Roy Liu

Nov. 19, 2019

Problem 1

In this problem, we compile and run "Vector Add" application on both CPU (single threaded) and GPU.

1.a Write and compile an example of "Vector Add" similar to lecture 02 slides both for CPU and GPU. Use argument N as the length of the vectors. Compare to the performance of CPU and GPU over the increasing value of N (1K, 10K, 100K, 1M, and 10M). Measure the execution time for CPU and GPU and show that how the CPU and GPU performance and relative GPU over CPU speedup scale with increasing job size (N as the length of array). When measuring the performance only focus on the kernel execution time and ignore the time for transferring the data between CPU and GPU. Make sure to elaborate your results (20pts).

A: GPU starts consuming time more, but when the data number increases, GPU run faster.

Iterations	Time(ms)	
	CPU	GPU_w/o data transfer
1K	0.06	8.79
10K	0.34	8.95
100K	2.67	10.84
1M	25.23	29.18
10M	246.23	205.96

1.b For job size of 1M (N=1M), vary the thread block size (TB= 128, 256, 512, 1024). Plot and demonstrate how execution time of the kernel (offset the execution time for data transfer) varies over different thread block sizes. Plot your results. Make sure to elaborate your results. (20pts)

A: with block size increase, the GPU runs slightly faster.

	Time (ms)
Block size	w/o data transfer
128	29.208
256	28.87
512	28.75
1024	28.26

1.c. plot the results for both 1.a and 1.b, this time consider the additional overhead of moving data between the host and device. Argue how the benefits of GPU accelerator changes when the overhead of data movement come to the picture. Make sure to elaborate your results. (10pts)

A: Generally GPU with data transfer consumes more running time.

Iterations	Time(ms)		
	CPU	GPU_w/o data transfer	GPU_with data transfer
1K	0.06	8.79	8.876
10K	0.34	8.95	9.077
100K	2.67	10.84	10.736
1M	25.23	29.18	29.205
10M	246.23	205.96	211.352

	Time (ms)	
Block size	w/o data transfer	with data transfer
128	29.208	29.398
256	28.87	29.389
512	28.75	29.351
1024	28.26	28.702

Problem 2:

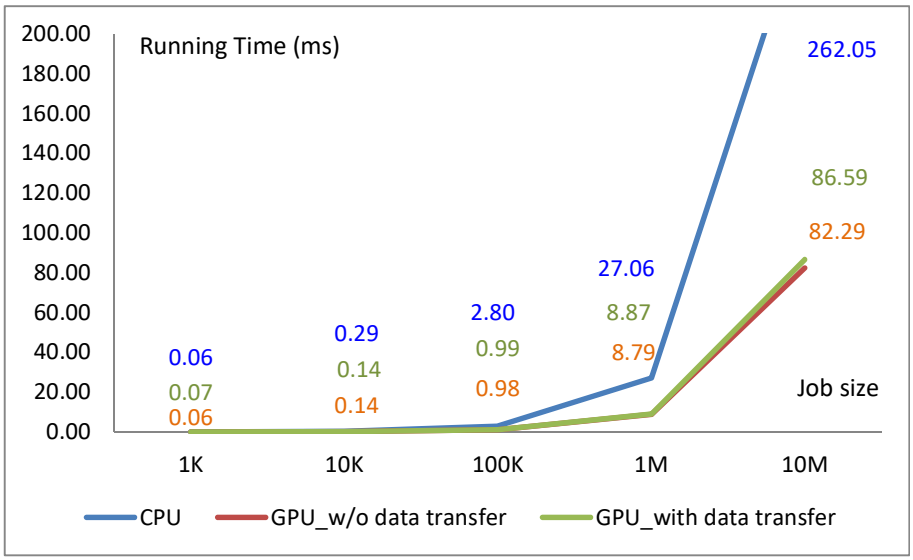
In this problem, we compile and run "1D Stencil" application on both CPU (single threaded) and GPU.

2.a Write and compile an example of "1D Stencil" similar to problem 1. Consider N as the job size (the length of array), and R as the radius of Stencil. Measure the speedup over CPU execution (with and without considering data movement between CPU and GPU) for increasing job size of N (1K, 10K, 100K, 1M, 10M), with the radius of 2 (R=2) and TB size=128. Plot your results including relative speed up with and without data movement. Make sure to elaborate your results (10pts).

Answer:

In this experiment, I composed "1D Stencil" application with three programs called *pb1_cpu.c, pb1a_gpu.cu* and *pb1b_gpu.cu*, which deliver the execution under CPU, GPU without data transfer between host and device and with the data transfer. The below figure indicates GPU runs faster than CPU, the bigger job size the faster. In job size of 10M, GPU's speed up reached to 220% (82.29ms vs 262.05 ms).The result depicts GPU is stronger to deal with the 1d stencil algorithm contributed by the parallel streaming processors.

Relatively, data transfer between host and device costs overhead in GPU scenario.



2.b Compared the performance of CPU and GPU over the increasing value of N=10K, but R=2, 4, 8, 16. Focus on the kernels execution time, and plot the relative performance with TB size =128. Make sure to elaborate your results (10pts)

Answer:

When radius increasing, CPUs running time slightly increased, while GPU relatively more flat.

Iterations	Time(ms)	
Radius	CPU	GPU_w/o share memory
2	0.30	0.138
4	0.40	0.137
6	0.55	0.138
8	0.65	0.156

3.c Explore the opportunities for using the shared memory to enhance the performance benefits on GPUs. Demonstrate results that plot benefits/limitation of shared memory with respect to increasing job size (N), and radius size (R). consider TB size of 128 for all experiments. Make sure to elaborate your results and logically analyze your findings (30pts).

A: When use share memory the running time improved.

Radius=4, TB=128

Numbers	Time(ms)	
	GPU_w/o share memory	GPU_share memory

1K	0.06	0.14
10K	0.16	0.15
100K	0.92	0.96
1M	8.67	8.57
10M	84.33	81.93

Running time n=10k, TB=128

Iterations	Time(ms)	
Radius	GPU_w/o share memory	GPU_share memory
2	0.137	0.138
4	0.138	0.138
6	0.138	0.138
8	0.138	0.141