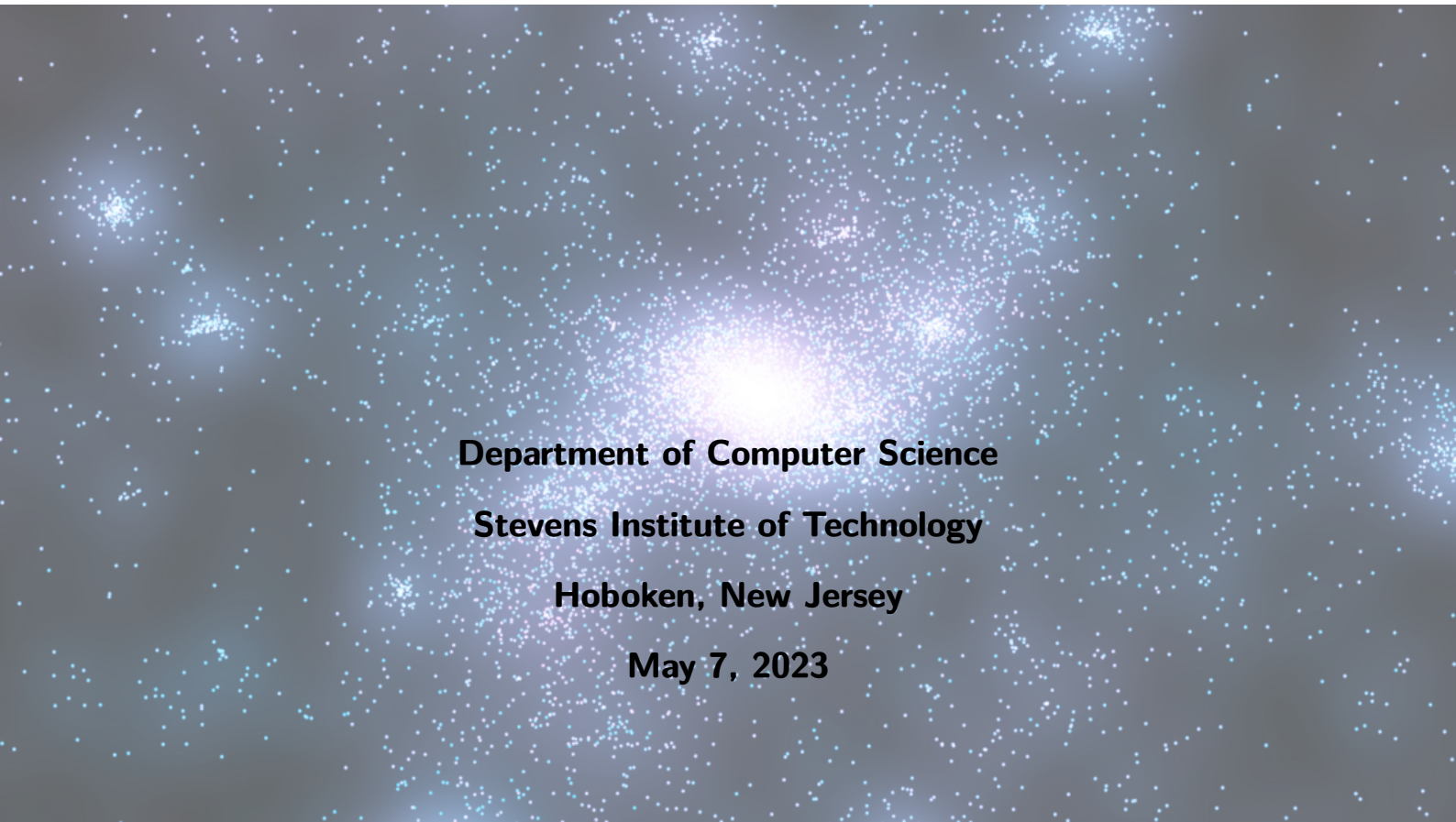
N-Body Simulation

Robert Feliciano

CS677: Final Project Report

Professor Mordohai



Department of Computer Science

Stevens Institute of Technology

Hoboken, New Jersey

May 7, 2023

Contents

1 | Introduction

For my project, I implemented a parallel brute force N-Body Simulation which runs in $O(n^2)$ time. The brute force approach is often times called the "all-pairs N-body simulation" as it simulates the interaction between all the pairs of bodies in the simulation, while other methods, such as the Barnes-Hut simulation, achieve better performance by approximating the interaction by reducing the number of interactions based on the distance between two bodies.

N-Body Simulations can be used to model the interaction between many different things, such as planets, galaxies, and particles. My simulation models the interaction between bodies exerting a gravitational force on one another, with a very large and dense mass being in the middle that also exerts a force on every body in the simulation.

The acceleration of a body due to the gravitational force exerted on it by another body can be modeled by the following equation:

$$a_i \approx G \cdot \sum_{j=1}^{N} \frac{m_j \cdot \vec{r_{ij}}}{(\|\vec{r_{ij}}\|^2 + \epsilon^2)^{\frac{3}{2}}} \tag{1.1}$$

The ϵ is known as a softening factor and is set to 1×10^{-8} in order to prevent division by zero and not have particles collide, but rather pass through each other. My simulation uses this equation to update the velocity of each particle over a small time-step $dt = 0.01$ by multiplying the result a_i by the time-step dt.

2 | Computer Specifications

The simulation was run locally on my PC using WSL2. The specifications are the following:

- CPU: AMD Ryzen 5 5600X, 6-core, 12-Thread
- GPU: NVIDIA - GeForce RTX 3070 Ti

These specifications are important as different CPUs will experience different speed-ups from OpenMP depending on how many threads are available. Along with that, different GPUs will also experience different results based on factors such as their streaming multiprocessor count.

3 | Suitability for GPU Acceleration

Since the interaction between one pair of bodies is completely independent from the interaction between another pair of bodies, this problem is embarrassingly parallel. The entire computation can be parallelized on the GPU. My implementation has the calculation of the force and the updating of every body's velocity vector done on the GPU and the updating of the position of each body is done in parallel on the CPU using OpenMP's SIMD directive once the velocity vector is completely updated for that time-step. With this impleentation, we achieve 100% parallelism for simulating the interaction between all N bodies.

4 | CPU Implementations

ah ah something cpu and then openmp and then yeah explain optimizations from compiler and from me yeah yup

5 | GPU Implementations

ah ah something direct port and then optimized yeah explain flush denorms and stuff

6 | Time Comparisons

yeah a lot of graphs and tables and stuff yeah and yeah explain the times and be like woahhhhhh thats a big time moment

7 | Discussion

what was cool about this brah? what could i do different or add to this and stuff yeah any limits bro?

8 | Conclusion

gpu rules parallelism is great yeah i be balling uh-huh yeah