,

Developing Dynamic Load Balancing library for wsiFoam

Xiaohu Guo[a], Scott Brown[b], Deborah Greaves[b]

[a] *Hartree Centre,*
Science and Technology Facilities Council,
Daresbury Laboratory, Warrington WA4 4AD UK
[b] *School of Engineering,*
University of Plymouth, Plymouth, PL4 8AA, UK

Abstract

Offshore and coastal engineering fields are using increasingly larger and more complex numerical simulations to model wave-structure interaction (WSI) problems, in order to improve understanding of safety and cost implications. Therefore, an efficient multi-region WSI toolbox, `wsiFoam`, is being developed within an open-source community-serving numerical wave tank facility based on the computational fluid dynamics (CFD) code OpenFOAM®, as part of the Collaborative Computational Project in Wave Structure Interaction (CCP-WSI). However, even using the efficiency of a multi-region approach, the computational expense of CFD is high, and hence there is a constant need to improve the efficiency of these high-fidelity codes. One way that this can be achieved is through the parallelisation of code to make use of high performance computing facilities. Although OpenFOAM® is parallel ready, historically the MPI performance has been considered sub-optimal, but recent developments have led to a number of performance improvements being implemented in OpenFOAM® v5.x along with new parallel I/O functionality. The developments have led to a significant performance benefit when employing a large number of processors, and it is vital that existing code is updated to be compatible with OpenFOAM® v5.x in order to utilise this functionality. However, OpenFOAM® v5.x still only offers static domain decomposition, which limits the choice of parallel load balancing methods to algorithms which can only take basic user defined arguments to aid in load balancing. These methods typically use blocking MPI communications that only consider the number of mesh cells and their spatial distribution (i.e. they assume homogeneity of load per mesh cell). As typical WSI simulations are often inhomogeneous with respect to the mesh, due to properties such as mesh motion and wave 'relaxation', these decomposition methods based purely on mesh resolution are likely to be sub-optimal in terms of computing resource usage.

Therefore, in this work the `wsiFoam` toolbox is updated to utilise the new parallel I/O functionality in OpenFOAM® v5.x. Furthermore, it is coupled with a newly implemented dynamic load balancing method, based on a new `ParMETIS` decomposition class, that considers all of the relevant parameters is required for efficient simulation and optimal parallel performance of WSI problems. Benchmarking of the new functionality shows that the load imbalance is a major performance bottleneck for dynamic mesh applications but substantial improvements in computational efficiency (up to 5 times speed up) and parallel scaling have been observed through use of the new library for WSI applications.

Keywords: Wave Structure Interaction; Dynamic Load Balancing; parMETIS; OpenFOAM®; MPI;

1 Introduction

The interaction of waves and structures is a vital consideration in the design of offshore and coastal structures, since it has implications on both the safety and cost-effectiveness of a concept. The need to improve understanding in wave-structure interaction (WSI) has led to increasingly large and complex numerical simulations being used in offshore and coastal engineering fields, such as the offshore renewable energy (ORE) sector which is considered to be of high national importance for the UK. The physics and scale of a particular problem are usually case-specific, and will lead to an unfeasible computational cost if a high fidelity model is used throughout a large domain. Therefore, a

multi-region numerical tool for assessing the influence of wave-structure interaction, `wsiFoam` [1], is being developed within an open-source community-serving numerical wave tank facility based on the computational fluid dynamics (CFD) code OpenFOAM®, as part of the Collaborative Computational Project in Wave-Structure Interaction (CCP-WSI). The multi-region approach aims to reduce computational cost by splitting a numerical domain into multiple regions, each of which solves a different set of equations, isolating the high-fidelity component purely to regions where it is a required approach (e.g. only using compressible solvers when aeration occurs). However, even with the multi-region approach, CFD is a highly computationally expensive model, and efforts are constantly being made to improve the efficiency.

One way to reduce the computational cost is parallelisation of code to utilise high performance computing resources which are becoming increasingly readily available. OpenFOAM® features a very object oriented design with distributed memory parallelism in mind, and all calls to the message passing interface (MPI) are encapsulated in a high level library called `Pstream`. Historically, OpenFOAM® MPI performance has been considered sub-optimal, but following a review by the Japanese Research Organisation for Information Science and Technology (RIST) [2] [3], a collection of enhancements were implemented based on tests using the `pimpleDyMFoam` solver. These improvements include dropping the master-slave model to exchange/receive buffer size information; changing the gather/scatter order; and reducing memory footprint for communication schedules. The resulting recommendations lead to a significant performance benefit when employing a large number of processors. OpenFOAM® v5.x has already incorporated these performance improvements, and with newly implemented parallel input/output (I/O), `wsiFoam` is now urgently required to be ported to use v5.x. Furthermore, OpenFOAM® currently employs static domain decomposition, which has a limited choice of parallel load balancing methods (simple, hierarchical, scotch, manual). These methods can only take basic user defined arguments to aid in load balancing and typically use blocking MPI communications that only consider the number of mesh cells and their spatial distribution (i.e. they assume homogeneity of load per mesh cell). As typical WSI simulations require additional CPU effort in terms of mesh motion, wave 'relaxation' and turbulence modelling, for example, which are in general inhomogeneous with respect to the mesh, decomposition methods based purely on mesh resolution are likely to be sub-optimal in terms of computing resource usage. Furthermore, for zonal CFD applications even the fundamental equations and solution methods are region specific. Therefore, a dynamic load balancing method that considers all of the relevant parameters is required for efficient simulation and optimal parallel performance of WSI problems.

