,

Developing Massively Parallel ISPH with Complex Boundary Geometries

Xiaohu Guo[a], Benedict D. Rogers[b]

[a] *Hartree Centre,*
Science and Technology Facilities Council,
Daresbury Laboratory, Warrington WA4 4AD UK
[b] *School of Mechanical, Aerospace and Civil Engineering,*
University of Manchester, Manchester, M13 9PL, UK

Abstract

In order to develop the 3-D incompressible smoothed particle hydrodynamics (ISPH) software into an attractive engineering tool for complex full-scale engineering problems, particularly in 3-D simulations, this project is to add new functionalities to deal with arbitrary complex geometries which will enable a wide range of applications with fluid-structure interaction. This requires multiple developments before such functionality can be fully and rigorously implemented. The aim of this project has therefore been to develop unique ISPH functionality for real applications that is scalable over many thousands of processing cores.

The following list highlights the major developments:

- Particle preconditioning kernel: divide different type of particles into cells, comparing with dynamic vector approach, we have added one additional array to record their starting address for each type of particles.
- Improvement of nearest neighbour list searching kernel by looping through each cells neighbour cells instead of saving the particles neighbour list, which reduces the memory footprint.
- Update and optimised the halo exchange kernel due to rearranged the data layout the carried out the above.
- Parallel implementation of the multiple boundary tangent method (MBT), this requires parallel implementation the kernels of testing particle in the body of complex geometries and solving the issue of mirror particles over generation.
- Parallel implementation of local uniform stencil method with solid object surface triangulation kernel
- Updated the moment equation solver and removed any unnecessary IF branches for the special treatment of boundary particles, velocity updates for only special types of particles and verified with dam break and still water cases with up to 100 million particles using up to 12,000 MPI partitions for Work Package 1.

Keywords: Incompressbile SPH; MBT; LUST; PETSc; Zoltan ; MPI;

1. The ISPH eCSE project

The stability, accuracy, energy conservation, boundary conditions of the projection based particle method such as incompressible smoothed particle hydrodynamics ISPH [1] have been greatly improved [2, 3, 4]. However, to make the code feasible for real practical problems requires that the code can solve the flow with arbitrary geometry structures.

Building on the successful work of previous eCSE-funded project (eCSE01-003), the ISPH eCSE06-09 project mainly comprised of four work packages (referred to as WP1, WP2, WP3, WP4). WP1 Implementing the preconditioning particles data offers the ISPH code flexibility and capability to implement the new boundary methods and the

new flow control equations. WP2 is to implement multiple tangent boundary method (MBT). WP3 is to implement local uniform stencil boundary method (LUST). WP4 is application verification and report.

The remaining part of this report is organised as follows: In the next section we describe using data preconditioning techniques to explore data locality and manage particle indices with various types of particles. Section 3 describes the parallel implementation of multiple tangent boundary method, and the Section 4 explains how the local uniform stencil boundary method is implemented. Finally, Section 6 are conclusions and the future work.

2. WP1: Preconditioning the particles data to explore the data locality and improve ISPH software extensibility

The previous project, eCSE01-003, had delivered a massively parallel ISPH code but without being able to represent complex geometries. For simulations with real flow applications often involve different types of the particles. For example, in Fig. 1a, there are fluid particles which can have different sub-types of particles if simulating multi-phase flow, the particles representing the floating rigid body, and the domain boundary particles. The particles representing the domain boundaries and the floating rigid body particles can be further divided according to different SPH boundary techniques (each of which employs different methodologies). When solving these applications at large scales, managing data locality is particularly challenging due to particles moving over time. In this WP we have introduced data preconditioning techniques for the first time to ISPH to explore data locality and manage particle indices with various types of particles.

ISPH is an interpolation technique where the value of a property (position, velocity, acceleration, pressure) is computed by summing the contributions from neighbouring particles (see [1] for full details). In order to localise the neighbour search, particles are stored in cells. These cells are square in 2-D or cubes in 3-D whose dimension is on the order of the radius of the interpolation region around each particle. Fig. 1a shows a schematic representation of cells containing particles for a highly simplified 2-D case.

