SBEL

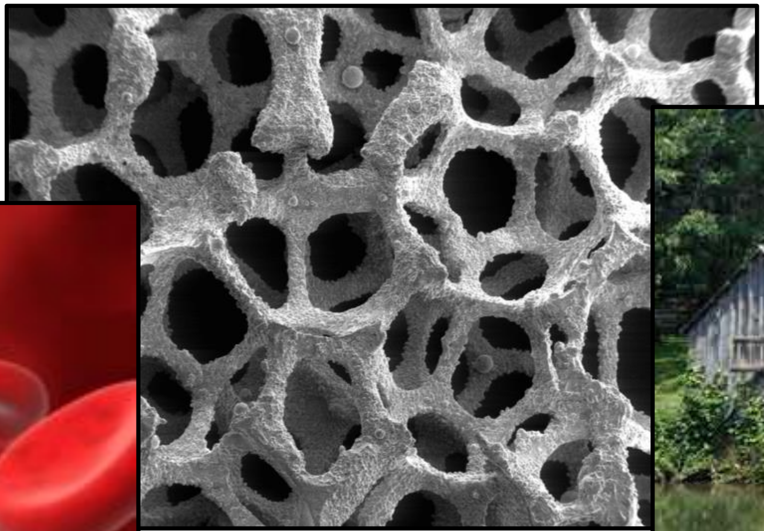
CHRONO::HPC
DISTRIBUTED MEMORY FLUID-SOLID INTERACTION SIMULATIONS

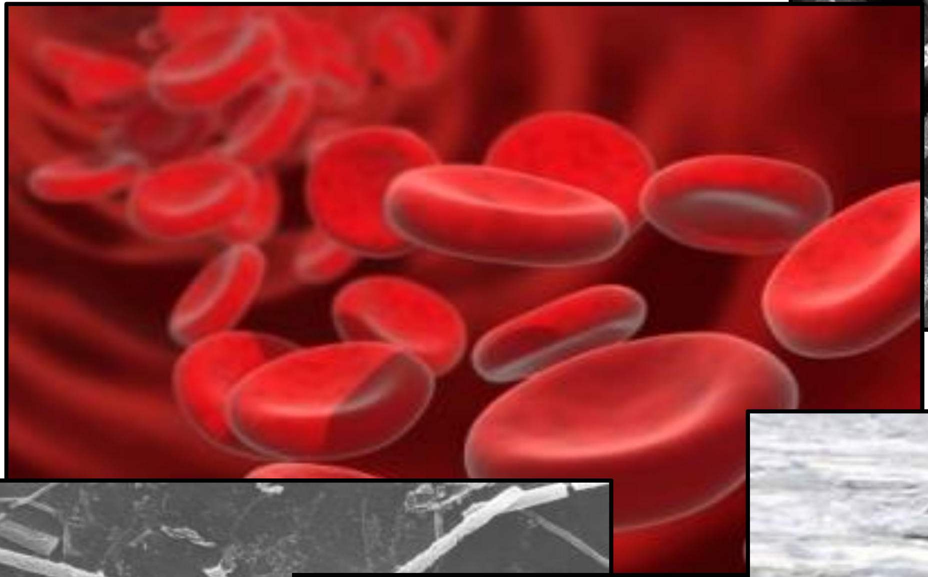
Felipe Gutierrez, Arman Pazouki, and Dan Negrut
University of Wisconsin – Madison

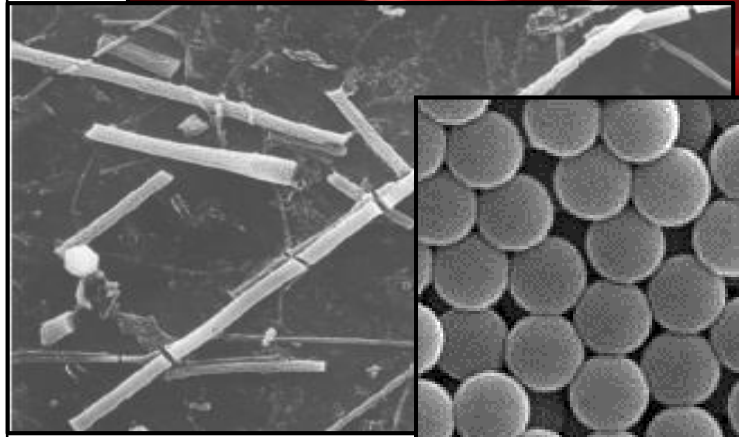
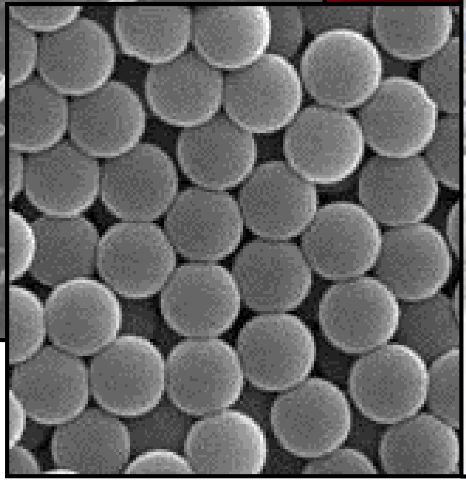
Support: Rapid Innovation Fund, U.S. Army TARDEC

ASME IDETC/CIE 2016 :: **Software Tools for Computational Dynamics in Industry and Academia**
Charlotte, North Carolina :: August 21 –24, 2016

