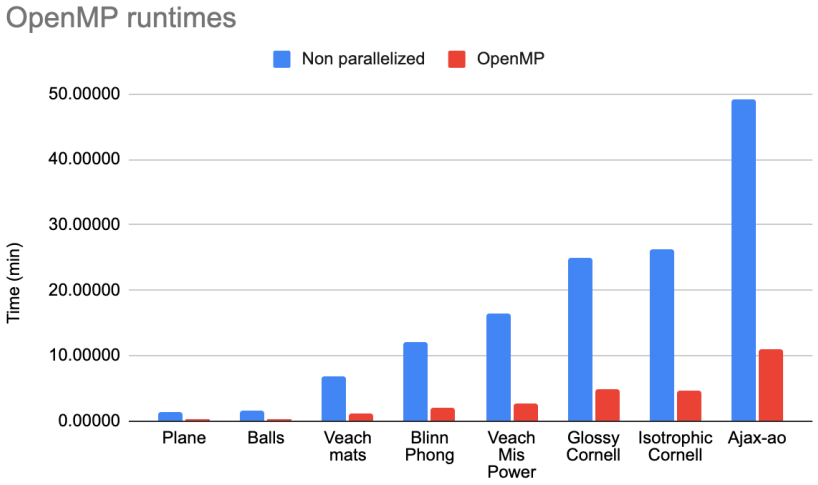
PARALLEL RAY TRACING

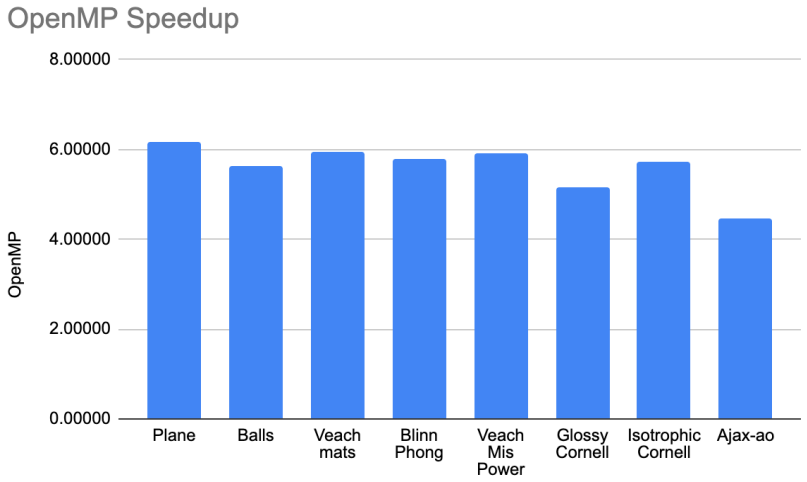
Summary: In our project, we used different parallelization techniques to improve on different aspects of ray tracing. We then applied these parallelization techniques to a wide variety of different scenes with varying difficulty levels measured in terms of scene materials, mediums, and number of objects. Based on the results measured, we delve into explanations for how these relatively performance gains arose.

For each pixel -> generate ray -> collision check -> reflect and recurse

OpenMP

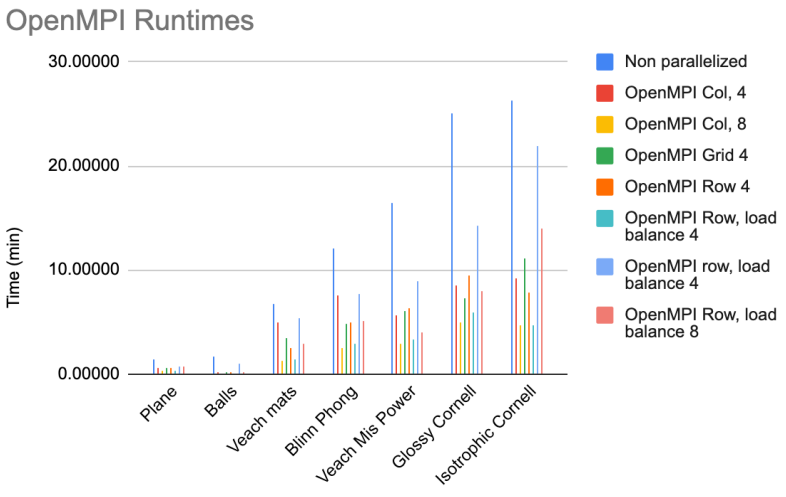
- Fastest overall
- Easy to implement
- Different granularities leading to different runtimes