Each particle is assigned a unique identification (ID) number. The list of particles is stored in an array **cell_vec** where the particles are stored in sequence according to type, that is particles representing fluid, boundaries and rigid bodies.

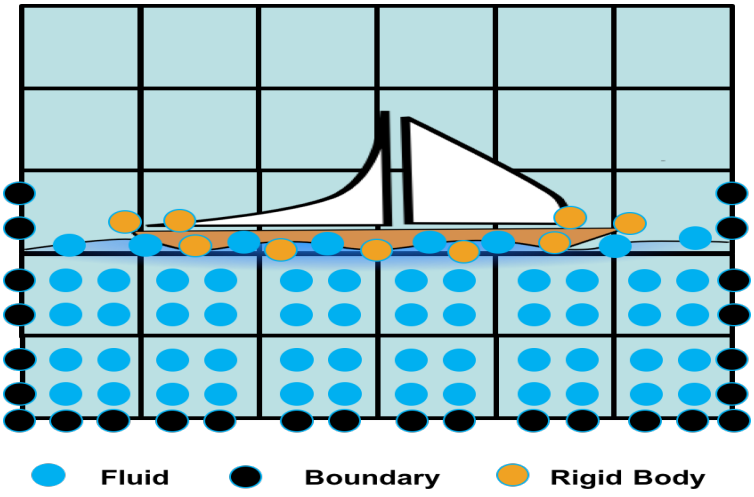
Considering extensibility, the ISPH3D code provides a choice of either a linked list or newly implemented preconditioned dynamic vector (PDV) [5] is enabled for the nearest neighbour searching operation. Fig. 1b shows how we use PDV with consideration of different types of particles. Similar to PDV [5], an array **offset** contains the values pointing to the index of the first particle in each cell, **cell_vec** contains the particle indices where each cell has fluid, boundary and rigid body particles arranged in sequence. Array **count** is used to count the number of fluid, boundary and rigid body particles in each cell. This new structure-of-array-of-structures(SOAOS) showed in Fig. 1b requires new implementations of both the nearest neighbour list searching kernel and the MPI halo exchange kernels for send buffer packing and receive buffer unpacking with consideration of different types of particles comparing with our previous approach[6]. Internally, the data layout/arrangement of types of particles are fixed for PDV data structures, but the sequence of generating the different types of particles in the prepossessing stage and then in input file can be any order according to the users' request.

3. WP2: Parallel implementation of the multiple boundary tangent method (MBT) for complex boundary geometries

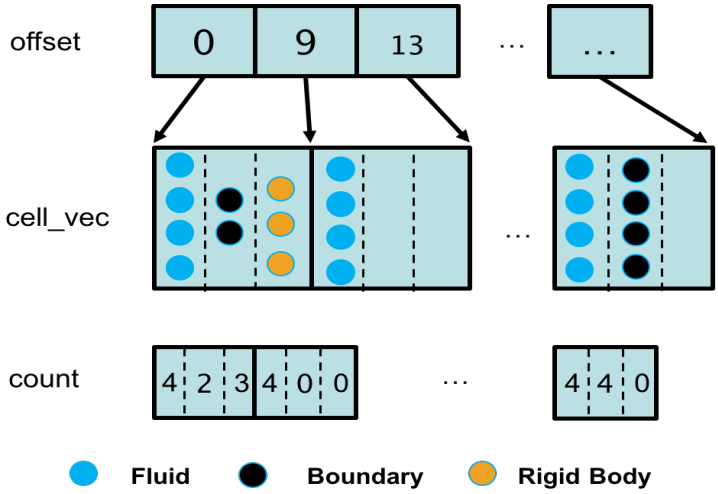
Accurate boundary method are one of the major challenges for SPH based methods. Indeed boundary conditions have been identified as a Grand Challenge by the international SPH community (http://spheric-sph.org). Large errors may caused by inaccurate approximation of complex physical boundary of object. Based on previous successful work [7], we have proposed to implement multiple boundary tangent method(MBT)[8] in this work package to facilitate the difficulties to approximate certain complex geometries.

