Extending DualSPHysics to massive CPU clusters

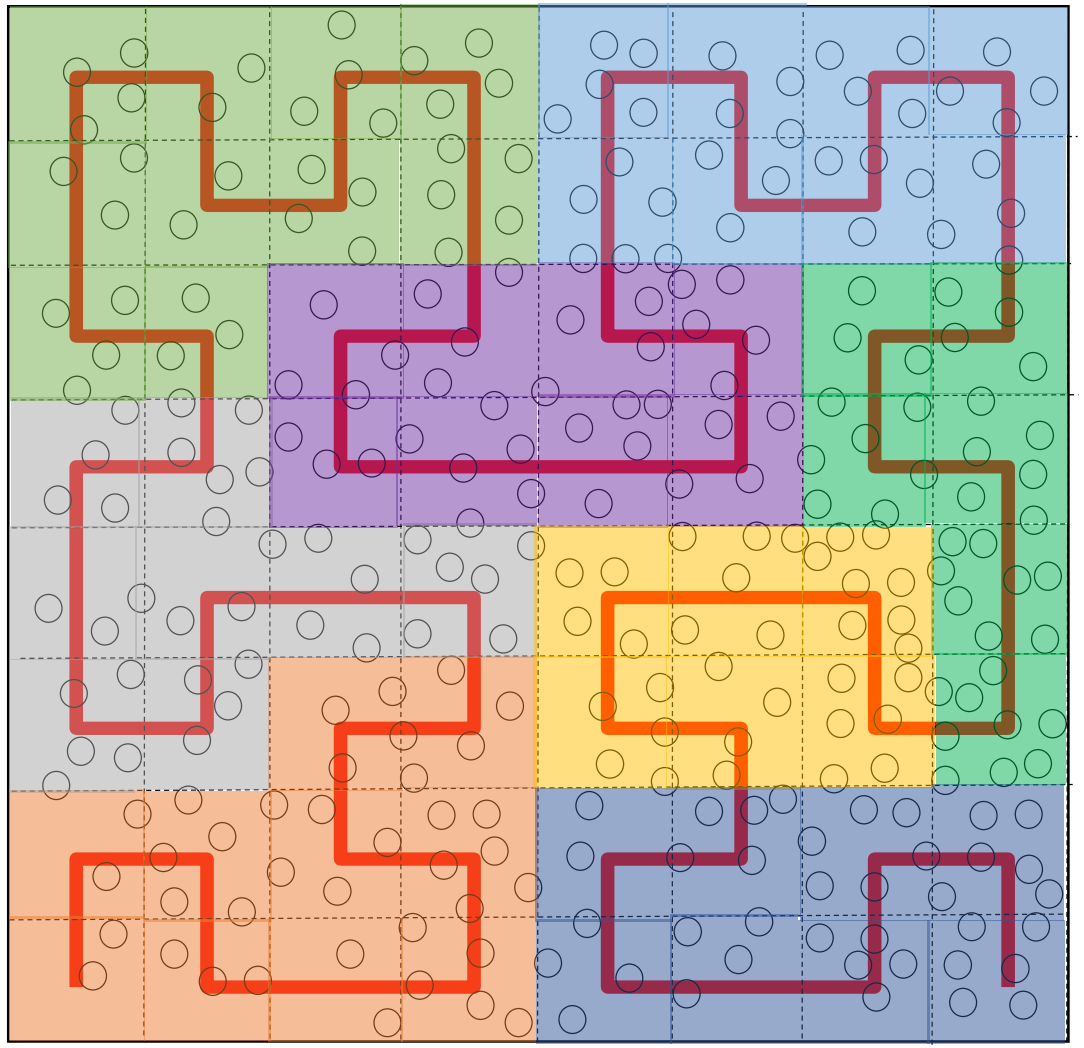
Athanasios Mokos, José Dominguez, Benedict D. Rogers

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

Funded by the eCSE, eCSE07-16

Outline of Presentation

- Motivation for Research
- Message Passing Interface
- Using MPI with DualSPHysics
  - Domain Decomposition
  - Halo Exchange
  - Asynchronous Communications
- Scalability
  - Dynamic Load Balancing
- Zoltan Library
  - Hlibert Space Filling Curve
- Using Zoltan with DualSPHysics
  - Cell and particle mapping
  - Load Balancing algorithm
- Future Work
  - Halo and Particle Exchange



SPH for real problems

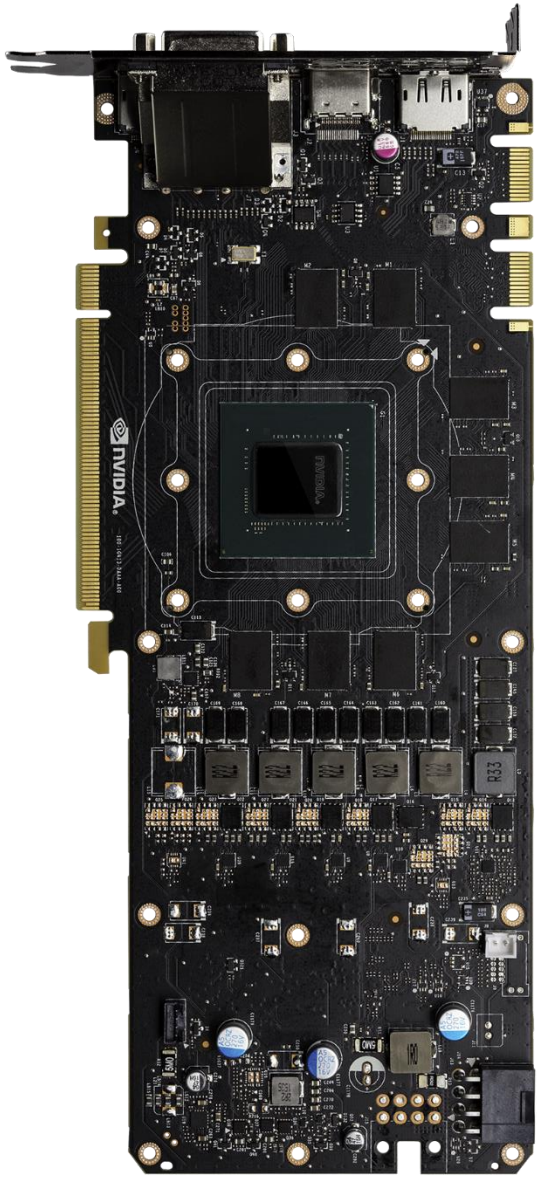
- Real-life applications are complex 3D flows
- Multi-scale problems with long runtimes
- SPH requires over 10^7 particles to model them
- Must do so as quickly as possible

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

Photo by University of Plymouth

Graphics Processing Units

- GPUs are excellent for SPH:
  - Massively Parallel, ideal for n-body simulations
  - Low cost and energy consumption (**Green Computing**)
- But…
  - Still in their infancy (less developed tools and compilers)
  - Significant speed drop when using double precision
  - Require specialised hardware (cannot take advantage of existing HPC infrastructure)
  - Require new investment in personnel

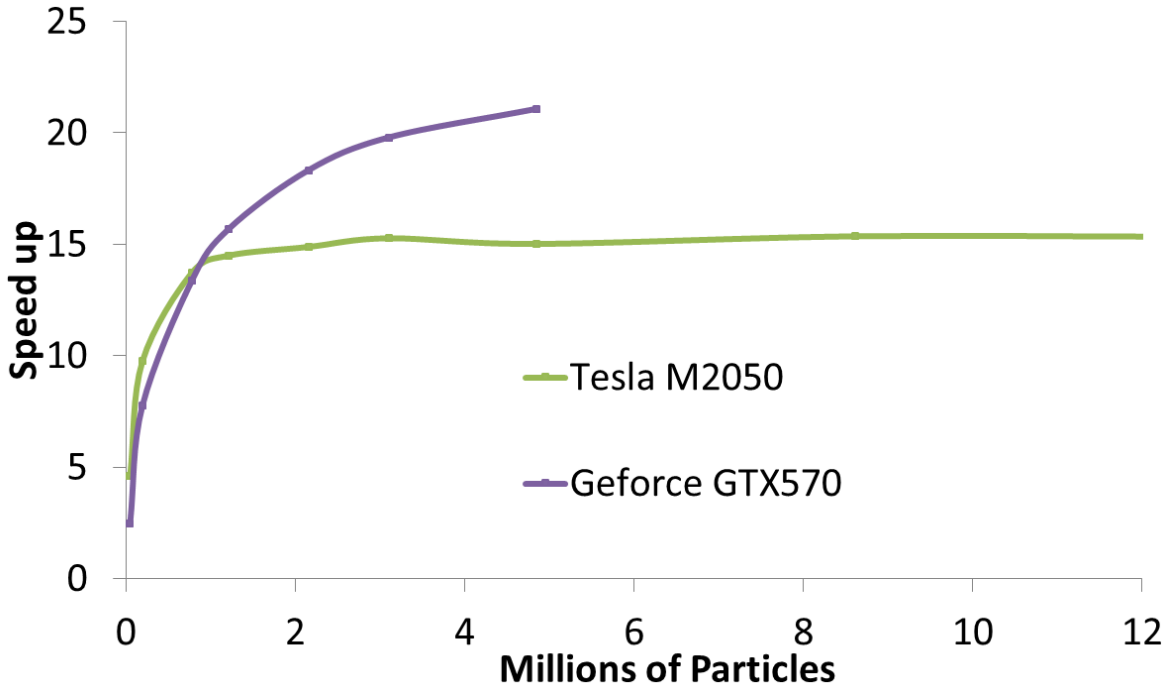


Nvidia GTX1080

Current State of DualSPHysics

- Developing a CPU version of DualSPHysics that can tackle these problems is an attractive proposition

- Current State of the CPU implementation:



- Highly optimised code for a single node
- Multiple execution options
- Pre- and post-processing tools
- OpenMP implementation

SOLUTION: Use multiple processing nodes

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 **100-1000s** of cores (about 100 cores needed for equivalent performance to GPU[1]
- Implementation of the **Message Passing Interface (MPI)** standard

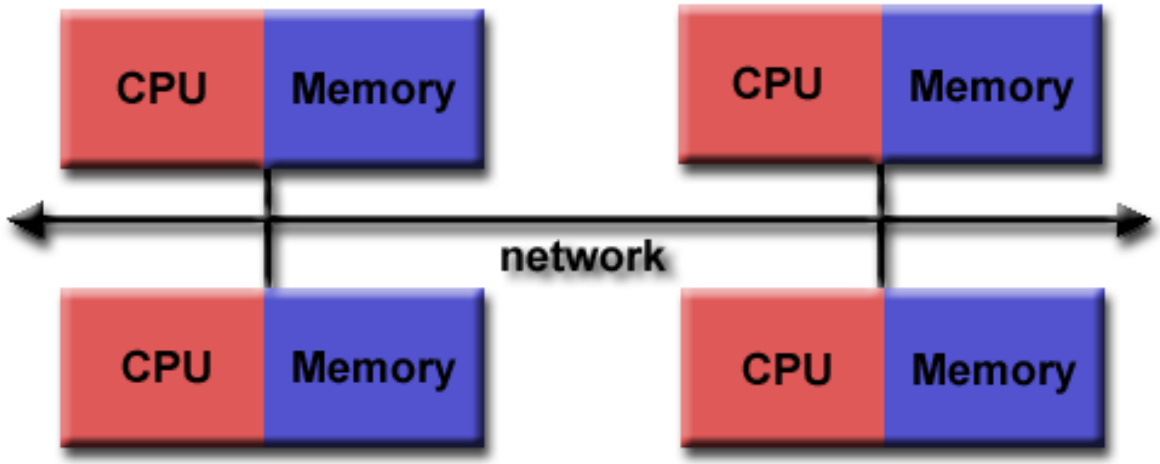
- Single node (OpenMP) -> Communication between different nodes (MPI)
- **AIM: Develop a hybrid OpenMP-MPI program that can scale to 1000s of cores**

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



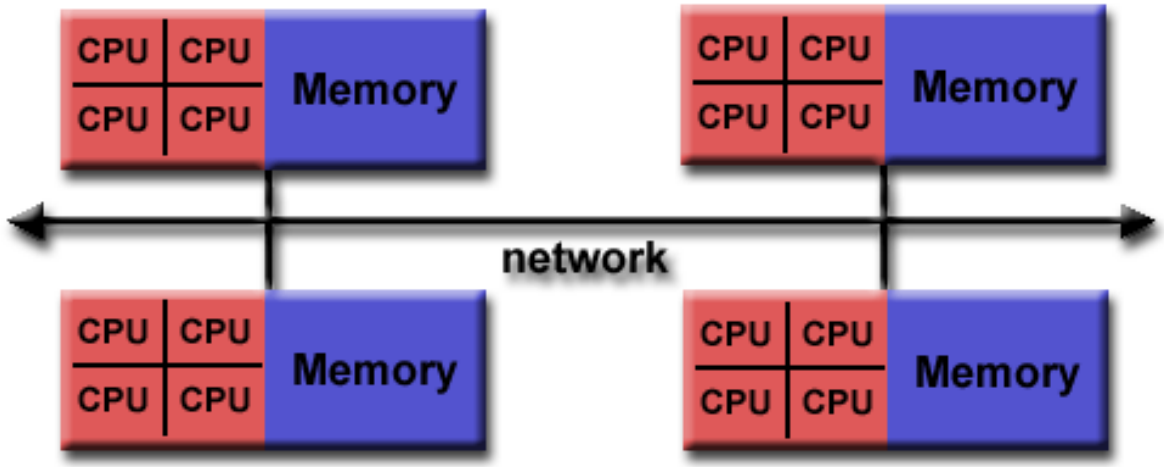
Message Passing Interface

- 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

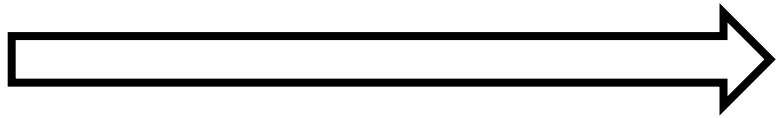
- Maintain DualSPHysics optimisation and structure
  - Cell-linked neighbor list[3]
  - Ease of use
  - Reduce changes in SPH computation
  - Limits options when creating particles and cells

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

Integrating MPI in DualSPHysics

Single node files

- JCellDivCpuSingle
- JPartsLoad4
- JSphCpuSingle



MPI files

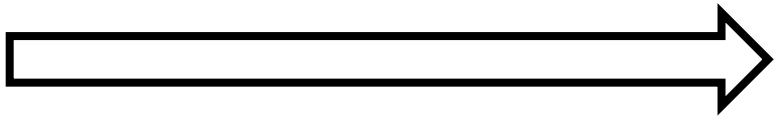
- CellDivCpuMPI
- ParticleLoadMPI
- SphCpuMPI