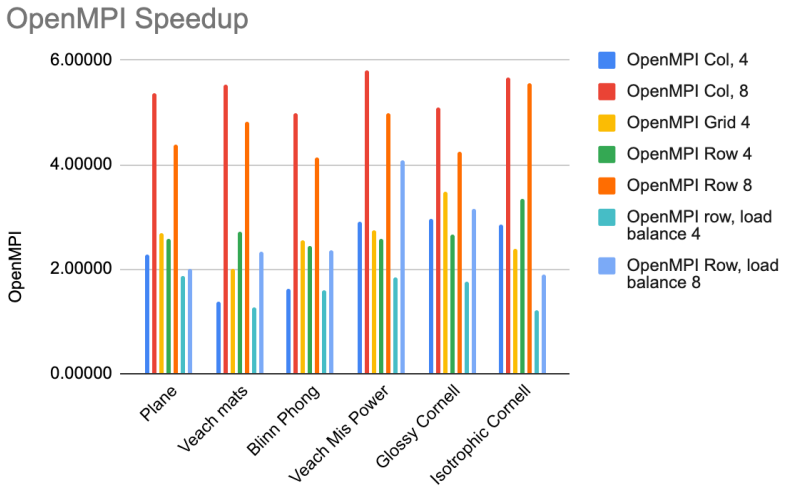




MPI

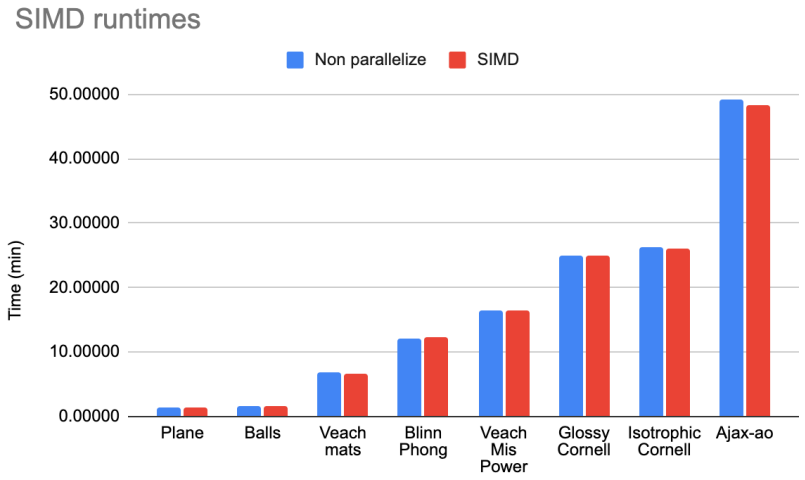
- Similar performance to OpenMP version
- Load balancing based on surface density
- Tradeoff between difficulty in message to send and ability to parallelize

