memory. In fact, shared memory latency is roughly 100x lower than uncached global memory latency. Shared memory is allocated per thread block, so all threads in the block have access to the same shared memory. Threads can access data in shared memory loaded from global memory by other threads within the same thread block. This capability (combined with thread synchronization) has a number of uses, such as user-managed data caches, high-performance cooperative parallel algorithms (parallel reductions, for example), and to facilitate global memory coalescing in cases where it would otherwise not be possible.

For this project, we're using tilling and shared memory to make the previous matrix multiplication algorithm even faster. The experiment will be done using tiles of sizes 8x8, 16x16 and 32x32. Block sizes will be set according to the tile sizes.

2. Development

A. Technical specifications

- CPU
  - Intel Core i7-4790k
  - Frequency: 4.4 Ghz
  - Cores: 4
  - Logic cores: 8
  - L1 cache: 256 KB
  - L2 cache: 1.0 MB
  - L3 cache: 8.0 MB
- GPU

- Nvidia GeForce GTX 980 Ti
- Memory: 6 GB GDDR5
- CUDA Cores: 2816
- GPU Clock Speed: 1.24 GHz
- Memory Clock Speed: 3505 MHz

B. Testing

For the testing, different configurations were selected to measure the difference in performance. CPU with OpenMP, GPU with CUDA and GPU with tiling and shared memory.

Different configurations were tested for GPUs, varying the tile size and the block size. To avoid memory issues, block sizes are set based on the tile size. All matrices were set to 2000 x 2000

For each of the settings, 10 tests were executed and the execution time was averaged.

A. GPU Testing

For GPU testing, different block sizes were selected with the same formula for the grid.

All times below are in milliseconds. For CPU, OpenMP was used with 6 threads. An average was calculated based on 20 tests of each configuration.

Tile and Block Size	CPU	GPU	Tilled GPU
8x8	11996.53	82.57	31.54
16x16	12354.49	52.32	18.98
32x32	12253.25	63.90	19.63

The most consistent and fast results were with tile and block size of 16x16, but tilled offering not much of a difference between sizes 16 and 32.

B. Comparison and Speedup

For speedup times, we have the following information:

	GPU time	GPU Tilled time	Speedup
8x8	82.57	31.54	2.617
16x16	52.32	18.98	2.756
32x32	63.90	19.63	3.255