Homework 3- Raytracing using OpenMP

Compile and running instructions:

The code was compiled using the flags `-mavx -O3 -fopenmp -lm`.

The code was executed on the peanut cluster in interactive mode, with `--cpus-per-task=32`.

The repo includes `CMakeLists.txt`.

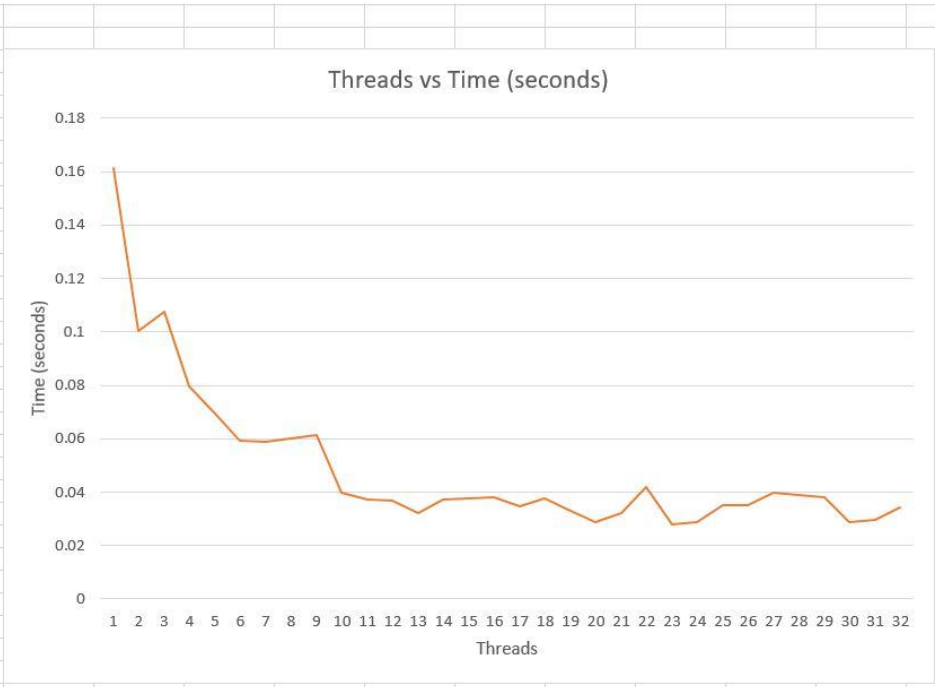
Functionality:

The code in the repo uses the algorithm specified in the document to simulate ray tracing on a sphere at a fixed distance from a light source at a fixed point. The output of the program includes the wall time for the runtime for each thread from 1 to 32 and a demo.bmp file that shows the illuminated sphere.

Timing results:

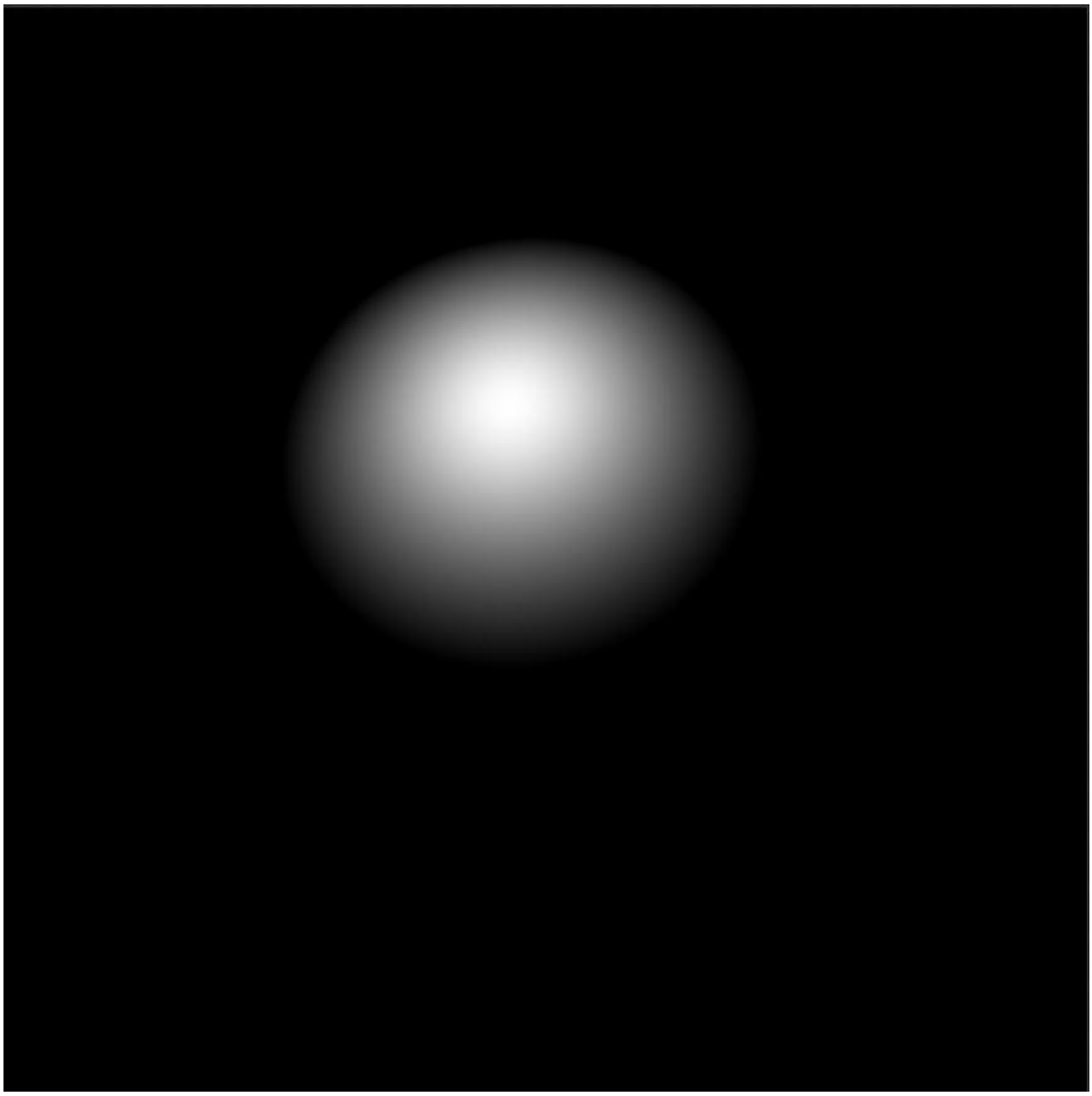
On the peanut cluster, in interactive mode, the following are the times for the threads (strong scaling) with an image size of 4096 * 4096 pixels:

Threads	Time (seconds)
1	0.161214
2	0.100222
3	0.107491
4	0.079387
5	0.069239
6	0.059156
7	0.058865
8	0.060114
9	0.061524
10	0.039874
11	0.037275
12	0.036979
13	0.032152
14	0.03708
15	0.037885
16	0.038078
17	0.034886
18	0.037601
19	0.033064
20	0.028939
21	0.032222
22	0.041745
23	0.027758
24	0.028721
25	0.035194
26	0.035003
27	0.03988
28	0.038852
29	0.038291
30	0.02864
31	0.029801
32	0.034392



Observations: There is a decrease in wall time until about 16 threads.

Picture: The image of the final .bmp file is as below.



Additional Comments:

The repo contains an alternative implementation that uses a struct of vectors. The problem with this was that OpenMP is a bit unpredictable with pointers to structs. This required me to malloc and dealloc a bunch inside the loop, which was driving up the execution time. So I went with an array implementation, which basically does everything in the stack, so I didn't have to worry about private or public variables in the parallel regions because all the variables that were declared inside the loop are private by default. I would love some insight on OpenMP and struct pointers being used as private variables in C.