Load Balancing Strategies for Parallel SPH

Pushkar Godbole*
Kunal Puri
P. Ramachandran

Department of Aerospace Engineering,
IIT Bombay

June 06, 2013

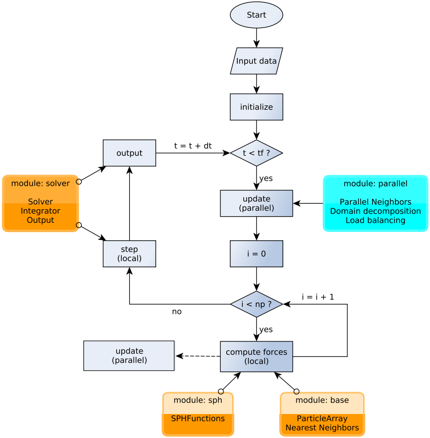
Outline

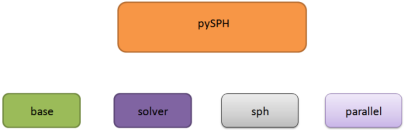
1 Smooth Particle Hydrodynamics
- The serial algorithm
- The parallel algorithm
- Load Balancing

2 Load-balancing
- Load Balancing techniques
- Algorithms
- How do they do it?

3 Results
- Illustration
- Results
- Conclusion and Further work

PySPH

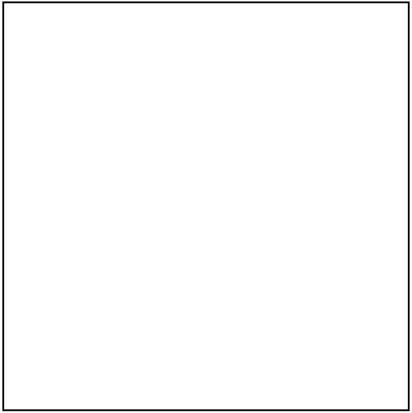

Start
Input data
initialize
t < tf ?
t = t + dt
output
yes
update
(parallel)
module: parallel
Parallel Neighbors
Domain decomposition
Load balancing
module: solver
Solver
Integrator
Output
step
(local)
i = 0
i < np ?
i = i + 1
no
yes
update
(parallel)
compute forces
(local)
module: sph
SPHFunctions
module: base
ParticleArray
Nearest Neighbors


pySPH
base
solver
sph
parallel

Serial algorithm in a nutshell

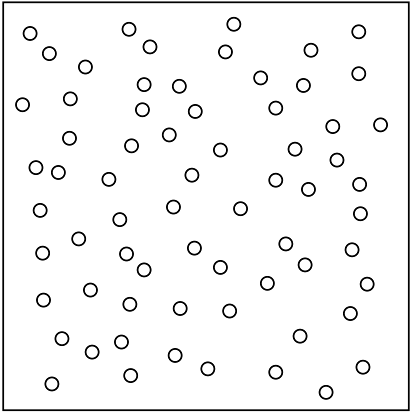
Serial algorithm in a nutshell

Given a domain...



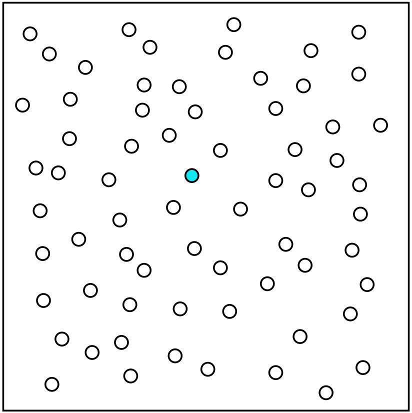
Serial algorithm in a nutshell

Discretized with *Particles*



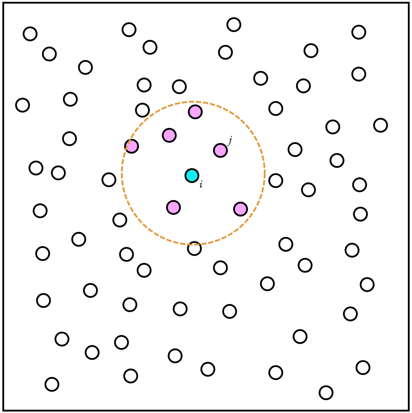
Serial algorithm in a nutshell

For every particle...