- Changes focused on:
  - Loading data from GenCase
  - 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

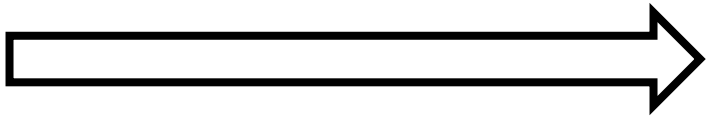


MPI files

- CellDivCpuMPI
- ParticleLoadMPI
- SphCpuMPI

New files created to handle:

- Node communication
- Domain Decomposition
- Halo Exchange

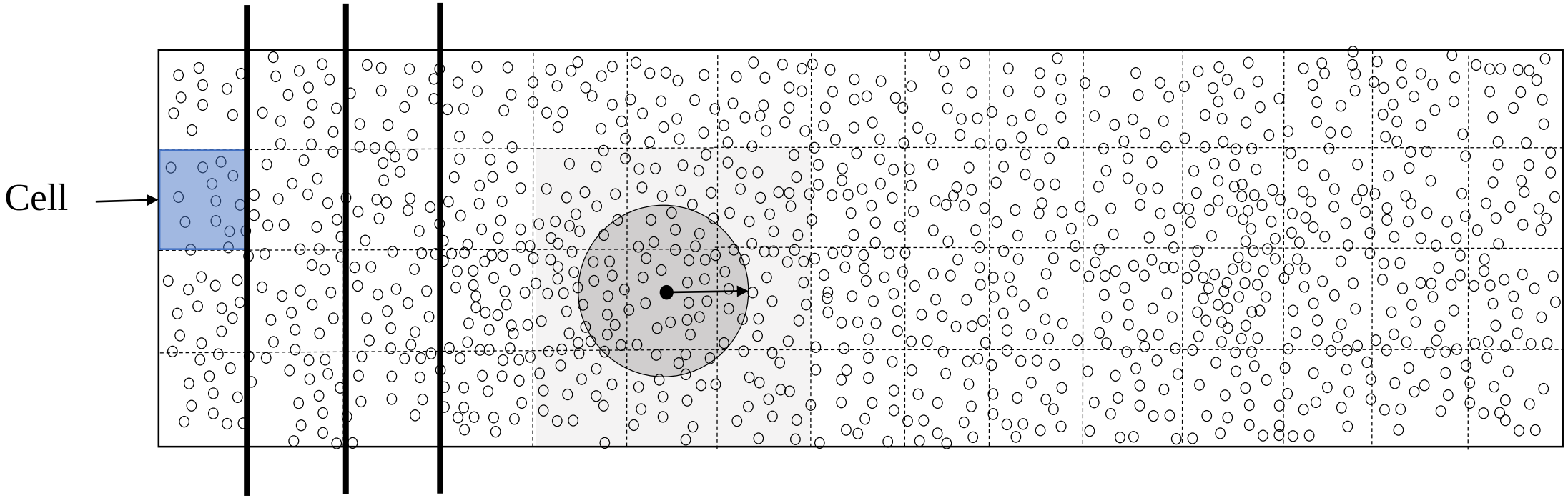


- BufferMPI
- DataCommMPI
- HostMPI
- InfoMPI
- SliceMPI
- SphMPI
- SphHaloMPI

Domain Decomposition

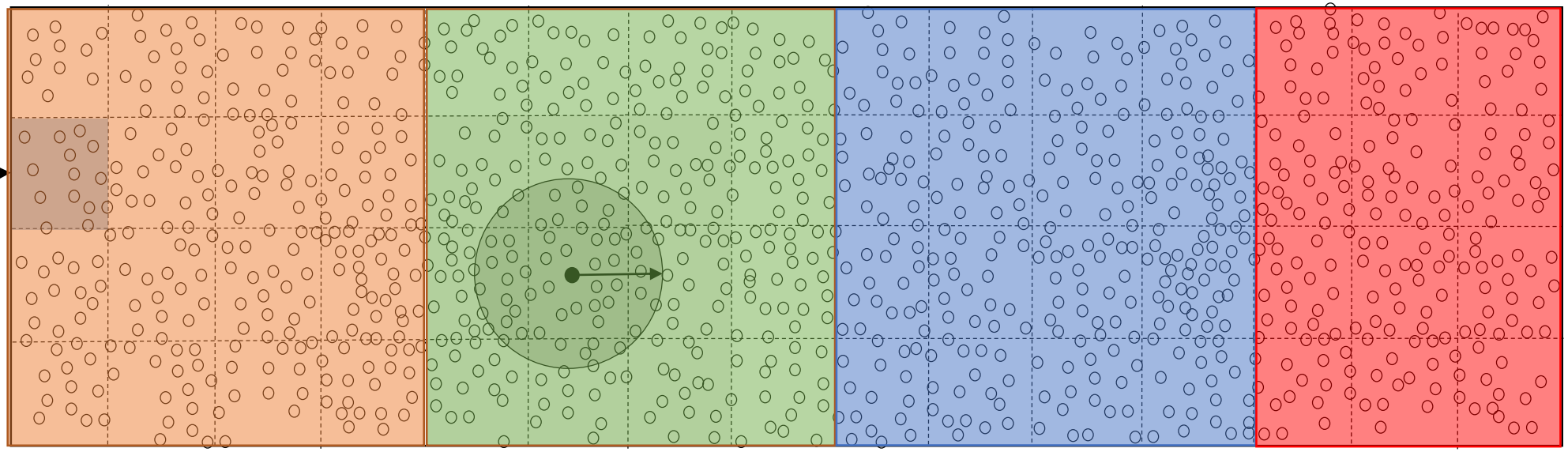
- Divide the domain between nodes
- Unique particle and cell list
- 1D decomposition through slices[2]

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



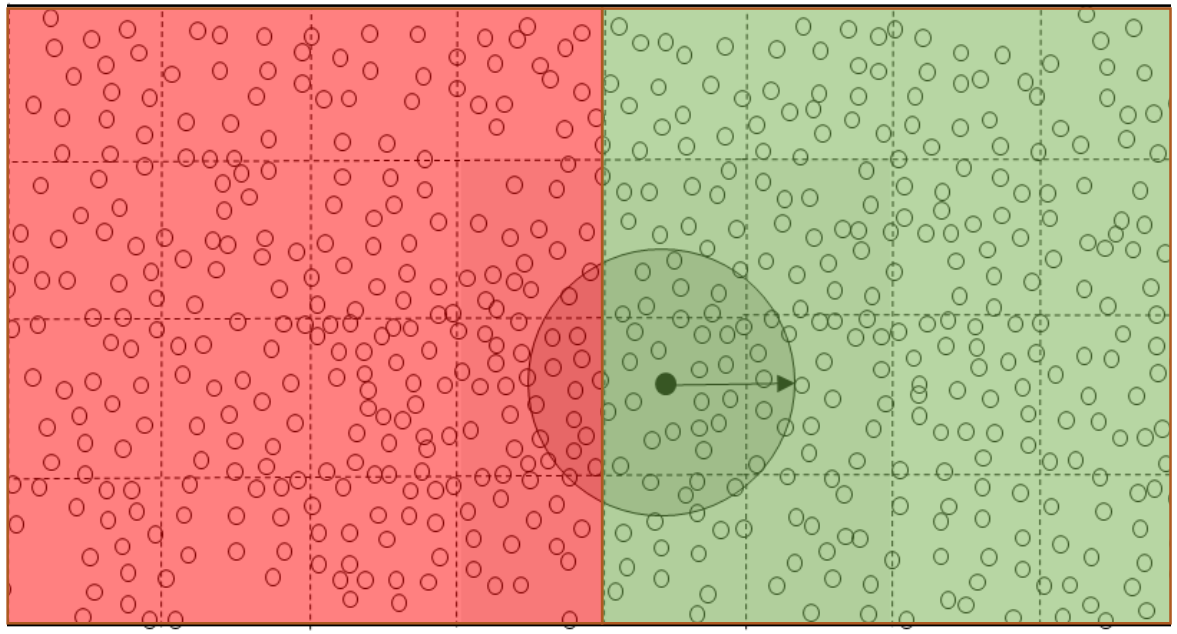
Domain Decomposition

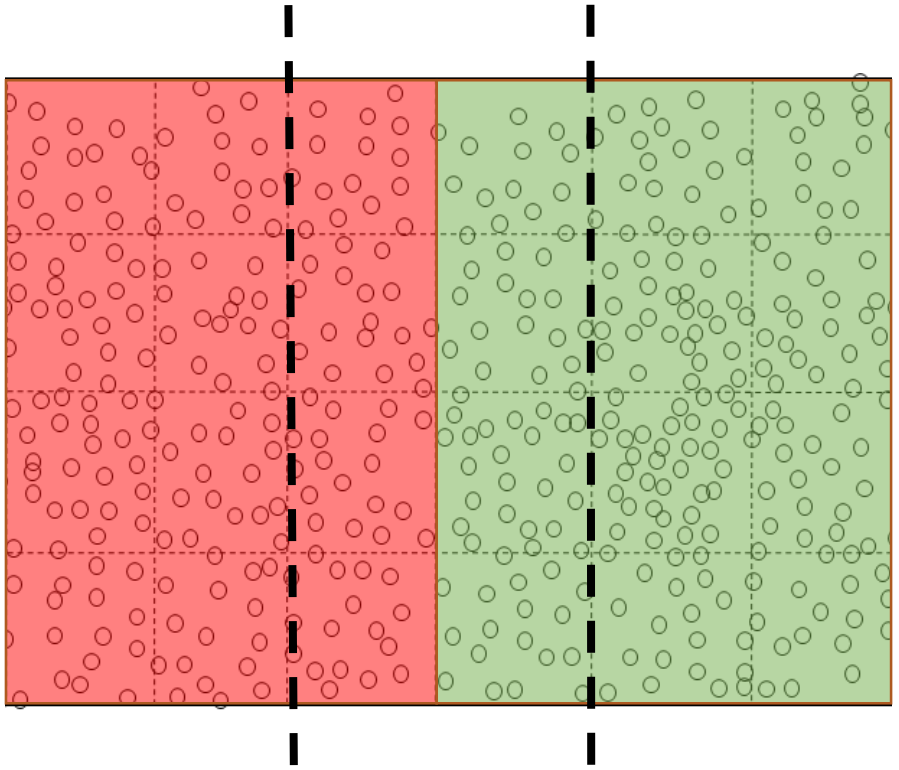
- Divide the domain between nodes
- Unique particle and cell list
- 1D decomposition through slices[2]



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

Halo Exchange

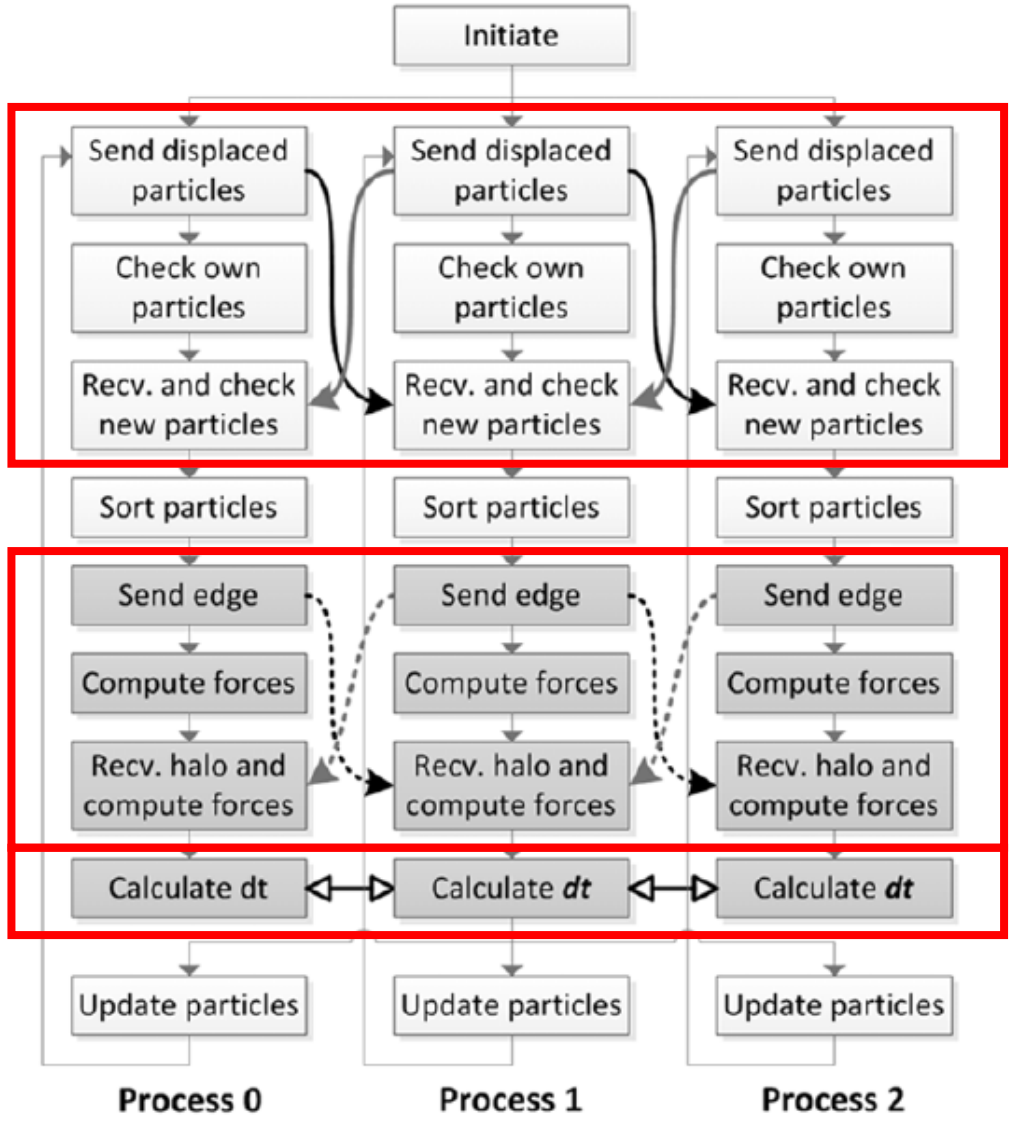




- Identify neighbouring particles in another process or particles moved from another process
- Transfer only the data of all potential neighbours
- Use a **halo** system for more efficiency[3]
- Only data from the neighbouring slice (distance 2h) are transferred
- Edge particles form the **halo** of the subdomain
- Similar procedure on every subdomain border

