University of Chicago

Winter 2023

GPU Ray Tracing with CUDA

Assignment 2 : High Performance Computing

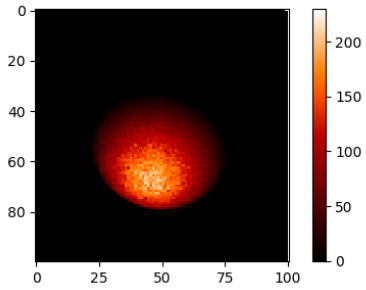
Divya Pattisapu

02-06-2023

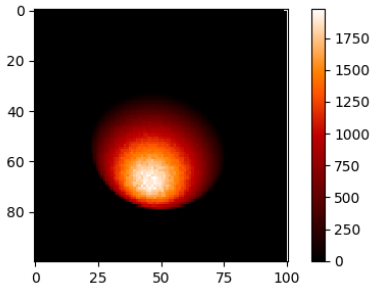
1 Question 1:

1.1 Serial implementation of the problem Statement

The serial implementation of the ray tracing algorithm takes a minute for 10 million rays on a 1000x1000 grid, but any larger value of Nrays runs for a very long time. It is infeasible to render larger grids serially. The following are two plots generated using serial implementation.



(a) 100x100 grid, Nrays = 10e5 takes 0.3 seconds



(b) 100x100 grid, Nrays = 10e7 takes 12.75 seconds

2 Question 2:

2.1 Use CUDA to parallelize your code to run on GPUs

2.1.1 Random Number Generation

- We generate random vectors for light ray generation using LCG pseudo random number generator
- We use seed = constant*thread_id to generate another seed for each thread which will be more spread out, leading to better random numbers using fast forwarding function. The input n to this = number of random numbers required to be generated. That is in our case, 2 for theta and phi times 1000 for the while loop times id. (n = 2000*Nrays*id)
- We pass this along with our seed to generate a new seed per thread using fast forwarding
- We then use LCG to n number of generate random doubles

2.1.2 Using Atomic Add

We use atomic add to increase the illumination in a cell. Since multiple rays generated from different threads can simulataneously access the grid, we require to add an atomic add function to avoid race conditions.

3 Question 3:

3.1 Optimal block and thread configuration

The best configuration for Nrays = 10 billion is with threads per block as 256.

Threads per block	Number of blocks	Time taken
16	88129089	2.39
32	44064545	1.37
64	22032273	1.38
128	11016137	1.37
256	5508069	1.37
512	2754035	1.49
1024	1377018	1.66

Table 1: Optimal configuration of threads per block

n	Nrays (Problem size)	Serial time	Parallel Time (NTPB = 256)	Speedup
100	100000	0.13	0.0028	46
100	1000000	1.28	0.0033	384
100	10000000	12.75	0.0130	983
1000	100000	0.59	0.0395	15
1000	1000000	5.89	0.0399	147
1000	10000000	58.48	0.0453	1292

Table 2: Parallelization speedup for different problem sizes (Nrays) and different grid sizes

4 Question 4:

4.1 Compare the runtime of the serial CPU version and CUDA GPU versions as functions of problem size

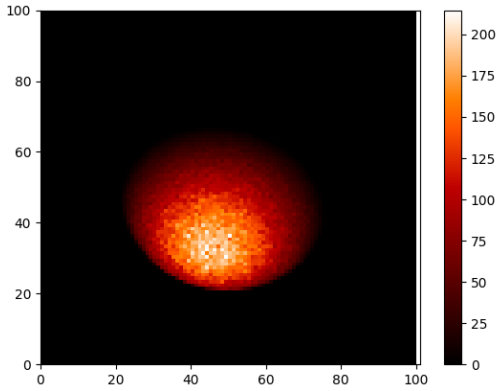
Using threads per block equal to 256, the following is the speedup for the GPU in comparison with the serial code.