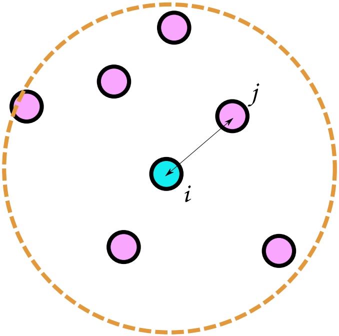
Serial algorithm in a nutshell

Find nearest neighbors...



Serial algorithm in a nutshell

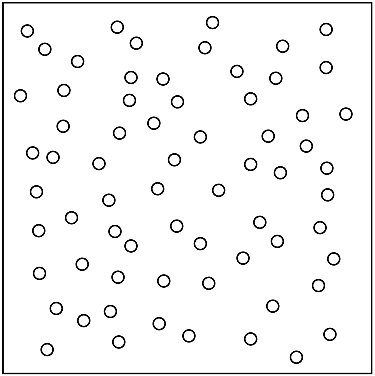
Compute interactions..



$$\frac{dU_i}{dt} = -\sum_{j \in \mathcal{N}(i)} m_j \mathcal{F}_{ij} \nabla_i W_{ij}$$

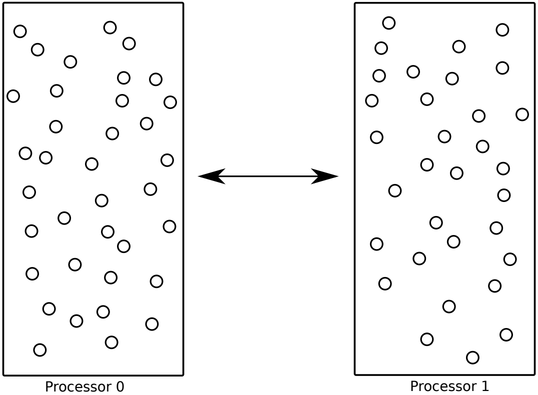
Doing it in Parallel

Given the domain discretized with *Particles*..



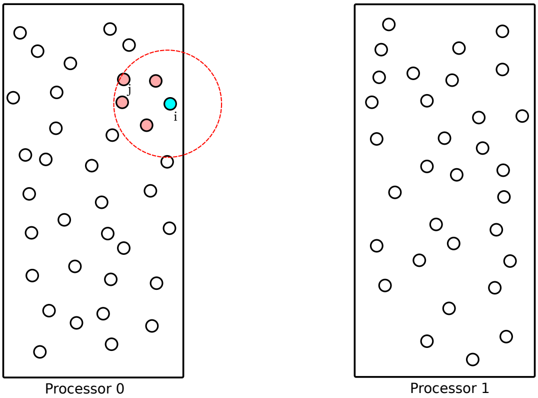
Doing it in Parallel

Partition it across processors : *Load-balancing*



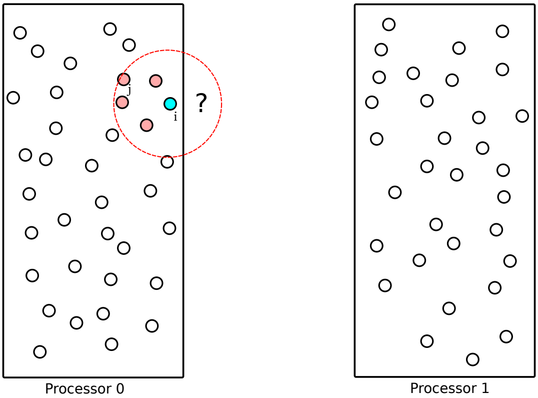
Doing it in Parallel

For every particle, find neighbors.



