Science & Technology Facilities Council

MANCHESTER 1824

Developing an Extensible, Portable, Scalable Toolkit for Massively Parallel Incompressible Smoothed Particle Hydrodynamics (ISPH)

Dr Xiaohu Guo

Hartree Centre, STFC

Dr Benedict Rogers, Dr Steven Lind, Prof. Peter Stansby

SPH Group, School of Mechanical, Aerospace and Civil Engineering, University of Manchester

SPHERIC BEIJING Oct 2017

Science & Technology Facilities Council

MANCHESTER 1824

Overview

- Motivations
- Data preconditioning for extensibility and neighbour list searching
- Domain Decomposition and Dynamic load balancing with unstructured communication toolkit
- Particles reordering for ISPH
- Scalable sparse Linear solver for ISPH
- Applications and Summary

Science & Technology Facilities Council

MANCHESTER 1824

Motivation

- The stability, accuracy, energy conservation, boundary conditions of ISPH have been greatly improved (*S. Lind, H. Gotoh and A. Khayyer*)

- Computational challenges:
  - Particles irregular distribution due to solving complex, nonlinear and distorted flow.
  - Inherited SPH intrinsic computing bounded property.
  - Due to requirement solving pressure Poisson equation using sparse linear solver, ISPH is both memory bounded and computing bounded.

A Toolkit to facilitate ISPH software development for scientific users.

Incompressible SPH Implementation

- ISPH_DF – A Divergence-free Projection method in SPH

$$\mathbf{r}_i^* = \mathbf{r}_i^n + \delta t\, \mathbf{u}_i^n$$

Particles are advected to $\mathbf{r}_i^*$

LOOP 1

$$\mathbf{u}_i^* = \mathbf{u}_i^n + \left(\nu \nabla^2 \mathbf{u}_i^n + \mathbf{F}_i^n\right)\delta t$$

Calculate $\mathbf{u}_i^*$ without pressure gradient.

LOOP 2

$$\nabla \cdot \left(\frac{1}{\rho}\nabla p^{n+1}\right)_i = \frac{1}{\delta t}\nabla \cdot \mathbf{u}_i^*$$

Solve pressure Poisson equation to get pressure P_i^{n+1}.

LOOP 3

$$\mathbf{u}_i^{n+1} = \mathbf{u}_i^* - \frac{\delta t}{\rho}\nabla p_i^{n+1}$$

Calculate velocity $\mathbf{u}_i^{n+1}$.

$$\mathbf{r}_i^{n+1} = \mathbf{r}_i^n + \delta t\left(\frac{\mathbf{u}_i^{n+1} + \mathbf{u}_i^n}{2}\right)$$

Particles' positions are centered in time.

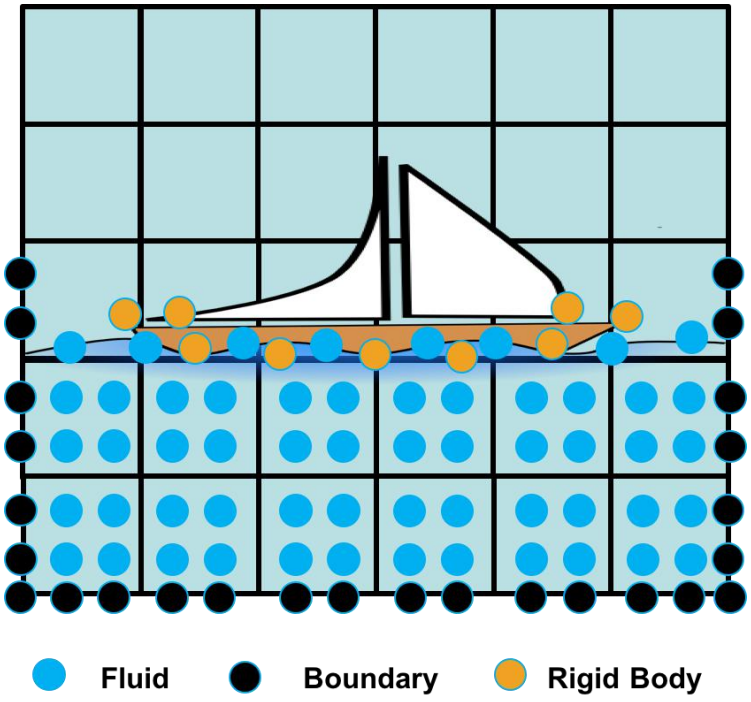
LOOP 4

$$\delta \mathbf{r}_s = -\mathcal{D}\left(\frac{\partial C}{\partial s}\mathbf{s} + \alpha\left(\frac{\partial C}{\partial n} - \beta\right)\mathbf{n}\right)$$

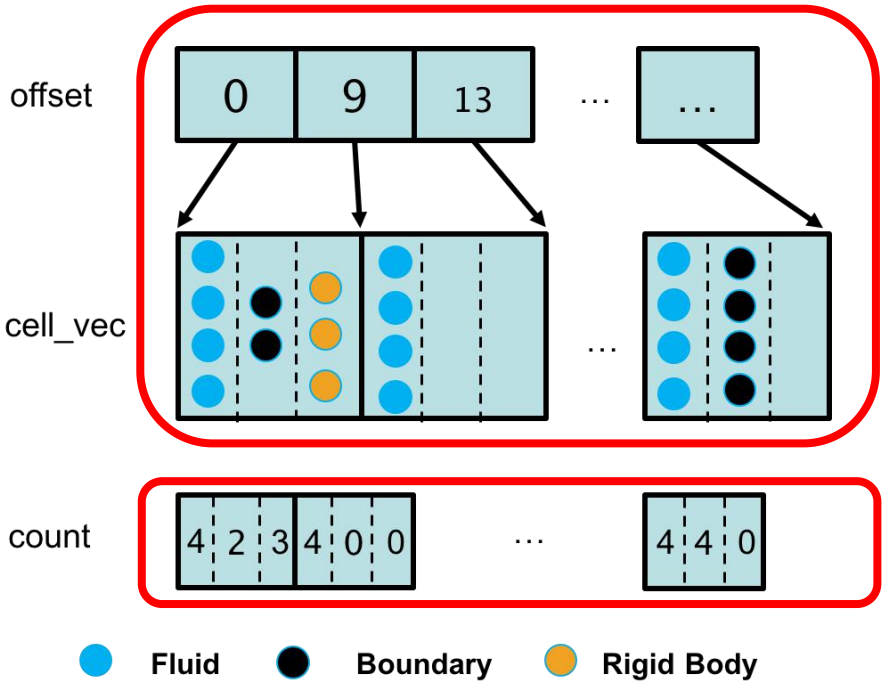
Shift particles to maintain stability.

