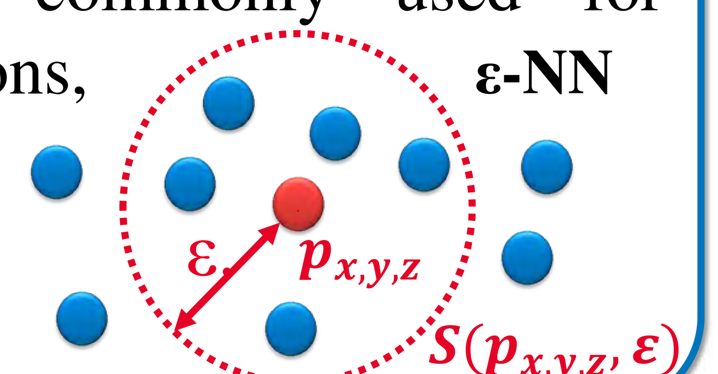
Out-of-Core Proximity Computation on GPU for Particle-based Fluid Simulations

Duksu Kim[1] Myung-Bae Son[2] Young J. Kim[3] Jeong-Mo Hong[4] Sung-Eui Yoon[2]

[1]KISTI (Korea Institute of Science and Technology Information) [2]KAIST (Korea Advanced Institute of Science and Technology) [3]Ewha Woman's University, Korea [4]Dongguk University, Korea

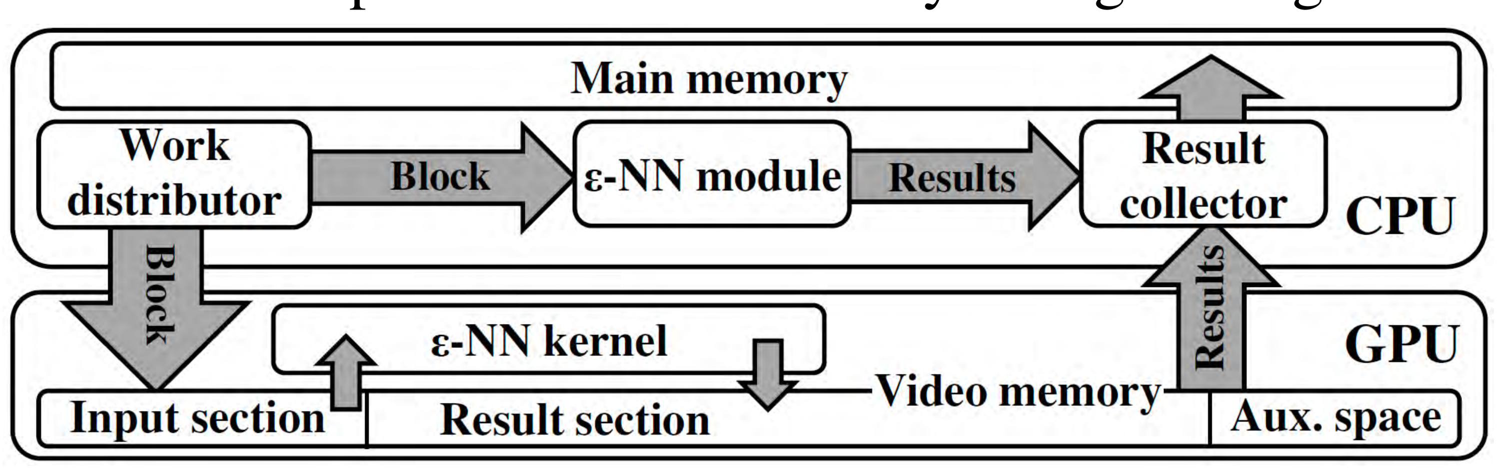
Introduction

Thanks to ever growing demands for higher realism and the advances of particle-based fluid simulation techniques, large scale simulations are getting increasingly popular across different graphics applications. To meet the demand of higher realism, a high number of particles are used for particle-based fluid simulations, resulting in various out-of-core issues. In this work, we present an out-of-core proximity computation, especially, epsilon-Nearest Neighbor (ε-NN) search, commonly used for particle-based fluid simulations, to handle such big data sets consisting of tens of millions of particles.



System Overview

- **Goal**: Efficiently find and store the neighborhood information for massive amount of particles that cannot be held at once by a GPU memory
  - Assumption: the CPU memory is large enough



- Use a uniform grid while determining cell indexes with Z-curve to exploit spatial locality
  - Commonly used in particle-based fluid simulations
- **Work distributor (CPU side)**
  - Divides the uniform grid into sub-grids (i.e. block) dynamically and assign them to available GPUs
- **ε-NN kernel/module (both GPU and CPU sides)**
  - Performs ε-NN for particles in the block
- **Result collector (CPU side)**
  - Takes results from GPUs and CPUs

Chicken-and-Egg Problem

- To fully utilize high performance GPU in an out-of-core manner, we need to divide the grid such that the size of the working set of each block should be smaller than the size of GPU memory
- Unfortunately, we cannot know the exact required memory size since we do not know the number of neighbors until we actually perform the query

$$s(B) = n_B s_p + S_n \sum_{p_i \in B} n_{p_i}$$

Unknown! # of neighbors for the particle p_i

- $s(B)$: required memory size for processing a block B
- n_B: the number of particles in the block
- s_p, s_n: the data sizes of storing a particle and a neighbor particles, respectively

Expect it!