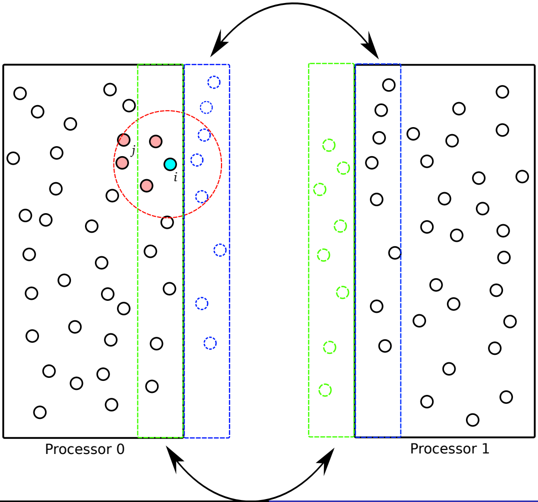
Doing it in Parallel

For every particle, find neighbors. *Oops!*



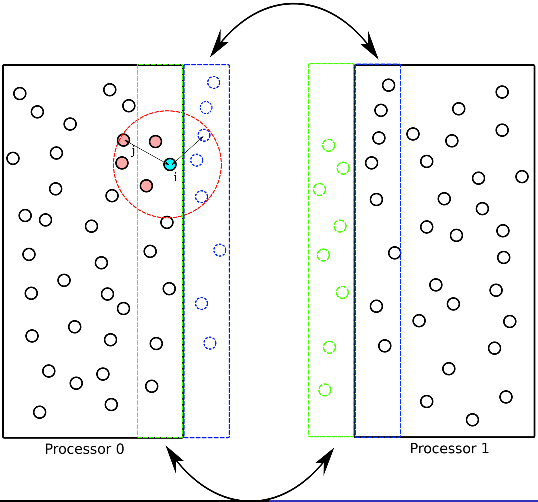
Doing it in Parallel

Exchange ghost data.



Doing it in Parallel

Exchange ghost data. And compute interactions..



Requirements : Parallel SPH

Features

Requirements

Requirements : Parallel SPH

Features

- *Local* particles assigned to processors
-
-

Requirements

- Equal distribution of workload
-
-

Requirements : Parallel SPH

Features

- *Local* particles assigned to processors
- *Ghost (Remote)* particles shared as halo region
-

Requirements

- Equal distribution of workload
- Minimum communication overhead
-

Requirements : Parallel SPH

Features

- *Local* particles assigned to processors
- *Ghost (Remote)* particles shared as halo region
- Due to the Lagrangian nature of SPH, particle distribution changes!

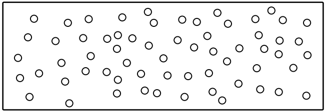
Requirements

- Equal distribution of workload
- Minimum communication overhead
- Dynamic Load Balancing

The right way

Load Balancing

Mapping of *objects* to processors and distributing data accordingly.

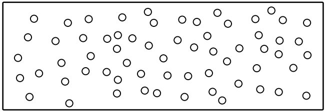


Given a domain

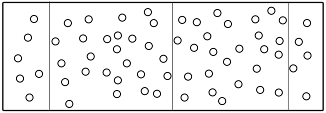
The right way

Load Balancing

Mapping of *objects* to processors and distributing data accordingly.



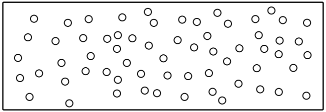
Given a domain



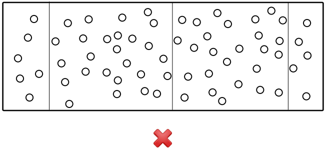
The right way

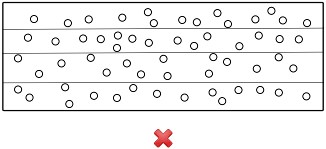
Load Balancing

Mapping of *objects* to processors and distributing data accordingly.