Therefore, this report presents work conducted on the implementation of a dynamic load balancing library for `wsiFoam`, which automatically balances multiple criterion simultaneously, and is capable of working with multi-phase and multi-physics simulations. To achieve this objective, `wsiFoam` will be updated to be compatible with OpenFOAM® v5.x allowing it to utilise the latest procedures and newly available IO functionality. Secondly, the current domain decomposition method will be modified to include the functionality of `ParMETIS`, an extension of the decomposition method `METIS`, which includes routines that are especially suited for parallel adaptive mesh refinement (AMR) computations and large scale numerical simulations. The algorithms implemented in `ParMETIS` are based on the parallel multilevel k-way graph-partitioning, adaptive repartitioning, and parallel multi-constrained partitioning schemes which have much smaller computing costs compared with `METIS`. Finally, the new dynamic load balancing library will be developed based on existing classes in the OpenFOAM® distribution (`dynamicFvMesh`) which will automatically balance the computing load on each MPI task.

The report is structured such that details of the porting of `wsiFoam` are presented in Section 2; modification of the domain decomposition library to include `ParMETIS` functionality in Section 3; development of the dynamic load balancing library in Section 4; performance bench-marking in Section 5; and the conclusions are drawn in Section 6.

2 WP1: Porting `wsiFoam` to OpenFOAM® v5.x

The focus of this work package was `wsiFoam`, a module developed within the OpenFOAM® environment for simulating WSI problems [1]. The `wsiFoam` solver is based on a multi-region approach, which allows higher fidelity codes to be applied solely in regions where they are necessary in order to increase computational efficiency. For example, if we consider the interaction of a wave with a fixed structure, the majority of the domain can be resolved using an incompressible solver, but in the near vicinity of the structure air may be entrained, and this could only be captured accurately using a compressible model. Presently, the `wsiFoam` solver considered here only considers two

types of region (incompressible or compressible), but due to the modularity of the code it will be developed to include cheaper models, such as fully non-linear potential approaches, to further improve efficiency in the future.

In this work package, `wsiFoam` has been updated to run in OpenFOAM® v5.x [4], which was the latest released version at the start of the project. The `wsiFoam` package was originally developed [1] in Foundation version v2.3.1 [5], and comprised of a multi-region solver largely formed from the existing two-phase solvers `interDyMFoam` (for incompressible regions) and `compressibleInterDyMFoam` (for compressible regions); libraries describing the coupled boundary conditions used at shared boundaries between regions; and additional libraries for turbulence modelling and six degree of freedom (6DoF) rigid body motion.

The solver and boundary condition libraries were updated to use the new syntax and procedures available in OpenFOAM® v5.x. However, the turbulence model libraries were removed, and instead the `wsiFoam` code was coupled with the existing libraries for this functionality in the OpenFOAM® v5.x distribution. As well as utilising the additional functionality and range of options available in the latest OpenFOAM® distribution, this has the added advantage that the code should be simpler to update in the future as new versions are released. Similarly, the 6DoF library was changed to utilise CCP-WSI's modified library (including additional functionality and development) which is available and regularly updated in the CCP-WSI repository. Additionally, the existing wave generation and absorption methods that were available in the original 2.3.1 version of `wsiFoam` have been replaced through coupling with the additional toolbox `waves2Foam` [6]. This toolbox adds a much wider range of expression-based wave theories to choose from, including the option to generate focused waves using superposition of linear components and has previously been successfully applied in a range of WSI problems such as wave impacts with fixed structures [7, 8]; wave energy convertors [9, 10]; floating tidal platforms [11]; and waves breaking on a beach [12]. Furthermore, `waves2Foam` provides the relaxation zone functionality which absorbs waves (and improves the inlet signal) by blending analytical and simulated values using a user-specified weighting function [6].