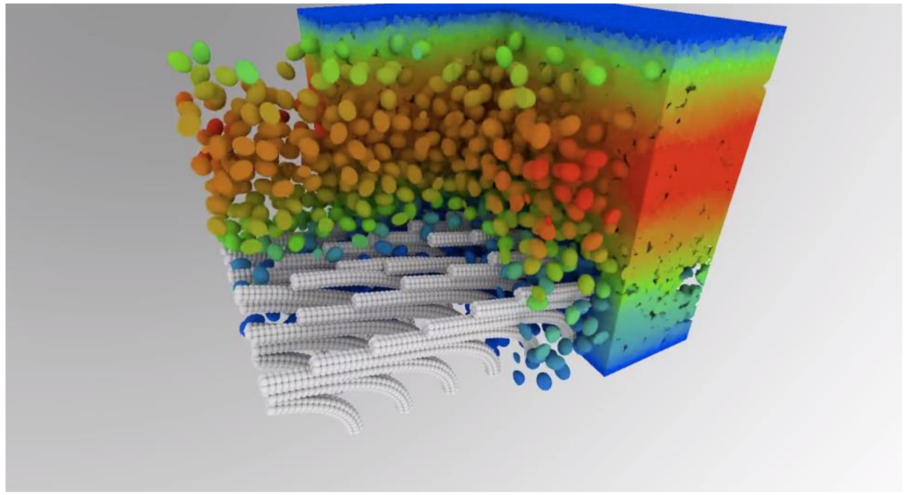
Motivation





The Lagrangian-Lagrangian framework



- Based on the work behind Chrono::FSI
- Fluid
  - Smoothed Particle Hydrodynamics (SPH)
- Solid
  - 3D rigid body dynamics (CM position, rigid rotation)
  - Absolute Nodal Coordinate Formulation (ANCF) for flexible bodies (nodes location and slope)
- Lagrangian-Lagrangian approach attractive since:
  - Consistent with Lagrangian tracking of discrete solid components
  - Straightforward simulation of free surface flows prevalent in target applications
  - Maps well to parallel computing architectures (GPU, many-core, distributed memory)
- *A Lagrangian-Lagrangian Framework for the Simulation of Fluid-Solid Interaction Problems with Rigid and Flexible Components, University of Wisconsin-Madison, 2014*

Smoothed Particle Hydrodynamics (SPH) method

- "**Smoothed**" refers to

$$\begin{aligned} f(\mathbf{x}) &= \int_S f(\mathbf{x}')\delta(\mathbf{x}-\mathbf{x}')d\mathbb{V} \\ &= \int_S f(\mathbf{x}')W(\mathbf{x}-\mathbf{x}',h)d\mathbb{V} + O(h^2) \\ &= \langle f(\mathbf{x})\rangle + O(h^2) \end{aligned}$$

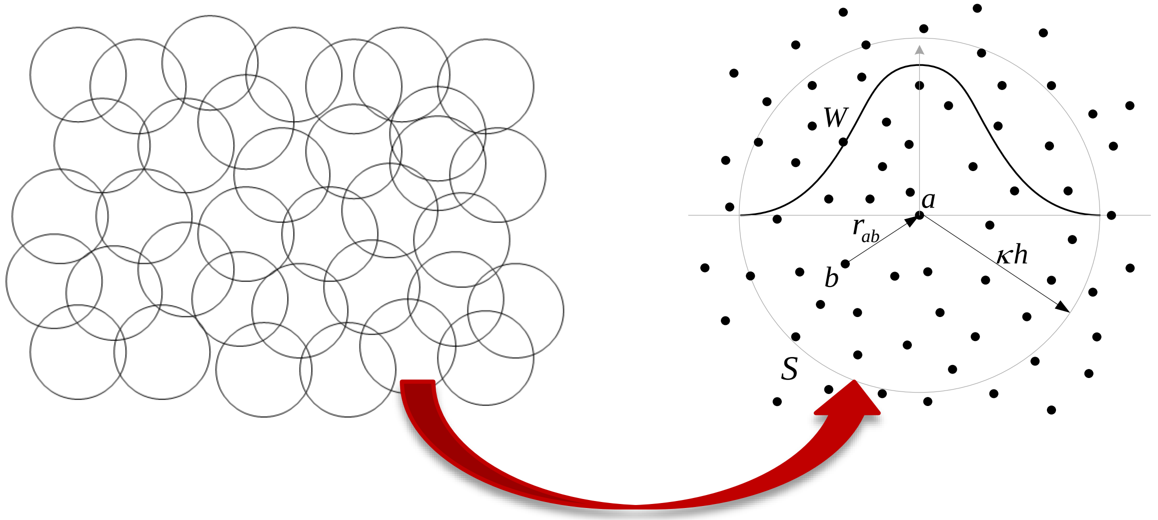
- "**Particle**" refers to

$$\begin{aligned} f(\mathbf{x}) &= \int_S \frac{f(\mathbf{x}')}{\rho(\mathbf{x}')}W(\mathbf{x}-\mathbf{x}',h)\rho(\mathbf{x}')d\mathbb{V} \\ &\simeq \sum_b \frac{m_b}{\rho_b} f(\mathbf{x}_b)W(\mathbf{x}-\mathbf{x}_b,h) \end{aligned}$$

