Lab4 c5441aa

Omid Asudeh
OSUID: 500149733

The final soft version.

Instructions:
Load cuda using: module load cuda
Compile using: make
Run part 1 using: ./lab4p1
Run part 2 using: ./lab4p2 “input_.bmp” “serial.bmp” “cuda.bmp”

Part 1

a)

	time	Gflops	#operations
serial	1041.560059	0.3958647001	412316860416
parallel	11.29	36.5205368	412316860416

b)
The serial version runs for around 1041 seconds (or 17 minutes) for 4096 dimensionality while the cuda does it almost 100 times faster in only 11 seconds.
Important note: I set the dimensionality of the serial part to 1024 for the grader to sanity check. But the result above are all tested for the actual dimensionality of 4096.
c)

- The grid has 4 blocks. Blocks are 1 dimensional. Each block has 1024 threads.
- How the number of operations needed calculated?

```
for(i=1 to 4096)
    for(j=1 to 4096)
        for(k=1 to 4096)
            sum+=(a[i*dim+k]*a[k*dim+j]);// 6 operation
```

Number_of_operations = 4096^3*6 = 412316860416
Gflops are calculated by dividing the number of operations needed by the time each version does the job.

Part 2

Timing and threshold summary:

image_name	serial	cuda_good	threshold	Improvement
coins.bmp	0.22	0.04	50	5.5
image01.bmp	3.54	0.54	43	6.555555556
image02.bmp	5.05	0.7	33	7.214285714
image03.bmp	12.94	1.82	40	7.10989011
image04.bmp	25.87	3.68	157	7.029891304
image05.bmp	4.61	0.69	55	6.68115942
image06.bmp	1.11	0.25	17	4.44
image07.bmp	2.69	0.37	37	7.27027027
image08.bmp	8.22	1.14	106	7.210526316
image09.bmp	3.52	0.56	41	6.285714286
average:	5.3398	0.979		6.529729298

CUDA Organization:

My program divides the input image to 32*32 pixel blocks. In other words, the 2D grid has (height/32)*(width/32) blocks. The blocks are also 2D. Each block has 32*32 = 1024 threads. The load balancing is one pixel per thread.

Improvement over serial version:

From the results in the previous table, the power of GPU is obvious. The performance is 6-7 times faster on average while it is more than 7 times faster for 50% of the inputs.

Challenges:

Actually, I first used another a thread hierarchy (one thread per line) which was a little bit better than serial and in some cases even worse. Therefore, I changed my hierarchy to maximize the usage of threads (one thread per pixel).