Ricardo Villacana

Accelerating the N-Body Gravitational Calculator Using CUDA

Video Link: https://youtu.be/PHWbimjlQvM

Overview:

This project aimed to accelerate the N-Body gravitational calculator using CUDA. The N-Body problem is a classical problem in physics that predicts the individual motions of a group of celestial objects interacting with each other gravitationally. Solving the N-Body problem is computationally demanding due to the requirement of calculating the interaction between every pair of bodies for every time step. This project aimed to leverage the parallel processing power of the GPU to speed up these computations.

CPU Implementation:

The N-Body simulation problem involves predicting the individual motions of a group of celestial bodies that are interacting gravitationally. The CPU implementation of the N-Body simulation problem is a sequential approach to this task.
The main steps in the CPU implementation of the N-Body simulation are:

1. Initialization: The initial conditions of the system are set. This involves specifying the initial positions, velocities, and masses of all the bodies. The positions and masses are assigned randomly, while the forces and velocities are given an initial value of zero. The bodies are represented as objects in an array.
2. Force Calculation: For each body, the gravitational force exerted by every other body is calculated. This is done using Newton's law of universal gravitation, which states that every particle of matter in the universe attracts every other particle with a force that is directly proportional to the product of their masses and inversely proportional to the square of the distance between their centers. The total force on each body is the vector sum of the forces due to all other bodies. Since every pair of bodies needs to be considered, this step involves nested loops and so it has a O(n^2) time complexity.
3. Update Positions and Velocities: Once the forces on each body are calculated, the positions and velocities of the bodies can be updated. The velocity of each body is updated by adding the acceleration (which is obtained by dividing the force by the mass of the body, using Newton's second law of motion) times the time step (set time difference between steps) to the current velocity. The position is then updated by adding the velocity times the time step to the current position.

4. Repeat: Steps 2 and 3 are repeated for the desired number of time steps to simulate the motions of the bodies over time.

In the CPU implementation, the calculations for each object are performed in sequence, with each object waiting for the previous one to complete before starting its calculation. This means that even though the calculations for each body are independent of each other, they are performed one after the other, which can be quite slow, especially when there are a large number of bodies. This is the primary motivation for accelerating the N-Body simulation using a GPU, which can perform many calculations simultaneously.

GPU Implementation:

The GPU was utilized to accelerate the application by parallelizing the computations of the force calculation and the movement of the objects. In the traditional N-Body problem, each body's new position is calculated sequentially, leading to an O(n^2) time complexity. By using a GPU, we could calculate the forces acting upon an object and the new positions of multiple bodies simultaneously, greatly reducing the computation time. The problem space was partitioned into threads and thread blocks in a way that each body's forces and new position were calculated by one thread. This allowed us to take full advantage of the GPU's parallel processing capabilities. The stage of the software pipeline that was parallelized was the calculation of the gravitational forces and the update of the velocities and positions of the bodies.

Evaluation/Results:

The evaluation of the project focused on the execution time comparison between the CPU and GPU implementations for both the force calculations and the particle movement calculations. Each of these computations was performed ten times with 10,000 celestial objects in the system, and the execution time was recorded for each step.

Force Calculation

The force calculations executed on the GPU showed consistent performance throughout the 10 steps, averaging around 23.1 ms. This contrasts with the CPU implementation, where the force calculation execution times ranged between 743.9ms and 770.9 ms, with an average time of 756.2 ms. This signifies that the GPU implementation for the force calculation was approximately 32.7 times faster than the CPU implementation.

Move Particles Calculation

The particle movement computations also exhibited significant improvements with the GPU. The GPU execution times varied between 0.008416 ms and 0.03008 ms, with an average time of about 0.012632 ms. On the other hand, the CPU execution times for moving particles ranged from 0.064 ms to 0.113 ms, with an average of about 0.0756 ms. This shows that the GPU implementation was approximately six times faster than the CPU for moving particles.

Additionally, the results from both the force calculations and the move particles calculations from the GPU and CPU matched at each step, confirming the correctness of the parallel implementation on the GPU.