- Cubic spline kernel (often used)

$$W(q,h) = \frac{1}{4\pi h^3}\begin{cases} (2-q)^3 - 4(1-q)^3\,, & 0 \le q < 1 \\ (2-q)^3\,, & 1 \le q < 2 \\ 0\,, & \text{otherwise} \end{cases}$$

$$\text{where } q \triangleq \frac{\|\mathbf{r}\|}{h}$$



Kernel Properties

$$\lim_{h\to 0} W(\mathbf{r},h) = \delta(\mathbf{r})$$

$$W(\mathbf{r},h) = W(-\mathbf{r},h)$$

$$\int_S W(\mathbf{r},h)\mathrm{d}\mathbb{V} = 1$$

$$\lim_{\mathbf{r}\to\infty} W(\mathbf{r},h) = 0$$

SPH for fluid dynamics

- Continuity

$$\frac{d\rho}{dt} = -\rho\nabla\cdot\mathbf{v}$$

$$\rho\nabla\cdot\mathbf{v} = \frac{\nabla\cdot\left(\rho^{\sigma-1}\mathbf{v}\right) - \mathbf{v}\cdot\nabla\rho^{\sigma-1}}{\rho^{\sigma-2}}.$$

- Momentum

$$\frac{d\mathbf{v}}{dt} = -\frac{\nabla p}{\rho} + \frac{\mu}{\rho}\nabla^2\mathbf{v} + \mathbf{f}$$

$$\frac{\nabla p}{\rho} = \frac{p}{\rho^\sigma}\nabla\left(\frac{1}{\rho^{1-\sigma}}\right) + \rho^{\sigma-2}\nabla\left(\frac{p}{\rho^{\sigma-1}}\right)$$

- In the context of fluid dynamics, each particle carries fluid properties like pressure, density, etc.

$$\frac{d\rho_a}{dt} = \sum_b m_b \left(\frac{\mathbf{v}_a - \mathbf{v}_b}{\rho_a^{\sigma-2}\rho_b^{2-\sigma}}\right)\cdot\nabla_a W_{ab}$$

$$\frac{d\mathbf{v}}{dt} = -\sum_b m_b\left(\frac{p_a}{\rho_a^\sigma\rho_b^{2-\sigma}} + \frac{p_b}{\rho_a^{2-\sigma}\rho_b^\sigma}\right)\cdot\nabla_a W_{ab} + \sum_b m_b \frac{(\mu_a+\mu_b)\mathbf{x}_{ab}\cdot\nabla_a W_{ab}}{\bar{\rho}_{ab}^2(x_{ab}^2 + \varepsilon\bar{h}_{ab}^2)}\mathbf{v}_{ab} + \mathbf{f}$$

$$\mathbf{x}_{ab} = \mathbf{x}_a - \mathbf{x}_b$$
$$W_{ab} = W(\mathbf{x}_{ab}, h)$$
$$\nabla_a = \partial/\partial\mathbf{x}_a$$

- Note: The above sums are done for millions of particles.

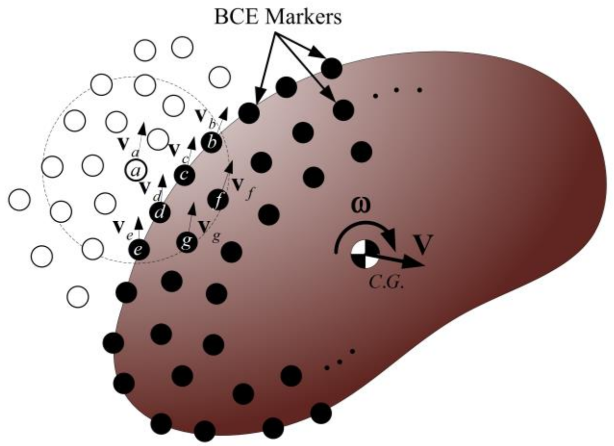
SBEL

Fluid-Solid Interaction (ongoing work)

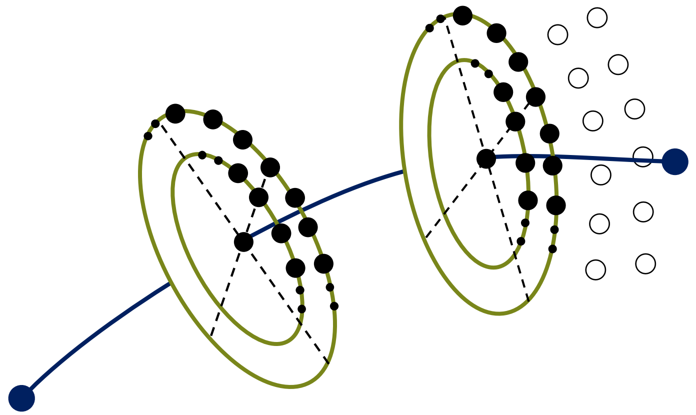
Boundary Condition Enforcing (BCE) markers for no-slip condition

- Rigidly attached to the solid body (hence their velocities are those of the corresponding material points on the solid)
- Hydrodynamic properties from the fluid

