PP Checkpoint Report

Parallel Ray Tracing

Jim Liu
Owen Wang

Make sure your project schedule on your main project page is up to date with work completed so far, and well as with a revised plan of work for the coming weeks. As by this time you should have a good understanding of what is required to complete your project, I want to see a very detailed schedule for the coming weeks. I suggest breaking time down into half-week increments. Each increment should have at least one task, and for each task put a person's name on it.

For the rest of the semester, we will focus on making the ray tracer real-time. The focus will be on continuing to increase the performance to real-time standards by optimizing the CUDA code, improve scaling on multiple GPUs and add in denoise methods.
The following is our new schedule: a): task for Jim, b): task for Owen

Week 14	a) Implement persistent threads method which is a global job dispatcher. b) Setup and run on a real GPU cluster.
	a) Analyze the performance of the global job dispatcher. b) Analyze and compare performance of MPI implementations.
Week 15	a) Using opengl to render the cuda texture. Implement a denoise method to improve the real time rendering quality under small samples b)Merge the persistent threads method with MPI, Implement a (different) denoise method to improve the real time rendering quality under small samples
	a) Implement a denoise method to improve the real time rendering quality under small samples b)Implement a (different) denoise method to improve the real time rendering quality under small samples
Week 16	a/b) Finalize and write report
	a/b) Prepare for the poster session.

One to two paragraphs, summarize the work that you have completed so far.

We successfully ported the CPU version ray tracer into CUDA to run on a GPU. For our starter code, we used an online ray tracing project: Ray Tracing In One Weekend. We added changes to the data structures for CUDA device code and also implemented the BVH tree traversal on GPU. We optimized the BVH tree traversal using a per-ray trace method (https://www.nvidia.com/docs/IO/76976/HPG2009-Trace-Efficiency.pdf).

Furthermore, we augmented the CUDA implementation with two MPI approaches. The first approach we used was to distribute an instance of the problem to every single processor. This means that each processor will render the entire image. The final result is then collected and averaged. This effectively increases the number of samples per pixel but maintains the same render time as one processor. Another approach was to split the work of rendering one image across different processors. We currently use a static interleaved schedule where processors work on rows of blocks. This method will reduce the render time of a single scene and may be used to increase performance for the purposes of real time rendering.

Describe how you are doing with respect to the goals and deliverables stated in your proposal. Do you still believe you will be able to produce all your deliverables? If not, why? What about the ”nice to haves”? In your checkpoint writeup we want a new list of goals that you plan to hit for the poster session.

We are on track with the first part of our original plan which was to make the ray tracing run on a GPU and optimize it. However, we are planning to pivot from our plan after week 13.
We are no longer planning on implementing the gradient domain method. The reason is that this method has a heavy focus on computer graphics (like valuing how to choose a path to shoot another ray) instead of parallel computing. The time to implement the single threaded version of this method will be significant, and there is not much to parallelize except for poisson solver (a very small portion of this method).

Therefore, we have decided that instead of using the gradient domain method, we wanted to make something more relevant to material covered in the course. We will parallelize the ray tracer by using multiple GPUs via MPI. We also want to include some denoising methods. These denoising methods will allow us to render scenes with lower samples per pixel to improve performance. This will make it possible to render the image in run time on OpenGL.

What do you plan to show at the poster session? Will it be a demo? Will it be a graph?

If we succeed in implementing the real time ray tracing, we will have a playable demo done in OpenGL and/or render a video.

