LAB 4 = Comparison of CUDA and serial versions of Matrix multiplication and Sobel

Submitted by - Shivani Sabhlok
500237896

Part 1

Summary of multiplication of 1024X1024 matrix populated with floating numbers between 1.0 to 2.0 –
CPU took more than 5 sec with around .4 GFLOPS per second
On the other hand, GPU took .02 sec with around 83 GFLOPS per sec

Device/Host	Time(sec)	GFLOPS per sec
CPU	5.1	.42
GPU	.02 (best result)	83.48

GPU results with different thread and block configurations with one grid –

Number of Blocks	Threads per Block	Time(sec)	GFLOPS per sec
2	1024	.22	9.5
4	512	.10	20.10
8	1024	.05	41.22
16	512	.02	81.58
16	1024	.02	83.48

Best combination on GPU is 16 blocks of 1024 with one grid.
This is a heavily parallelized version. We are processing 16 rows parallelly. 1024 threads was a good choice because each row and column has 1024 elements.
Using 16 blocks also proved to be a good choice because each block was assigned a SM and thus the speedup is tremendous.

```
-bash-4.2$ ./lab4p1_serial


********************************************************************************
Time taken on CPU (sec) = 5.158850
GFLOPS = 0.41587


********************************************************************************
-bash-4.2$ ./lab4p1_cuda


********************************************************************************
Time taken on CPU (sec) = 5.156477
GFLOPS = 0.416057

With 2 blocks and 1024 threads per block, Time taken on GPU (sec) = 0.223904
GFLOPS per sec = 9.581729
With 4 blocks and 512 threads per block, Time taken on GPU (sec) = 0.106814
GFLOPS per sec = 20.085228
With 8 blocks and 1024 threads per block, Time taken on GPU (sec) = 0.052092
GFLOPS per sec = 41.184508
With 16 blocks and 512 threads per block, Time taken on GPU (sec) = 0.026163
GFLOPS per sec = 82.001213
With 16 blocks and 1024 threads per block, Time taken on GPU (sec) = 0.025733
GFLOPS per sec = 83.371041
Results matched


********************************************************************************
```

Part 2

Summary of implementation of Sobel operator for edge detection performed on CPU as well as GPU. For the sample image coins.bmp of Height = 246 and Width = 300, the operation converged with a threshold of 50.

Timing comparison shows, GPU was 10 times faster than CPU.

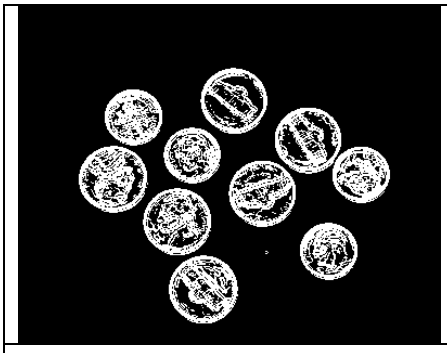
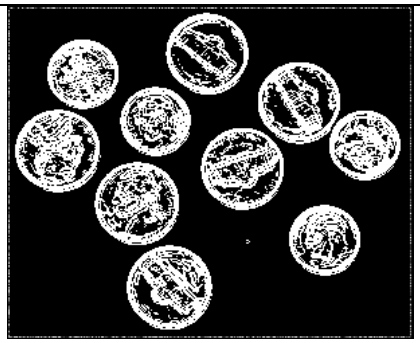
Device/Host	Time(sec)
CPU	.139
GPU	.014 (best result)

GPU results with different thread and block configurations with one grid –

Number of Blocks	Threads per Block	Time(sec)
64	64	.0149
128	128	.0145
64	32	.0156

The different combinations tried didn't differ significantly in their performance but the best combination for the given sample image is one grid with 128 blocks, each with 128 threads. The performance difference is not evident probably because of a small image and the dimensions of the image.

```
-bash-4.2$ ./lab4p2_serial coins.bmp s.bmp
Image Info ::
        Height=246 Width=300


********************************************************************************
Elapsed time for Sobel Operation on CPU (sec): 0.132522

Theshold: 50


********************************************************************************
-bash-4.2$ ./lab4p2_cuda coins.bmp c.bmp
Image Info ::
        Height=246 Width=300


********************************************************************************
With 64 blocks and 64 threads per block, Time for converging with 50 threshold on kernel: 0.015030 s
With 128 blocks and 128 threads per block, Time for converging with 50 threshold on kernel: 0.014499 s
With 64 blocks and 32 threads per block, Time for converging with 50 threshold on kernel: 0.015601 s


********************************************************************************
-bash-4.2$
```

	
SERIAL SOBEL OUTPUT	**CUDA SOBEL OUTPUT**