3.1. Basic MBT algorithm analysis

To use MBT method, we need to know the normal of each solid surface particles. As most of the CAD software tools provide the normal direction for the surface mesh of the solid object or the normal direction can be easily



(a) particle types



(b) PDV data structure

Figure 1: Diagram (a) is to show the different types of particles interaction in the same cell, in this case, fluid particles, boundary particles and rigid body particles, (b) is to show using PDV with consideration of different types of particles

calculated from CAD input, therefore we only allocate a vector (xnb, ynb, znb) to save the normal of the solid object surface particles.

The basic algorithm of MBT can be summarised in the Algorithm 1:

Algorithm 1: MBT Basic Algorithm Description

Step 1: Read in the solid object boundary particles normal vector into xnb, ynb, znb.
Distributed together with the particles into each MPI partition.
Step 2:in each partition
j=mbt_index
for $k = 1 \rightarrow total\_local\_Solid\_particles$ **do**
  Step 2a: Find all the neighbour fluid particles P_i
  while fluid particle P_i is solid surface particle P_k's nearest neighbour particle **do**
    Step 2b: generate the mirror particle $P^{'}$ for particle P_i
    Step 2c: check the mirror particle $P^{'}$ inside solid
    if $P^{'}$ inside solid **then**
      assign $P^{'}$ index j
      set coordinate of MBT particle: x(j), y(j), z(j)
      set velocity of MBT particle: u(j), v(j), w(j)
      save MBT particle j's parent fluid particle: irelation(j)=i
      j=j+1
    end if
  end while
end for

In the **Step 2c**, we have implemented a boolean function called **pointInObject** to decide if a particle is located in the body of solid object. A limitation of the current project is that as we have only tested this function for a cylinder, further tasks need to be done for general complex geometries.

During the creation of MBT particles, there is an over-creation of MBT particles due to the fact that the fictitious image particles generated by neighbouring boundary tangents overlap. The overlapping contributions of mirrored MBT particles can be eliminated by determining the number of times a given fluid particle is mirrored into the influence domain of the associated fluid particle with respect to a boundary particles tangent line [8]. The contribution of each fictitious image particle to SPH summations is then weighted using **mbtweight**.

Algorithm 2 gives the details of counting the over generation of MBT particles.

Algorithm 2: Counting MBT particles over generation

Step 3: counting mirror particle over generation for fluid particle P_i mbtweight=1
for $k = 1 \rightarrow total\_local\_fluid\_particles$ **do**
  for $j_1 = 1 \rightarrow total\_neighbour\_particlesof particleP_i$ **do**
    for $j_2 = 1 \rightarrow total\_neighbour\_particlesof particleP_i$ **do**
      if j_1 and J_2 are mbt particles **then**
        if irelation(j_1)= irelation(j_2) **then**
          mbtweight(j)=mbtweight(j)+1
        end if
      end if
    end for
  end for
end for

However, the over generation also causes the issue of preallocation the memory for the fictitious image particles. Presently, the memory requirement for the generation of fictitious image particles is estimated prior to simulation.

There are advantages and disadvantages for MBT boundary methods:

- For complex boundaries, we only need boundary surface particle normal, and it does not require initial particle generation outside of boundary. This makes less requirement of prepossessing tools.
- Due to MBT particles over generation, the memory requirement cannot be precisely controlled. This is not ideal for large-scale implementation.

4. WP3: Parallel implementation of local uniform stencil method (LUST)

With consensus in the ISPH community yet to emerge on the optimal boundary condition, the performance of different boundary conditions is still being assessed. The LUST wall boundary scheme [9] is based on a uniform stencil of particles that move with fluid particles and the stencil is only activated when the fictitious particles overlap with the boundary (example 2-D definition see Fig 2) is used to discrete the wall in respect to the fluid particle. We use a pre-defined lattice of particles as a template and superimpose the uniform lattice on the fluid particle whose support is truncated, filtering out all the fictitious particles inside the fluid domain. The procedure completes the wall truncated support and recovers approximate first and second-order consistency near the wall boundary. The advantage of LUST comes from the local point of symmetry generation and its local uniformed stencil methodology that can be applied to any complex arbitrary geometry.