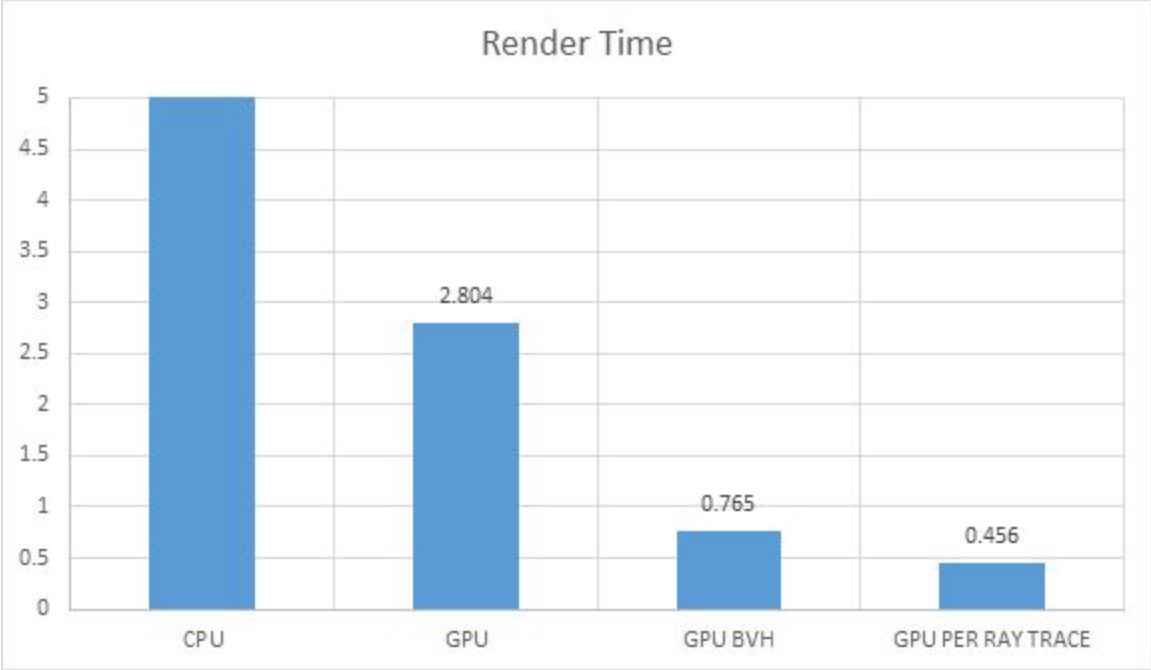
We will also document the methods we are using and have a graph showing the performance difference between each optimization technique that we explored.

Do you have preliminary results at this time? If so, it would be great to include them in your checkpoint write-up.

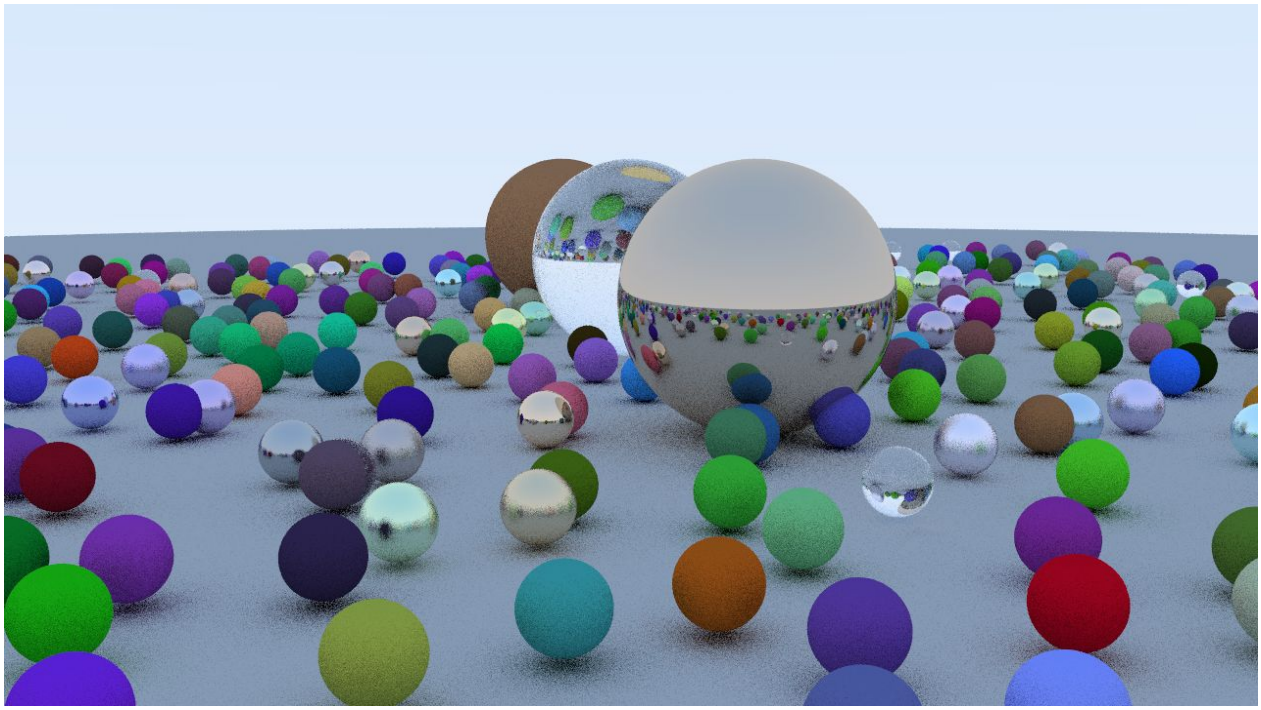
Initial performance record (GTX 1080):

