Massively Parallel OpenMP-MPI Implementation of the SPH Code DualSPHysics

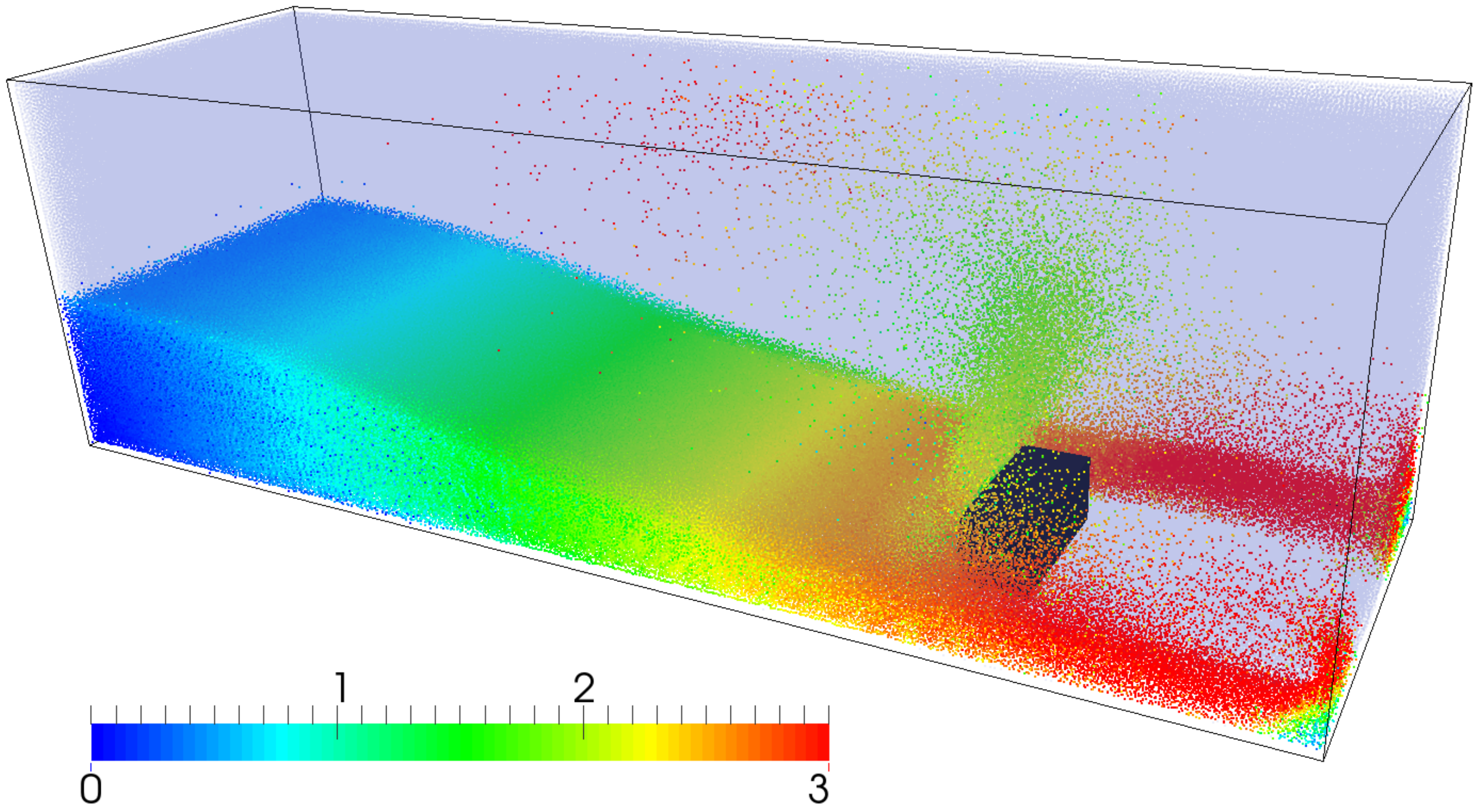
Athanasios Mokos, Benedict D. Rogers

School of Mechanical, Aeronautical and Civil Engineering
University of Manchester, UK

Funded by the eCSE, eCSE07-16

Outline of Presentation

- Motivation for Research
- Introduction to Meshless Methods
  - Introduction to SPH
- Message Passing Interface
  - Domain Division
  - Process Communication
  - Asynchronous communications
- Results
  - Runtime Results
- Optimization
  - Dynamic Load Balancing
  - Domain Decomposition
- 2D/3D Decomposition
  - Zoltan Library
  - Domain Decomposition Algorithms
- Conclusions and Future Work



Motivation for Research

- Primary focus on violent water flows with breaking free surface, e.g. wave impact/slamming or potentially explosive pipe flows

- Applications:
  - Coastal Defence
  - Offshore structures
  - Dam and river flooding

- Experiments are expensive and require significant investment

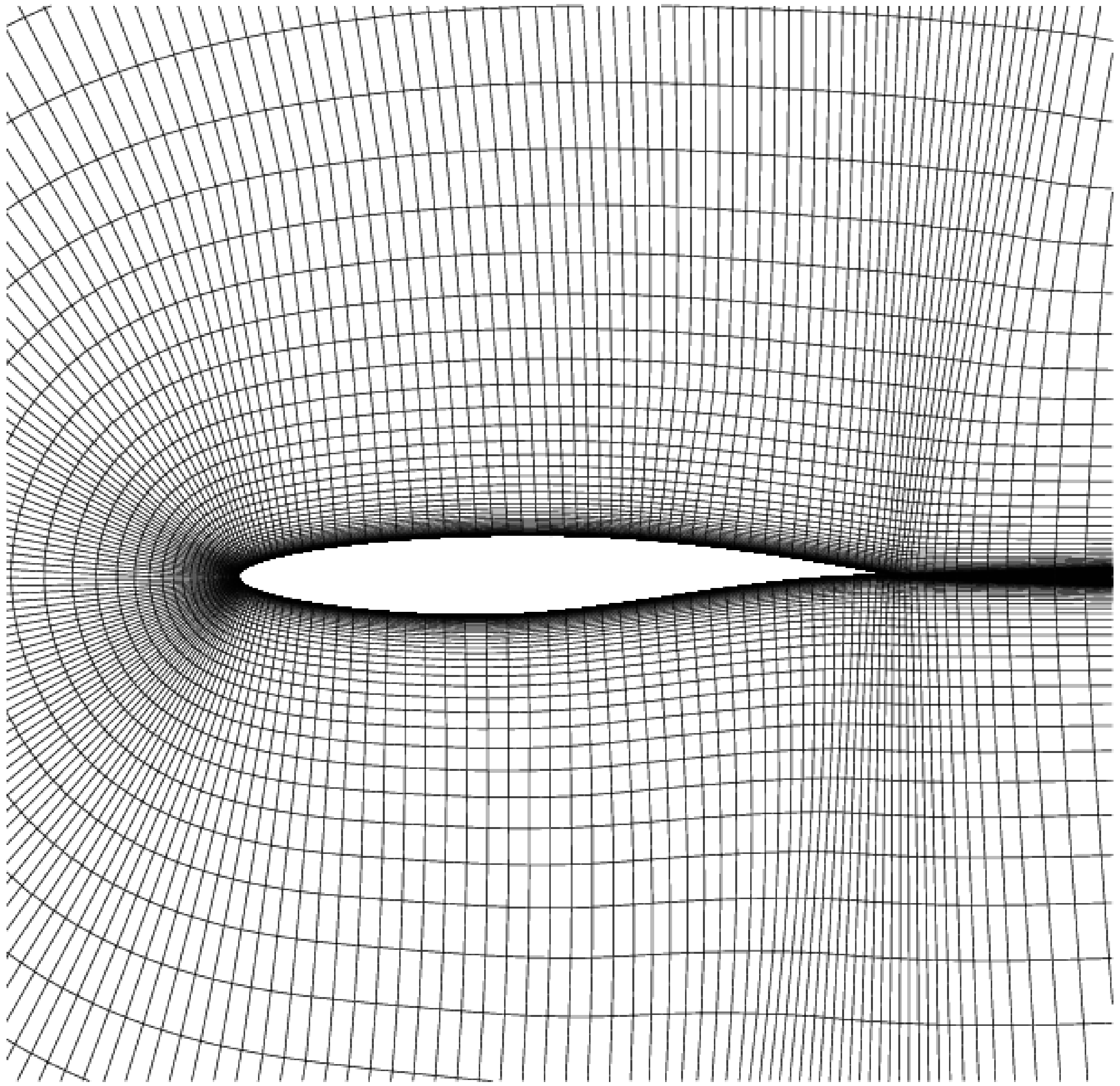
- Focus on computational methods

Whitehaven 2013 (North News)

Mesh-based Methods

- Most common methods used in both industry and academia:
  - Finite Difference Method (FDM)
  - Finite Element Method (FEM)
  - Finite Volume Method (FVM)
- Robust, well-developed and mature
  - Multiple algorithms and models
  - Adapted for every application

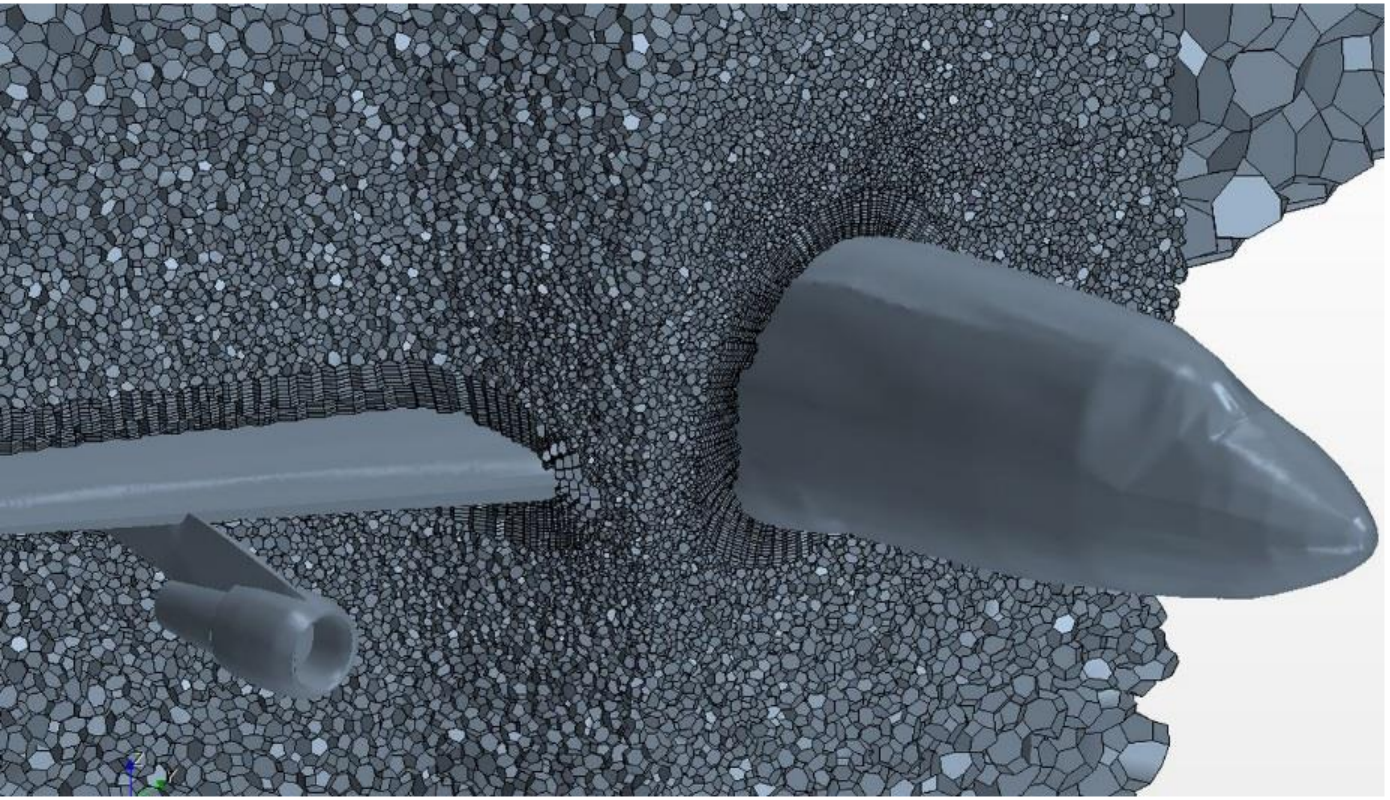
However…


Mesh around a RAE 2822 transonic aerofoil (NASA)

Mesh-based Methods

Meshing can be complex

Deformation?



Mesh around an airplane hull

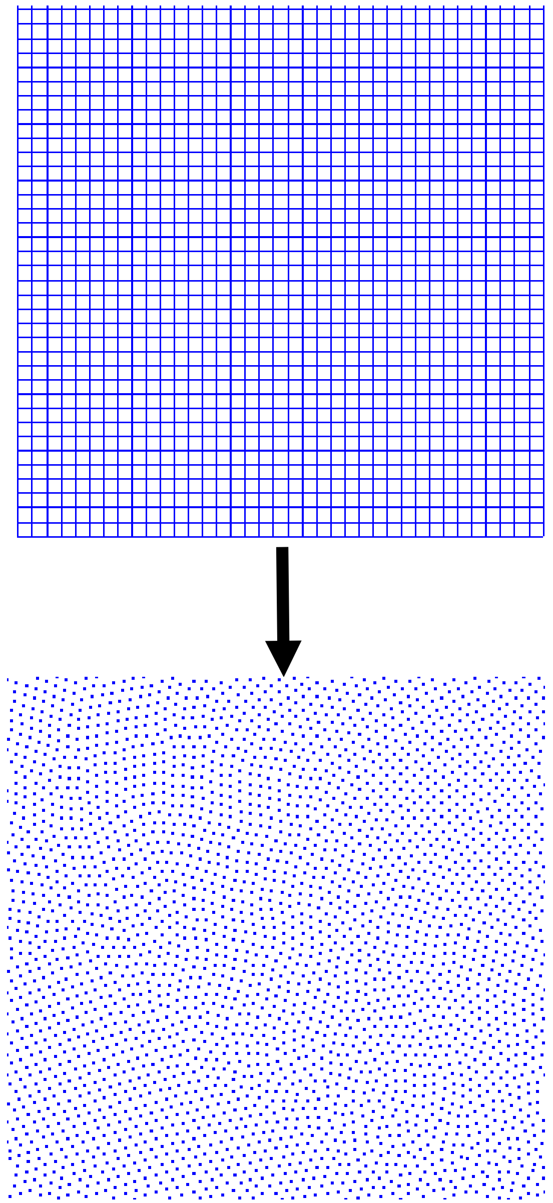
(Photo courtesy of Siemens)

Waves breaking on the shore

(Photo courtesy of the University of Plymouth)

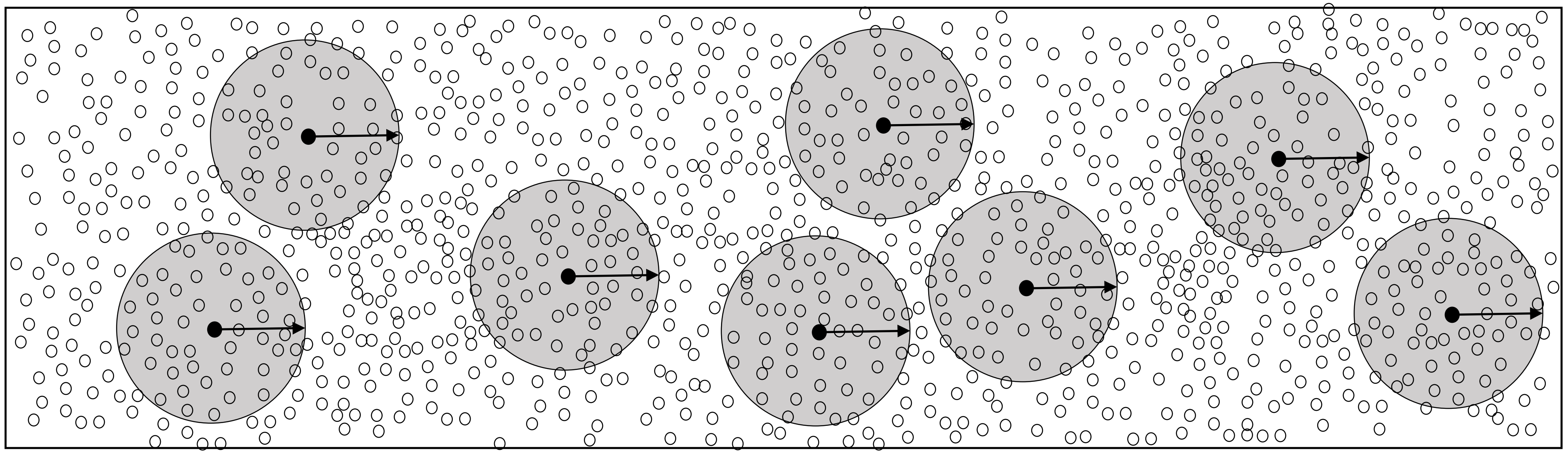
Meshless Methods

- Computation Points: Nodes -> **Particles**
- Particles are not connected with a grid
- Particles are not fixed but **move** with their own velocity and acceleration
- Each particle follows a unique trajectory
- Particles are described through **Lagrangian** derivatives: Rate of change along a trajectory



Local Interpolation

- Particles possess **properties** (density, pressure etc.) travelling with them
- Particles are linked to their current **neighbouring particles in space**
- Neighbours' values affect the properties of the particle through a summation
- Particle movement also affected by neighbours



Introduction to Smoothed Particle Hydrodynamics

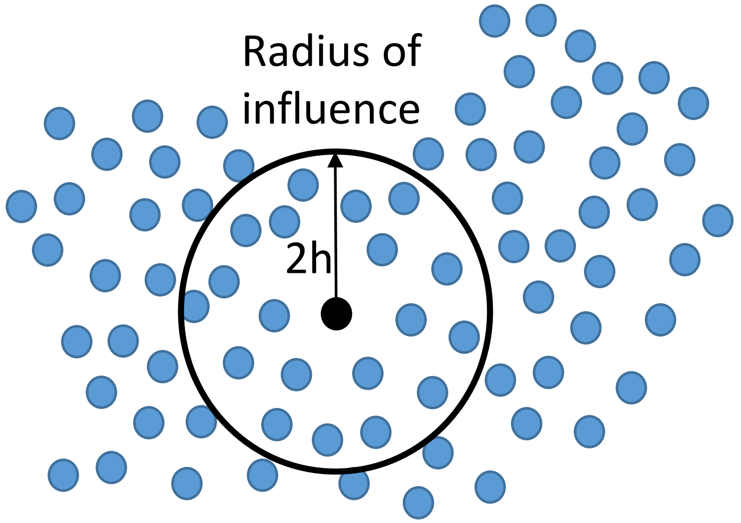
- SPH is a **Lagrangian meshless** method: Computation points (**particles**) move according to governing equations (Navier-Stokes Equations)

- **Basic idea**: The value of a function A(**r**) at point **r** in space is approximated as:

$$A(\boldsymbol{r}) = \int_{\Omega} A(\boldsymbol{r}')\delta(\boldsymbol{r}-\boldsymbol{r}')\,\mathrm{d}\Omega$$

- Properties computed through **local interpolation** with a weighting function (**kernel**) around each particle

$$\langle A(\boldsymbol{r})\rangle \approx \sum_{j=1}^{N} \frac{m_j}{\rho_j} A(\boldsymbol{r}_j) W(\boldsymbol{r}-\boldsymbol{r}_j, h)$$



Introduction to SPH

- Navier-Stokes Equations
  - Continuity $$\frac{\mathrm{d}\rho}{\mathrm{d}t} = -\rho\nabla \boldsymbol{v} \quad \rightarrow \quad \left\langle \frac{\mathrm{d}\rho_i}{\mathrm{d}t} \right\rangle = \sum_j m_j \left(\boldsymbol{u}_i - \boldsymbol{u}_j\right) \cdot \nabla_i W_{ij}$$
  - Momentum $$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} = -\frac{1}{\rho}\nabla p + \nu\nabla^2 \boldsymbol{u} + \boldsymbol{F} \rightarrow \left\langle \frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} \right\rangle = -\sum_j m_j \left[\frac{p_j}{\rho_j^2} + \frac{p_i}{\rho_i^2}\right] \nabla_i W_{ij} + \Pi_{ij} + \boldsymbol{F}_i$$

- Fluid Compressibility
  - Incompressible SPH ⟶ Poisson Equation
  - Weakly Compressible SPH ⟶ Equation of State

SPH for real problems

- Real-life applications are complex 3D flows
- Multi-scale problems with long runtimes
- SPH requires over 10^8 particles to model them
- Must do so as quickly as possible

OPTION: Use the inherent parallelism of the **GPU**

Photo by University of Plymouth

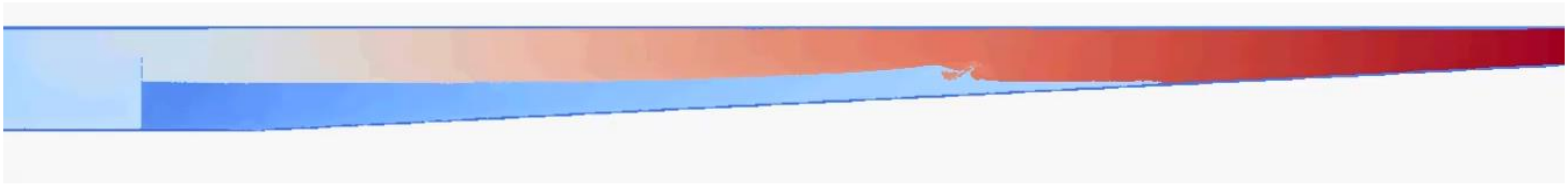
SPH Solver DualSPHysics

http://dual.sphysics.org/

- Open-source project, co-developed with the Universities of Vigo, Manchester, Parma and Lisbon

- Validated for violent water flows[1]

