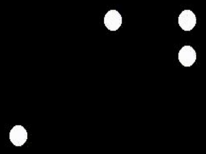
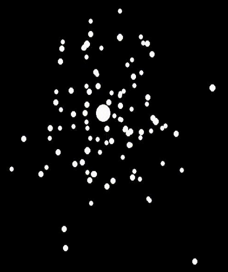
Parallelizing a Gravity Simulator

CS 205

Dennis Du, Aditi Raju, Rafay Azhar,
Sezim Yertanatov, Jared Ni

Problem to Solve





Being able to model gravity in the

universe has crucial scientific applications:

- Helps understand the complex dynamics of interactions b/w celestial bodies
- Useful for research in theoretical physics and supporting astrophysical discoveries
- Crucial for predicting celestial events and space mission planning
- Important educational tool for students and enthusiasts to play around with!

We want an **efficient** way to model and simulate gravity **accurately**, and to **update** and **visualise** our system in **real time**.

Mathematical Model

Newton's law of gravitation: calculate force of attraction between two objects (as 3D vector)

$$\mathbf{F} = G\frac{m_1 m_2}{|\mathbf{r}|^2}\frac{\mathbf{r}}{|\mathbf{r}|}$$

Newton's second law of motion: calculate acceleration (vector) due to gravity

$$\frac{d^2\mathbf{x}}{dt^2} = \frac{\mathbf{F}}{m}$$

Numerical integration to update velocity and position

$$\Delta\frac{d\mathbf{x}}{dt} \approx \frac{d^2\mathbf{x}}{dt^2}\Delta t \qquad \Delta\mathbf{x} \approx \frac{d\mathbf{x}}{dt}\Delta t$$

Merge objects that collide

Assumption: All objects are spheres, even after merging; masses and volumes add; momentum is conserved

Data

- Each spherical object represented by mass, radius, and xyz-coordinates of position and velocity at time 0 (initial conditions)
  - Updated throughout simulation due to gravitational interactions and collisions
- Sources of data for initial conditions
  - Randomly generate
    - simulate evolution of stable behavior from unstable beginning
  - Sloan Digital Sky Survey & data from Gaia mission
    - Solar System and Milky Way data

Why Parallelize?

- Want to be able to do large-scale simulations
  - E.g., the Milky Way contains hundreds of billions of stars
  - Although we might not simulate such a large problem size (depending on resources), there is value in being able to handle millions or billions of stars to simulate galaxy dynamics such as “violent relaxation”, where resources don’t fit in memory

- Need for real-time visualization and simulation that can scale well. Parallelization is thus a solution to the computational bottleneck of a sequential implementation.

It’s a lot of stars

Naive Implementation O(N^2) -> O(N*log(N/p)/p)

For each body **X** in the system:

  For each body **Y** in the system except for **X**:

    Find the contribution of **Y** to the acceleration of **X**

- Each of *n* stars must compute the sum of its gravitational interactions with all other (*n-1*) stars
  → quadratic time if done sequentially, for each timestamp
  - Thus, need more efficient to assign different star's computations to different threads

Parallel Implementation Plan

- SIMD approach that combines OpenMP and MPI
- Distributed memory with MPI allows us to take advantage of more memory space for large simulations – each process handles a sub-region of the overall gravitational system
    - Each process shares the state of the region assigned to it with neighboring processes
- Shared memory with OpenMP allows us to divide work of computing interactions within a particular sub-region among parallel threads
    - Each thread computes updates for a small subset of the objects

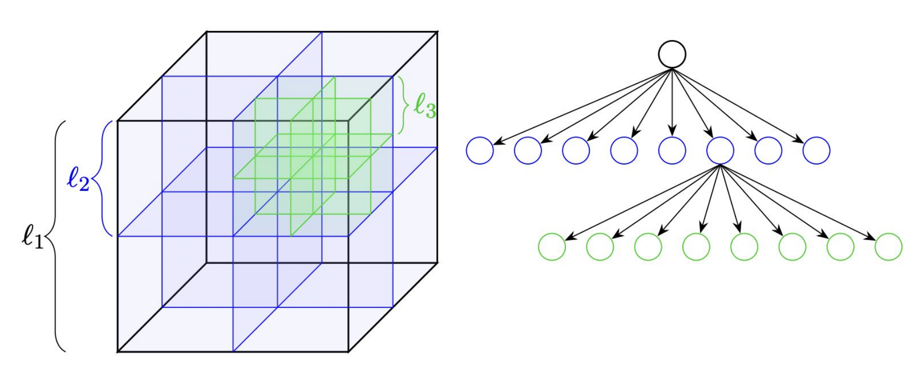
Quick overview of the steps

Key idea is to divide the simulation space into octrees via the Barnes-Hut method.

Each node carries data:

1. The Mass and position of the body (leaf node)

2. Center of mass and other coefficients required to calculate the force (internal node)

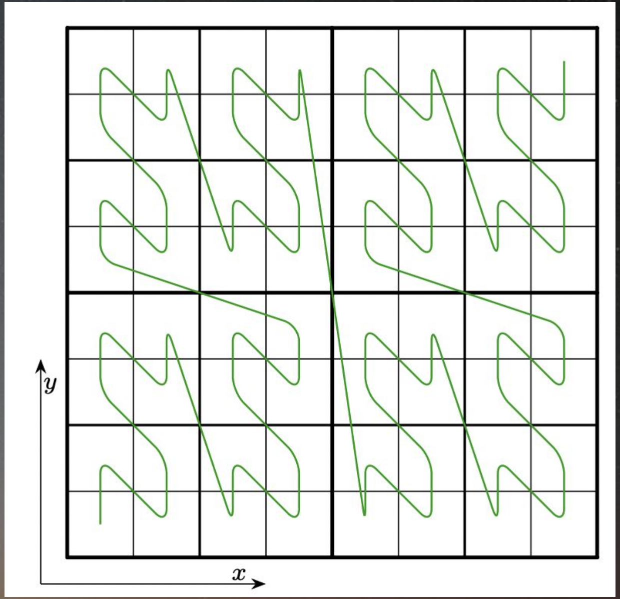


A. Brandt. On Distributed Gravitational N-Body Simulations. University of Western Ontario, 2022.

Quick overview of the steps

2. Assign (N/p) bodies to each process. Each process builds its local octree from the bodies it was given.

We will use a space-filling curve (Morton's ordering) to achieve a balanced partition of bodies between processes.



A. Brandt. On Distributed Gravitational N-Body Simulations. University of Western Ontario, 2022.

Quick overview of the steps

3. Processes merge their local octrees to create a global octree. We will perform a reduction by merging pairs of processes until the root process holds the final tree.

4. The root process broadcasts the global tree to other processes. Then, processes traverse the global tree from top to bottom in parallel to calculate the forces on each of their assigned bodies.

5. Update the velocity and position for each body using the calculated acceleration.

Expected result

$$O(N/p*log(N/p))$$