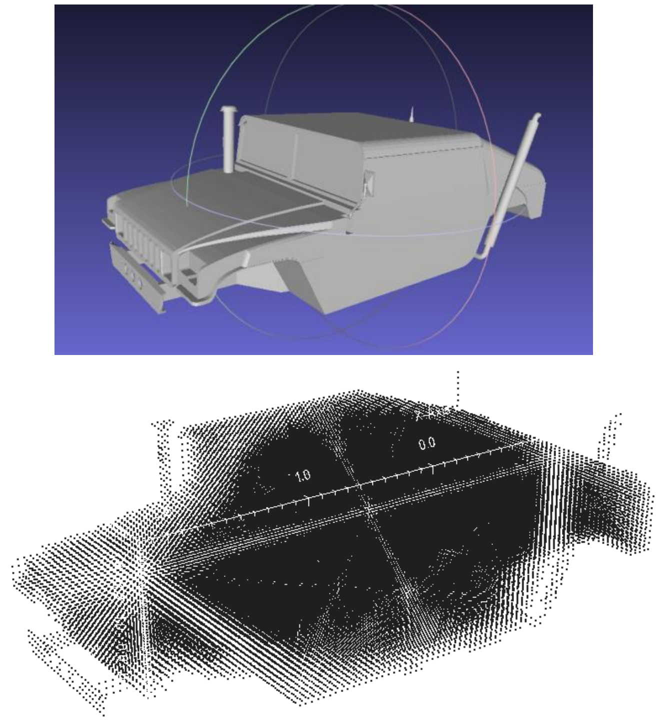
Rigid bodies/walls



Flexible Bodies



Example Representation

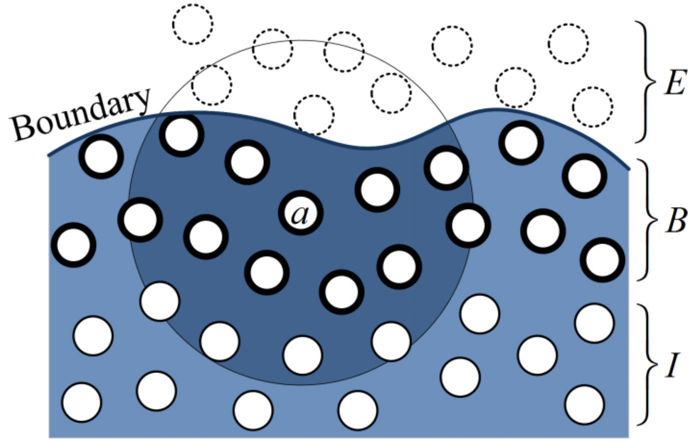


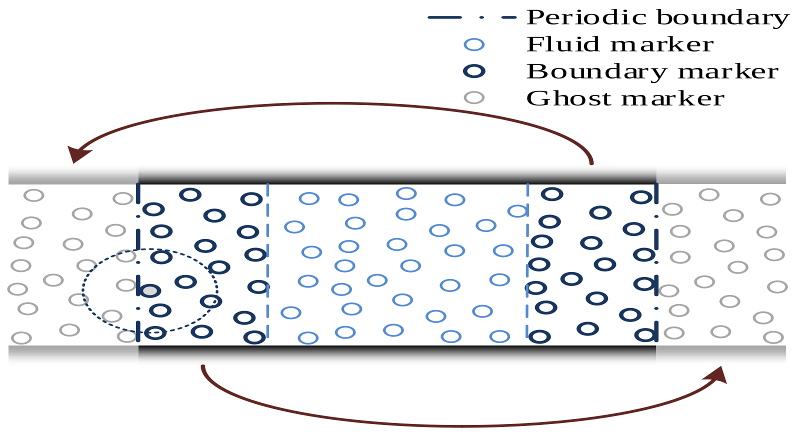
Current SPH Model

- Runge-Kutta 2nd order
  - Requires force calculation to happen twice per step
- Wall Boundary
  - Density changes for boundary particles as you would for the fluid particles.

$$\frac{d\rho_a}{dt} = \rho_a \sum_b \frac{m_b}{\rho_b} (\mathbf{v}_a - \mathbf{v}_b) \cdot \nabla_a W_{ab} \longleftrightarrow \rho_a = \sum_b m_b W_{ab}$$

- Periodic Boundary Condition
  - Markers who exit the periodic boundary, enter from the other side





Challenges for Scalable Distributed Memory Codes

- SPH is a computationally expensive method, hence, high performance computing (HPC) is necessary.
- High Performance Computing is hard.
  - MPI codes are able to achieve good strong and weak scaling, but… the developer is in charge of making this happen.
- Distributed memory challenges:
  - Communication bottlenecks > Computation bottlenecks
  - Load imbalance
  - Heterogeneity: processor types, process variation, memory hierarchies, etc.
  - Power/Temperature (becoming an important)
  - Fault tolerance
- To deal with these, we would like to seek
  - Not full automation
  - Not full burden on app-developers
  - But: a good division of labor between the system and app developers

Solution: Charm++

- Charm++ is a generalized approach to writing parallel programs
  - An alternative to the likes of MPI, UPC, GA etc.
  - But not to sequential languages such as C, C++, and Fortran
- Represents:
  - The style of writing parallel programs
  - The runtime system
  - And the entire ecosystem that surrounds it
- Three design principles:
  - **Overdecomposition, Migratability, Asynchrony**

Charm++ Design Principles

Overdecomposition

- Decompose work and data units into many more pieces than processing elements (cores, nodes, …).
- Not so hard: problem decomposition needs to be done anyway.