Asynchronous Communications

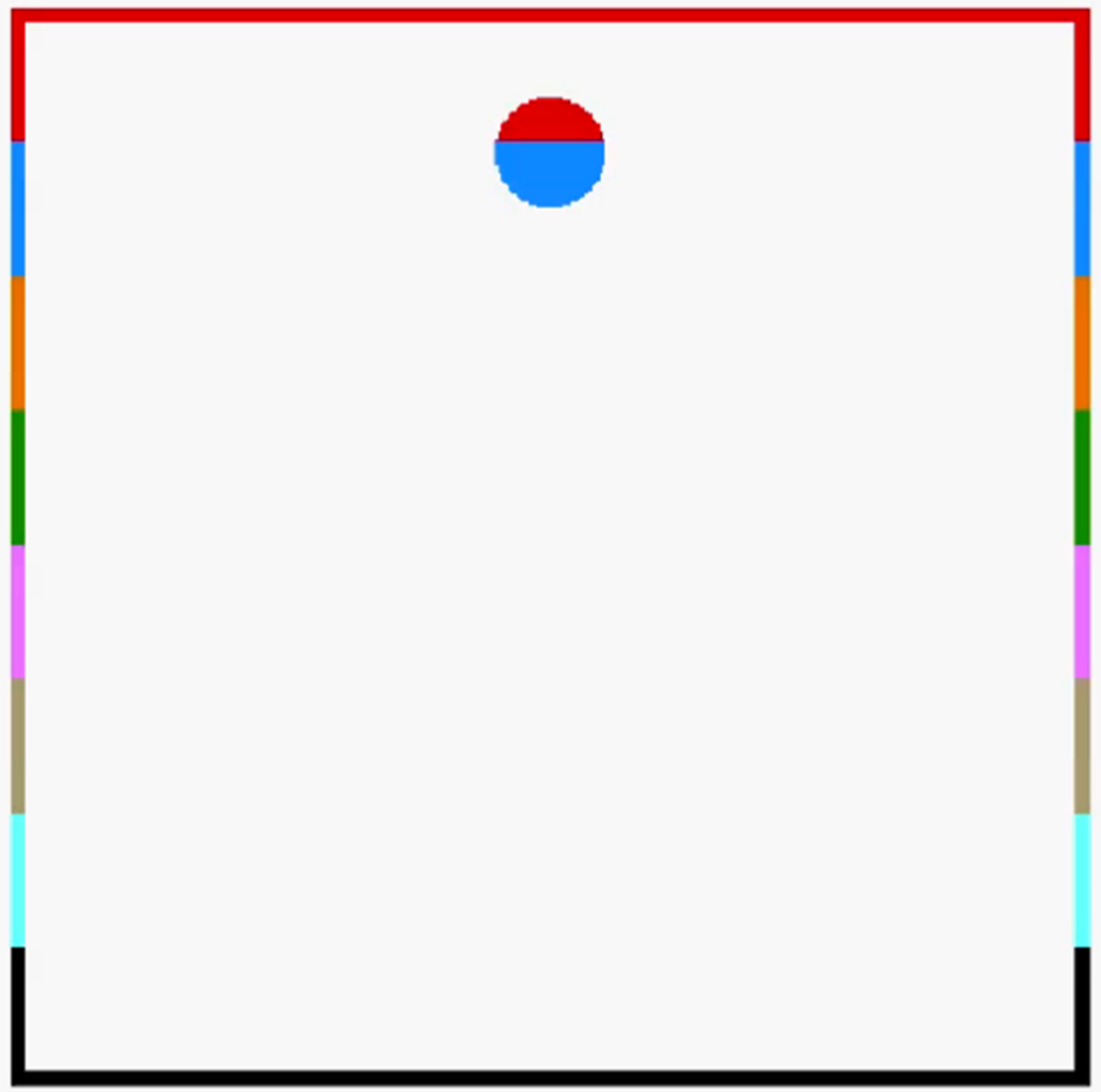
- Objective: Minimise waiting time for data transfer
- Neighbour list of interior particles processed while sending data of displaced particles
- Compute forces on interior particles while receiving halo data
- Processes synchronise when calculating the time step



(Dominguez et al. 2013)[2]