- As the number of threads per block increases, the number of blocks decreases. (nblocks = number of threads divided by the number of threads per block)
- The optimal configuration is with 256 threads per block. For this configuration, the number of blocks is 5508069, and the time taken is 1.36919 seconds.
- For larger NTPB (>=2048), the program doesn't execute. Probably due to the upper bound on number of threads per block on Midway.
- For a fixed value of Nrays, as the problem size increases, both the serial and parallel times also increase. However, the increase in parallel time is much smaller compared to that of the serial time. This is evident from the speedup column, where we can see that as the problem size increases, the speedup achieved by the parallel implementation also increases.
- For example, for Nrays = 100, the serial time for a problem size of 100000 is 0.13 seconds, while the parallel time (NTPB = 256) is only 0.0028 seconds. The speedup achieved is 46. For a problem size of 10000000, the serial time is 12.75 seconds, while the parallel time is only 0.0130 seconds. The speedup achieved is a remarkable 983.
- Similarly, for Nrays = 1000, the serial time for a problem size of 100000 is 0.59 seconds, while the parallel time is 0.0395 seconds. The speedup achieved is 15. For a problem size of 10000000, the serial time is 58.48 seconds, while the parallel time is 0.0453 seconds. The speedup achieved is 1292.

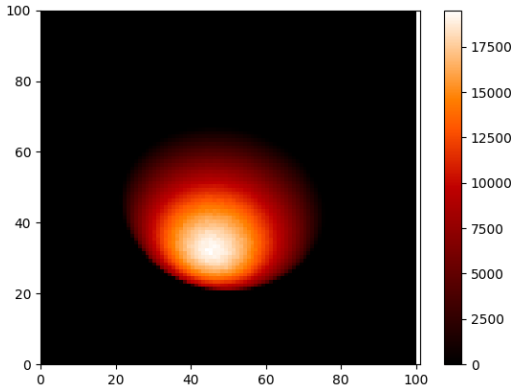
5 Comparing grid sizes and problem sizes

5.1 Question 5: Figure 1(d) shows 1000x1000 grid image for 10 billion rays

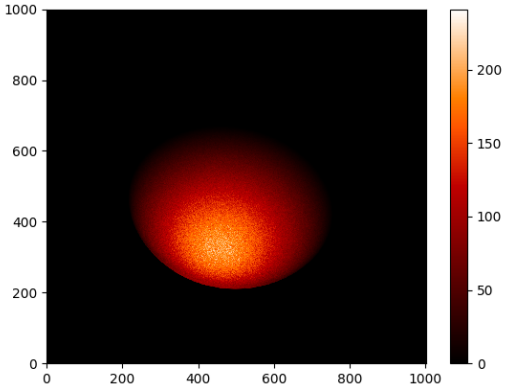
- For smaller grid size, large Nrays values leads to very large illumination ranges(sum of b values in the cell) as seen in the color ranges
- For 1 million rays, the 100x100 grid has a few unlit areas where it is almost completely illuminated in 100 million rays image in Figure 1(b). Any further Nray increase leads to almost no change in the plot (using variable pcolor ranges) except the illumination values increasing.
- If we compare images b and c, we see that when the larger grid size has very small illumination ranges in the plot, and the image is spotty. As grid size increases, we require more rays to illuminate the image
- Image d shows a plot for 10 billion rays on a 1000x1000 grid. This is still not as well lit as Image b due to the larger grid size.



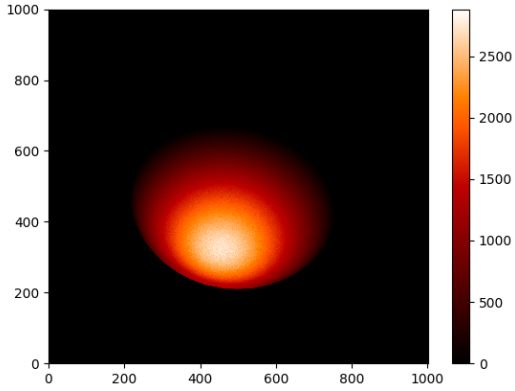
(a) 100x100 grid, Nrays = 10^6



(b) 100x100 grid, Nrays = 10^8



(c) 1000x1000 grid, Nrays = 10^8



(d) 1000x1000 grid, Nrays = 10^{10}

Figure 2: Comparing grid sizes and problem sizes