Multi-GPU Fluid Simulation with Smooth Particle Hydrodynamics

Progress Report

Donghao Ren, Junsheng Guo

What exactly are the goals of your project?

The goal is to implement a SPH fluid simulation on multiple GPUs. We are going to distribute the particles by vertical slices (like the first homework assignment, but in 3D). The slices should be adaptive, to maintain that each processor has roughly the same number of particles. The processors should communicate with each other to exchange particles that moves across the boundaries. Also, to correctly compute the forces, a region of ghost particles should be communicated. As said before, the SPH program has a simulation part and also a rendering part, we are only going to make the simulation part on multiple GPU, the recursive ray-tracing rendering is too complicated for 3 weeks… However, we probably will create a simpler renderer (without recursive computations) on multiple GPUs, if time permits.

What data are you using to validate your project?

There is no dataset for this problem. The simulation only need a initial configuration. We are going to use the dam-break example, in which a rectangular container is half-filled by water initially.

What experiments are you running? How will you measure and report the efficiency and scaling of your implementation?

Experiments will be the dam-break configuration described above. We are going to run it on different particle counts, and different number of GPUs, we measure the running time as a function of the number of particles and the number of GPUs used. Since the simulation can go forever if we desire, we will have a small run with few frames and long run that simulate the water until it reaches some kind of stable state.

What machine are you running on? How many processors/nodes/cores?

We've got an account on Lonestar, and made the program runnable on both Triton and Lonestar. Since the two machines differ in hardware and system, we may not compare results across them. In Triton, each GPU node has a 4 GTX680s, 32GB memory and 12 CPU cores.

We observed each GPU can run up to one million particles individually before it start to complain about unspecified launch failures or bad allocs. So we decided that each processor should run one million particles. Since we have access to 3 GPU nodes, each has 4 GPUs, we can run 12 million particles on them. This might be improved by reducing memory consumption, but not easy.

What language are you writing in?

C/C++, CUDA

What have you gotten done so far?

As mentioned in the proposal, we had a single GPU implementation from last quarter's CS280 course, we did a lot of structural modifications to refine the architecture of the code, some critical bugs that cause wrong results and system failures were fixed, and we also rewrote the entire rendering part, to make the resulting image look better. We are working towards using MPI to do inter-process communication, we've successfully compiled and run the program with MPI enabled on both machines, and begun to change the program so that different processors can handle different parts of the fluid.

In our one million particle experiments, each simulation frame runs in around 350ms, and each frame of rendering is also around 350ms. We are going to focus on multi-GPUs for the rest of the time, we will compare the running time for multi-GPUs with single GPU (with fewer particles, because one GPU cannot hold more than one million particles).

Please find our video demo here (one GPU, one million particles, 5000 frames):
https://donghaoren.org/projects/fluidsph-1m5000.mp4

What obstacles have you discovered?

The main obstacle is how to successfully communicate particles between processors. We also want to adaptively adjust the boundaries for each node, so a fast algorithm for that is needed.