- Includes pre- and post processing software

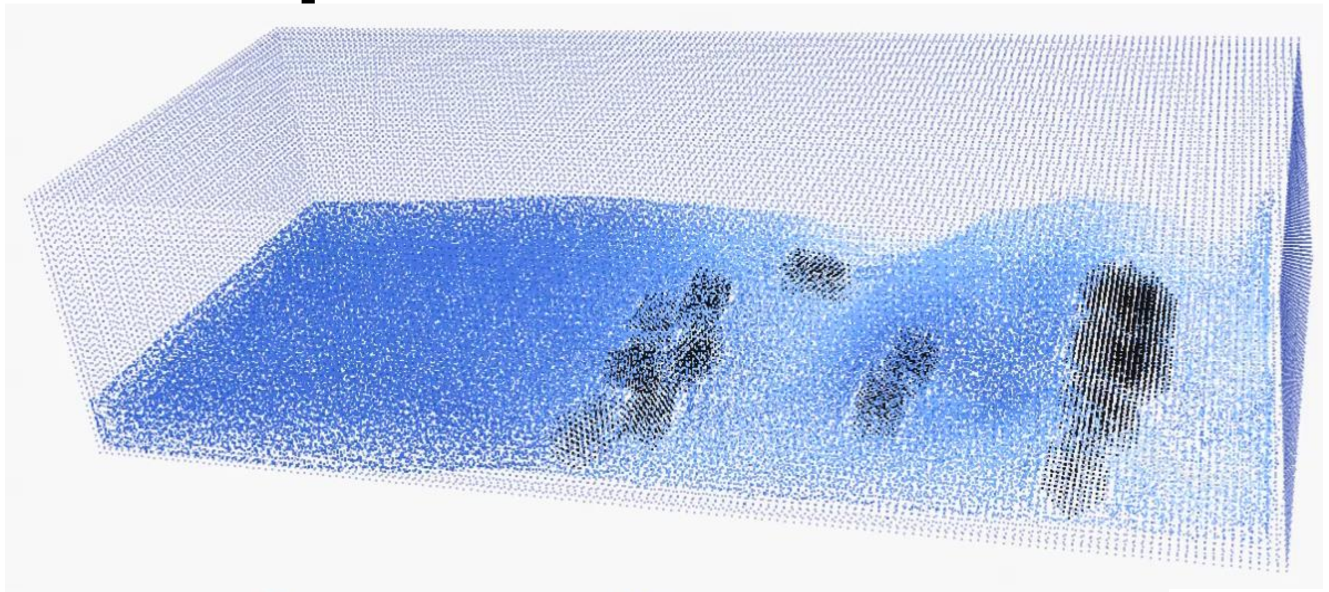


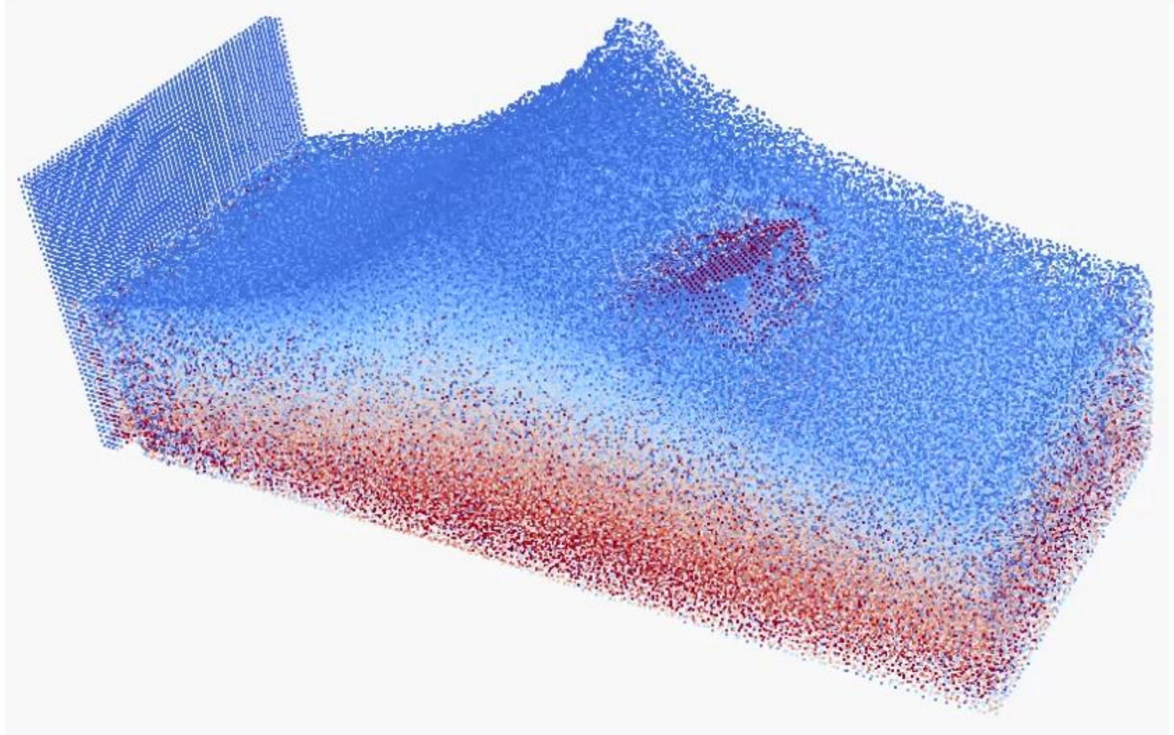
http://www.youtube.com/user/DualSPHysics/videos

Additional Capabilities

Integration with all existing capabilities of DualSPHysics

- Wave Generator
- DEM model
- Floating Bodies
- Air-Water multiphase model
- Solid-Water multiphase model
- Object motion





Current State of DualSPHysics

GPU

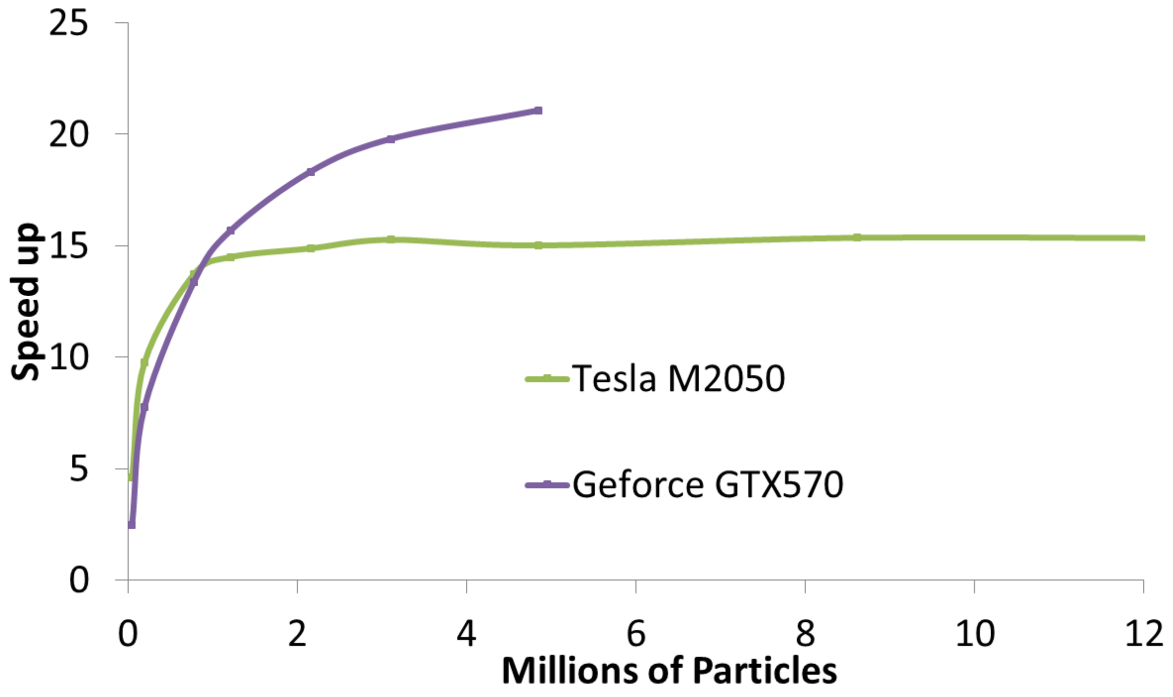
- Highly optimised code
- Multiple options
- Pre- and post-processing tools
- Able to take advantage of the inherent parallelism
- Simulates millions of particles in a few **hours**

CPU

- Highly optimised code
- Multiple options
- Pre- and post-processing tools
- OpenMP implementation
- Simulates millions of particles in a few **months**

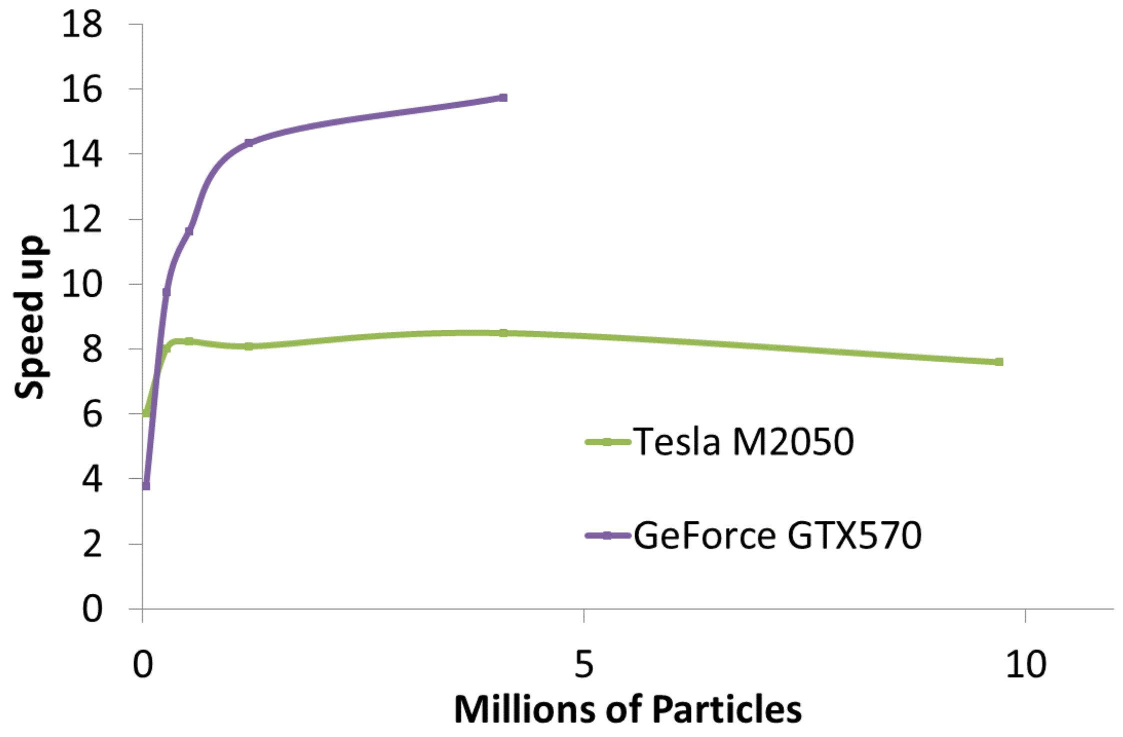
Current State of DualSPHysics

2D



Speedup up to **21** for a 6-year old card compared to an 8-thread OpenMP simulation

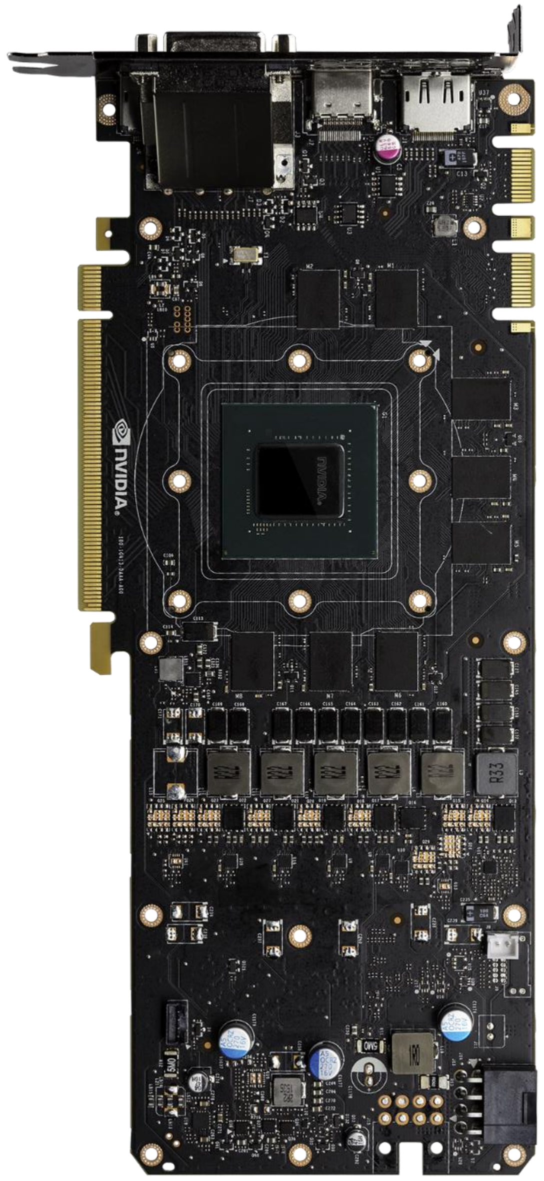
3D



Speedup up to **16** for a 6-year old card compared to an 8-thread OpenMP simulation

Current State of DualSPHysics

- GPUs are fantastic:
  - Massively Parallel, ideal for n-body simulations
  - Low cost and energy consumption (**Green Computing**)
- But…
  - Still in their infancy (less developed tools and compilers)
  - Single precision to maintain speed
  - The multi-GPU code is not easily portable
  - Require specialised hardware and additional investment (cannot take advantage of existing HPC infrastructure)
  - Industrial engineering companies still need convincing to invest resources and personnel


NVidia GTX1080

Motivation for Research

- Develop a **CPU code** with similar capabilities to the existing GPU code that can be used in HPC installations
- Massive Parallelism required: Ability to scale for **1000s** of cores
- Currently only local parallelism (OpenMP) -> Communication between different processors required
- Implementation of the **Message Passing Interface (MPI)** standard

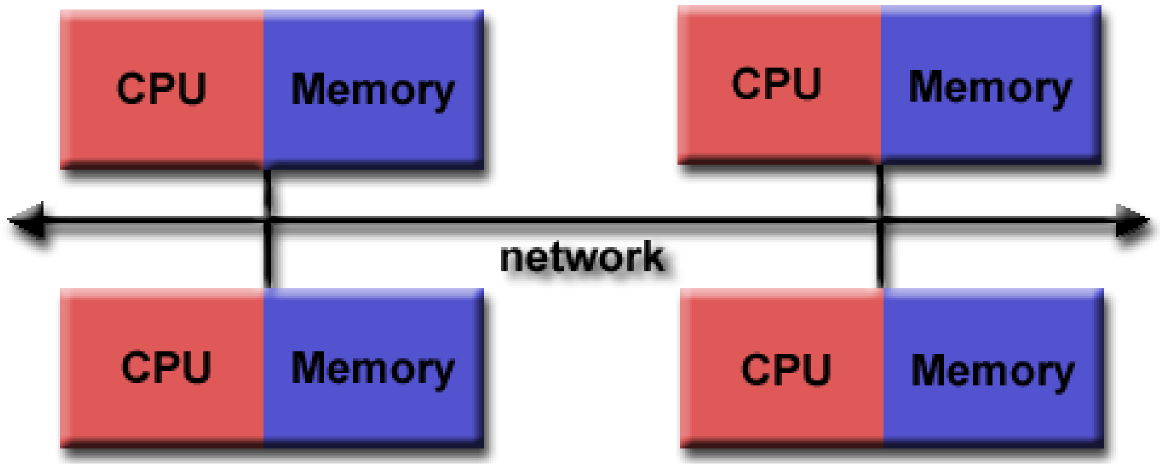
- **AIM: Develop a hybrid OpenMP-MPI program that can scale to 1000s of cores**

Message Passing Interface

- Standardised, independent and portable message parsing library specification

- **Message Passing**: Data is moved from one process to another through cooperative operations on each process. The recipient then selects the appropriate code to be executed.

Distributed memory model



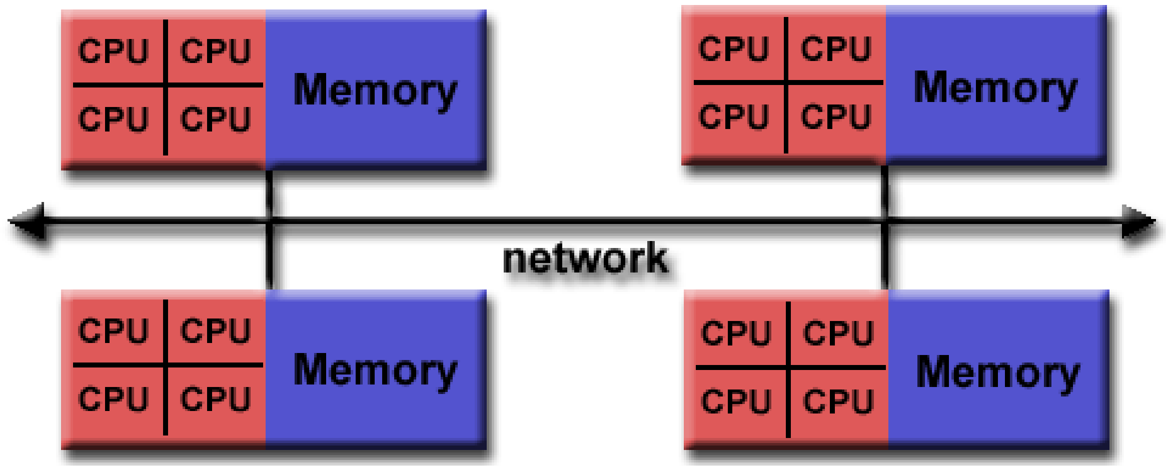
Message Passing Interface

- Standardised, independent and portable message parsing library specification

- **Message Passing**: Data is moved from one process to another through cooperative operations on each process. The recipient then selects the appropriate code to be executed.

OpenMP already developed so…

Hybrid memory model



Challenges of Integrating MPI

- Maintain DualSPHysics optimisation and structure
  - Cell-linked neighbor list[3]
  - Ease of use
  - Reduce changes in SPH computation
  - Limits options when creating particles and cells

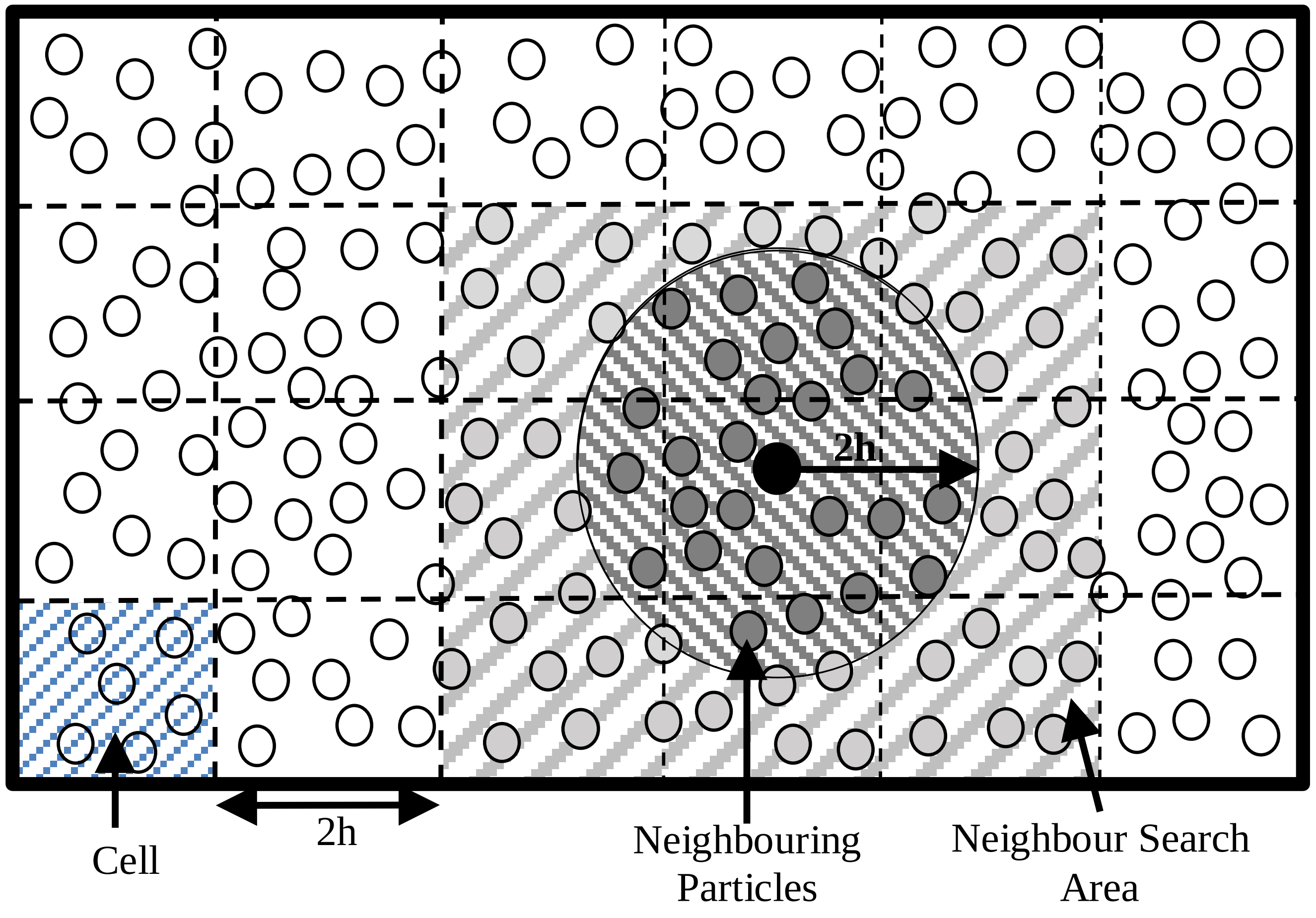
- Need to introduce new features
  - Focus on updating existing functions to work with multiple nodes
  - Create new files to handle communication and data transfer

SPH Solver DualSPHysics

http://dual.sphysics.org/

Cell-linked neighbour list[3]

- Algorithm that optimises neighbour searching
- Divide the domain into cells
- Cells remain constant throughout the computation
- Create a list linking particles and cells
- Search for neighbour particles only in adjacent cells



Integrating MPI in DualSPHysics

Single node files

- JCellDivCpuSingle
- JPartsLoad4
- JSphCpuSingle

MPI files

- CellDivCpuMPI
- ParticleLoadMPI
- SphCpuMPI

- Changes focused on:
  - Loading data from pre-processing software
  - Creating and updating the assignment of particles in cells
  - Handling and integrating the new features

Integrating MPI in DualSPHysics

Single node files

- JCellDivCpuSingle
- JPartsLoad4
- JSphCpuSingle

MPI files

- CellDivCpuMPI
- ParticleLoadMPI
- SphCpuMPI

New files created to handle:

- Node communication
- Domain Decomposition
- Halo Exchange

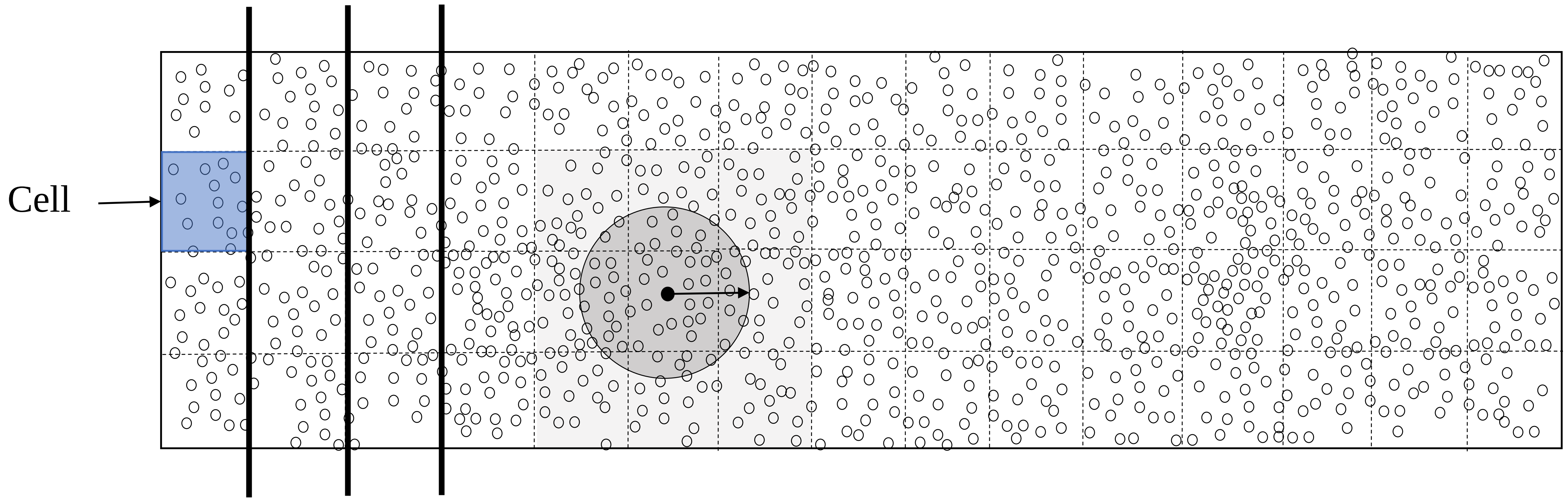
- BufferMPI
- DataCommMPI
- HostMPI
- InfoMPI
- SliceMPI
- SphMPI
- SphHaloMPI

Domain Decomposition

- Divide the domain between nodes
- Unique particle and cell list
- 1D decomposition through slices[2]

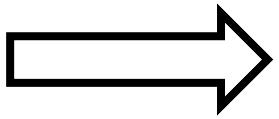
⇒

- Allows the simulation to use more particles
- Reduces local and global memory footprint
- Reduces the load on each CPU core

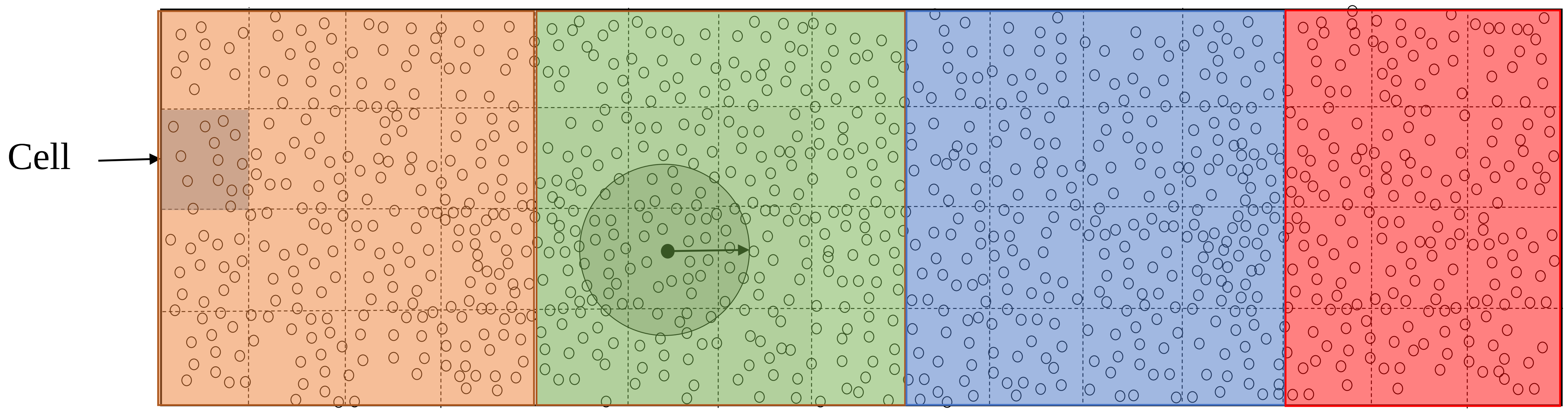


Domain Decomposition

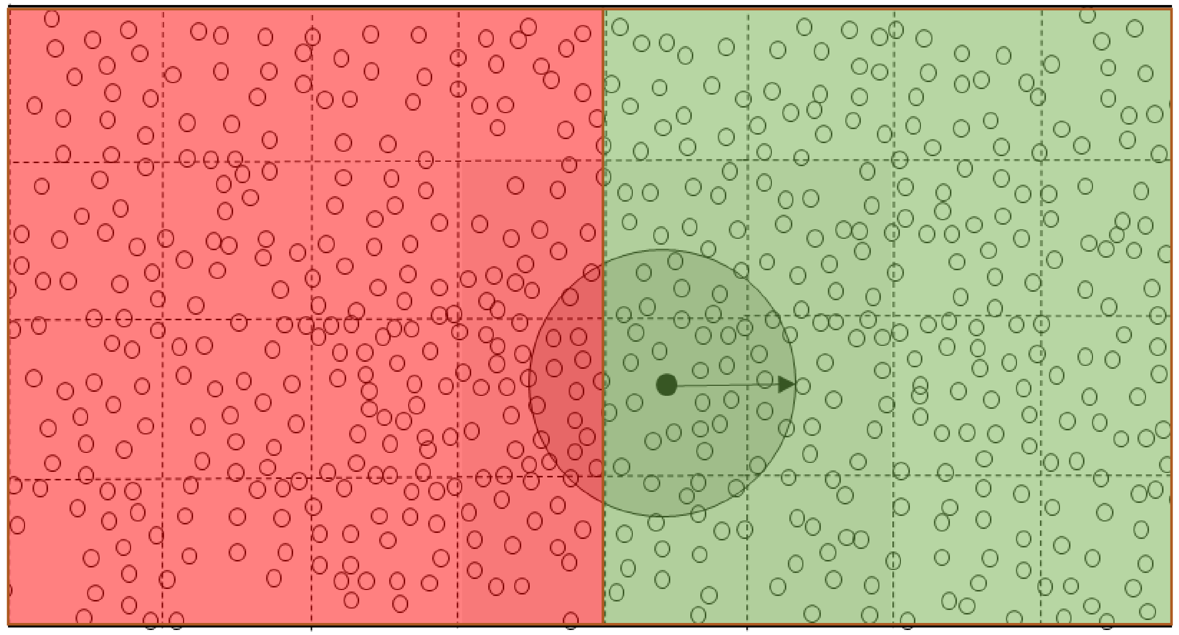
- Divide the domain between nodes
- Unique particle and cell list
- 1D decomposition through slices[2]

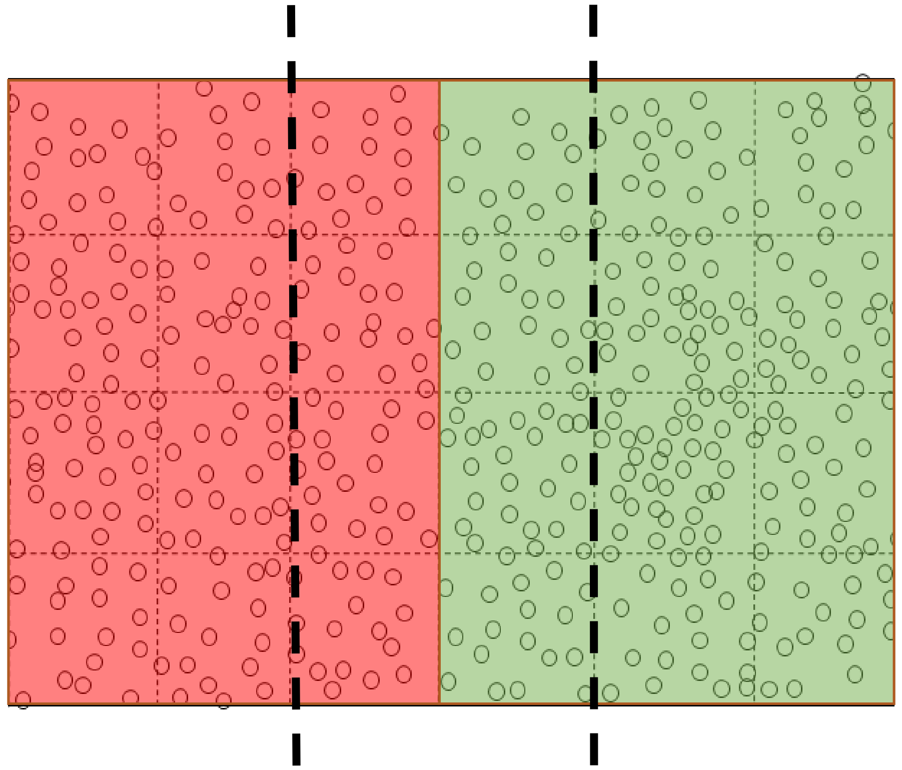


- Allows the simulation to use more particles
- Reduces local and global memory footprint
- Reduces the load on each CPU core



Halo Exchange





- Identify neighbouring particles in another process or particles moved from another process

- Transfer only the data of all potential neighbours

- Use a **halo** system for more efficiency[3]

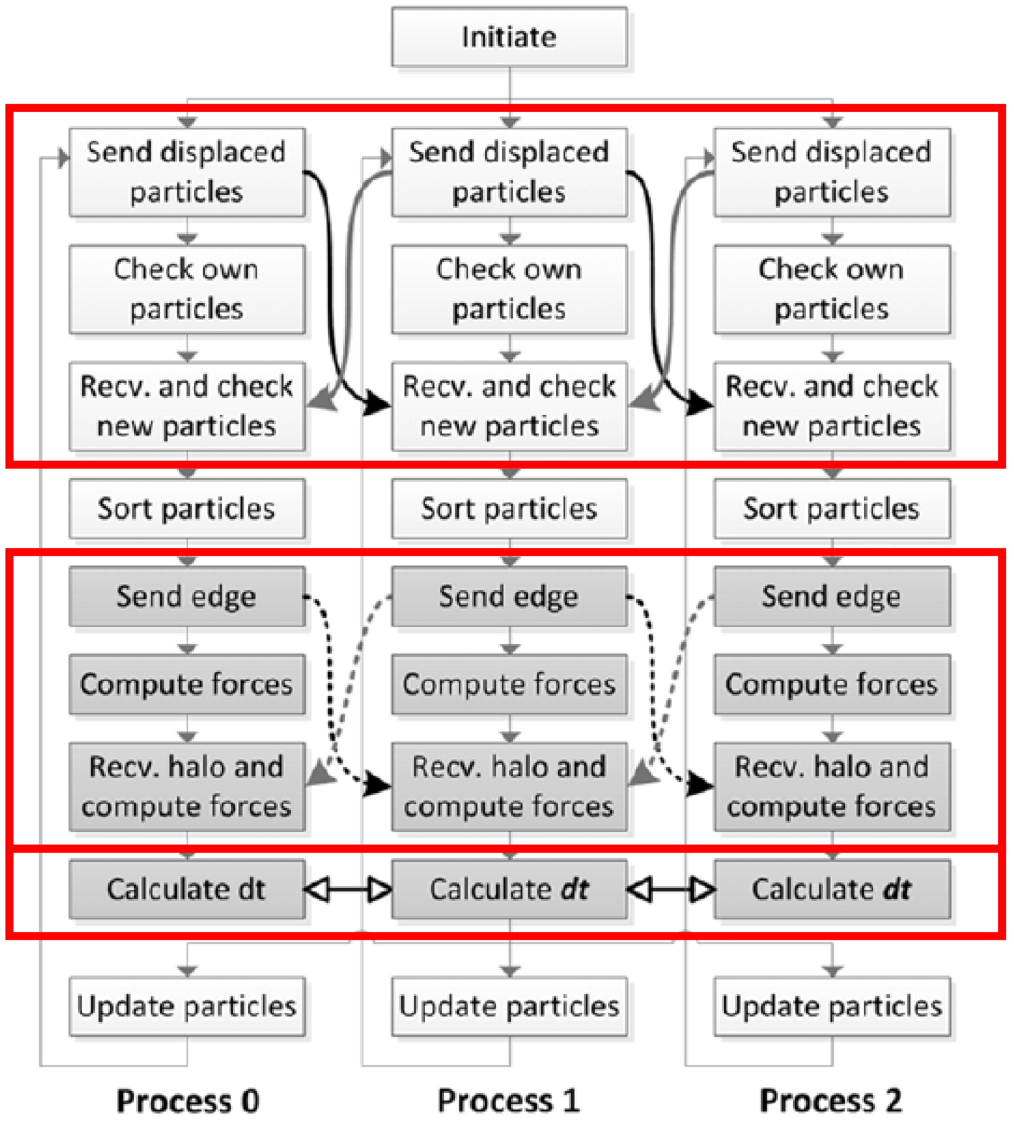
- Only data from the neighbouring slice (distance 2h) are transferred

- Edge particles form the **halo** of the subdomain

- Similar procedure on every subdomain border

Asynchronous Communications

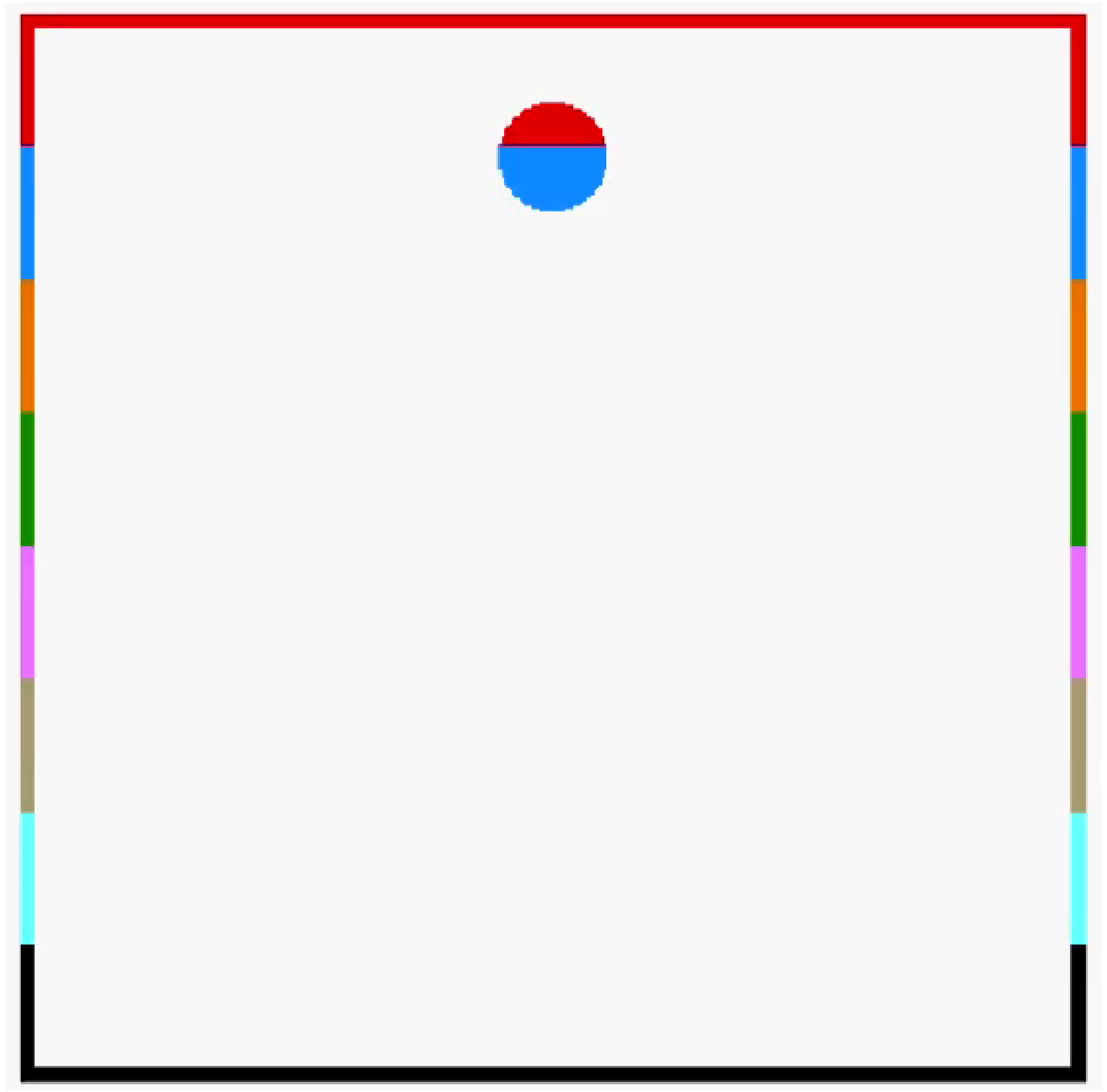
- Objective: Minimise waiting time for data transfer
- Neighbour list of interior particles processed while sending data of displaced particles
- Compute forces on interior particles while receiving halo data
- Processes synchronise when calculating the time step



(Dominguez et al. 2013)[2]

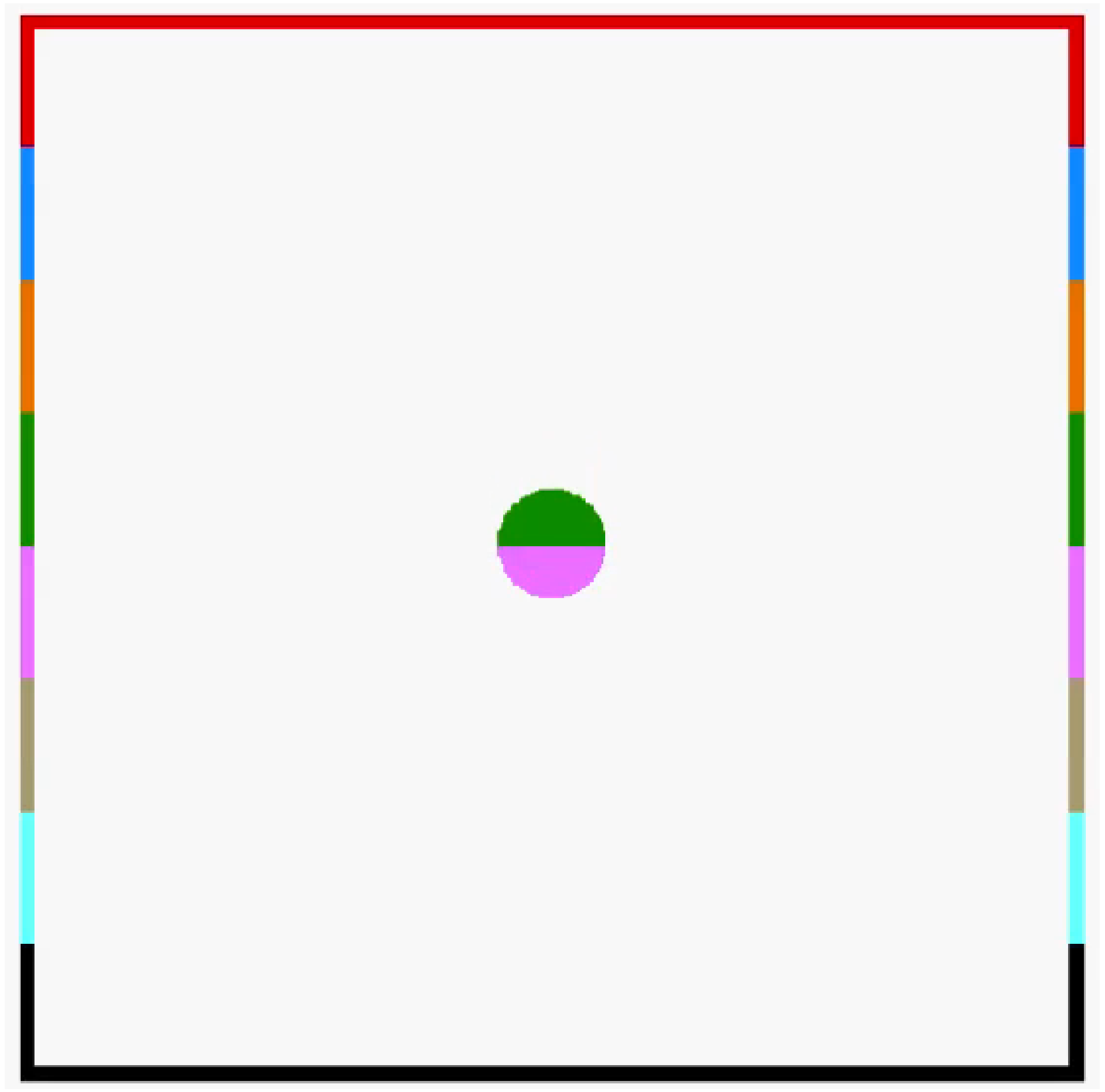
Results

- Execution for 8 processes
- Results identical to single-node DualSPHysics
- Results independent of the number of processes
- Portability: Code operates for both Windows and Linux in different processor architectures