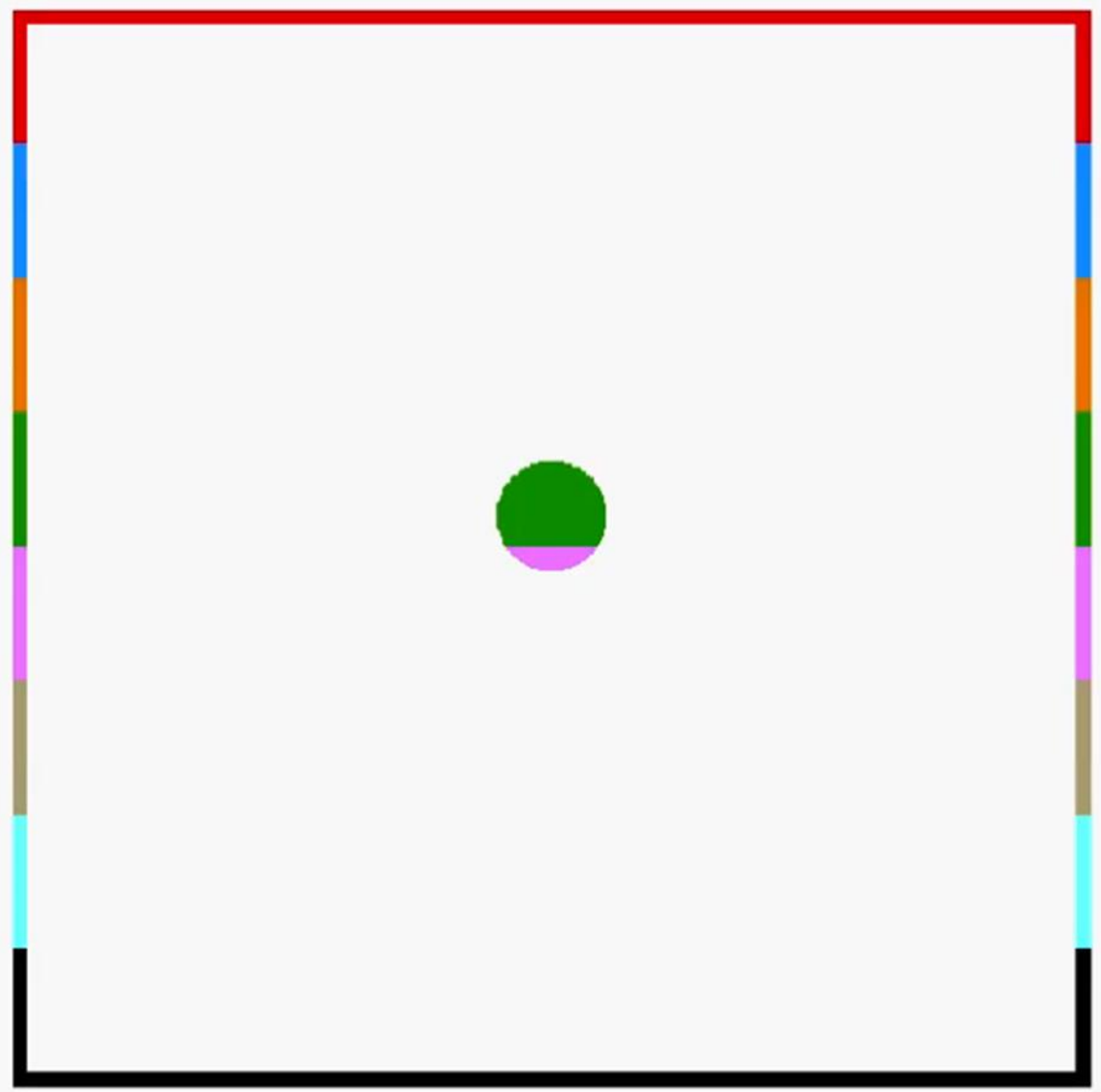
Results

- 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



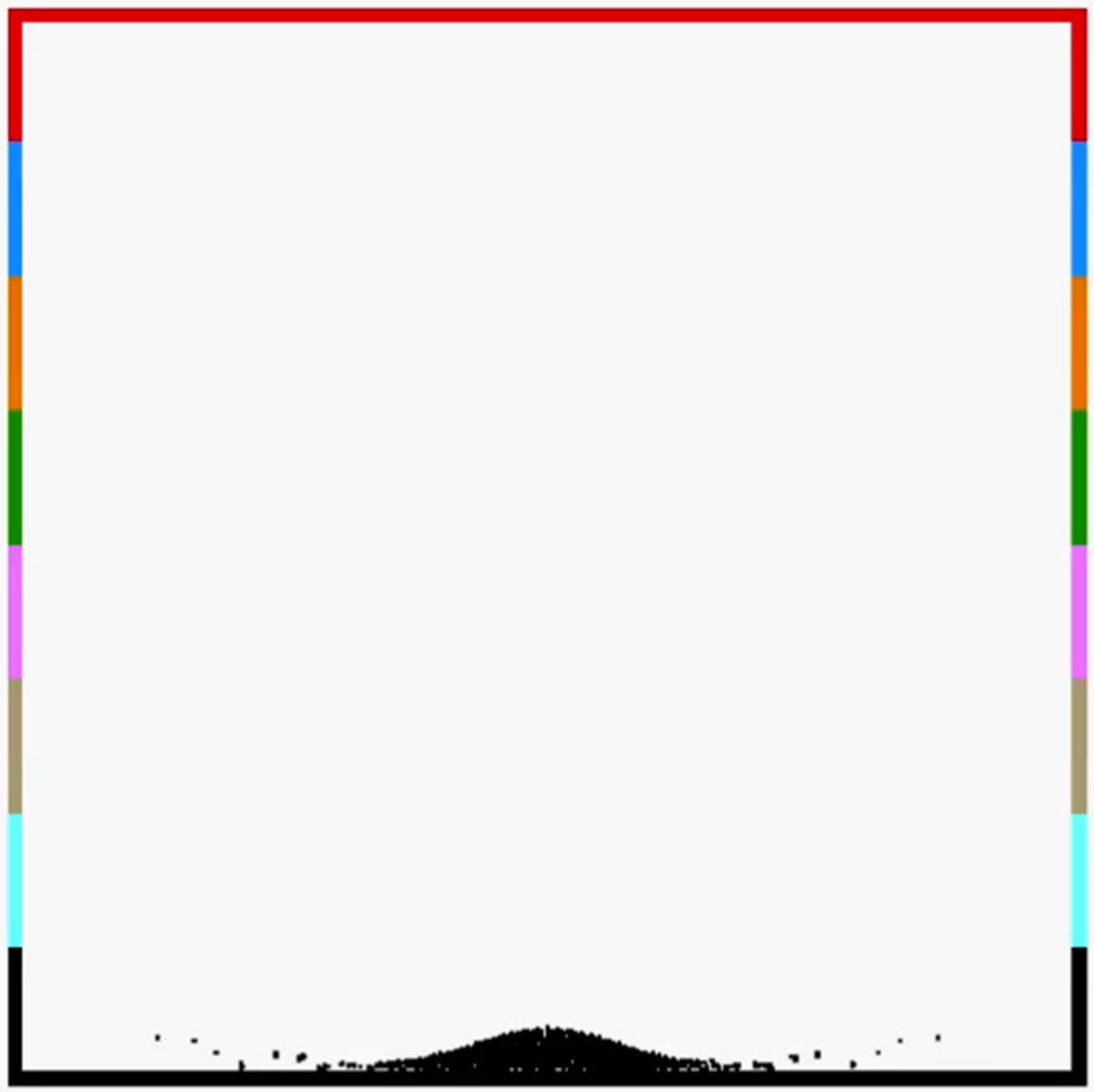
Results

- 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



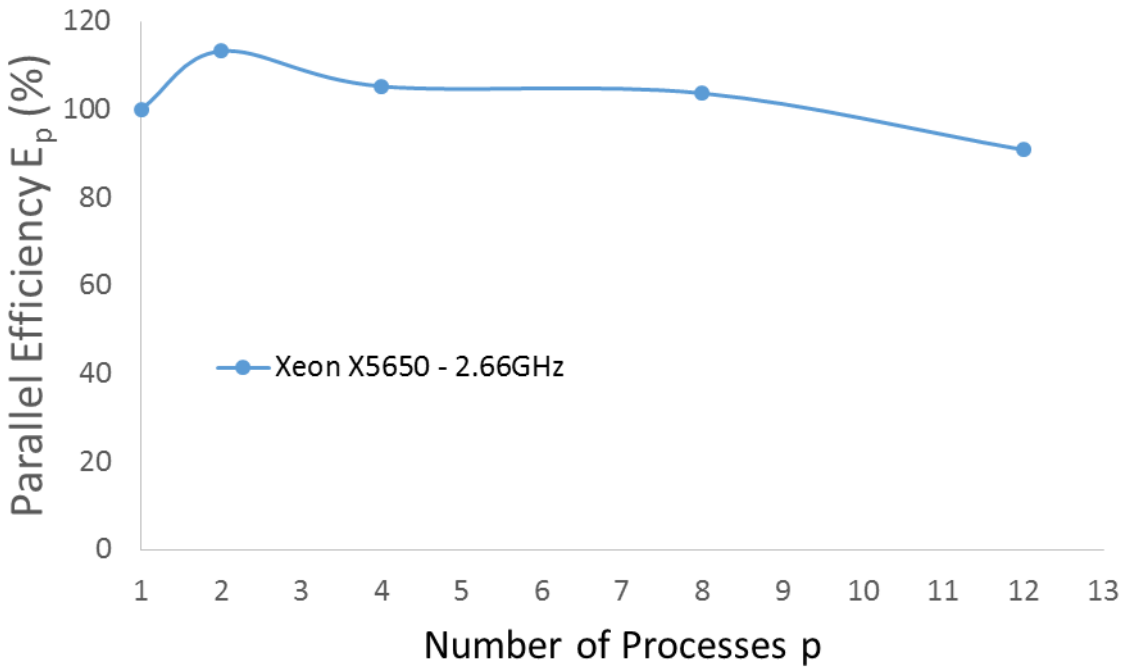
Results

- 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



Scalability

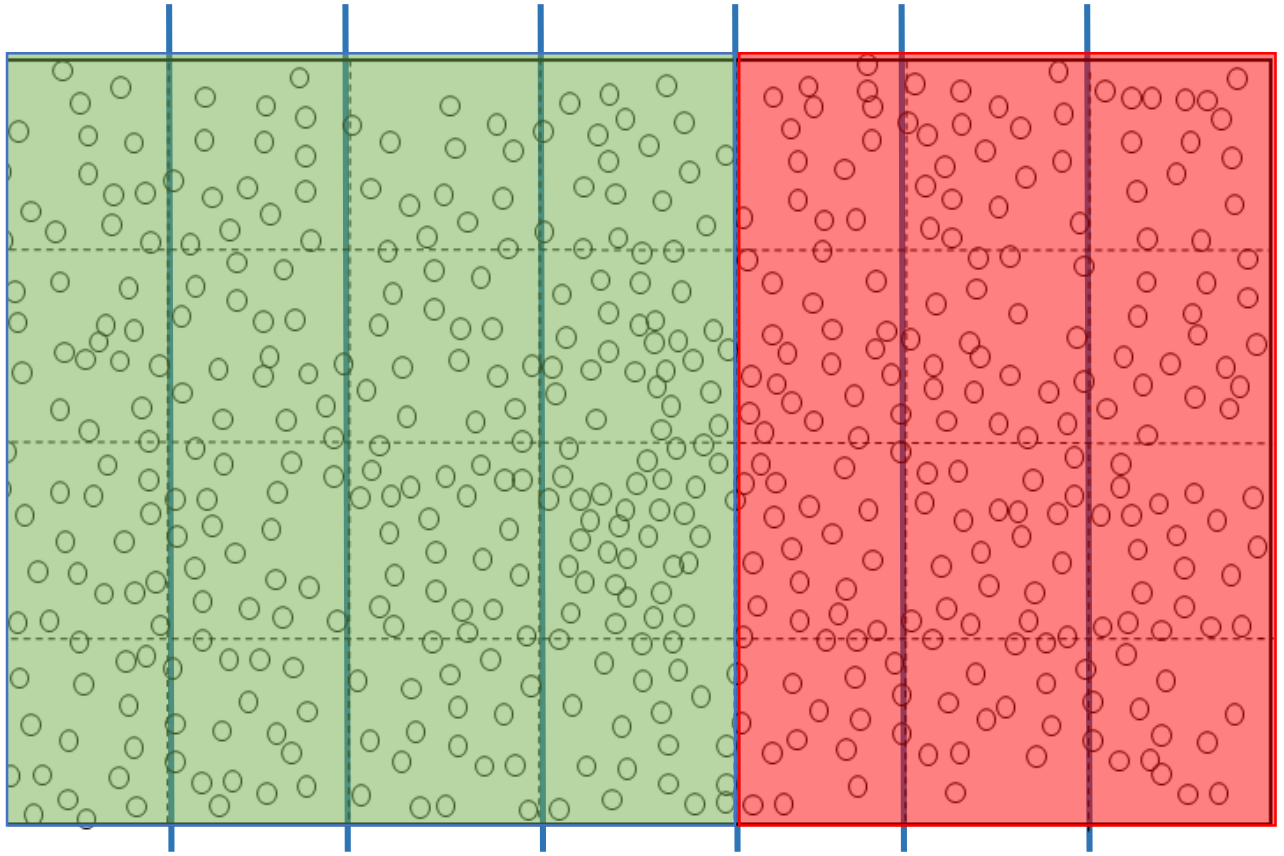
- Code can be further optimised
- Parallel Efficiency $E_p = \frac{T_p}{pT_1} 100\%$
- Possible release for small scale applications?
- Scalability issues do not allow efficient computation with ~100 processes
- 1D decomposition not scalable
- No load balancing



REMINDER: We need more than **10^7** particles for the target problems

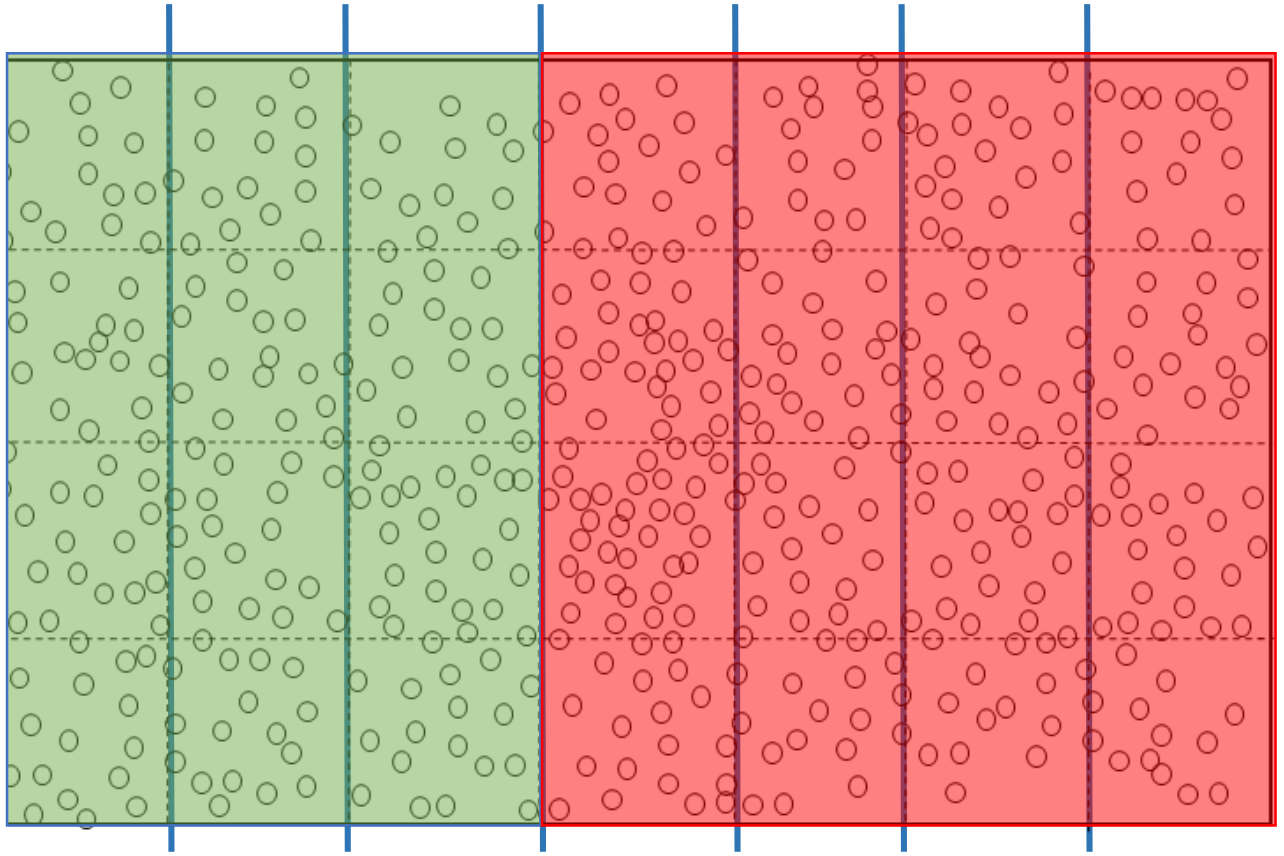
Dynamic Load Balancing

- 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



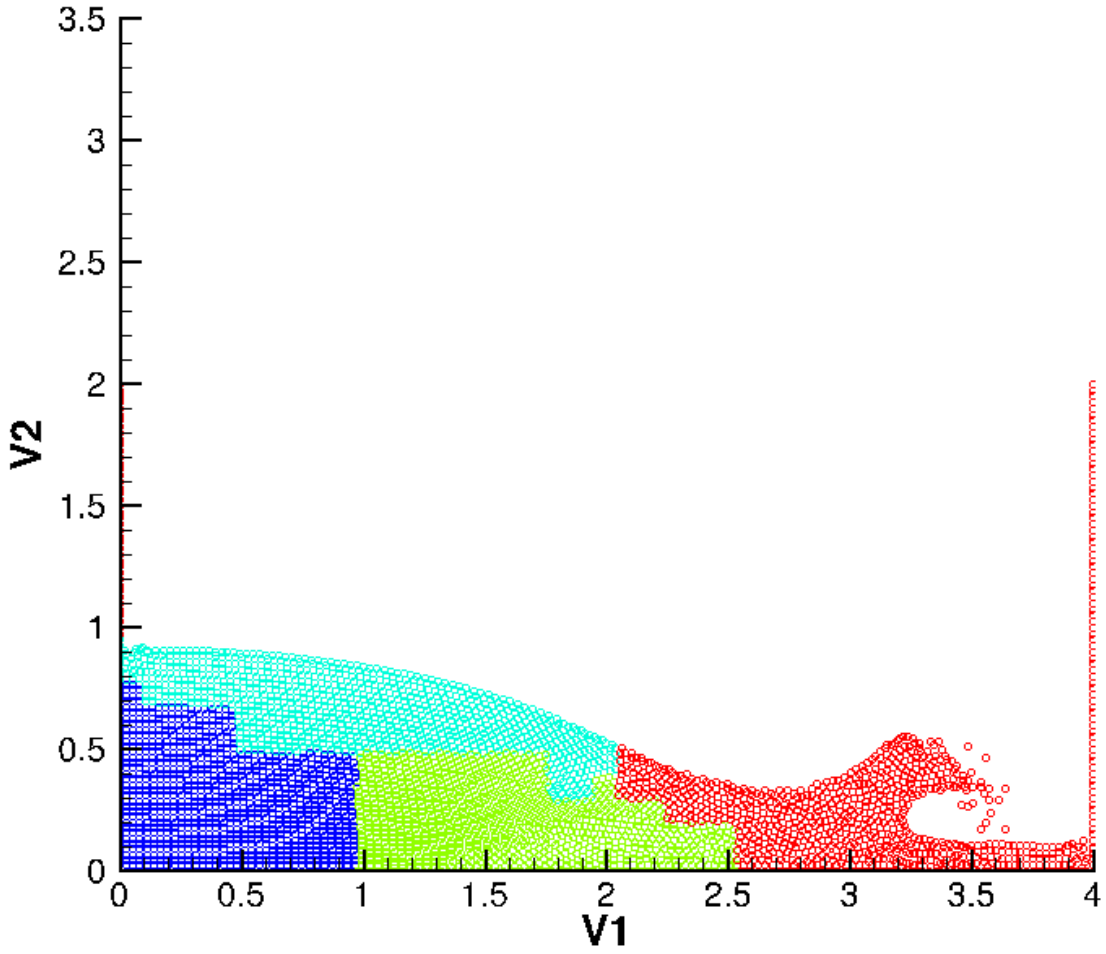
Dynamic Load Balancing

- 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

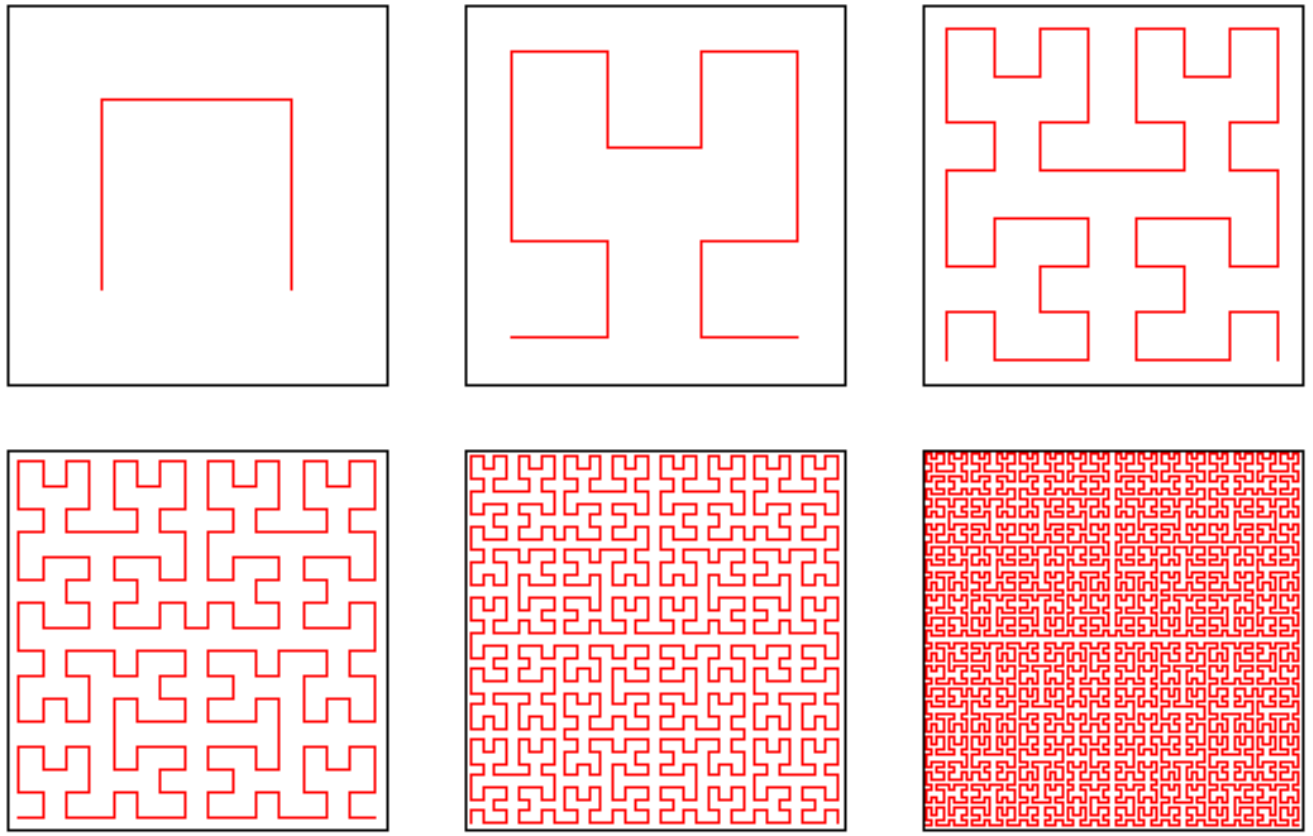
- Use of the Zoltan data management library[4]
- Library for the development and optimization of parallel, unstructured and adaptive codes
- Scalable up to 10^6 cores[4]
- Includes a suite of spatial decomposition and dynamic load balancing algorithms and an unstructured communication package
- Geometric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



Guo et al. (2013)[5]

