LIÈGE université

SPH method

Physical engineering master

Multiphysics Integrated Computational Project

Smoothed-Particle Hydrodynamics

Xavier Adriaens, Andrea Attipoe, Sébastien Brialmont
Gabriel Digregorio, Julien Dular, Romain Hanus.

Faculty of Applied Sciences,
University of Liège

June 2017

Introduction of the project

Goal of this project:

- Study the basic principles of the SPH method
- Implement an algorithm to simulate free surface problems with the SPH method
- Programming language used: C++

Description of the implementation:

- Motivation for parallelization
- General strategies for parallel implementation and time integration

Test of the code:

- Several test cases (Dam break, wave propagation, free falling cube of liquid, ...)
- Analyze of physical behaviour and performance of the code

Contents

Contents

Introduction of the method

SPH method is a lagrangian, meshfree method:

- No need to the generation of a mesh
- Particle discretization of the domain
- Meshing a domain is complex (3D geometry, discontinuities,...)

Applications of this method:

- Astrophysics (very first applications)
- Free surface flows
- Gas dynamics
- ...

Contents

Kernel approximation

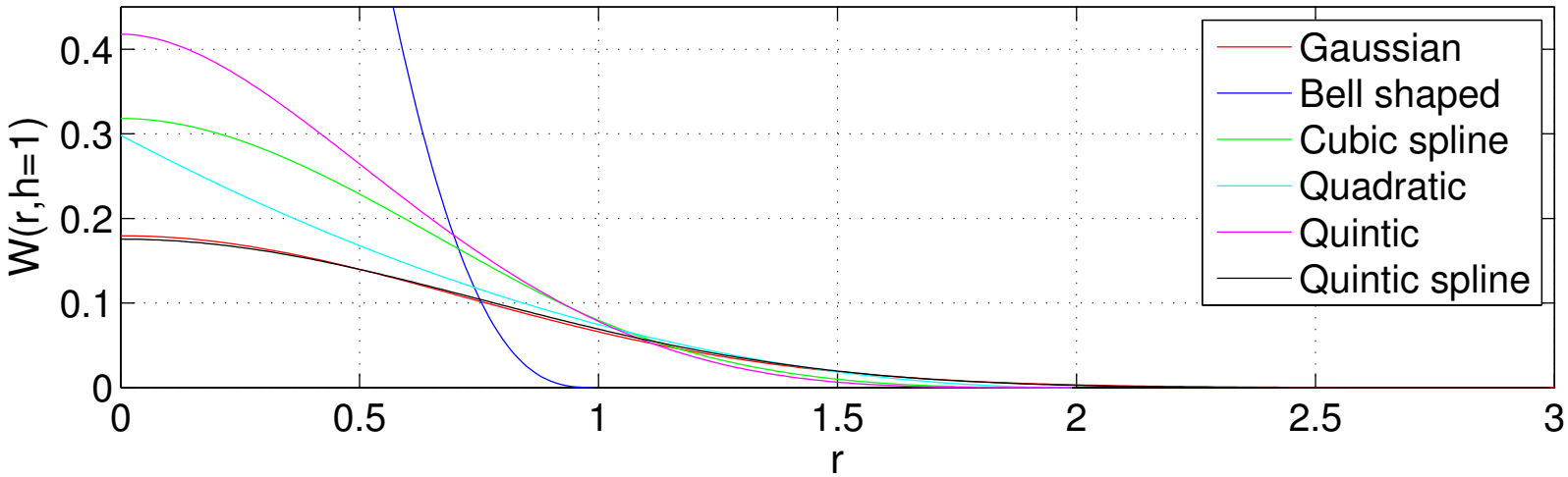
Kernel approximation

- Start from the integral representation

$$f(\boldsymbol{x}) = \int_{\Omega} f(\boldsymbol{x}')\delta(\boldsymbol{x} - \boldsymbol{x}')d\boldsymbol{x}'$$

- Substitute the Dirac function δ by a smoothing function W

$$\langle f(\boldsymbol{x})\rangle = \int_{\Omega} f(\boldsymbol{x}')W(\boldsymbol{x} - \boldsymbol{x}', h)d\boldsymbol{x}'$$

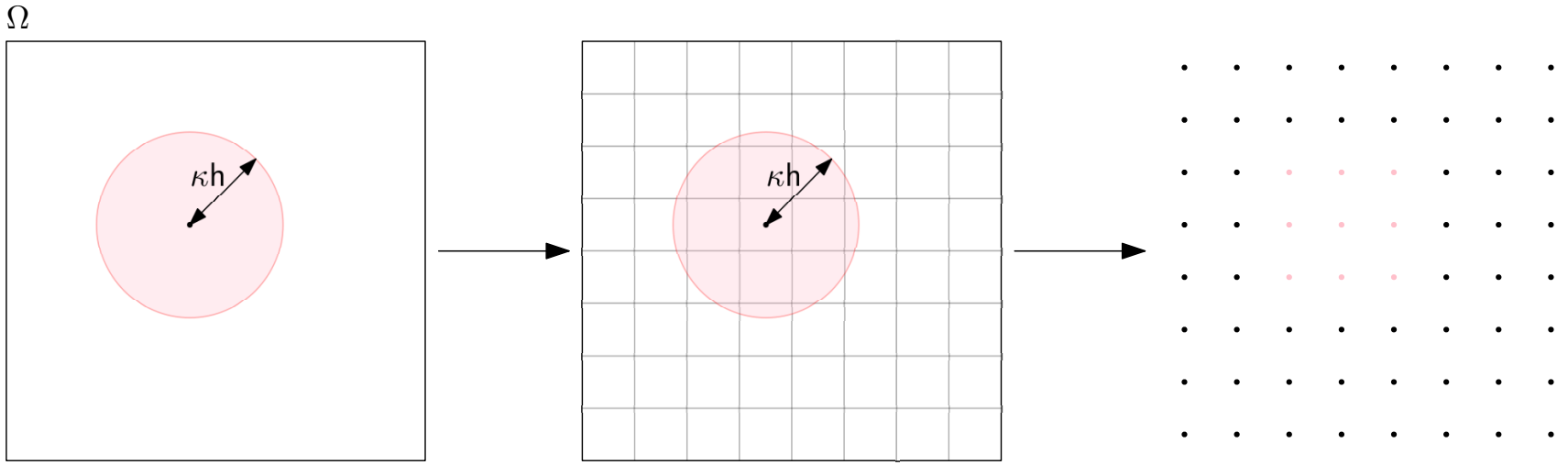


Particle approximation

Particle approximation

Replace the volume integral by a finite sum over new objects called *particles*

$$\langle f(\boldsymbol{x})\rangle = \int_{\Omega} f(\boldsymbol{x}')W(\boldsymbol{x}-\boldsymbol{x}',h) \approx \sum_{j=1}^{N_i} f(\boldsymbol{x}_j)W(\boldsymbol{x}_i-\boldsymbol{x}_j,h)\frac{m_j}{\rho_j}$$



Smoothing length influence

The smoothing length should be of the order of the initial spacing between particles s

Large smoothing length $\Rightarrow$ Many neighbors but high computational cost

Play/Pause | 5 neighbors | 30 neighbors

Conservation equations

Mass conservation

$$\frac{D\rho}{Dt} = -\rho \nabla \cdot \boldsymbol{v}$$

Kernel & particle approximations ⇒

$$\frac{D\rho_i}{dt} = \sum_{j=1}^{N} m_j \boldsymbol{v}_{ij} \cdot \nabla_i W_{ij}$$

Linear momentum conservation

$$\frac{D\boldsymbol{v}}{Dt} = \frac{1}{\rho} \nabla \cdot \boldsymbol{\sigma} + \boldsymbol{g}$$

Kernel & particle approximations ⇒

$$\frac{D\boldsymbol{v}_i}{Dt} = \sum_{j=1}^{N} m_j \left(\frac{\boldsymbol{\sigma}_i}{\rho_i^2} + \frac{\boldsymbol{\sigma}_j}{\rho_j^2} \right) \cdot \nabla_i W_{ij} + \boldsymbol{g}_i$$

Velocity definition

$$\frac{D\boldsymbol{x}}{Dt} = \boldsymbol{v}$$

Kernel & particle approximations ⇒

$$\frac{D\boldsymbol{x}_i}{Dt} = \boldsymbol{v}_i + \varepsilon \sum_{j=1}^{N} \frac{m_j}{\rho_j} \boldsymbol{v}_{ji} W_{ij}$$

Position update

Comparison between the SPH (in green) and XSPH ($\varepsilon = 0.25$) (in red) methods

Play/Pause

$\Rightarrow$ Results are globally identical

Constitutive equations

$$\sum_{j=1}^{N} m_j \left(\frac{\boldsymbol{\sigma}_i}{\rho_i^2} + \frac{\boldsymbol{\sigma}_j}{\rho_j^2} \right) \cdot \boldsymbol{\nabla}_i W_{ij} + \boldsymbol{g}_i \longrightarrow - \sum_{j=1}^{N} m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} + \Pi_{ij} \right) \boldsymbol{\nabla}_i W_{ij} + \boldsymbol{g}_i$$

Artificial viscosity Π_{ij}

$$\Pi_{ij} \begin{cases} \neq 0 & \boldsymbol{v}_{ij} \cdot \boldsymbol{x}_{ij} < 0, \\ = 0 & \boldsymbol{v}_{ij} \cdot \boldsymbol{x}_{ij} > 0 \end{cases}$$

- Produce shear and bulk viscosity
- Provide numerical stability

Pressure p

$$p = B \left(\left(\frac{\rho}{\rho_0} \right)^{\gamma} - 1 \right)$$

- Weakly compressible fluids equation of state

Viscosity influence

Play/Pause | 30 neighbors | 30 neighbors and less viscous

Integration schemes

$$\frac{D\boldsymbol{V}}{Dt} = \boldsymbol{F}(\boldsymbol{V}(t))$$

General form of the semi-discrete equations

Explicit Euler

$$\boldsymbol{V}_{t_0+k} = \boldsymbol{V}_{t_0} + k\boldsymbol{F}(\boldsymbol{V}(t_0))$$

- Small time step k is required for stability

Two steps Runge-Kutta

$$\boldsymbol{V}^* = \boldsymbol{V}_{t_0} + \frac{k}{2\theta}\boldsymbol{F}(\boldsymbol{V}_{t_0})$$

$$\boldsymbol{V}_{t_0+k} = \boldsymbol{V}_{t_0} + k\left((1-\theta)\boldsymbol{F}(\boldsymbol{V}_{t_0}) + \theta\boldsymbol{F}(\boldsymbol{V}^*)\right)$$

- Require two configurations computation
- Larger time step possible

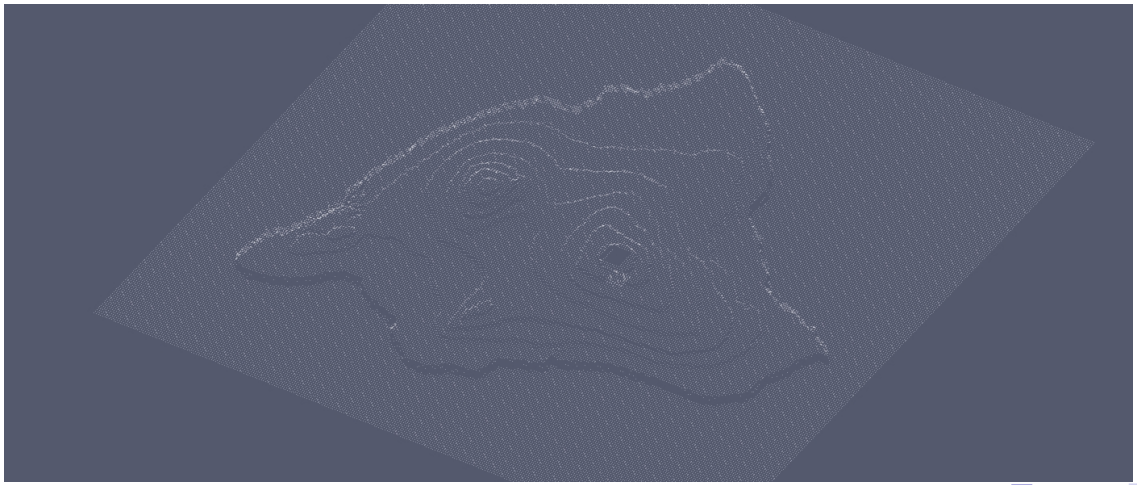
Boundary conditions

Dynamic boundary conditions

Model the boundaries with particles whose velocity and position is imposed.

Implemented boundary shape and movement

- Box of particles with translation or rotation movement
- Bathymetry *i.e* plane of equally spaced (in the $x - y$ plane) particles with a user-defined height (z)