Given a domain

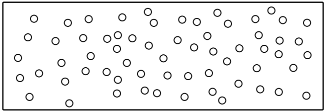




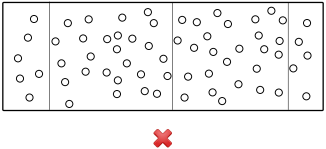
The right way

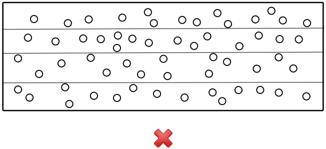
Load Balancing

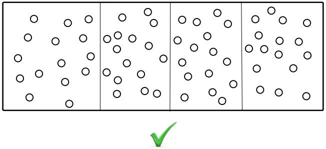
Mapping of *objects* to processors and distributing data accordingly.



Given a domain







Outline

1 Smooth Particle Hydrodynamics
- The serial algorithm
- The parallel algorithm
- Load Balancing

2 Load-balancing
- Load Balancing techniques
- Algorithms
- How do they do it?

3 Results
- Illustration
- Results
- Conclusion and Further work

Algorithm classification

Algorithm classification

Geometric partitioners

- Physical coordinates as input
- Most general for numerical work
- Natural for particle methods

Examples

- Recursive Coordinate Bisection (RCB)
- Recursive Inertial Bisection (RIB)
- Space Filling Curves (SFC)

Algorithm classification

Geometric partitioners

- Physical coordinates as input
- Most general for numerical work
- Natural for particle methods

Examples

- Recursive Coordinate Bisection (RCB)
- Recursive Inertial Bisection (RIB)
- Space Filling Curves (SFC)

Graph partitioners

- Data represented as a graph
- Inherently suitable for mesh-based methods
- Cell based graph-partitioning may be used for SPH

Examples

- METIS/ParMETIS
- PTScotch
- Hypergraph partitioning

Zoltan Data Management Library

What is it?

- Developed by Sandia National Laboratories
- Trilinos Project (9.0 September 2008)
- Zoltan v3.6 released in September 2011

What can it do?

- Dynamic Load Balancing
- Graph Coloring
- Dynamic memory management

Zoltan Data Management Library

What is it?

- Developed by Sandia National Laboratories
- Trilinos Project (9.0 September 2008)
- Zoltan v3.6 released in September 2011

Algorithms

- Geometric (RCB, RIB, HSFC)
- Graph and Hypergraph
- ParMETIS, PTScotch

What can it do?

- Dynamic Load Balancing
- Graph Coloring
- Dynamic memory management

Zoltan Data Management Library

What is it?

- Developed by Sandia National Laboratories
- Trilinos Project (9.0 September 2008)
- Zoltan v3.6 released in September 2011

What can it do?

- Dynamic Load Balancing
- Graph Coloring
- Dynamic memory management

Algorithms

- Geometric (RCB, RIB, HSFC)
- Graph and Hypergraph
- ParMETIS, PTScotch

Motivation

- PySPH parallel module
- PyZoltan
- Load balancing for particle methods

Recursive Coordinate Bisection (RCB)

Algorithm and Advantages

- Recursively subdivide domain
- Cuts are orthogonal to co-ordinate axes
- Fast and inexpensive
- Global decomposition is trivial

Disadvantages

- Not adaptive to rotations
- Leads to stretched halo-regions

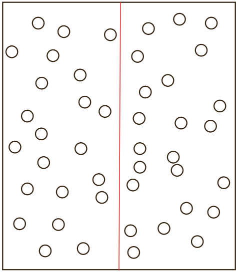
Recursive Coordinate Bisection (RCB)

