6th European Conference on Computational Mechanics (ECCM 6)
7th European Conference on Computational Fluid Dynamics (ECFD 7)
1115 June 2018, Glasgow, UK

# PROFILING AND APPLICATION OF THE MULTI-SCALE UNIVERSAL INTERFACE (MUI)

**A. Skillen[1], S.M. Longshaw[1], G. Cartland-Glover[1] , C. Moulinec[1], D.R. Emerson[1]**

[1] Scientific Computing Department,
The Science and Technology Facilities Council,
UK Research and Innovation,
Daresbury Laboratory,
Warrington,
WA4 4AD, UK
alex.skillen@stfc.ac.uk

**Key words:** Code Coupling, HPC, neutronics-CFD coupling

**Abstract.** The aim of this paper is to test the scalability of the multi-scale universal interface (MUI) code coupling library. This has been done by coupling two independent synthetic solvers to one another, which are then given an embarrassingly parallel workload. The quantity of work done by each synthetic solver, and the amount of data transferred, is representative of a typical multi-physics coupled problem. Both volumetric coupling (where whole fields are transferred), and interface coupling (where several variables at the two-dimensional interface between three-dimensional meshes) are tested. It is shown that the multi-scale universal interface scales well to a minimum of $\mathcal{O}(10^4)$ MPI tasks.

We also demonstrate the use of the MUI library in coupling a neutronics code (DYn3D) to a CFD code (Code_Saturne). We demonstrate this coupling by simulating a simplified molten salt fast nuclear reactor.

## 1 INTRODUCTION

As computer power advances, so too does our capacity for high fidelity simulation. An ultimate goal for high fidelity simulation – one that is gaining traction within both industry and academia – is digital twinning. A digital twin is a mapping of a physical product or process to a digital representation. Sufficiently high fidelity in the simulation is necessary to ensure the twin is representative of reality. Digital twinning promises to revolutionise industry by enabling unprecedented insight into the whole life-cycle of the product or process it represents, without the need for expensive (or potentially infeasible) experiments. Similarly, in science, a digital twin can be used to test the predicted outcomes of scientific theories under exotic conditions, to ensure observations in the real world match.

A key feature of a digital twin is the incorporation of multiple physical models within a single simulation (multi-physics simulation). Similarly, different components of the model

may describe effects at vastly different scales (multi-scale simulation). When simulating multi-physics or multi-scale problems in a High Performance Computing (HPC) environment, care must be taken to ensure the scalability of the method. By adopting an operator-splitting strategy, it is possible to decompose the complex multi-physics/scale system into a series of interlinked sub-problems, which can then be tackled individually. This has advantages over a monolithic coupling (where all governing equations are solved simultaneously) since legacy codes, employing specialised numerical algorithms, can be used with minimal effort, thereby preserving the many decades of effort that has gone into developing robust, validated scientific software.

The challenge with this partitioned approach to coupling is the communication of data from one code to the other in a way that does not introduce bottlenecks or significant sources of error. There are a number of code coupling libraries available. Some libraries provide a framework which controls the time-stepping and launch of individual solvers [1, 2, 3, 4]. These framework approaches to coupling often require background daemon processes to be running (to orchestrate the coupled simulation), as well as TCP ports to be opened in order to allow the transfer of data. This can have implications on the number of systems in which the coupler can run.

An alternative approach is to provide an MPI based communication interface, which acts as a layer in which each application can deposit or retrieve data. This has the advantage of being *universal*. That is, any code can be coupled to any other code, with minimal effort, rather than a bespoke solution of coupling A to B. Additionally, the interface approach should be expected to run on any HPC system, without the need for installing daemon processes. For a review of current code coupling projects, the reader is referred to the review chapter of Longshaw et al. [5].

In the sections that follow, we provide a brief overview of the 'multiscale universal interface' (MUI) [6], which is an interface coupling library developed originally at Brown University, but now is part of a collaboration between Brown University, IBM Research, Lawrence Berkeley National Laboratory, and STFC. We test the scalability of the library itself in transferring data and interpolating that data back to another solver. We will also briefly demonstrate the application of the MUI in coupling a neutronics code to CFD.