2.1 Verification

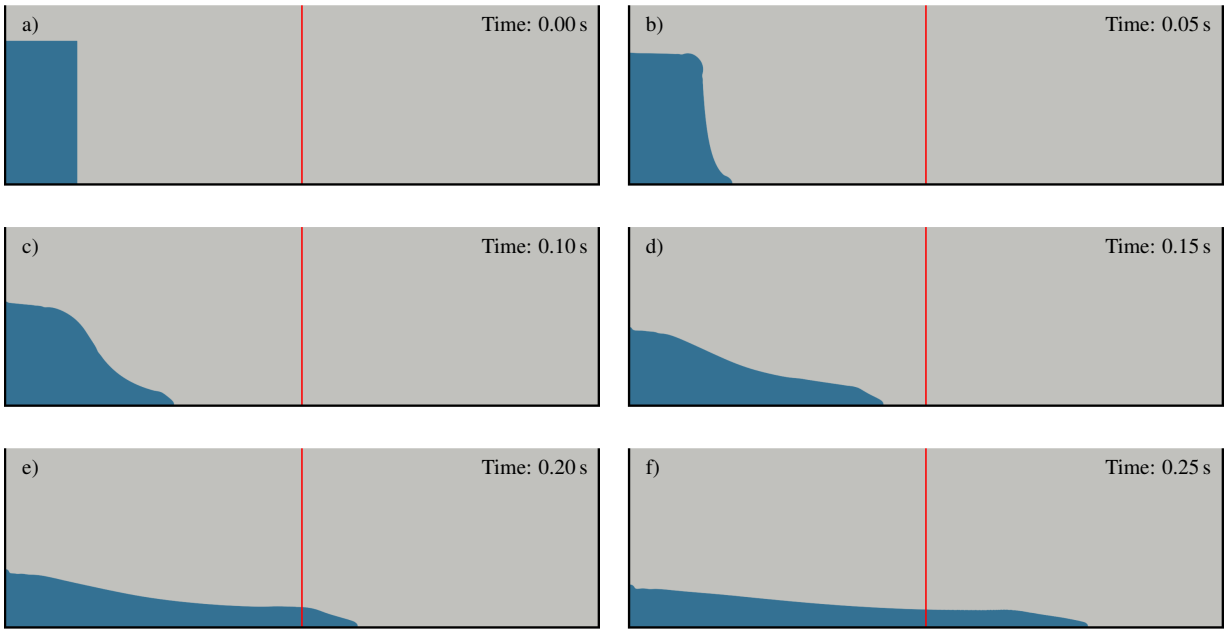

Figure 1: Snapshots of a simulated dam break crossing a shared boundary (red line) between an incompressible and compressible region using the finer mesh discretisation.

To test the newly ported version of `wsiFoam`, a simple quasi 2D test case is used to verify that the free surface progresses smoothly across the shared boundary between regions. The test case that will be considered in this report is a dam break simulation previously used as verification by Martínez Ferrer *et al.* (2016) when developing the

original version of `wsiFoam` [1]. As well as being a popular benchmarks for numerical models [13, 14, 1] this case will verify that both the compressible and incompressible regions have been implemented correctly. The domain is designed in the $x-z$ plane with dimensions 0.5 m $\times$ 0.15 m, uniformly discretised with an aspect ratio of 1. Two mesh discretisations have been considered: a relatively coarse mesh ($\Delta x = 2.5$ mm) as was used by Martínez Ferrer *et al.* (2016) [1]; and a finer mesh ($\Delta x = 0.25$ mm) to verify that the methodology is robust on fine mesh discretisations. The domain is split into two regions with a shared coupled boundary at $x = 0.125$ m: the left side is modelled as an incompressible two-phase fluid; the right side is modelled as a compressible two-phase fluid. The side and bottom boundaries are considered as walls with no slip conditions applied, and the top boundary is an open atmosphere boundary where the pressure is 1 bar. The water phase is initialised as a block of width $a = 0.06$ m and height $h = 0.12$ m in the incompressible region and is released at time $t = 0$ s (see Figure 1a).

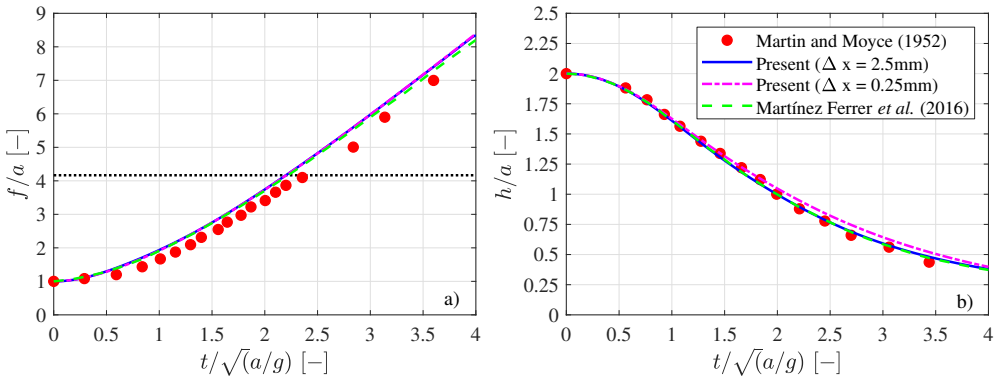

Figure 2: Normalised horizontal position of the dam break front (a) and height at the left wall (b) as a function of normalised time. The dotted black line indicates the shared boundary between regions.

