Porting and Optimisation of *Code_Saturne* on HECToR

Zhi Shang, Charles Moulinec, David R. Emerson[1], Xiaojun Gu

Computational Science and Engineering Department
Science and Technology Facilities Council, Daresbury Laboratory
Warrington WA4 4AD, UK

Abstract

The move towards petaflop computing will require scientific software to run efficiently on many thousands of processors. For computational fluid dynamics, this imposes new challenges. We need to be able to generate very large computational grids, in excess of one billion computational cells, to ensure the processors have enough work. In addition, we need to partition these large computational meshes for efficient execution on these large scale facilities. As most grid generation codes are serial and proprietary, there is little the user can do. However, the majority of mesh partitioning software is available open-source and this study aims to understand how these codes perform when we need to create an extremely large number of computational domains. In particular, we seek to run our fluid dynamics software on a petascale system with more than 100,000 cores. This work focuses on the open-source software, *Code_Saturne*, and investigates the issues associated with pre-processing. The mesh partitioning software considered in this report has been restricted to open-source packages such as Metis, ParMetis, PT-Scotch and Zoltan. Today, Metis is the *de facto* standard but is a sequential code and is therefore limited by memory requirements. Parallel mesh partitioning software, such as ParMetis and PT-Scotch, can overcome this limitation provided the quality of the partition (edges cut, load balance) remains good. During our study, we found that the time required to perform the partition of 121M tetrahedral elements varied with the package and found that Metis consistently required the least amount of time. However, in all cases, the time to perform the partition was always modest and was not found to be a significant issue. In contrast, the memory constraints did vary with the package and PT-Scotch could generate mesh partitions in parallel (up to 131072 domains) using only 16 cores whereas ParMetis 3.1.1 required a minimum of 512 cores to create the 131072 domains. An analysis of the metrics suggests that the larger number of cores required by ParMetis results in a partition with a poor load balance. In practice, however, the simulation run time did not reflect this observation and, for up to 1024 cores, ParMetis produced the lower time to solution. Above 1024 cores, and up to 8192 cores, the sequential version of Metis showed the best speed-up. For 2048 and 4096 cores, PT-Scotch provided a better performance than ParMetis. In general, all packages did a reasonable job and it is difficult to identify any specific trends that would lead to one package being clearly superior to the others.

Keywords: *Code_Saturne*, mesh partitioning, Metis, ParMetis, PT-Scotch, Zoltan, HECToR

[1] Corresponding author at: Department of Computational Science and Engineering, Science and Technology Facilities Council, Daresbury Laboratory, Warrington WA4 4AD, United Kingdom. Tel: +44 1925 603221; Fax: +44 1925 603634.
E-mail address: david.emerson@stfc.ac.uk (D.R.Emerson)

