Final Project Proposal:

CUDA Implementation of the modified Barnes Hut Algorithm for Galaxy simulation in 2D

Yue Zhao

1. Goals

1. Write a two-dimensional N particles simulation by gravitational forces based on the modified Barnes-Hut algorithm to simulate a simplified spiral galaxy.
2. Optimize the implementation appropriate to GPU and CUDA specifications.
3. Visualize the simulation with colored particles.
4. Carry out an performence analysis on the implementation.

2. Problem description:

The project uses N-body simulation to simulate a two-dimensional collisionless particles' behavior of N identical particles by gravitational forces and to simulate the evolution of the simplified galaxy.

The simplified model for spiral galaxies consist of a galactic disk and a galactic bulge with radius Rd, Rb and mass Md and Mb.

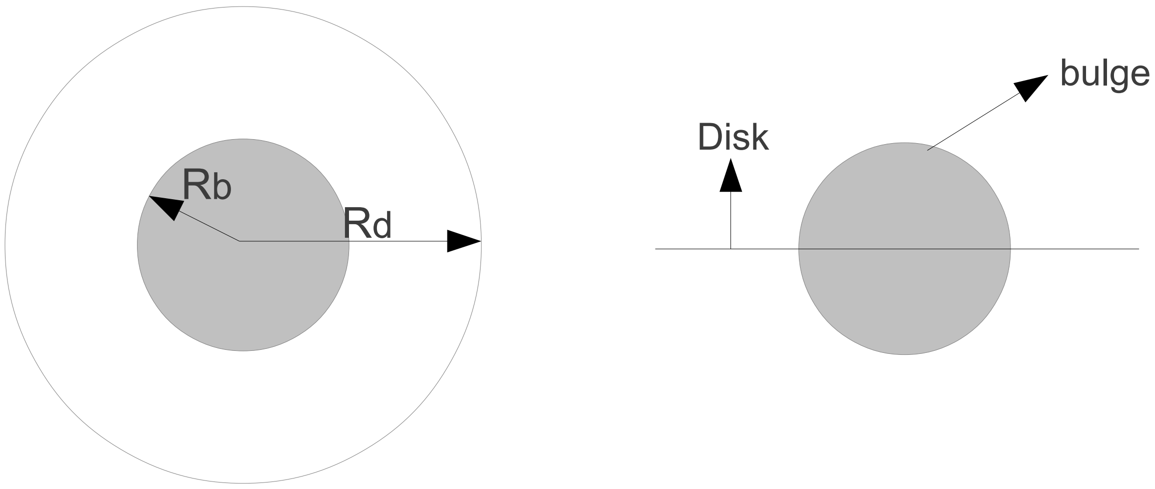

Fig 1. Simplified model of a galaxy

According to Newton's laws of motion and gravitation, the motion of N particles are affected by gravitational forces:

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

For the direct algorithm, the solution works as follows, giving a particle distribution , setting the initial velocity for every particles, and using the set of particles to find:

1. The net force acting on each particle at time *t.*
2. Each particle's acceleration at time *t* by using the net force.
3. Each particle's velocity at time $t + \Delta t$ by using its acceleration at time t.

4. Each particle's position at time $t + \Delta t$ by using its its velocity.

```
t=0
while t<t_end
    for i = 1 to N
    Compute f(i) = ∑ f(i,j) force on particle I.
    for i = 1 to N
    Move the particle i under f(i) for time t+dt
    t=t+dt
end while
```

Fig 2. Pseudo-code for naive algorithm

3. Methods

This algorithm seems to require at least O(n^2) time to work, because for every particles i, the force on i is f(i) = ∑ f(i,j) (j = 1 to N and j != i). And it's easy to implemented in parallel.

```
t=0
while t<t_end
    for_each i = 1 to N (parallel)
    Compute f(i) = ∑ f(i,j) force on particle I.
    for_each i = 1 to N (parallel)
    Move the particle i under f(i) for time t+dt
    t=t+dt
end while
```