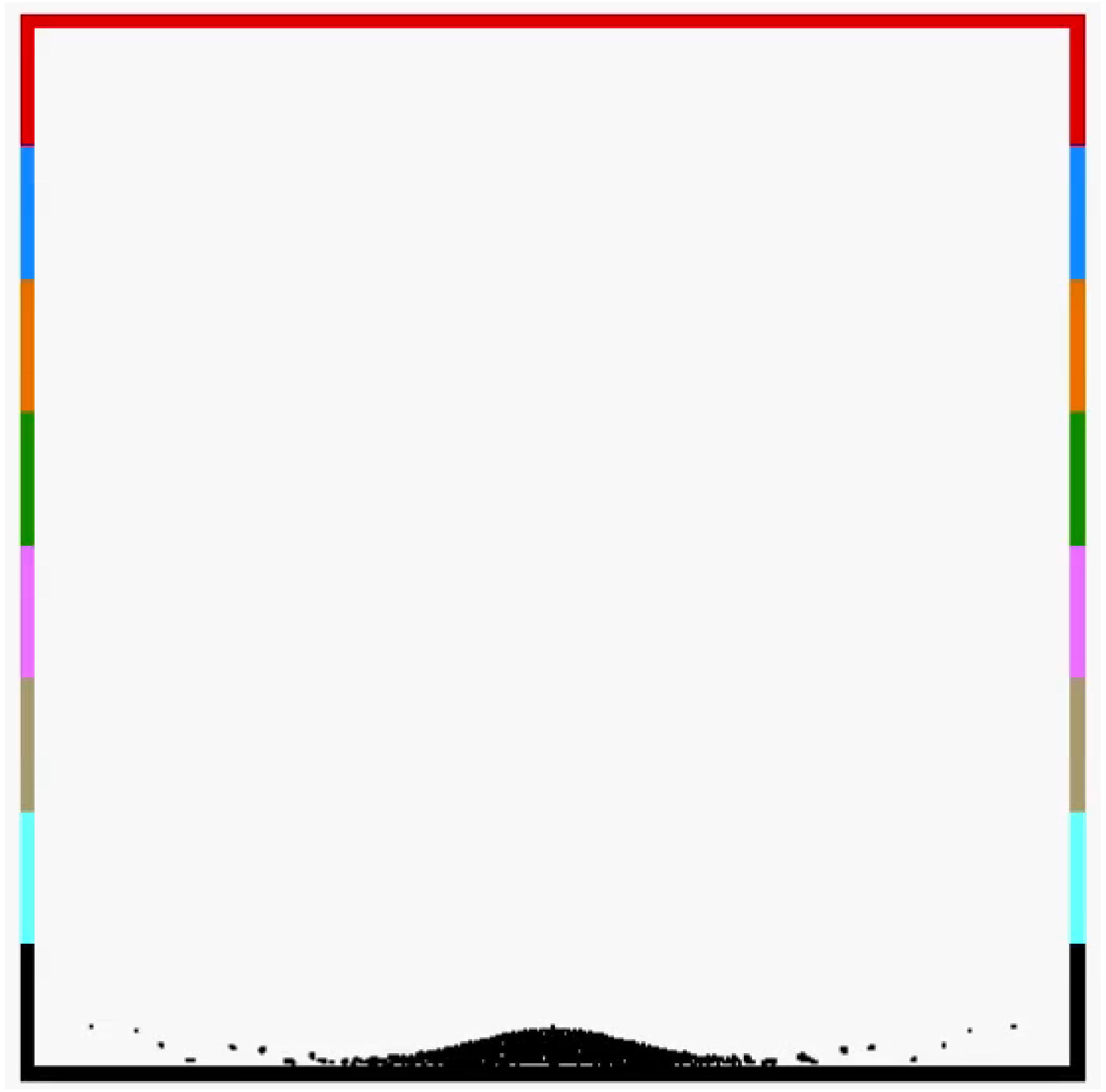
Results

- Execution for 8 processes
- Results identical to single-node DualSPHysics
- Results independent of the number of processes
- Portability: Code operates for both Windows and Linux in different processor architectures



Results

- Execution for 8 processes
- Results identical to single-node DualSPHysics
- Results independent of the number of processes
- Portability: Code operates for both Windows and Linux in different processor architectures

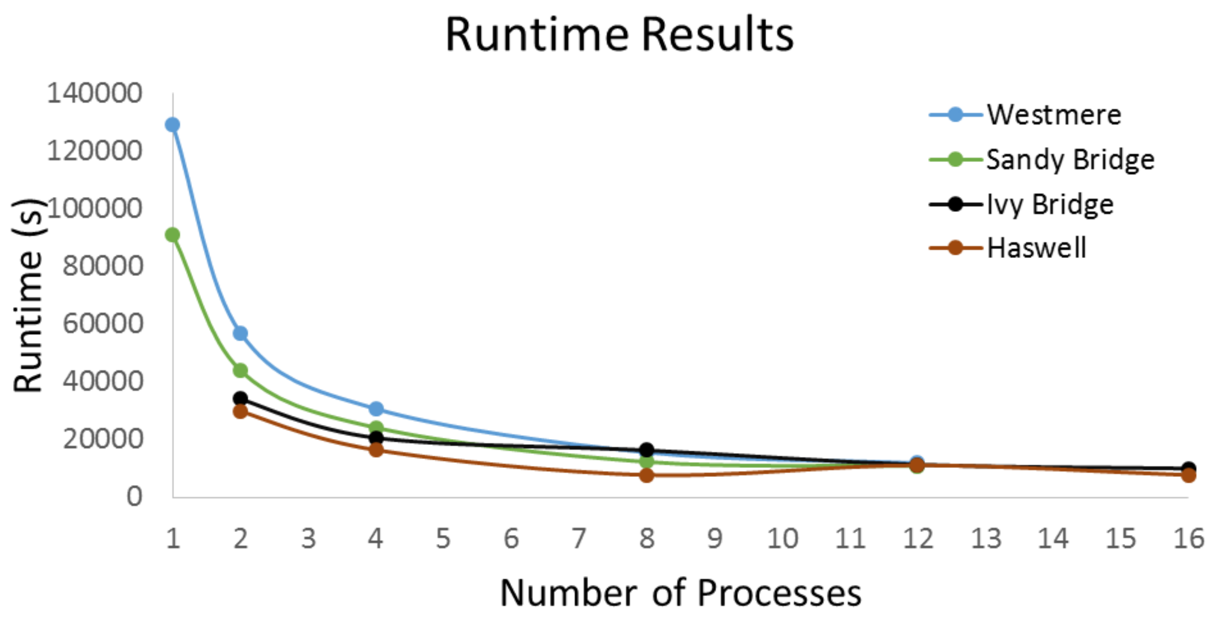


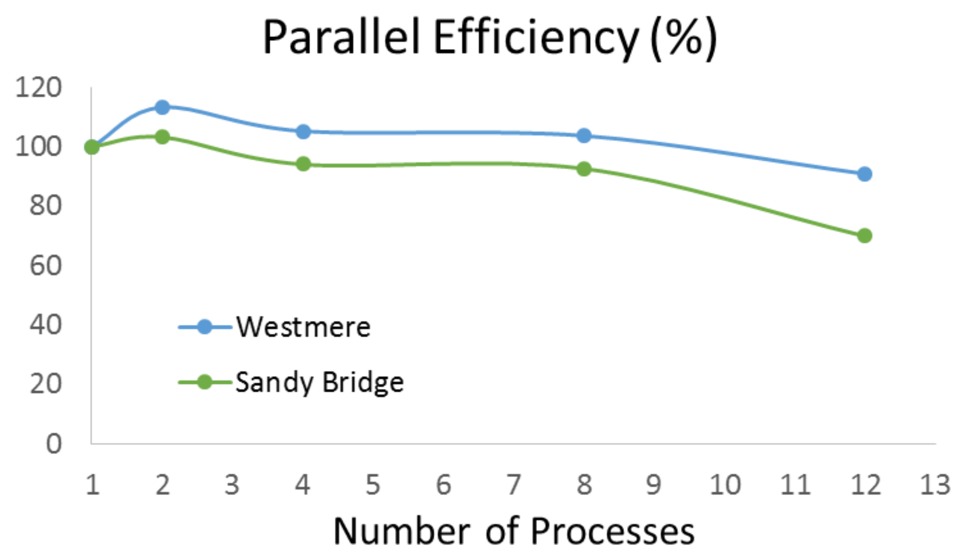
Runtime Results (small scale)

- Local execution for 1-16 processes
  - Westmere: Xeon X5650 - 2.66GHz (2x6-core)
  - Sandy Bridge: Xeon E5-2640 - 2.5GHz (2x6-core)
  - Ivy Bridge: Xeon E5-2650 v2 – 2.6GHz (2x8-core)
  - Haswell: Xeon E5-2690 v3 – 2.6GHz (2x12-core)

- Still Water case for 700,000 particles

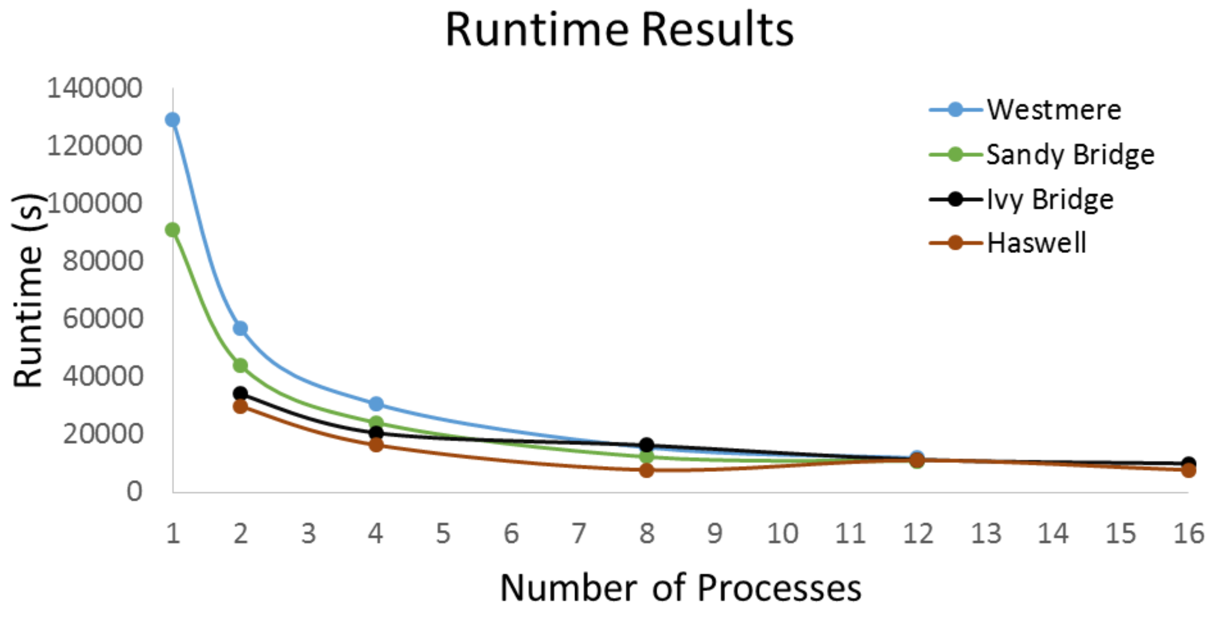
- Parallel Efficiency $$E_p = \frac{T_p}{pT_1} 100\%$$

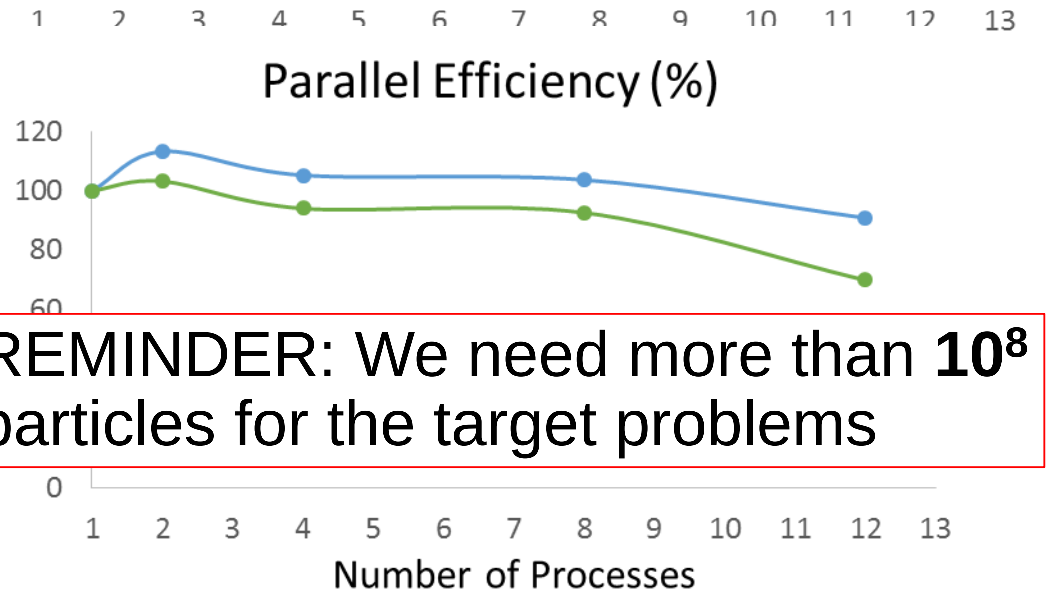




Scalability (small scale)

- Code can be further optimised
- Parallel Efficiency $$E_p = \frac{T_p}{pT_1} 100\%$$
- Scalability issues do not allow efficient computation with ~100 processes
- 1D decomposition not scalable
- No load balancing

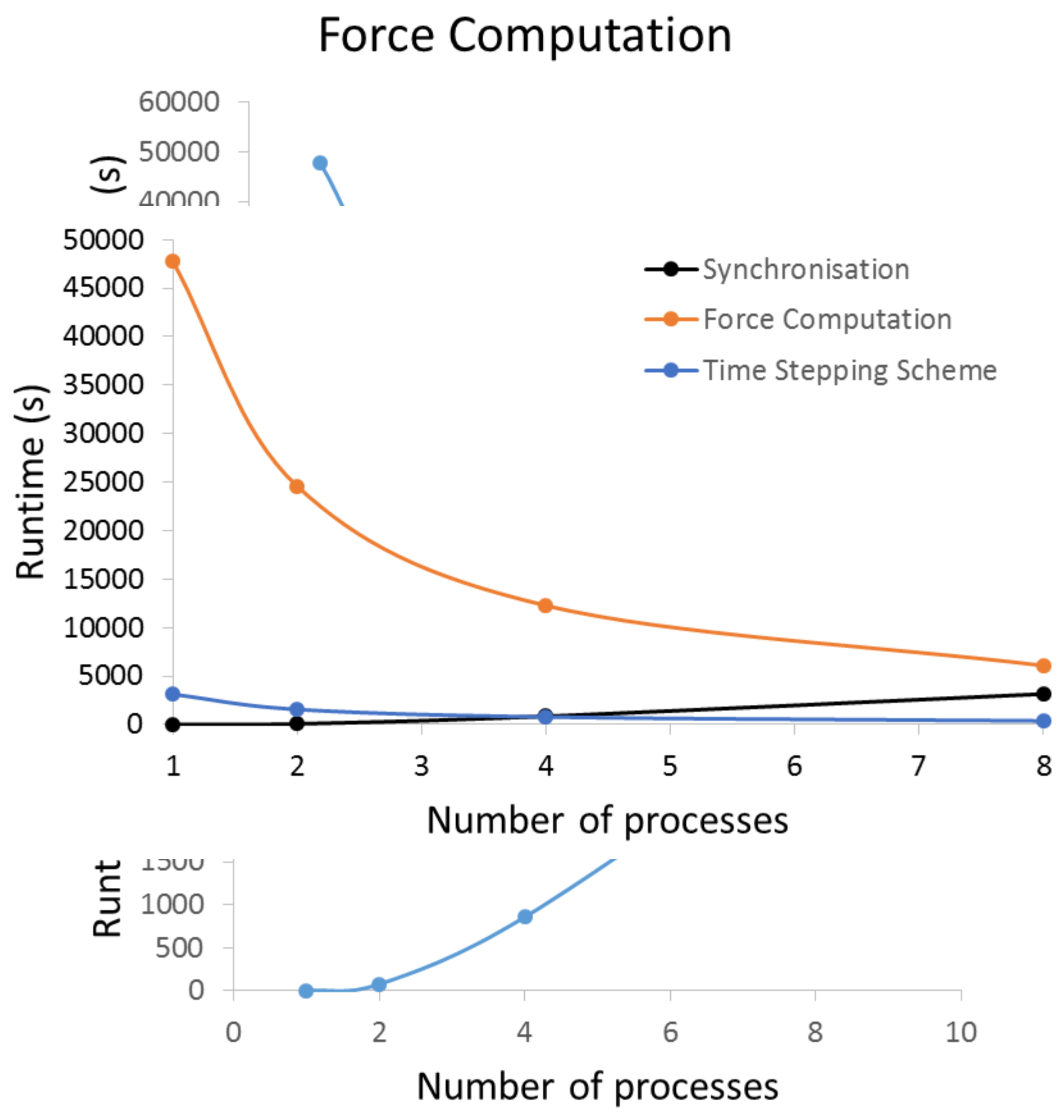




REMINDER: We need more than **10^8** particles for the target problems

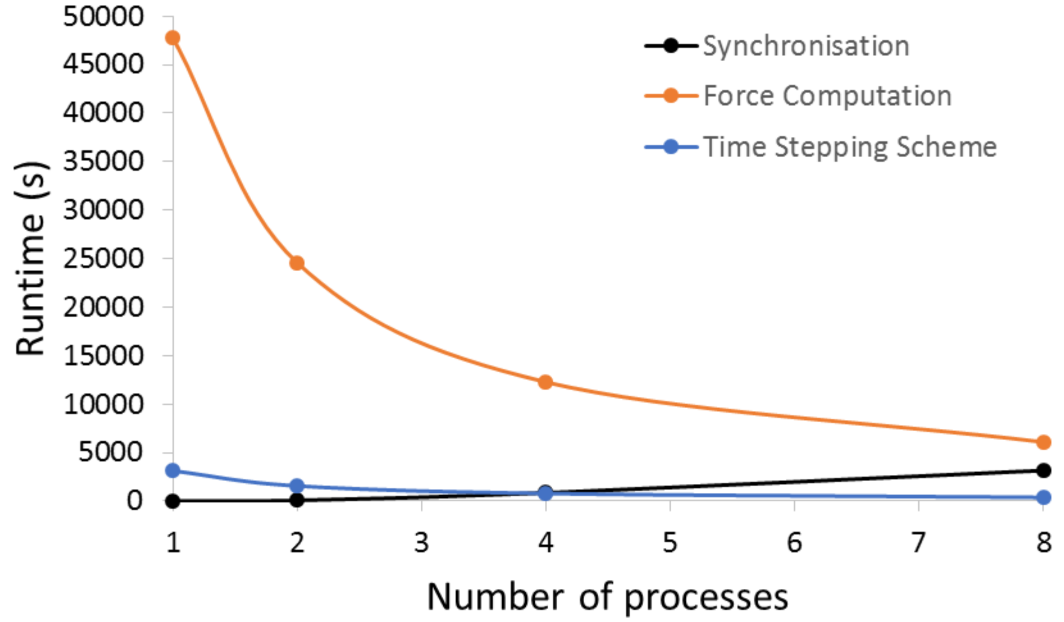
Runtime Results (small scale)

- Local execution for 8 processes
  - Intel Xeon E5507 at 2.27GHz
- Still Water case for 160,000 particles
- Synchronisation at the end of the time step slows the computation
- Current implementation : *MPI_Allreduce*



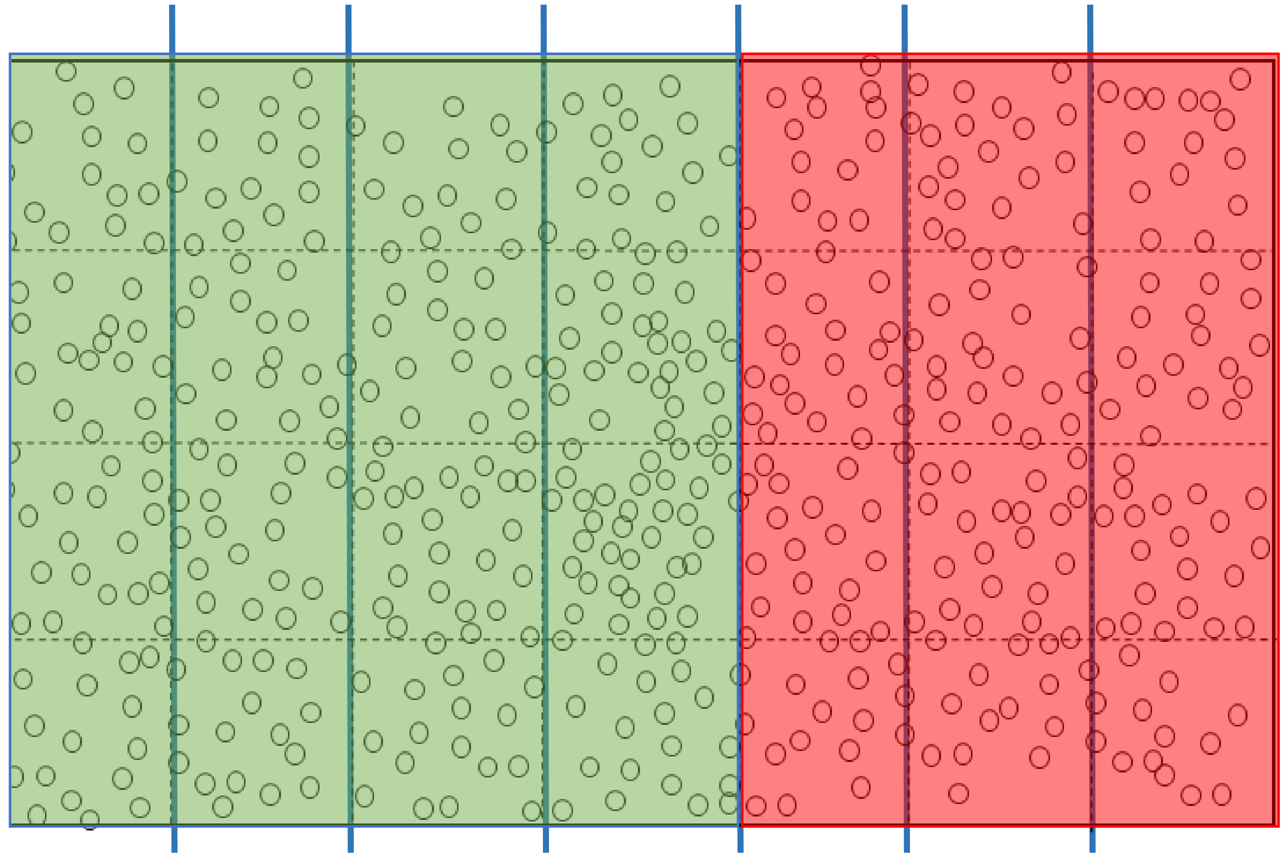
Runtime Results (small scale)