*ISPH (Cummins & Rudman 1999)

ISPH Data Preconditioning and Nearest Neighbour List Searching

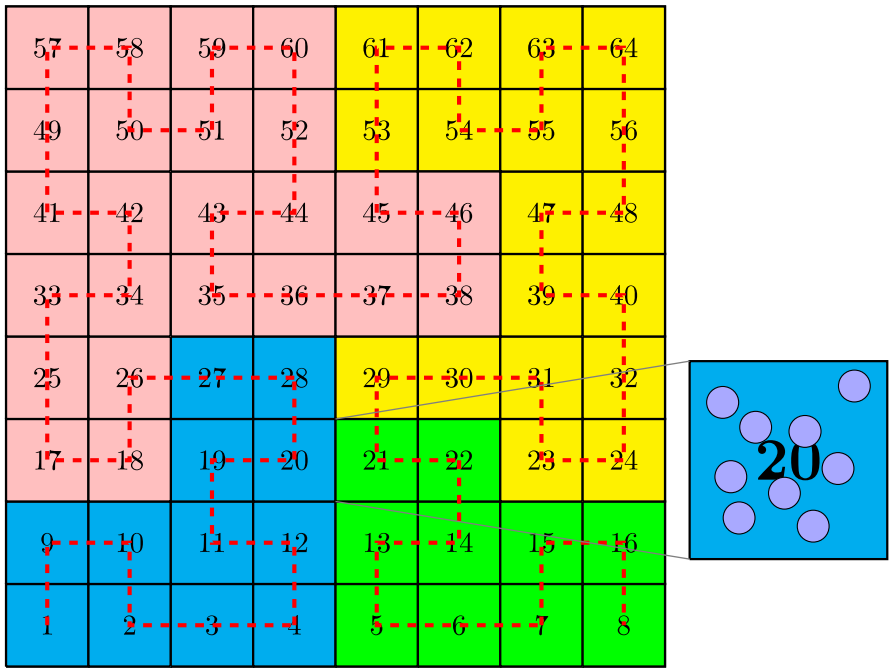


preconditioned dynamic vector
(J. M. Domnguez et.)



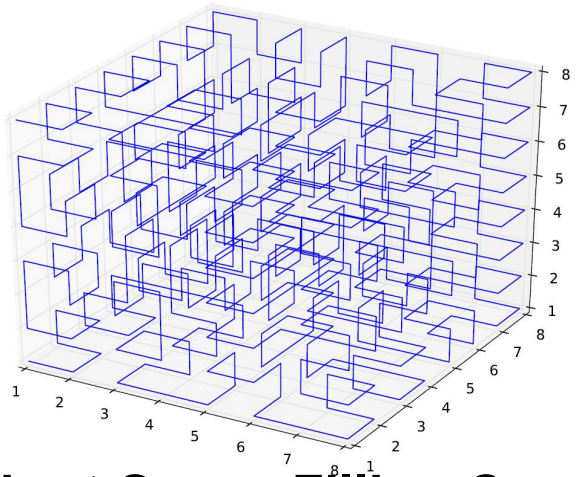
Provide extensibility if multiple types of particles involved.

Domain Decomposition and Dynamic load balancing with SFC

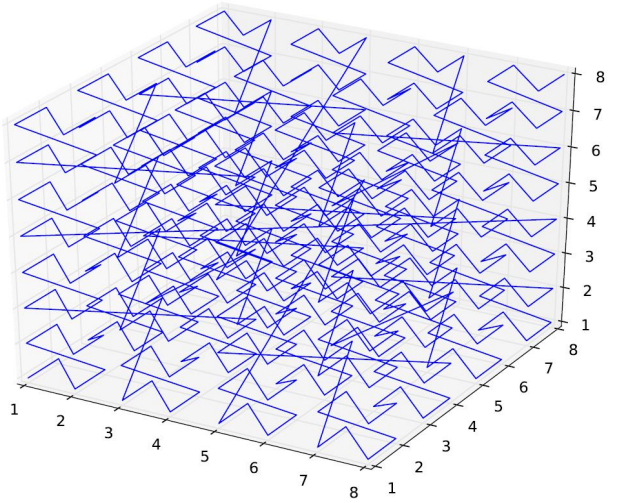


Partitions: P_0=blue, P_1=green, P_2=yellow, P_3=pink

We use the cell as partition object, calculate SFC according to cells

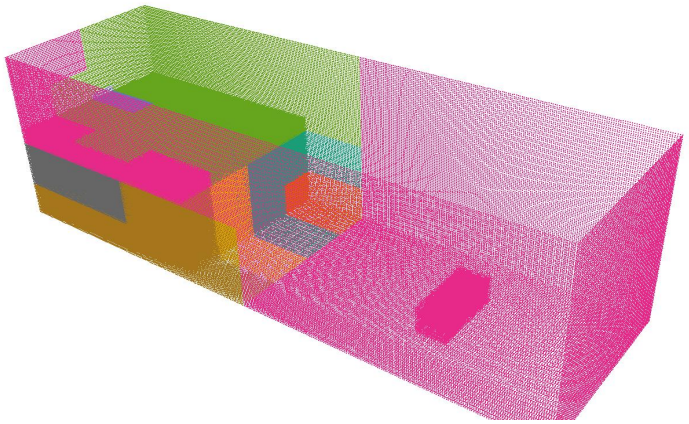


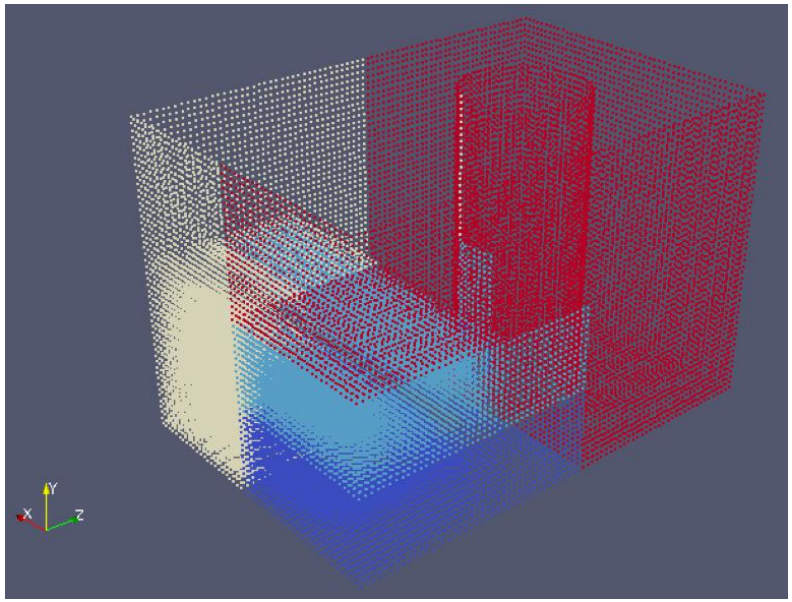
Hilbert Space Filling Curve:



Morton Space Filling Curve:

There is no extra effort for the domain decomposition and dynamic load balancing for solid objects

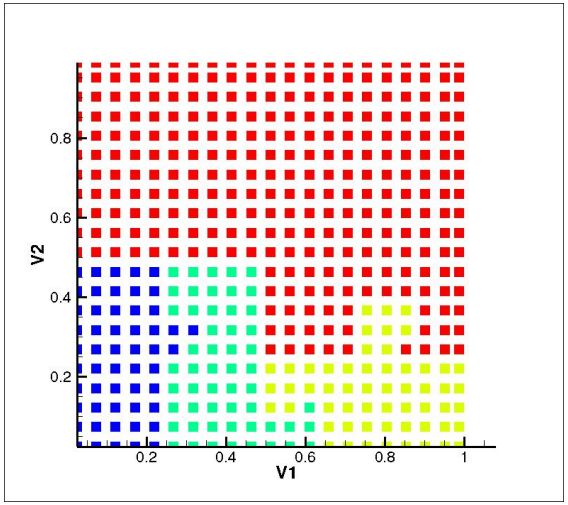




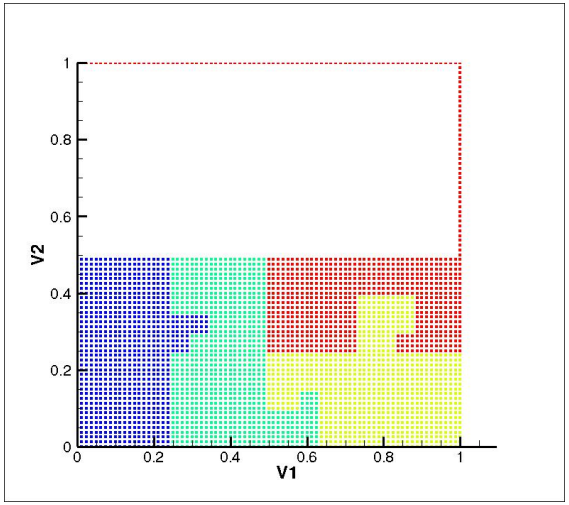
Colours denote partitions

Unstructured communication

- As a result of using space filling curve, the domain shape become irregular.
- Each processor has cells (contain particles) to send to other processors, but no processor knows what messages it will receive.
- ISPH produces a data structure called a **communication plan** which encapsulates the basic information about the communication operation



Cells Decomposition



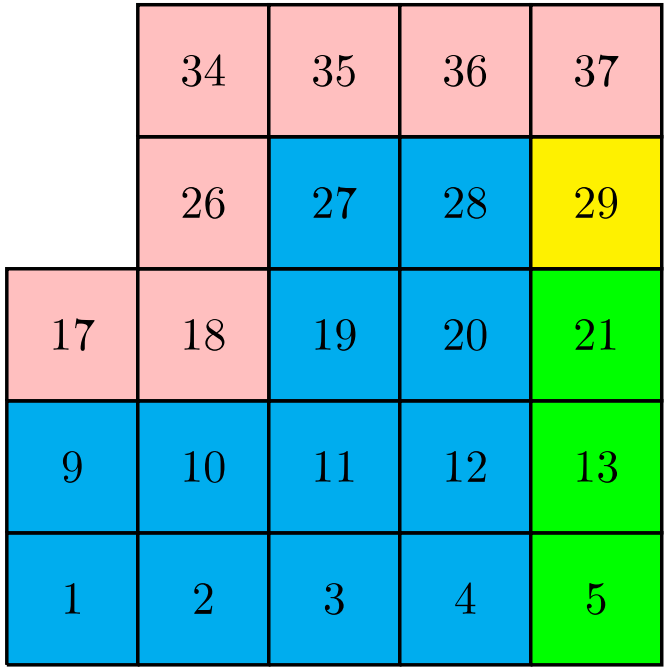
Particles Decomposition

Science & Technology Facilities Council

MANCHESTER 1824

ISPH Domain Decomposition and Dynamic load balancing software reengineering

Halo Exchange



Halo Send and Receive Plan for cells:

- $\mathbf{P_0}$:SEND(to send to $\mathbf{P_3}$): 9,10,11,19,27,28
- $\mathbf{P_0}$:SEND(to send to $\mathbf{P_2}$): 20,28
- $\mathbf{P_0}$:SEND(to send to $\mathbf{P_1}$): 4,12,20,28
- $\mathbf{P_0}$:RECV(from $\mathbf{P_3}$): 17,18,26,32,35,36,37
- $\mathbf{P_0}$:RECV(from $\mathbf{P_2}$): 29
- $\mathbf{P_0}$:RECV(from $\mathbf{P_1}$): 5,13,21

Partitions: $\mathbf{P_0}$=blue, $\mathbf{P_1}$=green, $\mathbf{P_2}$=yellow, $\mathbf{P_3}$=pink

Partition P0 halo plan setup

ISPH Halo Exchange initial implementation

```
Do t = 1 → total_number_of_timesteps
   !! setup halo exchange plan
   call halo_plan_setup
recvs)
   !! update halo objects
   call halo_update_integer()
   !! update the physical data blocks to
   !! an appropriate partition
   call halo_update_rea
    perform local calcul
   !! destroy halo exchan
   !! partition changed
    If (change) then
      call halo_plan_destroy(halo sends, halo
recvs)
    endif
end do
```

New Syntax of ISPH Halo Exchange

```
Do t = 1 → total_number_of_timesteps
   !! setup halo exchange plan
   setup(halo_sends,
   !! an appropriate partition
   call halo_update(halo_sends, halo_recvs,
Var)
   lculation for Var
   change plan if
   ed
   call halo_plan_destroy(halo sends, halo
recvs)
   endif
end do
```

No Big Deal !