Contents

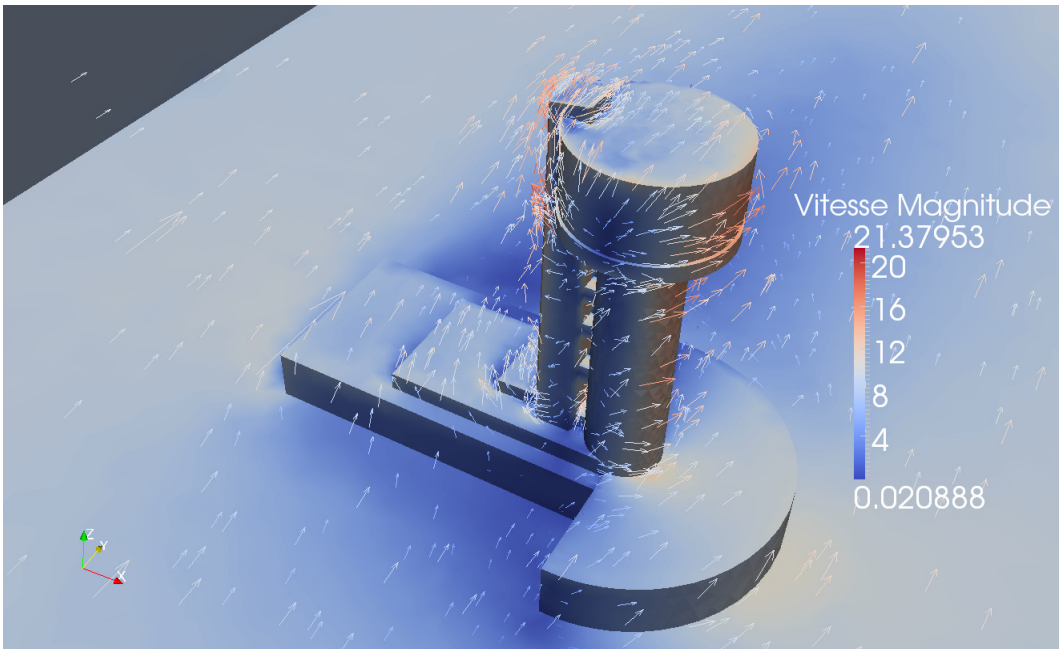
1. Introduction

The move towards hardware involving very large numbers of processing units, with state-of-the-art systems exceeding 100,000 cores, is highlighting many issues related to algorithmic scalability. However, for Computational Fluid Dynamics (CFD) software, a new challenge has emerged relating to the pre-processing stage. In common with many engineering topics, the system of equations (for CFD this is the Navier-Stokes equations) must be discretised onto a computational mesh. To run in parallel, the mesh needs to be partitioned into domains of equal size to ensure a good load balance. For structured grids, this is fairly straightforward but partitioning unstructured grids has always been more challenging. The move to petascale computing has made this challenge very immediate. As the computational meshes would have to be very large to run efficiently on 100,000 cores, the partitioning software would have to run in parallel. This project was to investigate how the partitioning software available would perform when creating domains involving very large core counts.

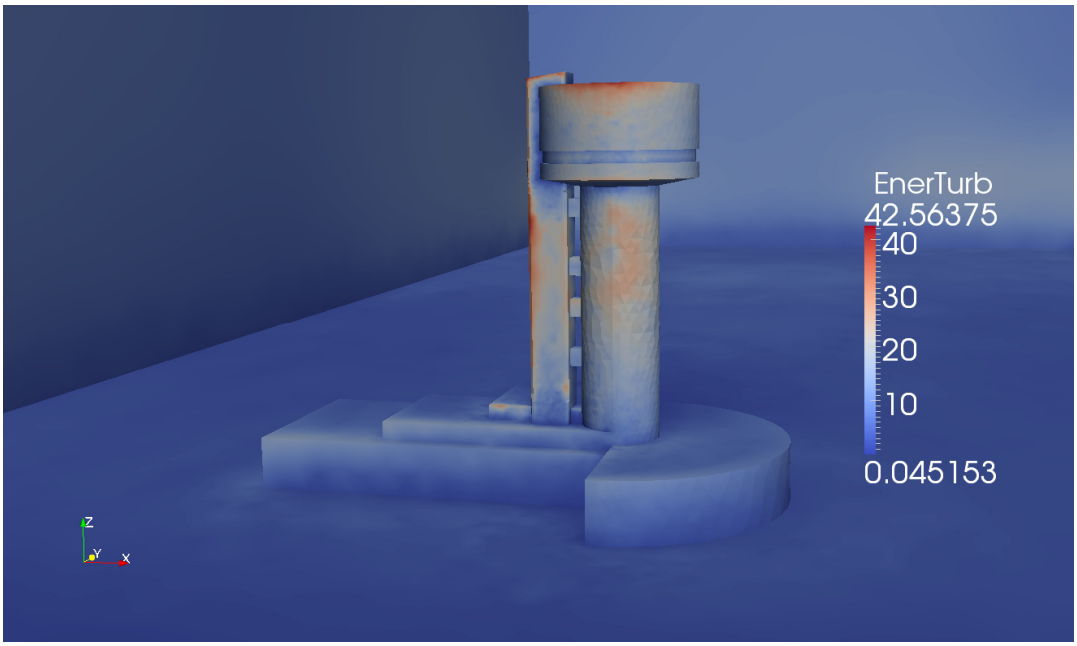
The Computational Fluid Dynamics (CFD) software, *Code_Saturne*, has been under development since 1997 by EDF R&D (Electricité de France) [1]. The software is based on a collocated Finite Volume Method (FVM) that accepts three-dimensional meshes built with any type of cell (tetrahedral, hexahedral, prismatic, pyramidal, polyhedral) and with any type of grid structure (unstructured, block structured, hybrid). This allows *Code_Saturne* to model highly complex geometries. It can simulate either incompressible or compressible flows with or without heat transfer and turbulence.