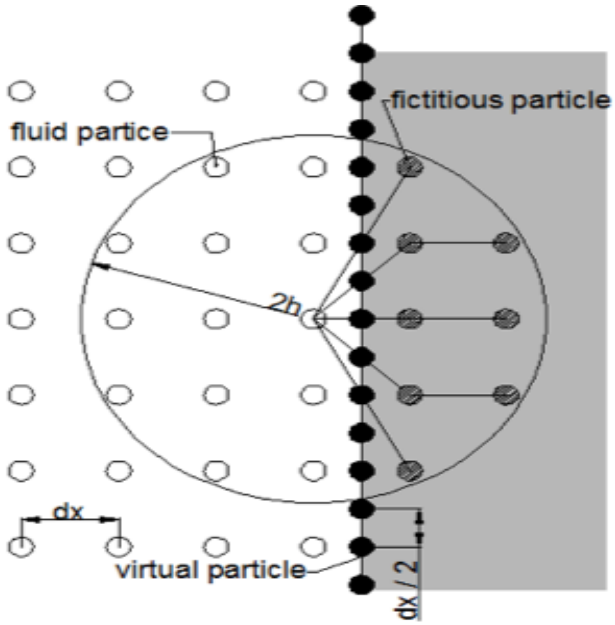


Figure 2: LUST boundary 2D diagram, [9]

4.1. LUST particles generation

We assume each fluid particle has uniform support of fictitious particles. When the support of a fluid particle is truncated from the solid wall represented by triangles, the latter arbitrary uniform stencil is applied to the fluid particle. By using the triangulated surfaces particles that are located within the fluid domain are discarded. The result is a uniform boundary stencil with regularly distributed fictitious particles.

The solid boundary in this work is represented by triangulated surfaces. When the support of a fluid particle is truncated, thus reducing the consistency of the local approximation, a pre-computed full support is imposed on the fluid particle. Fictitious particles within the domain are discarded and the remaining fictitious particles are used in the fluid interpolation by applying a local hydrostatic correction to the density and pressure. If the interior domain

particles are uniform then the approximation is first order consistent. Otherwise the interpolation consistency is reduced due to the interior domain disorder. Overall, the fictitious particle uniform stencil is said to be approximately first order consistent.

The Algorithm 3 gives the details of LUST boundary implementation.

Algorithm 3: LUST Basic Algorithm Description

Step 1: Precalculate the complete uniform support for arbitrary fluid particles within $2h$ support.
Step 2: Triangulation the solid object surface. calculate each surface triangle's centroid coordinates.
Distributed together with the particles into each MPI partition.
halo_update Solid boundary triangles.
Step 3:in each partition
j=lust_index0
for $k = 1 \rightarrow total\_local\_Solid\_particles$ **do**
  Find all the neighbour fluid particles P_i for each P_k
  while fluid particle P_i is solid surface particle P_k's nearest neighbour particle **do**
    generate the fictitious particles of the truncated area of the solid object for particle P_i
    if $P^{'}$ inside solid **then**
      save $P^{'}$ index j
      set velocity of fictitious particle: u(j), v(j), w(j)
      save fictitious particle j's parent fluid particle: irelation(j)=i
      j=j+1
    end if
  end while
end for

There are some key challenges that need to be solved in our implementation:

- Although the stencil for each near fluid object are the same, an if statement is required to identify when stencil particles are boundary particles or fluid particles, which is expensive for 3D simulations where the average stencil can contain up to 250 particles.
- Due to requirement of triangulation, we have to share all solid object particles.
- Triangles representing a 3-D surface will be distributed according to their centroid coordinates, but because we shared all solid particles objects, all triangles are also shared globally which will cause memory issues for large-scale simulations.

