Chuong Dinh Vu
Noah del Angel

Collatz

Question 1a) What compute times do you get (in seconds)? List them in a table where the first column lists the inputs in the order from the submission script and the second column lists the runtimes. Use two digits after the decimal point for the compute times.

Input	Time
200000000	0.07
200000000	0.07
2000000000	0.69
2000000000	0.67
10000000000	3.54
10000000000	3.55

Question 1b) Compared to the serial CPU runtime of 37.86 seconds for the 200 million sequence input, how many times faster is the GPU code?

The GPU code is 540.8 times faster than the CPU code.

Question 1c) For the largest measured input, how many thread blocks are launched?

For the largest input the number of thread blocks launched is 97,656,625.

Question 1d) For the largest measured input, what is the maximum sequence length?

The maximum length is 1133.

Question 1e) Based on this sequence length and the program input, explain why most of the calls to atomicMax are guaranteed to not change the value of maxlen.

Most of the lengths will not be greater than the old value in memory. Hardware support for the atomic operation prevents race conditions from producing incorrect results.

Fractal

Question 2a) What compute times do you get (in seconds)? List them in a table where the first column shows the input (in the same order as in the submission script) and the remaining columns show the compute time for the two code versions (first double and then float). Use three digits after the decimal point for the compute times.

Input	Time
Frame:32 Width:1024 (Double)	0.267
Frames:64 Width:1024 (Double)	0.476
Frames:32 Width:4096 (Double)	3.442
Frames:64 Width:4096 (Double)	6.882
Frame:32 Width:1024 (Float)	0.168
Frames:64 Width:1024 (Float)	0.337
Frames:32 Width:4096 (Float)	2.423
Frames:64 Width:4096 (Float)	4.844

Question 2b) How many times faster is the float version than the double version and why is it so much faster?

The float version is 1.42 faster times than the double version. This is because double has 2x more bits than float.

Question 2c) Why does the pic array on the GPU not have to be initialized before the kernel call?

We allocate all of the needed bits for the array, at the array head pointer location. From there onwards we can access elements of the array like normal.

Question 2d) How many pixels per second does the float code compute on the largest input? Based on this rate, would the GPU be fast enough to compute the fractal in real-time at 60 frames per second for a UHD monitor with 8.3 million pixels?

It is not possible to compute the frames per second in real-time. This is because the pixels/second for the float code is 221,638,347. Whereas, the needed pixels per second is 498,000,000. As a result, we can't do this in real-time.

Question 2e) When you run "./fractal_cuda 1024 32" on a Frontera login node (after running "module load cuda"), it does not work. Explain why not (explain the fundamental problem, do not just rephrase the error message in your own words).

You cannot run code on the login nodes in Fronterra, if this was possible Frontera would be extremely slow. The only way to run code on Frontera is to submit it to a job, then Frontera will run it on the appropriate compute node.

MIS

Question 3a) What compute times do you get (in seconds)? Present them and the input names in a table using the same order as in the submission script (from top to bottom). Use four digits after the decimal point.

Graph	Time
USA-road-d.USA.egr	0.0066
Europe_osm.egr	0.0148
R4-2e23.sym.egr	0.0183
Soc-LiveJournal.egr	0.0075
Uk-2002.egr	0.0232

Question 3b) Which one of the CPU arrays is no longer needed and why?

We no longer need the CPU priority array, because all of the blocks have access to the computed priority array and the CPU has no need to compute anything with the priority array.

Question 3c) Why do we have to break the serial mis function into two kernels?

If the init was in the mis function, it would be initialized with each step in the do-while. To prevent this performance drag, we remove init from the mis function.

Question 3d) Why do we not have to call cudaDeviceSynchronize() before stopping the timer?

cudaMemcpy() block the CPU completely until it is done, therefore there is no need for cudaDeviceSynchronize().

Question 3e) Make a copy of the submission script and replace the commands between the two date commands with nvprof ./mis_cuda /home1/00976/burtsche/Graphs/soc-LiveJournal1.egr, which runs the code through the profiler. Based on the output, how much time is spent in the init kernel and how much combined time in all mis kernel runs? List both times in milliseconds with two digits after the decimal point. How often is the mis kernel launched? Which of the listed GPU activities takes the most time?

Location	Time spent (millis)
init	0.06
Combined time	1.28

The mis kernel is launch 6 times.
The cudaMemcpy() takes the most with 48.659 ms