- Local execution for 8 processes
  - Intel Xeon E5507 at 2.27GHz
- Still Water case for 160,000 particles
- Synchronisation at the end of the time step slows the computation
- Current implementation : *MPI_Allreduce*



Dynamic Load Balancing

- Processes do not have the same workload (number of particles, inter-particle forces)
- Dynamic simulations – workload of each process changes constantly
- Options:
  1. Same number of particles
  2. Same execution time
- Option 1 is simpler to enforce
- Option 2 has higher potential but difficult to enforce



Dynamic Load Balancing

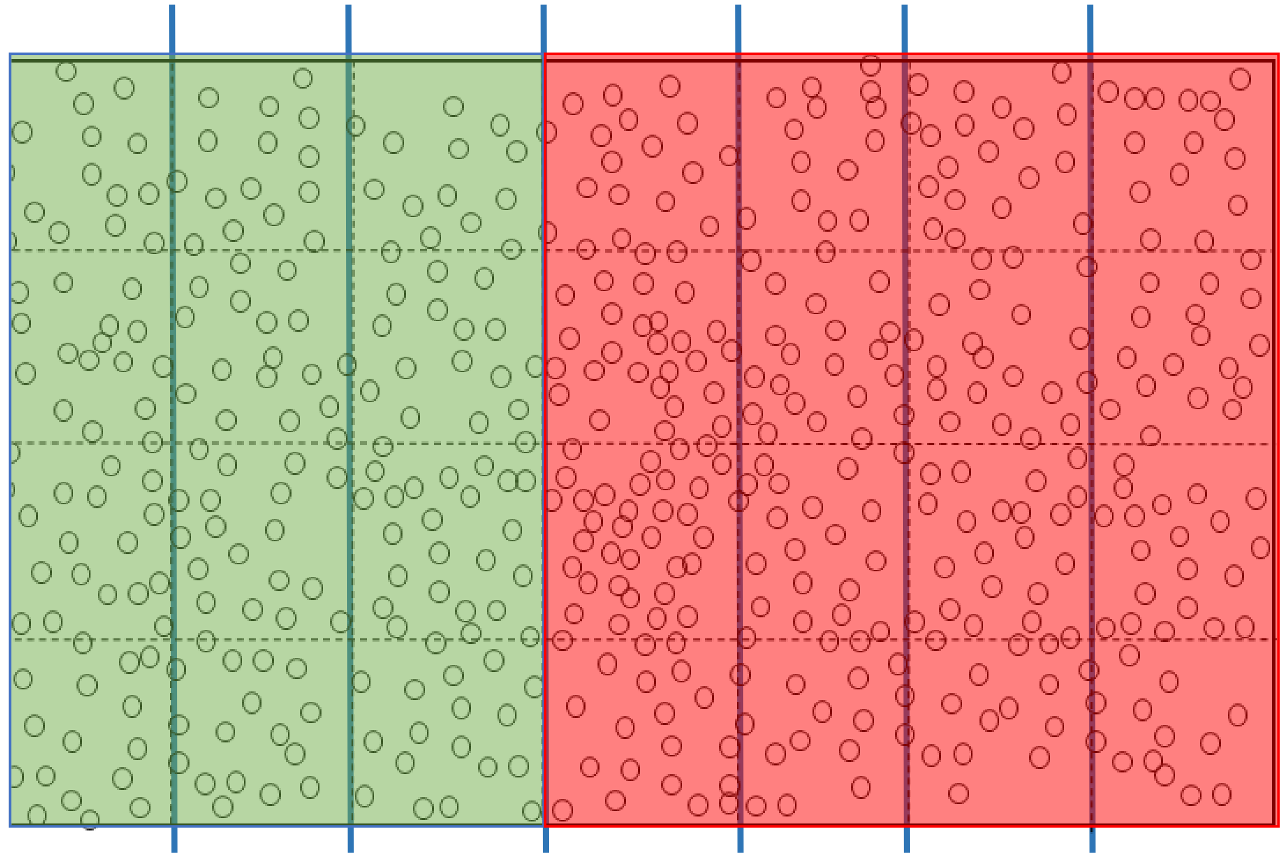
- Processes do not have the same workload (number of particles, inter-particle forces)

- Dynamic simulations – workload of each process changes constantly

- Options:
  1. Same number of particles
  2. Same execution time

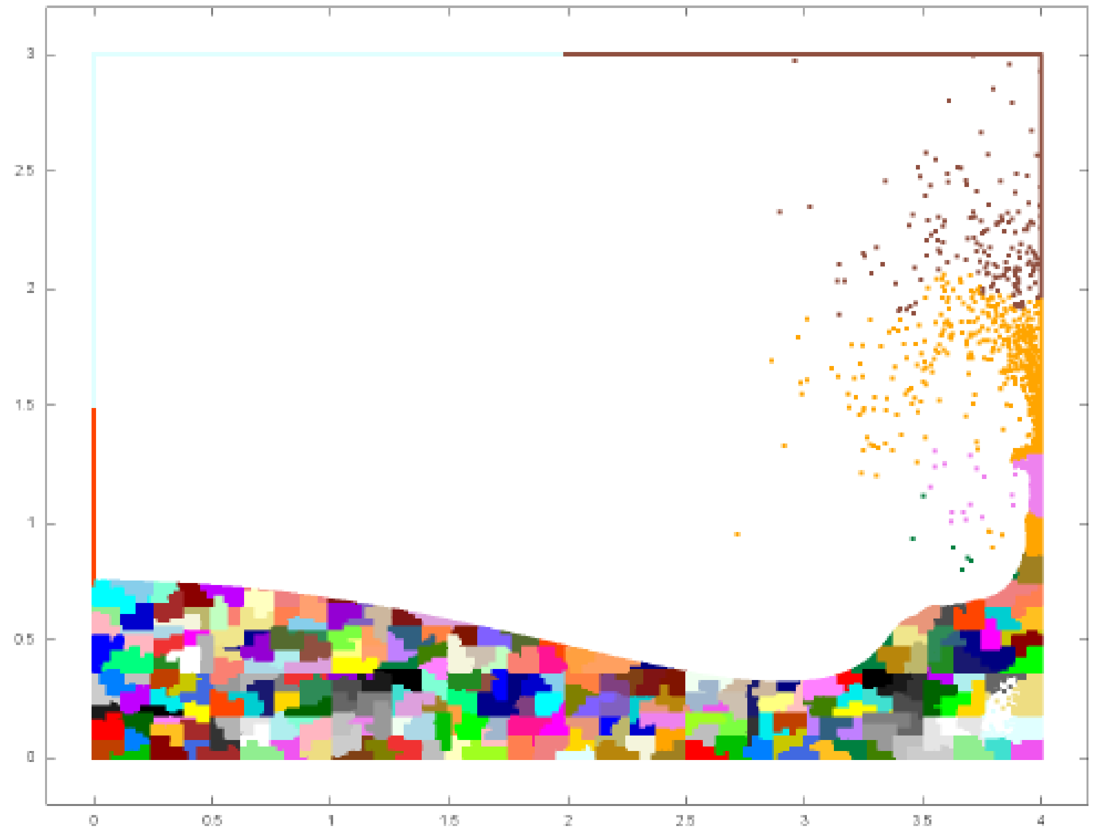
- Option 1 is simpler to enforce

- Option 2 has higher potential but difficult to enforce



The Zoltan Library

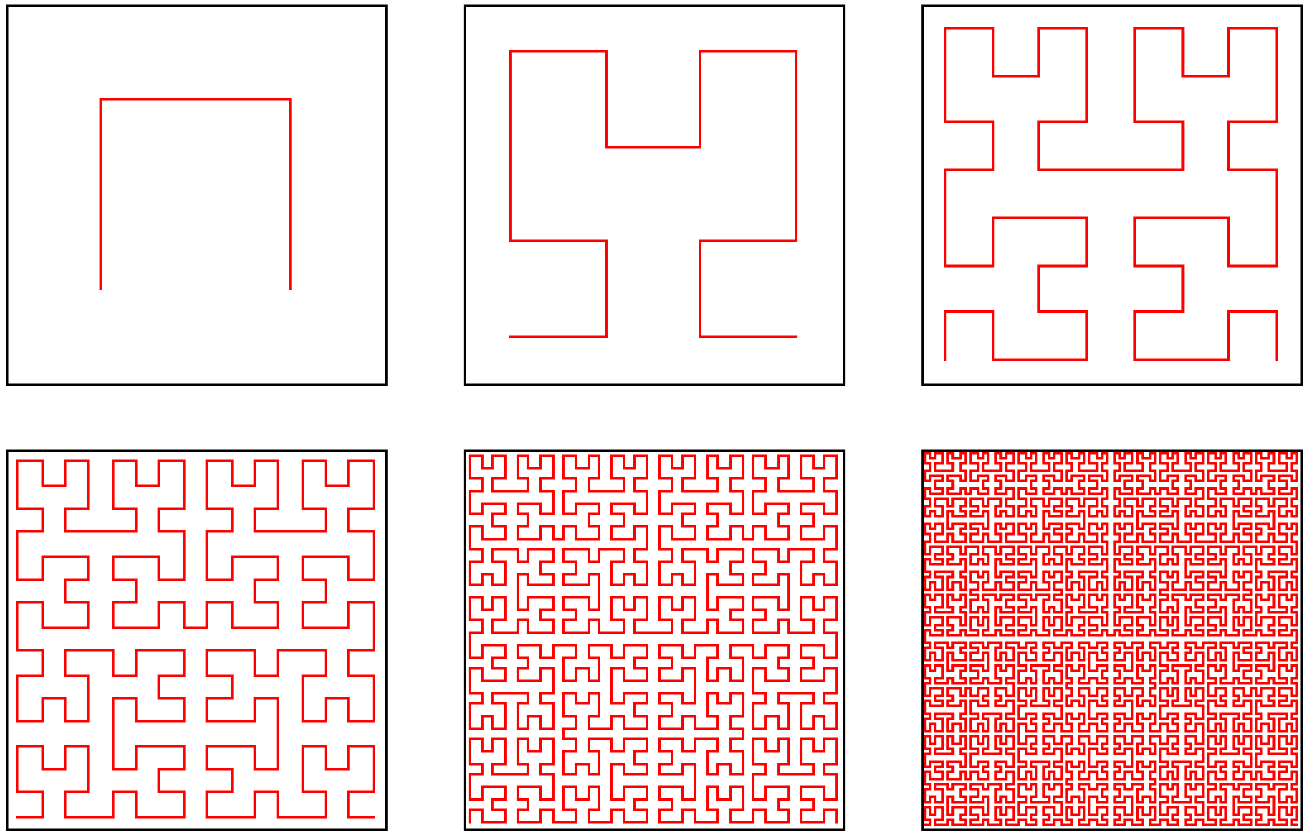
- Use of the Zoltan data management library[4]
- Library for the development and optimization of parallel, unstructured and adaptive codes
- Scalable up to 10^6 cores[4]
- Includes a suite of spatial decomposition and dynamic load balancing algorithms and an unstructured communication package
- Geometric Decomposition Algorithm:
  Hilbert Space Filling Curve (HSFC)



Dambreak at 1.1s for 256 partitions[5]

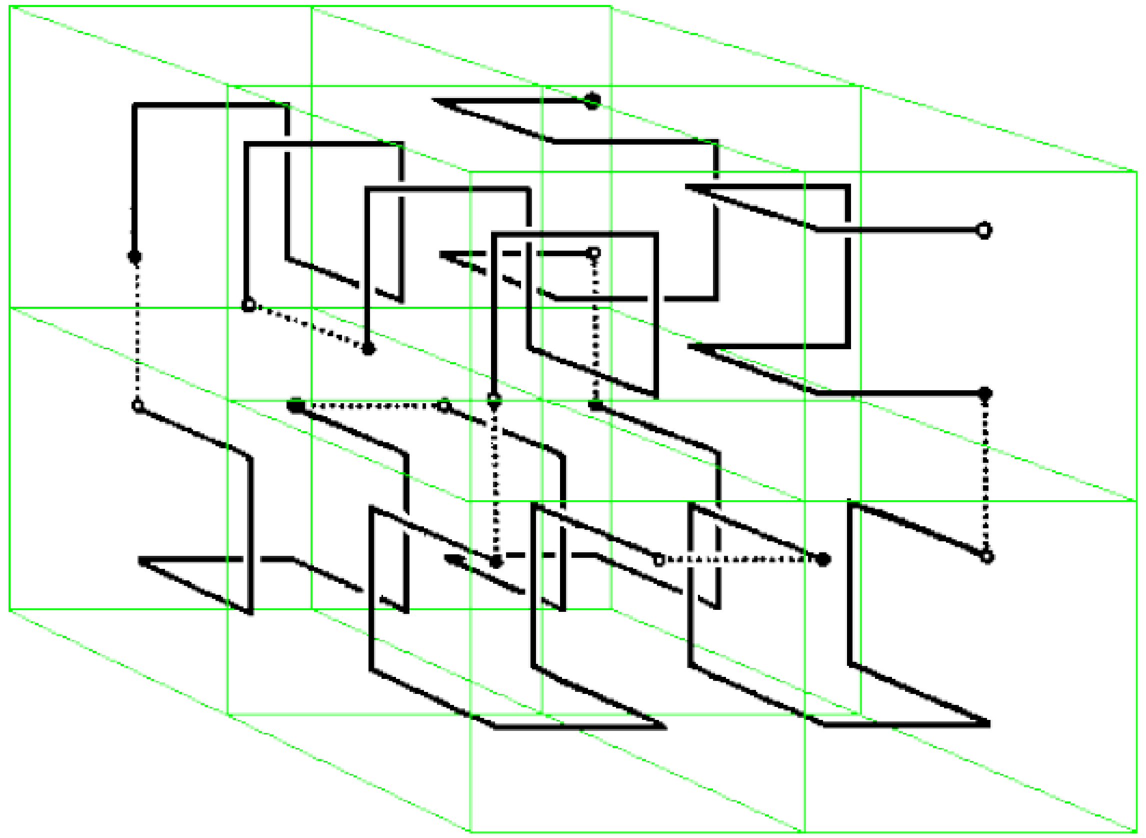
Hilbert Space Filling Curve

- A continuous fractal space-filling curve (containing the entire 2D unit square)
- Maps 2D and 3D points to a 1D curve
- Maintains spatial locality
- Already used for SPH[5]
- Irregular subdomain shapes (increased complexity of data transfer)



Hilbert Space Filling Curve

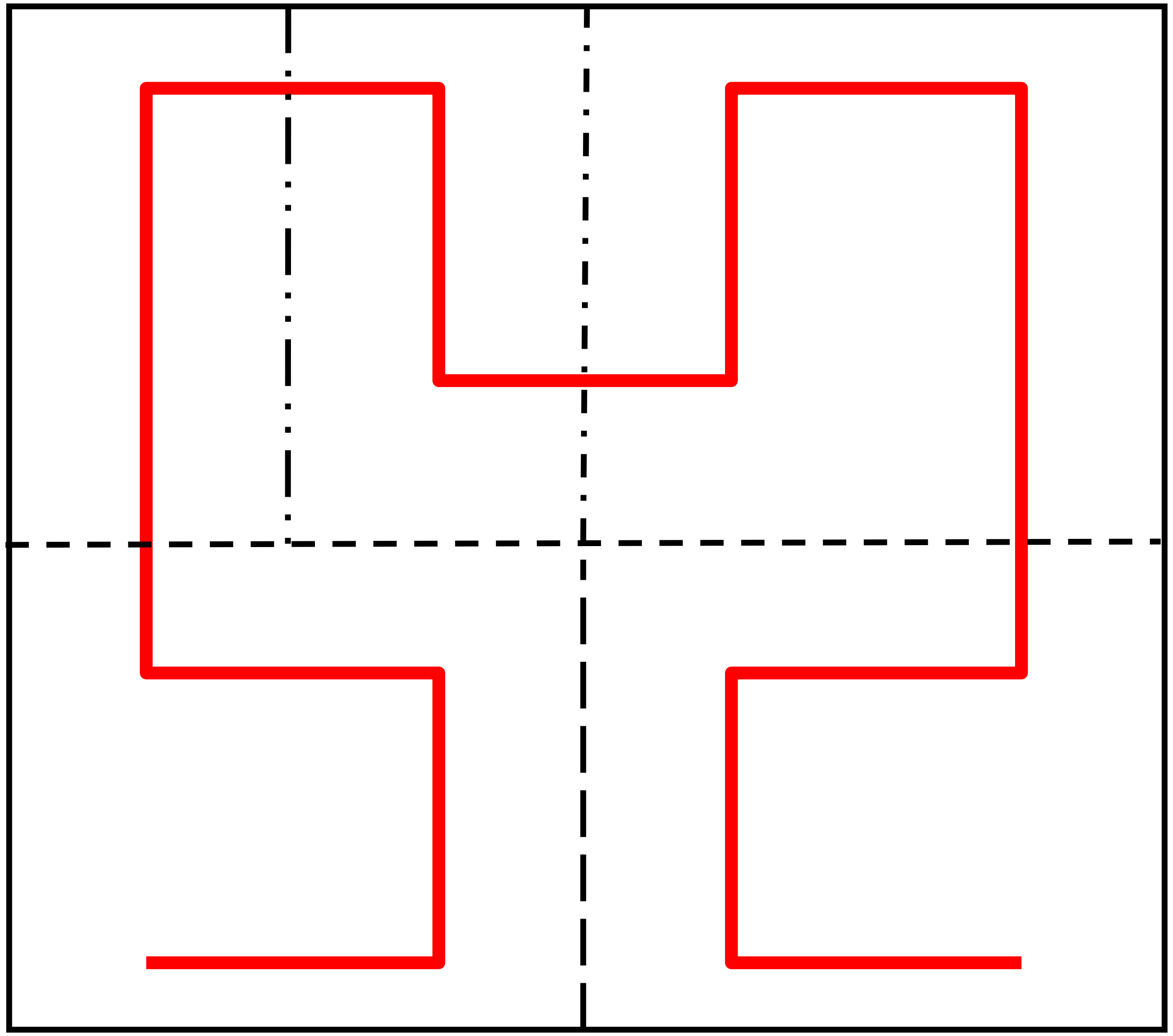
- A continuous fractal space-filling curve (containing the entire 2D unit square)
- Maps 2D and 3D points to a 1D curve
- Maintains spatial locality
- Already used for SPH[5]
- Irregular subdomain shapes (increased complexity of data transfer)



Guo et al. (2015)[7]

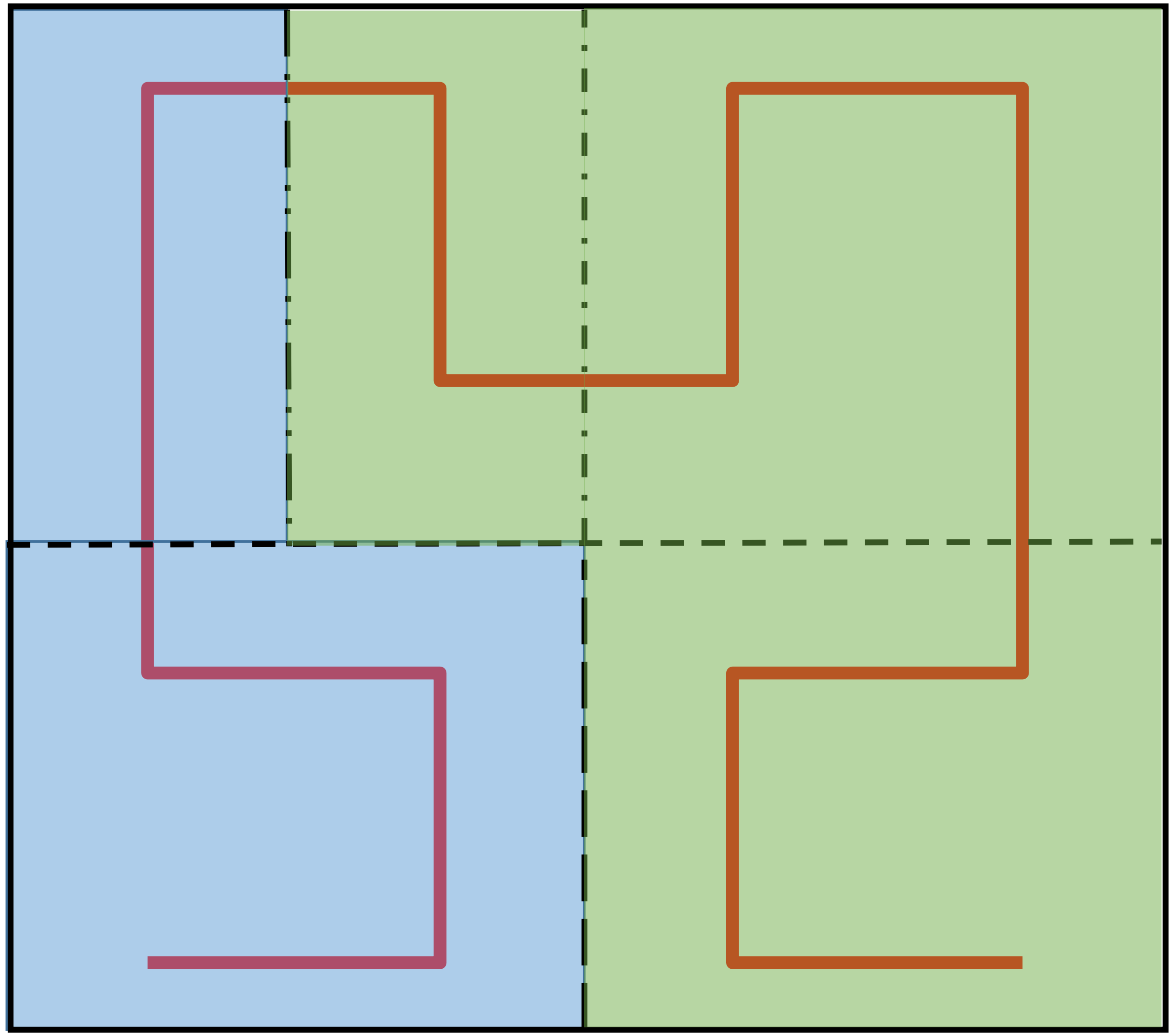
HSFC Algorithm

- HSFC maps cells on a 1D curve into the interval [0,1]
- Divides the curve into N ‘bins’ where N is larger than the amount of processes
- Sums bin weights from starting point, cutting off whenever the desired weight is reached
- Bins containing a cutting off point are further refined until the desired balance is achieved