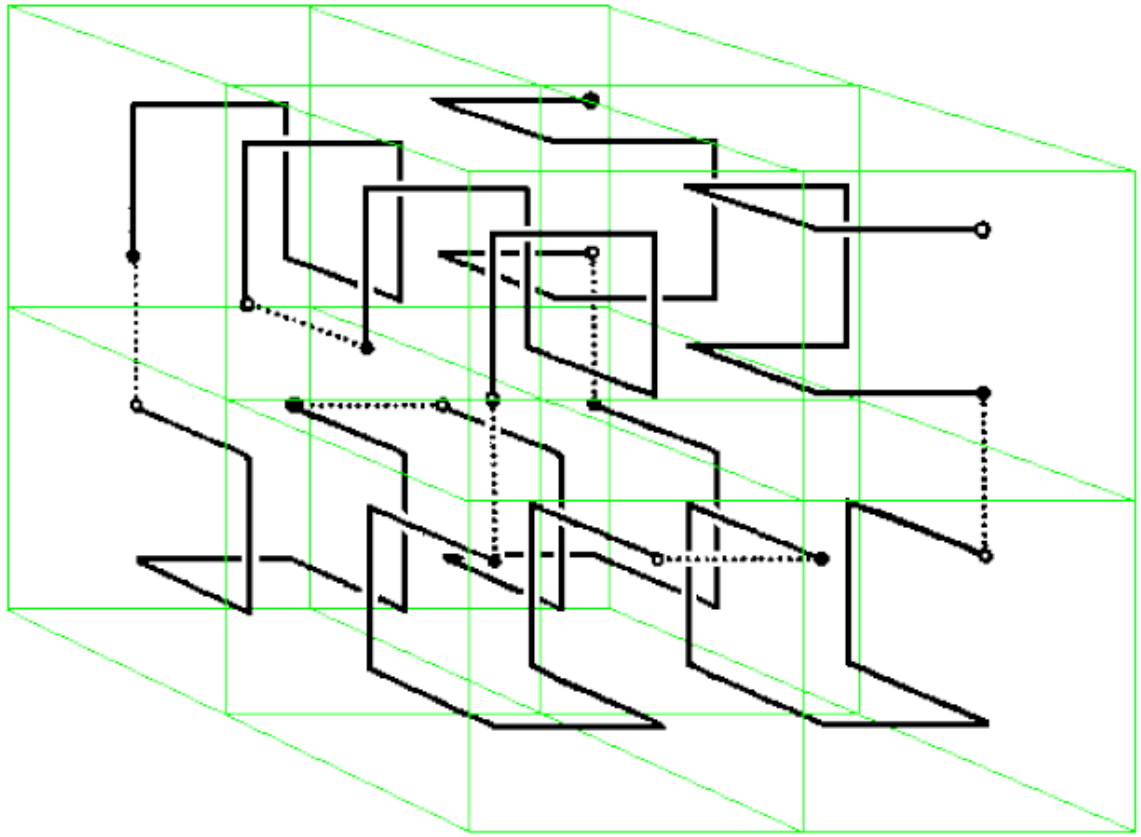
Hilbert Space Filling Curve

- 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

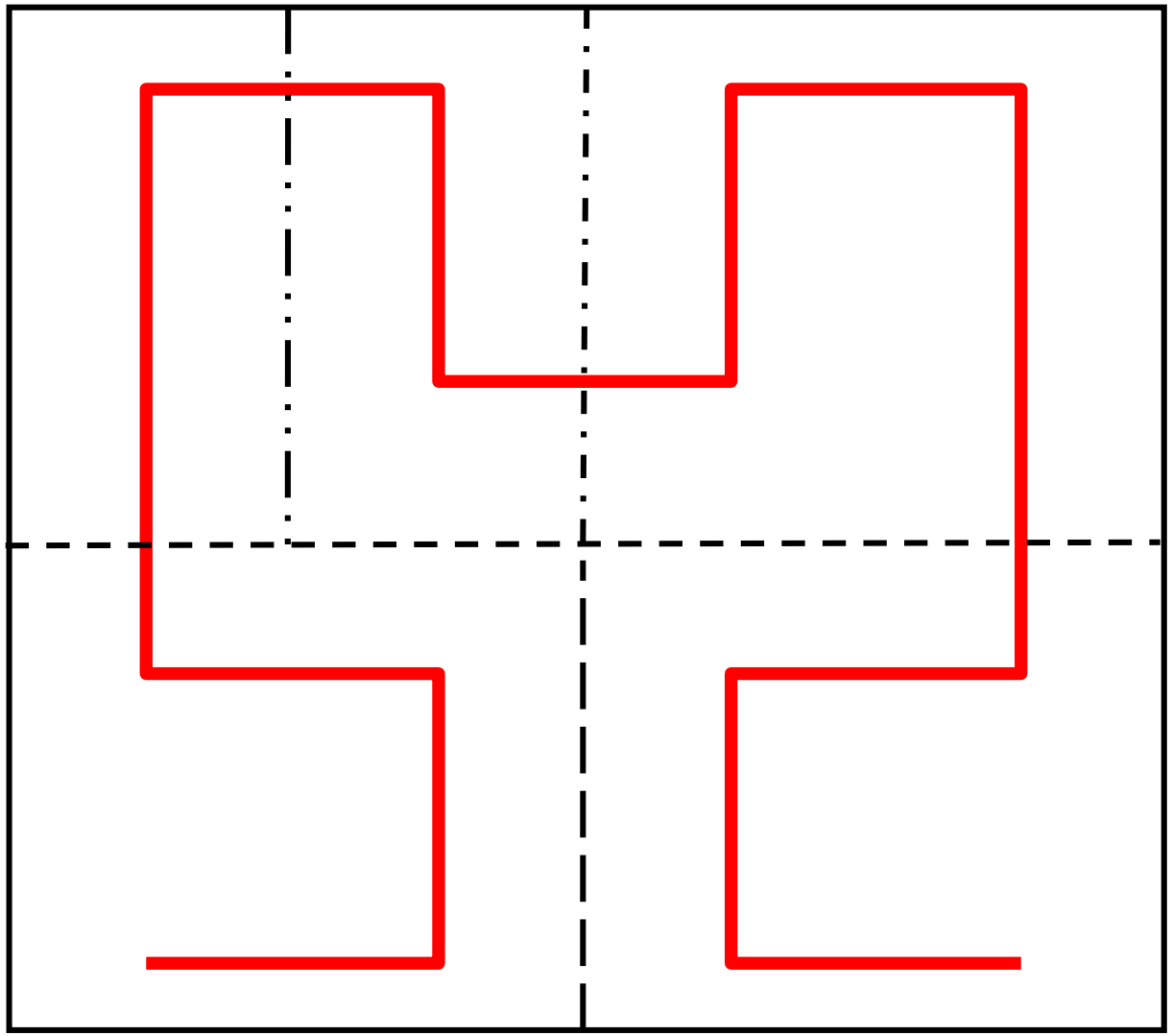
- A continuous fractal space-filling curve (containing the entire 2D unit square)
- Maps 2D and 3D points to a 1D curve
- Maintains spatial locality
- Already used for SPH[5]
- Irregular subdomain shapes (increased complexity of data transfer)



Guo et al. (2015)[7]

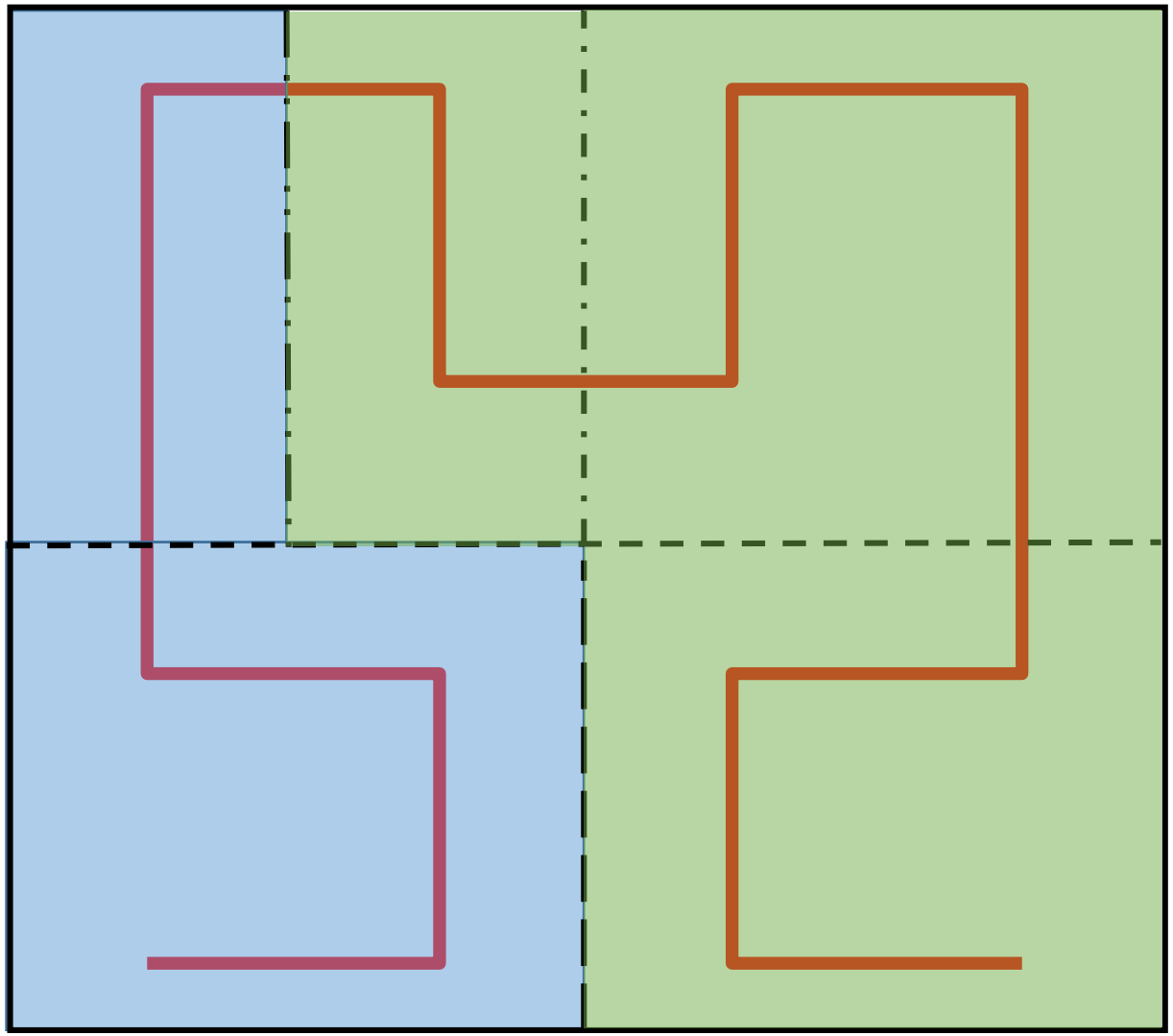
HSFC Algorithm

- 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



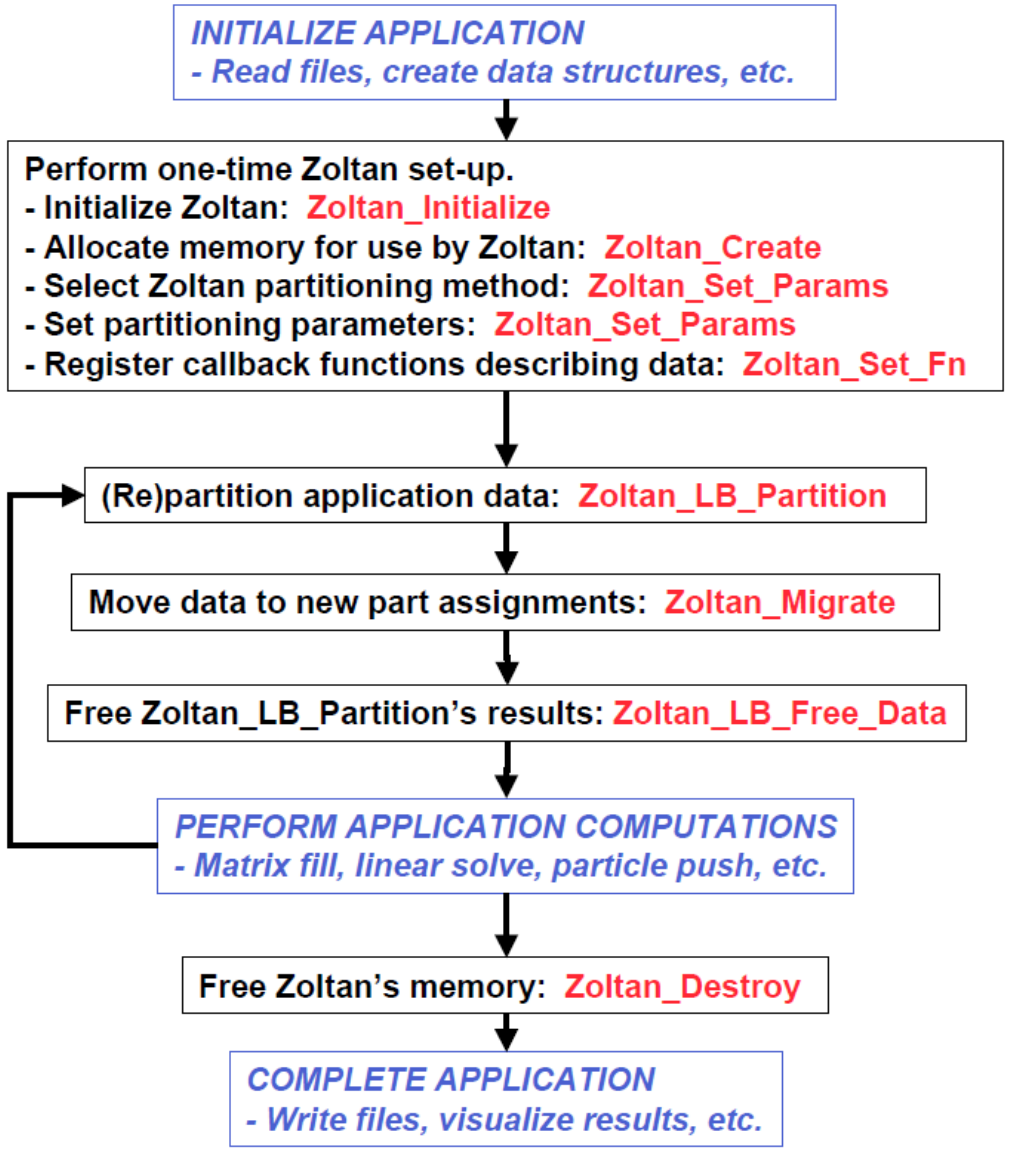
Using Zoltan in DualSPHysics

- 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



Devine et al. (2009)[4]

