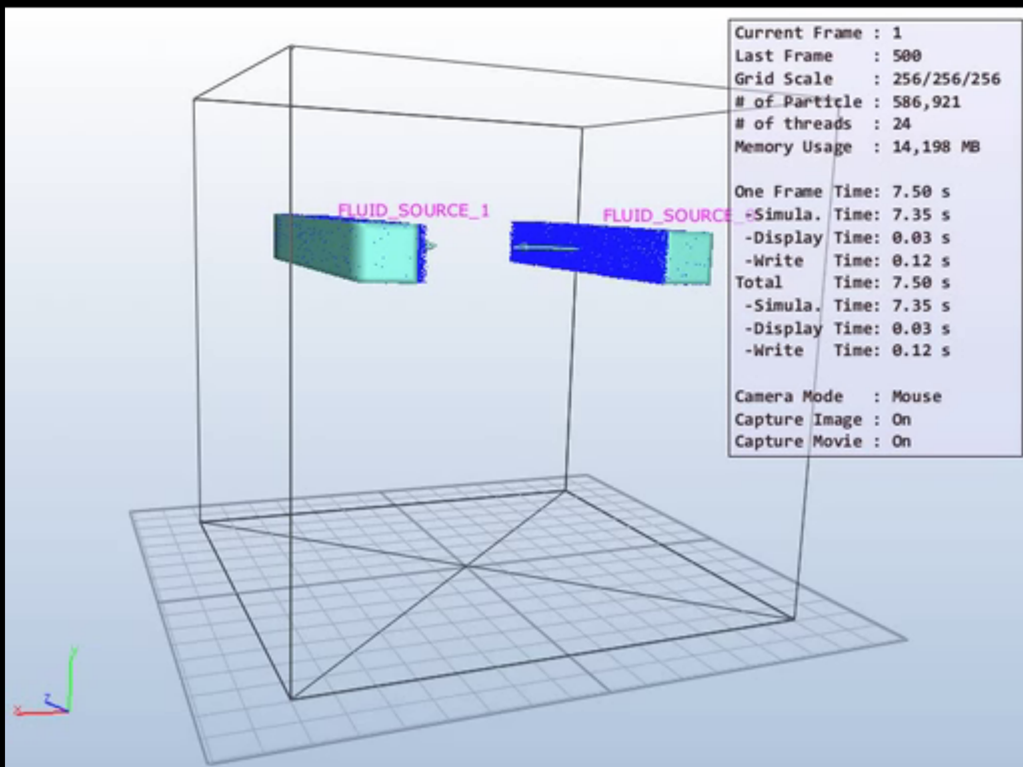
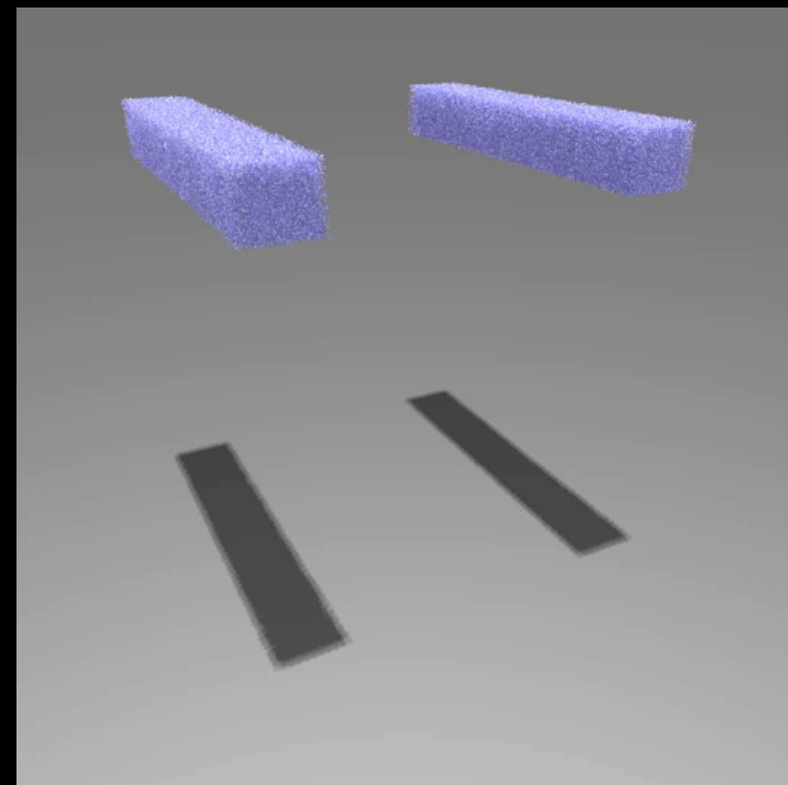
Out-of-Core Proximity Computation for Particle-based Fluid Simulation

HPG 2014

Presenter:
Duksu Kim   Myung-Bae Son
Young J. Kim   Jeong-Mo Hong   Sung-Eui Yoon

Particle-based Fluid Simulation





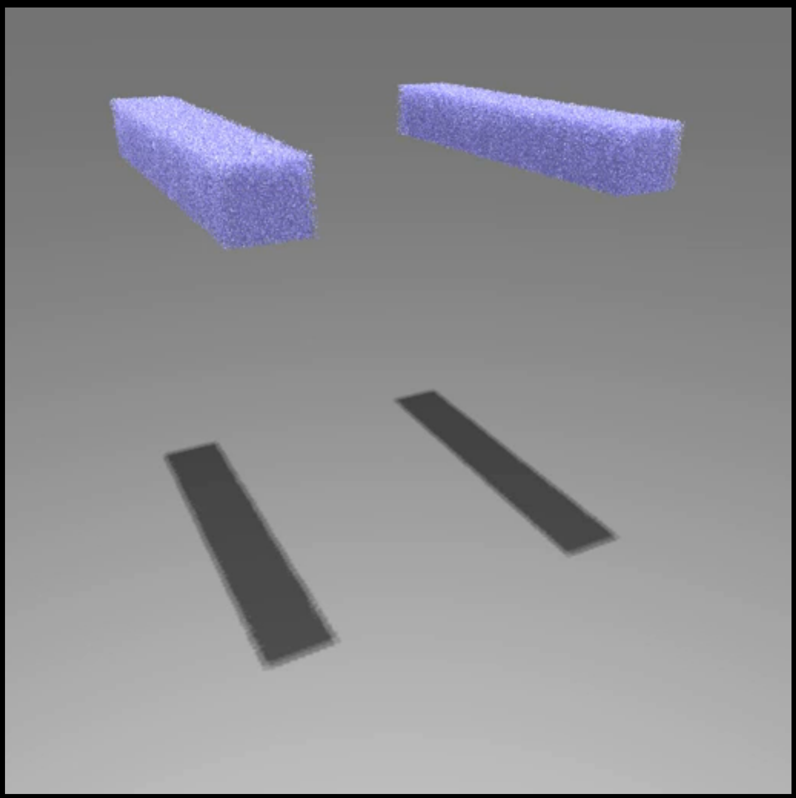
Motivation

- **To meet the higher realism, a large number of particles are required**
  - Tens of millions particles
- **In-core algorithm (previous work)**
  - Manage all data in GPU's video memory
  - Can handle up to **5 M** particles with **1 GB** memory for particle-based fluid simulation
- **Recent commodity GPUs have 1 ~ 3 GB memories (up to 12 GB)**

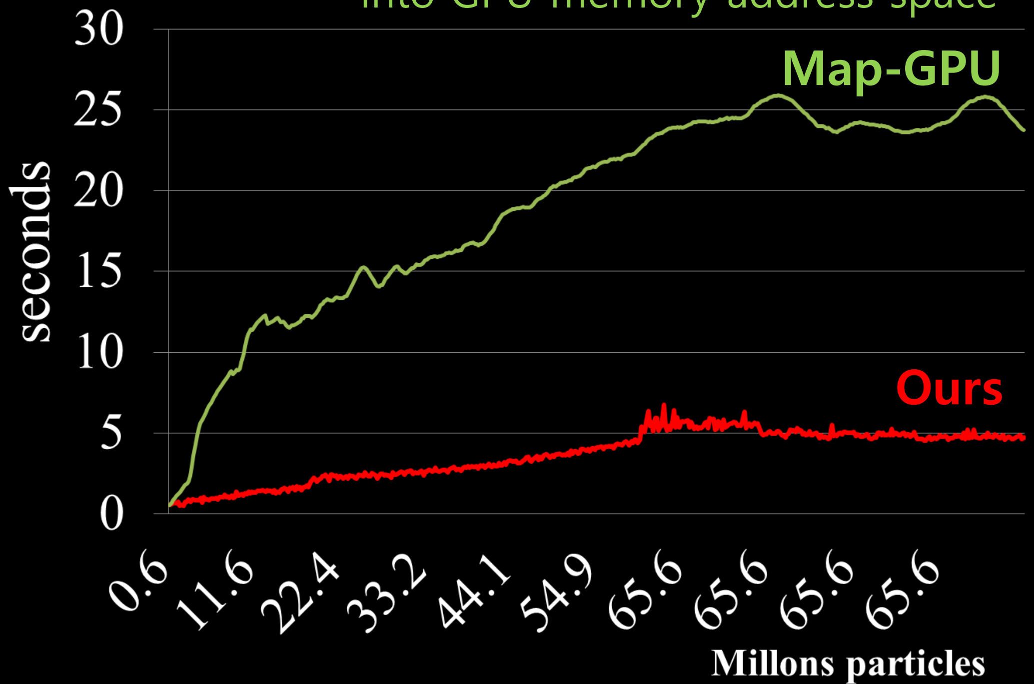
Contributions

- Propose out-of-core methods that utilize heterogeneous computing resources and process neighbor search for a large number of particles
- Propose a memory footprint estimation method to identify a maximal work unit for efficient out-of-core processing