From the outset, *Code_Saturne* was designed as a parallel code and works as follows: the pre-processor reads the mesh file and currently partitions the mesh with Metis or Scotch to produce the input files for the solver. Once the simulation is complete, the output is post-processed and converted into readable files by different visualization packages (such as ParaView). Parallel code coupling capabilities are provided by EDF's FVM library. Since 2007, *Code_Saturne* has been open-source and is available to any user [2]. To retain the open-source nature of the proposed work, we have only considered partitioning software that is freely available.

One significant advantage of *Code_Saturne* is its industrial pedigree. The code was originally designed for industrial applications and research activities in several fields related to energy production. These include nuclear power thermal-hydraulics, gas and coal combustion, turbo-machinery, heating, ventilation and air conditioning. To highlight the ability of the code to handle complex geometries, Figure 1 illustrates a CFD simulation of air flow around the Daresbury tower whereas Figure 2 involves water flow around the DARPA-2 submarine.



(a) velocity field



(b) turbulence kinetic energy

Figure 1: air flow around the Daresbury tower

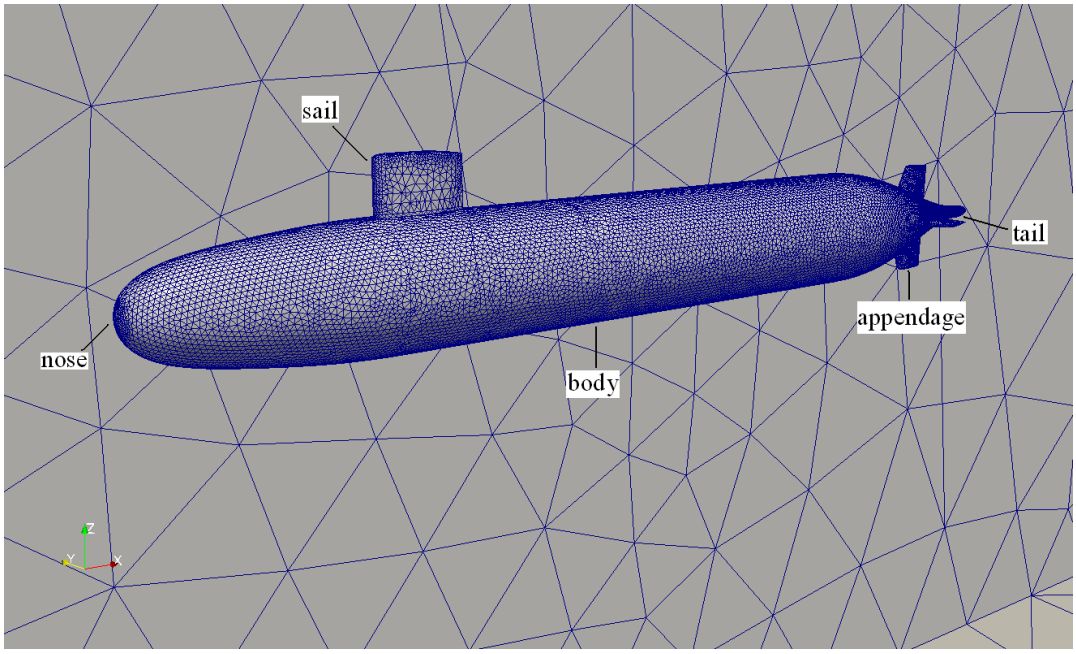


Figure 2: DARPA-2 submarine model

The submarine test case [3, 4], shown in Figure 2, was chosen to test the partitioning software and the scalability of *Code_Saturne* on HECToR. Basic details concerning the submarine's geometry are listed in Table 1.

Table 1

Geometry Length (L)	4.355 m
Body diameter (D)	0.507 m
Exit diameter	0.0075 m
Sail height (h)	0.206 m

The flow parameters, briefly listed in Table 2, correspond to the DARPA-2 experiment [3, 4].

Table 2

Free stream velocity	9 ms^{-1}
Reynolds number (based on geometry length)	3.89 10^7
Outlet pressure	2.01 10^5 Nm^{-2}

Table 3 lists the simulation parameters used for *Code_Saturne*. A standard wall function method was used for the near wall treatment.