Boundary conditions

Moving boundaries example

Play/Pause

Contents

Motivation for a parallel implementation

Sequential implementation

The computation time can become very high for large numbers of particles

Motivation for a parallel implementation

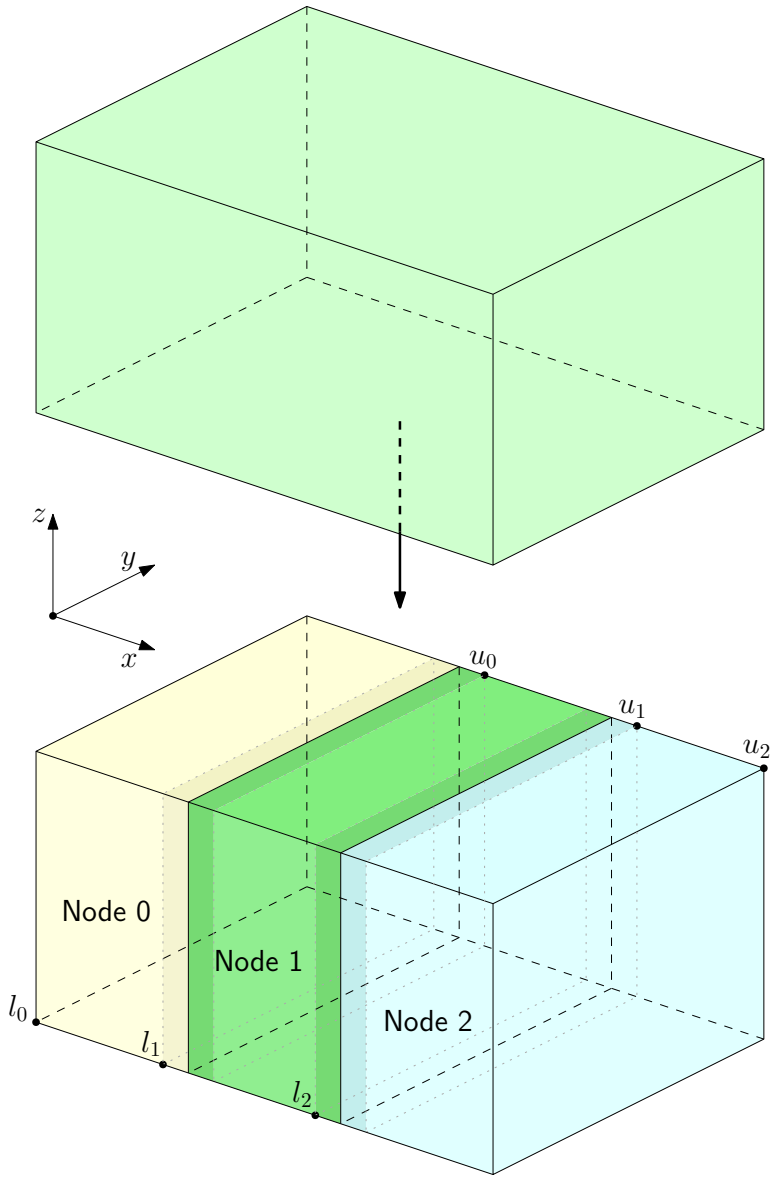
Sequential implementation

The computation time can become very high for large numbers of particles

Parallel implementation

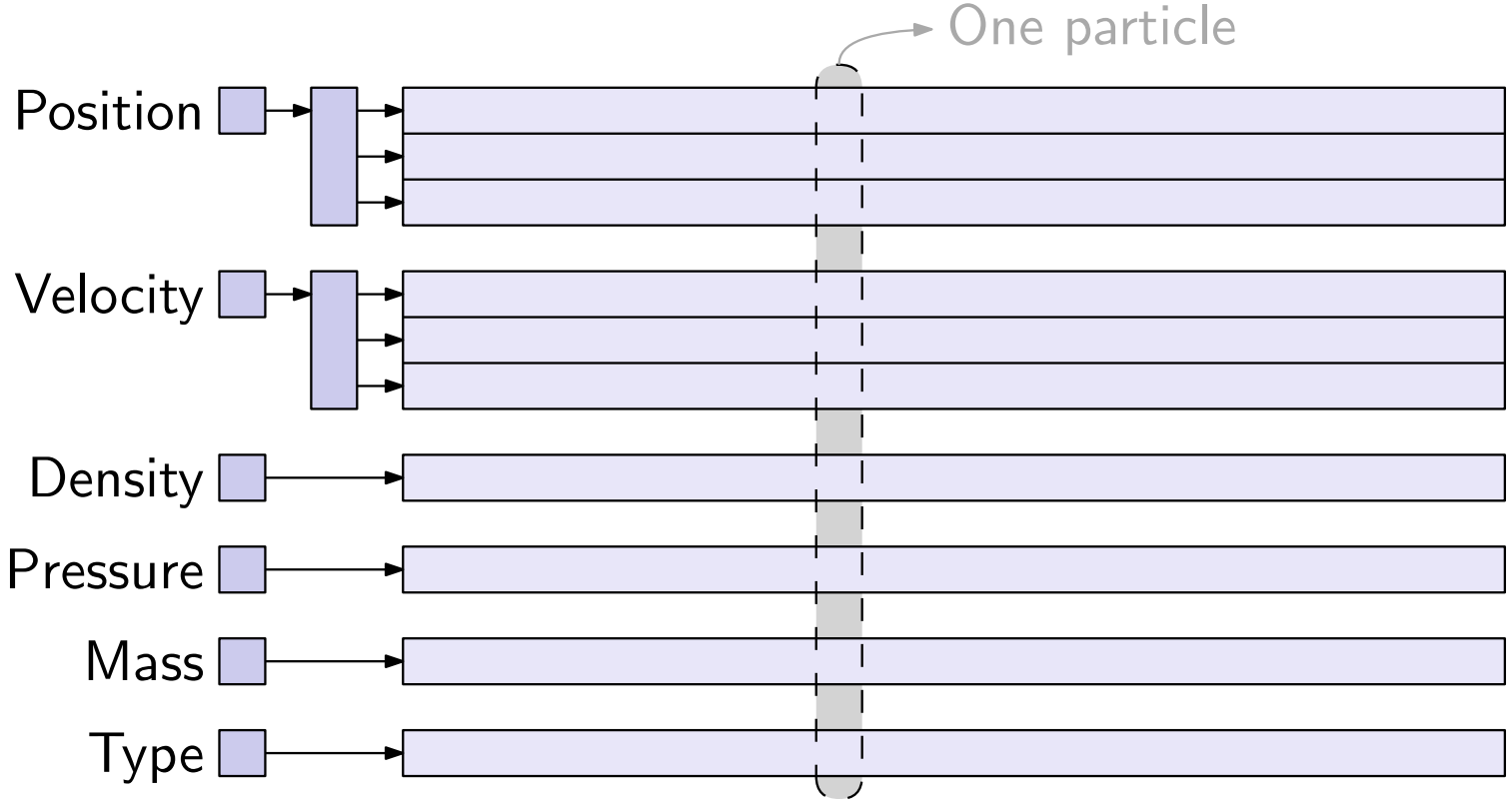
Share the computational load among the processors

- Cut the physical domain in slices
- Overlaps and communication are necessary



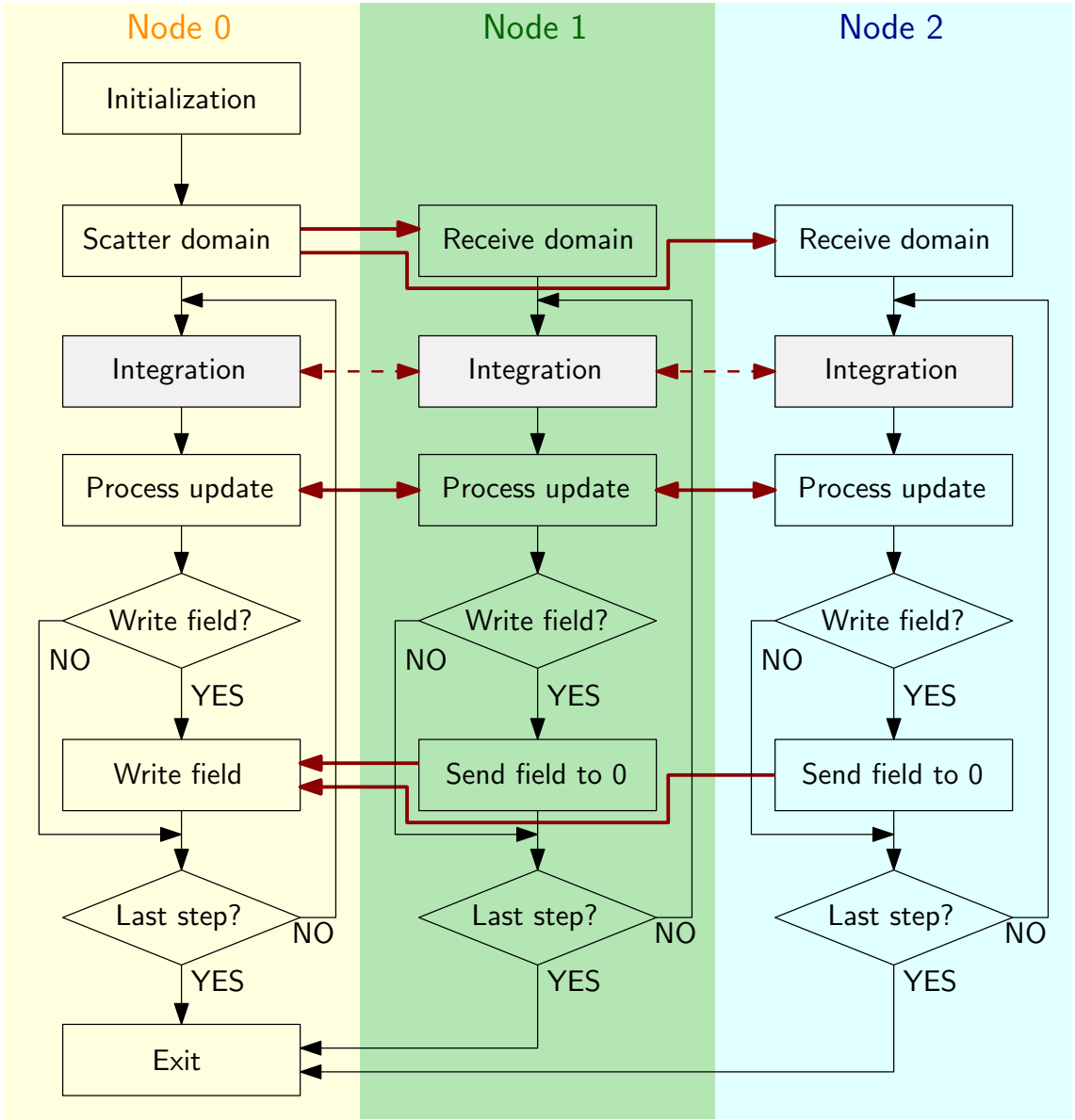
Data organization

- Particle variables are stored in vectors, **one vector per variable**

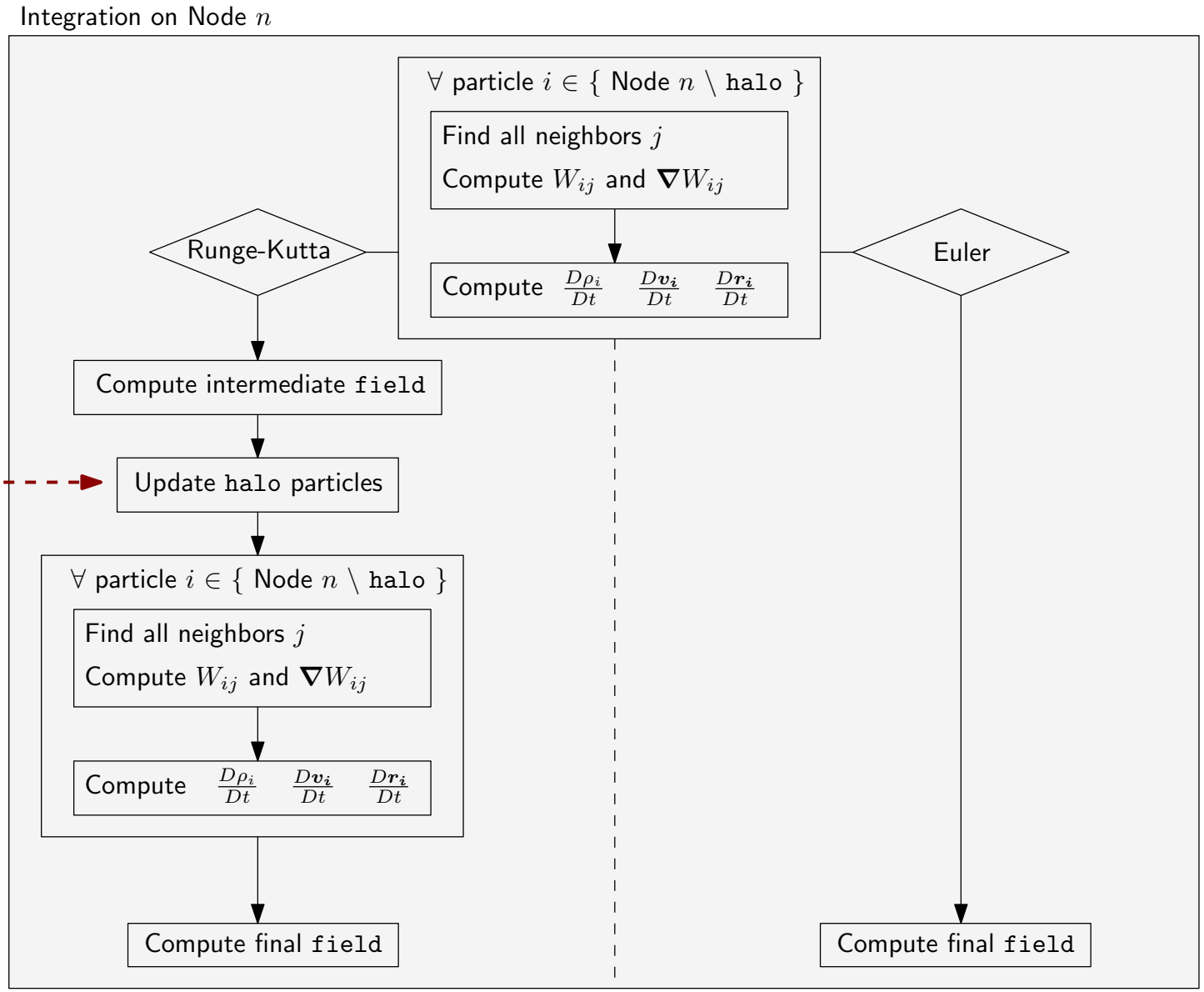


- This choice simplifies the distributed memory parallelization

General flowchart



Time integration

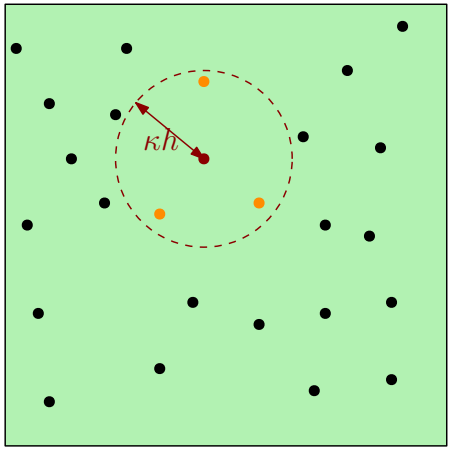


Neighbor search algorithms

Different search algorithms are possible

Neighbor search algorithms

Different search algorithms are possible

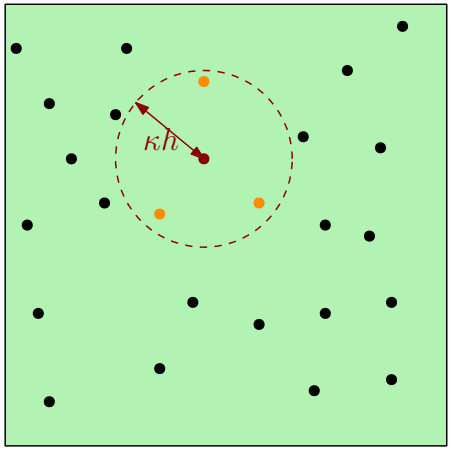


- Naive implemtation
- $\mathcal{O}(N^2)$ complexity

All-pair search

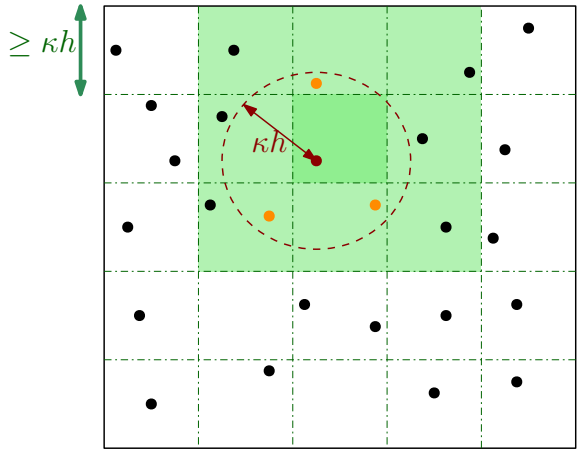
Neighbor search algorithms

Different search algorithms are possible



- Naive implemtation
- $\mathcal{O}(N^2)$ complexity

All-pair search

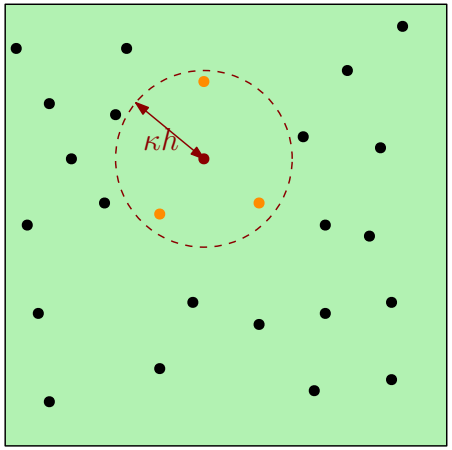


- Constant smoothing length κh
- $\mathcal{O}(N)$ complexity

Linked-list

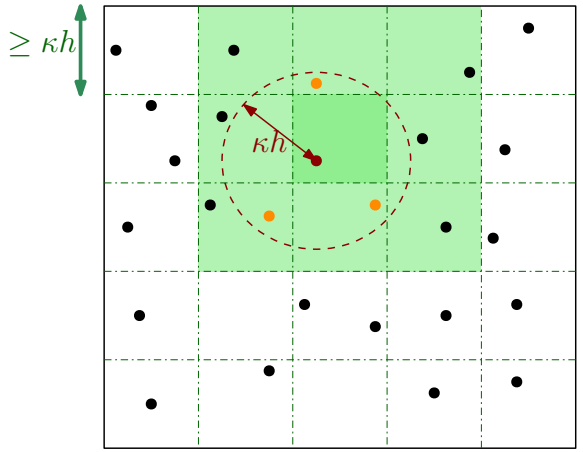
Neighbor search algorithms

Different search algorithms are possible



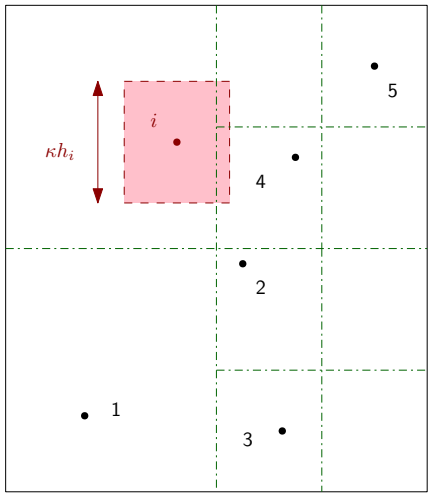
- Naive implemtation
- $\mathcal{O}(N^2)$ complexity