Result



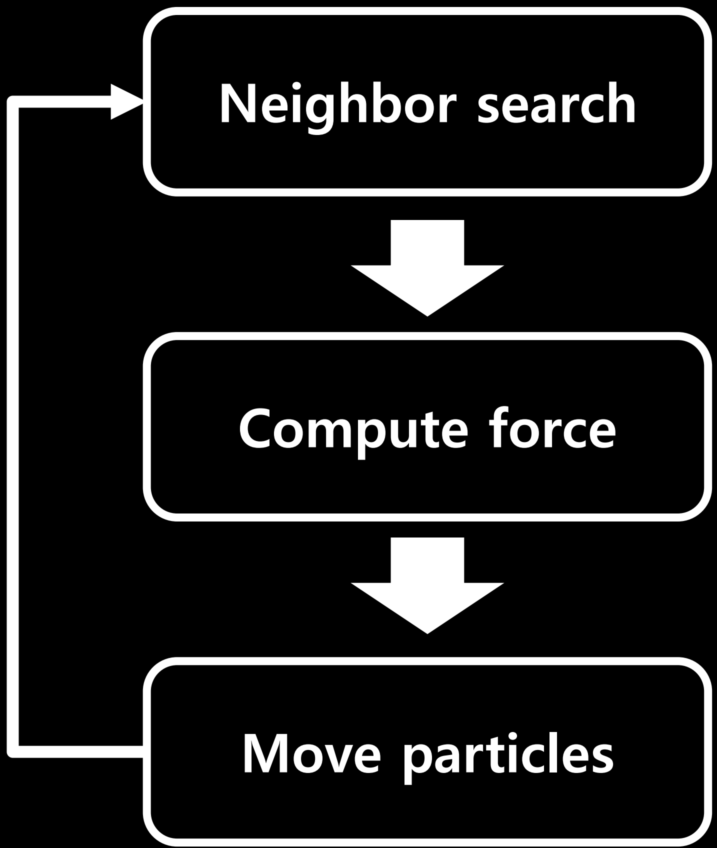
NVIDIA mapped memory Tech.
- Map CPU memory space
  into GPU memory address space



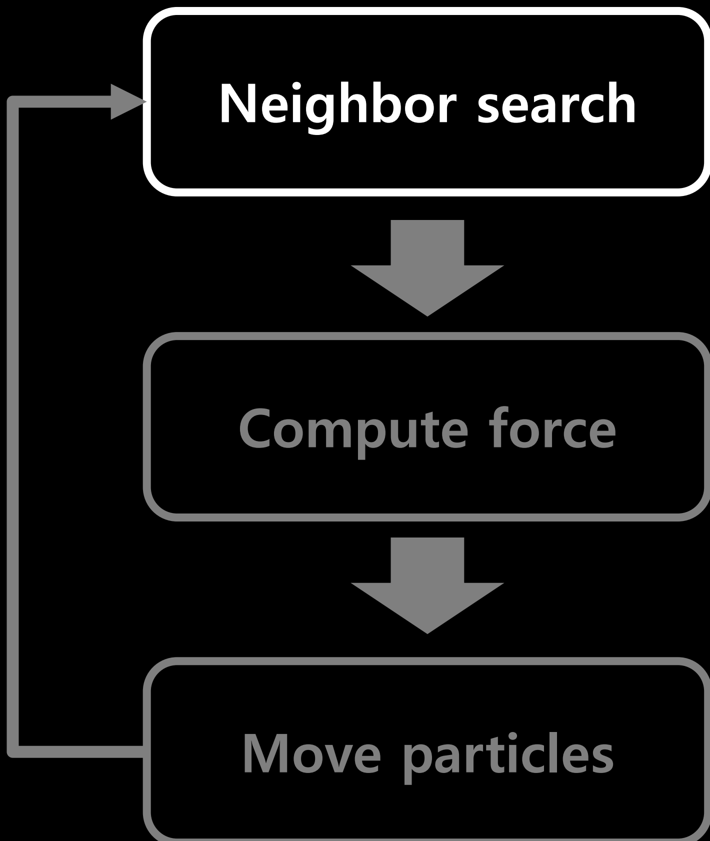
Up to 65.6 M Particles
Maximum data size: 13 GB