Table 3

Turbulence model	$k-\varepsilon$
y^+	$\approx$ 30 (within 25 to 70)
Discretisation	SIMPLE (steady state)
Solver	Algebraic multigrid

The test mesh is illustrated in Table 4.

Table 4

Number of cells	195,877
Number of interior faces	404,737
Number of boundary faces	9,324
Number of vertices	47,131

Figure 3 compares the results of the pressure coefficient (Cp) at different cross sections along the submarine's body with both experiment data and several CFD codes.

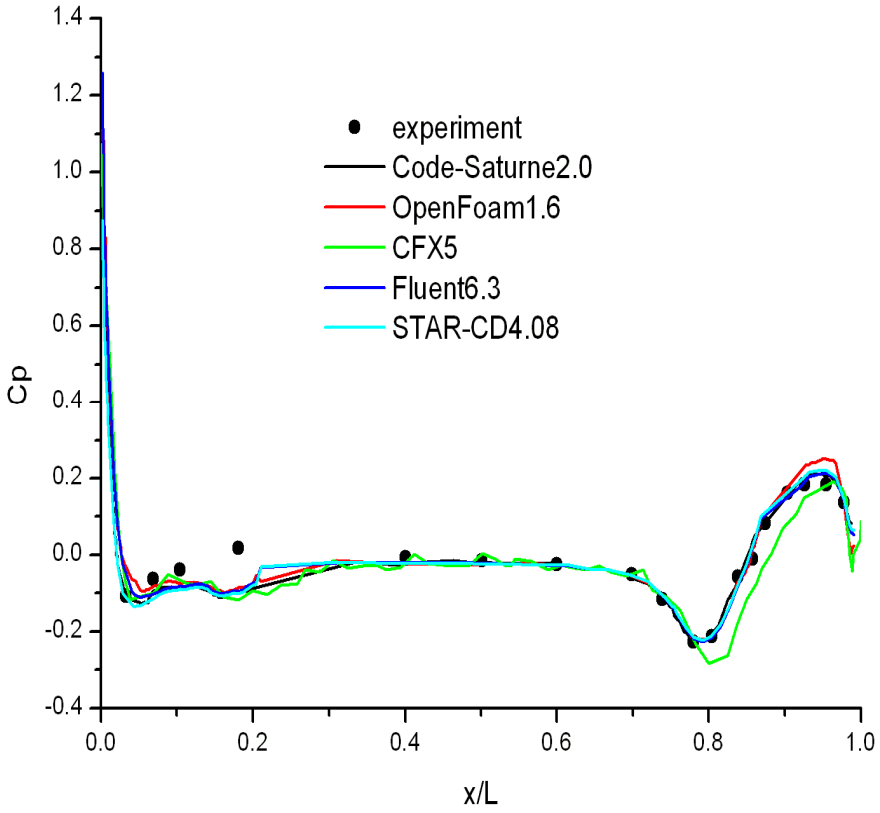


Figure 3: comparisons of pressure coefficients

From the comparisons in Figure 3, it can be seen that the results obtained by *Code_Saturne* are in good agreement with experimental data and a range of commercial software packages Fluent [5], STAR-CD [6], and CFX [7] and also the open-source software OpenFoam [8].

For this dCSE project, *Code_Saturne* 2.0.0-beta2 version will be used. Several open-source mesh partitioning packages have been investigated, including Metis, ParMetis, PT-Scotch and Zoltan. All packages were integrated into *Code_Saturne* 2.0.0-beta2 to analyse their parallel partitioning performance. A very large submarine test case was created to test the software and involved 121,989,150 (121M) tetrahedral cells.

2. Mesh partitioning software packages porting into *Code_Saturne*

In common with many parallel CFD codes, data communication is carried out by exchanging data via halo cells [9], as indicated in Figure 4. The halo cells represent the inner boundaries between different sub-domains. In this report, each sub-domain is allocated to a core or processor. All of the statistics relating to the quality of the mesh partitioning in this report include the halo cells.