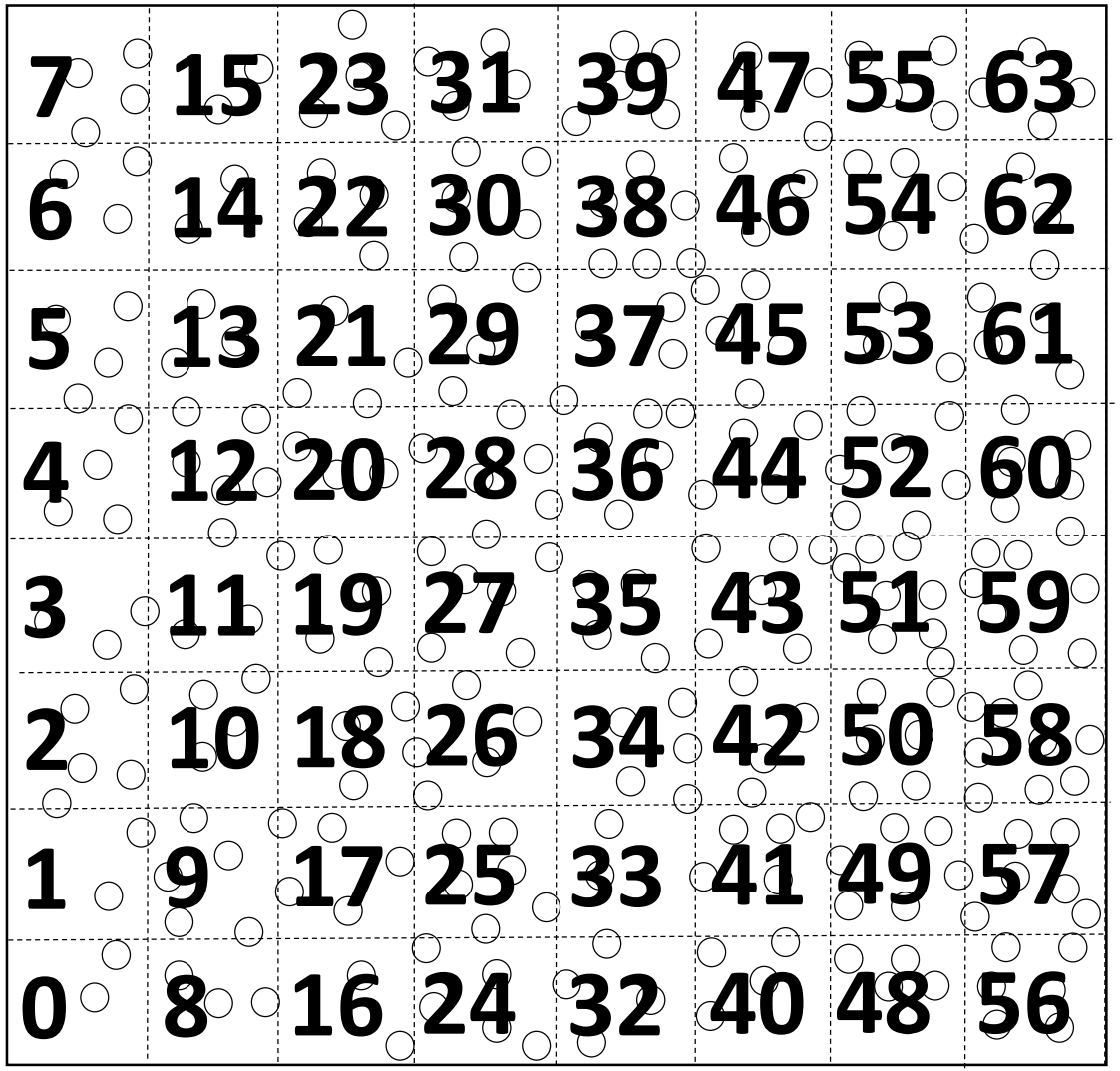
Using Zoltan in DualSPHysics

- 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)



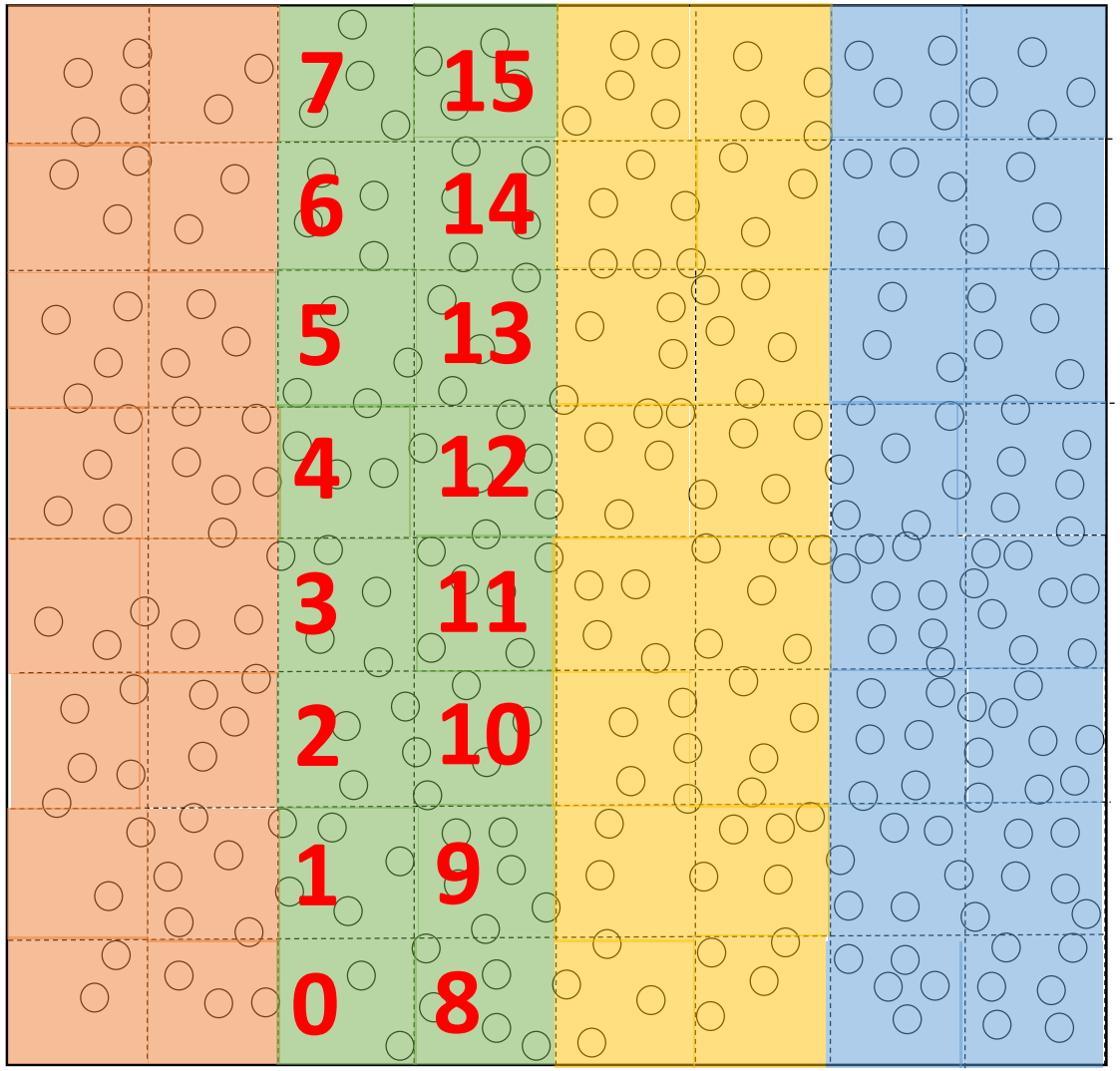
Using Zoltan in DualSPHysics

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

- Each process only holds local data

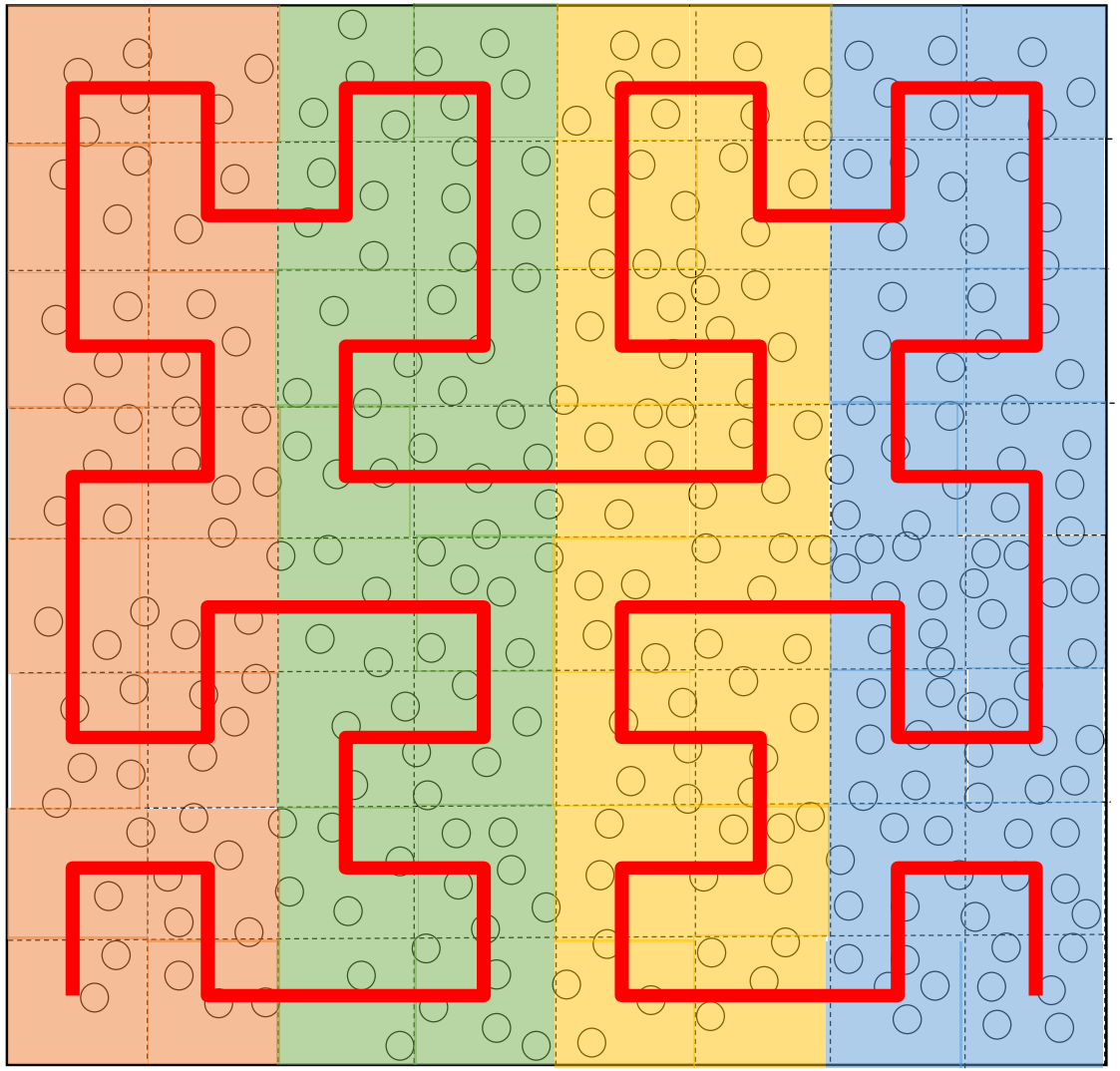
- Example: Domain divided in 64 cells containing 285 particles

- Initial domain split by 1D decomposition (Slices)