Big deal in terms of new model development !

Two Major steps to reduce memory footprint.

Not saving the neighbour list

LOOP 1

$$\mathbf{u}_i^* = \mathbf{u}_i^n + \left(\nu\nabla^2\mathbf{u}_i^n + \mathbf{F}_i^n\right)\delta t$$

LOOP 2

$$\nabla\cdot\left(\frac{1}{\rho}\nabla p^{n+1}\right)_i = \frac{1}{\delta t}\nabla\cdot\mathbf{u}_i^*$$

LOOP 3

$$\mathbf{u}_i^{n+1} = \mathbf{u}_i^* - \frac{\delta t}{\rho}\nabla p_i^{n+1}$$

LOOP 4

$$\delta\mathbf{r}_s = -\mathcal{D}\left(\frac{\partial C}{\partial s}\mathbf{s} + \alpha\left(\frac{\partial C}{\partial n} - \beta\right)\mathbf{n}\right)$$

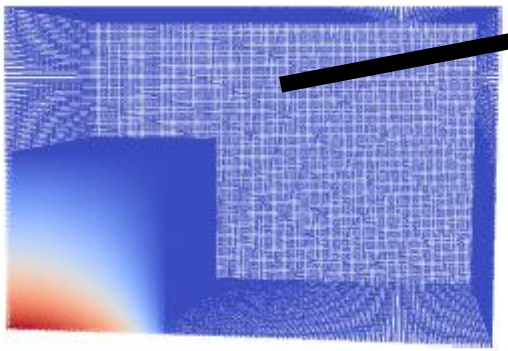
- Instead looping through all particles ***i***, we loop each cell
- In each cell, we then have a subloop to calculate all properties for each particle ***i***
- Such way, we don't need save each particle ***i***'s neighbour list.
- Within each loop, need reduce redundant calculations,

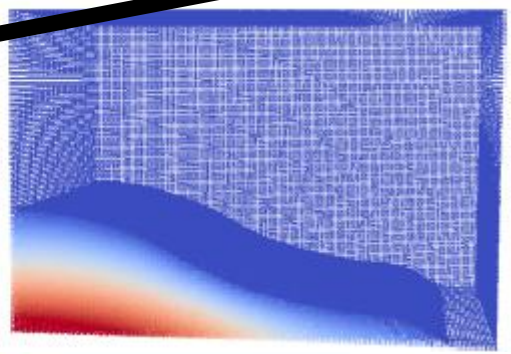
NOT saving neighbour list is much faster and also provide solutions for particles ordering.

Bounding box creation for MPI partition mapping

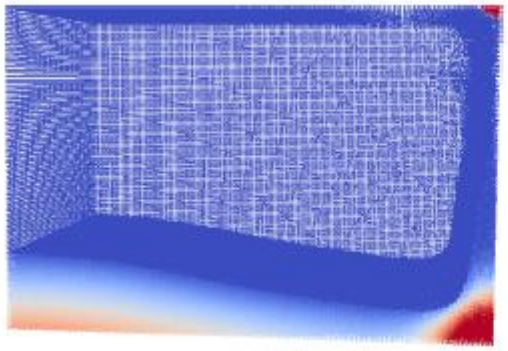
The number of cells can be huge.

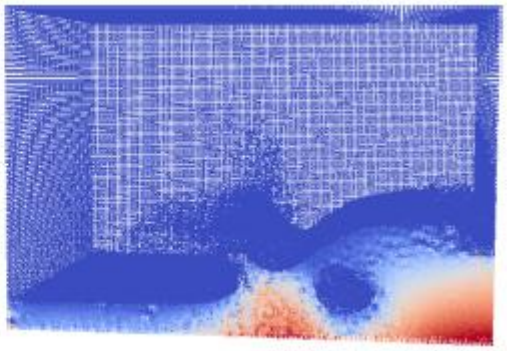
e.g. for 3D dam break, we have 1 billion particles, the number of cells is ≈1/300 billion