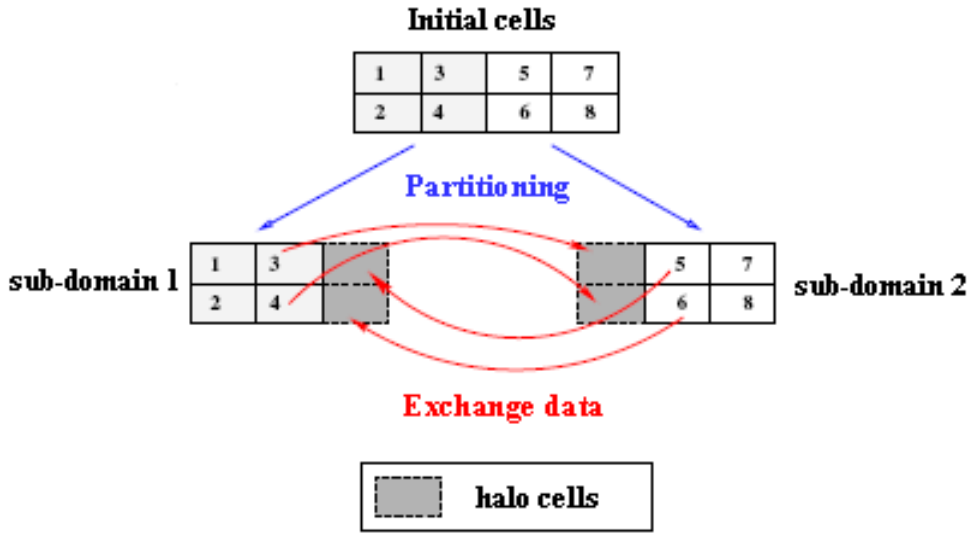


Figure 4: working mechanism of halo cells in *Code_Saturne*

2.1 Metis 5.0pre2

Metis 5.0pre2 is a sequential mesh partitioning software package [10] and is widely regarded as the *de facto* standard. It produces high quality partitioned meshes for efficient parallel execution. Since it is a serial version, the corresponding newest library of Metis 5.0pre2 was introduced through the pre-processor stage of *Code_Saturne*. The METIS_PartGraphKway function of Metis is employed by *Code_Saturne* to perform the mesh partitioning.

The connection of the libraries can be carried out through the following command lines in the installation file for *Code_Saturne*:

```
##########################
## Preprocessor Installation ##
##########################
METISPATH=$HOME/metis-5.0pre2
```

2.2 ParMetis 3.1.1

ParMetis 3.1.1 is a parallel version of the partitioning package [11]. One of the aims is to investigate its ability to produce high quality mesh partitioning in parallel. For tackling the very large meshes envisaged, it will not be possible to use the sequential version of Metis due to memory constraints. The latest version of the library, ParMetis 3.1.1, was introduced into

Code_Saturne to enable parallel mesh partitioning. The ParMETIS_V3_PartKway function of ParMetis is employed by *Code_Saturne* to perform the mesh partitioning.

The connection of the libraries, which is slightly different from the serial version, can be carried out through the following command lines during the parallel running in the 'runcase' file of the Kernel of *Code_Saturne*.

```
#####################################################
CS_LIB_ADD_metis="$HOME/ParMetis-3.1.1/libmetis.a"
CS_LIB_ADD_parmetis="$HOME/ParMetis-3.1.1/libparmetis.a"
#####################################################
```

2.3 PT-Scotch 5.1

An alternative and promising parallel partitioning software is PT-Scotch 5.1 [12]. Our goal was to assess the quality of the meshes created in parallel. The latest library of PT-Scotch 5.1 was introduced through the Kernel of *Code_Saturne*. The SCOTCH_dgraphPart function of PT-Scotch is employed by *Code_Saturne* to perform the mesh partitioning.

The connection of the libraries can be carried out through the following command lines during the parallel running in the 'runcase' file of the Kernel of *Code_Saturne*.

```
#####################################################
CS_LIB_ADD_ptscotch="$HOME/scotch_5.1/lib/libptscotch.a"
#####################################################
```

2.4 Zoltan 3.0

Zoltan 3.0 is another parallel software package that can be used for mesh partitioning [13]. For the results presented, geometric mesh partitioning was used. However, limited time restricted an in-depth analysis and our results reflect only a preliminary analysis.