Algorithm and Advantages

- Recursively subdivide domain
- Cuts are orthogonal to co-ordinate axes
- Fast and inexpensive
- Global decomposition is trivial

Disadvantages

- Not adaptive to rotations
- Leads to stretched halo-regions



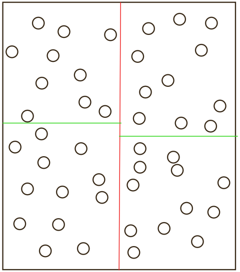
Recursive Coordinate Bisection (RCB)

Algorithm and Advantages

- Recursively subdivide domain
- Cuts are orthogonal to co-ordinate axes
- Fast and inexpensive
- Global decomposition is trivial

Disadvantages

- Not adaptive to rotations
- Leads to stretched halo-regions



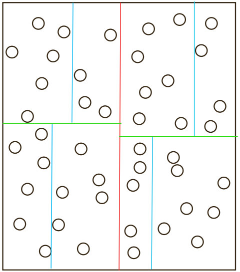
Recursive Coordinate Bisection (RCB)

Algorithm and Advantages

- Recursively subdivide domain
- Cuts are orthogonal to co-ordinate axes
- Fast and inexpensive
- Global decomposition is trivial

Disadvantages

- Not adaptive to rotations
- Leads to stretched halo-regions



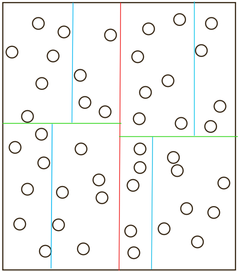
Recursive Coordinate Bisection (RCB)

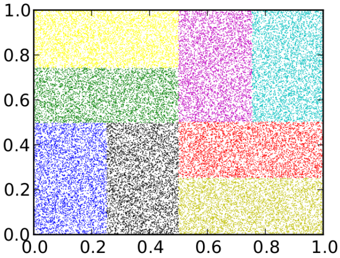
Algorithm and Advantages

- Recursively subdivide domain
- Cuts are orthogonal to co-ordinate axes
- Fast and inexpensive
- Global decomposition is trivial

Disadvantages

- Not adaptive to rotations
- Leads to stretched halo-regions





Recursive Inertial Bisection (RIB)

Algorithm and Advantages

- Variant of RCB
- Finds inertial axes
- Cuts are orthogonal to principal inertial axes
- Adaptive to rotations
- Lesser communication overhead

Disadvantages

- Eigenvector computations

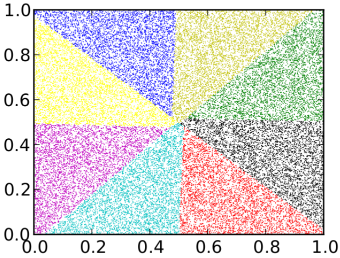
Recursive Inertial Bisection (RIB)

Algorithm and Advantages

- Variant of RCB
- Finds inertial axes
- Cuts are orthogonal to principal inertial axes
- Adaptive to rotations
- Lesser communication overhead

Disadvantages

- Eigenvector computations



Hilbert Space Filling Curves (HSFC)

Algorithm and Advantages

- Use a SFC $f : R^3 \rightarrow R$
- Order objects linearly
- Geometric locality

Disadvantages

- Particle distribution has *projections*
- Disconnected regions for complex geometries

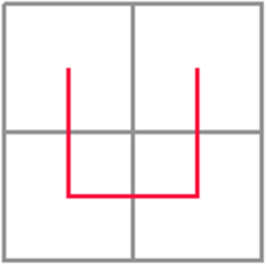
Hilbert Space Filling Curves (HSFC)

Algorithm and Advantages

- Use a SFC $f : R^3 \rightarrow R$
- Order objects linearly
- Geometric locality

Disadvantages

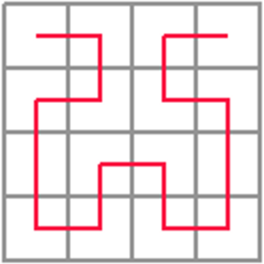
- Particle distribution has *projections*
- Disconnected regions for complex geometries