Testing done with 1280 * 720 resolution and 10 samples per pixel

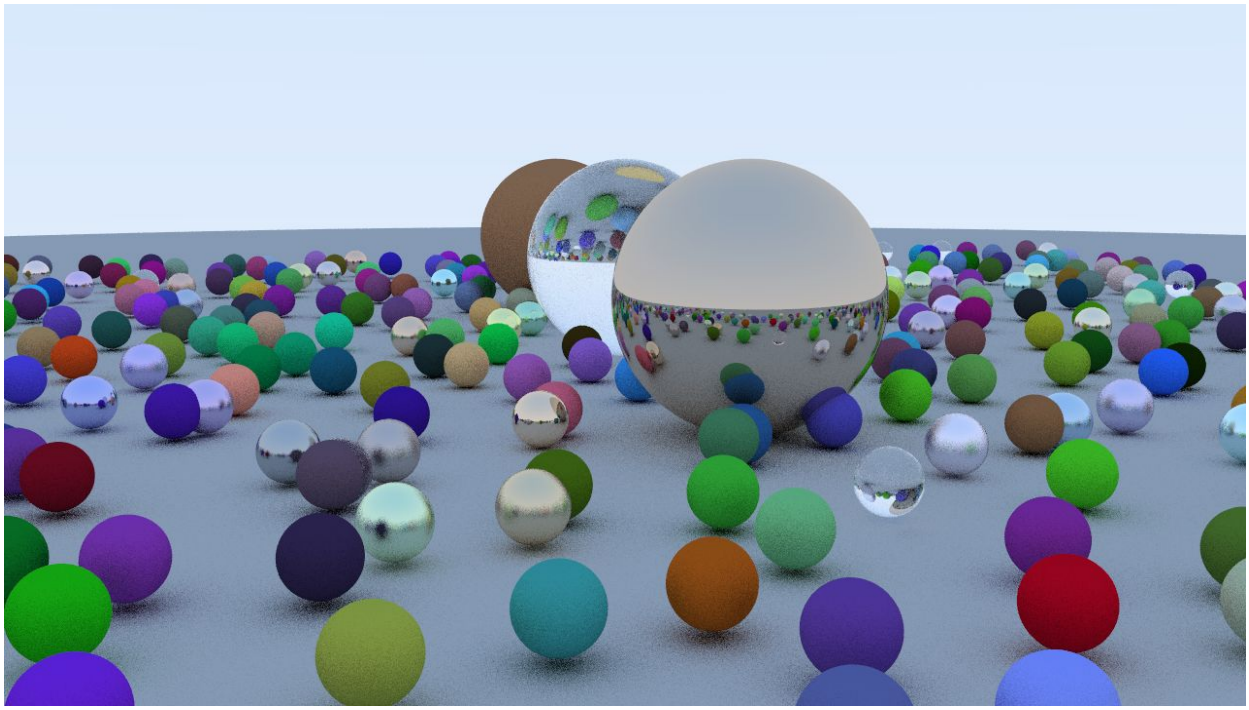
	CPU	GPU	GPU/BVH	GPU/BVH(while while)
Time	101.4s	2.804	0.765s	0.456s