5. WP5: Results and Performance Analysis

5.1. Test cases

Three test cases have been used in this project. The 3-D dam break with a dry bed has been used in this paper as the benchmark test case for work package 1 and the second is dam break cylinder post test case. Fig. 3 shows the advantage of using an SFC-based method to perform domain decomposition for 3-D complex fluid, the method we use here does not require extra effort to deal with trade-off between dynamic load balancing and data locality.

The third test case is flow over cylinder. Figure 4 shows the 3-D flow field around a confined circular cylinder in a channel with a cylinder of radius of $R = 0.5m$ located in the centre of the horizontal channel with a blockage ratio of $\beta = 0.5$ and a Reynolds number of $Re = 120$.

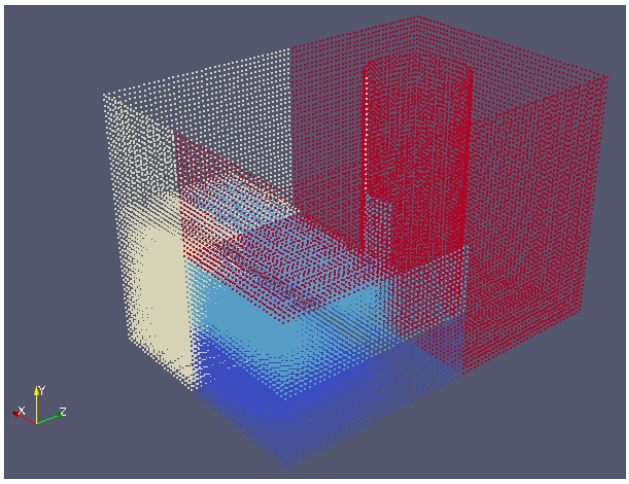


Figure 3: Example 3-D domain decomposition with HSFC with 4 MPI partitions for dam break impacting a cylinder post test case, different colours represent different partitions. There are no need special treatment for rigid cylinder and fluid during domain decomposition and dynamic load balancing

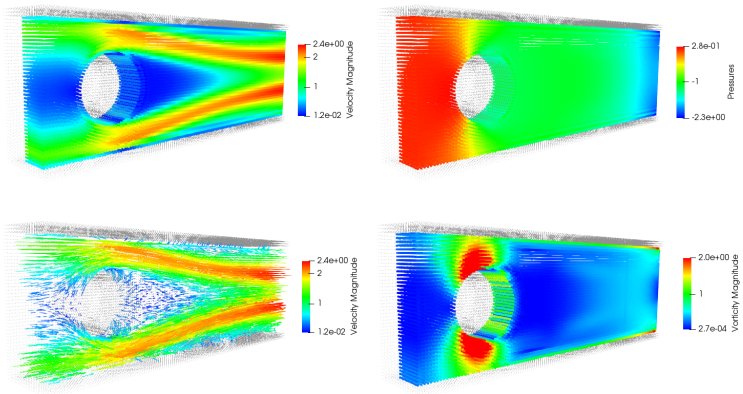


Figure 4: The Snapshot for Flow around cylinder test case which shows the velocity magnitude field, pressure, the velocity vector field, and the vorticity

5.2. Performance analysis

The total number of particles used for the benchmark is up to 100 million. The sparse linear solver is using multigrid preconditioner HYPRE BoomerAMG[10] and Krylov subspace method GMRES. The multigrid preconditioner HYPRE BoomerAMG has two phases, the first phase is setup phase including selection of coarse grids, creation for the interpolation operators, and the representation of the fine grid matrix operator on each coarse grid. The second phase is the solving phase containing matrix-vector multiply and the smoothing operators.