Figure 1 presents snapshots of the simulated water phase at times $t = 0.00$ s (a), $t = 0.05$ s (b), $t = 0.10\,s$ (c), $t = 0.15$ s (d), $t = 0.20$ s (e), $t = 0.25\,s$ (f). The shared boundary between the incompressible and compressible regions are indicated by the red vertical line in each plot. The snapshots indicate that, qualitatively, the free surface passes smoothly between regions, and in additional analysis it was confirmed that both the velocity and pressure field also have a smooth distribution across the shared boundary. Furthermore, Figure 2 shows quantitative analysis of the position of the water front along the bottom of the tank, f, (a) and the height of the water column along the left boundary of the tank, h (b). The solution from both of the present grids are compared with Martínez Ferrer *et al.*'s original `wsiFoam` solution [1] along with experimental data [15]. The results imply that the `wsiFoam` code has been successfully ported to the latest version since the coarser grid produces almost identical solutions as Martínez Ferrer *et al.* for both f and h. However it should be noted that there is a slight discrepancy in the water front position (Fig 2a) in the compressible region which is likely due to updated procedures implemented since the release of OpenFOAM® v2.3.1. The fine mesh produces a solution for the water front that is also smooth over the shared boundary between regions, indicating a robust coupling scheme. Overall, based on the information in Figures 1 and 2 the code is considered to have been implemented correctly: further tests were also conducted successfully to verify that the coupling with `waves2Foam` was working as expected, and that waves propagated through the shared boundary.

3 WP2: New domain decomposition class parmetisDecomp implementation with ParMETIS

In this work package, OpenFOAM®'s domain decomposition methodology is updated to include the functionality of the MPI-based parallel library `ParMETIS`. OpenFOAM® v5.x presently only offers static domain decomposition techniques, and this consequently limits the choice of parallel load balancing methods to algorithms which take basic user defined arguments to aid in load balancing (`simple`, `hierarchical`, `scotch`, `METIS`, `manual`)(see Fig 3). Furthermore, these methods typically use blocking MPI communications that only consider the number of mesh cells and their spatial distribution (i.e. they assume homogeneity of load per mesh cell). For WSI applications, a decomposition method based purely on the number of mesh resolution is considered to be sub-optimal in terms of

computing resource usage since there are multiple sources of CPU effort which are generally inhomogeneous such as mesh motion, wave 'relaxation' and turbulence modelling. Furthermore, the design of `wsiFoam` means that even the fundamental equations and solution methods are region specific. Therefore, a dynamic load balancing method that considers all of the relevant parameters is required for efficient simulation and optimal parallel performance of a typical WSI problem.

Based on the above discussion, the present domain decomposition method has been modified to include `ParMETIS`, which extends the functionality provided by `METIS` and includes routines that are especially well suited for parallel AMR computations and large scale numerical simulations. The algorithms implemented in `ParMETIS` are based on the parallel multilevel k-way graph-partitioning, adaptive repartitioning, and parallel multi-constrained partitioning schemes which have much smaller computing costs compared with `METIS`.

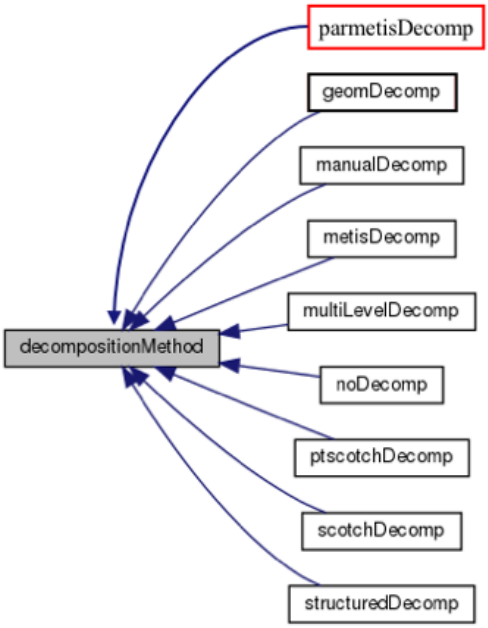

Figure 3: Implementation of the derived `parmetisDecomp` decomposition class relative to existing code.

In OpenFOAM®, the domain decomposition class is structured in an object orientated design, providing flexibility to introduce new methodologies/algorithms without needing to rewrite existing fundamental code. This is achieved using a runtime selection mechanism known as the "virtual constructor" in C++ which provides the ability to clone an object without knowing the object type. Though C++ does not support this as standard, this greatly simplifies the addition of derived classes, particularly a solver with generic boundary conditions. Presently, a selection of derived classes (e.g. `METIS` or `Scotch`) are available in OpenFOAM®, which contain their own routines/algorithms for domain decomposition, and these inherit generic member functions from the base class `decompositionMethod`. Therefore, the solver code is written using these generic member functions, and the details of the routines/algorithms can be inter-changed depending on the preference of the user by selecting the relevant derived class (Figure 3) at runtime. In this work, a new derived class, `parmetisDecomp`, has been created which uses the `ParMETIS` routines/algorithms. Since `PARMETIS` is restricted such that all of its routines must be called by at least two MPI partitions, the `parmetisDecomp` class verifies the number of MPI partitions being used, and if there is only one then the partition method is automatically switched to use `METIS`.

4 WP3: Developing dynamic load balancing mechanism for OpenFOAM®

In OpenFOAM®, mesh motion and topology changes are handled by the `dynamicFvMesh` base class, and the user can select the relevant dynamic mesh derived class through the `dynamicMeshDict` located in the `constant` directory.

Therefore, it is natural to use `dynamicFvMesh` as the base class for the implemention of a dynamic load balancing mechanism in OpenFOAM® (Figure 4).