## 2 The Multiscale Universal Interface (MUI)

The MUI[1] [6] is a header only library written in C$^{++}$. Wrappers are also available for coupling codes written in other languages. The basic premise of the MUI is to provide in interface which codes can *push* point data to. MUI works on point data in order to maintain the universal nature of the coupling; all domain specific representations of the data (for example on meshes) can be represented as a cloud of points. In this way, it is not necessary to have detailed knowledge of the underlying data structures describing mesh connectivity, for instance. The price to pay for this flexibility is increased cost in the sampling (interpolation), for example using radial basis functions to recover a field

[1]The library is open source, distributed under the Apache license, and can be downloaded at https://github.com/MxUI/MUI

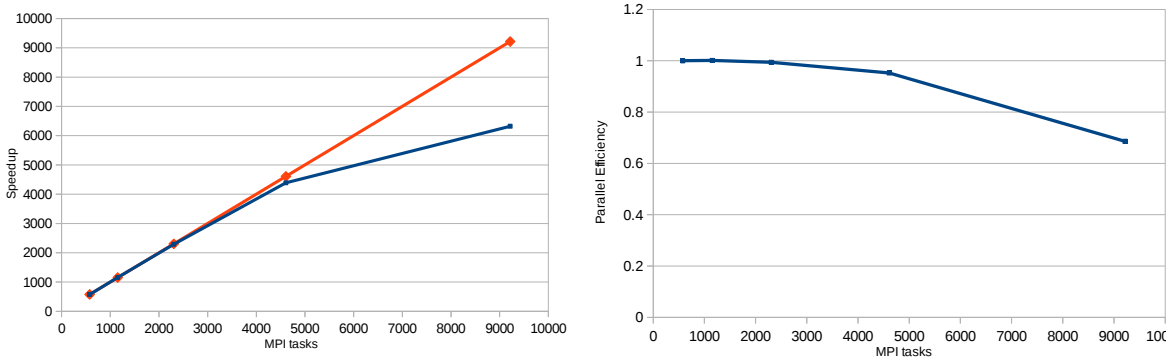


**Figure 1**: Volumetric coupling with MUI. 200M data points transferred at each time-step. Speedup (left) with ideal speedup in red. Parallel efficiency (right).

from a cloud of points.

Once these data points are all pushed, it is necessary to *commit* the data with a time-stamp, which initiates the data transfer through MPI. Other solvers connected to the interface can then *fetch* data at an arbitrary point in space and time (using interpolation where the data does not exist). The API of MUI is very simple. Indeed, it is possible to couple two solvers to one another by adding just a few lines of code to *push*, *commit* and *fetch*.

A key strength of MUI is it is extensible. By taking advantage of C$^{++}$ templates, the library makes it simple to implement additional samplers to perform the interpolation. In addition, MUI's underlying communication layer has been abstracted such that different communication protocols can be implemented with ease. It is possible to communicate via TCP across the internet, or via standard I/O using files, or even through POSIX shared memory directly. But more typically, the MPI Multiple Program Multiple Data (MPMD) mode is used. This allows an arbitrary number of codes to communicate with one another over MPI, and hence can be used on any HPC system with a working MPI implementation that supports the MPMD paradigm.

## 3 Scalability testing

In order to test the scalability of the MUI library in isolation to that of any scientific solver, a synthetic solver was written. This synthetic solver is given an embarrassingly parallel workload, which is fixed, in terms of CPU hours, for a given problem size. Checks were made to ensure this synthetic solver scales linearly to a minimum of $\sim 10,000+$ MPI tasks (in practice, it would be expected to scale to any arbitrary number of cores). The workload was designed to be typical of MPI parallelised finite volume Computational Fluid Dynamics solvers, such as Code_Saturne or OpenFOAM. The workload size was set to correspond broadly with our previous experience of Code_Saturne. A value of $1 \times 10^{-4}$ CPU seconds / timestep / cell was used (i.e. the equivalent of 1 second per timestep at 10M cells and $1,000$ MPI tasks). Two instances of this synthetic solver were coupled to one another.

