Homework 5 Report

1 Test Environment

I tested my code on a Google Compute Instance with NVIDIA Tesla V100 GPU, with CUDA 11.1.

2 Experiments and Results

2.1 CSR Scalar

Each block contains T threads. The grid is divided into m blocks.

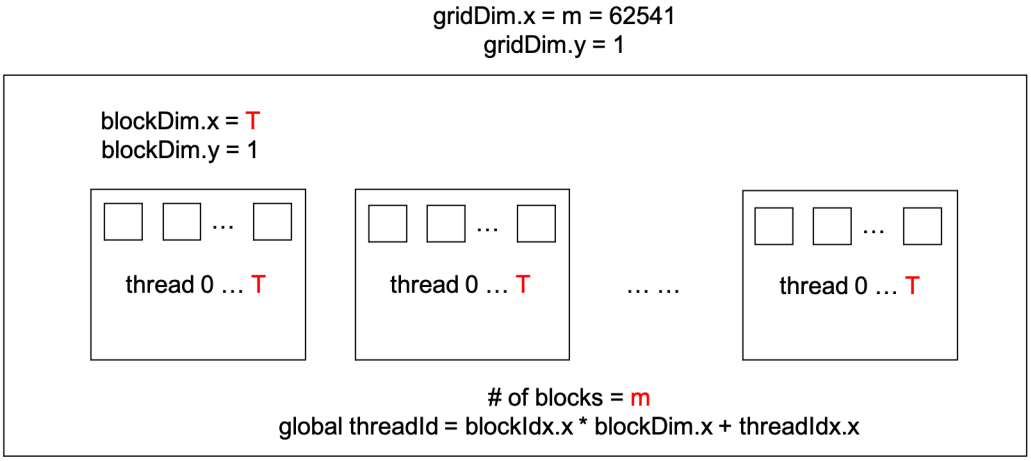

Figure 1. CSR Scalar: Divide Grid into Blocks

Experiment result for m blocks, each block contain 32, ... , 1024 threads.

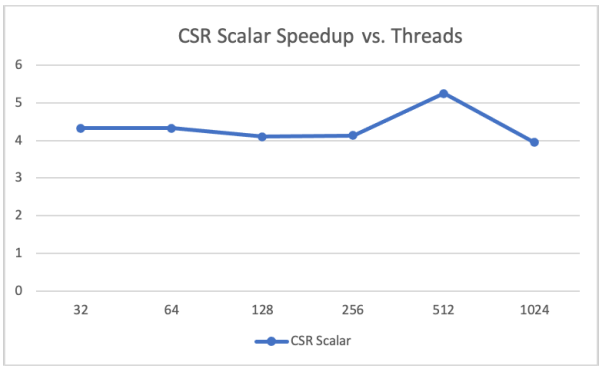

Figure 2. CSR Scalar: Experiment Results

2.2 CSR Vector Reduction with Shared Memory

This uses reduction. Each block contains T threads. The grid is divided into m blocks. An array is declared as share memory to store intermediate results. Syncthreads() is called when calculate local sums and reduce local sums to global sum.

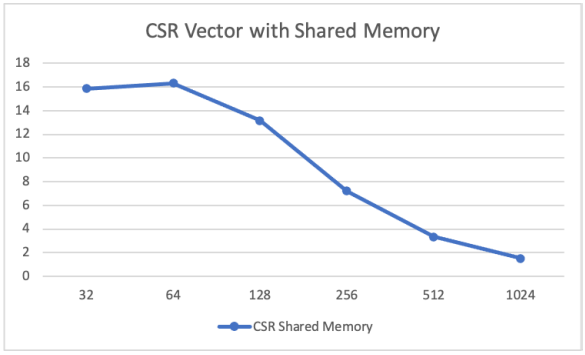

Figure 3. CSR Vector with Shared Memory: Result

2.3 ELL without Shared Memory

I implemented ELL similar to CSR scalar. Each block contains T threads. The grid is divided into m blocks.

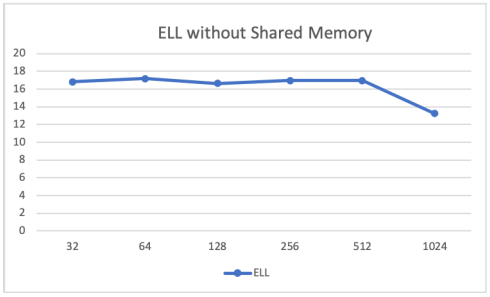

Figure 4. Ell without Shared Memory: Result

2.4 ELL with Shared Memory

This is similar to CSR reduction with shared memory.

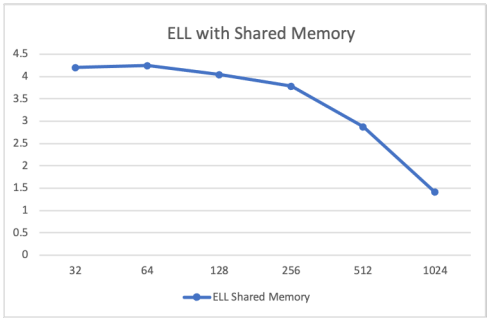

Figure 5. Ell with Shared Memory: Result

2.5 Performance Comparison

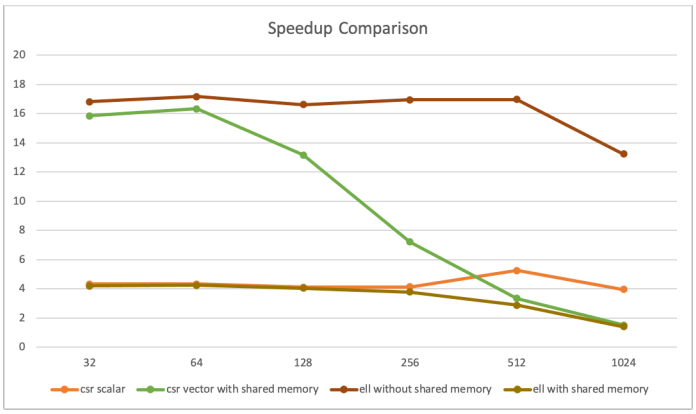

Figure 6. Performance Comparison

3 Findings

- The best performance is achieved by ELL without shared memory, which gains 17.16x speedup than CPU SPMV.
- CSR scalar implementation has some problems that led to poor performance. Since adjacent threads cannot access data in a contiguously way, the hardware cannot combine memory access for concurrent threads.
- CSR vector implementation addresses the problems that CSR scalar has. CSR vector accesses indices and data contiguously, and overcomes the problem of load balancing and memory access pattern.
- ELL addresses non-coalesced memory access of CSR by using padding and transposition on the sparse matrix. By allowing coalesced memory access, ELL outperforms CSR.
- ELL shared memory performs worse than without shared memory. This is most likely due to the fact that ELL already has a coalesced memory access pattern, by doing reduction with shared memory will cause more overhead.