- **Two hexa-core CPUs (192 GB Mem.)**
- **One GPU (3 GB Mem.)**

Particle-based Fluid Simulation

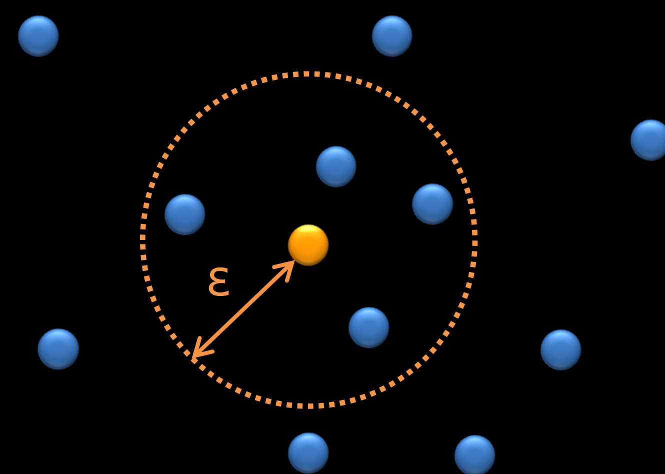


Particle-based Fluid Simulation



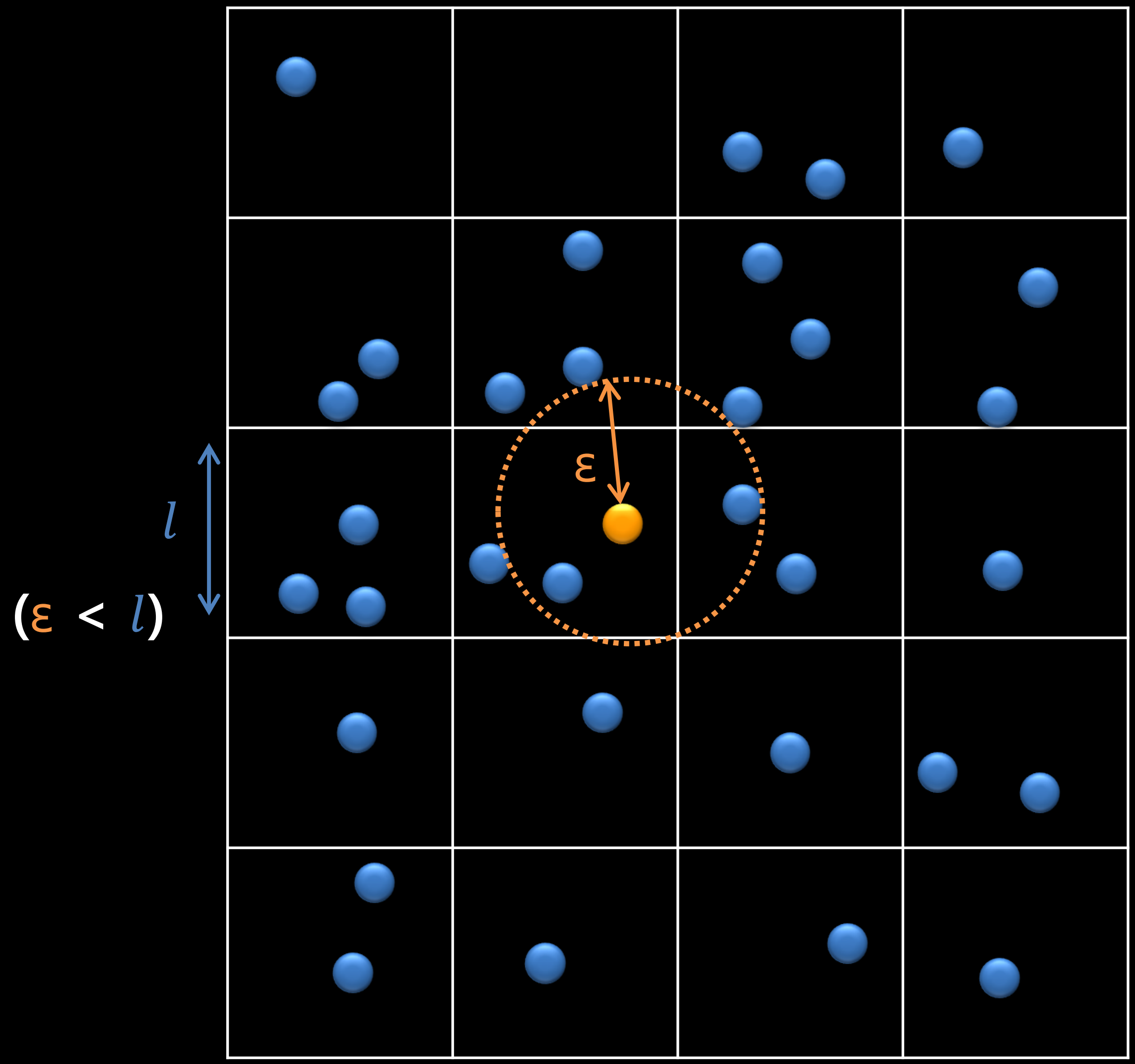
Performance bottleneck

- Takes 60~80% of simulation computation time

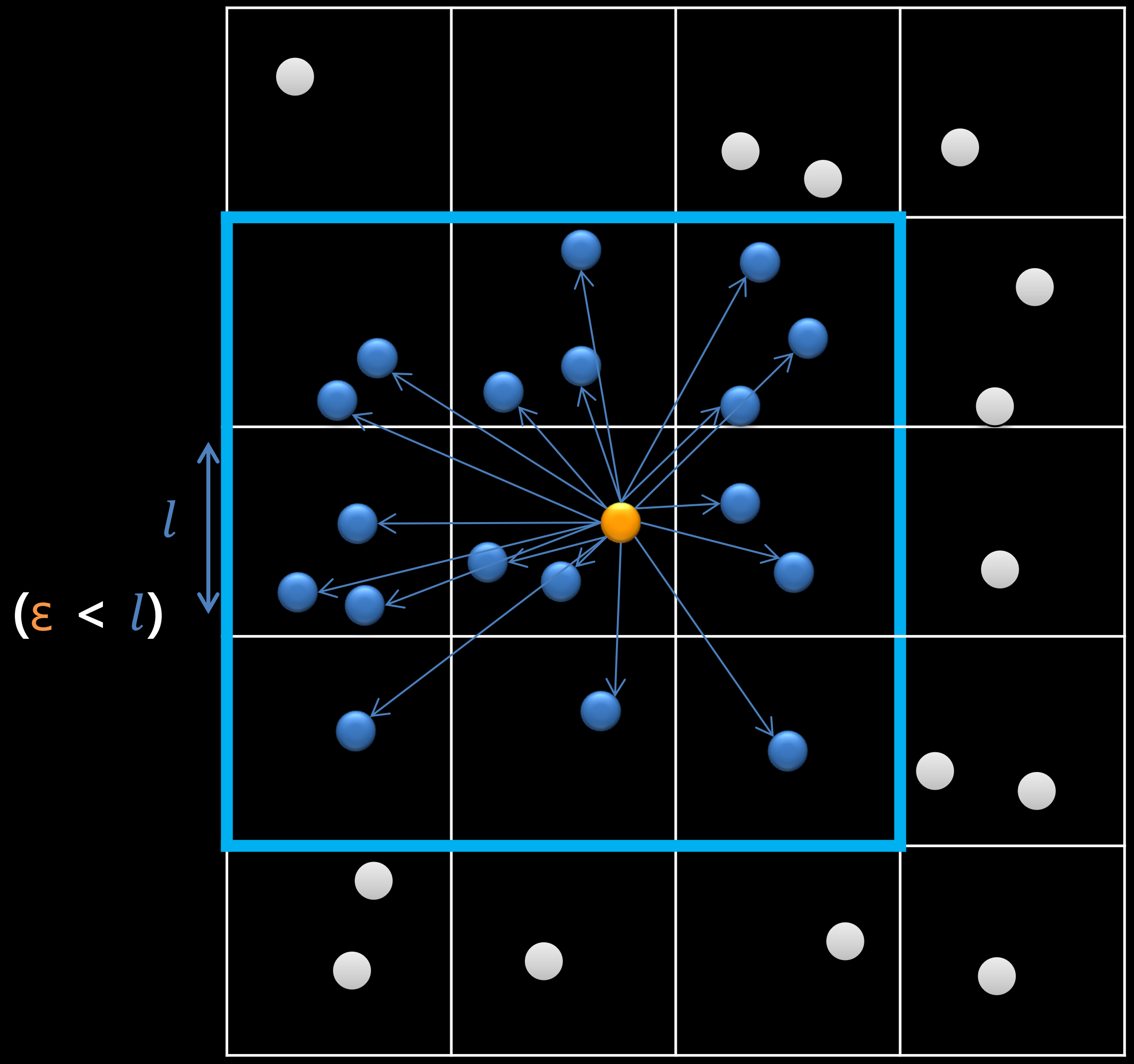


ε-Nearest Neighbor (ε-NN)