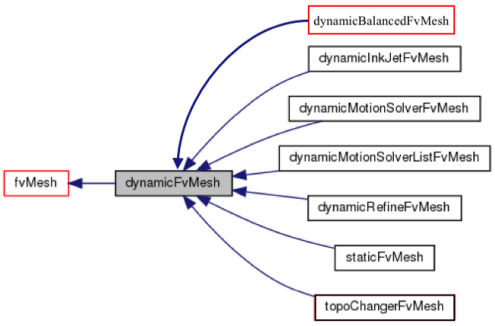

Figure 4: Dynamic Load Balancing Class Implementation based on dynamicFvMesh

In this work package, load imbalance issues caused by mesh refinement/unrefinement, which uses the derived class `dynamicRefineFvMesh`, are addressed. There are two stages for the coupled dynamic load balancing and mesh refinement process. The first stage is the mesh refinement: once the mesh topology is changed due to mesh refinement/unrefinement, the general `mesh.update` function defined by the `dynamicRefineFvMesh` class is called. The second stage is the dynamic load balancing process: once the mesh has been updated, the algorithm described in Algorithm 1 is used to determine if dynamic load balancing is required and update mesh decomposition if necessary. The maximum imbalance value, *maxImb*, defined as

$$maxImb = max\left(\frac{locNumCells - IdealLocNumCells}{IdealLocNumCells} \times 100\%\right), \tag{1}$$

where *locNumCells* and *IdealLocNumCells* represent the number of cells in each partition and ideal number of cells in each partition, respectively, is used as a requirement indicator for load balancing based on an acceptable imbalance tolerance *imbTol*, generally set as 10.00%. Once the maximum imbalance exceeds the imbalance tolerance ($MaxImb > imbTol$), the coarse level 0 dual graph will be created and used for redecomposition. Each graph node represents a cell of the coarse level 0 mesh, and has been assigned a weight based on the weighting function:

$$nodeWeight = nodeWeight + numLeaves. \tag{2}$$

The `ParMETIS` decomposition class is then used to decompose the coarse level 0 mesh based on the assigned weight of each node. Consequently, the resulting partition will have roughly equal weights, normalising the load between processors to reduce imbalance.

Algorithm 1: Dynamic Load Balance Algorithm for DynamicRefineFvMesh

Step 1: Update refinement fields within each MPI partition.
Step 2: Check present level of imbalance with equation (1)
if $maxImb > imbTol$ **then**
  construct the level 0 graph nodes
  calculate each nodes weight with equation (2)
  call repartition with `ParMETIS`
  redistribute the mesh and variable fields
  correct boundary conditions for all fields
end if

5 WP4: Results and Performance Analysis

5.1 Test case

The dynamic load balancing library is verified using an existing test case in the OpenFOAM® distribution for the `interDyMFoam` solver modified to use the `wsiFoam` with a single incompressible region. The simulation is a 3D dam break within a box-shaped domain ($1 \times 1 \times 1$ m), with an obstacle (a cuboid with dimensions $0.25 \times 0.25 \times 0.25$ m) at the centre of the domain. The top of the domain is considered to be an atmospheric boundary condition ($p = 1$ bar), and the remaining sides and the obstacle are considered as solid walls with no-slip conditions applied. The initial dam break is a block of water ($0.6 \times 0.1875 \times 0.75$ m) located in one corner of the domain (see Figure 5a), that is released at time $t = 0$. The initial mesh is uniformly discretised with an aspect ratio of 1 (cubic cells), and automatic mesh refinement is applied based on the values of two fields: the free surface (refined up to octree level 2); and the non-hydrostatic pressure (refinement up to octree level 3). This ensured that the mesh is only refined when it is required, both temporally and spatially, and consequently the total number of cells is constantly changing.

This test case is particularly useful for verification the new dynamic load balancing library for two primary reasons: firstly, the load can be balanced using the total number of cells as an indicator for the need to redistribute load; and secondly, in OpenFOAM® parallel simulations, the domain is typically decomposed at time $t = 0$ and attempts to balance the total number of cells for each processor. Without dynamic load balancing this decomposition is not updated at runtime, potentially leading to a substantial imbalance in CPU effort if, for instance, most of the automatic mesh refinement happened to occurs in a single CPU domain the consequence would be that the remaining processors would waste a lot of time waiting. Figure 5 shows snapshots of the volume of fluid isosurface ($\alpha_1 = 0.5$) for the dam break with obstacle case run on eight processors with dynamic load balancing enabled at $t = 0$ s (a), $t = 0.02$ s (b), $t = 0.12$ s (c), $t = 0.26$ s (d), $t = 0.36$ s (e), $t = 1.1$ s (f). The coloured regions indicate the processor that is solving each part of the domain, and the grey lines show the mesh discretisation at each time. Comparing the snapshots it is clear that the processors are constantly being reallocated based on the number of cells in each region. If we consider the snapshots at times $t = 0.0$ s and $t = 0.02$ s (Figures 5a and b), it seems that the mesh has been rapidly refined around the free surface between the two frames, and this has led to more processors being allocated to this section of the domain (at $t = 0$ s only 4 processors can be seen, whereas all 8 are visible at $t = 0.02$ s). Based on these snapshots, it is concluded that the dynamic load balancing library has been implemented correctly into OpenFOAM® v5.x, and that it works well when number of cells is used as the indicator function, i.e. cases where automatic mesh refinement is used.