The latest version of Zoltan 3.0 is introduced through the Kernel of *Code_Saturne*. The library can be introduced through the following command lines during the execution of the 'runcase' file of the Kernel of *Code_Saturne*.

```
#############################################################
CS_LIB_ADD_zoltan="$HOME/Zoltan_v3.0/src/Obj_generic/libzoltan.a"
#############################################################
```

2.5 Mesh partitioning quality

The submarine geometry previously discussed will be the standard test case. For our tests, it has 121,989,150 computational cells (121M). The key factors associated with good parallel performance for unstructured grids are load balance and statistics associated with neighbours and halo cells, and the selected partitioning packages Metis 5.0pre2, ParMetis 3.1.1, PT-Scotch 5.1 and Zoltan (RIB) 3.0 are used.

Due to memory constraints, all the partitioning of Metis 5.0pre2 was carried out on a 96 GB SGI machine in Daresbury Laboratory [14]. The peak value of memory used by Metis 5.0pre2 for partitioning the 121M case into 8192 domains is around 32 GB, which is beyond the amount of memory available on HECToR Phase 2a processing node. However, partitioning the grid in parallel can eliminate the memory limits for large scale mesh partitioning. In addition to the partitioning tools discussed, *Code_Saturne* has its own tools for creating meshes in parallel. The two approaches available are Space-Filling Curves (SFC) [15] and a simple partitioning strategy that just divides the computational cells by the processor number. At the time of this study, these tools were in the beta stage of development and testing by EDF and it was not possible to generate domains in parallel.

A special focus on the parallel graph partitioning tools, e.g. ParMetis 3.1.1 and PT-Scotch 5.1 is considered, for the generation of 32 to 131072 sub-domains from the 121M original grid. Tables 5 and 6 give the minimum number of processors ParMetis 3.1.1 and PT-Scotch 5.1 should be run on, depending on the number of sub-domains, the time spent for the graph transfer, the partitioning time, the number of edge-cuts, the load balance, the maximum number of neighbours a sub-domain might have.

The load balance is defined as the ratio between the number of cells of the smallest sub-domain divided by the number of cells of the largest sub-domain.

ParMetis 3.1.1 requires at least 32 processors to perform the partitioning up to 8192 sub-domains, and at least 512 for the 131072 case. Conversely, PT-Scotch 5.1 only requires 16 processors for all the partitions. The available partitioning strategy clearly depends on the number of processors the parallel partitioner is run on and this could have an impact on the quality of the partition obtained.

ParMetis 3.1.1's graph transfer time scales very well (speed-up of 12.5 instead of ideal 16 going from 32 to 512 processors) and PT-Scotch's graph transfer time on 16 processors is about twice as costly as ParMetis 3.1.1 on 32 processors, which indicates that both partitioning tools exhibit similar performance. On 32 processors, partitioning takes only about 10% more time for 8192 domains than for 32 sub-domains for ParMetis 3.1.1. For 16384 domains and above, 64 to 512 processors are used, but less than 40 seconds are required for ParMetis 3.1.1 to complete. The PT-Scotch 5.1 partitioning time is much longer (from just over 220 seconds for 32 sub-domains to almost 520 seconds for 131072 sub-domains). In practice, the times are all very modest and demonstrate that the computational time associated with large-scale partitioning is not a major issue. However, as expected, memory constraints do impact on how the partitioning is performed for large scale problems.

An important measure of the resulting partition is the number of edge cuts, which indicates how the load is balanced across domains and the amount of communication to be performed between sub-domains. For this particular case, the partitions generated by ParMetis 3.1.1 lead to a slightly larger number of edge-cuts compared to those obtained by PT-Scotch 5.1. In general, for partitions up to 65536 sub-domains, the percentage error between the two software packages is less than 14% and typically below 10%. However, for the largest partition, which involves 131072 sub-domains and is typical of the scale needed for a realistic petascale platform, it is about 57% larger ParMetis 3.1.1.

Considering the two software packages under investigation, PT-Scotch 5.1 produces a good load balance (above 83% for all the cases), whereas ParMetis 3.1.1 indicates a high degree of

variability with generally poor load balancing for a large number of sub-domains. Finally, the maximum number of neighbours associated with a partition is seen to increase with both ParMetis 3.1.1 and PT-Scotch 5.1 but, in general, are less with PT-Scotch. These metrics suggest that better performance should be observed with *Code_Saturne* for sub-domains obtained by PT-Scotch 5.1 rather than by ParMetis 3.1.1.

