Molecular Dynamics (MD) simulator for water using CUDA

Chunzhi Wu and Vincent Tombari

URL

https://github.com/TranzWu/15418-final_project

Summary

In this project, we are going implement a parallelized version of Molecular Dynamic engine to simulate water using CUDA.

Background

Molecular Dynamics (MD) is a widely used and powerful computational technique that provides the insight into the dynamic behavior of molecular systems. It has been used wide in many different fields like drug discovery, materials science and biological systems.

Here's the pseudo code of Velocity Verlet algorithm, which is common used in molecular dynamics simulation to integrate the equations of motion.

```
dt = time_step  # Time step
mass = particle_mass  # Mass of the particle

# Initial conditions
position = initial_position
velocity = initial_velocity
force = compute_force(position)  # Function to compute the force at the
initial position

# Integration loop
for step in range(num_steps):
    # Velocity Verlet algorithm

    #update velocity
    velocity += 0.5 * force / mass * dt

    # Update position
    position += velocity * dt

    # Update force at the new position
    force = compute_force(position)

    # Update velocity
    velocity += 0.5 * force / mass * dt
```

Parallelizing MD code is important because the simulation time scales as N^2. As a result parallelizing MD code is essential for addressing the computational demands of large and complex systems.

Challenges

This problem requires a lot of computation due to the scale and number of particles in simulation. There is a high-level of parallelizability since each particle's forces can be computed on it's own thread.

The workload is simple to balance, as we can simply assign a particle to each thread. Each particle's forces may be computed independent of the computation of other particle's forces.

The amount of communication is fairly high, as we will have to synchronize all particle updates each frame. We believe we can optimize this by segmenting our particle space and only synchronizing locally.

There is not too much divergenexecution, as the force equation will be the same for all particles, but every particle will have very different parameters to the force computations.

One large problem is keeping track of each particle's neighbors. A particle's update is determined only by a small subset of neihboring particles. With the particles moving around rapidly, this neighbor set will be rapidly changing. This may cause a significant increase in synchronization of this neighbors datastructure. Maintaining the neighbors set with minimal synchronization will be a large challenege in our project.

One last challenge is simply in the complexity of the force computations. The forces acting on a particle are the result of many different molecular properties. Computing all of these efficiently and updating the molecules accordingly will be challenging.

Resources

We will be using the same GHC machines which we used in homework 2.

Goals and deliverables

Goals plan to achieve

1. Build a serial version of working MD code (including functions to calculate energies of the system, temperature and pressure)
2. Use CUDA to parallelize the MD code
3. Optimize the CUDA code to make it faster (for example, combine certains part of the functions together to reduce repeated calculation)
4. Achieve at least 20x speedup.

Goals hope to achieve

1. Implement an acceleration data structure called a 'neighbor list' to further accelerate the code.
2. Implement additional functions to calculate some important properties of the system. For example, for the transport properties, we can calculate the Mean Squared Displacement (MSD) of the system and diffusion coefficients. For structural properties, we can calculate the radial distribution function of the system.
3. Parallelize the above mentioned functions with CUDA.

Demo

Show the speedup graph for both the running time of the system and the running time for the additional functionalities.

Plaform choice

GPUs, which CUDA utilizes, are designed for parallel processing and consist of thousands of cores. This architecture is highly conducive to the parallel nature of MD simulations, where calculations for each particle or group of particles can be performed concurrently (A lot of MD system contains at least thousands of particles).

Schedule

Time	Task
11/13 - 11/19	Implement velocity verlet algorithm (including velocity functions, position function and force function)
11/13 - 11/26	Implement the entire sequential version of the MD code
11/27 - 12/3	Implement CUDA version of the code
12/4 - 12/10	Optimize the code
12/10 - 12/14	Optimize extra functionalities (e,g, radial distribution function), finish the final report