5.2 Performance analysis

Benchmark tests are performed with `wsiFoam` for two initial mesh resolutions, $\Delta x = 7.8$ mm and $\Delta x = 3.9$ mm, leading to an approximate simulation size of 2 and 17 million cells, respectively. Each case has been run for varying numbers of nodes (each with 24 cores/MPI partitions) on the ARCHER HPC facility, up to a total of 64 nodes (1536 cores/MPI partitions). Figure 6 shows the wall time measurement for the test case dam break with and without using dynamic load balancing for the 2M cell case on 24 cores (a) and the 17M cell case on 768 cores (b). The results show that using dynamic load balancing leads around 4 times speed up for 2M case and 5 times speedup for 17M case compared to when it is not used.

Figures 7a and 7b show the maximum imbalance (equation 4) and cost of the required `mesh.update` function when using dynamic load balancing for the 17M cell case. The maximum imbalance (Figure 7a) is very high at the beginning of the simulation and then is kept at an acceptably low imbalance throughout the remaining simulation by the dynamic load balancing process. Furthermore, when there is a high maximum imbalance, the cost of the mesh update function (Figures 7b) is also very expensive. As the number of MPI partitions/cores increases, the dynamic load balancing is required more frequently but the cost of the `mesh.update` function decreases. This is thought to be due to communication volume decreasing as the number of MPI partitions increases.

Figure 8 shows that `wsiFoam` scales strongly when using dynamic load balancing library with 17M mesh cells. Compared to the 16 nodes (384 cores) simulation, the efficiency of 32 nodes (768 cores) and 64 node simulations are about 80.9% and 53.1%, respectively.

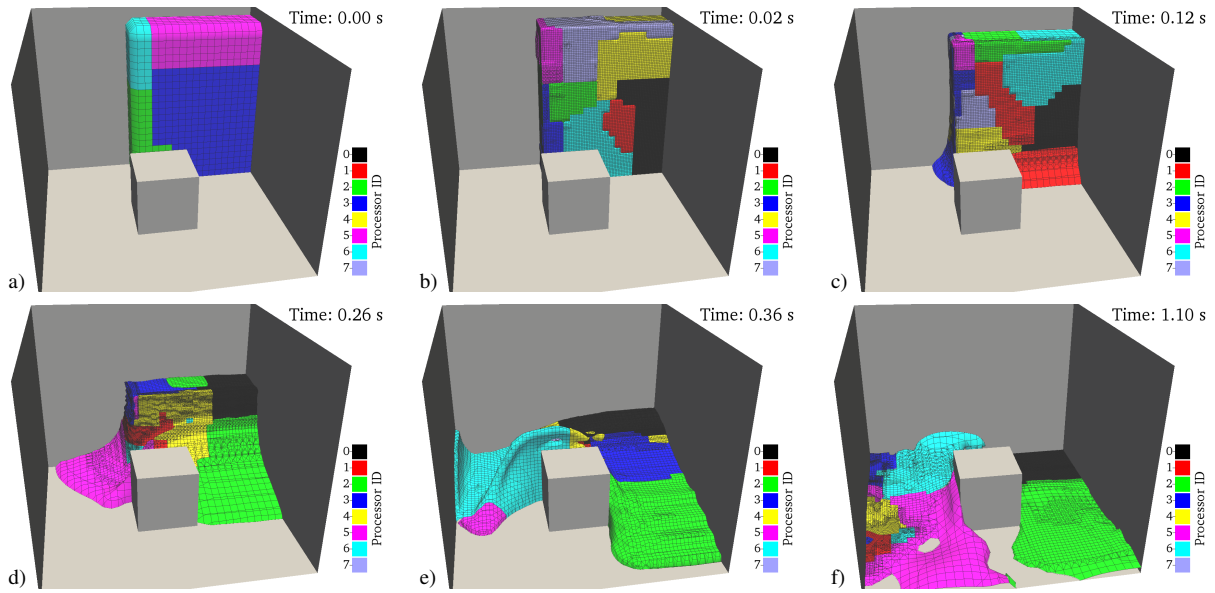

Figure 5: Snapshots of the volume of fluid isosurface ($\alpha_1 = 0.5$) from the dam break with obstacle case using eight processors, adaptive mesh and dynamic load balancing. The colours represent the sections being solved by each processor, and the grey lines show the mesh discretisation.

6 Conclusions and Identification of Future Work

The whole `wsiFoam` software framework has been successfully ported to OpenFOAM® v5.x and has been verified using relevant multi-region benchmarks for incompressible and compressible flow. The initial benchmark has showed the advantage of using collated parallel I/O format which has substantially reduced the number of files when using large number of MPI partitions. However, it must be used together with thread-enabled MPI in order to get the performance advantage.