We are using the UK National HPC platform ARCHER, which is Cray XC30 system. ARCHER compute nodes contain two 2.7 GHz, 12-core E5-2697 v2 (Ivy Bridge) series processors. Each of the cores in these processors can support 2 hardware threads (Hyper-threads). Within the node, the two processors are connected by two QuickPath Interconnect (QPI) links. Standard compute nodes on ARCHER have 64 GB of memory shared between the two processors. The memory is arranged in a non-uniform access (NUMA) form: each 12-core processor is a single NUMA region with local memory of 32 GB. Access to the local memory by cores within a NUMA region has a lower latency than accessing memory on the other NUMA region. There are 4544 standard memory nodes (12 groups, 109,056 cores) and 376 high memory nodes (1 group, 9,024 cores) on ARCHER giving a total of 4920 compute nodes (13 groups, 118,080 cores), providing a total of 2.55 Petaflops of theoretical peak performance.

The efficiency is obtained with the following formula:

$$S_p = \frac{T_1}{T_p * p} * 100.00\% \tag{1}$$

where T_1 is the wall time with 1 node, each node comprises 24 Intel E5-2697 cores, T_p is the wall time with p nodes ($p \geq 1$). S_p is the statistic generally used to show the code scalability.

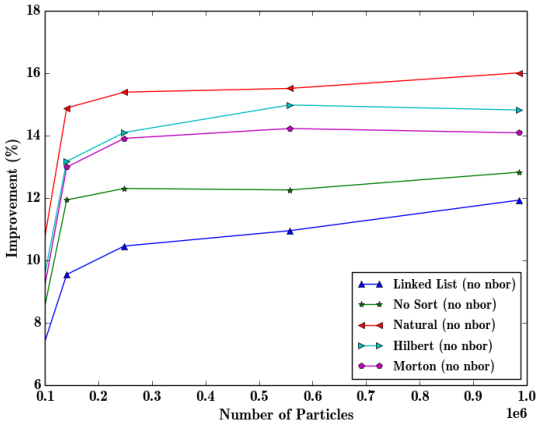


Figure 5: The improvement in time without the solver when reordering owned particles without saving the neighbour list compared with the original linked list in serial

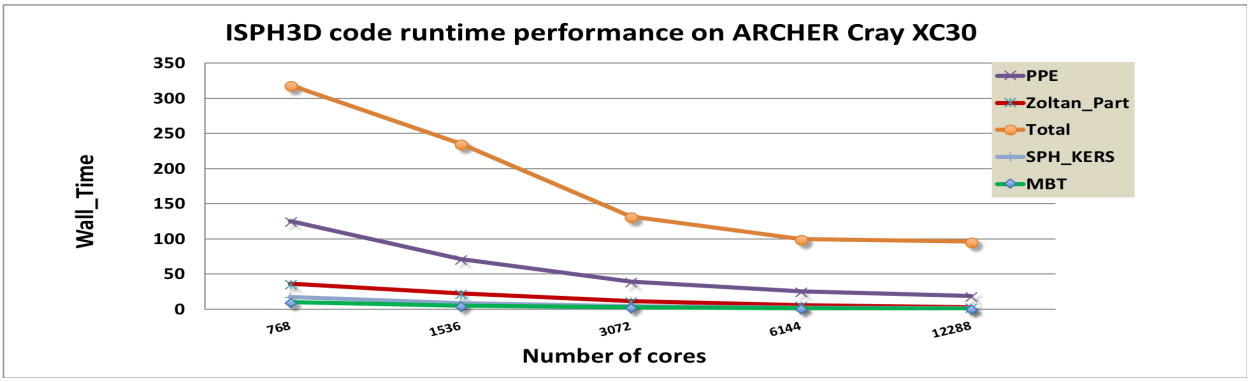
In Section 2, the implementation of the nearest neighbour list searching data structure was described, this results in a new implementation particles ordering. As it is not necessary to save the neighbour list, this has not only reduced the memory footprint, but also results in a better performance improvement for particles ordering. The resulting improvement in performance when not saving the neighbour list calculation for different ordering methods is shown in Fig. 5. Not saving the neighbour list implementation has resulted in improvements up to 16%. This is largely due to the reduction of redundant calculations of smoothing gradients and other floating point operations, and providing a more favourable pattern of data accesses by looping over cells.

