Tree Algorithm of N-body Simulation

CUDA Parallel Programming HW07 / B07202025

Motivation

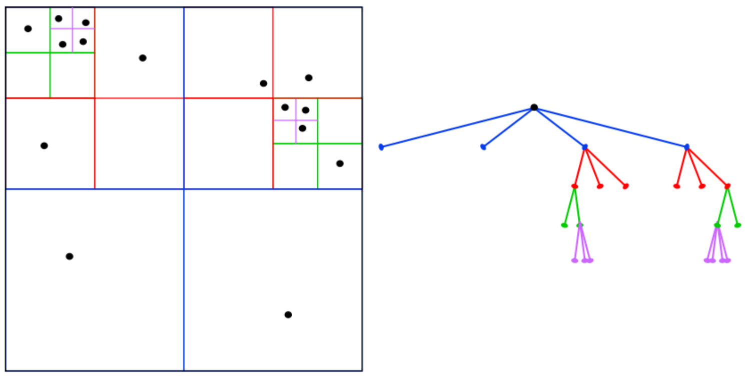
The dynamic of a self-gravitation system can be simulated in two different ways. Naively one would think of direct N-body integration, which involves computation of $N(N-1)/2$ forces to be calculated in each time step. The second way is to model the potential field by the density distribution in the region of interest (Solving the Poisson equation) and calculated the force on each particle. Number of computations of this approach grows only $Nlog(N)$. In 1986, Josh Barnes and Piet Hut published an algorithm that directly calculated the force interactions between each particles or 'clusters' and the cost growth with only $Nlog(N)$, which is known as the Tree Algorithm of N-body simulation.

If we want to calculate force of a N-body system that acts on a single particle A, we directly sum the forces of the neighboring particles and view the particles far from A as a single particle, presented by its center of mass. Details of the algorithms and the parallelization is described in the following sections.

Algorithm

1. Build the Quad Trees (in 2D, Oct Trees in 3D)

   We first construct the quad-trees to store the information of each particles. We begin with an empty large cell that can contain all of the particles. Next, we load the particle into the box one by one. As long as there are two particles exists in a single cell, we divide the cell into four sub-cells. At the end, each cell (no matter how large or small) will contain only one single particle.



   As for the tree structure, each node of the tree contains the information of the corresponding cell, including its sub-cells. The nodes records the number and the center of mass of particles inside as long as the side length. If the particles distribute uniformly, the height of the tree has an upper bound of $log_4 N$ and the cost of building the tree is $Nlog_4 N = O(NlogN)$.

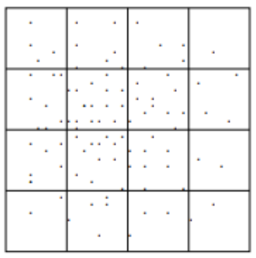
2. Compute the Force on each particles

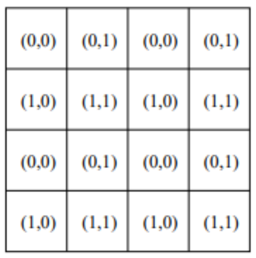
   We want to calculate force of node n on the single particle k. If there is only one particle in node n, calculates the force of it and k directly. If not, calculate the distance of particle k and the center of mass of the node. Let the distance be r and the side length of the node n to be D. Set a fixed accuracy parameter θ. If $\frac{D}{r} < \theta$, means that the distance between the particle k and the node n is larger than the side length of the cell. We can view the particles in the cell as a cluster and use only their total mass and center of mass rather than calculate the particles inside it respectively. To calculate the force of a single particle takes at most $O(logN)$ since the depth of the tree is $log_4 N$ while the step takes total $N\, log_4 N = O(NlogN)$

3. Evolve the velocity and position of the particles in the system with the known force. We calculate the total potential energy and the kinetic energy to estimate the error of each time step.

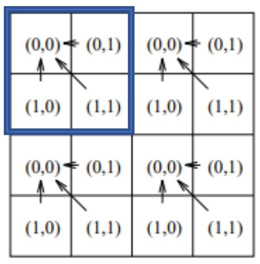
Parallelization

1. Domain decomposition and mapping
   Divide the domain into r sub-domains. The number of sub-domains (r) is larger than the number of processors (p).
2. Tree construction
   Each processors calculate the tree of each sub-domains. This calculation doesn't require communication between each processors. Then, the local trees are merged to form global tree. This process is similar to parallel reduction. The blue rectangle in the right figure presents a block. We use shared memory to record the total mass and the position of center of mass in each thread. Last, we return the information of each block and calculate the up-layer of the tree.

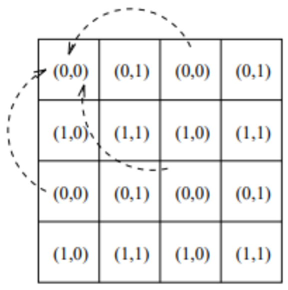

(a)



(b)



(c)



(d)

3. Force evaluation
   For a single particle k, there are three situations that should be considered.
   (1) k-to-all broadcast : Starting from the root of the tree, particles far from the target particle k is considered as a single one and the information can be taken from nodes near the roots. Since these node will be frequently accessed, we let them to be accessible to every threads.
   (2) Force evaluation : For particles near k, the information is in the local node and one can calculate it in the each processor at the same time.
   (3) All-to-all broadcast : There are still some particles that is close to particle k but in the different nodes. We pass the information of k (three float number) to the corresponding processor and calculates forces at that processor. Passing the force (also three float number) to the origin processor and sum them up.
4. Accrue force and move particles
   Simply do this in each processors. There are still room of improvement of load balance between each processors.

Excepted Results

We except to construct a CUDA code that allow users to input the initial position and velocity of particles, and output the result of these particles after a period of time. There are still some hyperparameters that can set by users such as the accuracy parameter θ and the time step in each evolution. The system will interact with gravitational force while it can easily be replaced by Columb’s law or other interactions. We plan to output the position and velocity of each particle and plot the result with python.

References

1. This project is inspired by the candidate final project ‘Tree Algorithm’ in the Computational Astrophysics Course, 2021, NTU.
2. A hierarchical O(N log N) force-calculation algorithm, Josh Barnes & Piet Hu, 1986.
3. Scalable Parallel Formulations of the Barnes-Hut Method for n-Body Simulations, Ananth Y. Grama, Vipin Kumar, and Ahmed Sameh, 1994.
4. Cosmology Applications : N-Body Simulations, Lecture Slides of Dr. James Demmel and Dr. Kathy Yelick, University of California, Berkeley, 1999.