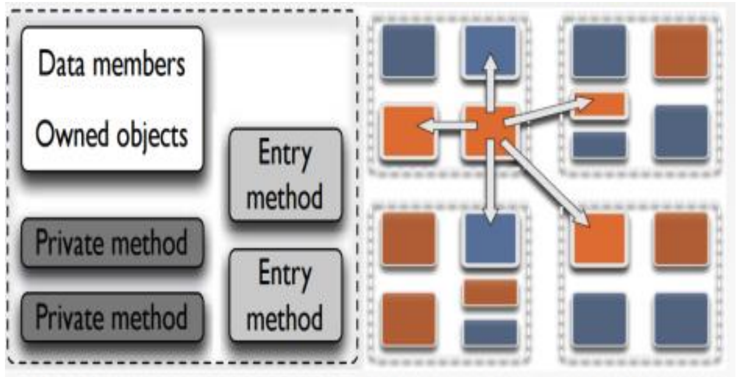
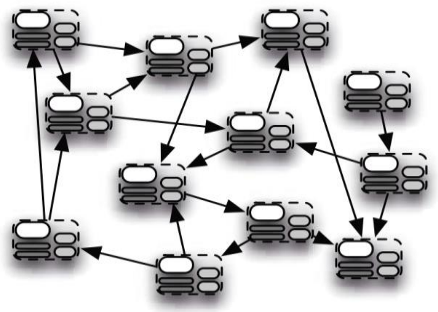


Figure 1: Single Chare Object (left). Overdecomposition; multiple chares in each execution unit exchanging data (right).

Migratability

- Allow data/work units to be migratable (by runtime and programmer).
- Communication is addressed to logical units (C++ objects) as opposed to physical units.
- Runtime System must keep track of these units



(b) Programmer's view: Collection of interacting chares

Asynchrony

- Message-driven execution
  - Let the work unit that happens to have data ("message") available execute next.
  - Runtime selects which work unit executes next (user can influence) → Scheduling

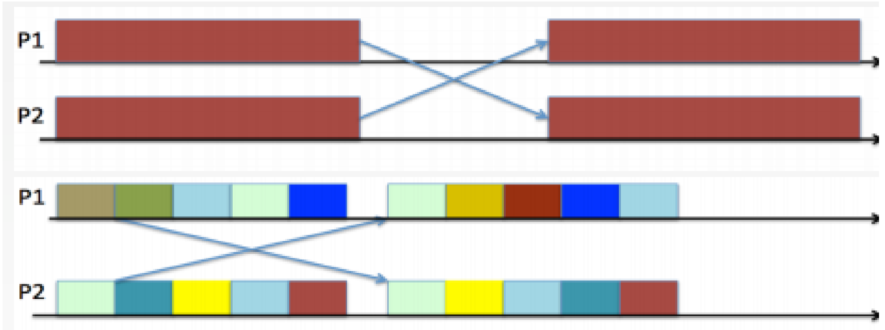


Figure 3: Compute idle time in MPI (top). Reduced idle times due to overdecomposition (bottom).

Realization of the design principle in Charm++

- Overdecomposed entities: **chares**
  - Chares are C++ objects
  - With methods designated as “entry” methods
    - Which can be invoked asynchronously by remote chares
  - Chares are organized into indexed collections
    - Each collection may have its own indexing scheme
      - 1D, ..7D
      - Sparse
      - Bitvector or string as an index
  - Chares communicate via asynchronous method invocations: **entry methods**
    - A[i].foo(....); A is the name of a collection, i is the index of the particular chare.
- It is a kind of task-based parallelism
  - Pool of tasks + pool of workers
  - Runtime system selects what executes next.

Charm-based Parallel Model for SPH

- Hybrid decomposition (domain + force)
  - Inspired by NaMD (molecular dynamics application)
  - Domain Decomposition: 3D Cell Chare Array.
    - Each cell contains fluid/boundary/solid particles.
    - Data Units
    - Indexed: (x, y ,z)
  - Force decomposition: 6D Compute Chare Array
    - Each compute chare is associated to a pair of cells.
    - Work units.
    - Indexed (x1, y1, z1, x2, y2, z2)
- No need to sort particles to find neighbor particles (overdecomposition implicitly takes care of it).
- Similar decomposition to LeanMD.
  - Charm++ Molecular Dynamics mini-app.
  - Kale, et al. "Charm++ for productivity and performance". PPL Technical Report, 2011.

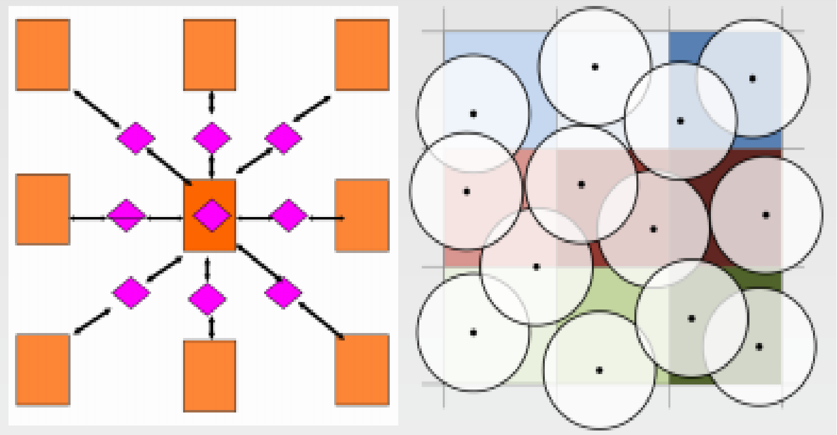
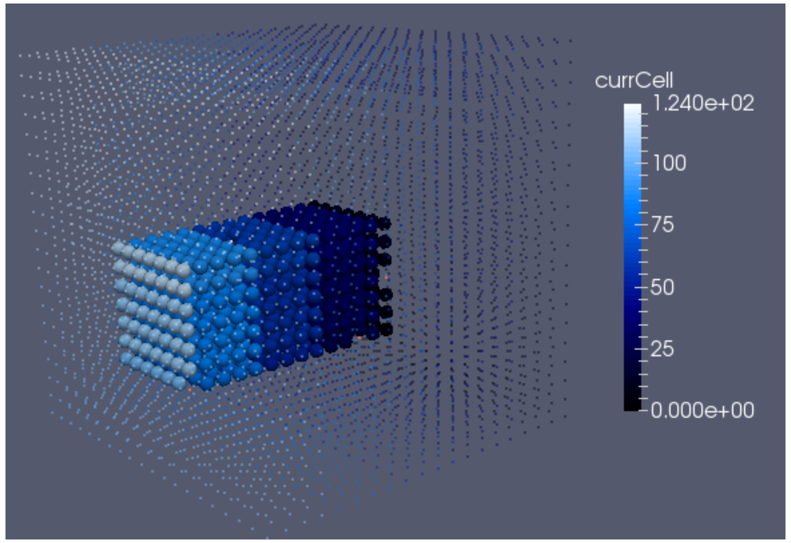