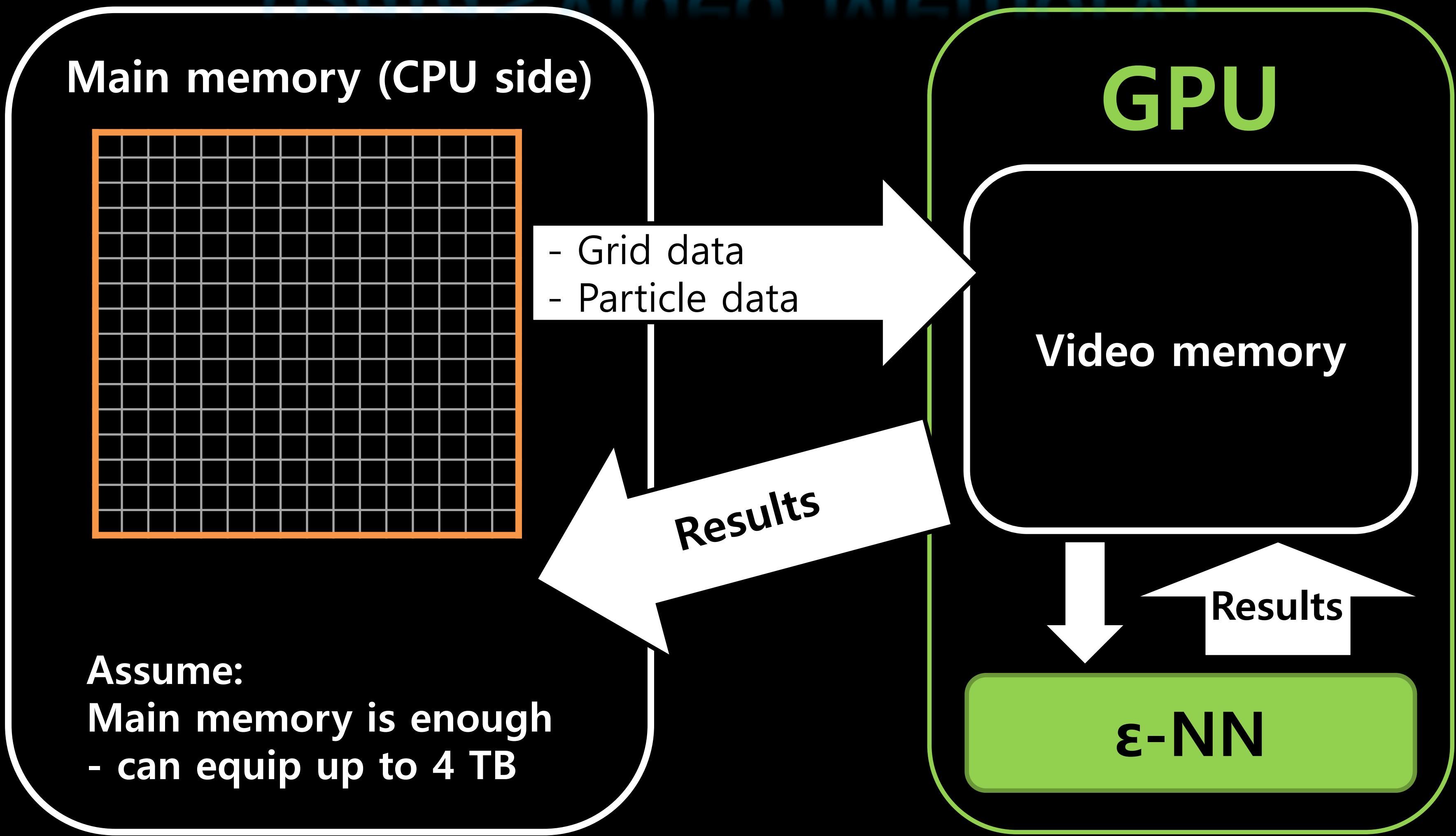
Preliminary: Grid-based ε-NN



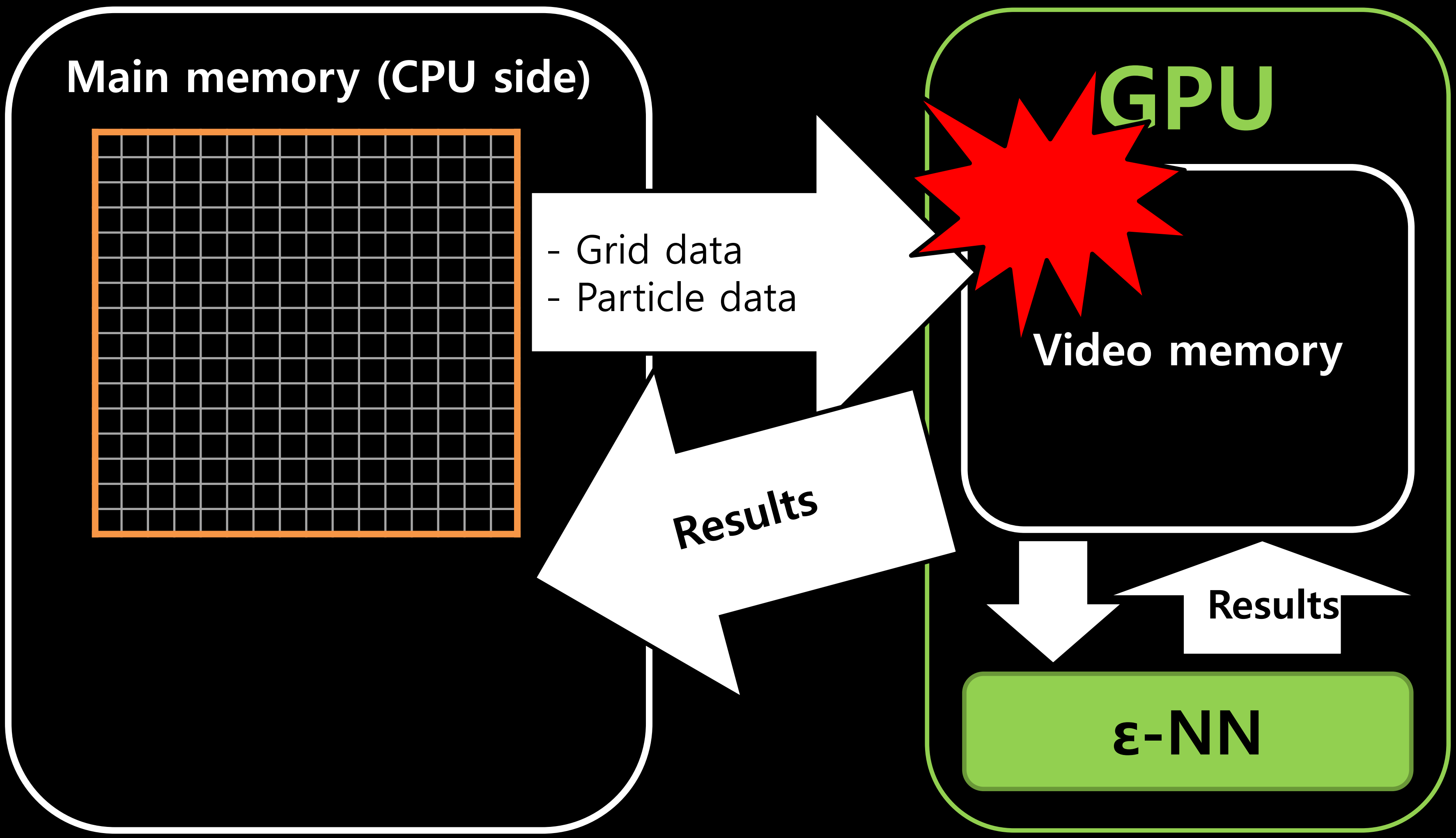
Preliminary: Grid-based ε-NN



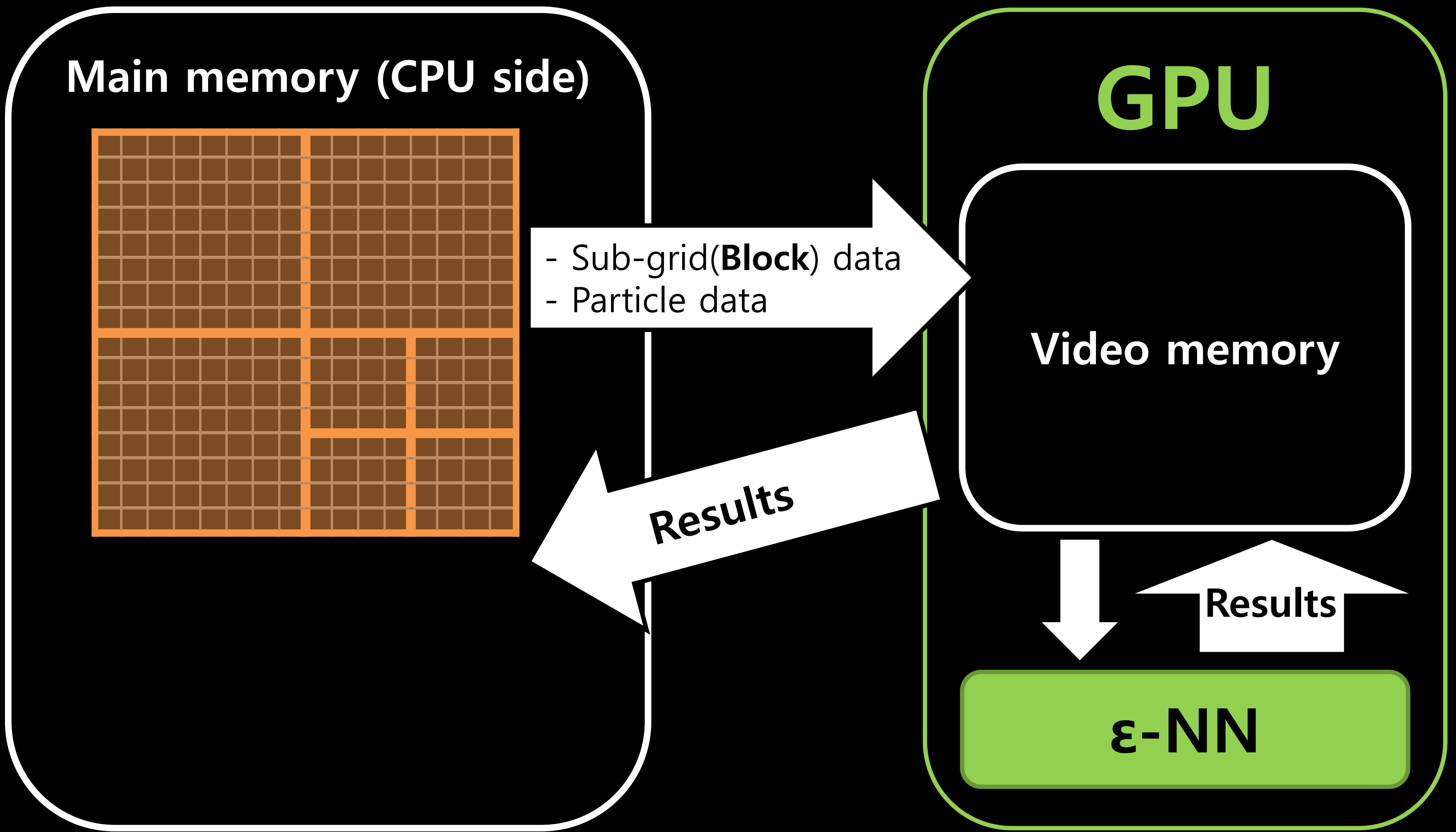
In-Core Algorithm (Data<Video Memory)