Our Approach

- Assuming the local uniform distribution
  - Particles distributions tend to show local uniformity around each cell in particles-based fluid simulations
- Then, the number of neighbors is proportional to the overlap volume between the search sphere and cells weighted by the number of their associated particles

$$E(p_{x,y,z}) = \sum_i n(C_i) * \frac{Overlap(S(p_{x,y,z}, \varepsilon)), C_i)}{V(C_i)}$$

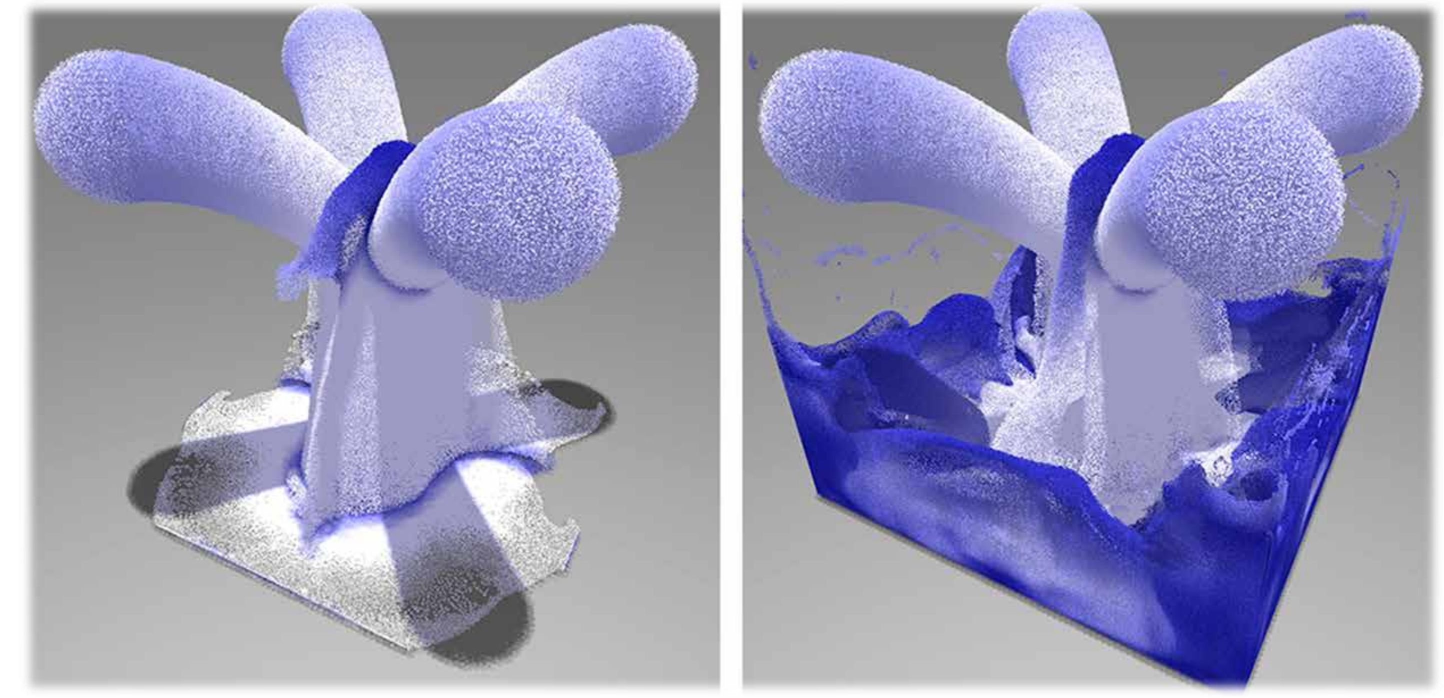
- $E(p_{x,y,z})$: expected number of neighbors of the particle
- C_i : cells having overlap region with the search sphere $S(p_{x,y,z}, \varepsilon)$
- $n(C_i)$: the number of particles in the cell
- $Overlap(\cdot,\cdot)$: overlap volume
- $p_{x,y,z}$: particle p located at (x, y, z)
- $V(C_i)$: volume of the cell

- To avoid high computational overhead we compute the average, expected number of neighbors of particles in a cell, and use it for all particles in the cell