HSFC Algorithm

- HSFC maps cells on a 1D curve into the interval [0,1]
- Divides the curve into N ‘bins’ where N is larger than the amount of processes
- Sums bin weights from starting point, cutting off whenever the desired weight is reached
- Bins containing a cutting off point are further refined until the desired balance is achieved



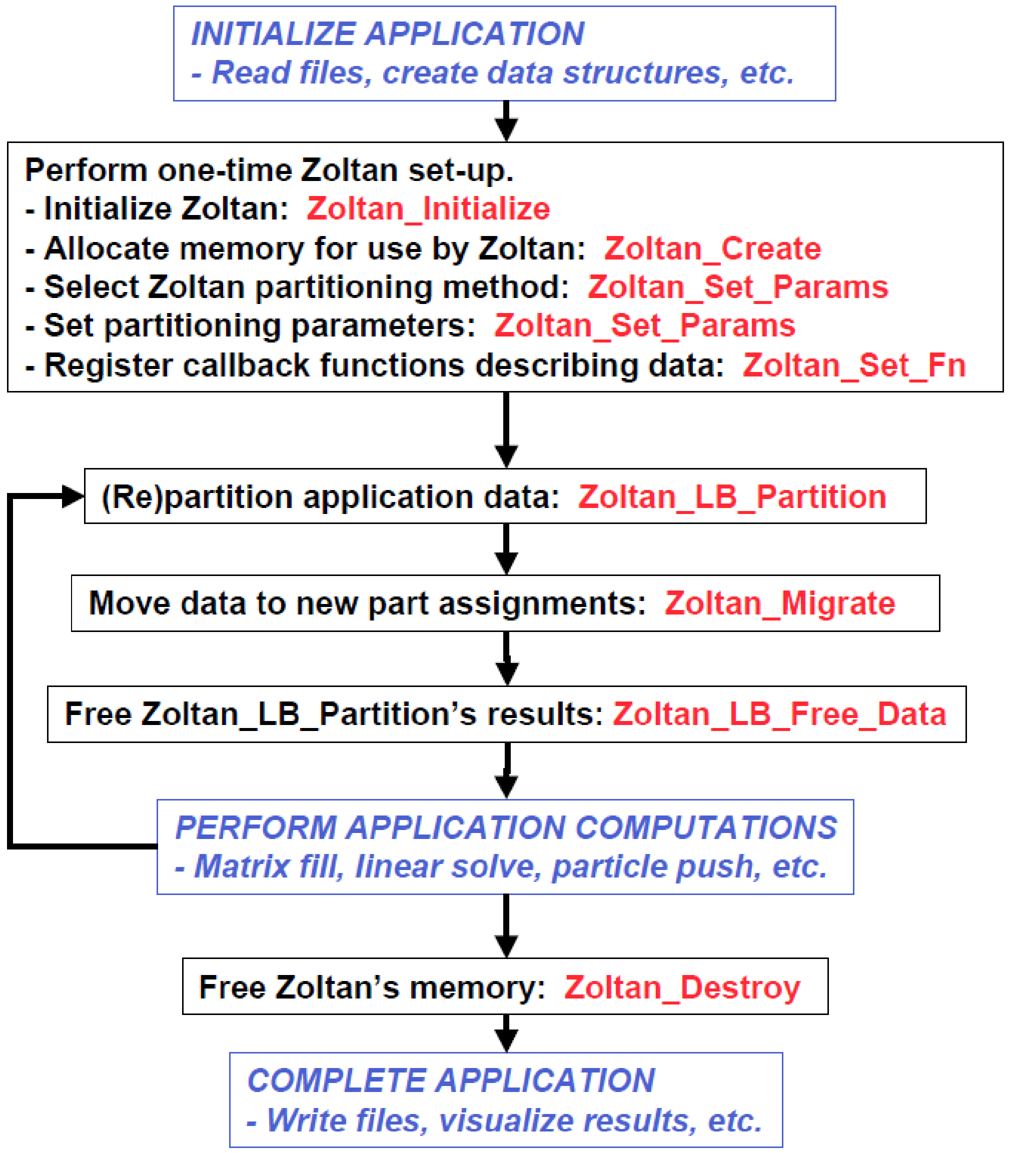
Using Zoltan in DualSPHysics

- Domain Decomposition and Load Balancing through Zoltan

- Main Partitioning Parameter: Cells
  - Significantly smaller number than particles
  - Allow for load balancing
  - Position does not change

- Load Balancing through Cell Weights
  - Based on particle number[5] (Current)
  - Based on execution time

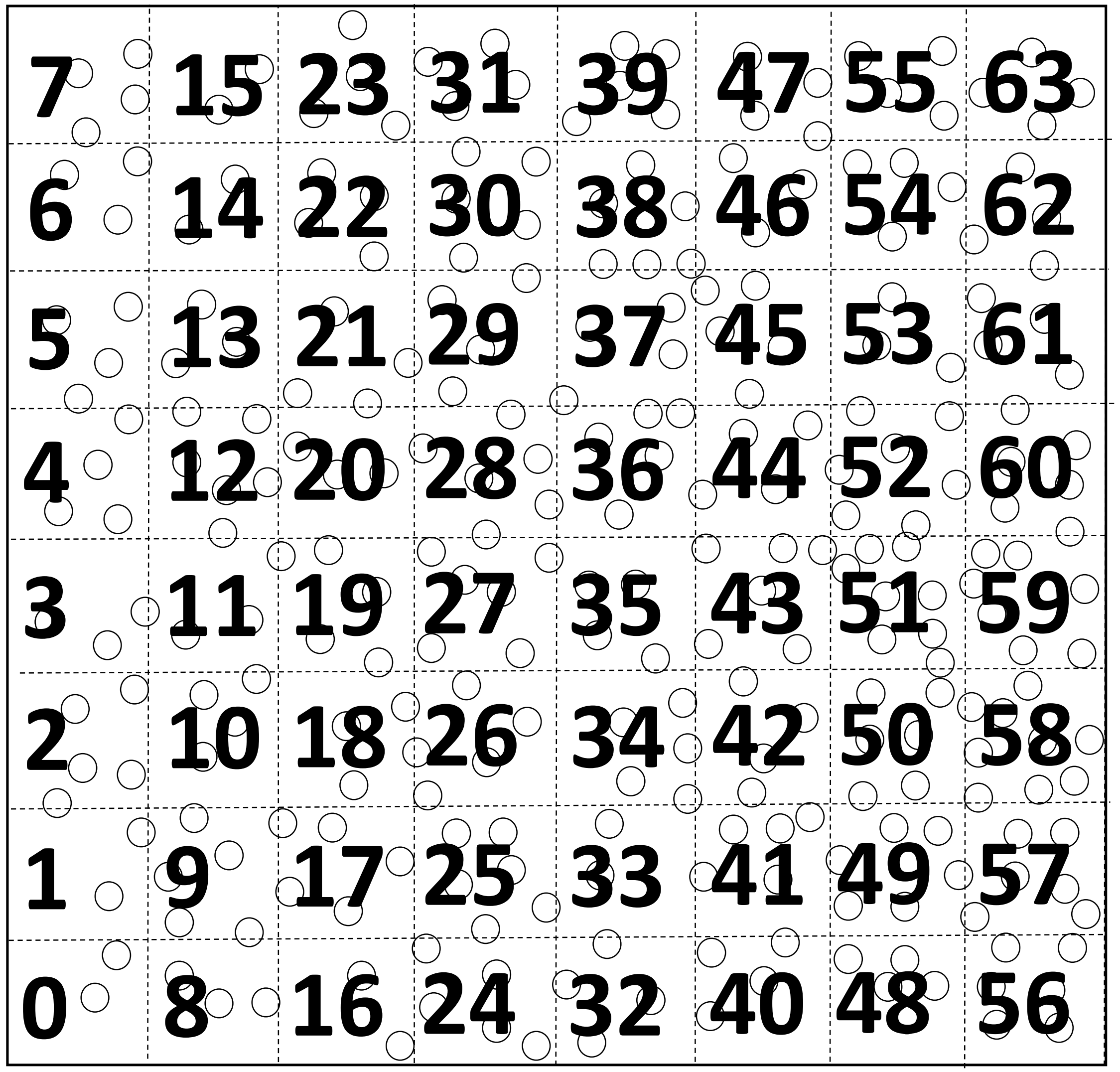
- Automatic migration through Zoltan_Migrate
  - Low complexity of data transferred


INITIALIZE APPLICATION
- Read files, create data structures, etc.
Perform one-time Zoltan set-up.
- Initialize Zoltan: Zoltan_Initialize
- Allocate memory for use by Zoltan: Zoltan_Create
- Select Zoltan partitioning method: Zoltan_Set_Params
- Set partitioning parameters: Zoltan_Set_Params
- Register callback functions describing data: Zoltan_Set_Fn
(Re)partition application data: Zoltan_LB_Partition
Move data to new part assignments: Zoltan_Migrate
Free Zoltan_LB_Partition's results: Zoltan_LB_Free_Data
PERFORM APPLICATION COMPUTATIONS
- Matrix fill, linear solve, particle push, etc.
Free Zoltan's memory: Zoltan_Destroy
COMPLETE APPLICATION
- Write files, visualize results, etc.

Devine et al. (2009)[4]

Using Zoltan in DualSPHysics

- New arrays created:
  - Global Cell ID
  - Local Cell ID
  - Cell Coordinates
  - Cell Weights
- Each process only holds local data
- Example: Domain divided in 64 cells containing 285 particles
- Initial domain split by 1D decomposition (Slices)



Global Cell ID

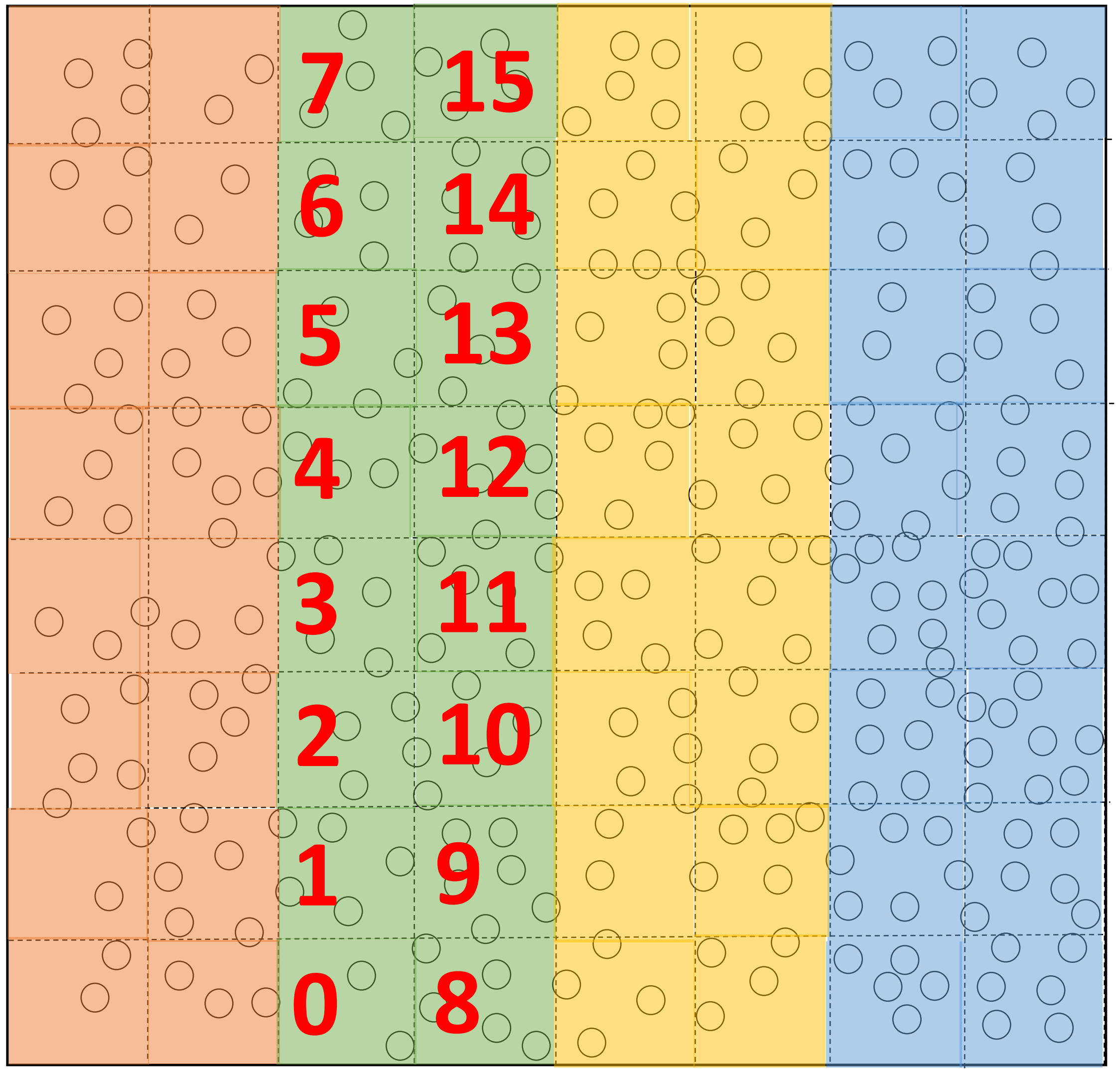
Using Zoltan in DualSPHysics

- New arrays created:
  - Global Cell ID
  - Local Cell ID
  - Cell Coordinates
  - Cell Weights

- Each process only holds local data

- Example: Domain divided in 64 cells containing 285 particles

- Initial domain split by 1D decomposition (Slices)



Local Cell ID

Using Zoltan in DualSPHysics

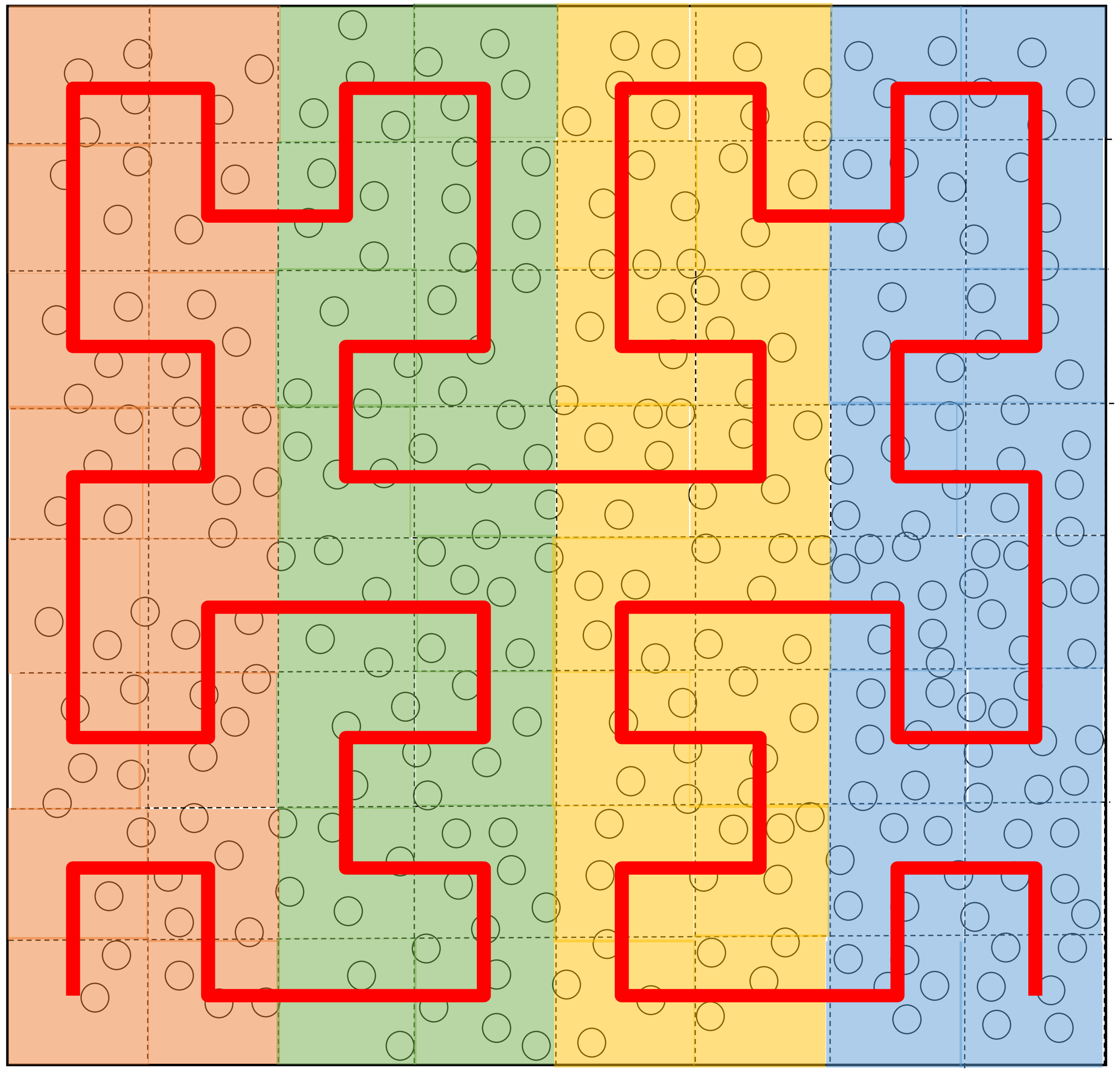
- Cell weights[5]: $w_C = \dfrac{N_{pc}}{N_{pt}}$

- Data is sent to Zoltan

- HSFC algorithm is applied

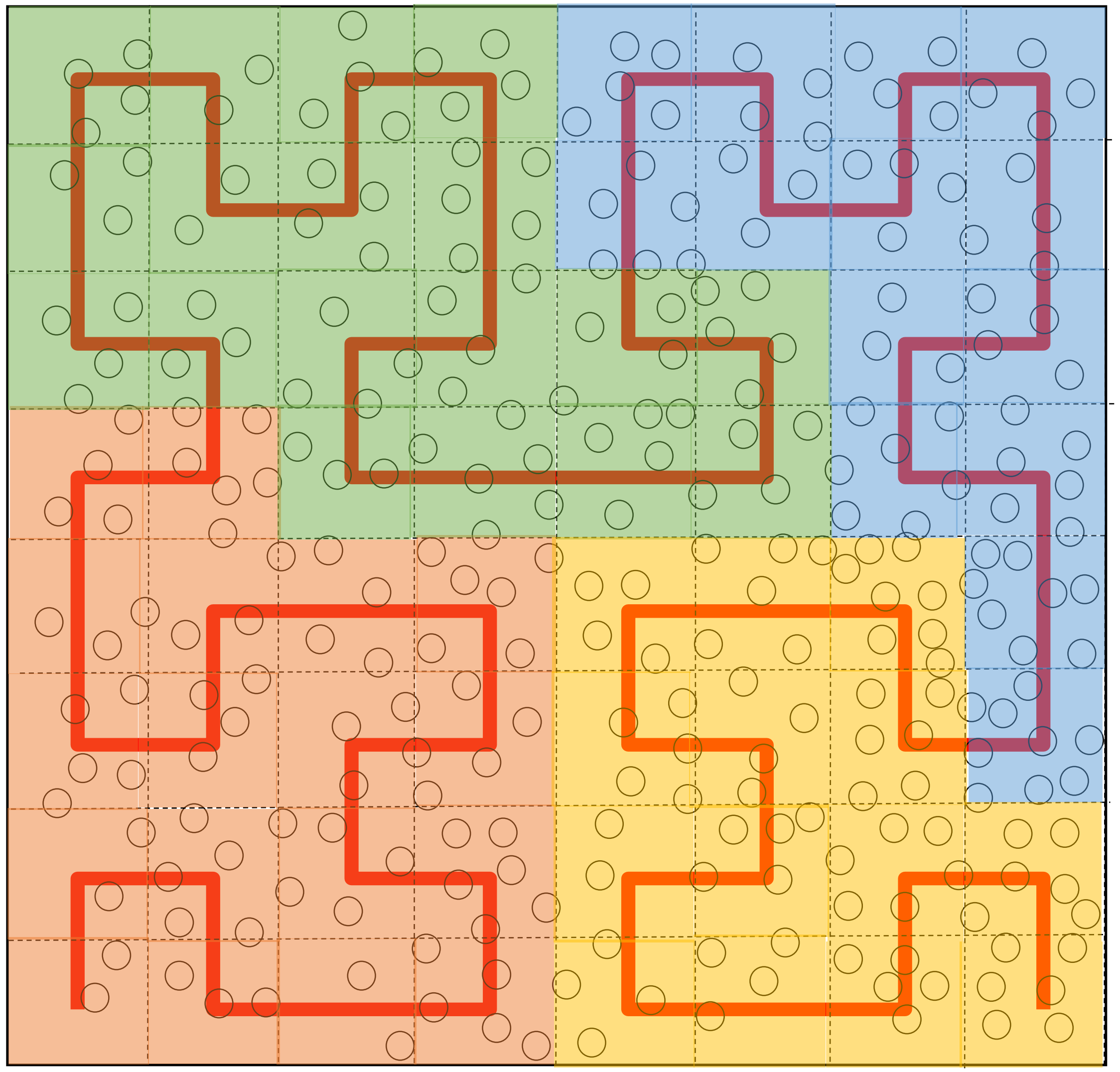
- Zoltan Output:
  - Global Cell IDs of imported cells
  - Global Cell IDs of exported cells
  - Destination process

- Cell data automatically migrated using AUTO_MIGRATE option



Using Zoltan in DualSPHysics

- GlobalCellID is updated:
  - Exported cells removed
  - Imported cells added
- Particles are also imported and exported
- Data reordered creating new cell-linked neighbour list
- LocalCellID is updated
- Algorithm applied only when imbalance exceeds 20%