Table 5: Metrics for the 121M case - ParMetis 3.1.1

Domains	Minimum processors	Graph transfer time (s)	Partitioning time (s)	Edge cuts	Load balance	Max neighbours
32	32	22.22	49.28	1034756	0.94	23
128	32	22.40	49.69	2014746	0.88	25
512	32	22.35	50.17	3406940	0.66	35
2048	32	22.74	51.59	5647363	0.34	76
8192	32	22.31	55.73	9205459	0.36	72
16384	64	11.90	32.05	11683666	0.43	Not available
32768	64	12.12	36.72	14607593	0.40	Not available
65536	128	6.33	35.70	18474955	0.68	Not available
131072	512	1.79	35.49	36006281	0.31	Not available

Table 6: Metrics for the 121M case - PT-Scotch 5.1

Domains	Minimum processors	Graph transfer time (s)	Partitioning time (s)	Edge cuts	Load balance	Max neighbours
32	16	47.15	221.04	924584	0.93	15
128	16	43.33	257.66	1779971	0.92	26
512	16	43.56	296.48	3143164	0.90	43
2048	16	44.87	338.25	5299619	0.87	46
8192	16	53.51	385.38	8755670	0.86	50
16384	16	51.68	412.05	11173729	0.86	54
32768	16	56.27	442.80	14188598	0.84	64
65536	16	62.99	478.25	18075714	0.83	61
131072	16	66.61	519.08	22911992	0.83	50

2.6 Parallel performance on HECToR

For the work presented here, all of the tests have been performed on HECToR Phase2a (Cray XT4) [16]. Figure 9 shows the CPU time per time step as a function of the number of cores for *Code_Saturne*'s simulations running on up to 8192 cores and pre-processed by Metis 5.0pre2 (on the 96 GB SGI machine located at Daresbury Laboratory, as described in Section 2.5), ParMetis 3.1.1, PT-Scotch 5.1 and Zoltan (RIB) 3.0. The CPU time per time step decreases for all of the simulations as the number of cores is increased, with ParMetis 3.1.1 partitioning leading to the fastest simulation but no really significant difference up to 512 cores. From 1024 cores on, Zoltan (RIB) geometric partitioning has a clear effect on the poor performance of *Code_Saturne*. This is to be expected, as geometry-based partitioning tools usually do not perform as well as graph-based tools, with the number of edge-cuts and neighbours being very large. From 2048 on, Metis 5.0pre2 produces better results than ParMetis 3.1.1 and PT-Scotch 5.1. In Tables 5 and 6, PT-Scotch 5.1 generally showed better metrics than ParMetis 3.1.1 and for 2048 and 4096 cores, this is confirmed by *Code_Saturne* performing better. However, there is no conclusive evidence to suggest the improved metrics offered by PT-Scotch 5.1 results in the best code performance.

