Modeling Galaxies: Dynamic System Simulations

Hans Montero, Rhys Murray
{hjm2133, ram2269}@columbia.edu

cs4995.003 Final Project Proposal
November 24th, 2020

1 n-body Problem

Given a set of celestial bodies with mass, initial velocity, and initial position, we would like to simulate the motion of these bodies over time under the influence of gravity. Such simulations allow us to model the collisions and interactions of large-scale galaxy clusters. While there is a closed form solution for $n = 2$, no such formula exists for $n \geq 3$, so computationally expensive numerical solutions are required. These numerical methods vary in their approaches to calculating the effect of gravity on each body. We know from kinematics that the gravitational force on one body by another separated by distance r is given by the following (where G is the gravitational constant):

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

A naïve algorithm would run in $\mathcal{O}(n^2)$ time, where for each time step, the algorithm calculates the net force on a given body by iterating over the entire set of bodies and accounting for every single body, regardless of distance. This algorithm clearly will not scale well at the galaxy-level with a huge number of bodies. Further overhead would be added by calculating the positions of the bodies at each step and displaying them. We must seek a more efficient algorithm if we wish to seamlessly model large systems over more fine-grained periods of time.

2 Barnes-Hut Approximation

The Barnes-Hut Approximation seeks to cut down computation by grouping very distant masses together into one larger mass. The first step is to divide up the n bodies into a quadtree (for 2D simulations) to group together nearby masses. Then, for each body in the tree, we calculate the contribution of other bodies in the same way as the naïve algorithm. However, if a group of bodies is sufficiently far away, we aggregate them and use their combined mass and center of gravity for our computation. By leveraging this approximation, the algorithm's time complexity improves to $\mathcal{O}(n \log n)$. Whether a region is considered "distant" or not depends on the ratio of its size to its distance from the body. If this ratio exceeds a threshold value, the region is approximated as above. This threshold value can be adjusted depending on desired speed or accuracy of the simulation.

3 Parallelization

To further optimize this approximation algorithm, we can parallelize the two major computational steps. First, the quadtree construction can be delegated to four threads, as each "quadrant" of the tree is computed independently of the others. We expect to see some minor speed up here, as we are not guaranteed to see even workloads for each of those quadrant constructing threads. Second, and more importantly, we can parallelize the quadtree traversal for calculating the gravitational force on a certain body – one thread per body. This is a perfect example of data parallelism, given the enormous amounts of bodies in realistic models and the fact that these traversals are independent of one another. Parallelizing this step should greatly speed up the runtime of the algorithm, much more so than the parallelization of the quadtree construction. To visualize the algorithm at work, we'll probably output the positions of the bodies into a csv format which can later be ingested by a simple Python plot-renderer or even a Haskell graphics library.