Data > Video Memory

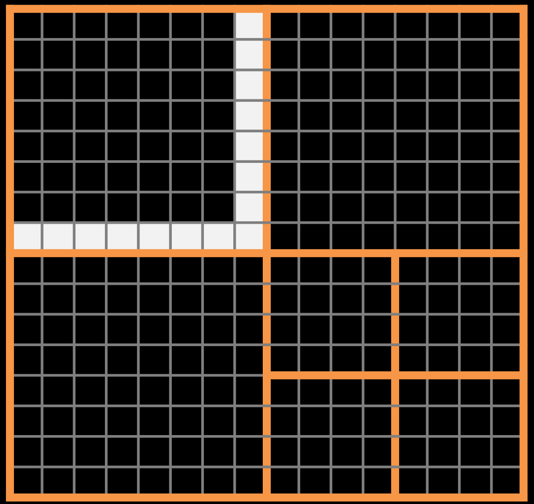


Out-of-Core Algorithm



Boundary Region

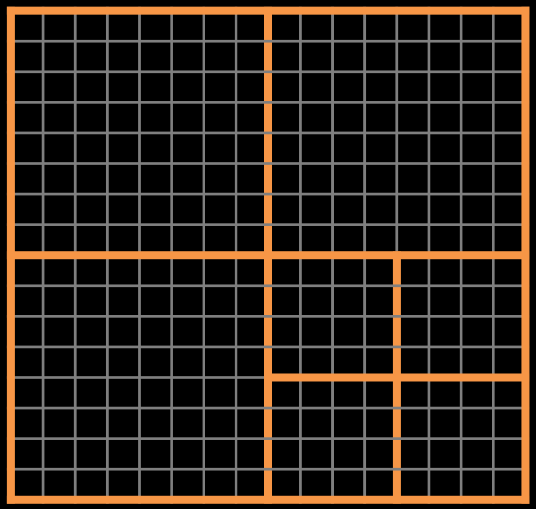
- Required data in adjacent blocks
- Inefficient to handle in an out-of-core manner



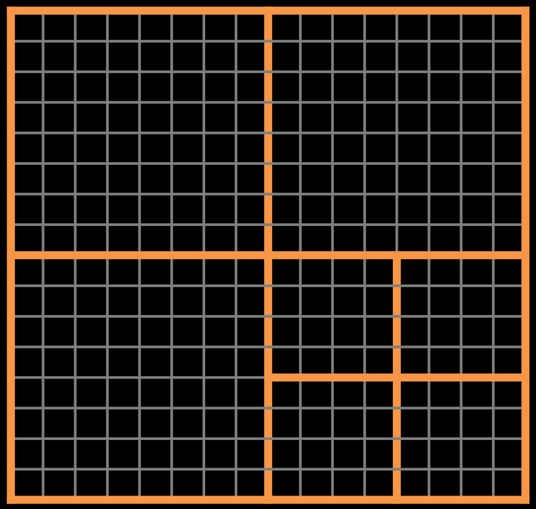
Boundary Region

- **Required data in adjacent blocks**
- **Inefficient to handle in an out-of-core manner**
- **Multi-core CPUs handle the boundary region**
  - CPU (main) memory contain all required data
  - Ratio of boundary regions is usually much smaller than inner regions

How to Divide the Grid ?



How to Divide the Grid ?



- **Goal: Find the largest block that fits to the GPU memory**
  - Improve parallel computing efficiency
    - Process a large number of particles at once
    - Minimize data transfer overhead
  - Reduce the boundary region
    - As the ratio of boundary region is increased, the workload of CPU is increased

Required Memory Size for processing a block, B

of particles in B

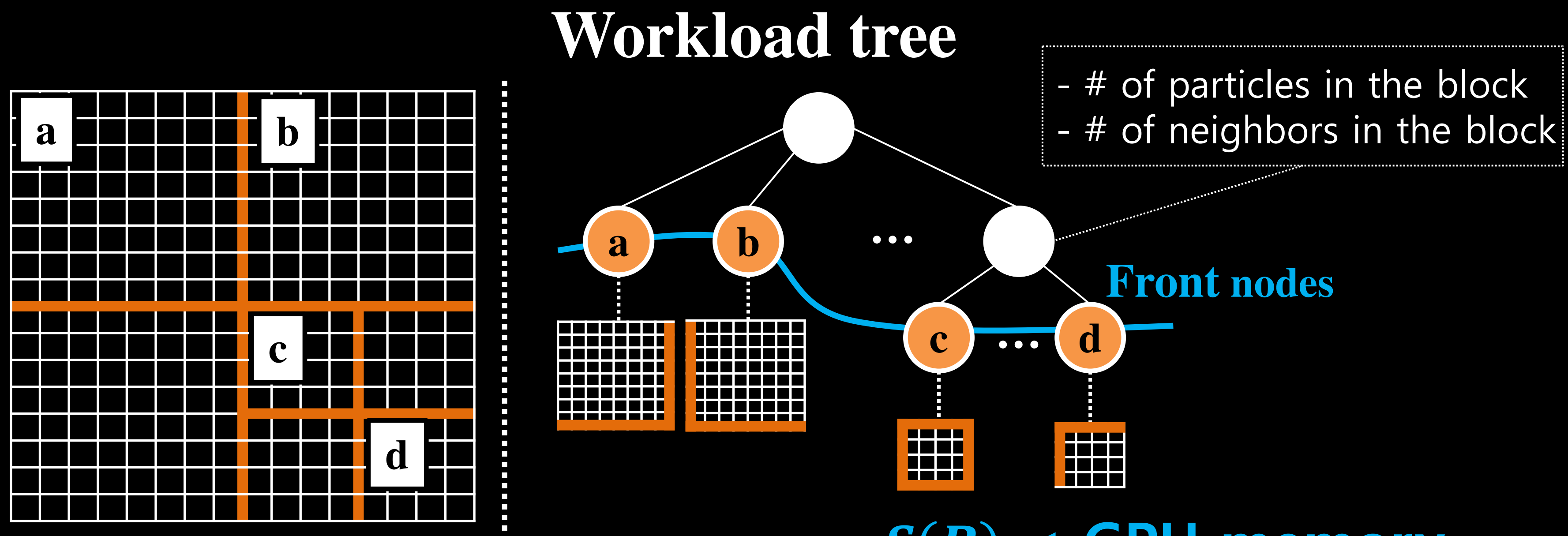
# of neighbor particles for the particle i (p_i)

$$S(B) = n_B S_p + S_n \sum_{p_i \in B} n_{p_i}$$

Data size for storing a particle

Data size for storing a neighbor info.

Hierarchical Work Distribution



$S(B)$ < GPU memory

Chicken-and-Egg Problem

of particles in B

# of neighbor particles for the particle *i*, p_i

$$S(B) = n_B S_p + S_n \sum_{p_i \in B} n_{p_i}$$

Data size
for storing a particle

Data size
for storing a neighbor info.

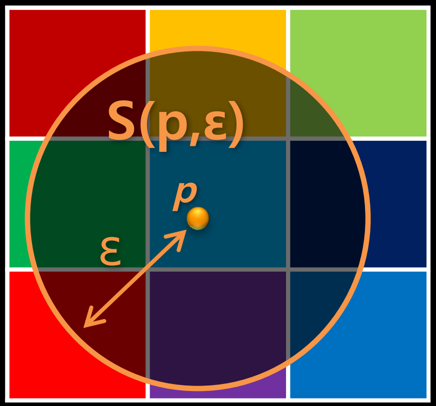
Chicken-and-Egg Problem

$$S(B) = n_B S_p + S_n \sum_{p_i \in B} n_{p_i}$$

Our approach:
Estimation the number of neighbors for particles

Problem Formulation

- **Assumption**
  - Particles are uniformly distributed in a cell
- **Idea**
  - For a particle, the number of neighbors in a cell is proportional to the overlap volume between the search sphere and the cell weighted by the number of particles in the cell



Expected Number of Neighbors of a particle p located at (x, y, z)

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

- C_i : cells of $p_{x,y,z}$ and its adjacency cells
- $n(C_i)$: the number of particles in the cell
- $Overlap(S(p_{x,y,z}, \varepsilon), C_i)$: overlap volume between them
- $V(C_i)$: volume of the cell

Problem Formulation

- **Compute $E(p_{x,y,z})$ for each particle takes high computational overhead**
- **Instead, (approximation)**
  - Compute the average $E(p_{x,y,z})$ for particles in a cell
  - Use the value for all particles in the cell