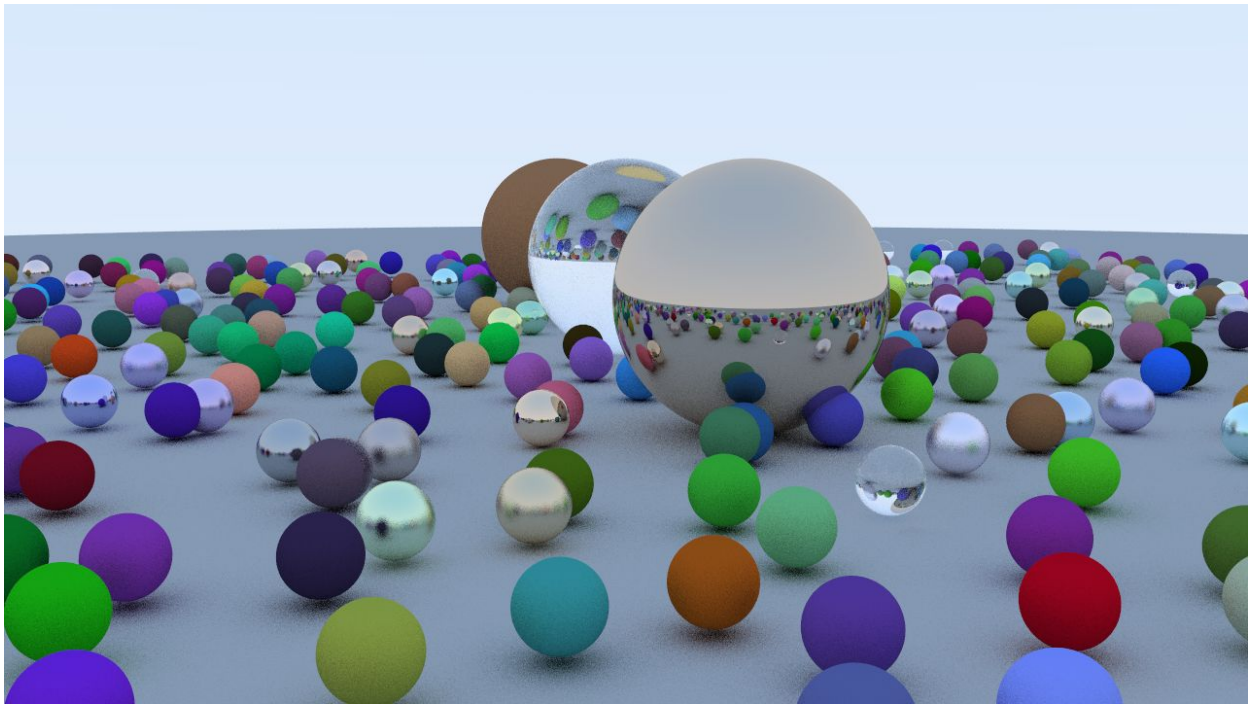
We have also generated images to demonstrate the first MPI approach of averaging the results across different processors. For comparison, we have included an image on 1 processor with 80 samples/pixel.



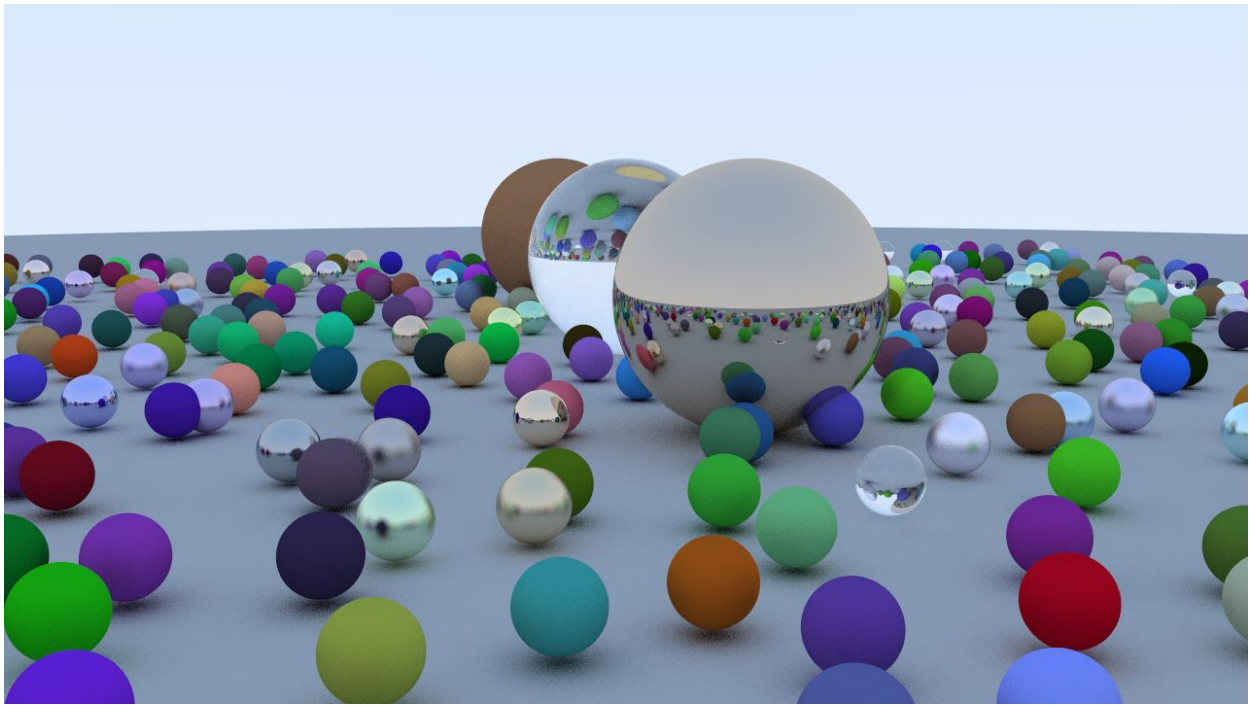
1280 x 720: 10 samples/pixel, 1 processor



