

Multicore Computing Assignement 4

Joshua BRIONNE - 50241647



Compilation & Execution

Compilation has been performed on **Ubuntu 22.04.3 LTS (Linux)**.

1. Environment

Property	Value
OS Name	Linux
OS Version	6.8.0-52-generic
Architecture	amd64
Available processors (cores)	8
Max memory (MB)	3936
CUDA Environment	Google Colab

2. Compilation

2.1 OpenMP

```
g++ -fopenmp openmp_ray.cpp  
  
./a.out <number_of_threads>
```

2.2 CUDA

```
nvcc -arch=sm_75 cuda_ray.cu  
  
./a.out
```

🕒 Execution Time (in seconds)

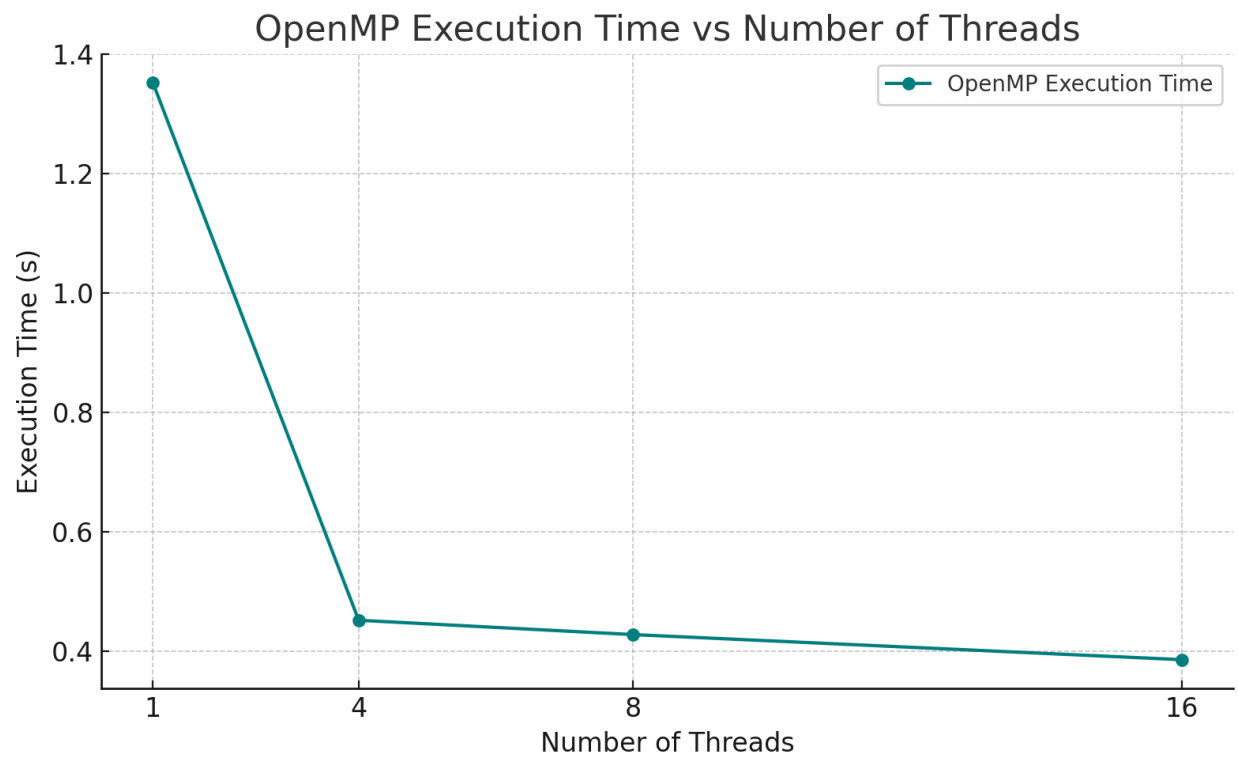
OpenMP

Threads	Time (s)
1	1.352
4	0.452
8	0.428
16	0.386

CUDA

Type	Time (s)