Fig. 6 shows the overall performance of the ISPH3D running 10 time steps, Fig. 6a gives the wall time of each components/kernels in the ISPH3D. Fig. 6b gives the efficiency of each components/kernels. The efficiency defined in Equation 1. The PPE gives overall performance of pressure Poisson solver using GMRES and HYPRE BoomerAMG as preconditioner. Zoltan_Part gives the costs of domain decomposition and dynamic load balancing using zoltan.

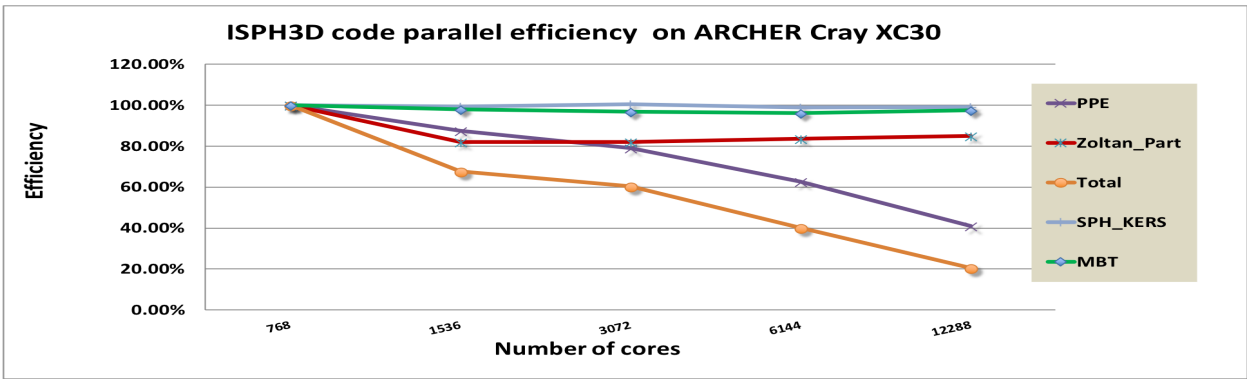
MBT is the cost of newly implemented multiple boundary tangent method. SPH_KERS are those SPH method related kernels.

In the Fig. 6a, we can see the majority of the walltime is spent in solving the pressure Poisson equation which is one of the aims of an efficient parallelisation of ISPH. The MBT and other SPH related kernels account very small percentage compared with pressure solver. The partition and repartition using zoltan(see Zoltan_Part) are also not negligible.

In the Fig. 6b shows the scalability of each kernel. Firstly, we can see the ISPH kernels and MBT are able to scale linearly. With up to 12288 cores, it can still achieve 99.04% and 97.5% efficiency respectively. The MBT's good scalability is due to mirror particles generation is local operation and there are no MPI communications are involved at this stage. Although zoltan partition and repartition costs are not negligible, but their scalibility are also very good. In the PPE solver part, With 12288 cores, The overall pressure Poisson solver can still reach more than 40% efficiency. However, the scalability of those relatively low cost parts, like PCSETUP, MatAssemble becomes worse (only 20% efficiency with 12,000 cores) when using a large number of cores. They now become the bottleneck that needs to be addressed for the complete scalability of ISPH3D. These issues need to be further investigated.



(a) ISPH3D Walltime Analysis



(b) ISPH3D Parallel Efficiency Analysis

Figure 6: Strong scaling results for the whole ISPH3D using up to 512 XC30 nodes (12288 cores).

6. Conclusions and Identification of Future Work

New developments to enable incompressible smoothed particle hydrodynamics (ISPH) for complex geometries for massive simulations have been presented. For the purpose of software scalability and extensibility, a data preconditioning technique has been implemented to redesign the nearest neighbour searching data structure to easily accommodate multiple types of particles (fluid, domain boundary, floating object). The capability to deal with irregular distributed particles in parallel has been demonstrated with HSFC partition method.

A point-in-object kernel has been implemented for mirror particle generation with curved solid boundaries and solved the issue of mirror particles over generation for MBT method. Due to complexity of testing objects with complex geometries, the methods have been tested using a 3-D cylinder. This has allowed the further task to be identified to extend the point-in-polyhedron algorithm for more general complex geometries.