Figure 4: Hybrid decomposition: cell chares (orange) and compute chares (pink) (left). Particle grouped by cell, showing the interaction radius (right).



Algorithm (Charm-based SPH)

1. Init each Cell Chare (very small subdomains)
2. For each subdomain create the number of Compute Chares

The following instructions happen in parallel for each Cell/Compute Chare.	
Cell Array Loop (For each time step)	Compute Array Loop (For each time step)
3. SendPositions to each associate compute chare	4. When calcForces → SelfInteract OR Interact
6. Reduce forces from each compute chare	5. Send resulting forces
7. When reduce forces update properties at halfStep	
Repeat 3-7, but calc forces with marker properties at half step.	
8. Migrate Particles to Neighbors	
9. Load Balance every n steps	

SBEL

Charm-based Parallel Model for FSI (ongoing work)

- Particles representing the solid will be contained with the fluid and boundary particles.
- Solid Chare Array (1D Array)
  - Particles keep track of the index of the solid they are associated with.
  - Once computes are done they send a message (invoke an entry method) to each solid they have particles of.
  - Do a force reduction and calculate the dynamics of the solid.

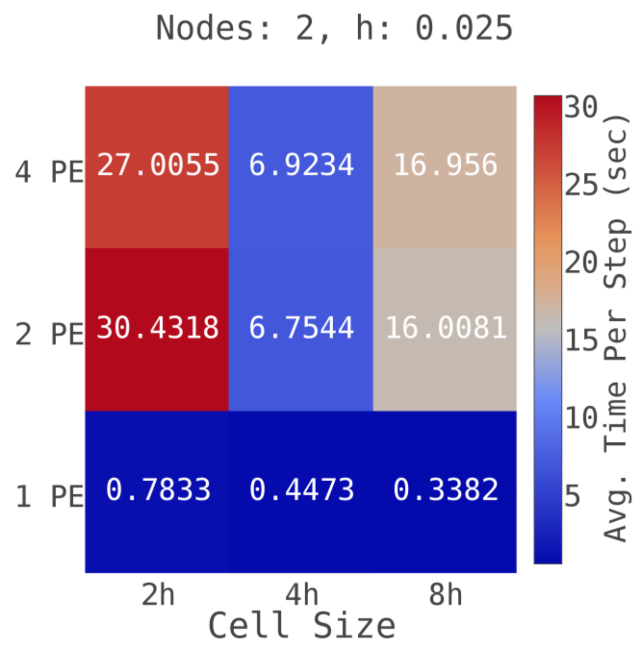
Charm++ In Practice

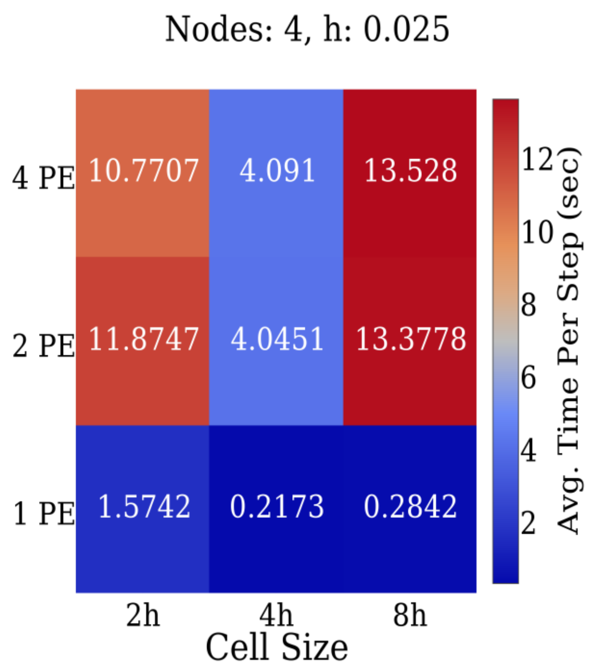
- Achieving optimal decomposition granularity
  - Average number of markers allowed per subdomain = Amount of work per chare.
  - Make sure there is enough work to hide communications.
  - Way too many chare objects is not optimal → Memory + Scheduling overheads
- Hyper Parameter Search
  - Vary Cell Size → Changes total number of cells and computes.
  - Vary Charm++ nodes per physical node → Feed comm network at max rate.
    - Varies number of communication and scheduling threads per node.
    - System specific. Small clusters might only need a single Charm++ node (1 communication thread), but larger clusters with different configurations might need more)