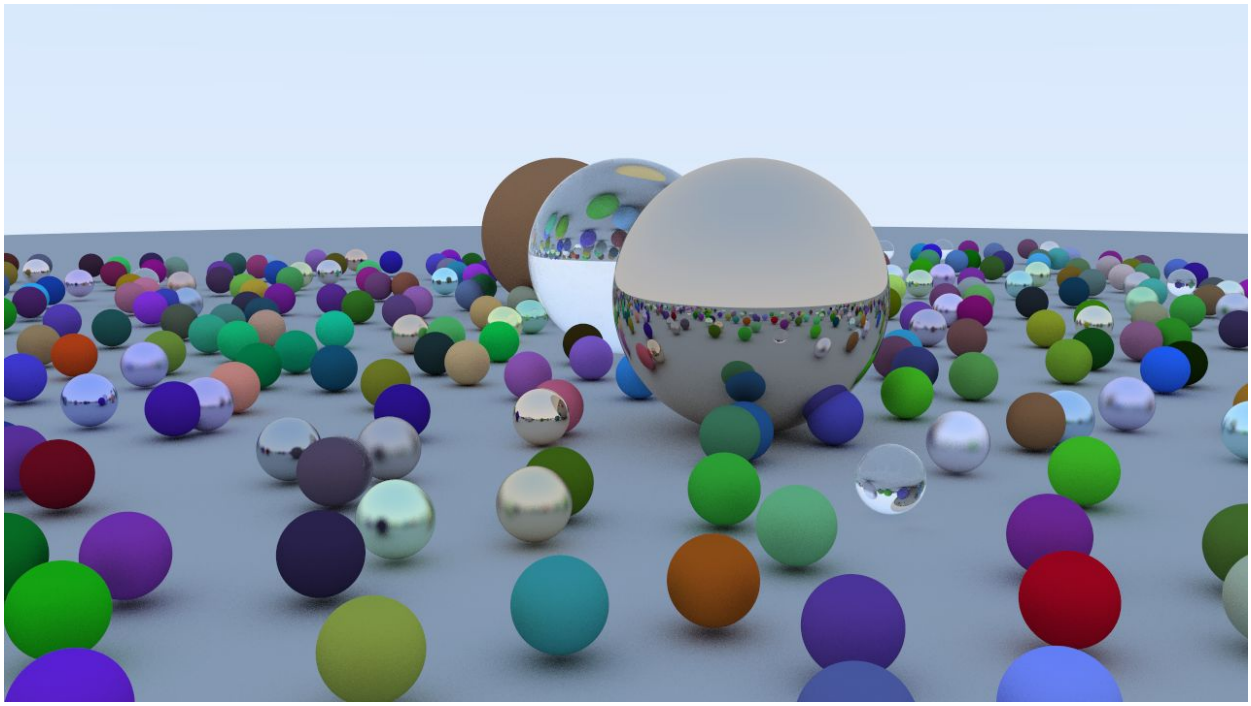
1280 x 720: 10 samples/pixel, 2 processors



1280 x 720: 10 samples/pixel, 4 processors



1280 x 720: 10 samples/pixel, 8 processors



1280 x 720: 80 samples/pixel, 1 processors

List the issues that concern you the most. Are there any remaining unknowns (things you simply don't know how to solve, or resources you don't know how to get) or is it just a matter of coding and doing the work? If you do not wish to put this information on a public web site you are welcome to email the staff directly.

- The CUDA code in release mode in visual studio is problematic. One problem which occurred was that the code successfully compiled in debug mode but not in release mode.
- Our original plan was to use the Bridges supercomputing cluster to test our MPI implementation. However, we have encountered a bottleneck as we haven't been able to acquire an allocation for the GPU resources despite hours of waiting. If we are not able to use Bridges, we will look into either using the Latedays cluster or setting up our own on AWS.