The Average, Expected Number of Neighbors of particles in a cell C_q

Expensive to compute at runtime

$$E(C_q) = \frac{1}{V(C_q)} * \int_0^l \int_0^l \int_0^l E(p_{x,y,z})\,dx\,dy\,dz$$

- l is the length of a cell along each dimension
- $p_{x,y,z}$ is a particle positioned at (x, y, z) on a local coordinate space in C_q

The Average, Expected Number of Neighbors of particles in a cell C_q

$$E(C_q) = \frac{1}{V(C_q)} * \int_0^l \int_0^l \int_0^l E(p_{x,y,z})\,dx\,dy\,dz$$

$$= \frac{1}{V(C_q)} * \sum_i n(C_i) * \frac{D(C_q, C_i)}{V(C_i)}$$

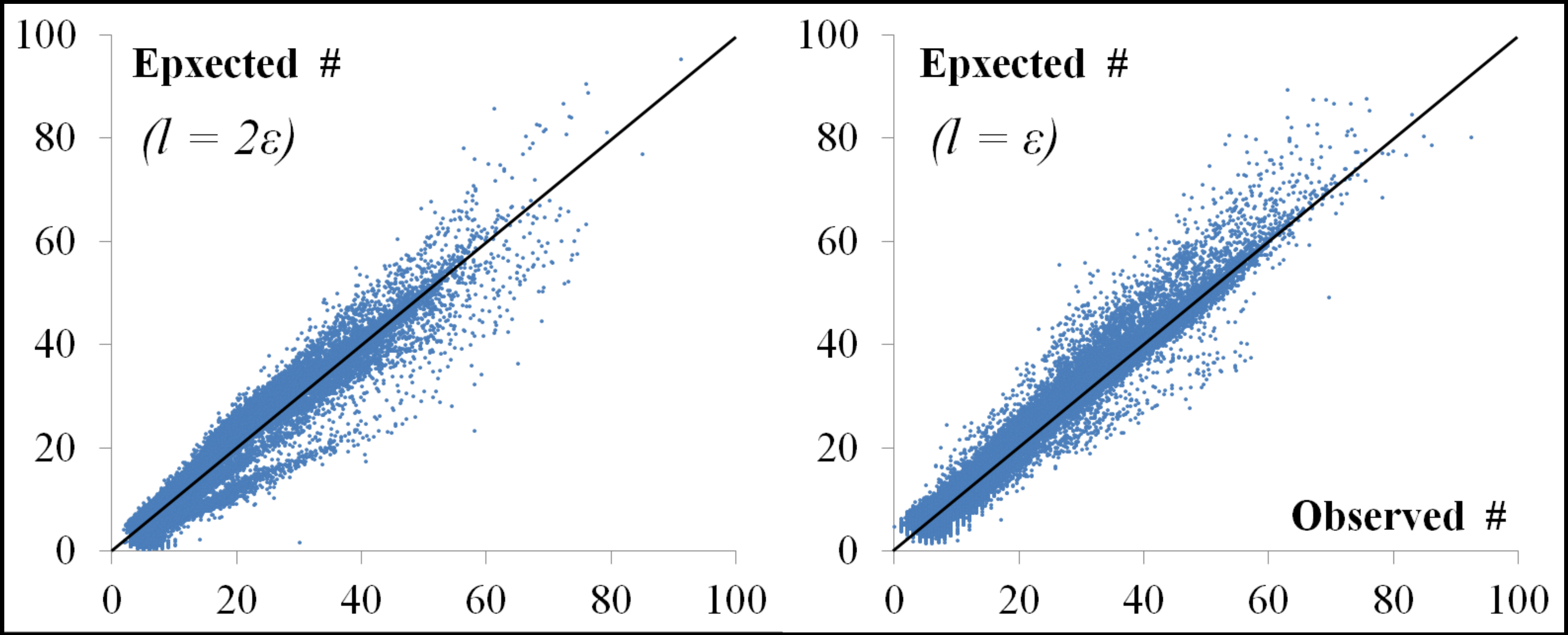
$$D(C_q, C_i) = \int_0^l \int_0^l \int_0^l Overlap(S(P_{x,y,z}, \varepsilon), C_i)\,dx\,dy\,dz$$

The Average, Expected Number of Neighbors of particles in a cell C_q

- **Pre-compute $D(C_q, C_i)$**
  - The value depends on the ratio between l and ε values
  - l and ε are not frequently changed by user
  - Use the Monte-Carlo method with many samples (e.g., 1 M)
- **Use look-up table at runtime**

$$D(C_q, C_i) = \int_0^l \int_0^l \int_0^l Overlap(S(P_{x,y,z}, \varepsilon), C_i)\,dx\,dy\,dz$$