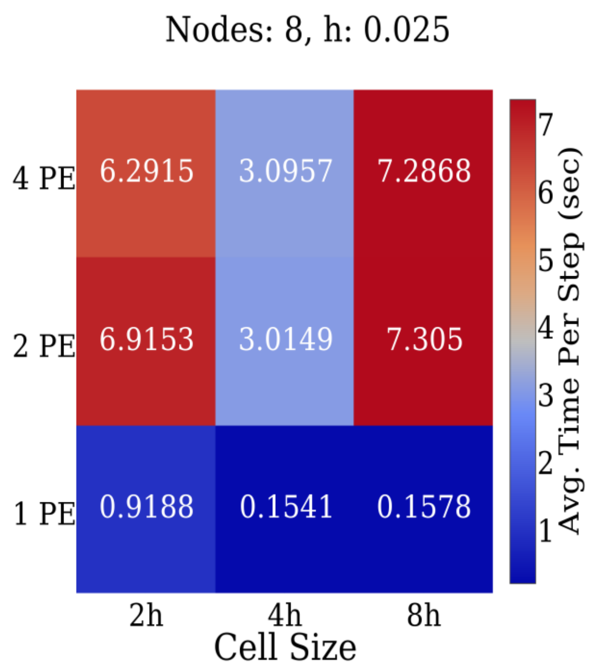
Charm++ Nodes\CellSize	**2 * h**	**4 * h**	**8 * h**
aprun -n 8 **-N 1** -d 32 ./charmsph +ppn 31 +commap 0 +pemap 1-31	Average times per time step		
aprun -n 16 **-N 2** -d 16 ./charmsph +ppn 15 +commap 0,16 +pemap 1-15:17-31			
aprun -n 32 **-N 4** -d 8 ./charmsph +ppn 7 +commap 0,8,16,24 +pemap 1-7:9-15:17-23:25-31			

Results: Hyper parameter Search

- Hyper parameter search for optimal cell size and Charm++ nodes per physical node. Nodes denotes physical nodes (64 processors per node), and h denotes the particle interaction radius.
- H = Interaction radius of SPH particles.
- PE = Charm++ node (equivalent to MPI rank).

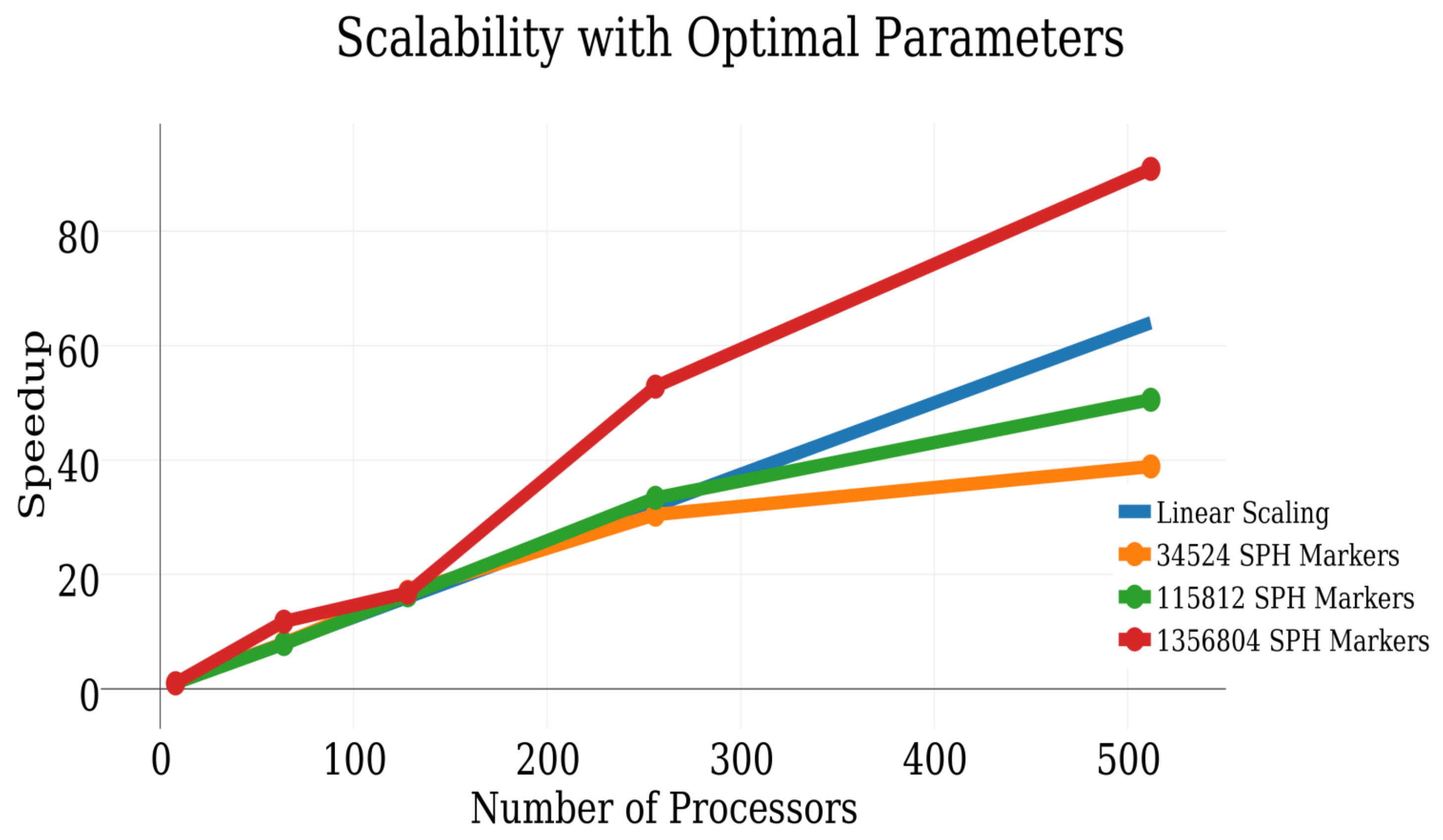






Results: Strong Scaling

- Speeups calculated with respect to an 8 core run (8-504 cores).