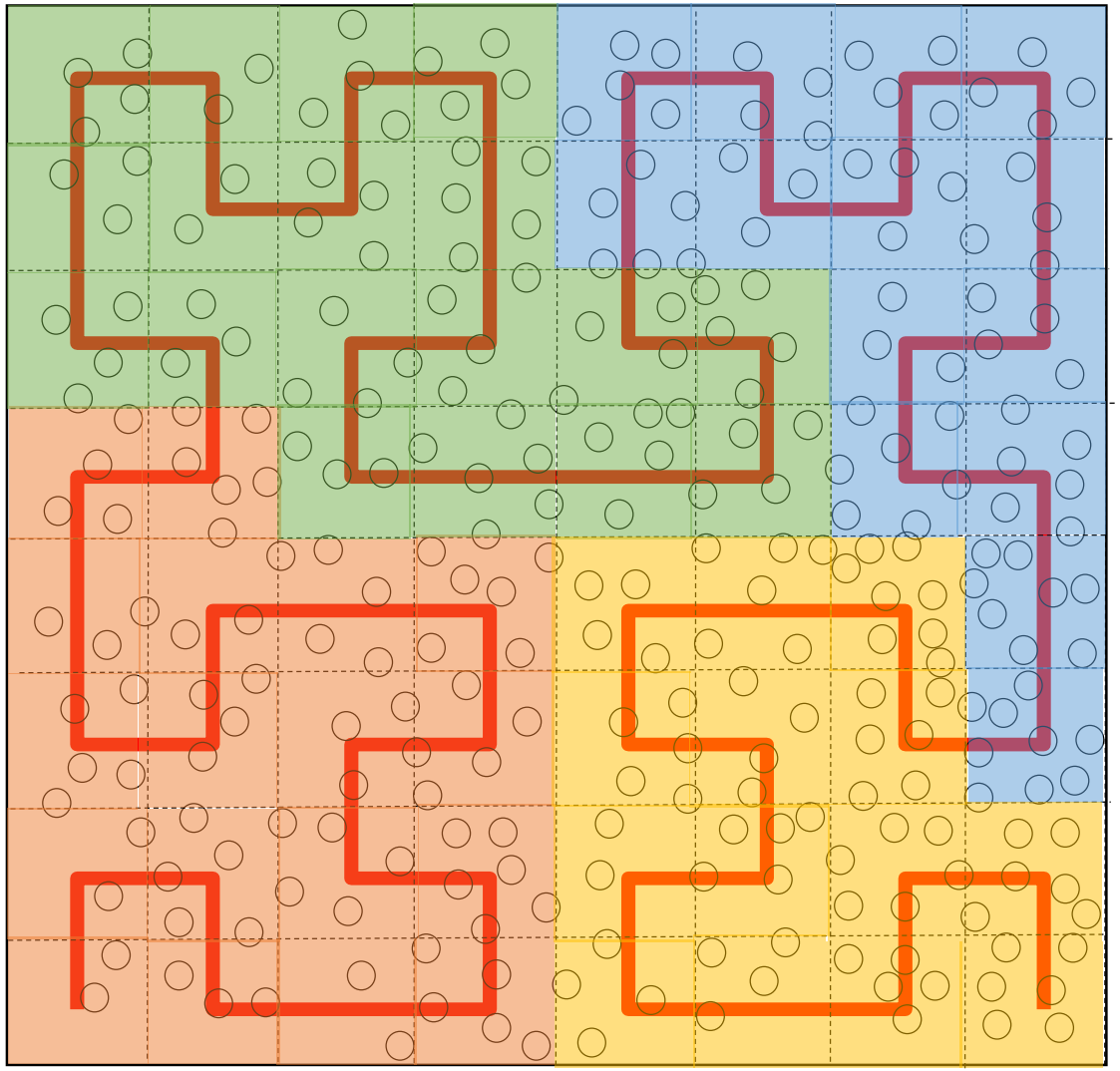
Using Zoltan in DualSPHysics

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

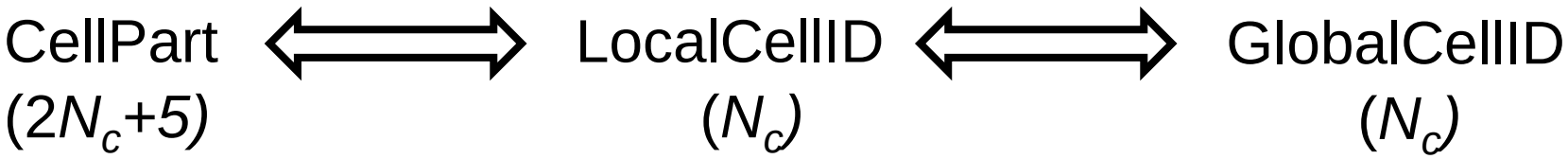
- GlobalCellID is updated:
  - Exported cells removed
  - Imported cells added
- Particles are also imported and exported
- Data reordered creating new cell-linked neighbour list
- LocalCellID is updated
- Algorithm applied only when imbalance exceeds 20%



Particle Mapping

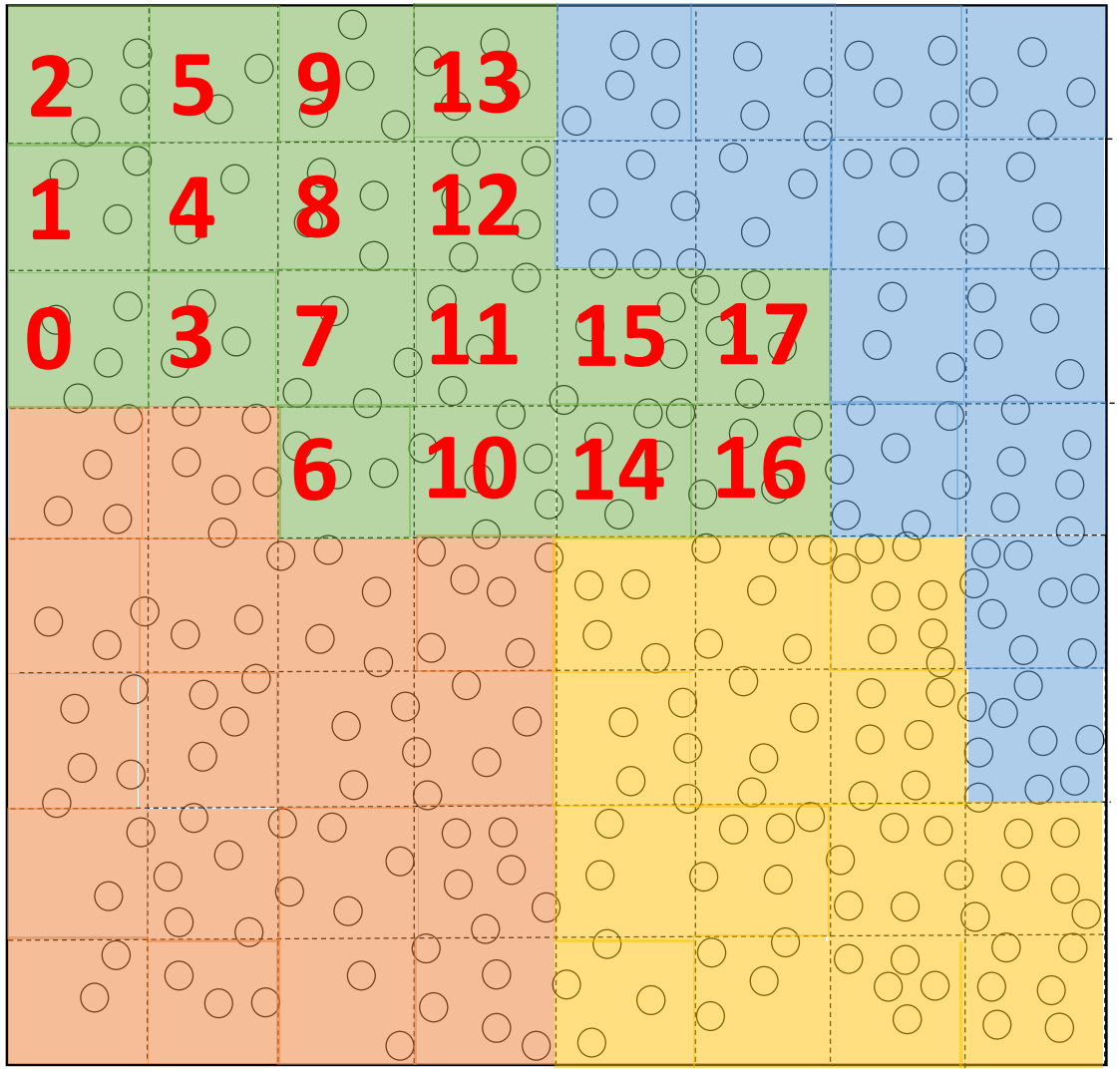
- Connection between cells and particles needed
- Existing DualSPhysics array: **CellPart**
- CellPart can be easily mapped on LocalCellID
- LocalCellID acts as intermediary between CellPart and GlobalCellID
- Not the most elegant solution

If N_c number of local cells



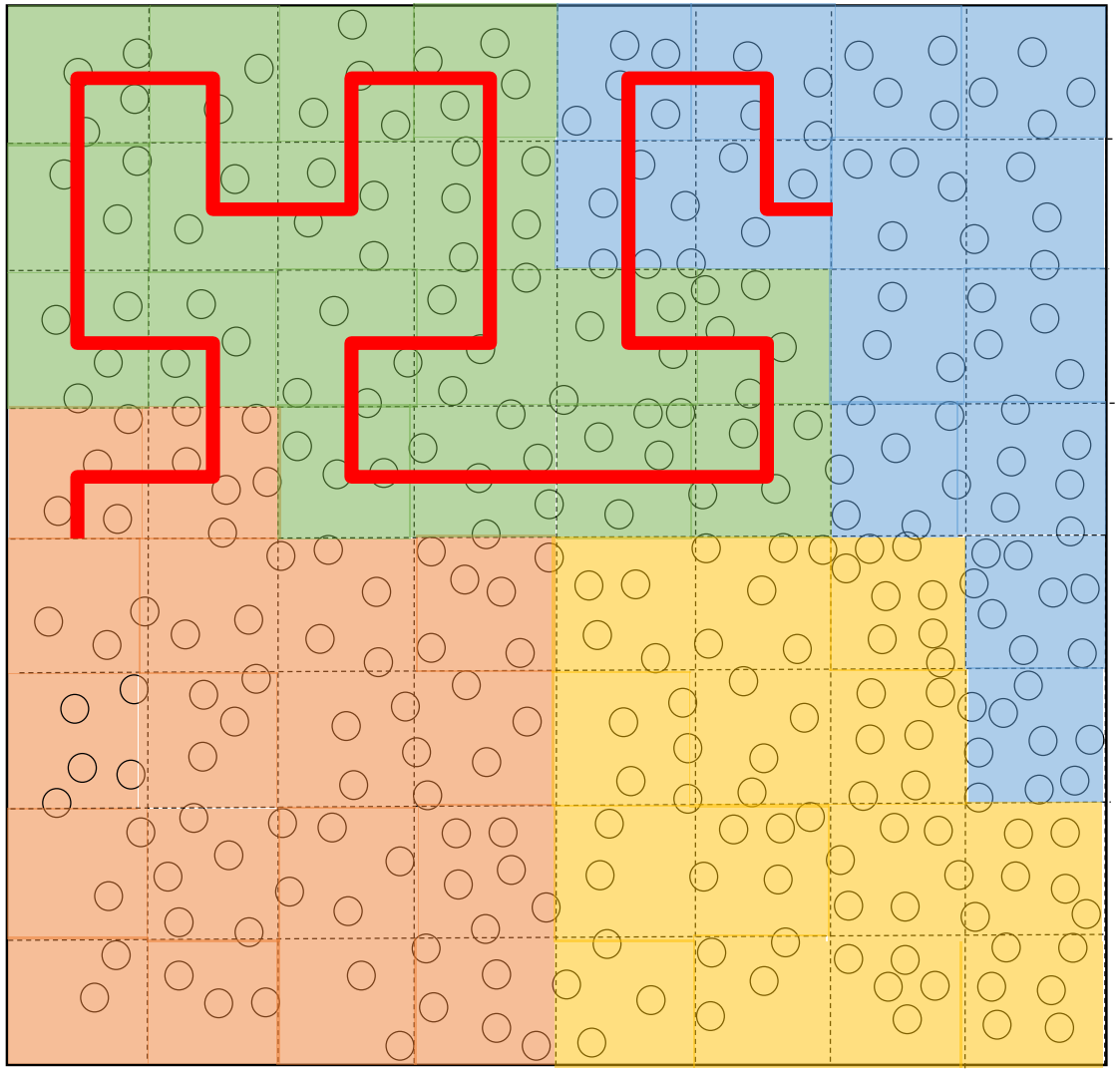
Particle Reordering

- 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]



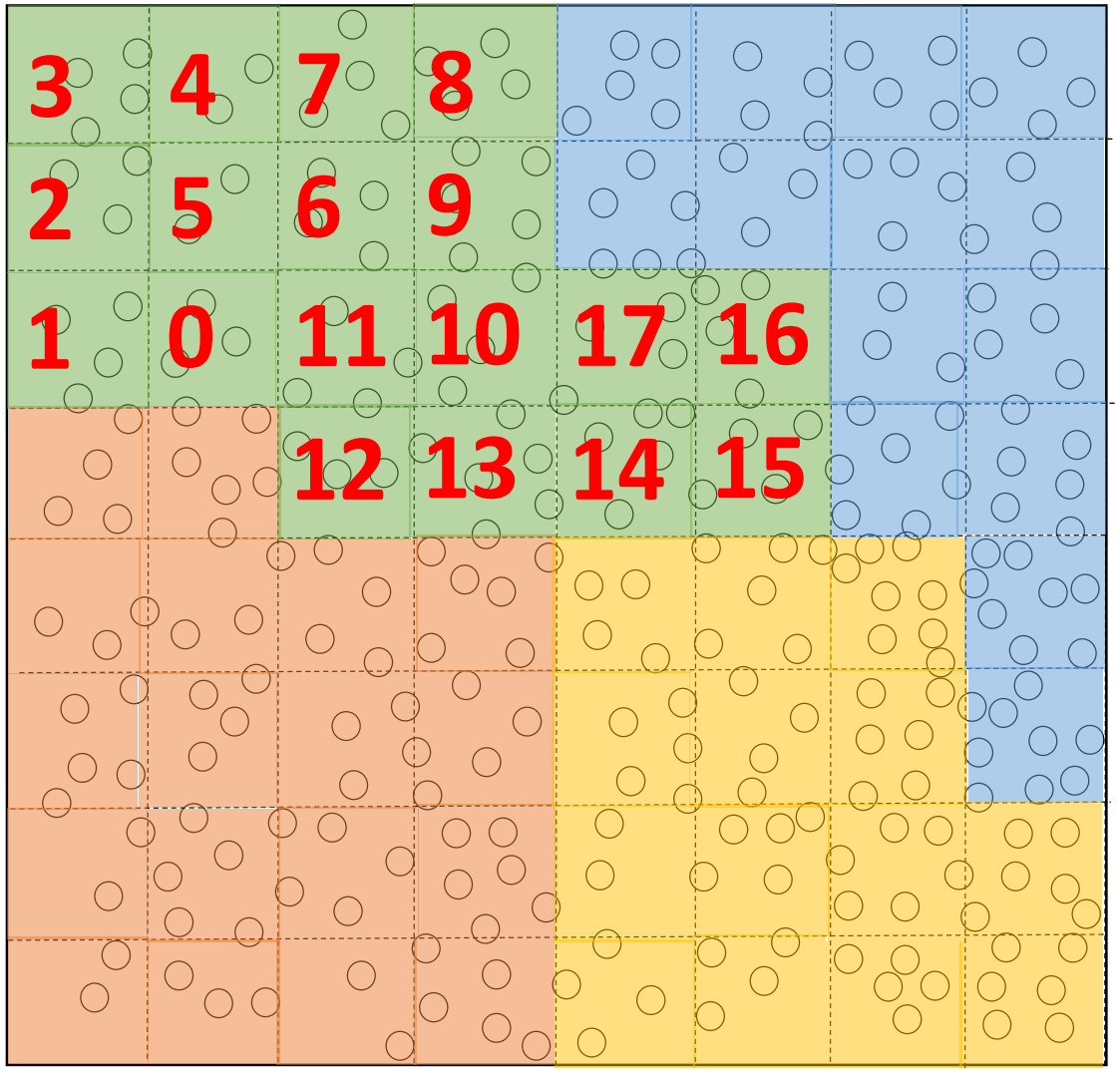
Particle Reordering

- 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

- 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]