Cases were run on the UK national supercomputing service ARCHER, which is a Cray

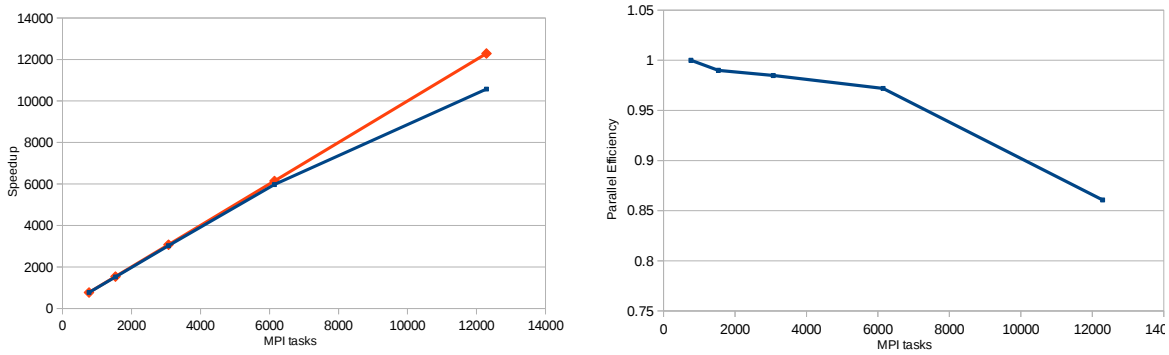


**Figure 2**: Interface coupling with MUI. 3M data points transferred at each time-step. Speedup (left) with ideal speedup in red. Parallel efficiency (right).

XC30 supercomputer with 12-core Intel Ivy Bridge series processors.

### 3.1 Volumetric coupling

Figure 1 shows the parallel efficiency and speedup for a volumetric coupled case (typical of neutronics-CFD coupling, amongst others) with 200M data points being transferred (100M points in each solver). It can be seen the performance is broadly acceptable. To understand the drop in efficiency at the highest core-counts considered, profiling of the codes with CrayPat has been performed. Table 1 shows the bottleneck lies within MPI itself; this is the unavoidable cost of communication. The cost of the MUI sampler (i.e. interpolation) is fairly small (∼8%), even when processing a relatively large number of data points.

**Table 1**: Profiling of MUI for the volumetric coupled case at maximum core-count.

| Function | % samples |
|---|---|
| MPI_Test | 23.3 |
| MPI_Recv | 4.0 |
| Synthetic Solver work | 62.5 |
| MUI sampler | 7.6 |

### 3.2 Surface coupling

Figure 2 shows the coupling through a surface (typical of conjugate heat transfer, fluid structure interaction, etc.). The mesh size is again 100M for each solver, but in this case, the number of data points transferred is 3M (1.5M per solver). This is representative of a surface coupling in which several variables are transmitted. It can be seen that, in this case, the performance is better. The parallel efficiency is quite acceptable (above 85%) up to $12,288$ MPI tasks. Table 2 shows the breakdown of time by function for the surface coupled case. It can be seen the overhead of MUI interpolation is low (4%).