Validation



- Correlation = 0.97
- Root Mean Square Error (RMSE) = 3.7

~~Chicken-and-Egg Problem~~

Expected number of neighbors

$$S(B) = n_B S_p + S_n \sum_{p_i \in B} n'_{p_i}$$

~~Chicken-and-Egg Problem~~

Expected number of neighbors

$$S(B) = n_B S_p + S_n \sum_{p_i \in B} n'_{p_i} + S_{Aux}$$

Auxiliary space to cover the estimation error

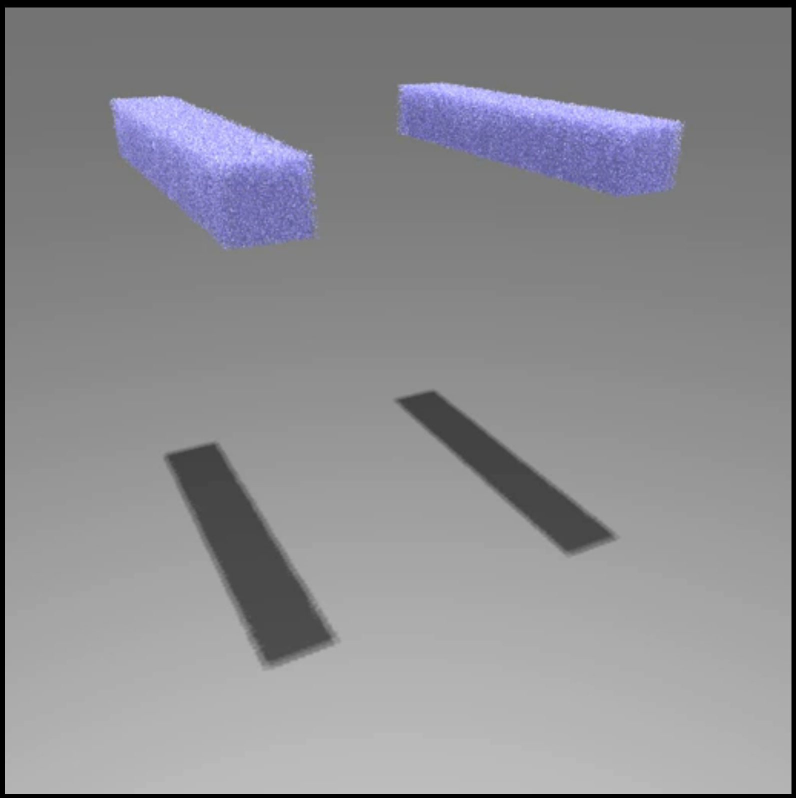
$$S_{Aux} = 3.7 * n_B S_n$$

RMSE

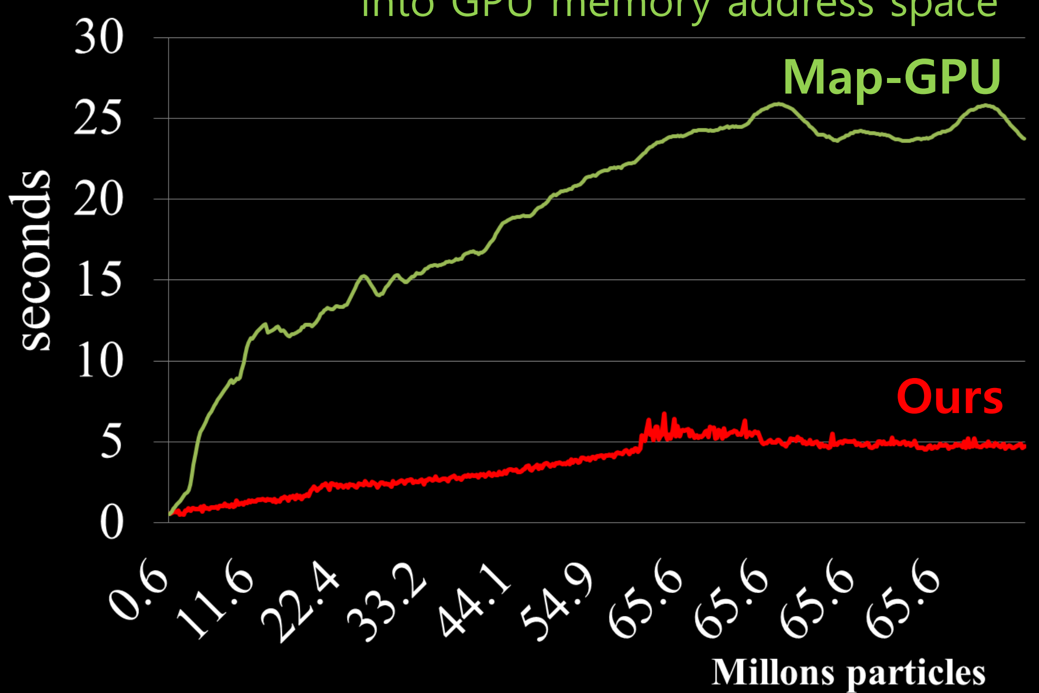
Results

- **Testing Environment**
  - Two hexa-core CPUs
  - 192 GB main memory (CPU side)
  - One GPU (GeForce GTX 780) with 3 GB video memory

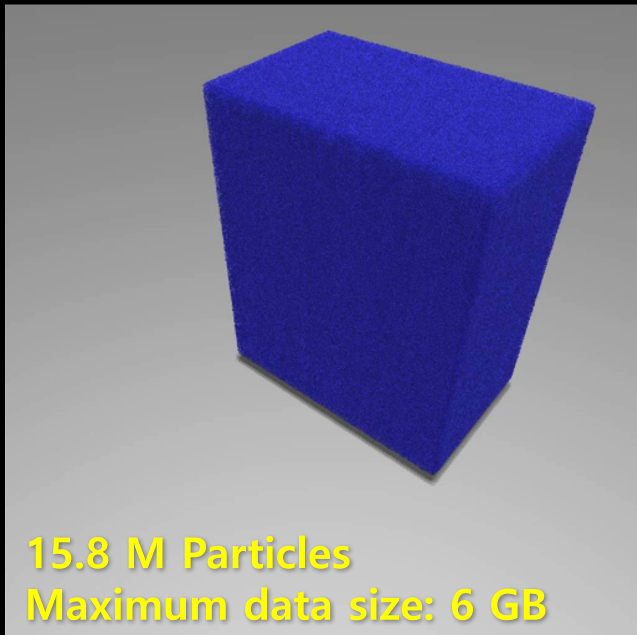
Results

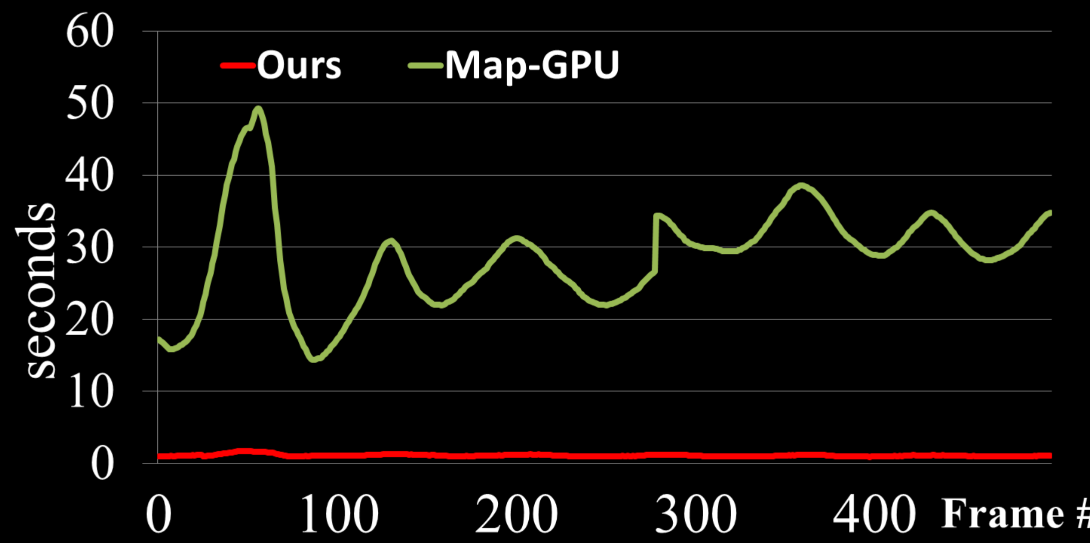


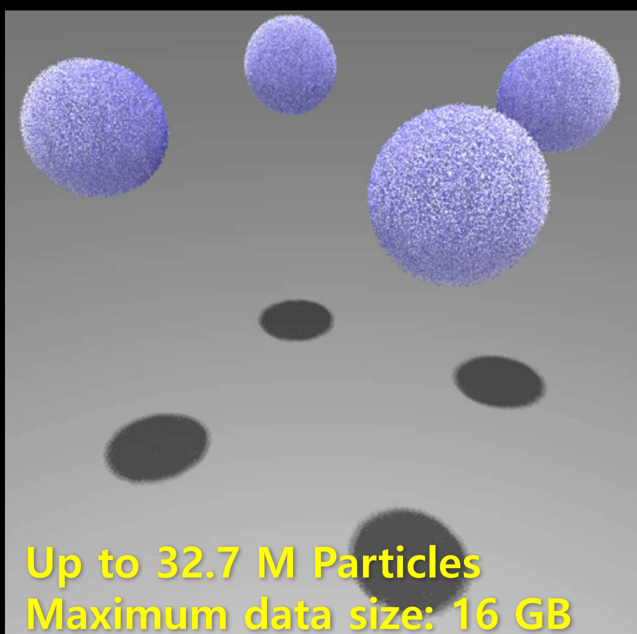
NVIDIA mapped memory Tech
- Map CPU memory space
  into GPU memory address space