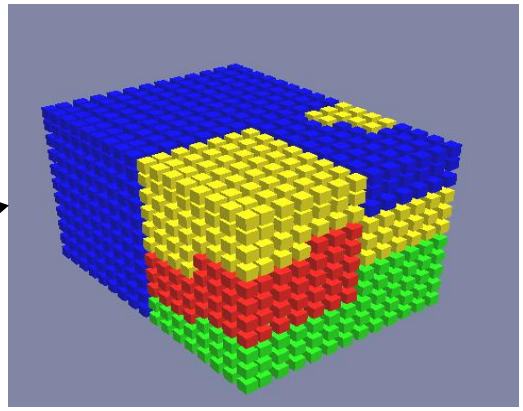
ble

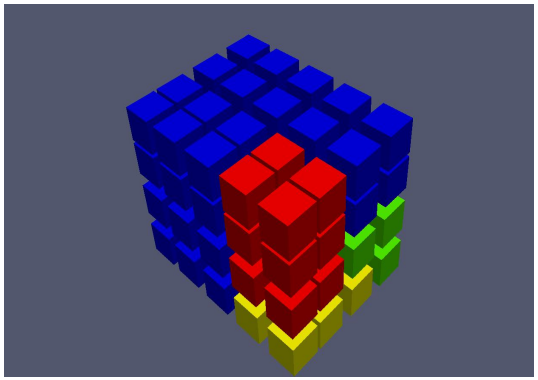




ells

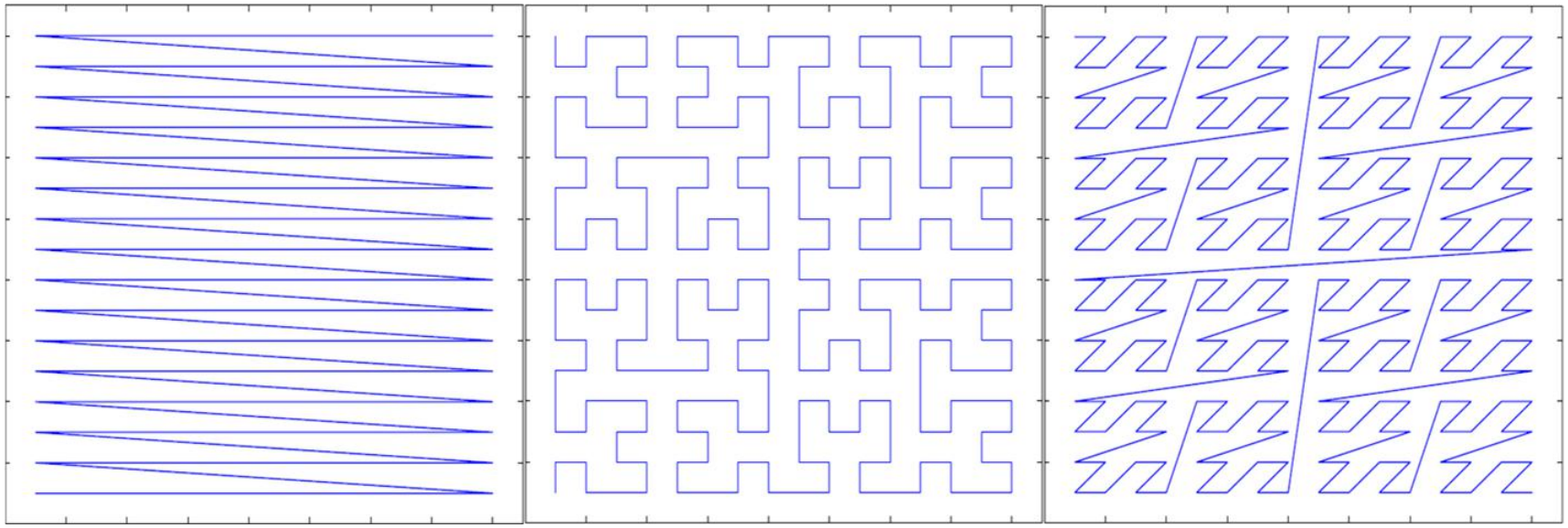
Cells





Key Point: **Number of Bounding Boxes** **≈ 20% Number of 2*h* cells**

Particles reordering for ISPH



Particle Reordering – Why?

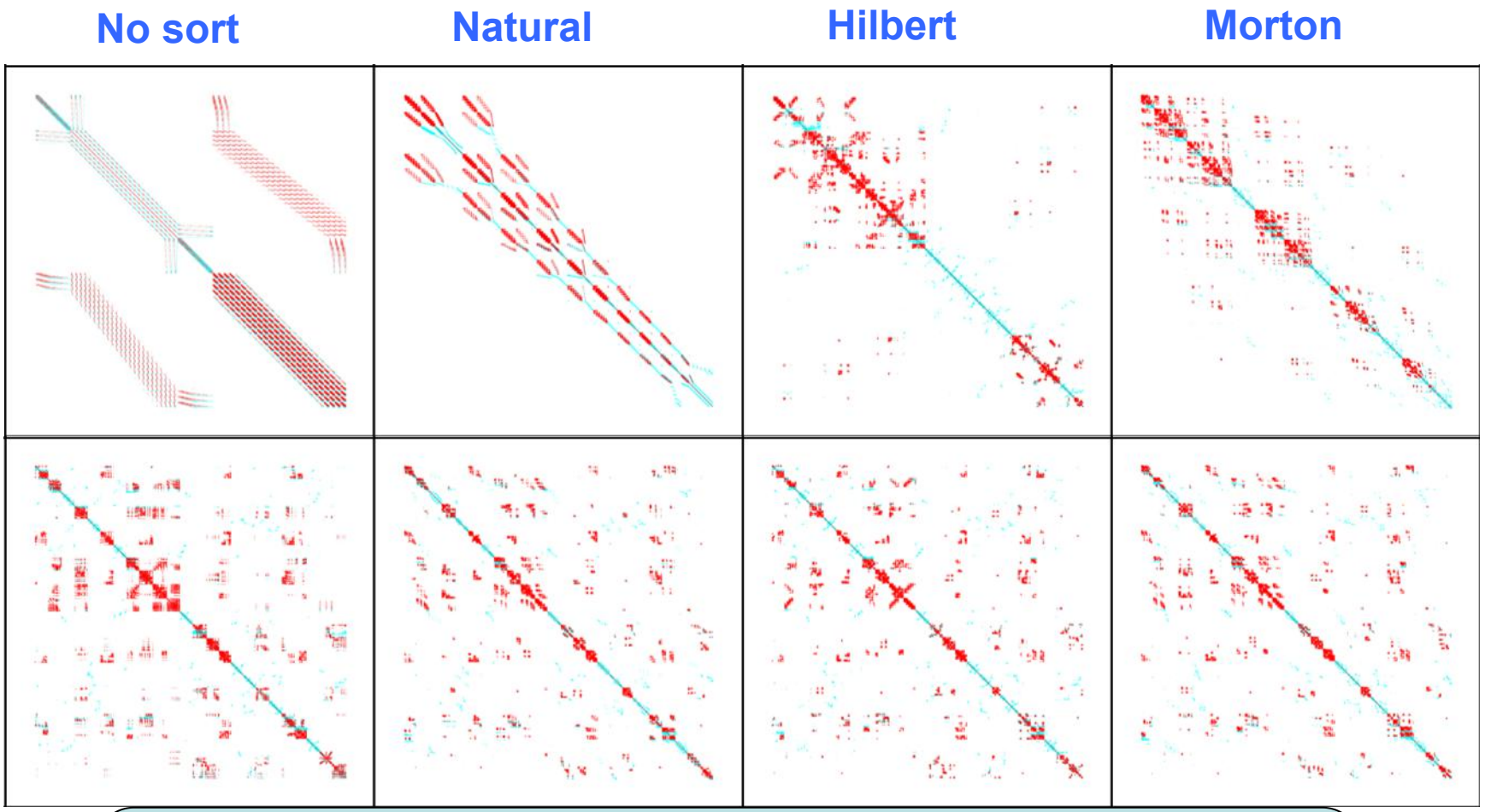
By reordering particles to be close to each other in memory to reduce cache access & hence reduce computation time

How?

Particles order are ordered by ordering the cells that contain them using either Natural, Hilbert SFC or Morton SFC.