Fig 3. Pseudo-code for naive parallel algorithm

However, if N is huge, the algorithm could still be slow.
Here is the classical Barnes-Hut(BH) algorithm that can improve O(n^2) to O(nlogn) or even better.[1]
The improvement is mainly for computing f(i) = ∑ f(i,j):

1. Subdivided the space by using quad tree structure.
2. Groups of distant bodies in the same cell can be treated as one in force calculation

```
t=0
while t<t_end
    Build the Quad Tree
    For each node in tree compute center and total number of particles in each node.
    for_each i = 1 to N
        Compute the f(i) by using the center and total number of nodes in
        quad tree(if they are far enough).
    Move the particle i under f(i) for time t+dt
    t=t+dt
end while
```

Fig 4. Pseudo-code for Barnes-Hut algorithm

The modified Barnese-Hut algorithm is by building an interaction list for each particle in a cell containing a modest number of particles and reusing the list for each particle in the cell to improve the performance[2]. Classical BH algorithm normally loop the particles from 1 to N and compute the force by traversing the tree. However, the modified BH algorithm loops over the members of the selected set of cells C. Suppose c is a cell in C, then all particles in c shared the same interaction list for computing the forces on each particle. A cell c is in C if it encloses a total of n_{crit} or fewer particles and its parent cell encloses more than n_{crit} particles. So this modified algorithm reduces the calculation cost roughly by a factor of n_{crit} , where n_{crit} is the average number of particles in the group C. As increasing the n_{crit} , the number of interaction lists reduces, but the interaction list will also be longer. The optimal value of n_{crit} strongly depends on the ratio of the speed of the host PC to that of the GPUs [3]. The normal BH algorithm requires only O(nlogn) operations per time step. The modified BH algorithm may reduce the force computing time based on the factor n_{crit} .

According to the algorithm, we can parallelize each computation step with using GPU. And each thread computes the force on a single particle by traversing the tree, which means that the improvement in speedup will be significant by the amdahl's law.

The BH algorithm is challenging to implement in CUDA. Because:

1.It repeatedly builds and traverses an irregular tree based data structure.
2.Perform a lot of pointer-chasing memory operations.
3.Typically expressed recursively.[1]

In this algorithm, it used the quad tree structure to map data, and the calculation of forces on a particle becomes traversing tree elements. In order to solve these problems, what should be implemented are as follows:

1.Building and traversing the tree must be implemented in iteration form.
2.The quad tree should be array-based structure instead of the nodes structure in heap.
3.Keep data on GPU between kernel calls.[1]

Performance analysis

1. The bottleneck of the memory bandwidth

In this algorithm, the approach runs the entire Barnes Hut algorithm on the GPU avoids slow data transfers between the CPU and the GPU. And in every time-step the blocks must fetch all the data from global to threads. So reducing the global memory bandwidth is essential to improving the performance.

2. Maximum problem size for the implementation

Because the whole algorithm runs in GPU and the tree traverse code must be converted into iterative code (needs a stack), so the memory consumption in GPU will significantly increase. So the problem size is limited by the memory size in GPU and data structure.

3. The performance comparison of modified BH and classical BH.

To evaluate the improvement (or not) of the implementation of modified BH than the classical BH. Considering the improvement in bandwidth, memory consumption, load balancing in force computing,

4. Reference

[1] Martin Burtscher, An Efficient CUDA Implementation of the Tree-Based Barnes Hut n-Body Algorithm , Chapter 6, GPU Computing Gems Emerald Edition (2011) 75-92.

[2] J. Barnes, A modified tree code: don’t laugh; it runs, J. Comput. Phys. 87 (1) (1990) 161–170.

[3] Tsuyoshi Hamada. etc, A novel multiple-walk parallel algorithm for the Barnes–Hut treecode on GPUs – towards cost effective, high performance N-body simulation, 24 (2009) 21–31, SpringerLink.