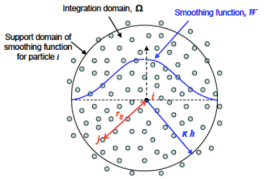
A Parallel GPU Implementation of Smoothed Particle Hydrodynamics

Qiuchen Guo, Mechanical Engineering
Lun Jiang, Civil & Environmental Engineering

Introduction to SPH



- Some particle properties are determined by an averaging over neighboring particles

$$A_s(r) = \sum_b m_b \frac{A_b}{\rho_b} W(r - r_b, h)$$

Consider weakly compressible flow:

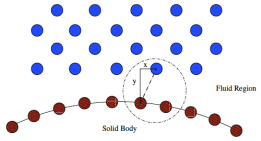
$$\rho(r) = \sum_b m_b W(r - r_b, h)$$

$$\frac{dv_i}{dt} = a_i^{pressure} + a_i^{viscosity} + a_i^{boundary} + a_i^{gravity}$$

Fluid-fluid interaction

Fluid – boundary interaction
Fluid – body interaction

Fluid-Boundary Particle Interaction



$$f_{k\alpha} = nR(y)P(x)$$

- N is unit normal vector of boundary surface
- x is tangential distance
- y is normal distance
- f_i is force on solid particle *i* by water particle *a*

Move the body:

$$M\frac{dV}{dt} = \sum_k m_k f_k$$

$$I\frac{d\Omega}{dt} = \sum_k m_k (r_k - R_0) \times f_k$$

Total force on solid particle k is:

$$f_k = \sum_{a \in WPS} f_{ka}$$

According to Newton's law,

$$m_k f_k = -m_a f_a$$

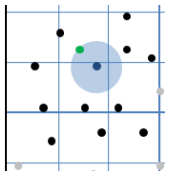
Problem Statement – Weakly compressible flow

- $\frac{d\rho_a}{dt} = \sum_b m_b \nabla_a W(r_a - r_b, 2h)$ — Calculate fluid density
- $P = B\left[\left(\frac{\rho}{\rho_0}\right)^{\gamma} - 1\right]$ — Calculate pressure
- $\frac{DV}{Dt} = -\frac{1}{\rho}\nabla P + g + f_a$ — Calculate fluid acceleration(velocity, position)
- $M\frac{dV}{dt} = \sum_k m_k f_k$ — Move solid body

Implementation of Serial Code

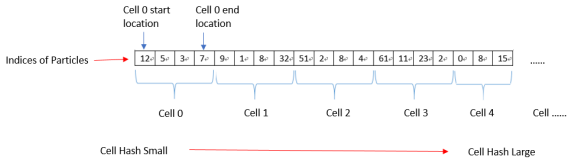
```
Initialization of fluid, boundary, body particles

double simulation_time = read_timer( );
  for(int step=0; step < NSTEPS; step++)
  {
    Build_table();// Build Hash Table that store cell number for
each particle
    Compute_density_pressure_kernel();
    Compute_force();//Compute Pressure, boundary force
    Advection();//Advance in time for fluid particles
    Move_body();//Calculate body motion

  }
```

- Create a linked list for each cell, link all particles within its cell.
- Build a hash table to store the head pointers of all linked lists.
- Calculate the cell hash value (index in hash table) based on cell location.



Implementation of GPU Code

```
Thrust::sort_by_key(key_begin, key_end, value);
```



- Build an index array to store all indices of particles.
- Calculate the cell hash value (as keys to sort later) based on cell location.
- Sort the indices array using "thrust::sort_by_key ()" by cell hash value.
- Record the starting and ending locations in the index array, within which all particles belong to one cell.
- Each thread handle one particle.

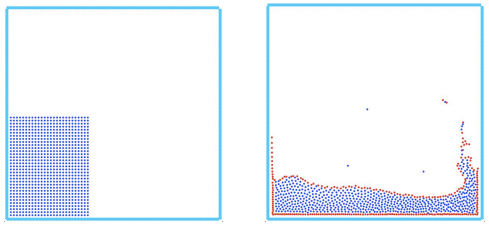
Visualization Using OpenGL

```
# include <GL/gl.h>
# include <GL/glu.h>
# include <GL/glut.h>
```

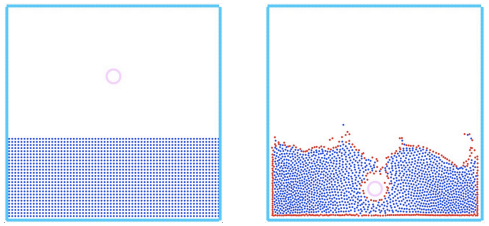
- Inner Fluid Particle (Blue)
- Surface Fluid Particle (Red)
- Boundary Particle (Cyan)
- Solid Body Particle (Pink)

Results

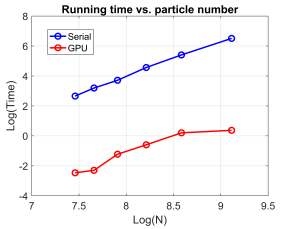
Case 1: Dam Break (fluid-structure interaction)



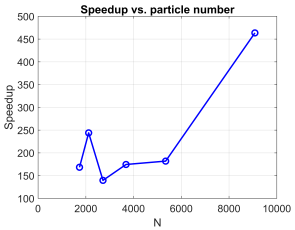
Case 2: Water entry of a ball (fluid-structure interaction)



Scale of Serial SPH code



SpeedUp of CUDA GPU code



Summary and On-going Work

- Implemented a serial SPH code in C++, with performance order O(N).
- Implemented a parallel SPH code in CUDA with performance order O(N).
- Use global memory of GPU in improve the performance of CUDA code.
- Get the speed-up of over 450 times.

- Use block shared memory to store particle in one cell and its neighbors to further optimize the performance of CUDA code.