→ **GAIN:** Light overhead to calculate SFC using bit shifting

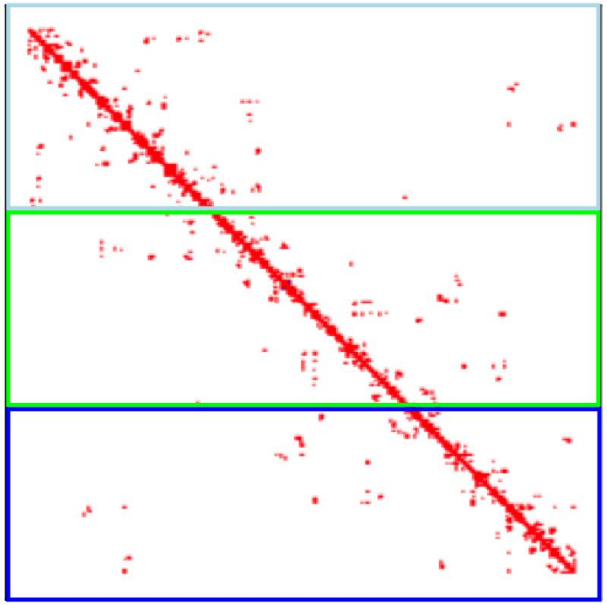
Effects of sorting on 3D PPE matrix structure



With particles sorting, in Serial, reduce matrix bandwidth. In parallel, needs efficient global particles sorting—expensive !

Flexible Sparse linear solver implementation using PETSc

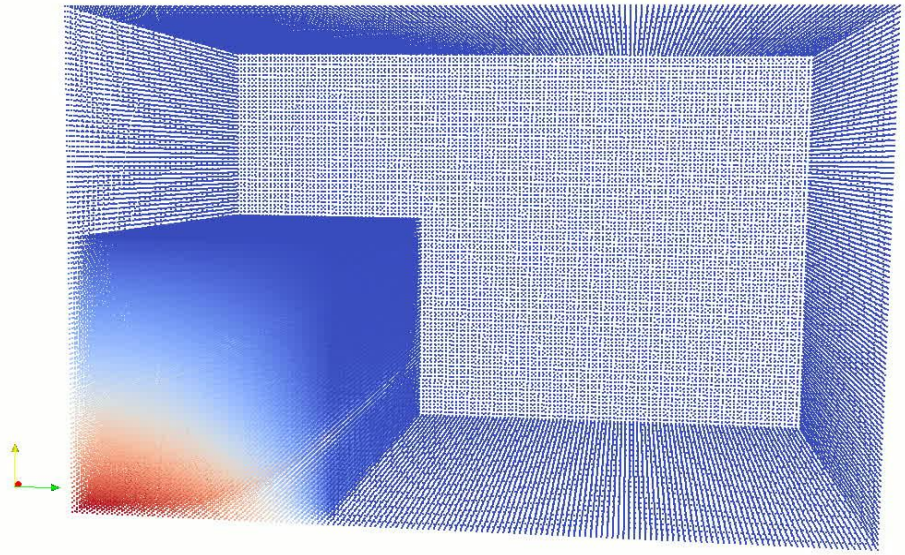
- We use sparse linear solver library PETSc which provide rich options of Krylov subspace solvers and preconditioners.
- Assembly by each row with MatsetValues, not MatsetValue
- Involves set boundary values, correction method, this has be done before MatAssemblyBegin/End, to reduce memory access to PETSc memory space.
- Provide a efficent way to explore PETSc preconditioners and Krylov subspace solvers

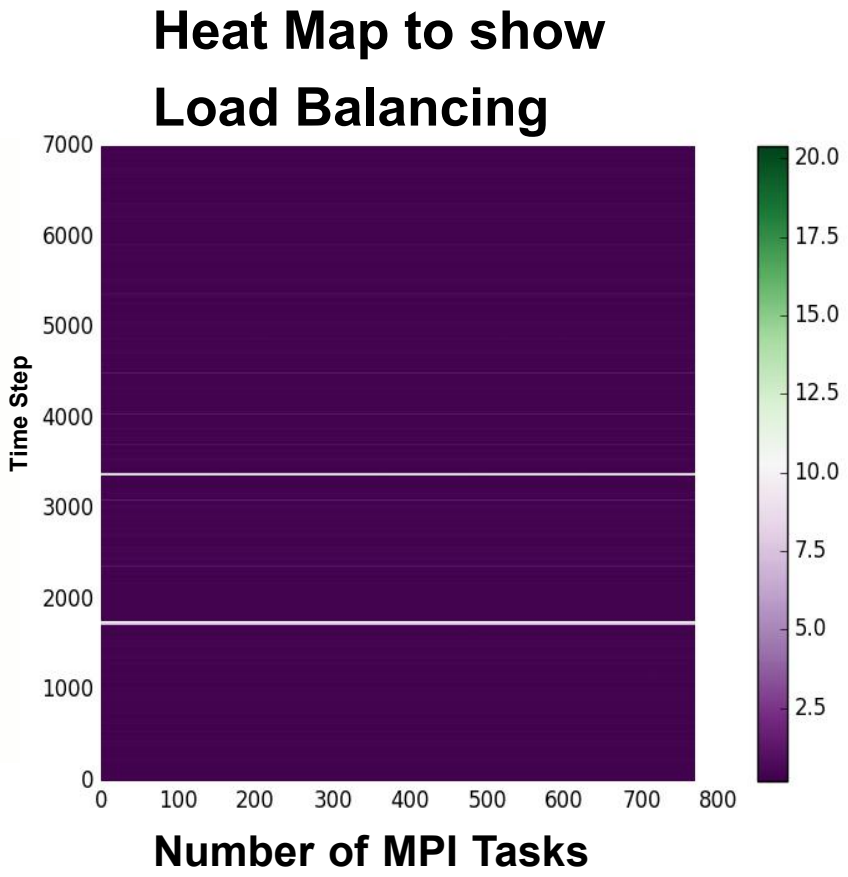


PETSc partition matrix by rows, each colour represent the partition of the PPE matrix