Hilbert Space Filling Curves (HSFC)

Algorithm and Advantages

- Use a SFC $f : R^3 \rightarrow R$
- Order objects linearly
- Geometric locality



Disadvantages

- Particle distribution has *projections*
- Disconnected regions for complex geometries

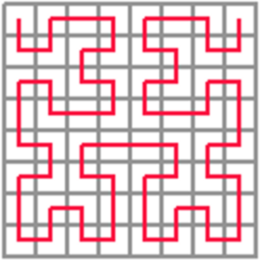
Hilbert Space Filling Curves (HSFC)

Algorithm and Advantages

- Use a SFC $f : R^3 \to R$
- Order objects linearly
- Geometric locality

Disadvantages

- Particle distribution has *projections*
- Disconnected regions for complex geometries



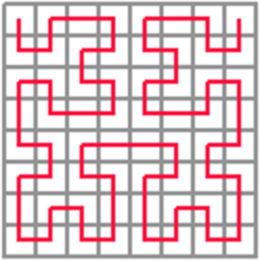
Hilbert Space Filling Curves (HSFC)

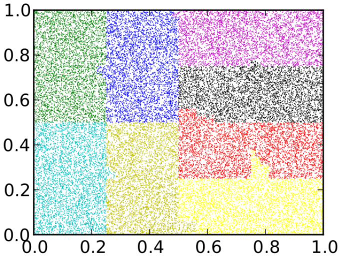
Algorithm and Advantages

- Use a SFC $f : R^3 \rightarrow R$
- Order objects linearly
- Geometric locality

Disadvantages

- Particle distribution has *projections*
- Disconnected regions for complex geometries





Graph partitioning

Algorithm and Advantages

- Interpret particle/cell data as graph nodes and neighborhood as edges
- Cell based graph partitioning has fixed/known neighbors
- Generates closed and compact partitions

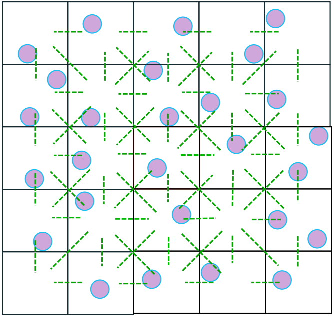
Disadvantages

- Inherently mesh-based
- Can not be used on particle data directly
- Slower than geometric methods

Graph partitioning

Algorithm and Advantages

- Interpret particle/cell data as graph nodes and neighborhood as edges
- Cell based graph partitioning has fixed/known neighbors
- Generates closed and compact partitions



Disadvantages

- Inherently mesh-based
- Can not be used on particle data directly
- Slower than geometric methods

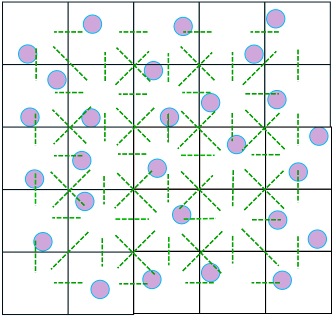
Graph partitioning

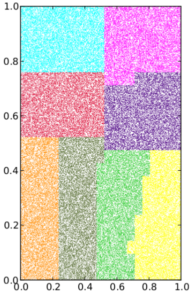
Algorithm and Advantages

- Interpret particle/cell data as graph nodes and neighborhood as edges
- Cell based graph partitioning has fixed/known neighbors
- Generates closed and compact partitions

Disadvantages

- Inherently mesh-based
- Can not be used on particle data directly
- Slower than geometric methods

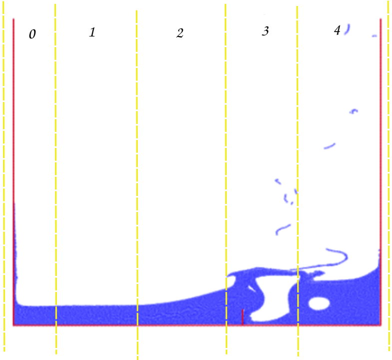




How do they do it?