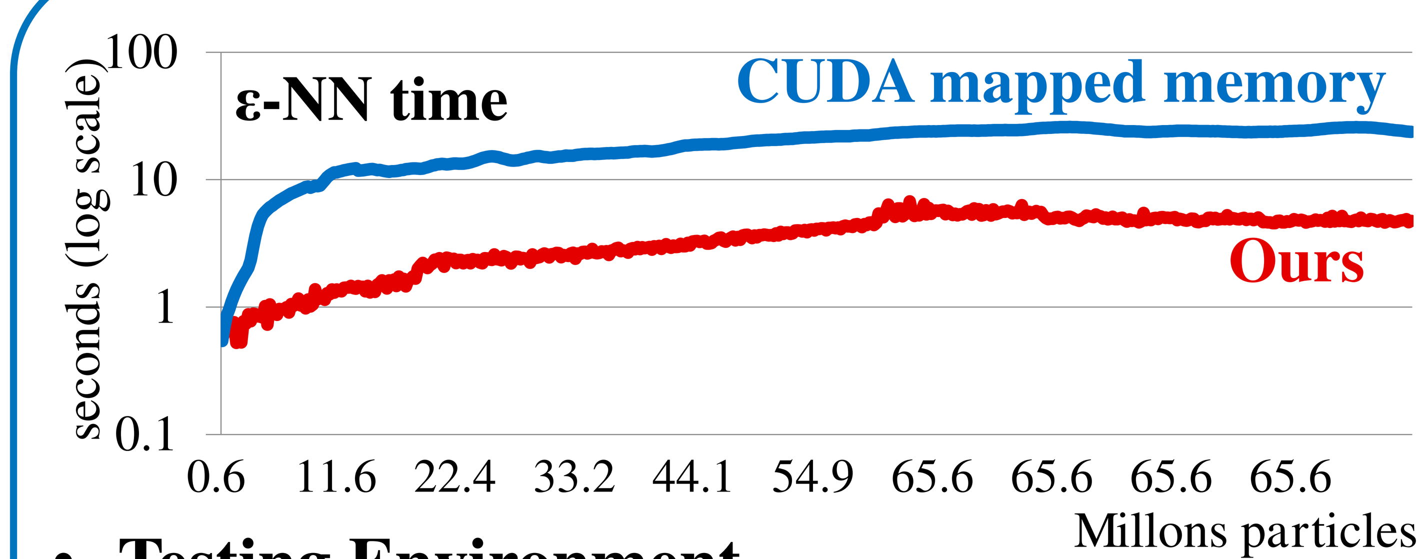
$$E(C_q) = \frac{1}{V(C_q)} * \int_0^l \int_0^l \int_0^l E(p_{u,v,w}) du\, dv\, dw$$

/* Please see the paper for more details */

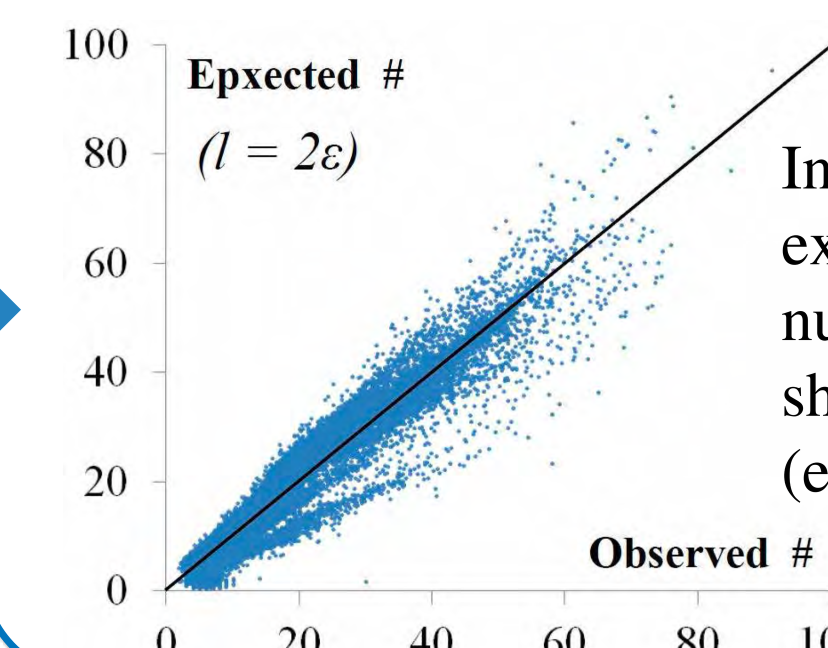
- Based on the expected # of neighbors, we find a maximal work unit (block) the GPU can handle at once with our hierarchical work distribution method
  - To maximize GPU utilization efficiency



Results



- **Testing Environment**
  - Two hexa-core CPUs / 192GB main memory
  - One GPU (GTX 780, **3GB**)
- **Benchmarks**
  - Consisting of up to **65 M particles**
  - Up to **16 GB** memory space is required
- Achieve up to **26 X** higher performance over using the mapped memory technique of CUDA
- Show up to **51 X** higher performance with twelve CPU cores and one GTX 780 over using a single CPU core
  - 6.3X compared with using only twelve CPU cores



In our tested benchmarks, expected and observed number of neighbors show high correlations (e.g., 0.97)

* Parts of this work was presented at HPG 2014

* **Project homepage**: http://sglab.kaist.ac.kr/OOCNNS