A new dynamic load balancing library has been developed using the number of mesh cells per MPI partition as the weight function to balance the load. This approach lends itself to simulations where the number of mesh cells is constantly changing, and is therefore highly suited to adaptive mesh refinement cases, which deploy refinement/unrefinement techniques. A test case using this adpative mesh functionality was used to benchmark the new library and shows a substantial increase in both the wall time (4-5 times speed up) and the scalibility (greater than one thousand cores with more than 50% efficiency) of the `wsiFoam` solver. However, restricting use of the dynamic load balancing library to cases with adaptive mesh refinement is non-generic and rules out a large number of current WSI problems. Typically, WSI simulations of floating objects in OpenFOAM® use mesh deformation to track the motion of the structure [9, 10, 16]. Since, as standard, OpenFOAM® is only capable of utilising one form of dynamic mesh at a time, these cases are incompatible with adaptive mesh refinement, although recently progress has been made in this area [17]. To utilise dynamic load balancing in generic WSI simulations it is imperative that the weighting function is developed to work with cases with fixed numbers of mesh cells in the future, based on a timing-based technique.

Furthermore, there are additional limitations which must be addressed in the future in order to make dynamic load balancing a robust and practical tool. Presently there are issues when mesh unrefinement is required along with the dynamic load balancing library. Further performance improvement can also be achieved by removing unnecessary I/O (reading dictionaries from files) particularly related to the `refinementHistory` class implementation. Once the generic functionality and these issues are addressed, the dynamic load balancing approach should be thoroughly benchmarked against existing numerical simulations, such as those proposed as part of the CCP-WSI Blind Test Series [18].

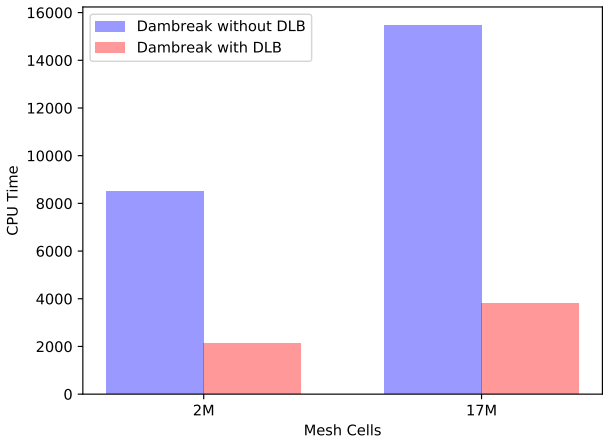

Figure 6: Performance Comparision between Dynamic load balancing and without dynamic load balancing for Dam Break with Obstacle test case

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

References

[1] P. Martínez Ferrer, D. Causon, L. Qian, C. Mingham, and Z. Ma, "A multi-region coupling scheme for compressible and incompressible flow solvers for two-phase flow in a numerical wave tank." *Computers and Fluids*, vol. 125, pp. 116–129, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S004579301500376X

[2] Openfoam parallel improvement. [Online]. Available: http://www.openfoam.com/releases/openfoam-v1606+/parallel.php#parallel-improvements

[3] Openfoam 4.x release. [Online]. Available: https://openfoam.org/release/4-0/

[4] H. Weller, C. Greenshields, W. Bainbridge, M. Janssens, and B. Santos, "OpenFOAM 5.0," 2017. [Online]. Available: https://openfoam.org/release/5-0/

[5] H. Weller, C. Greenshields, M. Janssens, A. Heather, W. Bainbridge, and S. Ferraris, "OpenFOAM 2.3.1," 2014. [Online]. Available: https://openfoam.org/release/2-3-1/

[6] N. G. Jacobsen, D. R. Fuhrman, and J. Fredsøe, "A wave generation toolbox for the open-source CFD library: OpenFOAM®," *International Journal for Numerical Methods in Fluids*, vol. 70, pp. 1073–1088, 2012. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1002/fld.2726

[7] Z. Hu, D. Greaves, and A. Raby, "Numerical wave tank study of extreme waves and wave-structure interaction using OpenFOAM®," *Ocean Engineering*, vol. 126, pp. 329–342, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0029801816303936

[8] S. Brown, P.-H. Musiedlak, E. Ransley, and D. Greaves, "Numerical simulation of focused wave interactions with a fixed FPSO using OpenFOAM 4.1," in *Proceedings of the 28th International Ocean and Polar Engineering Conference*, Sapporo, Japan, 2018, pp. 1–8. [Online]. Available: https://www.onepetro.org/conference-paper/ISOPE-I-18-012

[9] E. Ransley, D. Greaves, A. Raby, D. Simmonds, M. Jakobsen, and M. Kramer, "RANS-VOF modelling of the wavestar point absorber," *Renewable Energy*, vol. 109, pp. 49–65, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960148117301659?via%3Dihub

[10] E. Ransley, D. Greaves, A. Raby, D. Simmonds, and M. Hann, "Survivability of wave energy converters using CFD," *Renewable Energy*, vol. 109, pp. 235–247, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960148117301799?via%3Dihub

[11] E. Ransley, S. Brown, D. Greaves, S. Hindley, P. Weston, E. Guerrini, and R. Starzmann, "Coupled RANS-VOF modelling of floating tidal stream concepts." in *Proceedings of the 4th Marine Energy Technology Symposium (METS)*, Washington, D.C., USA, 2016, p. 5.