mpirun –np 3 isph3d –pc_type hypre –pc_hypre_type boomeramg –ksp_type gmre

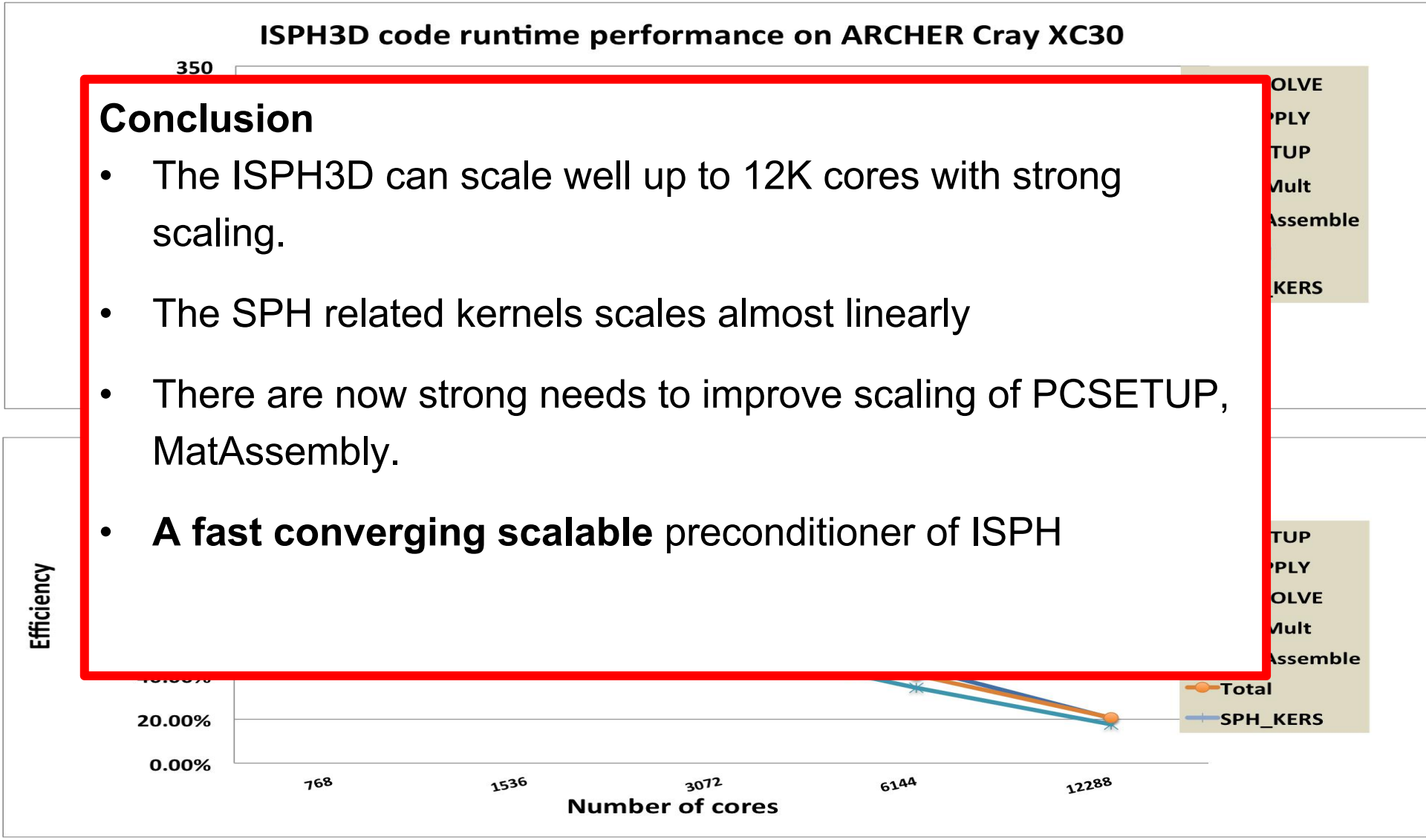
Dynamic Load Balancling for 3D Dambreak with Dry Bed using ISPH





With around 100 million partilces using 768 MPI partitions

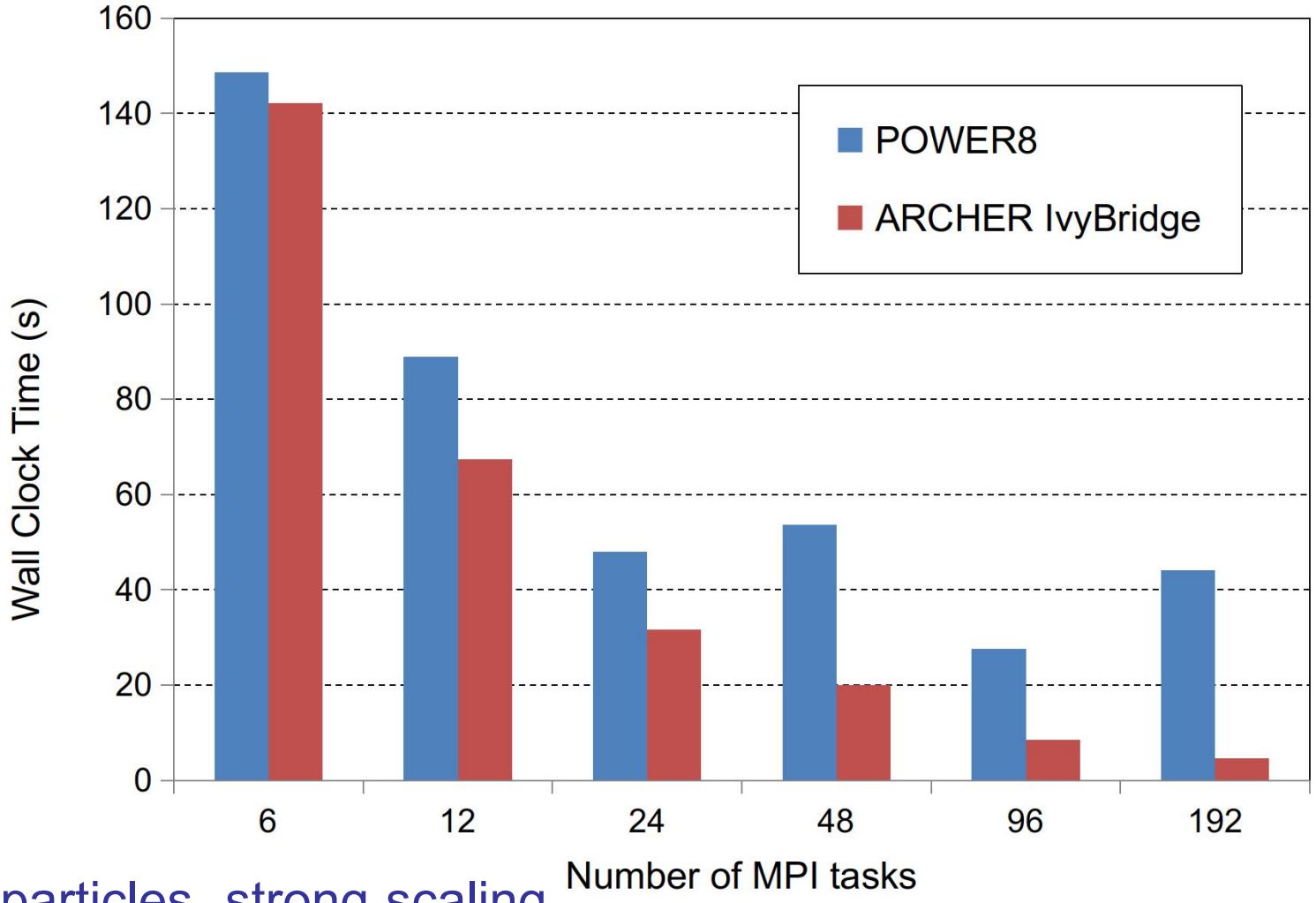
ISPH3D Overall Performance Analysis: strong scaling