Results: Dam break Simulation

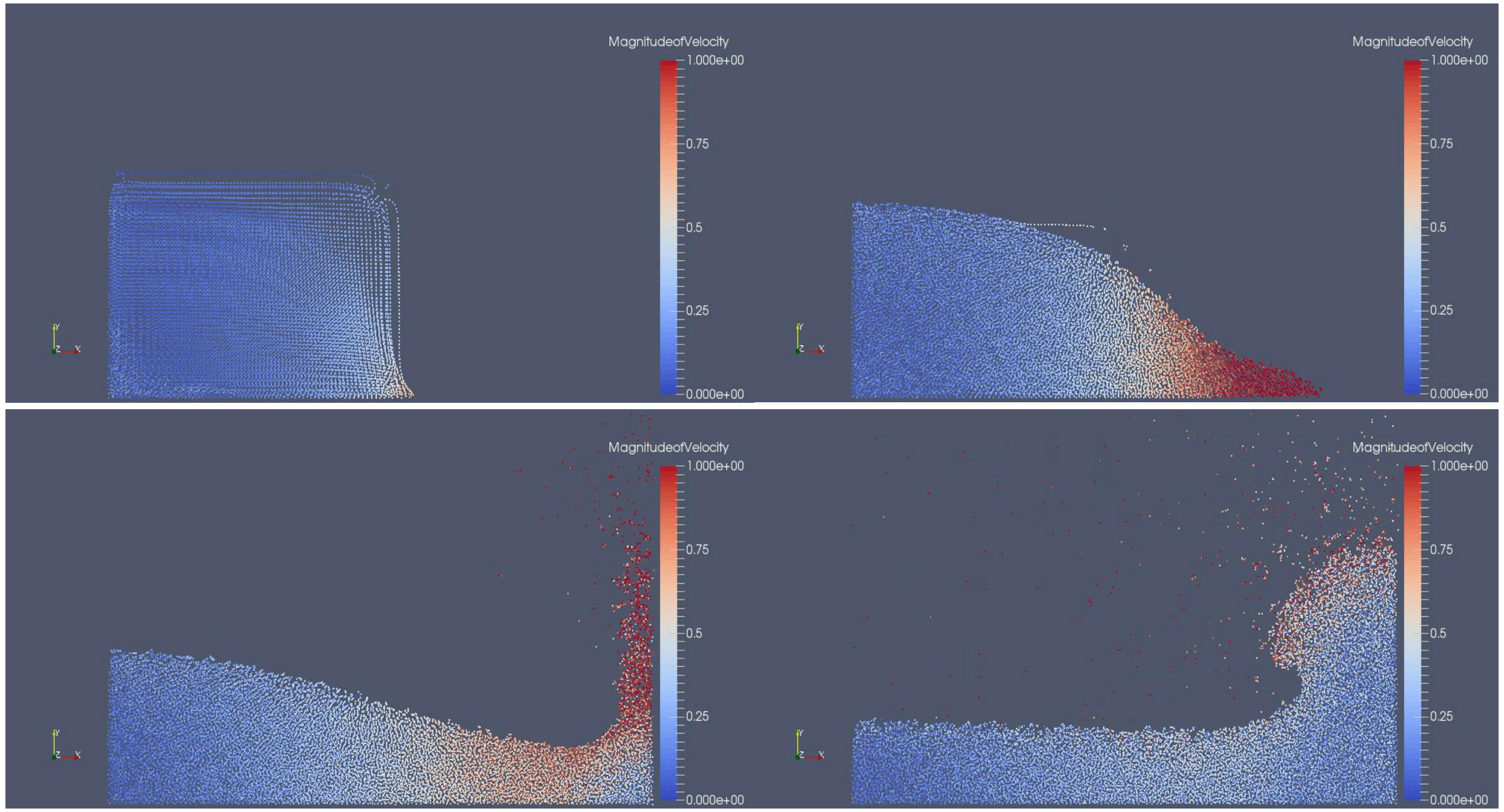


Figure 3: Dam break simulation (139,332 SPH Markers).

Note: Plain SPH requires hand tuning for stability.

Future Work (a lot to do)

- Improve the current SPH model following the same communication patterns for kernel calculations
  - Density Re-initialization.
  - Generalized Wall Boundary Condition
    - Adami, S., X. Y. Hu, and N. A. Adams. "A generalized wall boundary condition for smoothed particle hydrodynamics." *Journal of Computational Physics*231.21 (2012): 7057-7075.
    - Pazouki, A., B. Song, and D. Negrut. "Technical Report TR-2015-09." (2015).
- Validation
- Hyper parameter search and scaling results on larger clusters.
  - Some bugs in HPC codes only appear after 1,000+ or 10,000+ cores.
- Performance+scaling comparison against other distributed memory SPH codes.
- Fluid-Solid Interaction
  - A. Pazouki, R. Serban, and D. Negrut, A Lagrangian-Lagrangian framework for the simulation of rigid and deformable bodies in fluid, Multibody Dynamics: Computational Methods and Applications, ISBN: 9783319072593, Springer, 2014.

Thank you!

Questions?

Code available at: https://github.com/uwsbel/CharmSPH