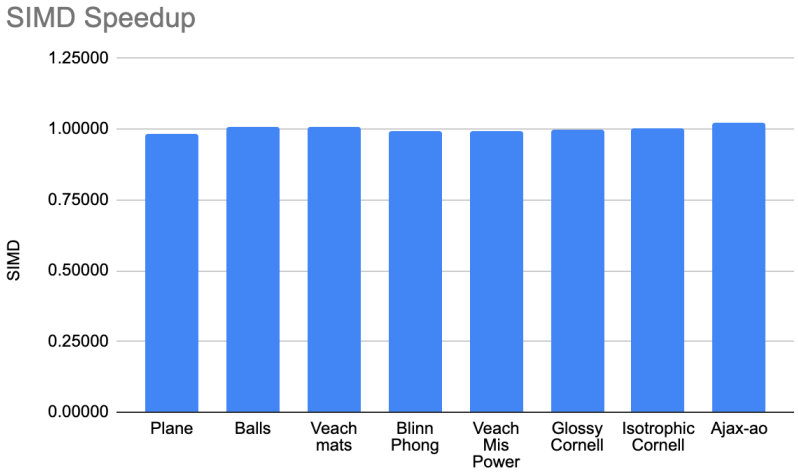




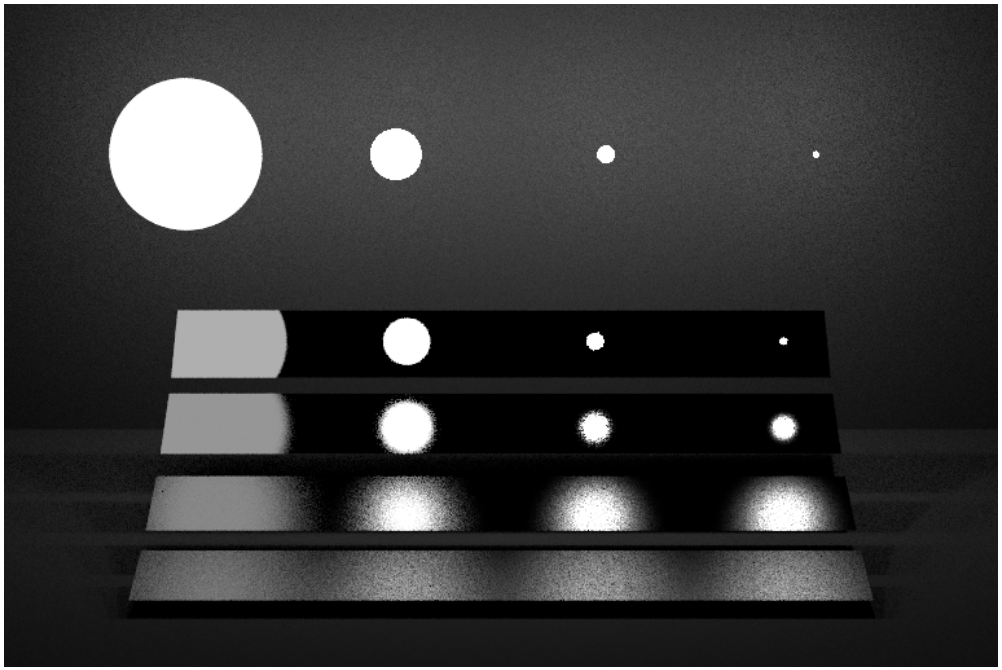
SIMD

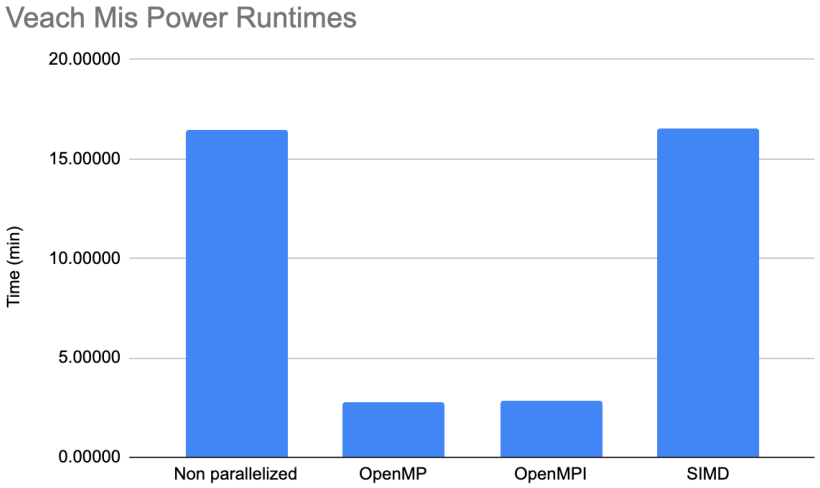
- Extremely difficult due to conflicting principles
- Diverging instructions cause low speedup
- Tradeoff between sample generation and memory access