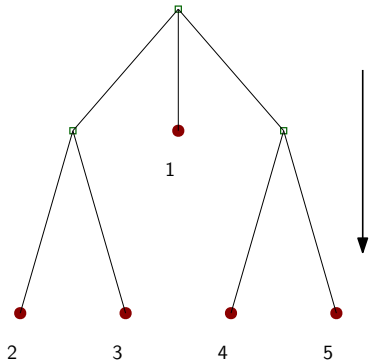
All-pair search



- Constant smoothing length κh
- $\mathcal{O}(N)$ complexity

Linked-list





- Variable smoothing length κh
- $\mathcal{O}(N \log(N))$ complexity

Tree search

Neighbor search algorithms

Implemented algorithm

The linked-list search:

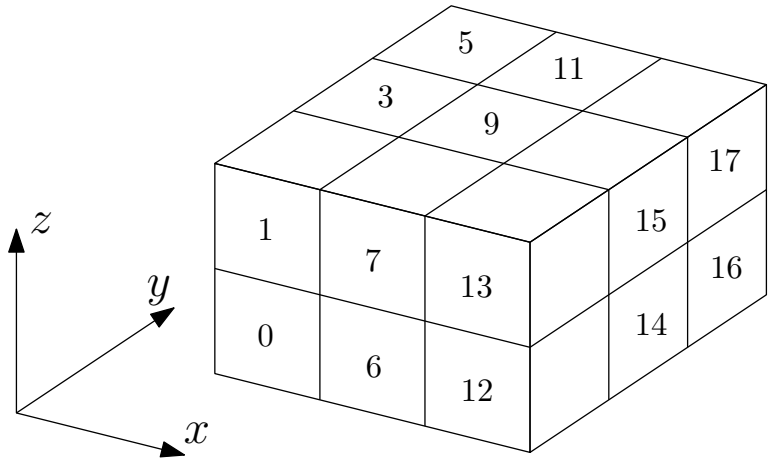
- constant smoothing length
- easier parallelization

Implementation

-List of particles in each boxes.
-Two nested loops, one for the boxes, one for the particles inside each boxes
-Storage of the neighbors of 2 kinds:

- Incidence matrix to use symmetry
- Own neighbor storage list

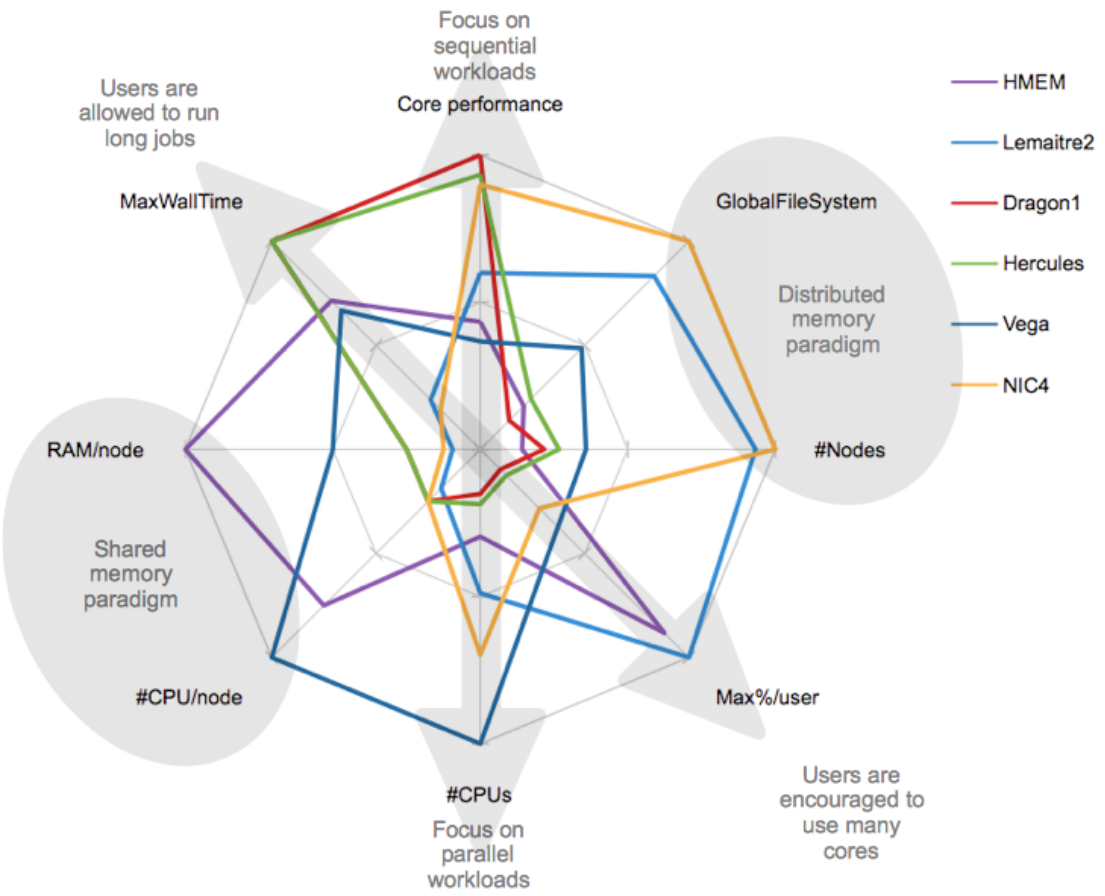
-Use of the known distance to compute kernel



Computer clusters

NIC4 features

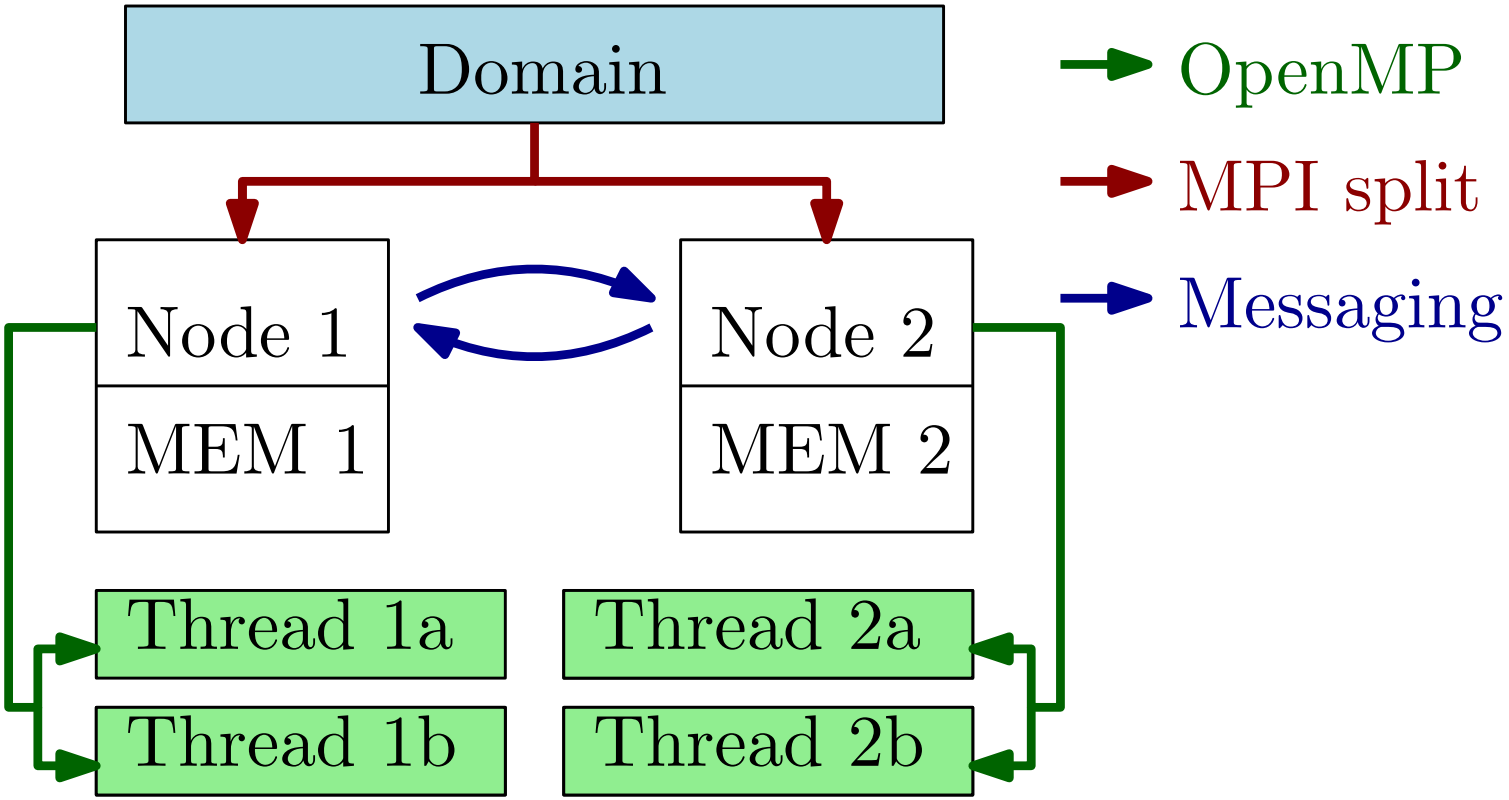
- 120 computing nodes, 64GB of RAM each
- 144TB parallel file system



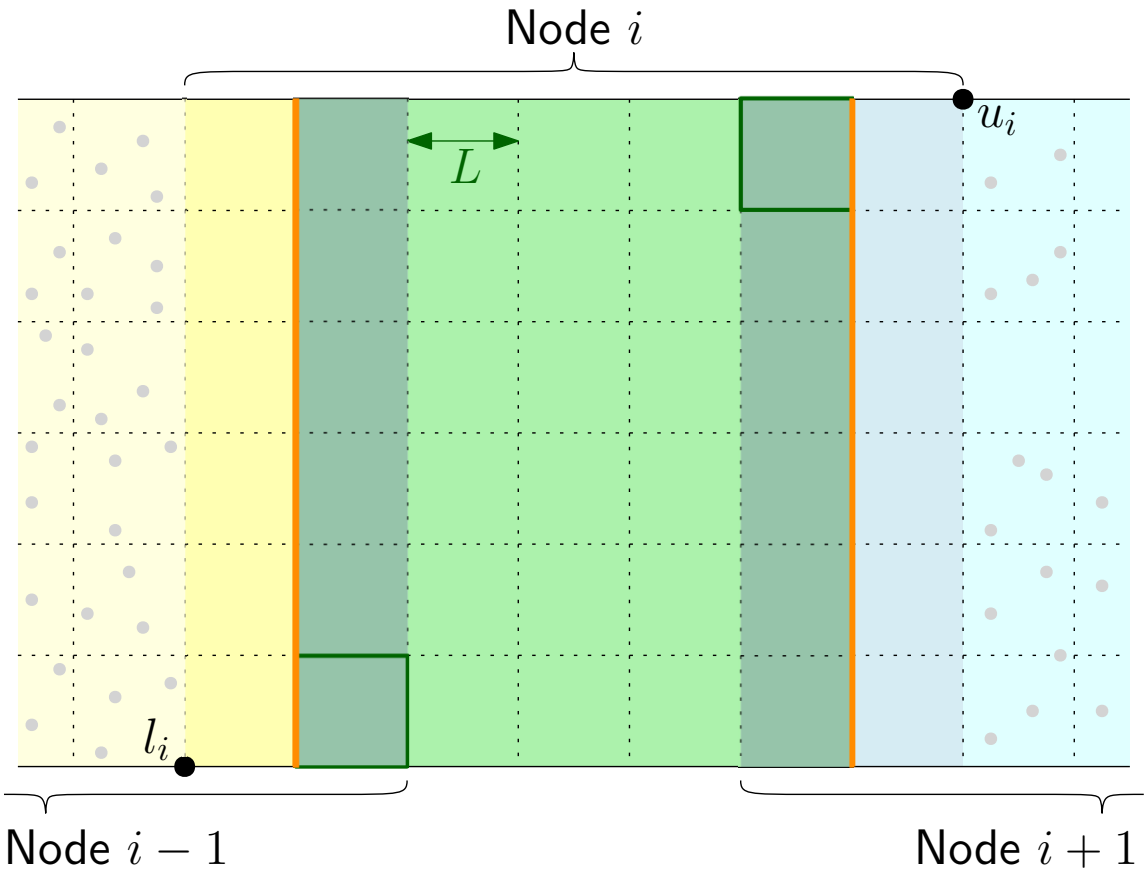
©CECI

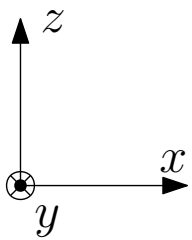
Parallelization approaches

Distributed or shared memory parallelization:



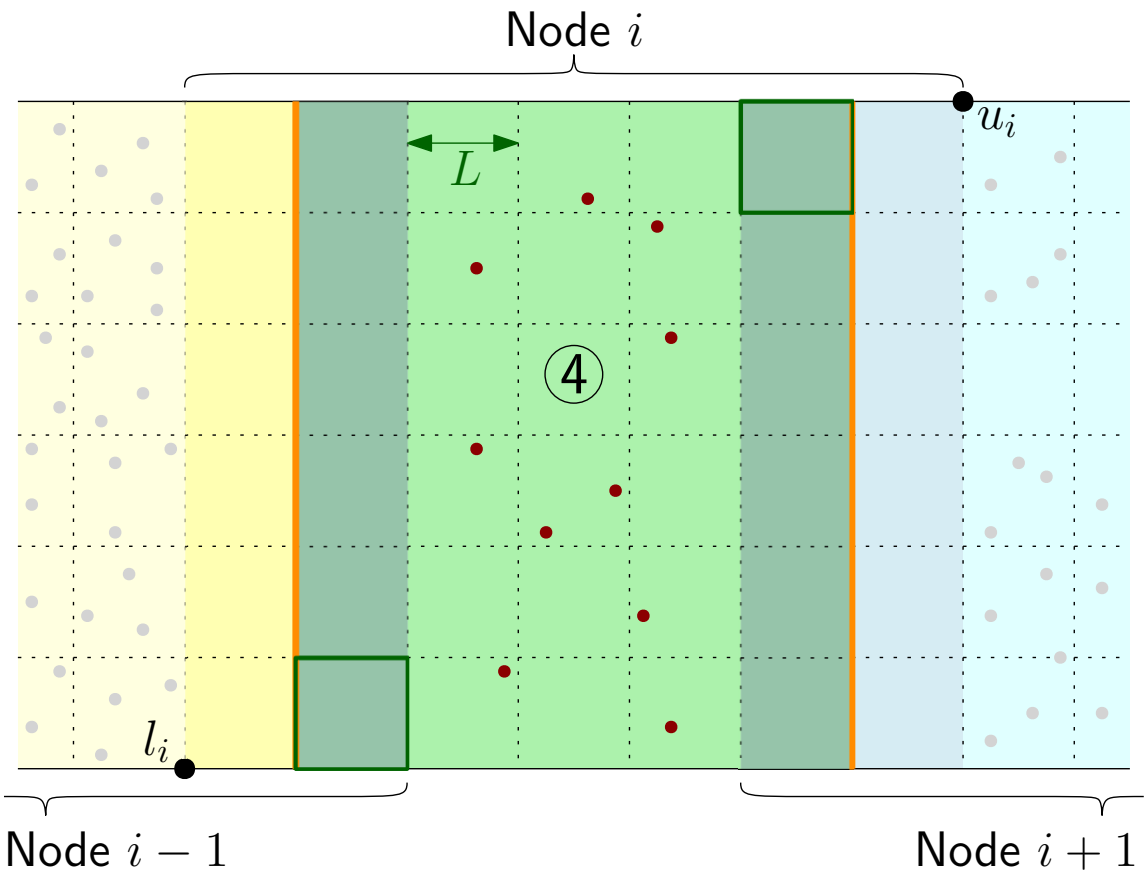
Particles sorting

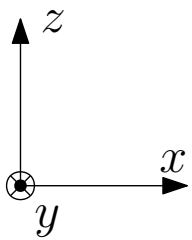




Particles in node i ( )