Particle Mapping

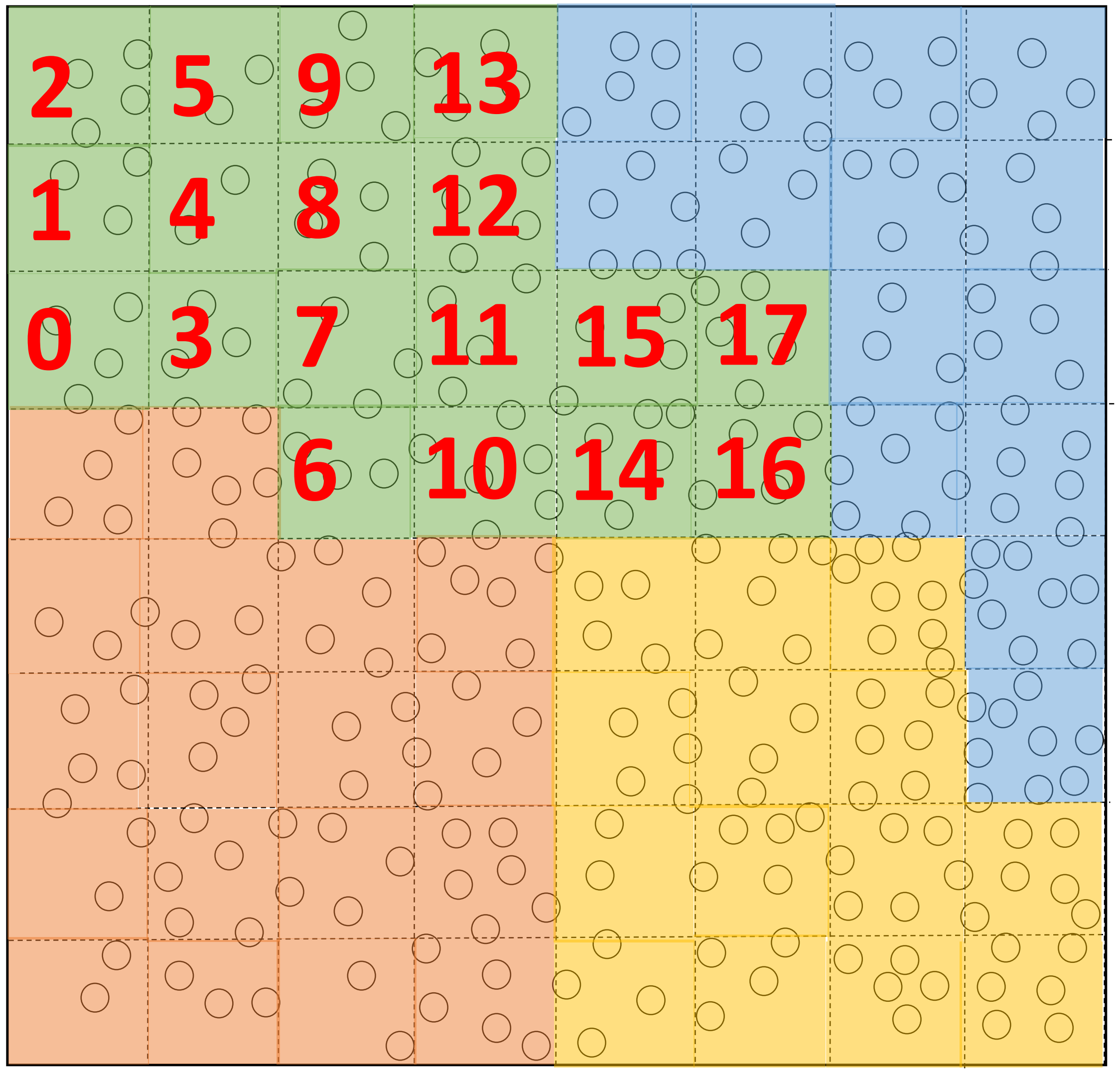
- Connection between cells and particles needed
- Existing DualSPhysics array: **CellPart**
- CellPart can be easily mapped on LocalCellID
- LocalCellID acts as intermediary between CellPart and GlobalCellID

If N_c number of local cells

$$\begin{tikzcd}
\begin{array}{c}\text{CellPart}\\(2N_c+5)\end{array} \arrow[r, Leftrightarrow] & \begin{array}{c}\text{LocalCellID}\\(N_c)\end{array} \arrow[r, Leftrightarrow] & \begin{array}{c}\text{GlobalCellID}\\(N_c)\end{array}
\end{tikzcd}$$

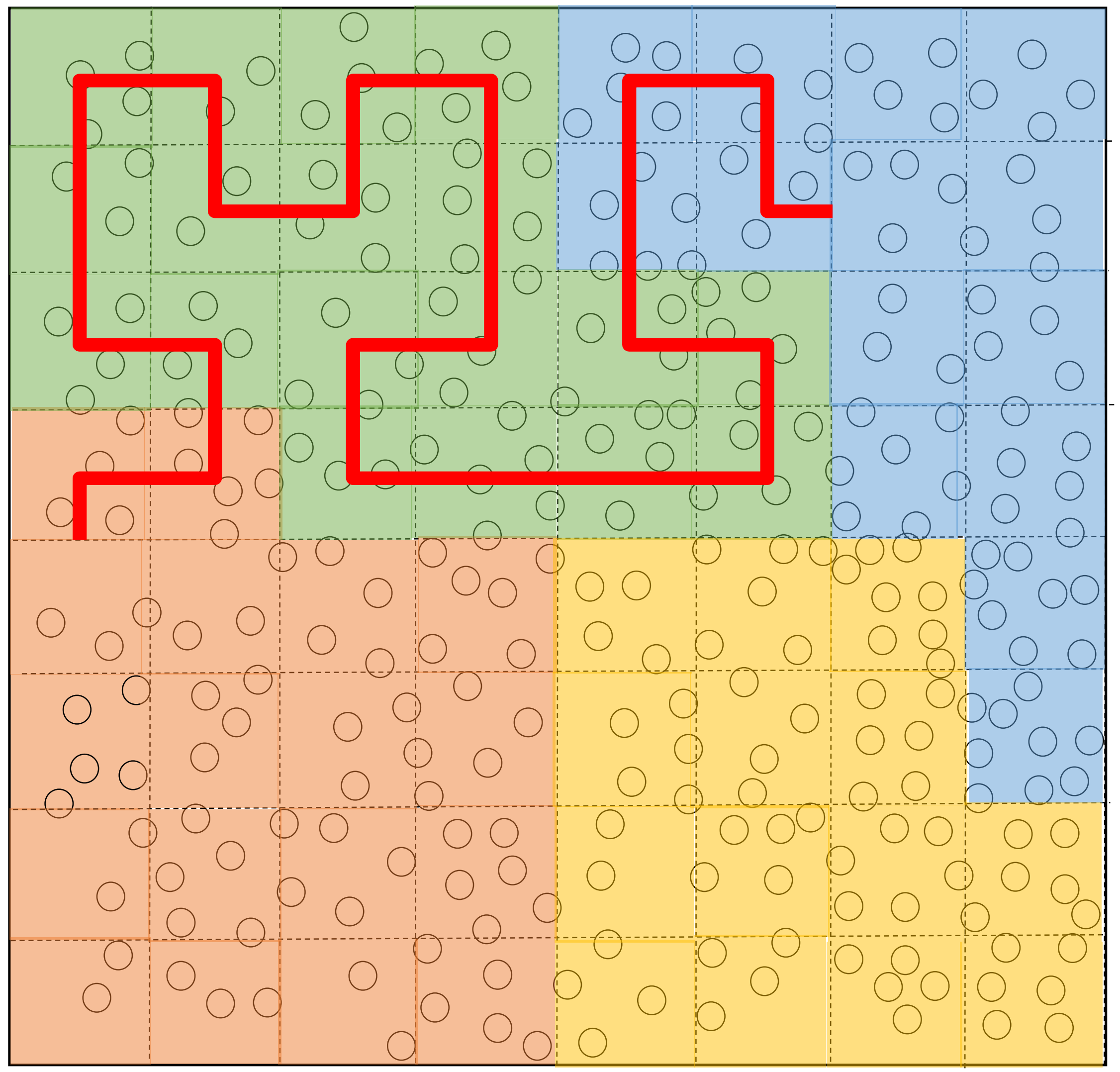
Particle Reordering

- Particles need reordering to maintain local spatial locality
- Currently, particle data reordered using single node algorithm
- Same for LocalCellID – allows mapping to Cellpart
- GlobalCellID is constant
- Better option: reorder along HSFC path[5]



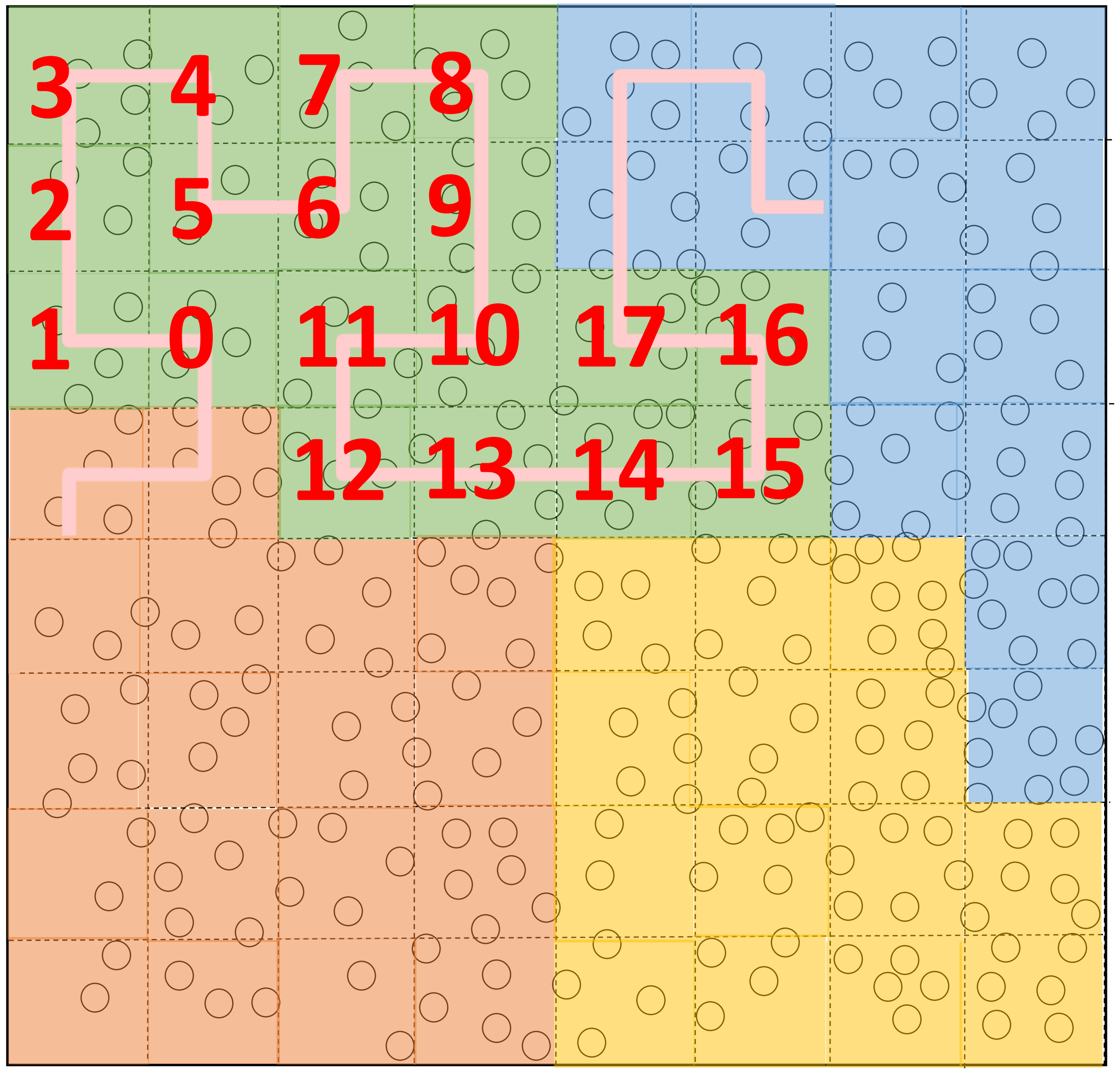
Particle Reordering

- Particles need reordering to maintain local spatial locality
- Currently, particle data reordered using single node algorithm
- Same for LocalCellID – allows mapping to Cellpart
- GlobalCellID is constant
- Better option: reorder along HSFC path[5]

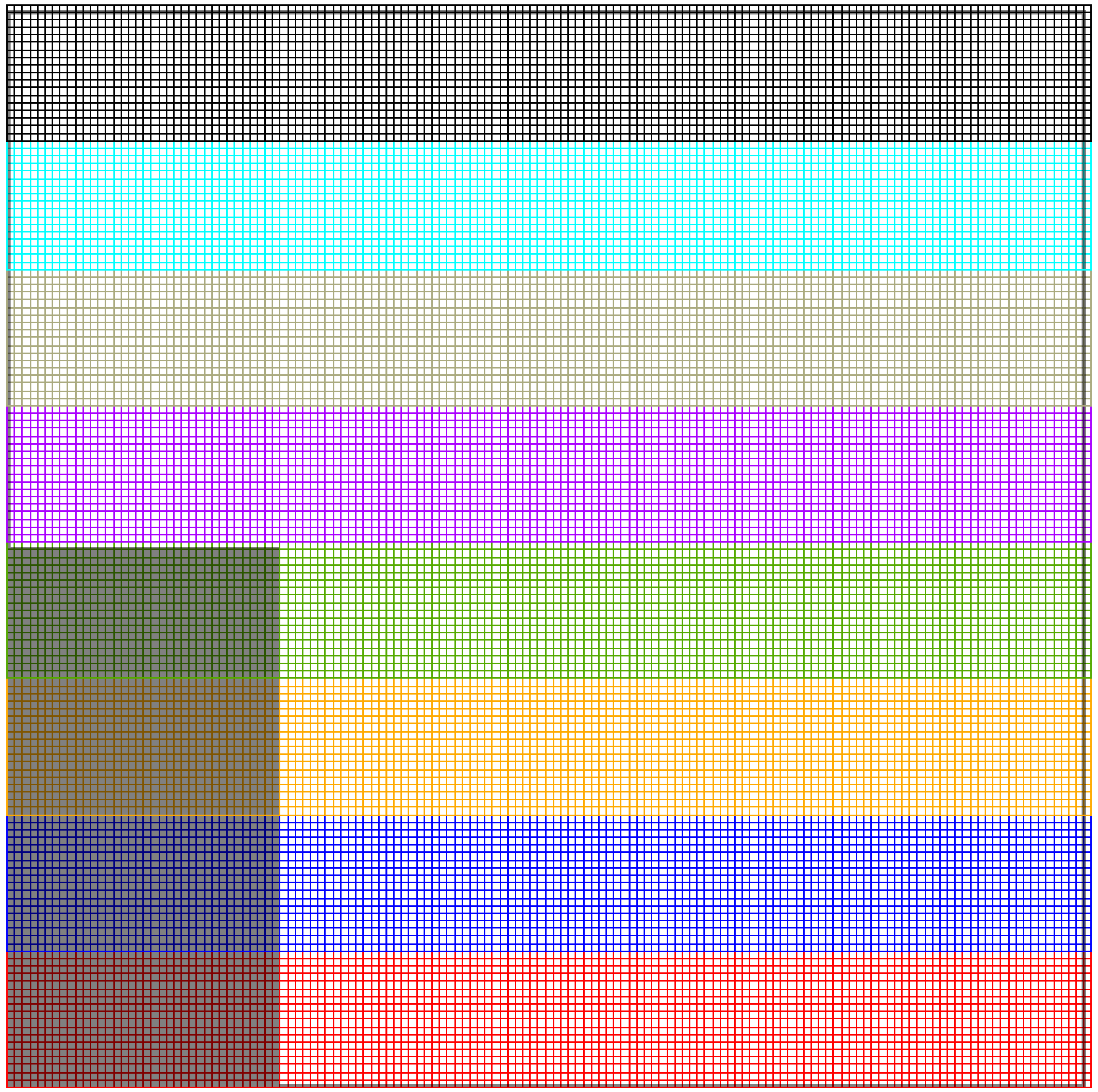


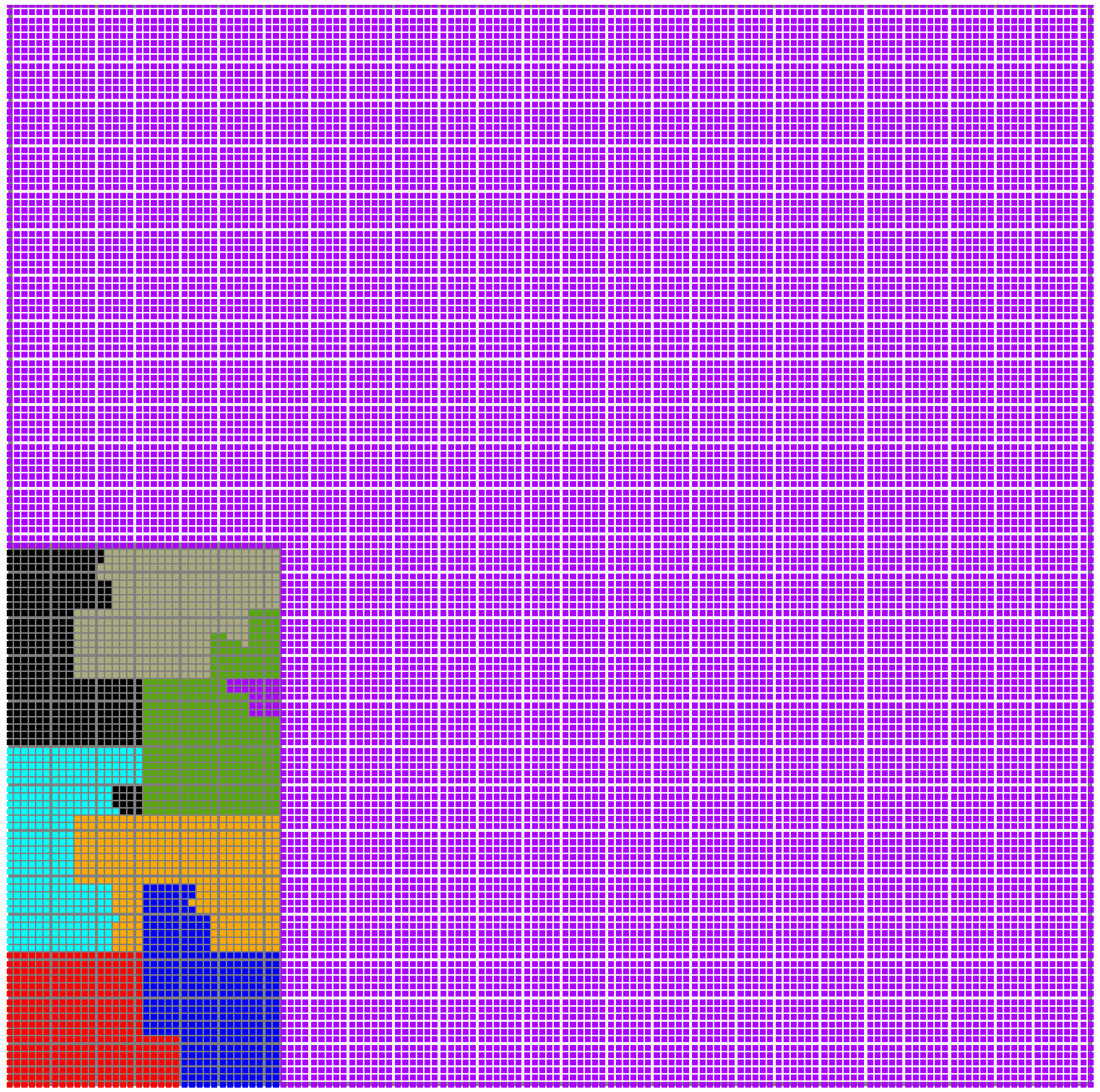
Particle Reordering

- Particles need reordering to maintain local spatial locality
- Currently, particle data reordered using single node algorithm
- Same for LocalCellID – allows mapping to Cellpart
- GlobalCellID is constant
- Better option: reorder along HSFC path[5]