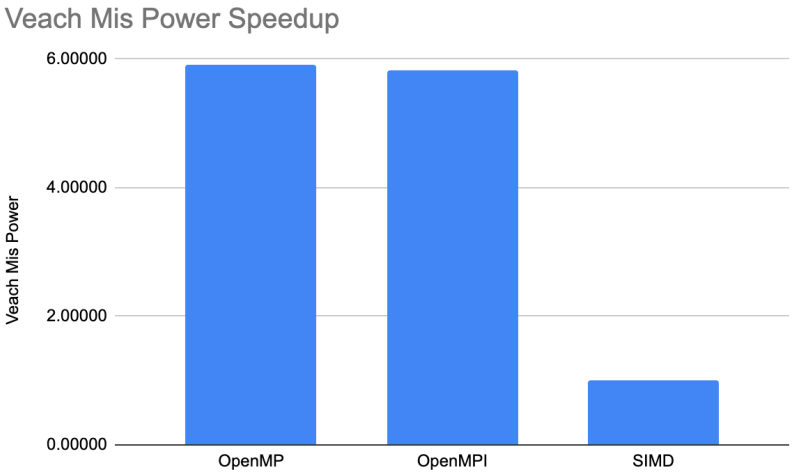




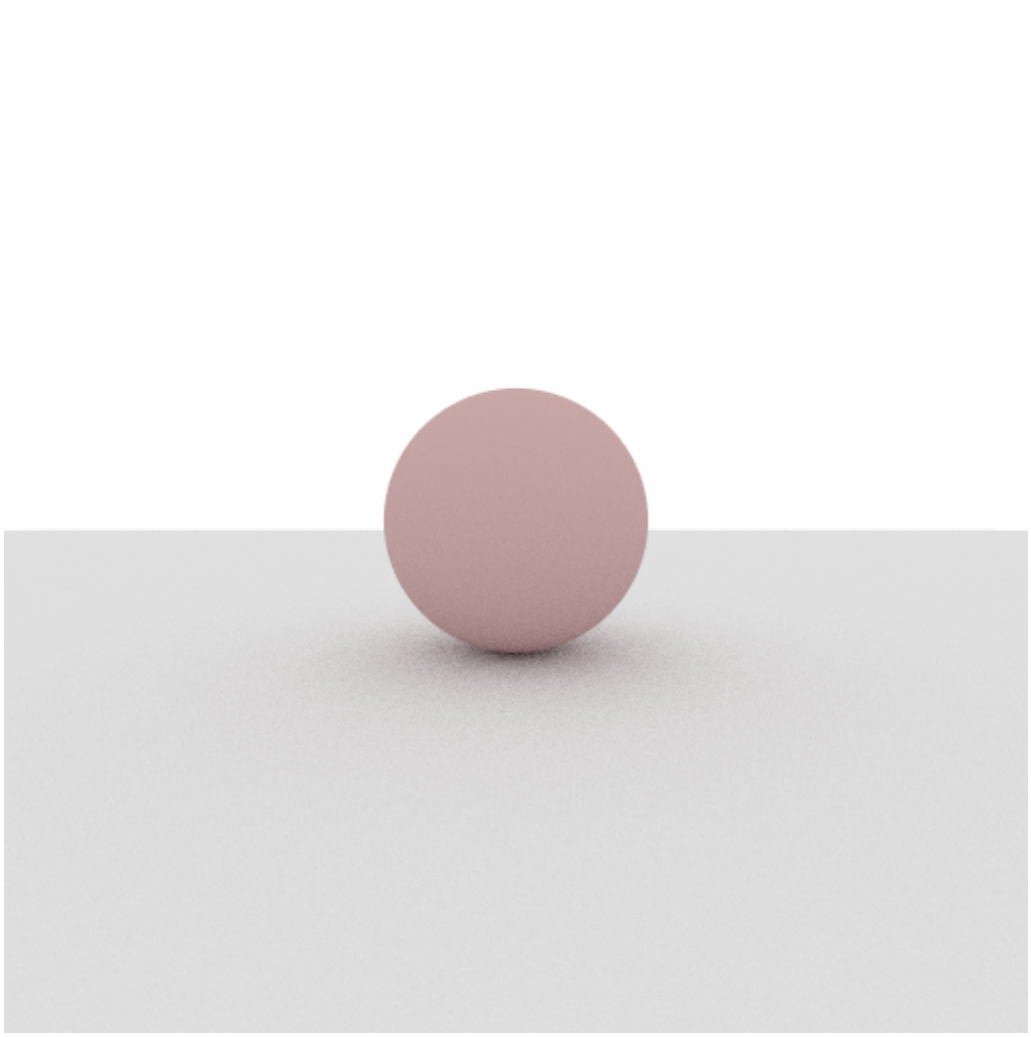
Veach Mis Power

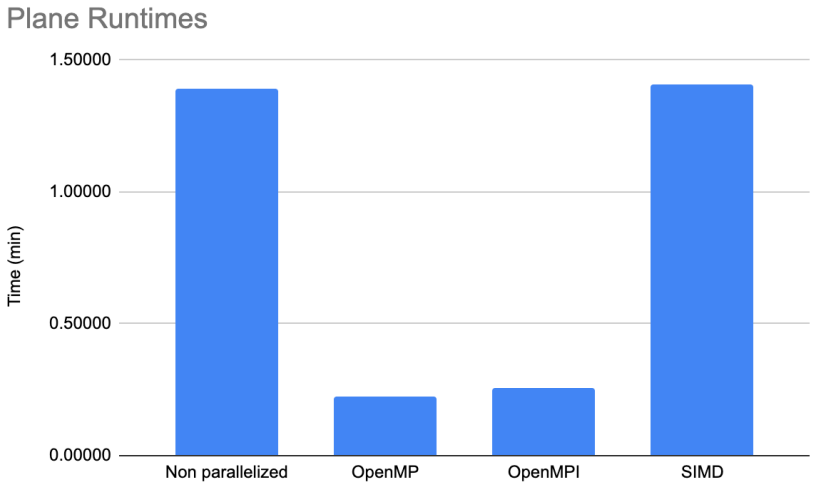


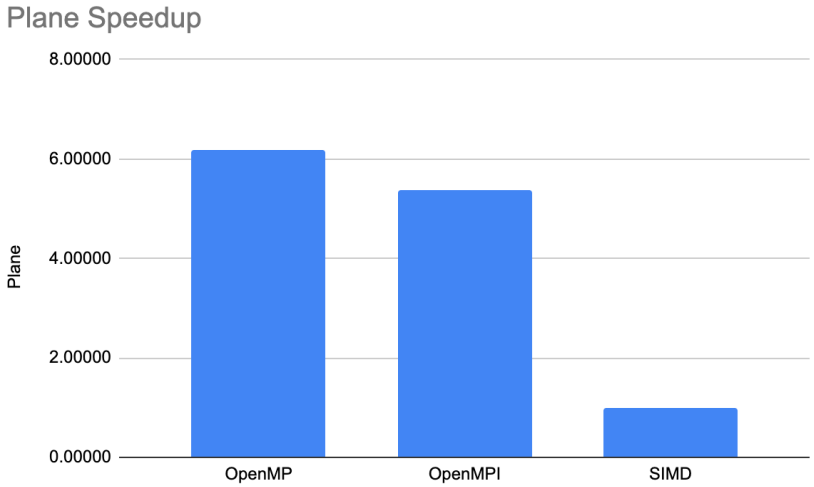




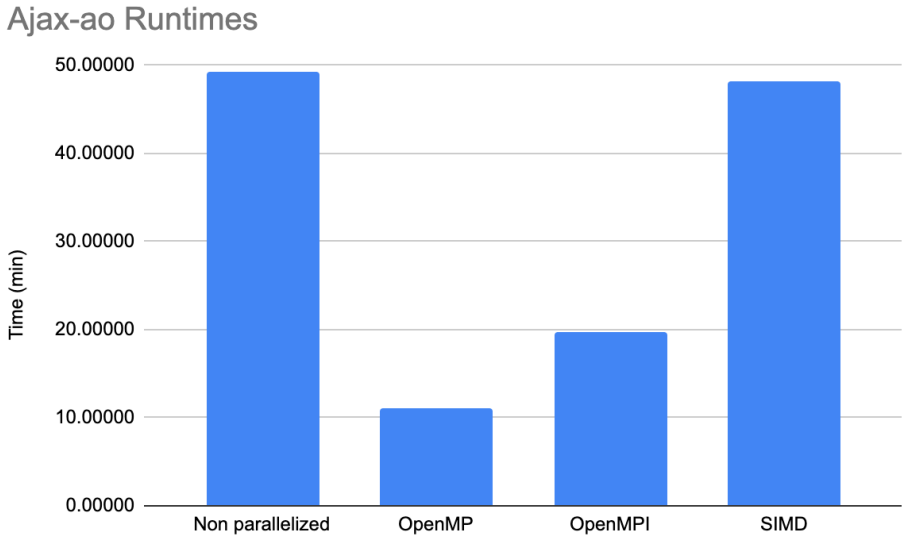