Particles sorting

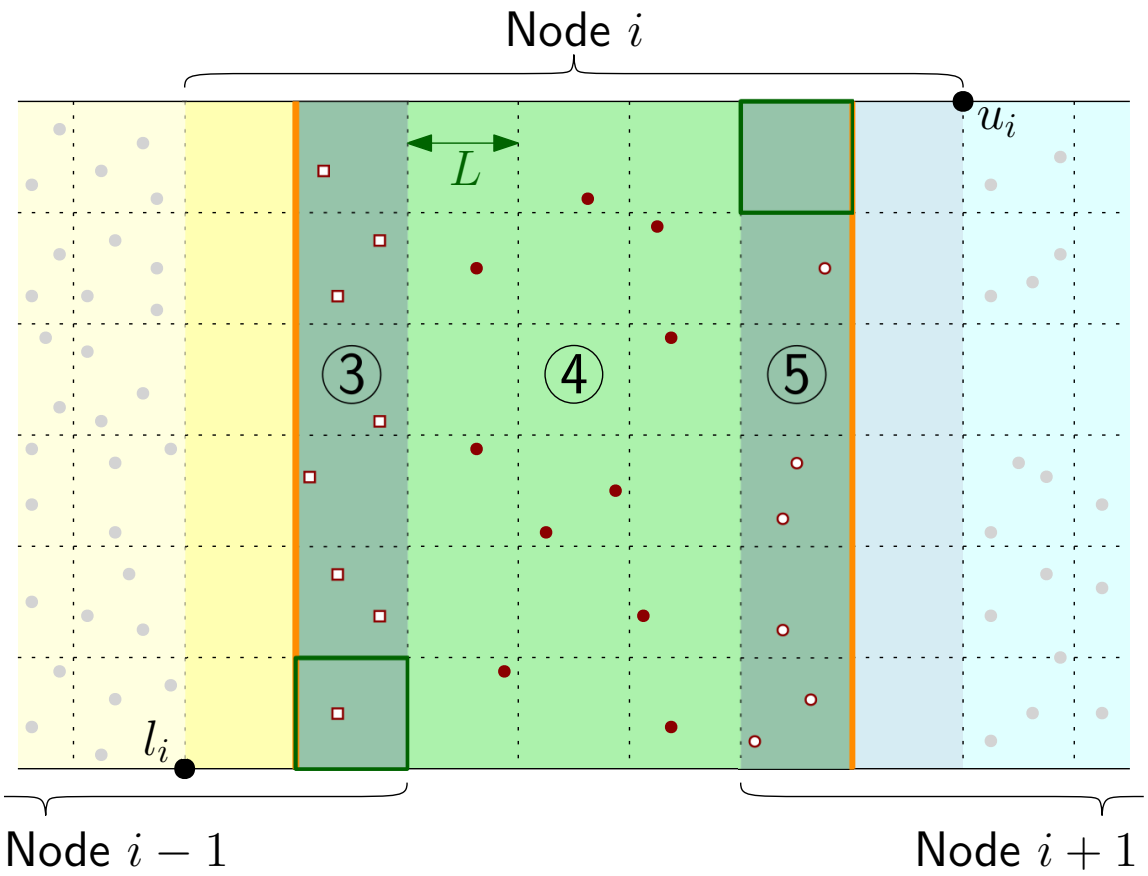


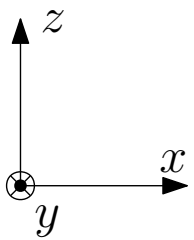


Particles in node i (•••••••••••)

4

Particles sorting

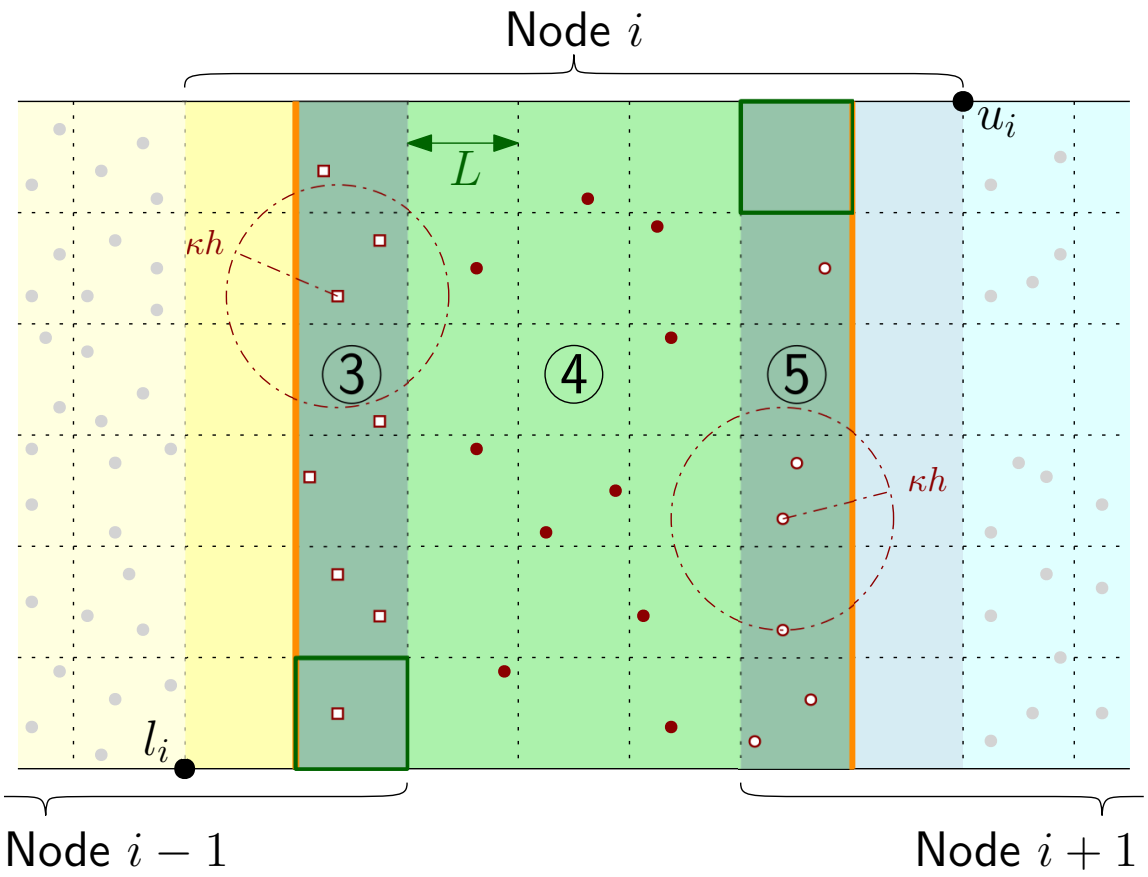


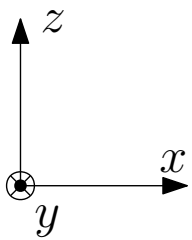


Particles in node i (▫▫▫▫▫▫▫●●●●●●●●●○○○○○○)

3 4 5

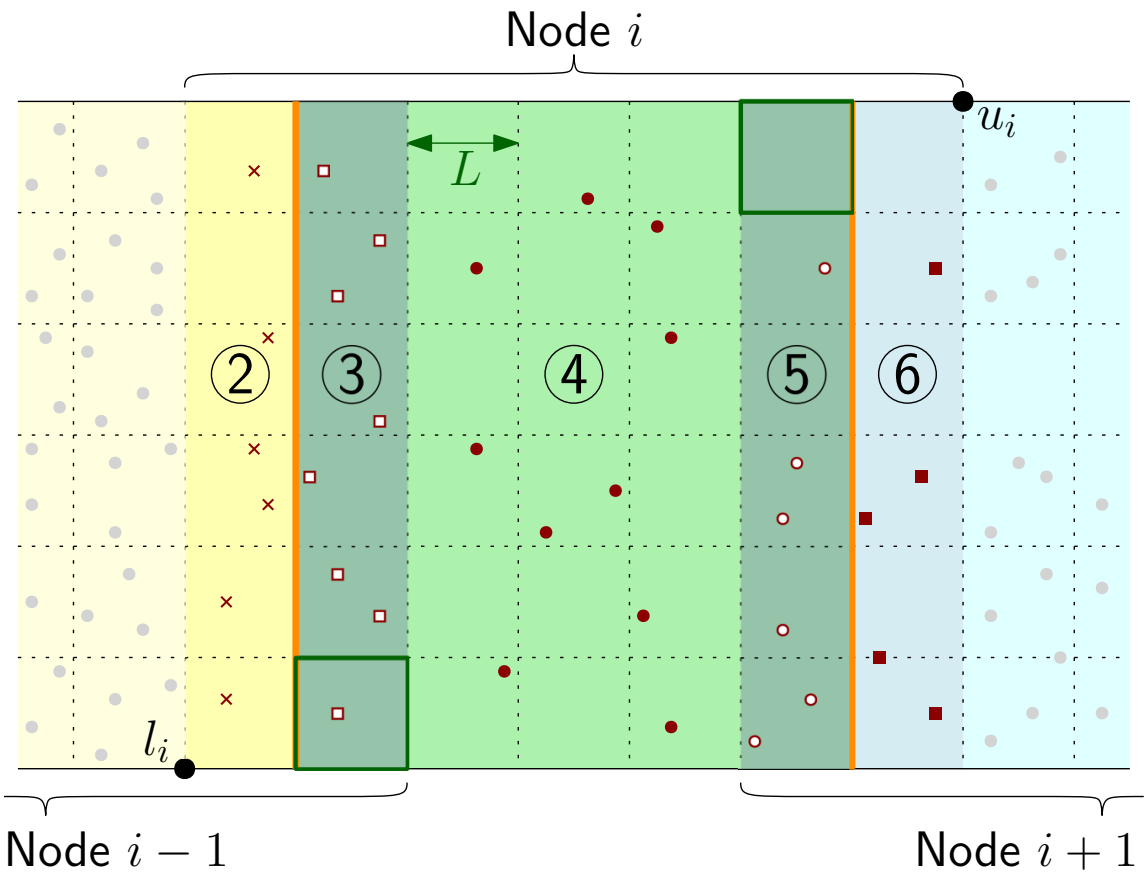
Particles sorting

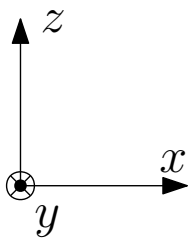


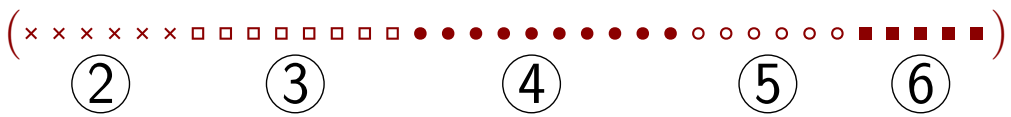


Particles in node i ( )