- SPhysics
- GADGET2
- Ferrari et al.

SPhysics



- Incompressible free surface flows
- Fixed domain
- Super-linear parallel speed-up

GADGET2

- Astrophysics and Cosmology
- Massive N-Body and SPH simulations
- Free/Periodic boundaries
- 2D & 3D Space Filling Curves for Load-balancing

Ferrari et al.

- Free surface flows
- Cells as partitioning objects
- Serial METIS graph-partitioner for Load-balancing

Scope of this work

Algorithms

- Geometric (RCB, RIB, HSFC)
- METIS

Applications

- Free surface flows

Scope of this work

Algorithms

- Geometric (RCB, RIB, HSFC)
- METIS

Applications

- Free surface flows

- Partitioning quality
- Execution times
- Scale-up

Limitations of this work

- 2D
- Load balancing at every time step
- Particle based partitioning
- Graph partitioners not benchmarked

Outline

1 Smooth Particle Hydrodynamics
- The serial algorithm
- The parallel algorithm
- Load Balancing

2 Load-balancing
- Load Balancing techniques
- Algorithms
- How do they do it?

3 Results
- Illustration
- Results
- Conclusion and Further work

Problem description

Problem

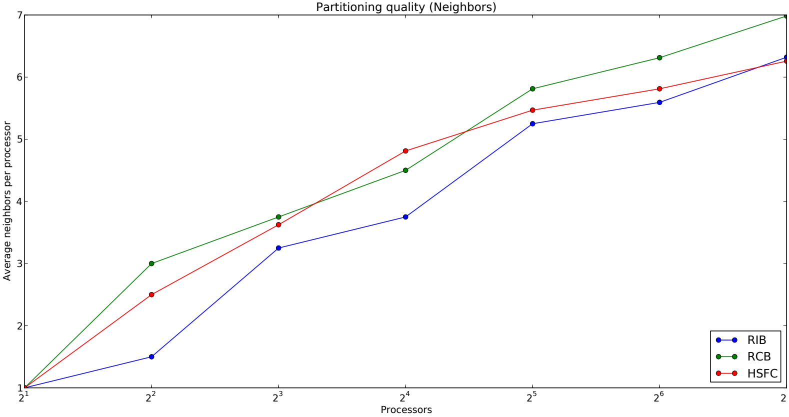
- Dam break, Sloshing flows
- Fluid + boundary conditions
- $N_p \approx O(0.1M)$

Algorithms

- RCB
- RIB
- HSFC

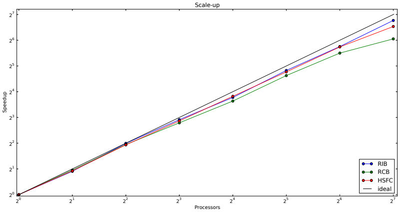
Neighbors per processor

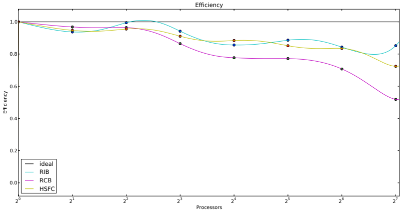
Dam-break, $N_p \approx 10M$, per-iteration