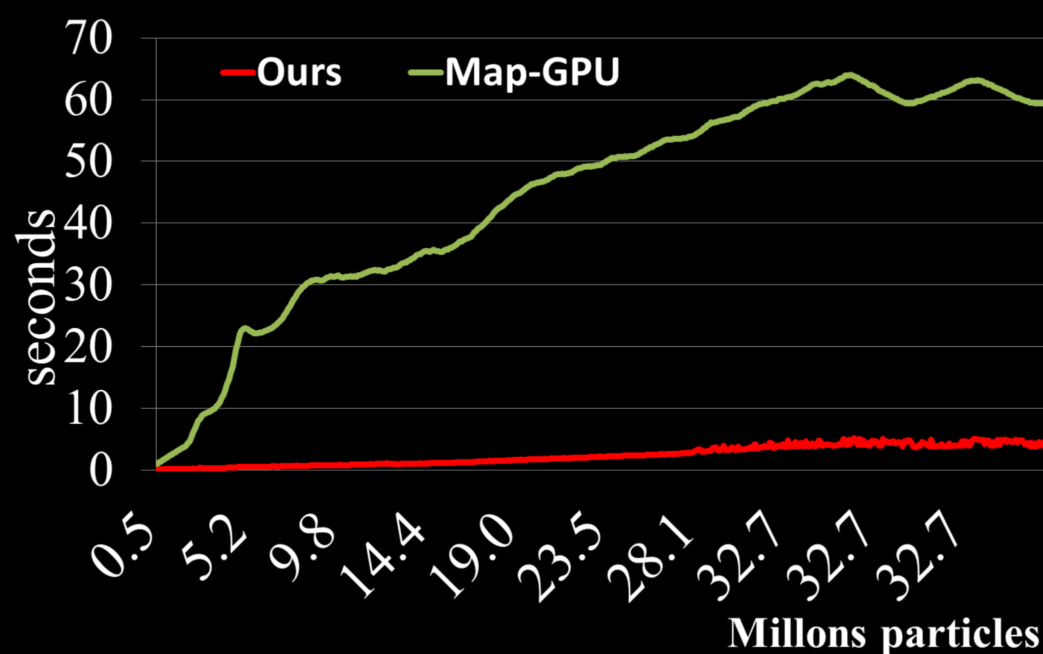


Up to 65.6 M Particles
Maximum data size: 13 GB

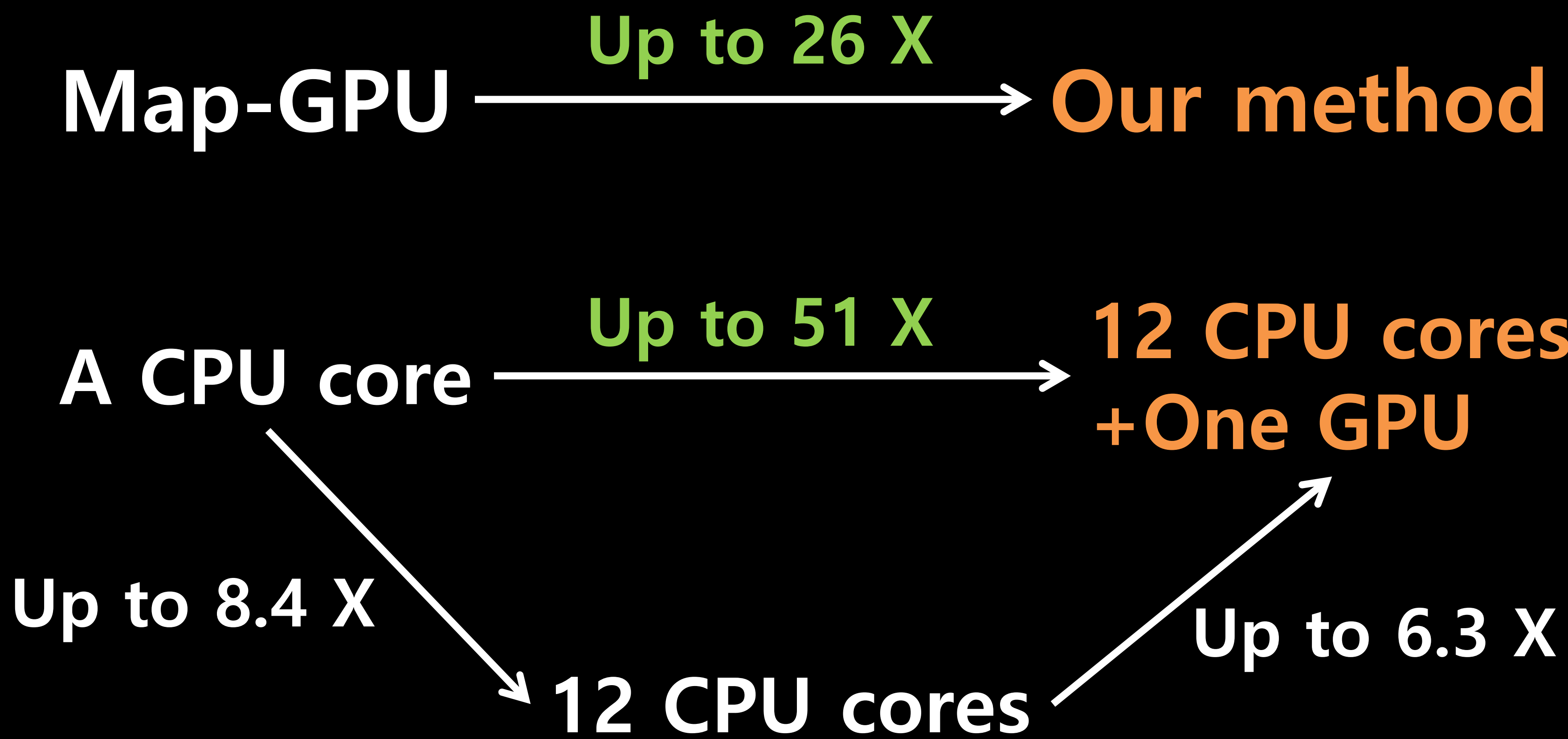

15.8 M Particles
Maximum data size: 6 GB


Ours
Map-GPU
seconds
Frame #
0
10
20
30
40
50
60
100
200
300
400


Up to 32.7 M Particles
Maximum data size: 16 GB


Ours
Map-GPU
seconds
0
10
20
30
40
50
60
70
0.5
5.2
9.8
14.4
19.0
23.5
28.1
32.7
32.7
32.7
Millons particles

Results


Map-GPU
Up to 26 X
Our method
A CPU core
Up to 51 X
12 CPU cores
+One GPU
Up to 8.4 X
12 CPU cores
Up to 6.3 X

Conclusion

- **Proposed an out-of-core ε-NN algorithm for particle-based fluid simulation**
  - Utilize heterogeneous computing resources
  - Utilize GPUs in out-of-core manner
  - Propose hierarchical work distribution method

Conclusion

- **Proposed an out-of-core ε-NN algorithm for particle-based fluid simulation**
- **Presented a novel, memory estimation method**
  - Based on expected number of neighbors

Conclusion

- Proposed an out-of-core ε-NN algorithm for particle-based fluid simulation
- Presented a novel, memory estimation method

- Handled a large number of particles
- Achieved much higher performance compared with a naïve OOC-GPU approach

Future Work

- Extend to support multi-GPUs
- Improve the parallelization efficiency by employing an optimization-based approach
- Extend to other applications

Thanks!

Any questions?

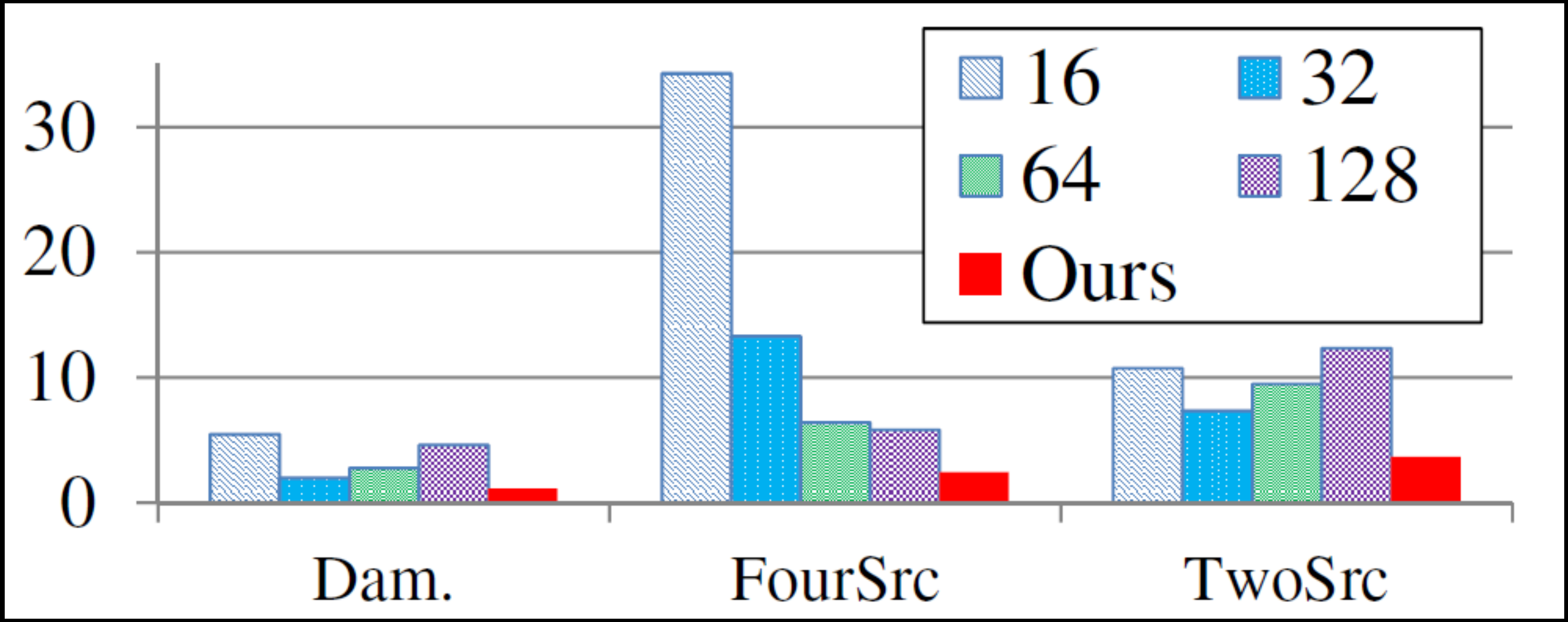
(bluekdct@gmail.com)

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

- Benchmark scenes are available in the homepage
- Source code will be available in the homepage

Benefits of Our Memory Estimation Model

- **Fixed space VS Ours**



Benefits of Hierarchical Workload Distribution

- **Larger block size shows a better performance**
  - E.g., using 32^3 and 64^3 block sizes takes 22% and 30% less processing time in GPU than using 16^3 blocks on average

Benefits of Hierarchical Workload Distribution

- But, the maximal block size varies depending on the benchmarks and region of the scene
- Compared manually set fixed block size based on our estimation model, hierarchical approaches shows 33% higher performance on average