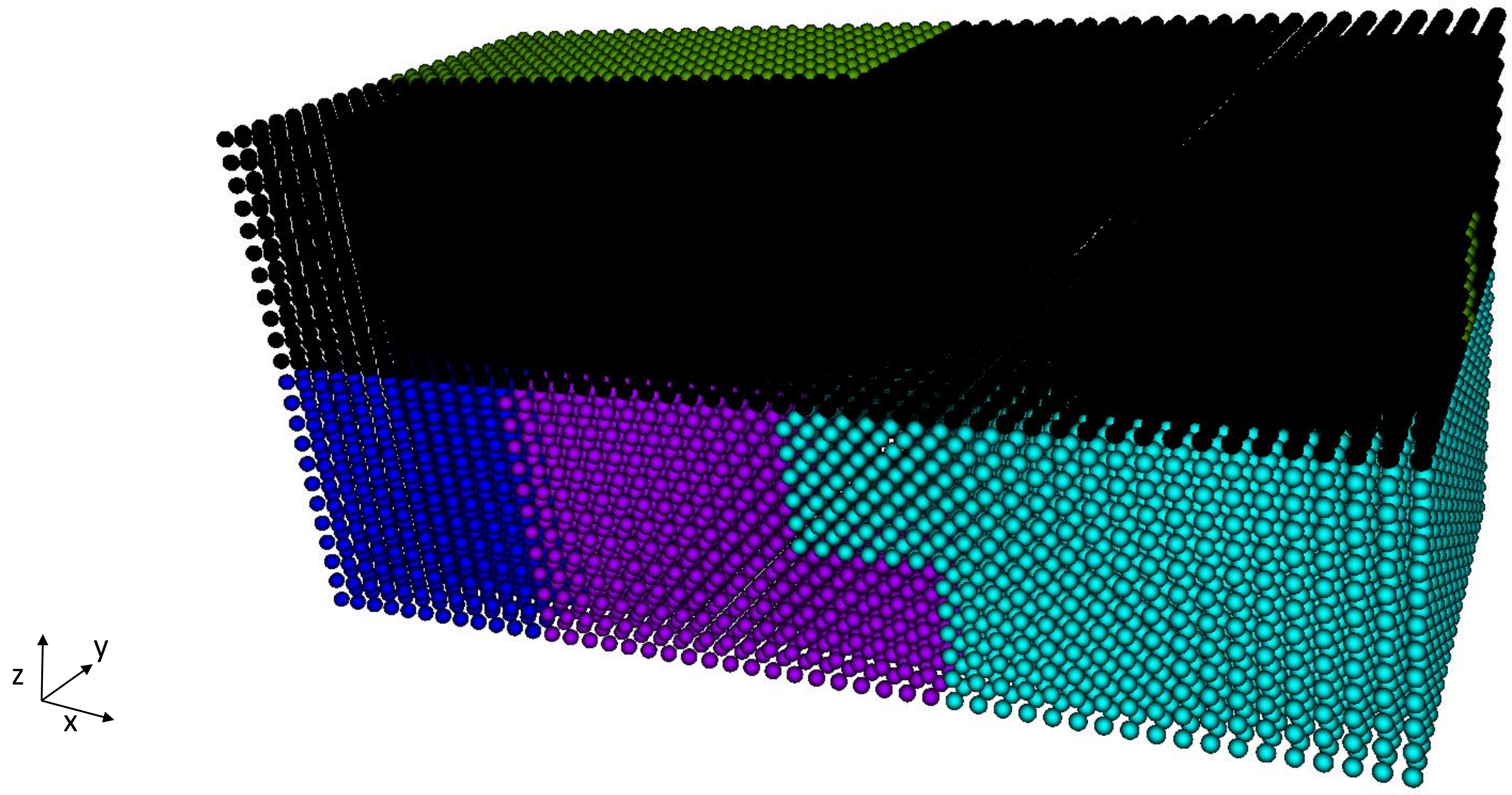


Partition Results

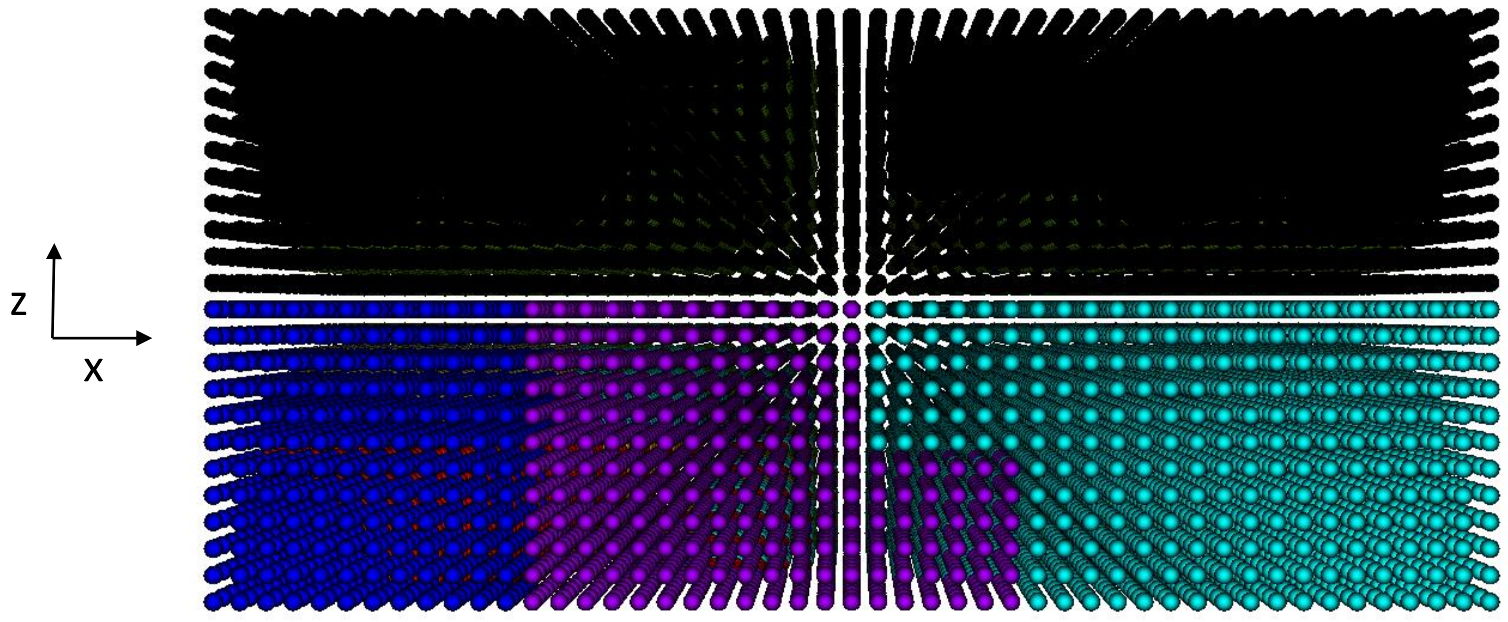


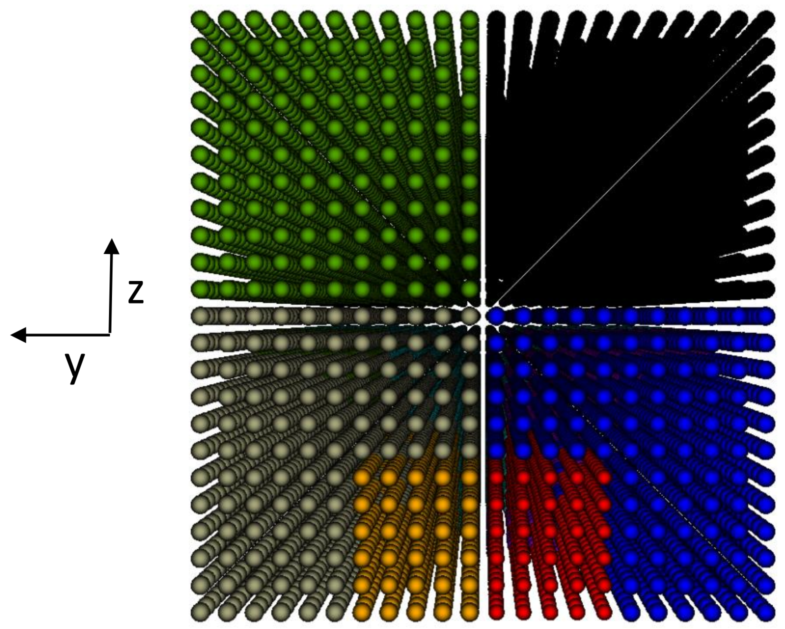


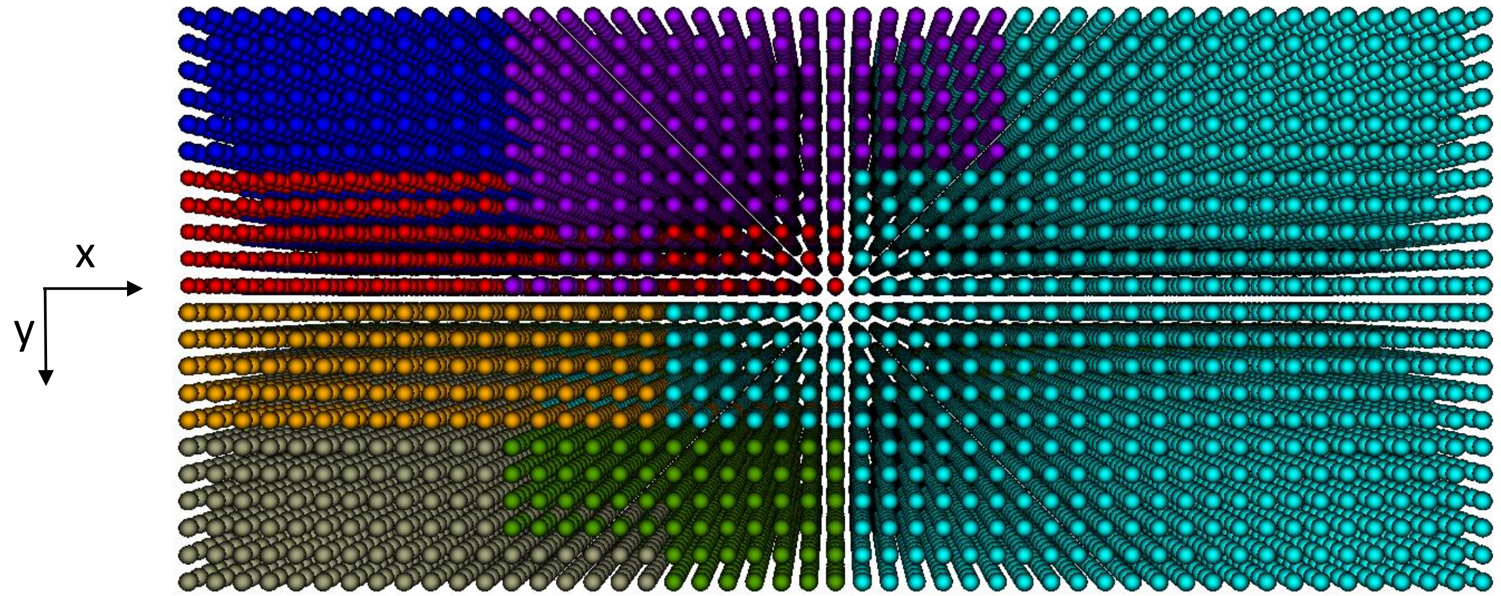
Partition Results



Partition Results





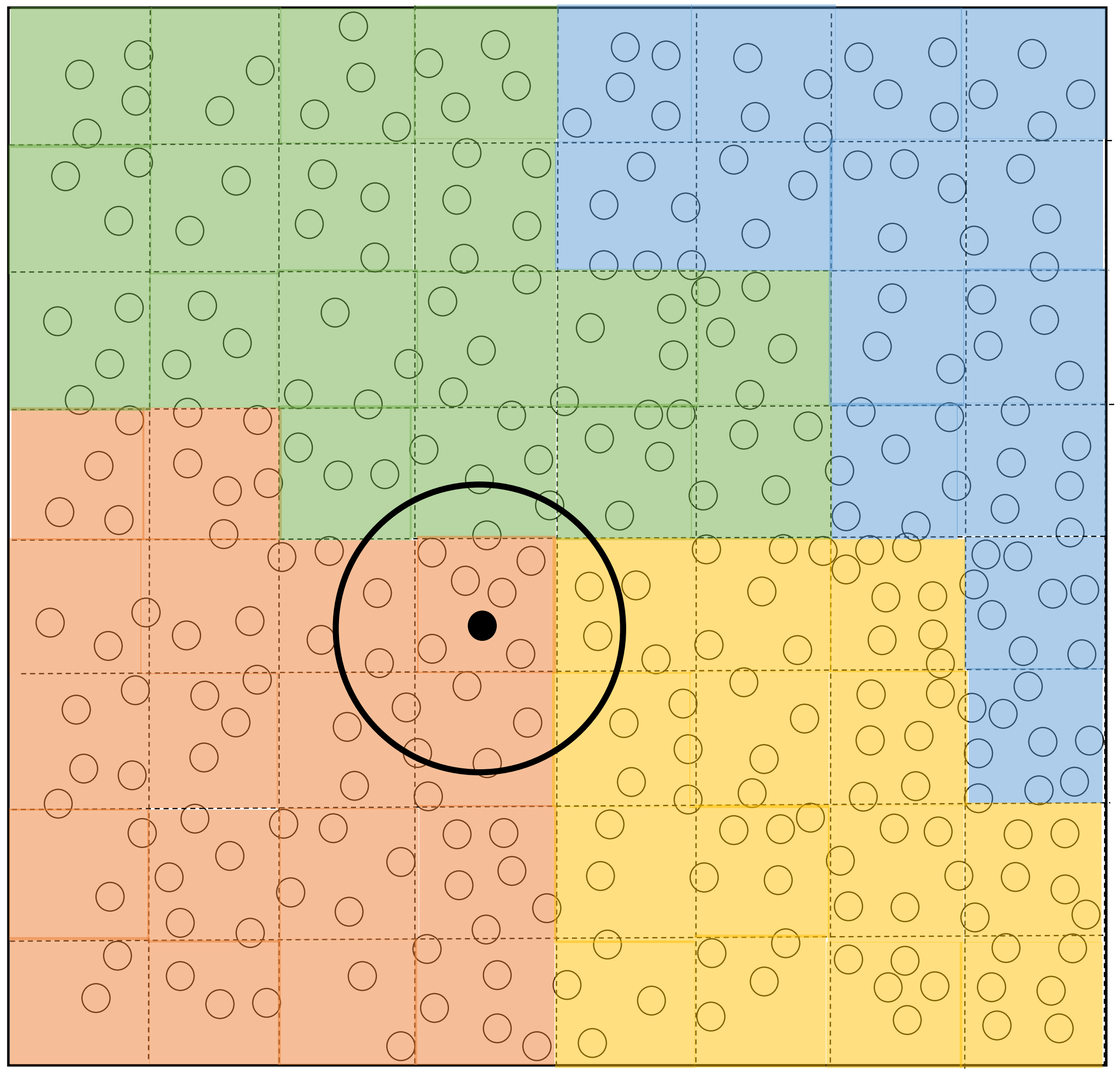


Future Work

- Complete a fully working version of the DualSPHysics MPI code
  - Halo Exchange
  - Particle Exchange
- Assess the code capabilities and validate
- Optimisation
- New I/O functions required – Transition to the Hierarchical Data Format (HDF5)
- Execution to large HPC clusters for 1000s of cores

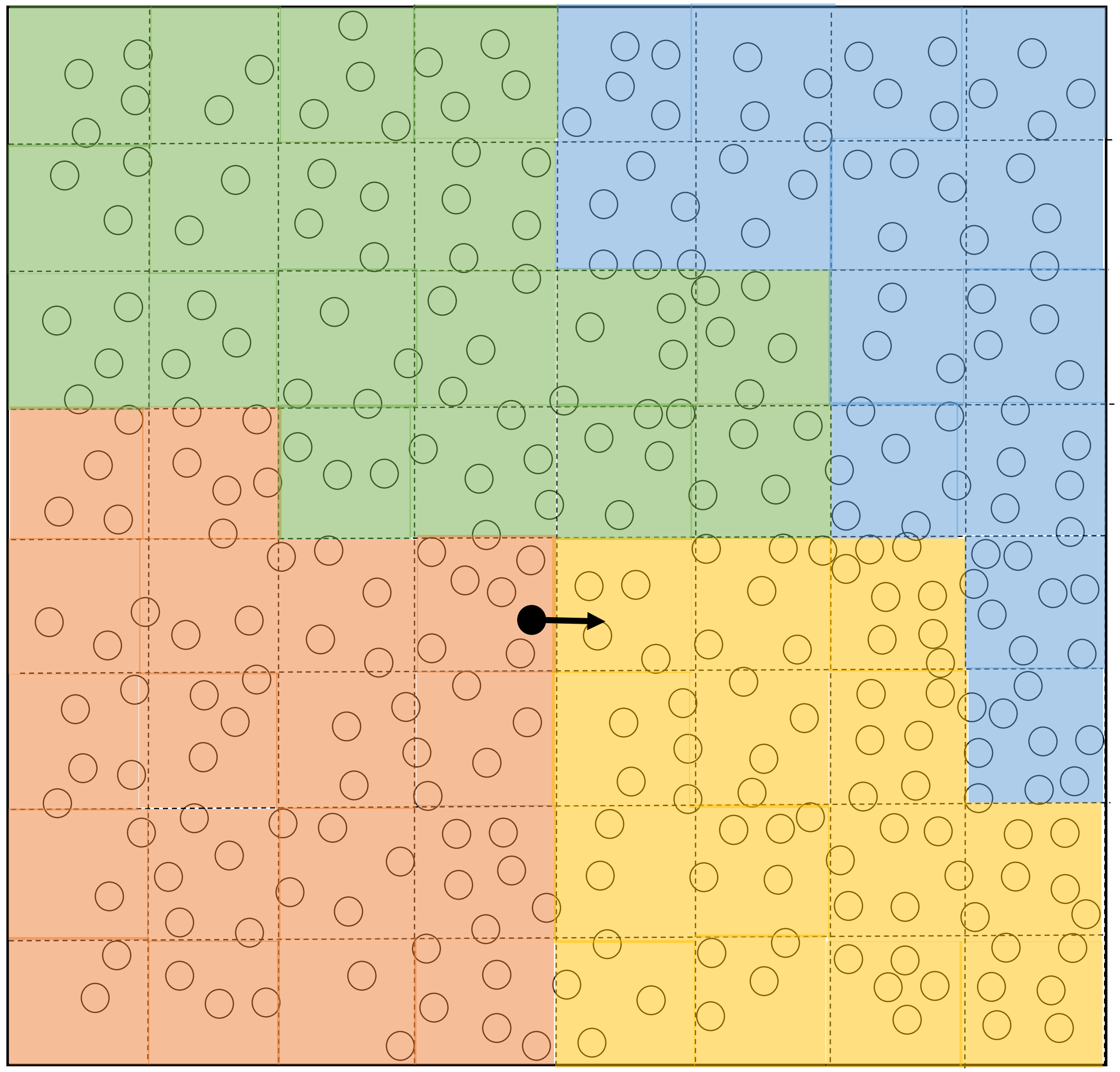
Halo Exchange

- Halo exchange reworked using cells
- Neighbouring cells explicitly known through GlobalCellID
- Identify processes the particles are in and transfer data
- Packing and unpacking algorithms same as previous code



Particle Exchange

- Particles can move out of the cell
- New cell may be in a different process
- Use Cell coordinates to identify edges of the process' domain
- Identify process and cell the particle moves into
- Use same packing/unpacking algorithm
- Process needs to be completed before reordering particle data



References

[1]Crespo, A.J.C., J.M. Dominguez, B.D. Rogers, M. Gomez-Gesteira, S. Longshaw, R. Canelas, R. Vacondio, A. Barreiro, and O. Garcia-Feal, *DualSPHysics: Open-source parallel CFD solver based on Smoothed Particle Hydrodynamics (SPH).* Computer Physics Communications, 2015. **187**(0): p. 204-216.

[2]Valdez-Balderas, D., J.M. Dominguez, B.D. Rogers, and A.J.C. Crespo, *Towards accelerating smoothed particle hydrodynamics simulations for free-surface flows on multi-GPU clusters.* Journal of Parallel and Distributed Computing, 2013. **73**(11): p. 1483-1493.

[3]Dominguez, J.M., A.J.C. Crespo, D. Valdez-Balderas, B.D. Rogers, and M. Gomez-Gesteira, *New multi-GPU implementation for smoothed particle hydrodynamics on heterogeneous clusters.* Computer Physics Communications, 2013. **184**(8): p. 1848-1860.

[4]Devine, K., E. Boman, R. Heaphy, B. Hendrickson, and C. Vaughan, *Zoltan Data Management Service for Parallel Dynamic Applications.* Computing in Science & Engineering, 2002. **4**(2):p.90-97.

[5]Guo, X., B.D. Rogers, S. Lind and P.K. Stansby, New Massively Parallel Scheme for Incompressible Smoothed Particle Hydrodynamics (ISPH) for Highly Nonlinear and Distorted Flow, in *Computer Physics Communications*, under publication.

[6]Guo, X., S. Lind, B.D. Rogers, P.K. Stansby, and M. Ashworth, *Efficient massive parallelisation for incompressible Smoothed Particle Hydrodynamics with 10^8 particles*, in *8th International SPHERIC Workshop*. 2013: Trondheim, Norway.

[7]Guo, X., B.D. Rogers, S. Lind, P.K. Stansby, and M. Ashworth, *Exploring an Efficient Parallel Implementation Model for 3-D Incompressible Smoothed Particle Hydrodynamics,* in *10th International SPHERIC Workshop*. 2013: Trondheim, Norway.

MANCHESTER 1824
The University of Manchester

Thank you

archer

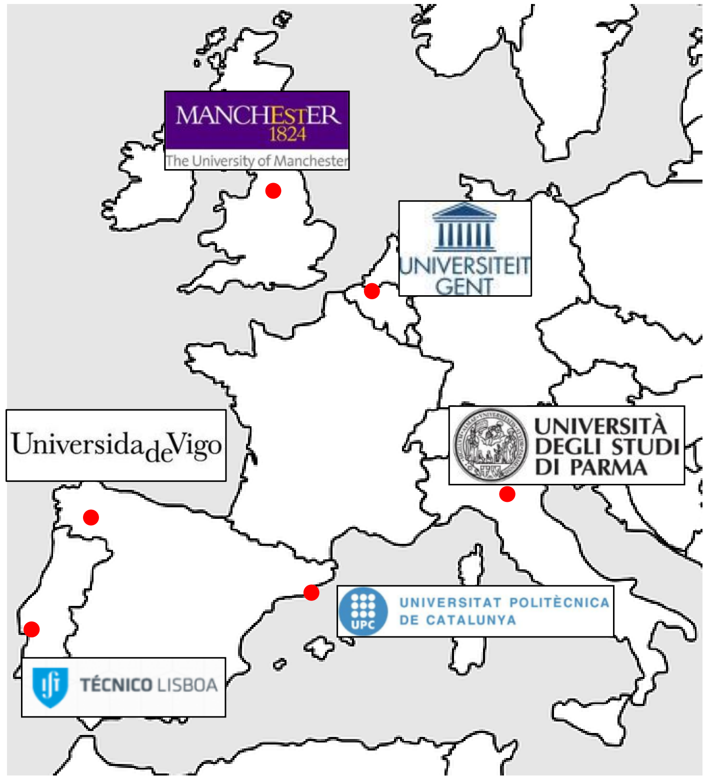
Acknowledgements

- U-Man: Georgios Fourtakas, Peter Stansby, Steve Lind
- STFC: Xiaohu Guo, Stephen Longshaw
- U-Vigo: Alex Crespo, Moncho Gomez-Gesteira
- U-Parma: Renato Vacondio

@SPH_Manchester

Free open-source **DualSPHysics** code:
http://www.dual.sphysics.org

cpu gpu
DualSPHysics


MANCHESTER 1824
The University of Manchester
UNIVERSITEIT GENT
UniversidadeVigo
UNIVERSITÀ DEGLI STUDI DI PARMA
UPC UNIVERSITAT POLITÈCNICA DE CATALUNYA
TÉCNICO LISBOA

Additional Models

- Viscosity:

$$\Pi_{ij} = \sum_{j}^{N} \frac{m_j}{\rho_i \rho_j} \left(\mu_j + \mu_i\right) \boldsymbol{u}_{ij} \frac{\boldsymbol{r}_{ij} \cdot \nabla W_{ij}}{\left|\boldsymbol{r}_{ij}\right|^2}$$

- δ-SPH:

$$\left\langle \frac{d\rho}{dt} \right\rangle = \rho_i \sum_{j} \frac{m_j}{\rho_j} \left(\boldsymbol{u}_i - \boldsymbol{u}_j\right) \cdot \nabla_i W_{ij} + D_i \qquad D_i = \delta h c_s \sum_{j}^{N} 2 \frac{m_j}{\rho_j} \left(\rho_j - \rho_i\right) \frac{\boldsymbol{r}_{ij} \cdot \nabla W_{ij}}{\left|\boldsymbol{r}_{ij}\right|^2}$$

- Quintic Wendland kernel:

$$W_{ij} = \alpha_D \left(1 - \frac{q}{2}\right)^4 \left(2q + 1\right) \text{ where } \mathrm{q} = \frac{\boldsymbol{r}_\mathrm{i} - \boldsymbol{r}_j}{h}$$

- Equation of state:

$$P(\rho) = P_0 \left[\left(\frac{\rho}{\rho_0} \right)^{\gamma} - 1 \right]$$

- 2nd order Velocity Verlet time marching scheme