The novel local uniform stencil (LUST) boundary method has been implemented. In 3-D,this methods involves triangulation of solid surface. Due to the triangulation it has been identified that the entire solid object must be shared across partitions which is not efficient in terms of memory use.

Comparing the boundary techniques of MBT and LUST, both have requirements of using point-in- polyhedron algorithm, which needs further development for more general complex geometries. In addition, they both have issues to predict the memory usage prior to simulation which may hinder the usage for larger-scale problems.

Flow around the cylinder has been used to benchmark the code and the dam break with cylinder obstacle test case on ARCHER. Both showed the promising efficiency with 12,000 cores using up to 100 million particles. This remains to be further developed.

Acknowledgement

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

Reference

[1] S. Lind, R. Xu, P. Stansby, and B. Rogers, "Incompressible smoothed particle hydrodynamics for free-surface flows: A generalised diffusion-based algorithm for stability and validations for impulsive flows and propagating waves," *Journal of Computational Physics*, vol. 231, no. 4, pp. 1499 – 1523, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0021999111006279

[2] H. Gotoh and A. Khayyer, "Current achievements and future perspectives for projection-based particle methods with applications in ocean engineering," *Journal of Ocean Engineering and Marine Energy*, vol. 2, no. 3, pp. 251–278, 2016. [Online]. Available: http://dx.doi.org/10.1007/s40722-016-0049-3

[3] S. Yeylaghi, B. Moa, P. Oshkai, B. Buckham, and C. Crawford, "ISPH modelling for hydrodynamic applications using a new MPI-based parallel approach," *Journal of Ocean Engineering and Marine Energy*, vol. 3, no. 1, pp. 35–50, 2017. [Online]. Available: http://dx.doi.org/10.1007/s40722-016-0070-6

[4] A. Khayyer, H. Gotoh, and Y. Shimizu, "Comparative study on accuracy and conservation properties of two particle regularization schemes and proposal of an optimized particle shifting scheme in ISPH context," *Journal of Computational Physics*, vol. 332, pp. 236 – 256, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0021999116306465

[5] J. M. Domnguez, A. J. C. Crespo, M. Gomez-Gesteira, and J. C. Marongiu, "Neighbour lists in smoothed particle hydrodynamics," *International Journal for Numerical Methods in Fluids*, vol. 67, no. 12, pp. 2026–2042, 2011. [Online]. Available: http://dx.doi.org/10.1002/fld.2481

[6] X. Guo, S. Lind, B. D. Rogers, P. K. Stansby, and M. Ashworth, "Efficient massive parallelisation for incompressible smoothed particle hydrodynamics (isph) with 10^8 particles," in *The 8th SPHERIC Workshop*, vol. 1, 2013, pp. 397–402.

[7] A. Skillen, S. Lind, P. K. Stansby, and B. D. Rogers, "Incompressible smoothed particle hydrodynamics (SPH) with reduced temporal noise and generalised fickian smoothing applied to bodywater slam and efficient wavebody interaction," *Computer Methods in Applied Mechanics and Engineering*, vol. 265, pp. 163 – 173, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0045782513001448

[8] M. Yildiz, R. A. Rook, and A. Suleman, "Sph with the multiple boundary tangent method," *International Journal for Numerical Methods in Engineering*, vol. 77, no. 10, pp. 1416–1438, 2009. [Online]. Available: http://dx.doi.org/10.1002/nme.2458

[9] F. G., D. J. M., V. R., N. A., and R. B. D., "Local uniform stencil (lust) boundary conditions for 3-d irregular boundaries in dualsphysics," in *The 9th SPHERIC Workshop*, vol. 1, 2014, pp. 103–110.

[10] A. Baker, M. Schulz, and U. M. Yang, "On the performance of an algebraic multigrid solver on multicore clusters," in *VECPAR 2010, J.M.L.M. Palma et al., eds., vol. 6449 of Lecture Notes in Computer Science*. Springer-Verlag, 2011, pp. 102–115.