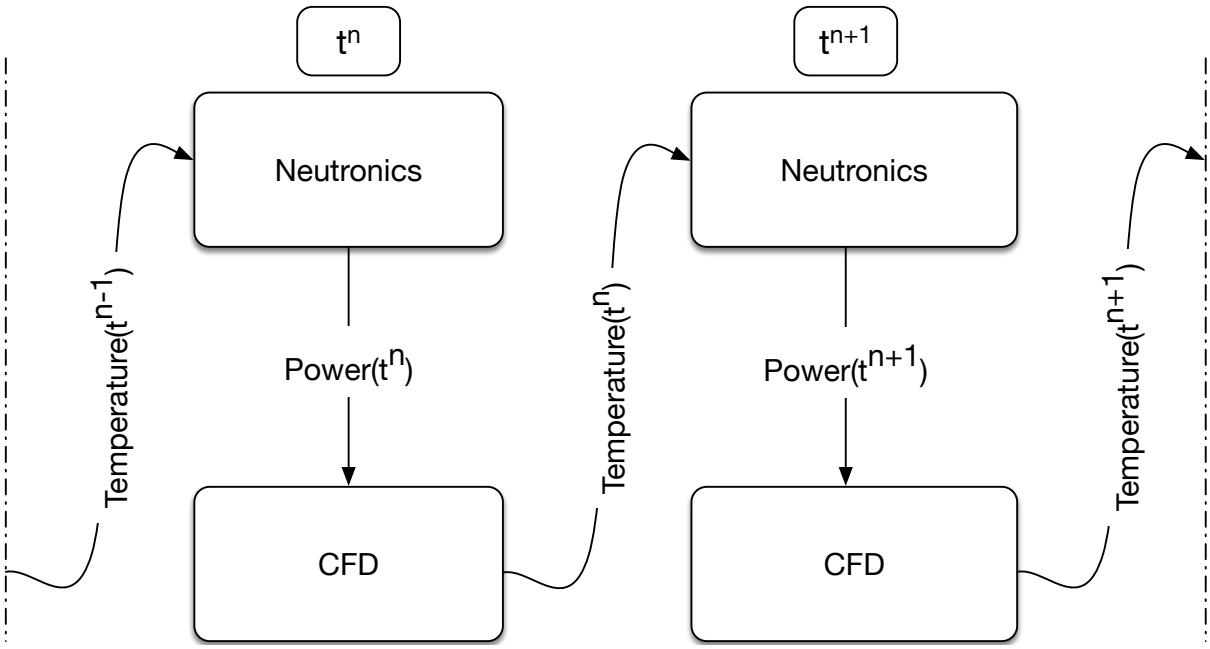


**Figure 3**: Data communication pattern employed for neutronics-CFD coupling.

**Table 2**: Profiling of MUI for the surface coupled case at maximum core-count.

| Function | % samples |
|---|---|
| MPI_Test | 10.1 |
| MPI_Recv | 0.1 |
| MPI_Barrier | 6.8 |
| Synthetic Solver work | 77.7 |
| MUI sampler | 4.0 |

## 4 Neutronics & Computational Fluid Dynamics (CFD) Coupling

In neutronics coupling, thermal energy released by fission events is transferred to the fluid. The resulting change in the fluid temperature in turn alters the transport of the neutrons. This two-way coupling is depicted in Figure 3.

MUI is used to couple DYN3D [7, 8], a deterministic neutronics code, to Code_Saturne [9, 10], an open source finite volume CFD code. This coupling is achieved with minimal intrusion into the codes (only a few extra lines of code in each case).

In order to verify the coupling was working as expected, a simple cylindrical reactor was simulated. The geometry comprises a cylinder of diameter 1.06m and height 3.04m with a uniform velocity inlet condition applied to the base of the cylinder and a zero gradient outlet condition applied to the top. Meshes comprising approximately $8,000$ cells were used for both the fluid and neutronics. Both codes were ran in serial for this test. The thermal power of the neutronics was set to Q=3GW and an average temperature increase

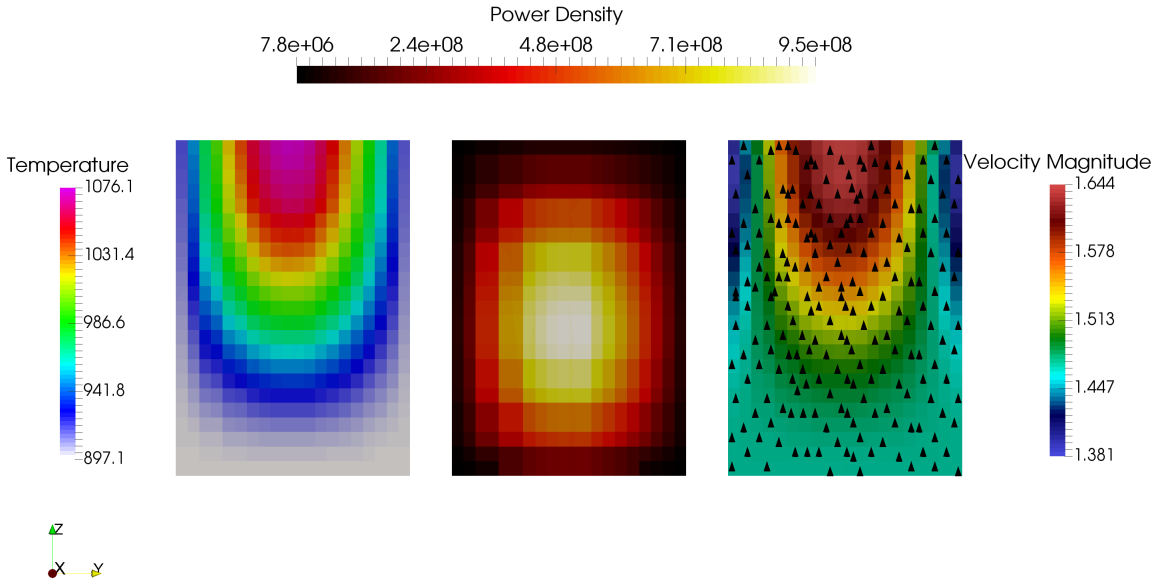


**Figure 4**: Neutronics-CFD coupling.

of $\Delta T = 100$K over the height was selected to determine the velocity condition via the expression $U = Q/(C_p A \Delta T)$ ($C_p$ = specific heat capacity; $A$ = cross-sectional area of the inlet). The fluid is a molten salt in which fissile material is dissolved (LiF:$PuF_3$:$UF_4$). The rate of the nuclear reactions is dependent on the density of the fluid and the probability of a nuclear reaction both of which are dependent on the local temperature.