Conclusion

- The ISPH3D can scale well up to 12K cores with strong scaling.
- The SPH related kernels scales almost linearly
- There are now strong needs to improve scaling of PCSETUP, MatAssembly.
- **A fast converging scalable** preconditioner of ISPH

ISPH Performance on IBM Power8

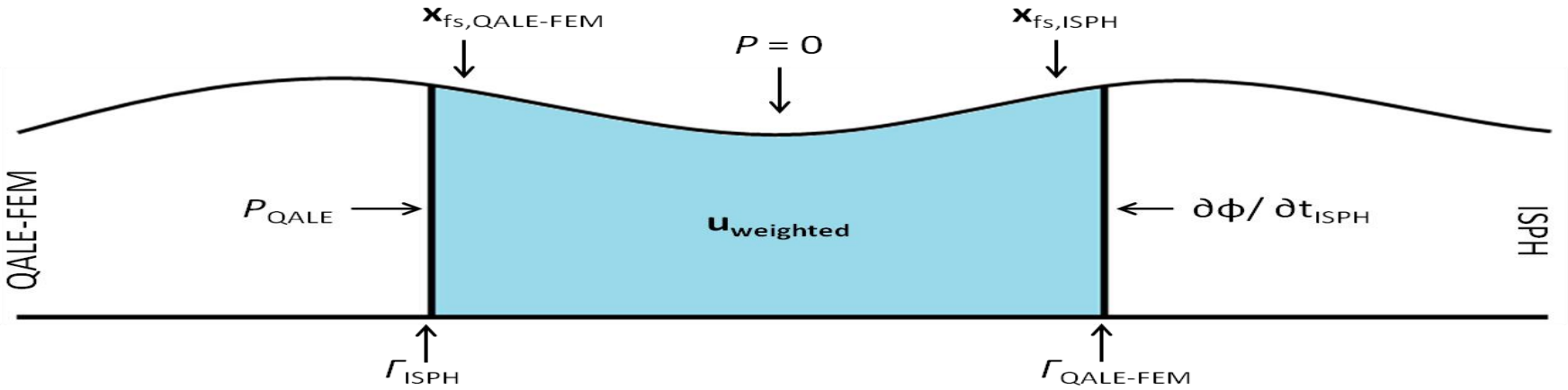


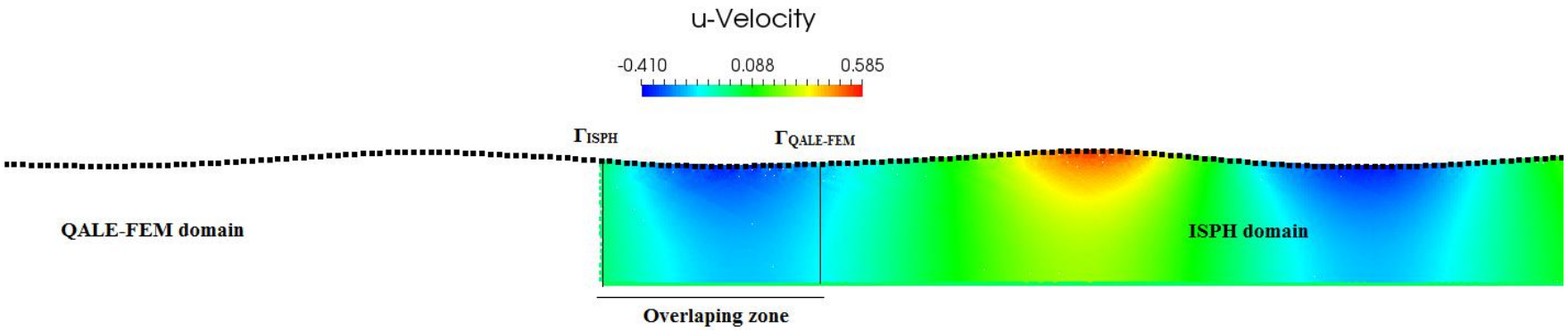
ISPH 5M particles, strong scaling

ARCHER Cray XC30 each node 2x 2.7 GHz, 12-core E5-2697 v2 (Ivy Bridge)

Scales well to 24 cores, SMT helps at 96 virtual cores (SMT=4)

Application: Coupling of ISPH + QALE-FEM





Summary so far …

- Domain Decomposition and Dynamic load balancing
- Efficient nearest neighbour searching kernel Implementation
- Particles reordering: performance improvements
- Flexible implementation of using PETSc solving PPE for ISPH
- The code can now scale well up to tens of thousands cores which enable the ISPH to work on large-scale real applications
- Although it's implemented together with ISPH, it can also benefit other projection based particle method as a separate parallel library – **separation of concern for MPI based parallelism**

Things happening …

- **Portability study** for heterogeneous computing architectures.
- Handling **complex geometries**
- Fast converging and scalable preconditioners designed for ISPH
- Boundary conditions, MBT, Inlet/Outlet.
- support of **multiphysics coupling** in large scale.
- **Preprocessing and post processing tools** targeting to complex geometries with large number of particles for ISPH3D
- Efficient parallel **I/O.**
- **Strong scaling** for billion particles above

Acknowledgements

- The authors would also like to acknowledge the funding support under the embedded CSE programme of the ARCHER UK National Supercomputing Service (http://www.archer.ac.uk). The authors would also like to thank the EPCC eCSE support team for their help throughout this work.
- The work was partially funded by the EPSRC Grant ”SERT: Scale-free, Energy-aware, Resilient and Transparent Adaptation of CSE Applications to Mega-core Systems ”. Grant Number:EP/M01147X/1.
- The work was carried out on ISPH software which was original funded by the EPSRC Grant ”An incompressible smoothed particle hydrodynamics (ISPH) wave basin with structure interaction for fully nonlinear and extreme coastal waves”. Grant Number: EP/H018603/1, EP/H018638/1.

Science & Technology
Facilities Council

Thanks !