Parallel Functional Programming Project Proposal: Parallel Ray Tracer

Garrison Grogan, UNI gg2652

November 2019

I intend to write a very simple parallel ray tracer in Haskell. A ray tracer is a computer graphics program which, given a description of a scene, produces a photo realistic image through a ray tracing algorithm. The ray tracing algorithm in short shoots many rays from light sources in the scene, and calculates intersections between those rays and objects in the scene. Depending on the light source types implemented and material types implemented, this can produce a wide range of effects like reflections, refractions, global illumination, diffusion, etc.

Millions of rays are used to produce a single image, and rays are independent of each other so the application is a great use case for parallelization. Besides load balancing problems with the rays, there are also data structures which can be used to accelerate the tracing process. Most of these involve putting scene objects in some type of tree. These acceleration data structures should be implementable fairly easily in a functional language like Haskell.

Due to time limitations, I will focus on implementing only a simple raytracing model with only one or two features like reflection and diffusion. I will also focus on only having primitives like spheres, boxes, and planes working before attempting to add general 3D models. I will implement the program on a single thread, then move on to parallelizing it, hopefully adding at least one acceleration data structure.

In terms of deliverables, I should have the single threaded ray tracer with primitive shapes up within a few days. I will spend the rest of my time parallizing that. With any time left over I will attempt to implement the other features mentioned above.