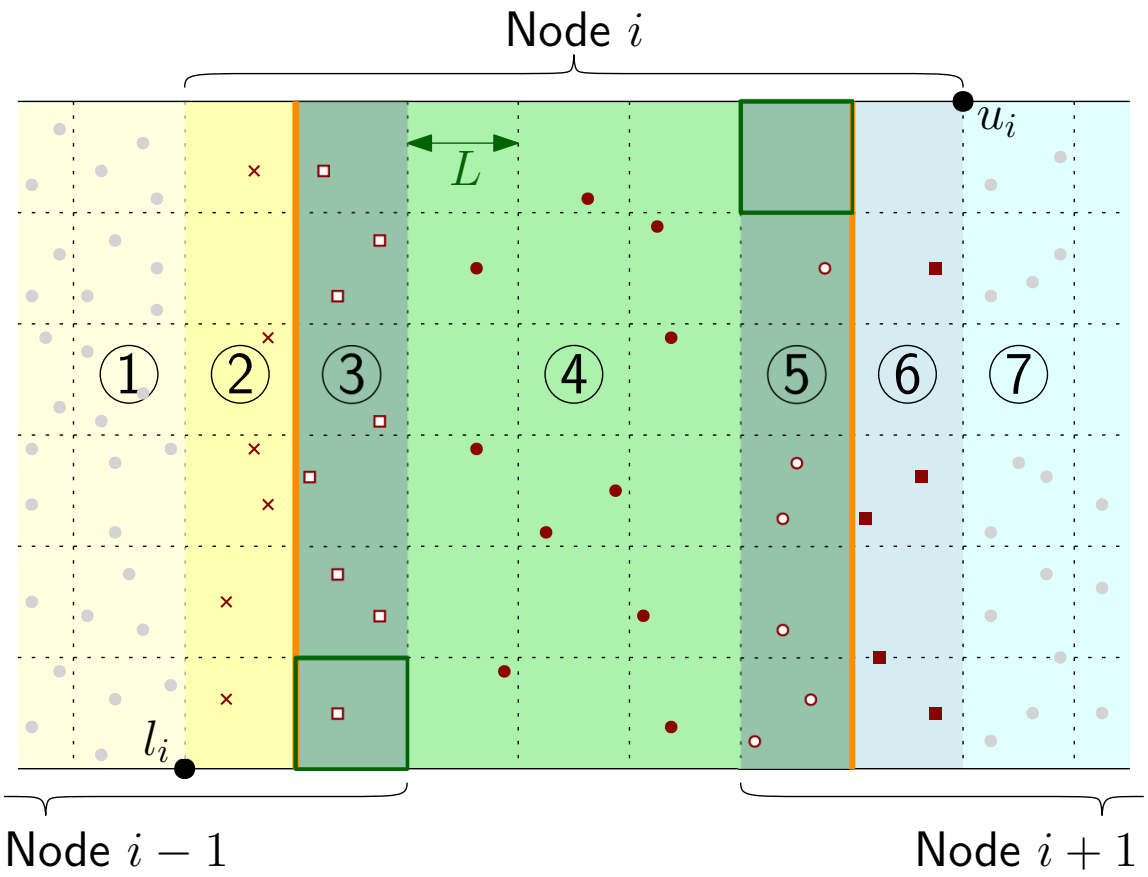
Particles sorting

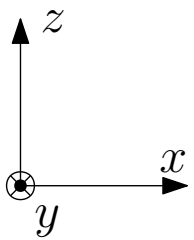


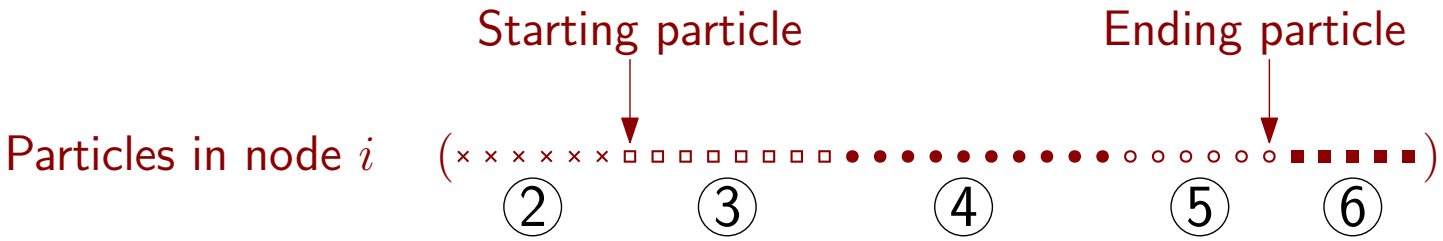


Particles in node i 

Particles sorting

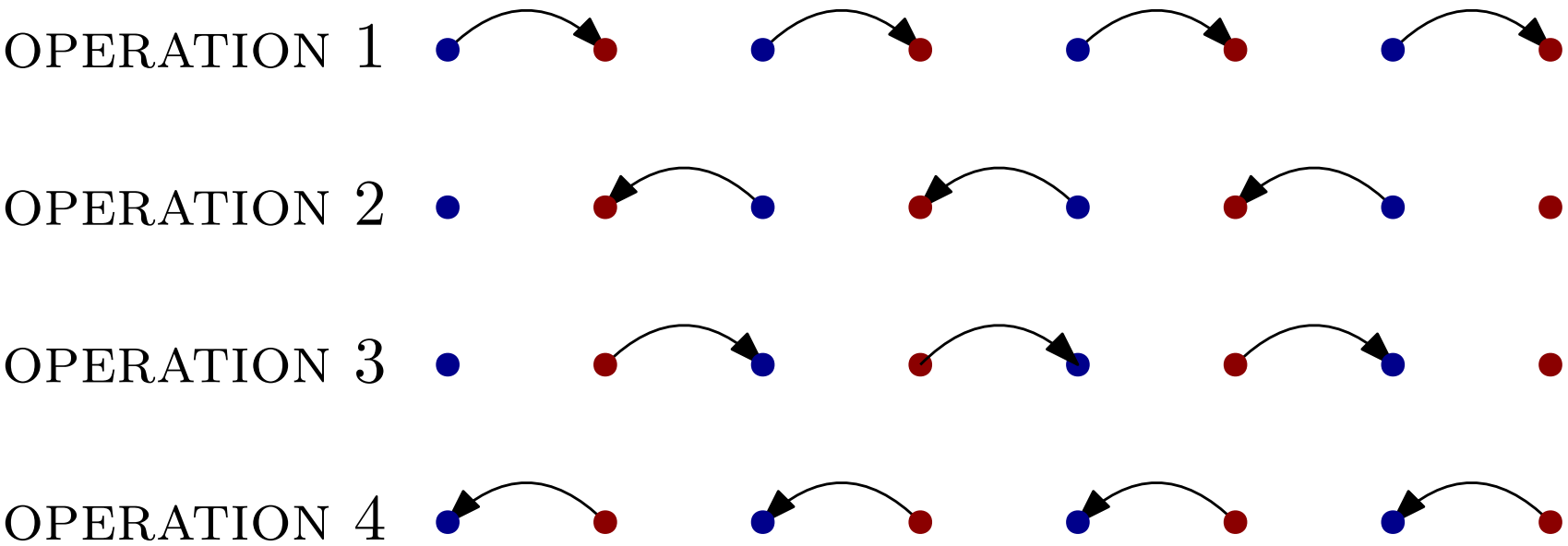






MPI communication scheme

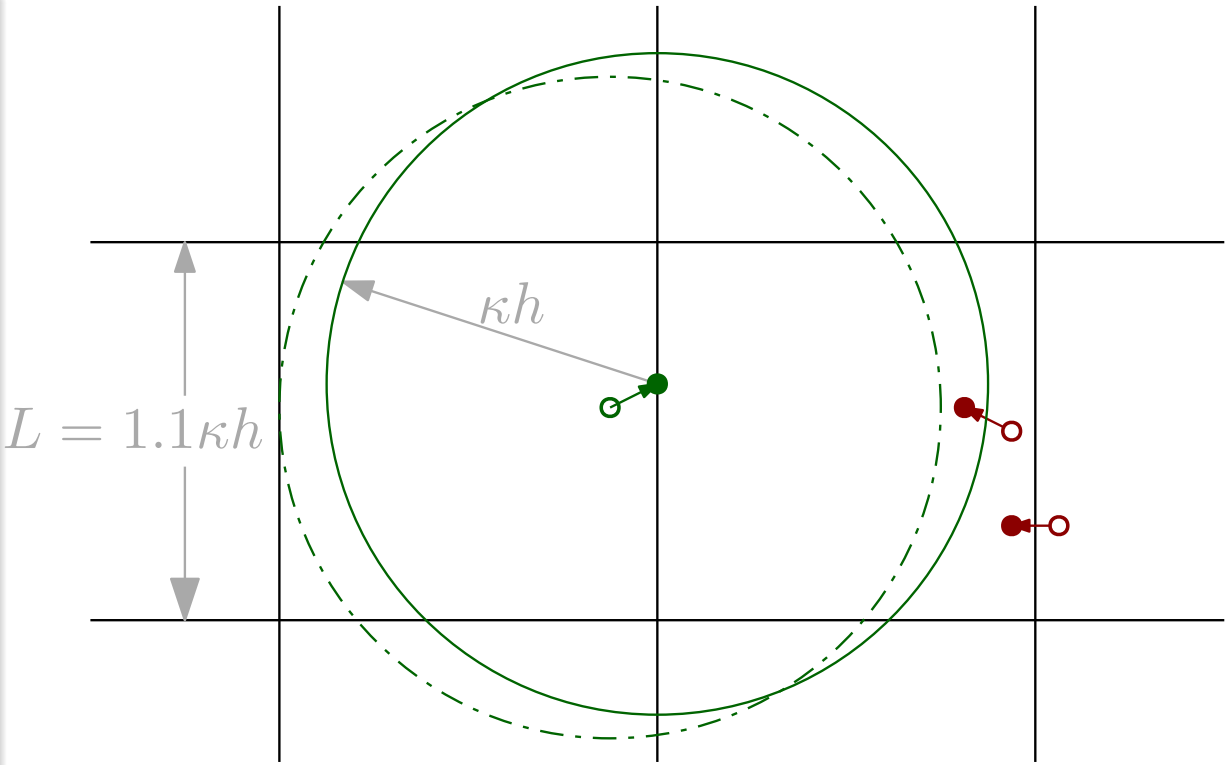
Edges and halos sharing on both sides for each node:



Box sizes

Objectives

- Limit the MPI exchanges at an intermediate time step
- Avoid incoming halo particles to influence the current ones
- Limit L for an efficient neighbour search

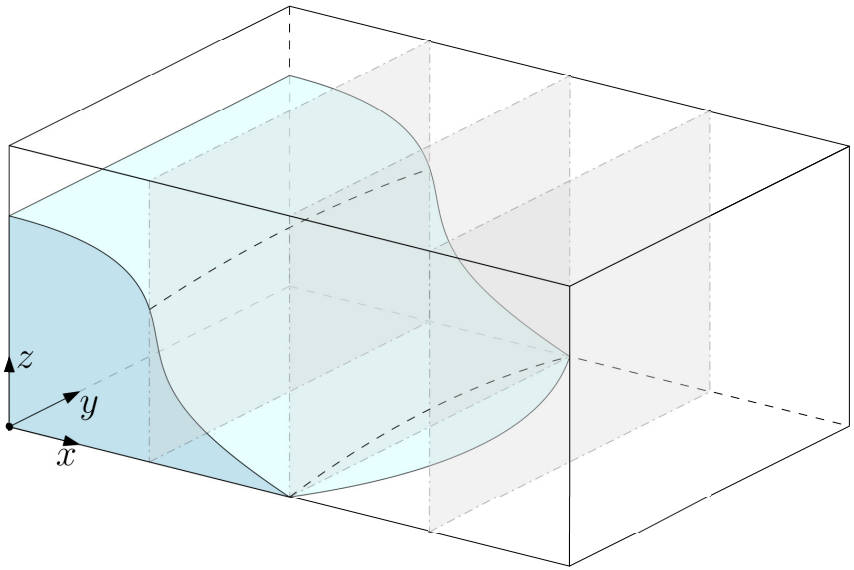


Compromise

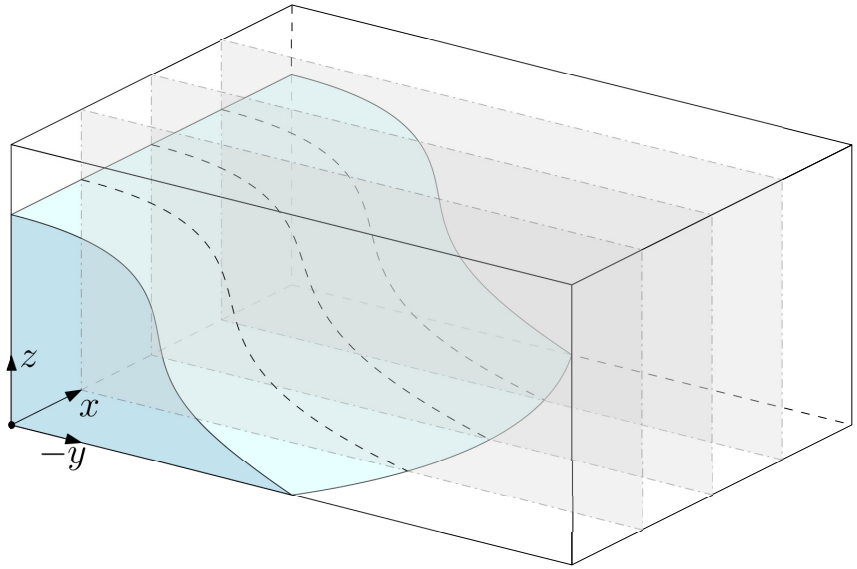
The box size is chosen to be $L = 1.1\kappa h$ with an appropriate time step

Domain partition importance

The load balancing is dependent on the partition direction:

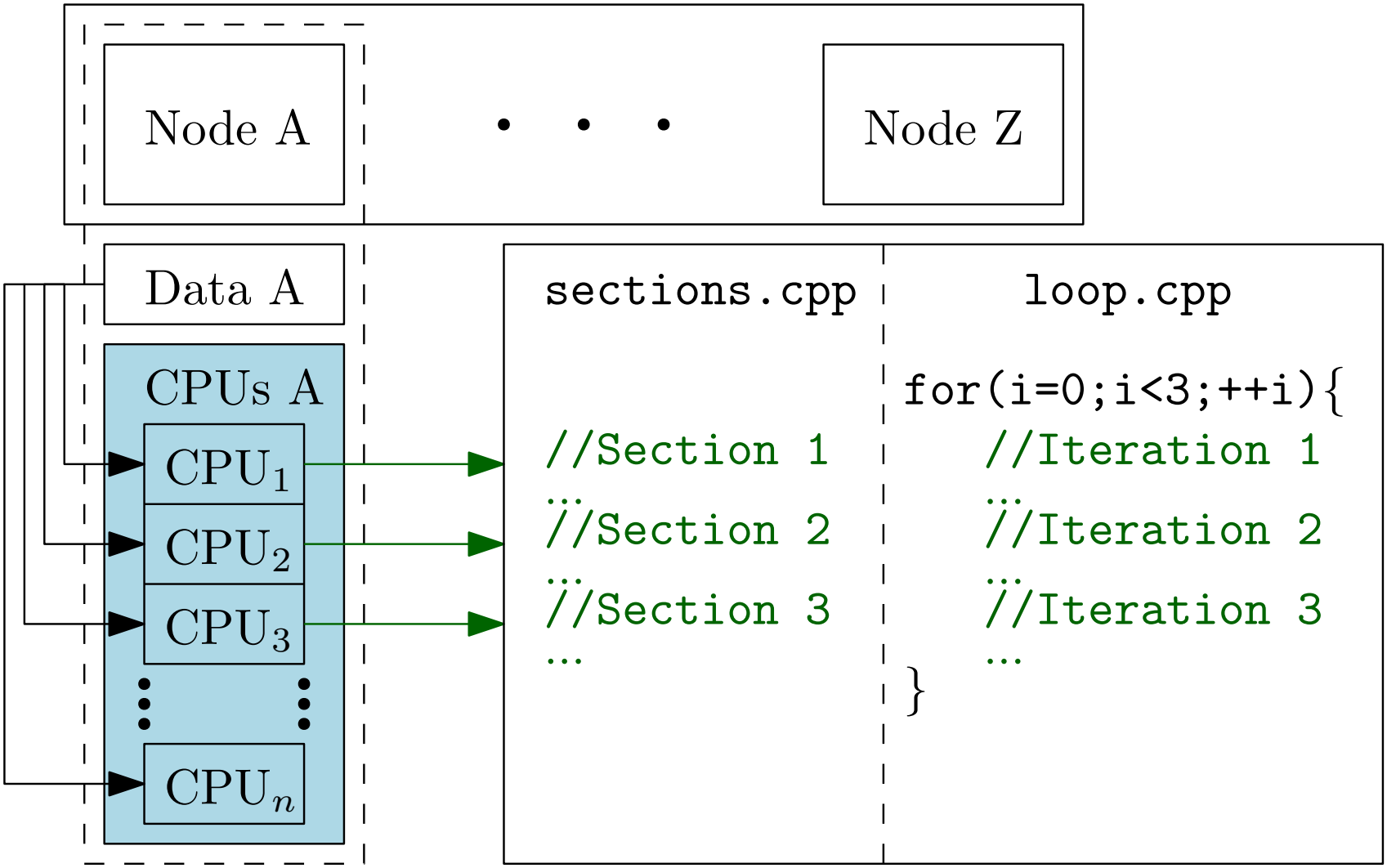


Poor load balancing



Efficient load balancing

Shared memory parallelization

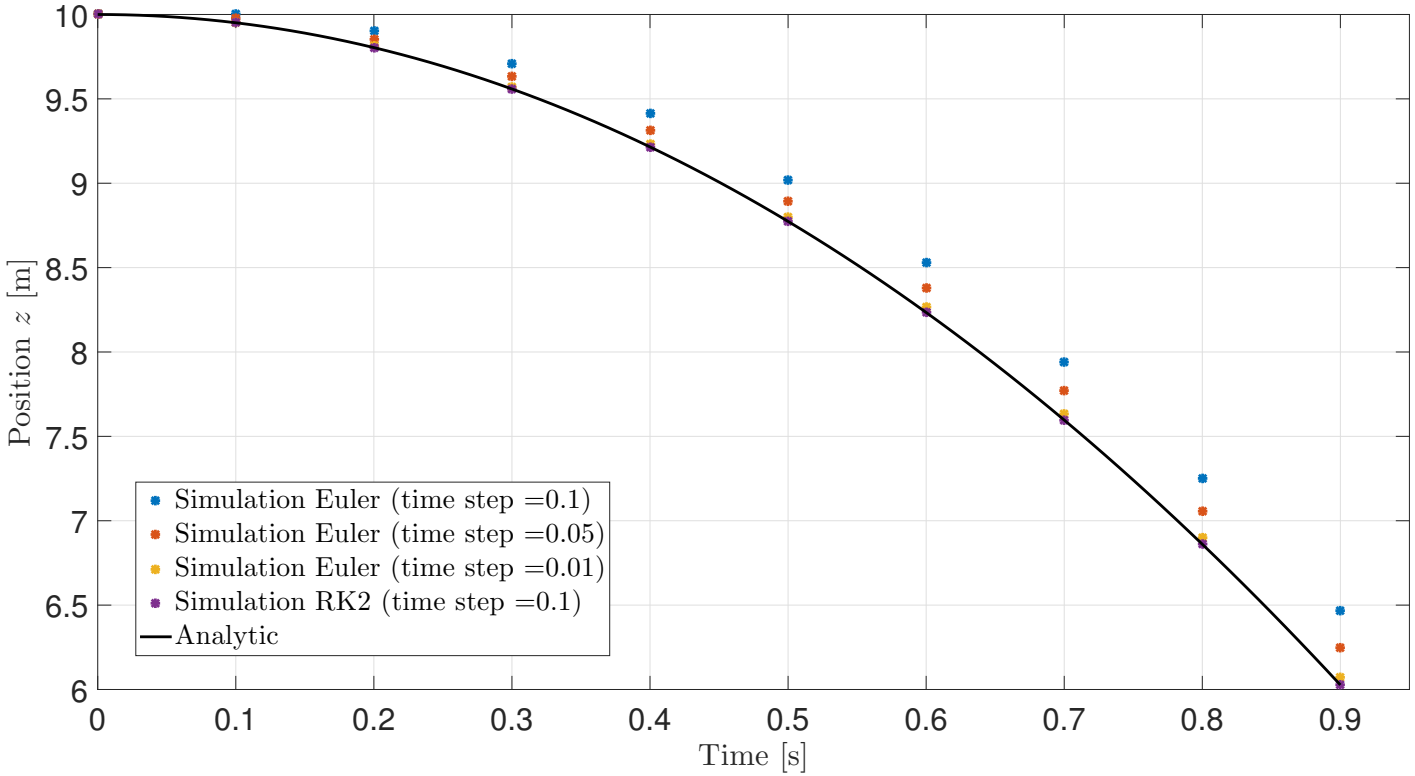


Contents

Free falling cube

Comparison

- Euler: velocity always smaller than the analytical solution
- RK2: the solution is exactly reached