<u>Output Example:</u>

```
GPU Force Calculation execution time: 23.1018 ms
CPU Force Calculation execution time: 752.497 ms
The force calculations from the GPU and CPU match!
GPU Move Particles Calculation execution time: 0.01152 ms
CPU Move Particles Calculation execution time: 0.067 ms
The Move Particles calculations from the GPU and CPU match!
Completed step 1/10
GPU Force Calculation execution time: 23.0859 ms
CPU Force Calculation execution time: 761.561 ms
The force calculations from the GPU and CPU match!
GPU Move Particles Calculation execution time: 0.009568 ms
CPU Move Particles Calculation execution time: 0.066 ms
The Move Particles calculations from the GPU and CPU match!
Completed step 2/10
GPU Force Calculation execution time: 23.1149 ms
CPU Force Calculation execution time: 770.855 ms
The force calculations from the GPU and CPU match!
GPU Move Particles Calculation execution time: 0.03008 ms
CPU Move Particles Calculation execution time: 0.099 ms
The Move Particles calculations from the GPU and CPU match!
Completed step 3/10
GPU Force Calculation execution time: 23.0897 ms
CPU Force Calculation execution time: 763.023 ms
The force calculations from the GPU and CPU match!
GPU Move Particles Calculation execution time: 0.009344 ms
```

```
CPU Move Particles Calculation execution time: 0.113 ms
The Move Particles calculations from the GPU and CPU match!
Completed step 4/10
GPU Force Calculation execution time: 23.0867 ms
CPU Force Calculation execution time: 743.886 ms
The force calculations from the GPU and CPU match!
GPU Move Particles Calculation execution time: 0.010432 ms
CPU Move Particles Calculation execution time: 0.076 ms
The Move Particles calculations from the GPU and CPU match!
Completed step 5/10
GPU Force Calculation execution time: 23.1013 ms
CPU Force Calculation execution time: 764.174 ms
The force calculations from the GPU and CPU match!
GPU Move Particles Calculation execution time: 0.013568 ms
CPU Move Particles Calculation execution time: 0.073 ms
The Move Particles calculations from the GPU and CPU match!
Completed step 6/10
GPU Force Calculation execution time: 23.0898 ms
CPU Force Calculation execution time: 750.636 ms
The force calculations from the GPU and CPU match!
GPU Move Particles Calculation execution time: 0.008416 ms
CPU Move Particles Calculation execution time: 0.066 ms
The Move Particles calculations from the GPU and CPU match!
Completed step 7/10
GPU Force Calculation execution time: 23.1013 ms
CPU Force Calculation execution time: 751.26 ms
The force calculations from the GPU and CPU match!
GPU Move Particles Calculation execution time: 0.012224 ms
CPU Move Particles Calculation execution time: 0.064 ms
The Move Particles calculations from the GPU and CPU match!
Completed step 8/10
GPU Force Calculation execution time: 23.0819 ms
CPU Force Calculation execution time: 759.152 ms
The force calculations from the GPU and CPU match!
GPU Move Particles Calculation execution time: 0.009824 ms
CPU Move Particles Calculation execution time: 0.068 ms
The Move Particles calculations from the GPU and CPU match!
Completed step 9/10
GPU Force Calculation execution time: 23.108 ms
CPU Force Calculation execution time: 745.24 ms
```

```
The force calculations from the GPU and CPU match!
GPU Move Particles Calculation execution time: 0.010752 ms
CPU Move Particles Calculation execution time: 0.073 ms
The Move Particles calculations from the GPU and CPU match!
Completed step 10/10
```

<u>How to Run the Code:</u>

1. Make sure you have the CUDA toolkit installed.
2. Use the command “make” to compile the program, assuming you have the makefile
3. Run the program with the command “./main” and the necessary components are randomly generated and the computation will start.

<u>Problems Faced:</u>

While significant progress was made in speeding up the N-Body gravitational calculator using the GPU, the attempts I made to implement the shared memory as a further optimization step encountered significant obstacles. For reasons that could not be completely determined, the incorporation of shared memory led to the results from the GPU calculations zeroing out. Despite some significant troubleshooting and analysis, I could not identify the exact cause of this issue. This problem is something that I could explore in the future for potential improvement in the program's performance.