[12] S. Brown, D. Greaves, V. Magar, and D. Conley, "Evaluation of turbulence closure models under spilling and plunging breakers in the surf zone," *Coast. Eng.*, vol. 114, pp. 177–193, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378383916300400

[13] A. Murrone and H. Guillard, "A five equation reduced model for compressible two phase flow problems," *Journal of Computational Physics*, vol. 202, pp. 664–694, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999104003018

[14] Z. Ma, D. Causon, L. Qian, C. Mingham, H. Gu, and P. Martínez Ferrer, "A compressible multiphase flow model for violent aerated wave impact problems," *Proceedings of the Royal Society A*, vol. 470, p. 25, 2014. [Online]. Available: http://rspa.royalsocietypublishing.org/content/470/2172/20140542.short

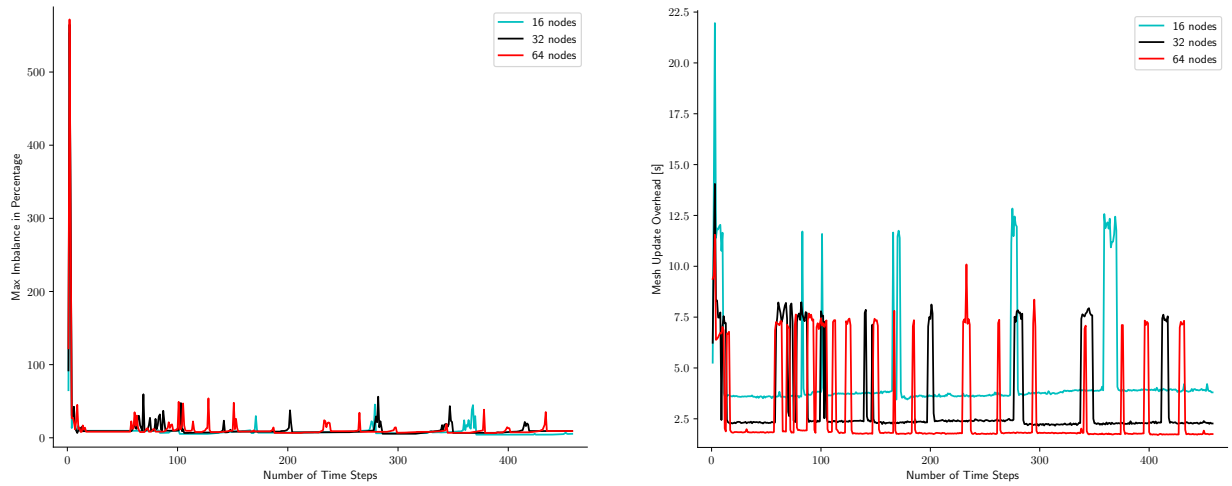

Figure 7: The maximum imbalance (a) and mesh update cost (b) as a function of time step number. These results were obtained for the 17M cell 'dam break with obstale' test case using dynamic load balancing.

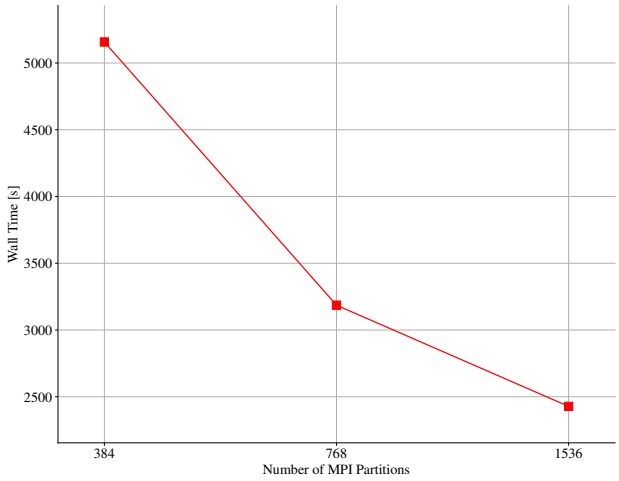

Figure 8: Scaling of the 'dam break with obstacle' test case using `wsiFoam` solver with 17M mesh cells

[15] J. Martin and W. Moyce, "Part IV. An experimental study of the collapse of liquid columns on a rigid horizonal plane." *Proceedings of the Royal Society A*, vol. 244, pp. 312–324, 1952. [Online]. Available: http://rsta.royalsocietypublishing.org/content/244/882/312

[16] C. Windt, J. Ringwood, J. Davidson, E. Ransley, M. Jakobsen, and M. Kramer, "Validation of a CFD-based numerical wave tank of the Wavestar WEC," in *Advances in Renewable Energies Offshore*, G. Soares, Ed. Taylor and Francis Group, 2018, pp. 439–446.

[17] C. Eskilsson and J. Palm, "Simulations of floating wave energy devices using adaptive mesh refinement," in *Advances in Renewable Energies Offshore*, G. Soares, Ed. Taylor and Francis Group, 2018, pp. 431–438.

[18] CCP-WSI Working Group, "CCP-WSI Blind Test Series'," 2018. [Online]. Available: https://www.ccp-wsi.ac.uk/blind_test_workshops