Crashed cube

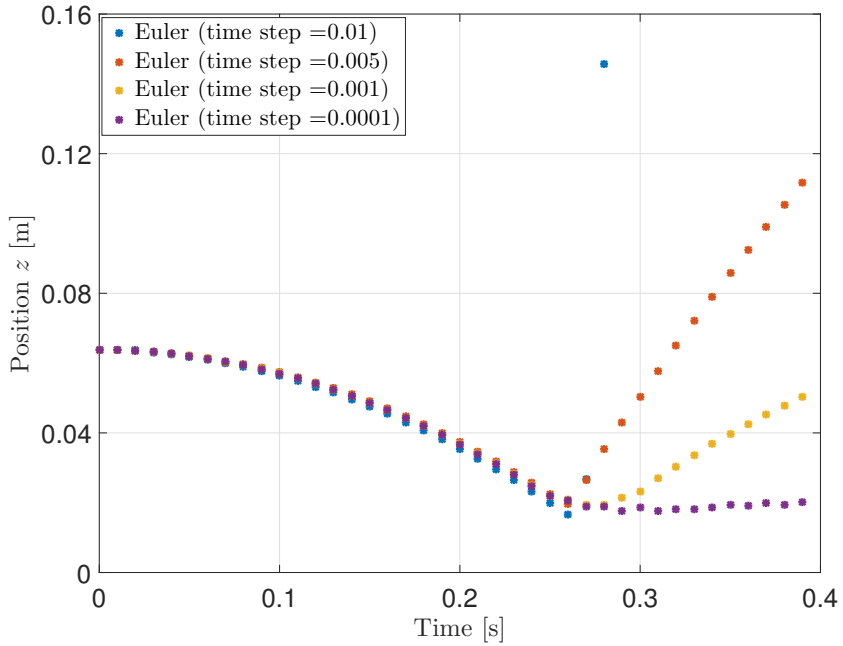
Comparison

Crash test corresponds to the worst case for numerical integration

- More stable solution with RK2
- Use a larger step time when using RK2

Euler

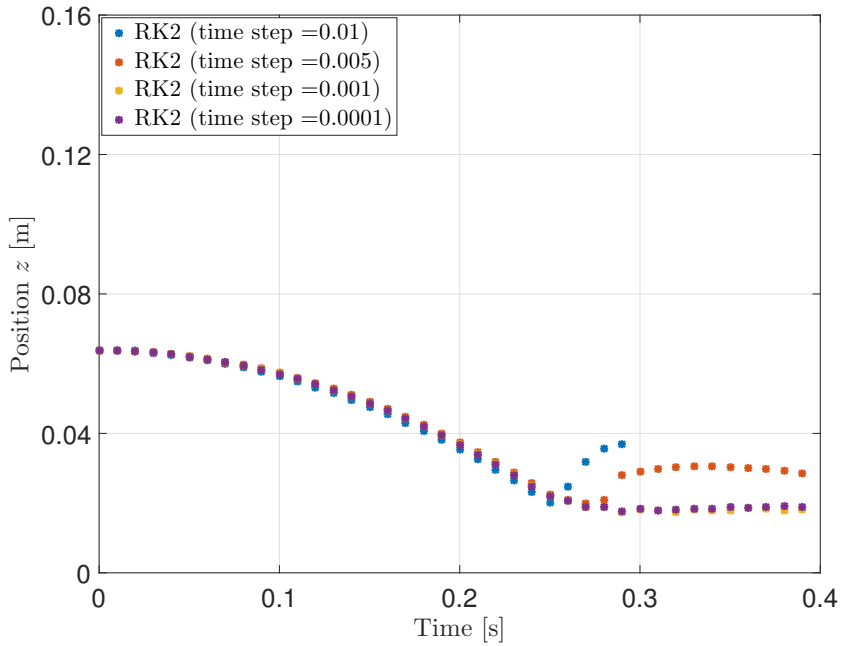
- Divergence and unstable
- Rebound on the surface



Crashed cube

Runge-Kutta order 2

- tends to the exact solution (more stable)
- less sensitive to the time step

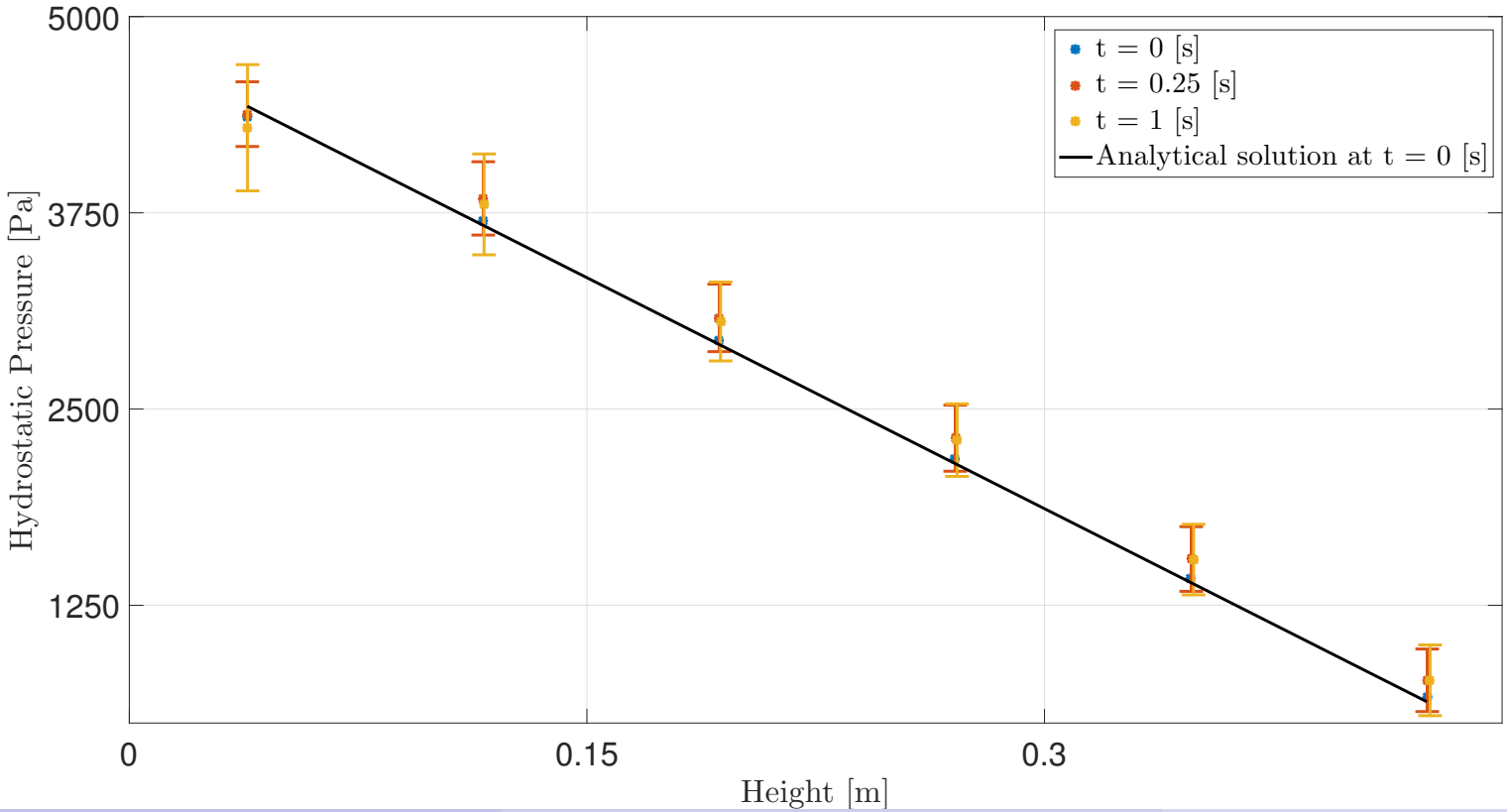


Still fluid

Hydrostatic presssure

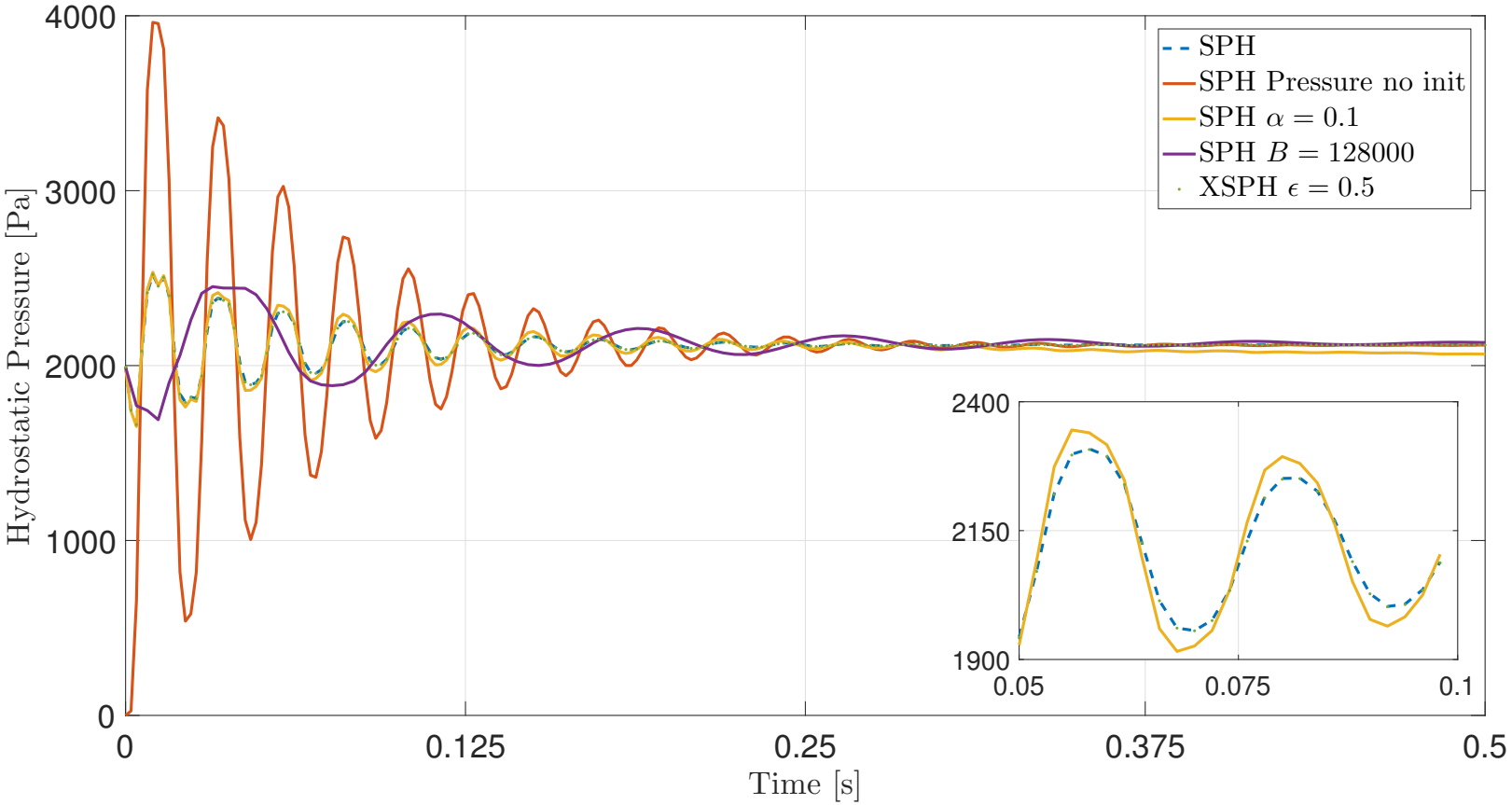
For a given depth, the hydrostatic pressure

- is close to the analytical solution
- oscillates until equilibrium



Still fluid

Hydrostatic presssure at middle depth

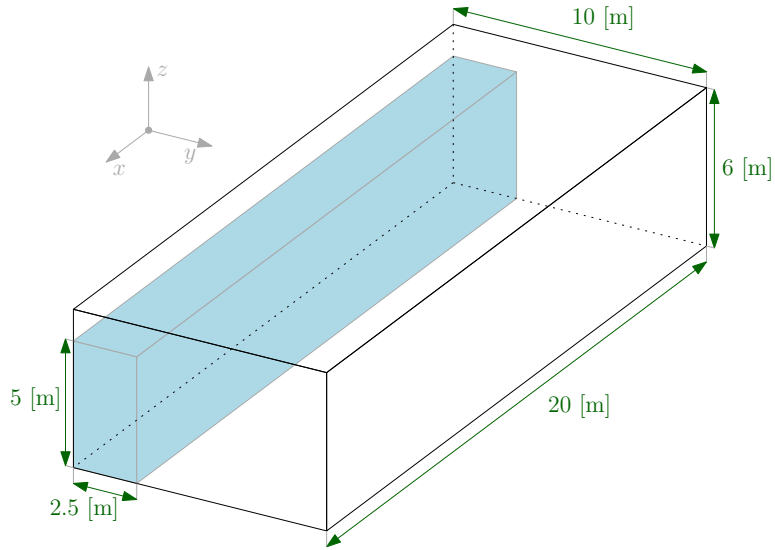


Dam break test and performance

Dam break

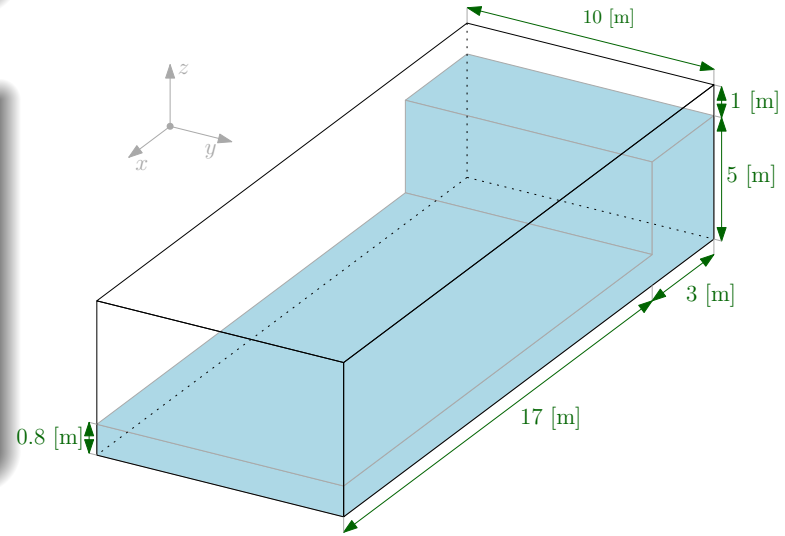
Well-suited geometry in order to check simultaneously:

- the physical behaviour
- the performances of the code



Ideal geometry

- Domain divided through the x-direction
- The first geometry is ideal unlike the second one



Simulation clip

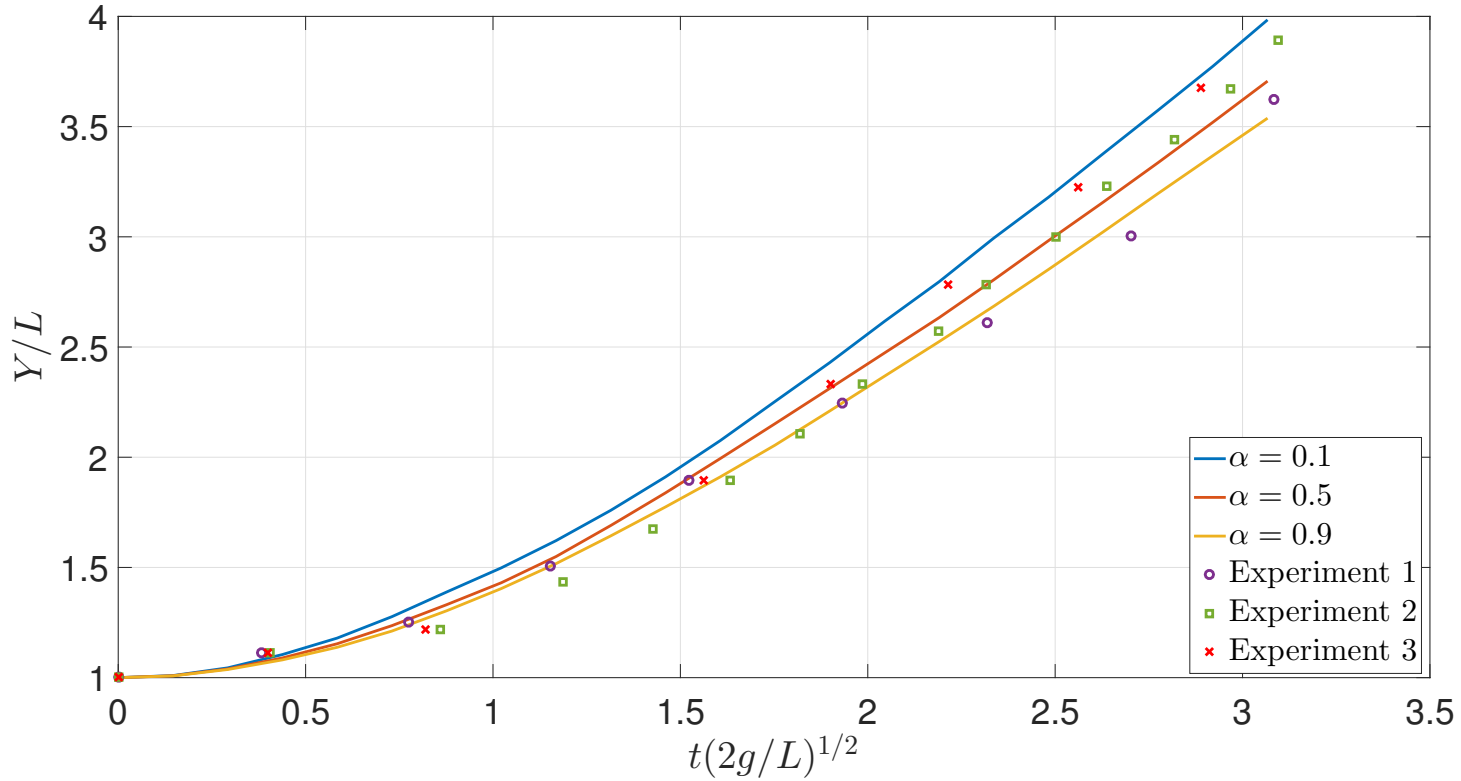
Simulation with 32 000 particles

Play/Pause

Dam break test and performance

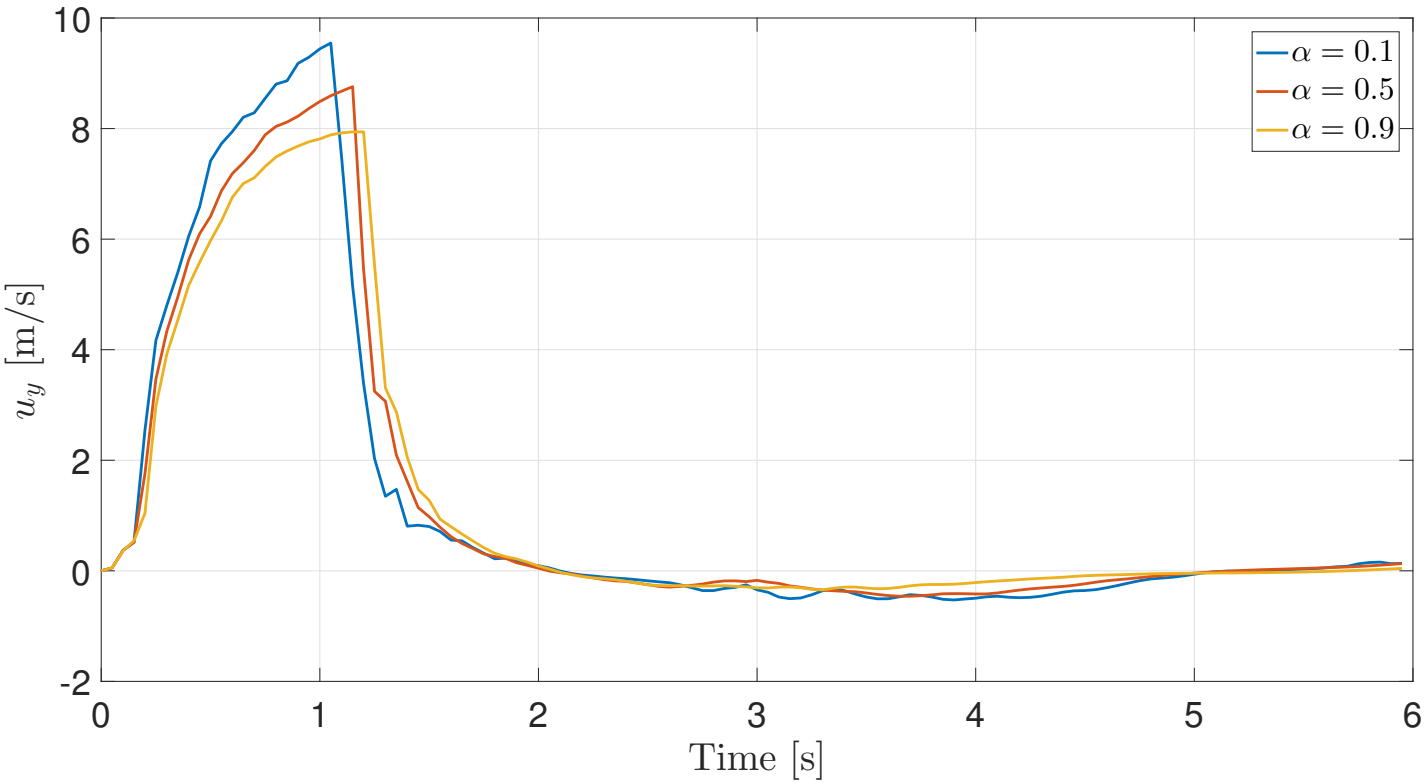
Dam break analysis

- Good agreement with experimental results
- Slight differences depending on the viscosity parameter α



Dam break test and performance

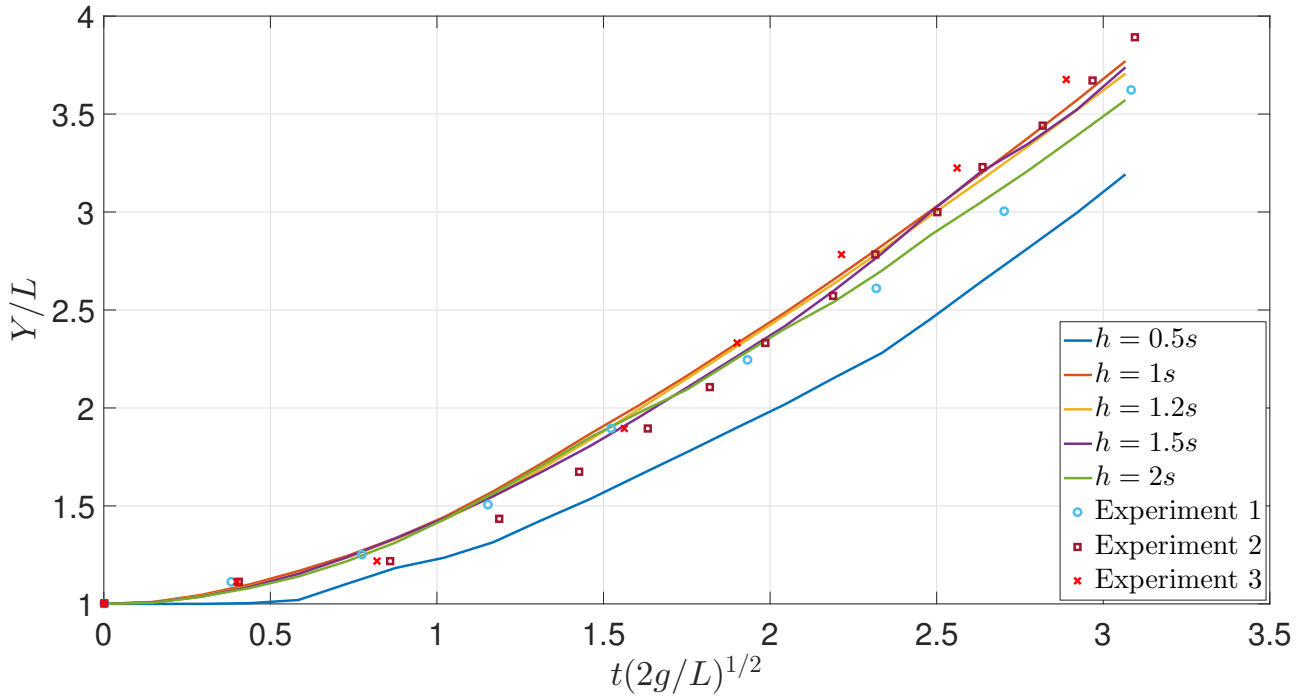
The speed of the wave front decreases when viscosity increases



Dam break test and performance

Dam break analysis

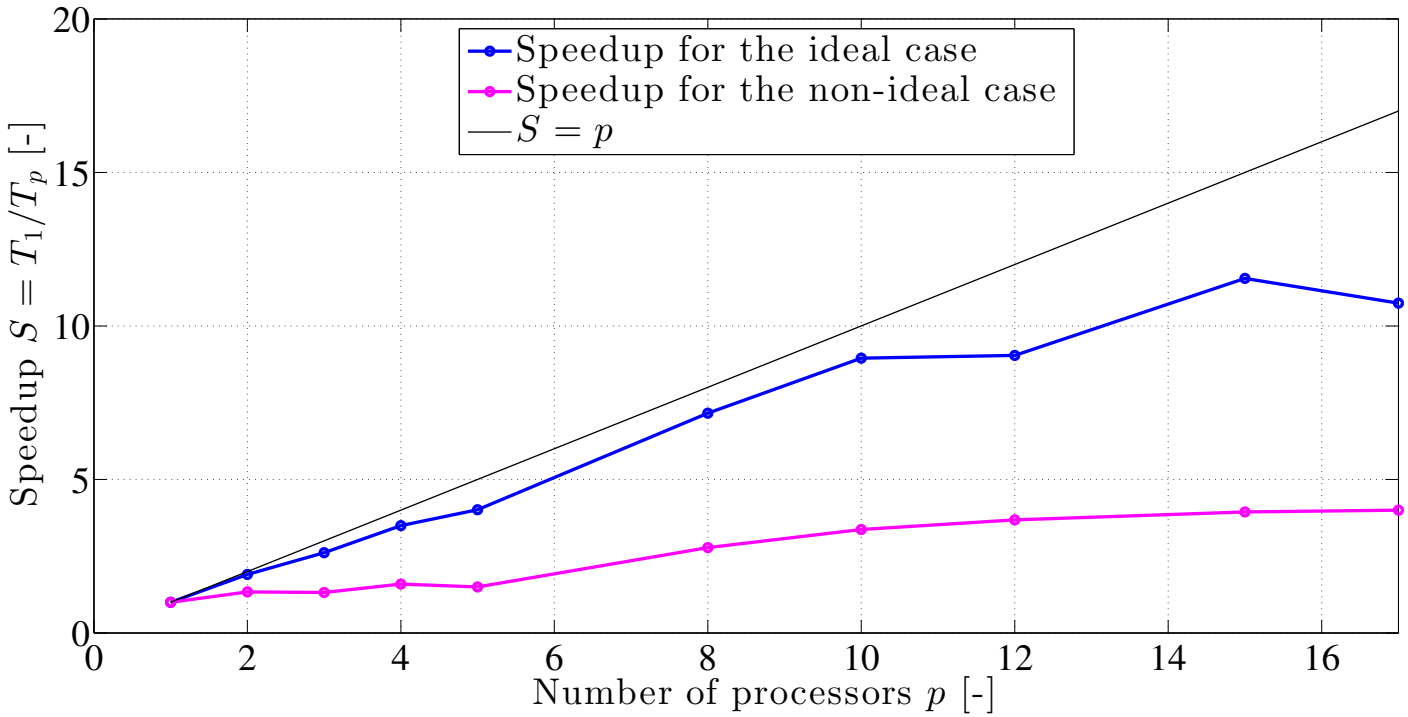
- Increase the smoothing length to take into account interactions with more particles
- Computational cost increases with h.
- $h = 1.2\ s \ \Rightarrow$ good compromise



Dam break test and performance

Strong scaling with MPI parallelization

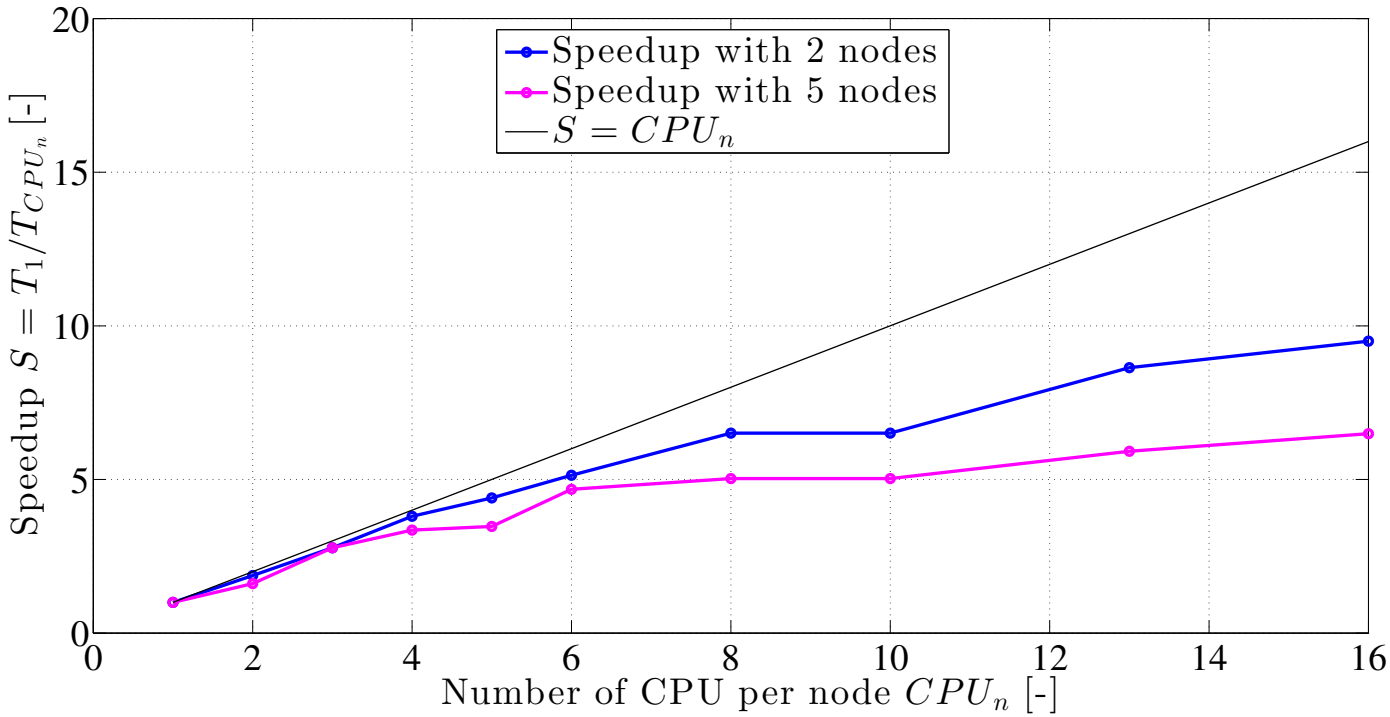
- Speed-up of the non-ideal case much lower than the ideal case
- Speed-up close to $S = p$ but little fluctuation when p increases
- Number of boxes not always a multiple of the number of process



