Drexel University

Electrical and Computer Engineering Dept.

ECEC-413

Assignment 8:

Jacobi Iteration

Minjae Park

John Truong

Professor Naga Kandasamy

Date: 05/28/2020

Jacobi Iteration implemented using CUDA:

CUDA is a parallel computing platform and API created by Nvidia to allow parallel computing at the GPU level. Exhaustive numbers of rather trivial operations can be mass-parallelized with the hundreds of cores on the GPU, providing more parallel efficiency than some CPUs.
The setup for CUDA API can be found in *compute_on_device* in jacobi_iteration.cu. One major difference between the CUDA kernel and compute gold, is that the kernel has an additional function that is used to update the x solution value. This is due to the fact that grid level threads syncing doesn't exist.

For the naive kernel, the thread block and grid is base off of the #define constant (THREAD_BLOCK_SIZE and number of rows in matrix A). For optimized kernel that uses tiling, it is configured based off of the #define constant (TILE_SIZE and number of rows in matrix A). In the kernel code we have to be checking for convergence (which is done on the host side as the SSD value is transmitted back to the host from the device).

In the naive approach of solving the jacobi iteration, each thread is responsible for a row in the matrix. To compute the SSD value, row reduction was used and then by using mutex locks after adding the SSD pointer. For the optimized approach for the jacobi iteration, more shared memory is being used. For this we implemented the tiles, once the tiles are populated; half of the threads in the black do the summation calculations. This step is repeated as the tiles go across their respected rows. Once it is finished, the new x values are found and the computation for SSD happens just like how it occurs in the naive approach.

Results: Speed Up:

Evaluating execution time on Drexel Xunil-05 server
xunil-05 has a Nvidia 1080 GTX GPU

Matrix Size	CPU	GPU naïve (Thread Block Size = 128)	GPU optimized (Tile Size = 16)	GPU naïve (Thread Block Size = 256)	GPU optimized (Tile Size = 8)
512 x 512	3.628	0.602	1.214	0.840	0.976
1024 x 1024	28.566	2.238	7.779	3.255	4.895
2048 x 2048	240.306	12.596	52.022	13.934	28.076

Table 1: Jacobi Iteration CUDA Data on Xunil-05

Matrix Size	CPU	GPU naïve (Thread Block Size = 128)	GPU optimized (Tile Size = 16)	GPU naïve (Thread Block Size = 256)	GPU optimized (Tile Size = 8)
512 x 512	3.628	602.66%	298.85%	431.90%	371.72%
1024 x 1024	28.566	1275.96%	367.09%	877.30%	583.37%
2048 x 2048	240.306	1907.80%	461.93%	1724.60%	855.91%

Table 2: Jacobi Iteration CUDA Speed up

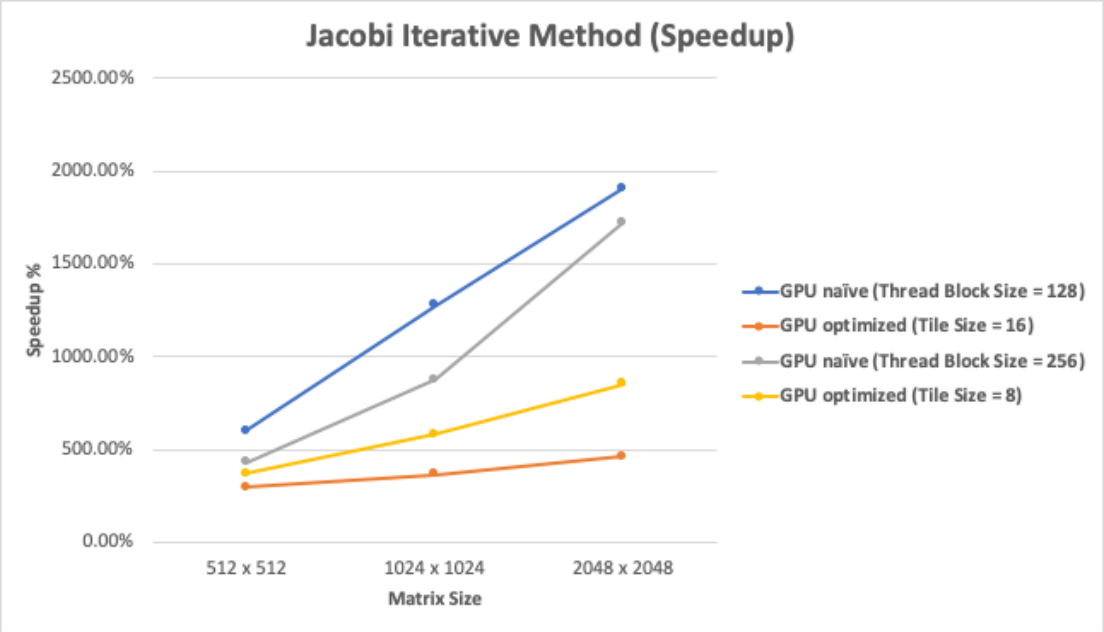

Figure1: Speedup per Thread Block Size and Tile Size