CUDA	0.135

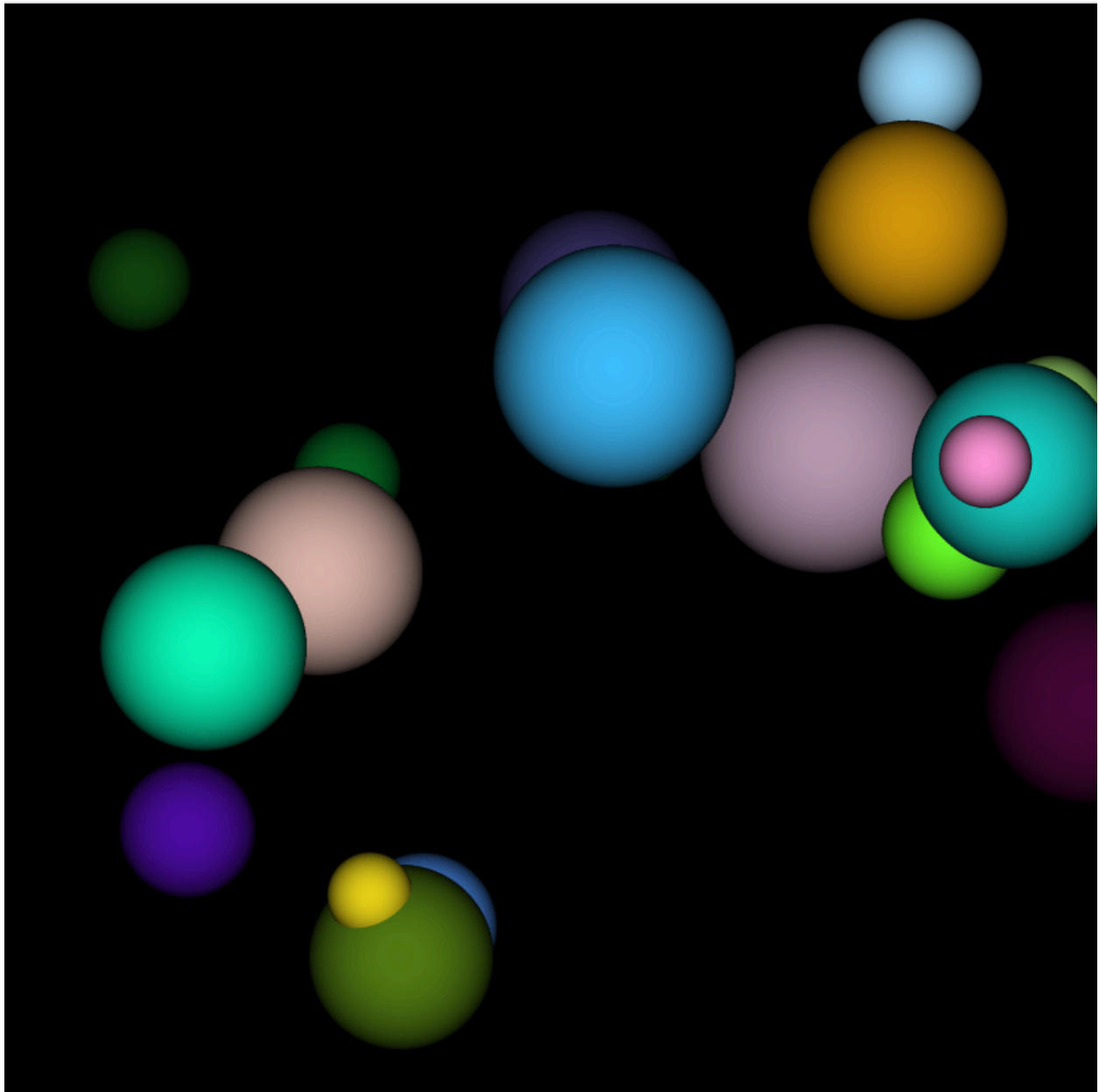
Execution Time Graph



3. Screenshots

3.1 OpenMP

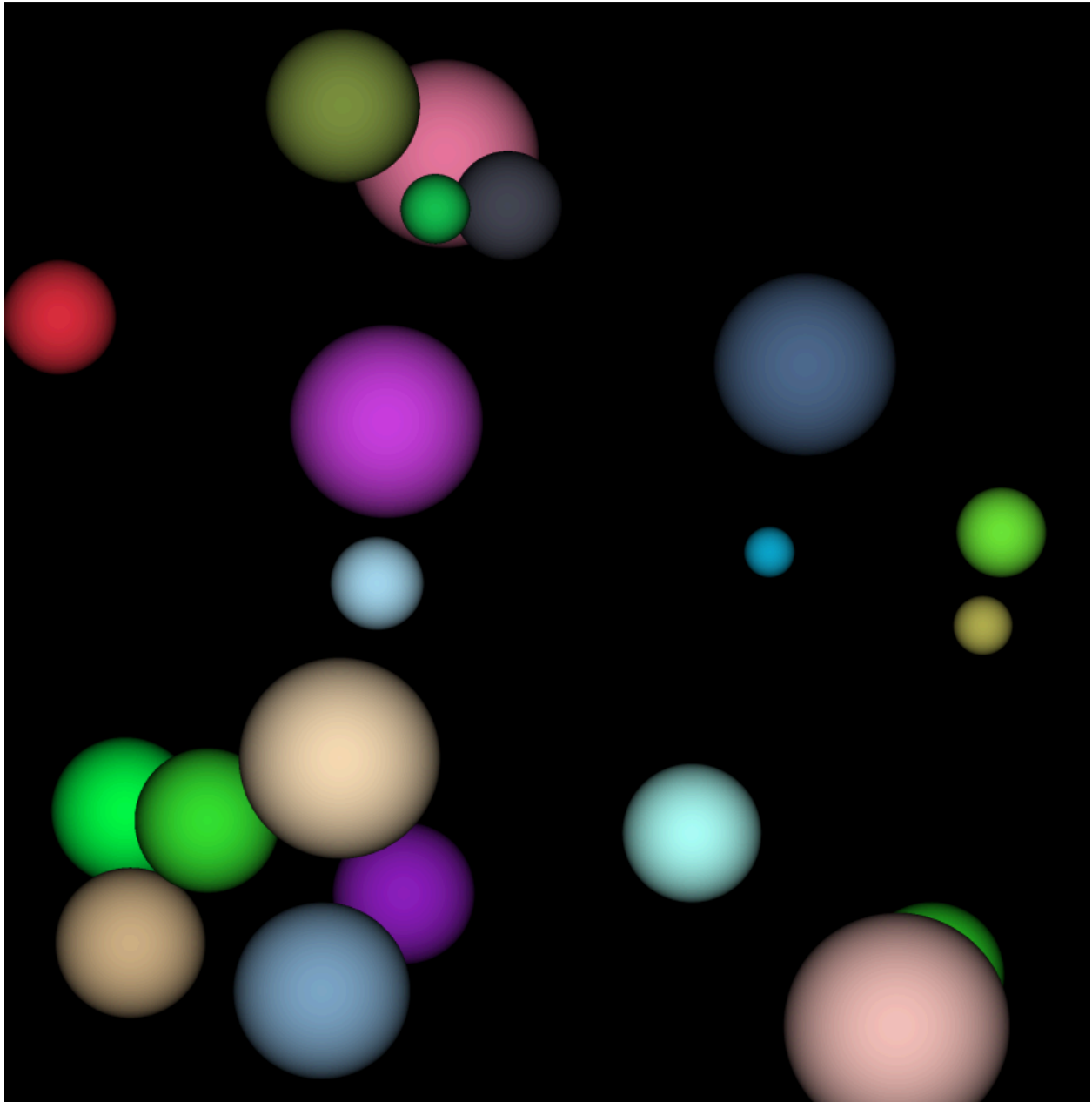
```
→ problem1 git:(main) x ./a.out 16  
OpenMP (16 threads) ray tracing: 0.386 sec  
[result.ppm] was generated.  
→ problem1 git:(main) x █
```



4.2 Cuda

```
!./a.out
```

CUDA ray tracing: 0.001 sec
[result.ppm] was generated.



4. Performance Analysis

4.1 OpenMP

The OpenMP implementation shows a significant reduction in execution time as the number of threads increases. The time taken to complete the ray tracing task decreases from 1.352 seconds with 1 thread to 0.386 seconds with 16 threads, demonstrating the effectiveness of parallelization in reducing computation time.

4.2 CUDA

The CUDA implementation achieves an impressive execution time of just 0.001 seconds, showcasing the power of GPU acceleration for parallel processing tasks. This is a substantial improvement over the OpenMP implementation, highlighting the efficiency of using CUDA for computationally intensive tasks like ray tracing. This demonstrates how a GPU can handle parallel tasks much more efficiently than a CPU, especially for operations that can be massively parallelized and repetitive, such as ray tracing in computer graphics.