Dam break test and performance

Strong scaling with OpenMP parallelization

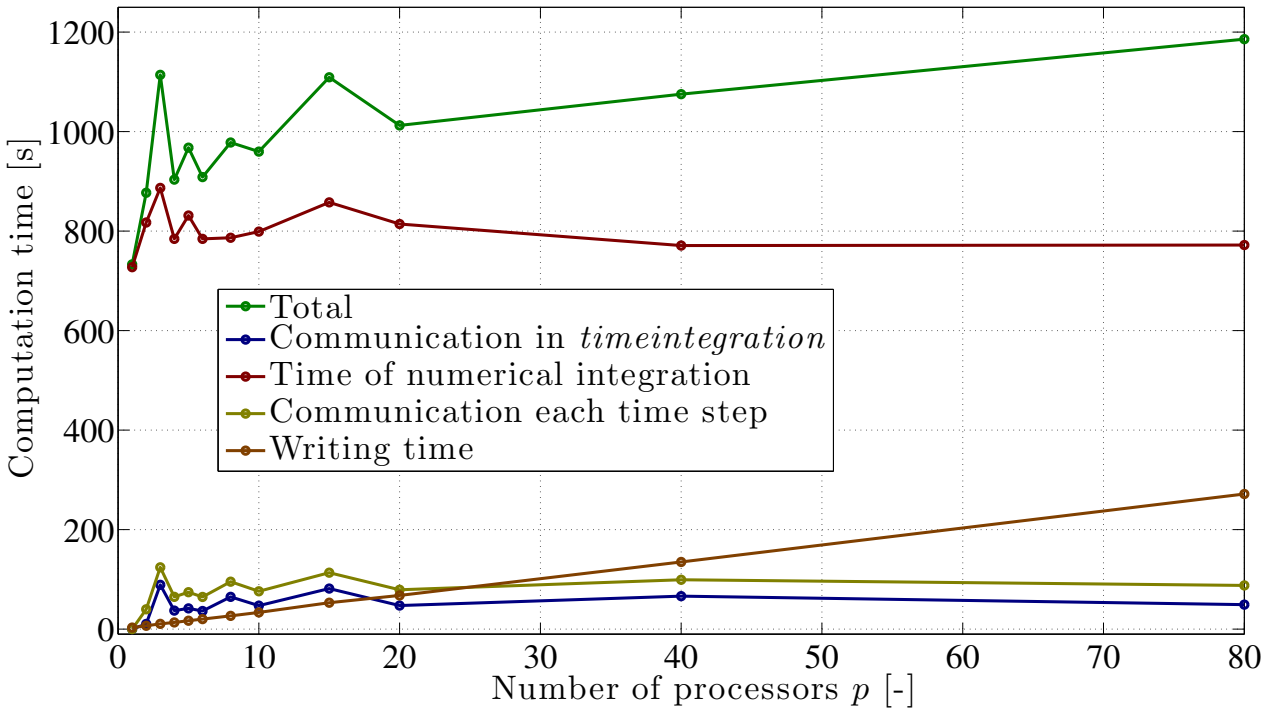
- Test on 56 700 particles
- Deviation from the ideal speed-up when the number of CPU per node increases



Dam break test and performance

Weak scaling

- 43 000 particles/CPU
- Complexity of the writing part is $\mathcal{O}(N)$
- Constant communication time between the nodes whatever the size of the domain



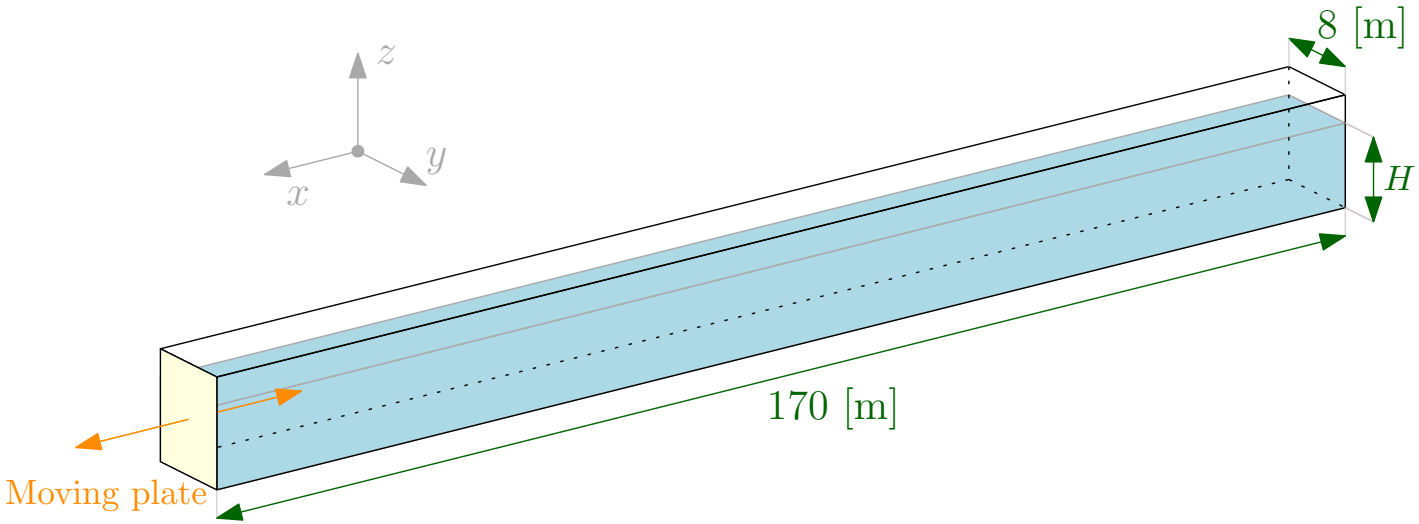
Wave propagation

Linear inviscid gravity waves

Theory for a constant depth profile predicts a phase propagation speed

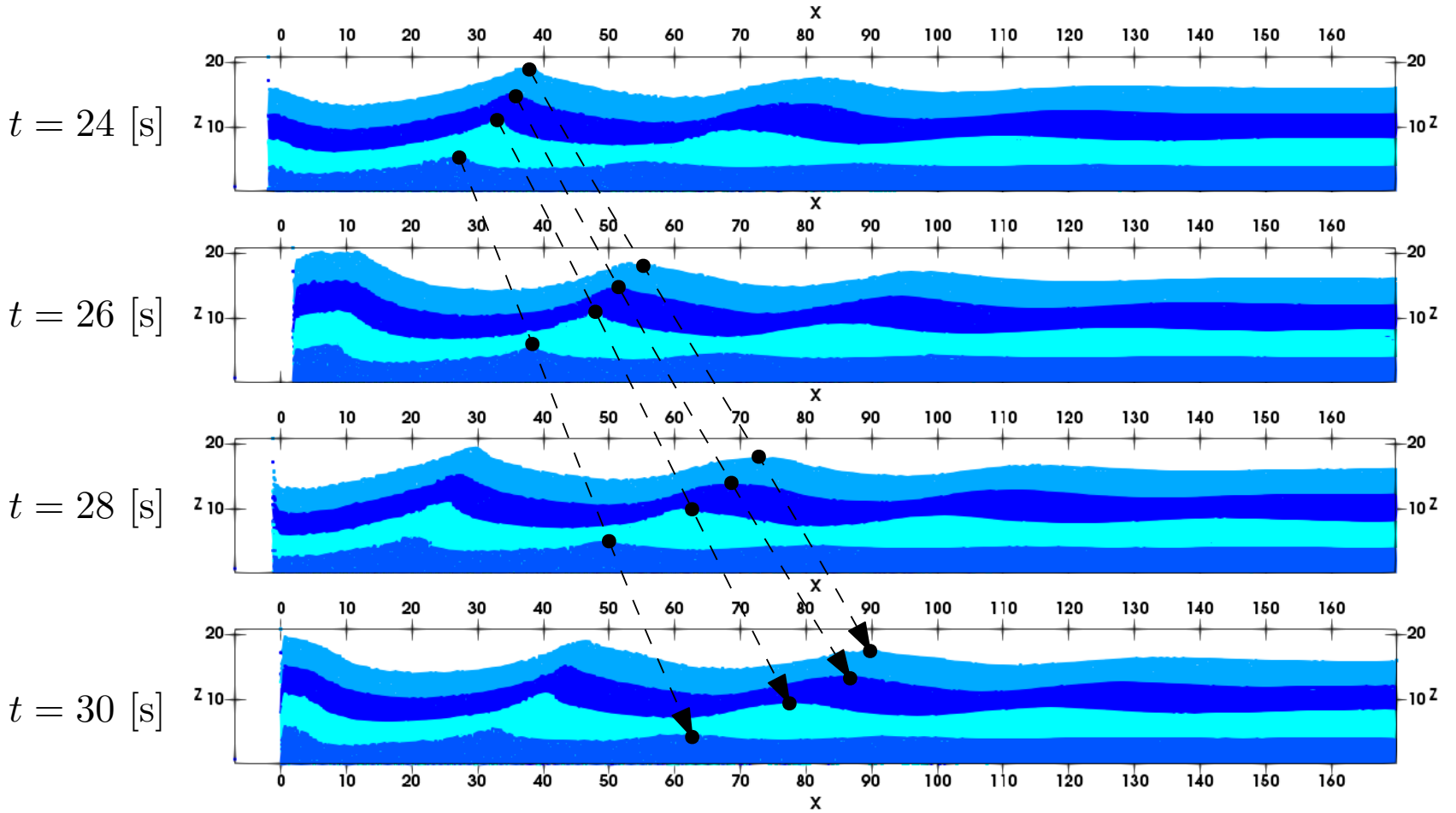
$$c = \sqrt{\frac{g\lambda}{2\pi} \tanh \frac{2\pi H}{\lambda}} \simeq \sqrt{gH} \quad \text{for} \quad \lambda \gg 2\pi H$$

Gravity waves model :



Moving plates exciting the fluid of variable height H in a periodic movement.

Wave propagation

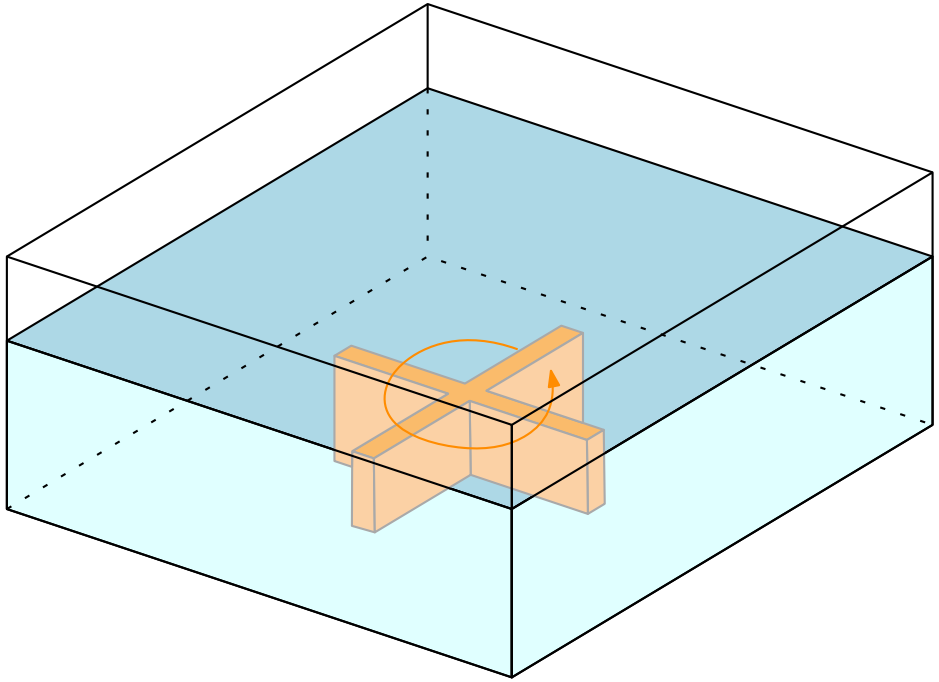


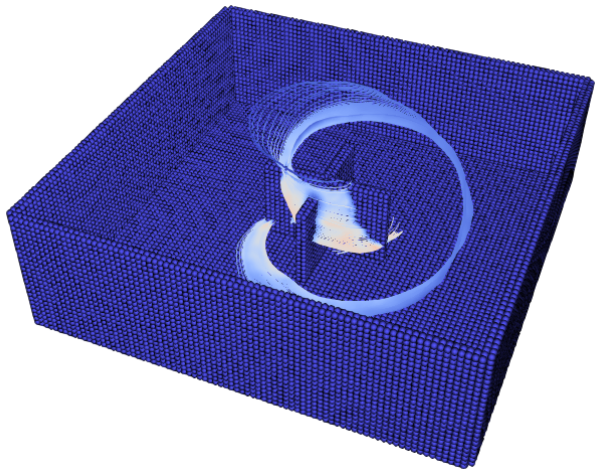
Simulation clips

Play/Pause | 1D wave | 1D wave - Superposition | 2D wave

Fluid mixing

Fluid tank mixed by a rotating cross





$\Rightarrow$ A complex 3D flow is generated

Play/Pause

Simple gravity model

Gravitational acceleration is modified:

- central force
- amplitude $\propto 1/r^2$ (bounded to avoid divergence)

$\Rightarrow$ Rough model for a large gravitational mass and lighter bodies

Play/Pause

Contents

Conclusion

The implemented code

- solves 3D fluid dynamics problems
- handles moving boundaries
- is parallel

Global performance (1000 time steps $\approx$ 1 sec. of simu. for $L \approx 10$ m)

- 40 seconds per 10^3 particles and per 1000 time steps on 1 CPU
- 9 minutes per 10^6 particles and per 1000 time steps on 80 CPU

Extensions for further work could consist in implementing

- a physical viscosity model
- fluid sources and sinks
- a dynamic domain sharing

Thank you for your attention!

Play/Pause

Acknowledgement

The authors would like to thank Mr. Louis GOFFIN without whom the following work could not have been accomplished, as it is based on his masters thesis *"Development of a didactic SPH model"*, published in 2013. The authors are also grateful to Pr. Christophe GEUZAINE and Dr. Romain BOMAN for the assistance and the help provided throughout the writing of both the code and the report, as well as for the many hours spent on this project.