Figure 4 shows the thermal power density (MW $m^{-3}$), temperature (K), and velocity (m/s) (enhanced by buoyancy, due to the heating from the neutronics). The calculation quickly reaches a steady state. The peak in power is shifted slightly lower than the centre of the cylinder, which demonstrates the influence of temperature on the fluid density and reaction probability as this peak would be located in the centre of the domain when these effects are negligible.

## 5 Conclusion

In this paper, we have tested the scalability of the MUI library in isolation. It was shown that the library scales well to at least $\mathcal{O}(10^4)$ MPI tasks. The MUI library is therefore suitable for large scale coupled HPC multi-physics and multi-scale problems.

In cases where the scalability starts to tail off, profiling has shown that the bottleneck lies within the message passing, rather than in the MUI library, and hence this bottleneck would likely be present in any MPI based coupling effort, and solutions therefore need to be considered from the perspective of solving difficult MPI communication patterns introduced by coupling. The overall overhead of the MUI library itself (i.e. without MPI

overheads) has been shown to be a few % for these large scale coupled problems. This overhead is predominantly in the sampler; since MUI operates on point clouds of data (rather than mesh data), interpolation must be conducted in order to reconstruct data on a remote mesh.

We have briefly demonstrated the use of the MUI in coupling a neutronics code to a CFD solver. This simple test acts as a demonstration of MUI working in a practical case. In the full talk, we will expand on these results further, and conduct a larger scale simulation of a fuel assembly with coupled neutronics, conjugate heat transfer, and CFD.

## 6 Acknowledgements

We thank the Engineering and Physical Sciences Research Council (EPSRC) for financial support under program grant EP/N016602/1 and EPSRC grants EP/N033841/1 and EP/R001618/1. This work was further supported by the STFC Hartree Centre's Innovation Return on Research programme, funded by the Department for Business, Energy & Industrial Strategy. The simulations on ARCHER were performed under the SLA between EPSRC and STFC.

## REFERENCES

[1] J. Borgdorff, M. Mamonski, B. Bosak, D. Groen, M.B. Belgacem, K. Kurowski, and A.G. Hoekstra. Multiscale computing with the multiscale modeling library and runtime environment. *Procedia Computer Science*, (2013) **18**:10971105.

[2] V.S. Mahadevan, E. Merzari, T. Tautges, R. Jain, A. Obabko, M. Smith, and P. Fischer. High-resolution coupled physics solvers for analysing fine-scale nuclear reactor design problems. *Phil. Trans. R. Soc. A*, (2014) **372(2021)**:20130381.

[3] J. Borgdorff, M. Mamonski, B. Bosak, K. Kurowski, M.B. Belgacem, B. Chopard, D. Groen, P.V. Coveney, and A.G. Hoekstra. Distributed multiscale computing with MUSCLE 2, the multiscale coupling library and environment. *Journal of Computational Science*, (2014) **5(5)**:719731.

[4] W. Joppich and M. Kurschner. MpCCI - a tool for the simulation of coupled applications. *Concurr. Comput.: Pract. Exper.*, (2006) **18(2)**:183192.

[5] S.M.Longshaw, A. Skillen, C. Moulinec, and D.R. Emerson. Code Coupling At Scale: Towards The Digital Product. *Advances in Parallel, Distributed, Grid and Cloud Computing for Engineering* (2017).

[6] Tang, Y.H., Kudo, S., Bian, X., Li, Z. and Karniadakis, G.E., Multiscale universal interface: a concurrent framework for coupling heterogeneous solvers. *Journal of Computational Physics*. (2015) **297**:13–31.

[7] S. Kliem, Y. Bilodid, E. Fridman, S. Baier, A. Grahn, A. Gommlich, E. Nikitin, U. Rohde. The reactor dynamics code DYN3D. *Kerntechnik*, 2016 **81(2)**, 170172.

[8] U. Rohde, S. Kliem, U. Grundmann, S. Baier, Y. Bilodid, S. Duerigen, E. Fridman, A. Gommlich, A. Grahn, L. Holt, Y. Kozmenkov, S. Mitta. The reactor dynamics code DYN3D models, validation and applications. *Progress in Nuclear Energy*, 2016. **89**, 170190.

[9] F. Archambeau, N., Mechitoua, M., Sakiz. Code Saturne: a finite volume code for the computation of turbulent incompressible flows - Industrial Applications. *International Journal on Finite Volumes*, 2004. **1**, 1-62.

[10] Y. Fournier, J. Bonelle, C. Moulinec, Z. Shang, A. Sunderland, J. Uribe. Optimizing Code Saturne computations on Petascale systems. *Computers and Fluids*. 2011. **45**, 103108.