Scale-up & Efficiency

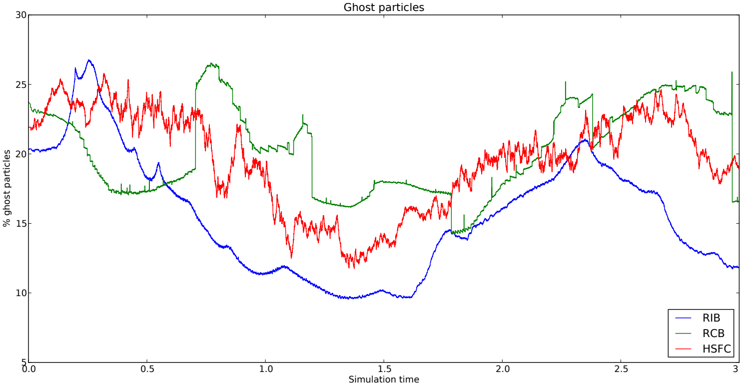
Dam-break, $N_p \approx 10$M, per-iteration





Ghost particles

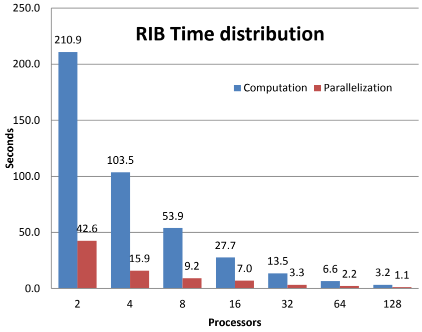
Dam-break, $N_p \approx 1M$, 8 partitions, Simulation time $= 3sec$

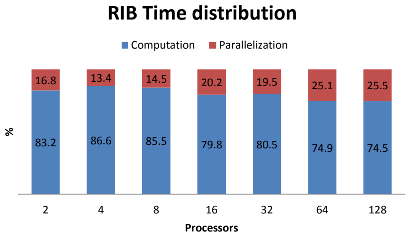


Method	Average % ghosts/iteration
RCB	20.24
HSFC	19.57
RIB	15.59

Time distribution : RIB

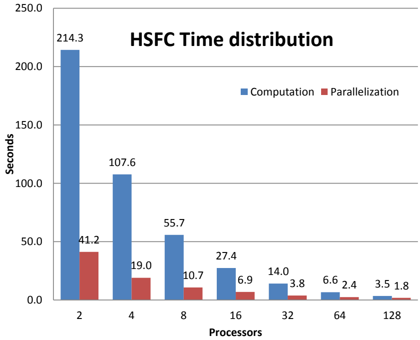
Dam break, $N_p \approx 10M$, per-iteration

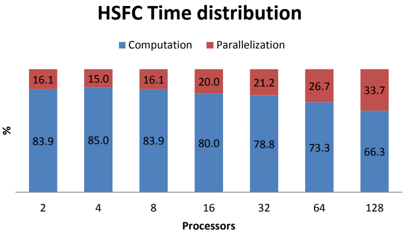




Time distribution : HSFC

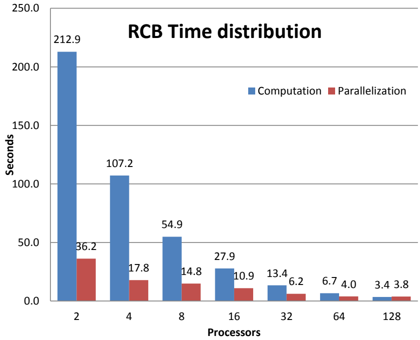
Dam break, $N_p \approx 10M$, per-iteration

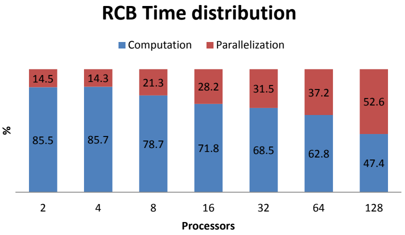




Time distribution : RCB

Dam break, $N_p \approx 10\text{M}$, per-iteration





Further work

- 3D
- Evaluation of Graph based partitioners with cell-partitioning
- Periodic load-balancing
- PyZoltan as the PySPH's parallel module

Code

- PySPH: `http://pysph.googlecode.com`
- zsph: `http://bitbucket.org/kunalp/zsph`

Thank you!