Plane

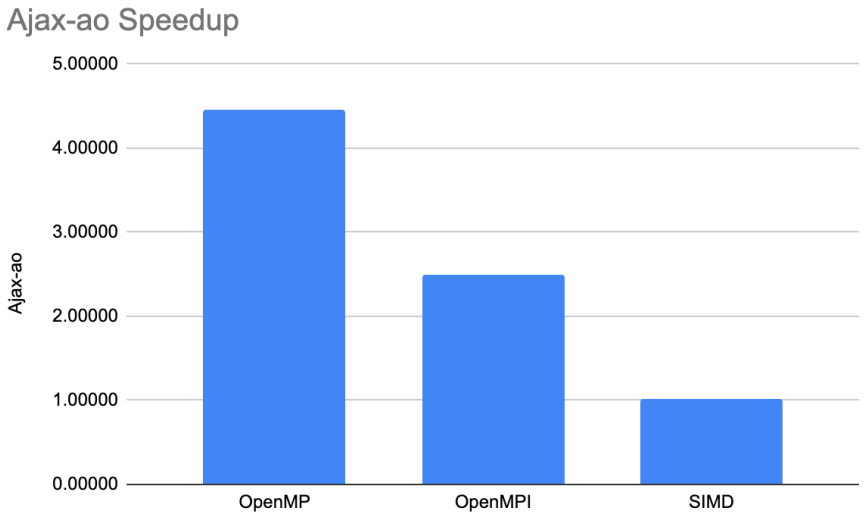



Plane Runtimes
1.50000
1.00000
0.50000
0.00000
Time (min)
Non parallelized
OpenMP
OpenMPI
SIMD


Plane Speedup
8.00000
6.00000
4.00000
2.00000
0.00000
Plane
OpenMP
OpenMPI
SIMD

Ajax-ao


Ajax-ao Runtimes
50.00000
40.00000
30.00000
20.00000
10.00000
0.00000
Time (min)
Non parallelized
OpenMP
OpenMPI
SIMD


Ajax-ao Speedup
5.00000
4.00000
3.00000
2.00000
1.00000
0.00000
Ajax-ao
OpenMP
OpenMPI
SIMD

Isotropic Cornell Box