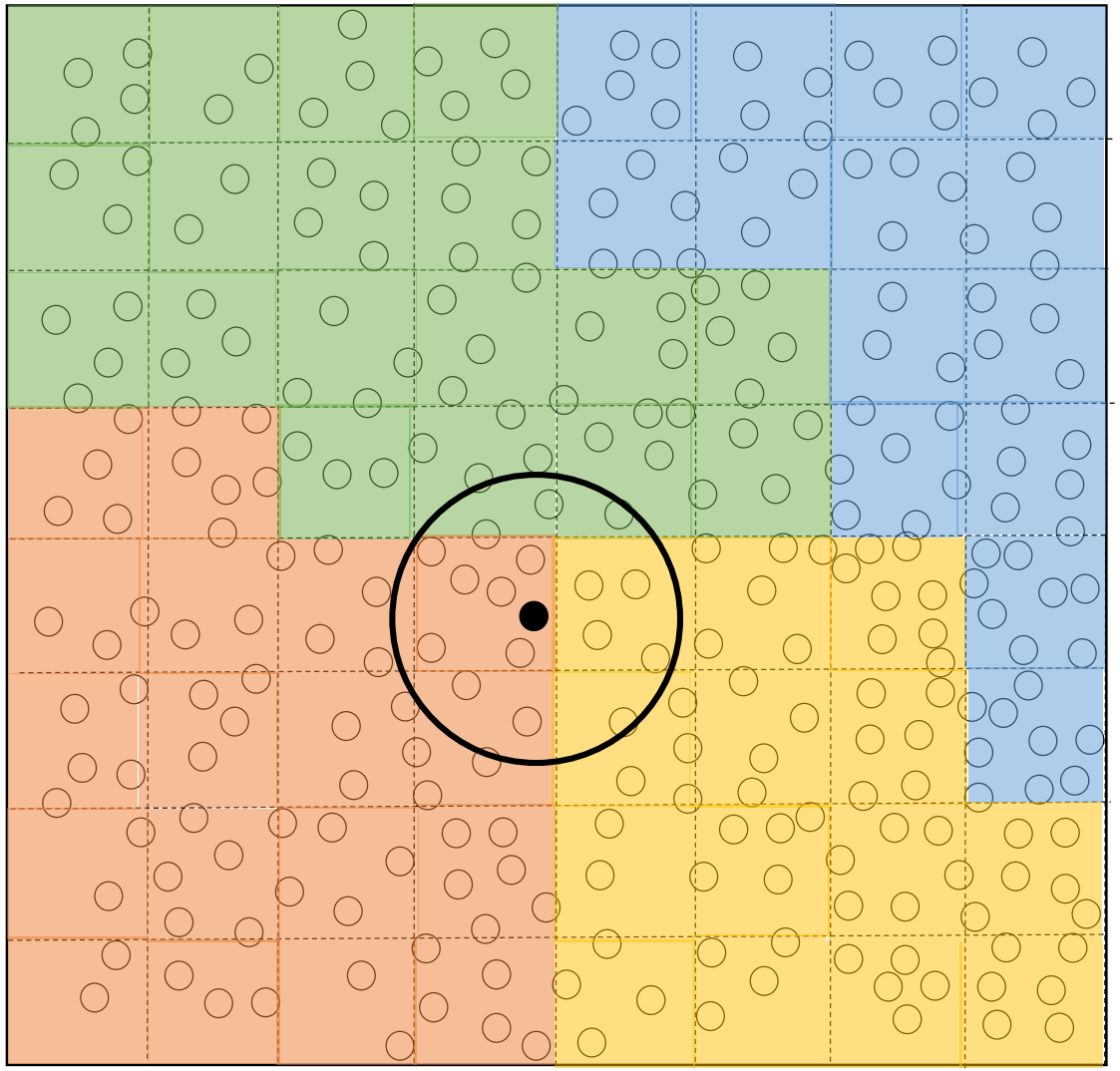


Future Work

- Complete a 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

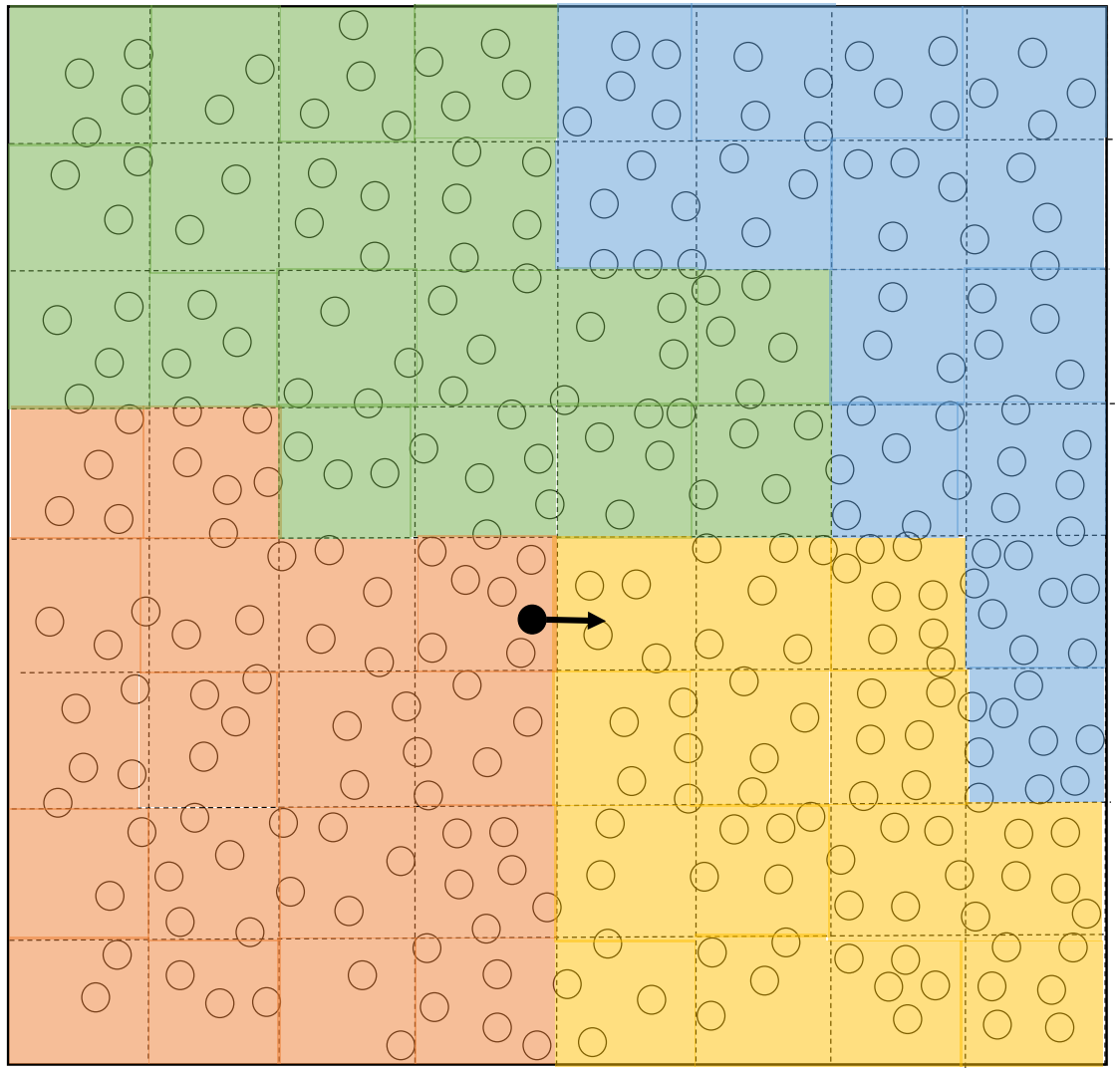
Halo Exchange

- 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

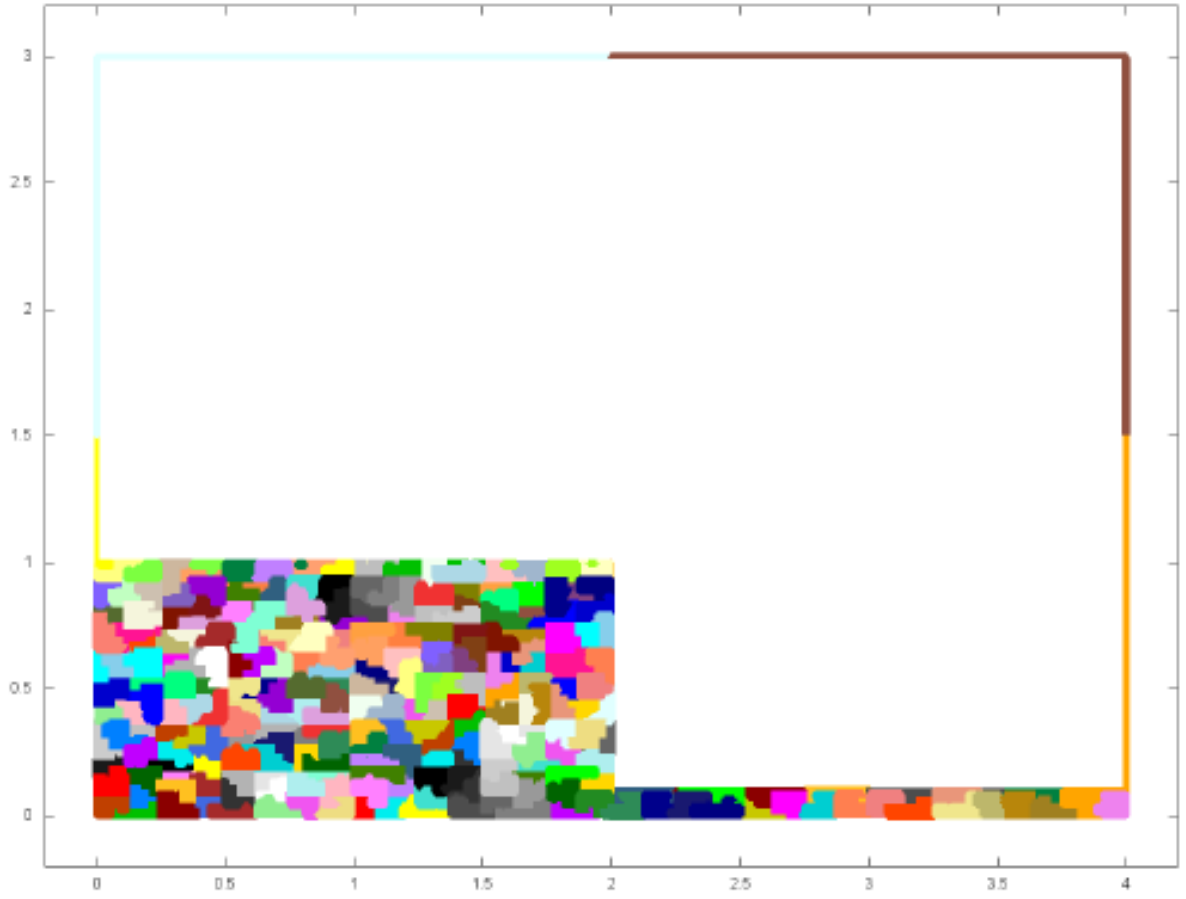


Particle Exchange

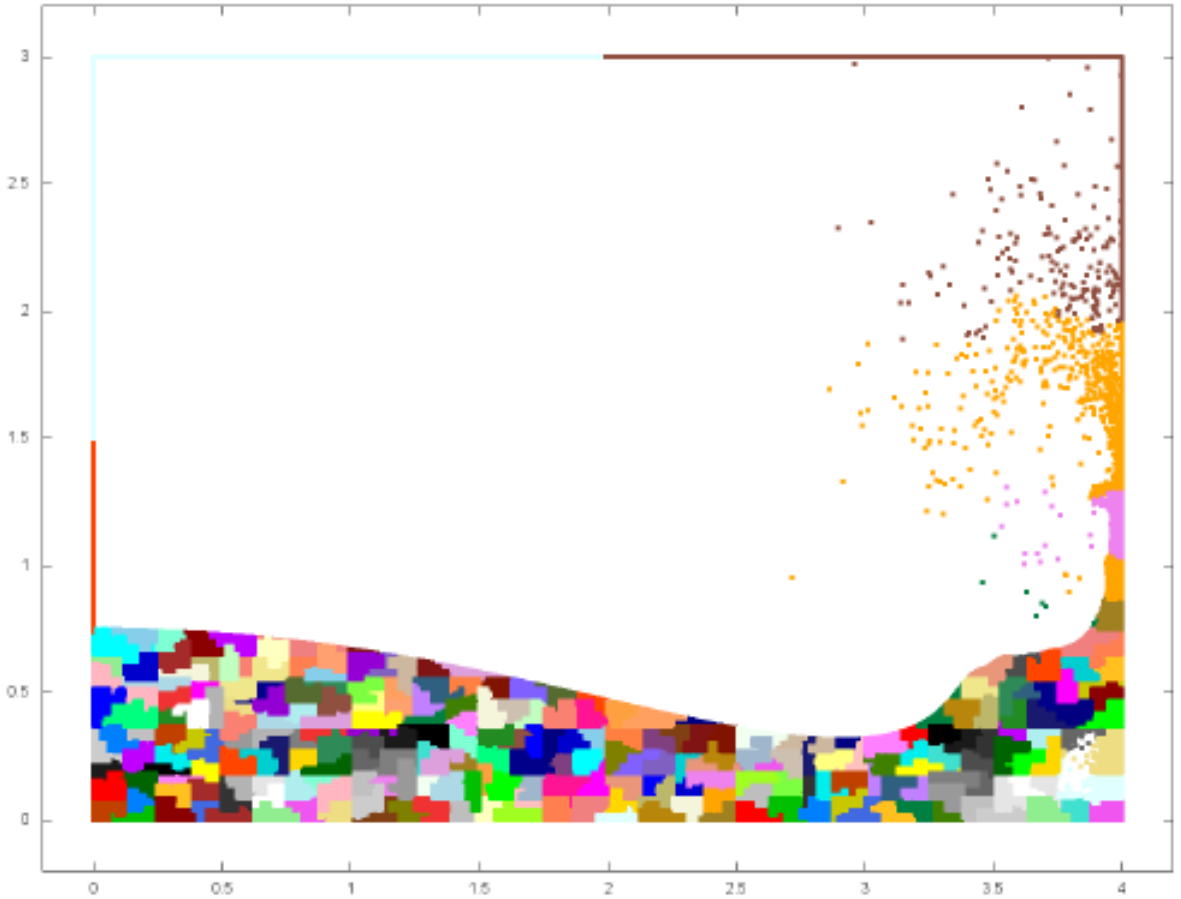
- Particles can move out of the cell
- New cell may be in a different process
- Use Cell coordinates to identify edges of the process' domain
- Identify process and cell the particle moves into
- Use same packing/unpacking algorithm
- Process needs to be completed before reordering particle data



Potential



- Dambreak at 0s for 256 partitions[5]



- Dambreak at 1.1s for 256 partitions[5]

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.

Thank you

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