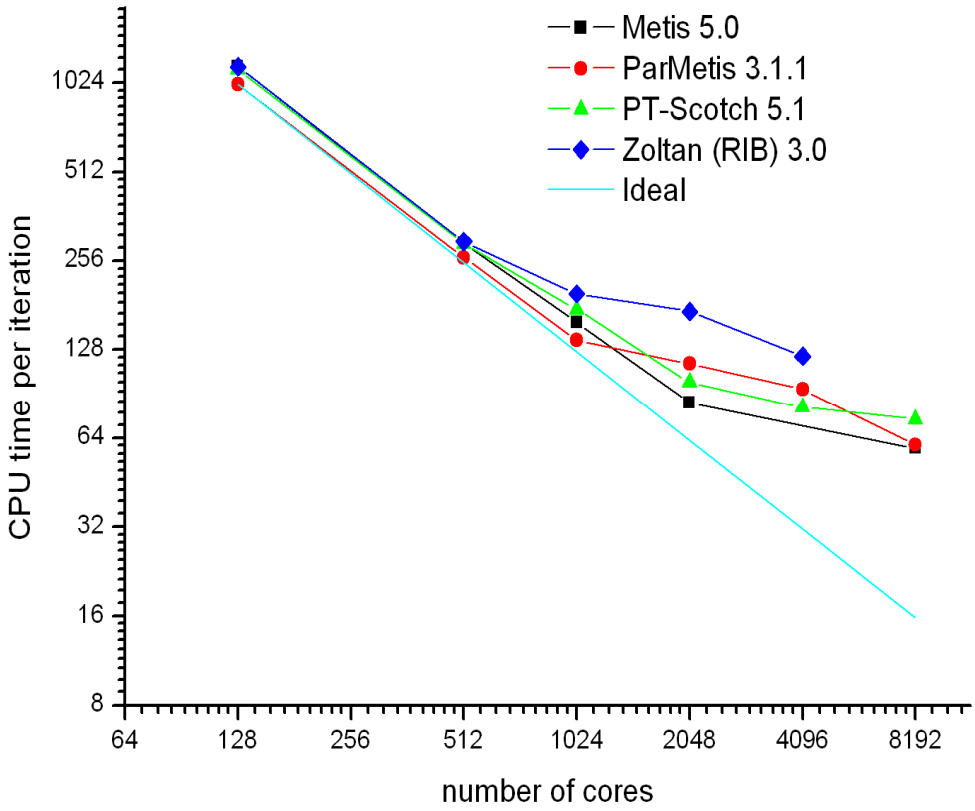


Figure 9: CPU time per time step as a function of the number of cores

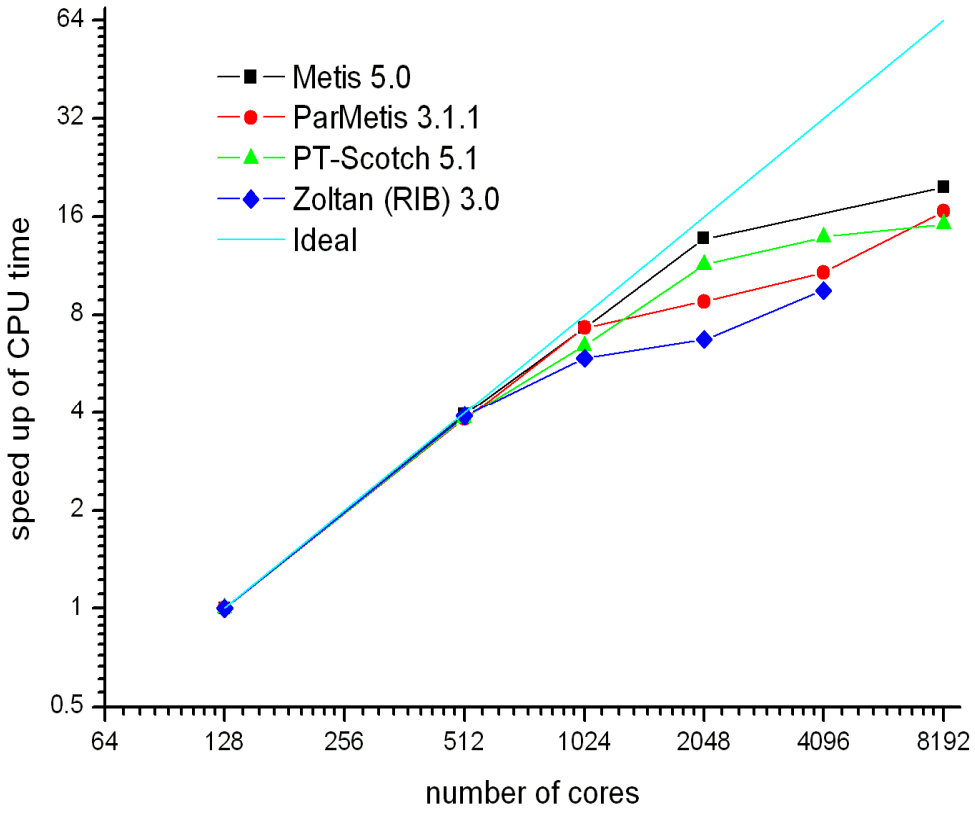


Figure 10: Speed-up as a function of the number of cores

Figure 10 shows the speed-up based on the CPU time per iteration as a function of the number of cores. Metis 5.0pre2 demonstrates the best performance with a speed-up almost ideal up to 2048 cores, whereas Zoltan (RIB) 3.0 exhibits the poorest performance. Overall, PT-Scotch is the best parallel partitioner for 2048 and 4096 cores, as shown by *Code_Saturne*'s speed-up, which is very close (about 10% lower) to the speed-up obtained when Metis 5.0pre2 is used as the partitioning tool. Despite the poor metric indicators of ParMetis 3.1.1, especially at high core counts, the code generally performs well and at lower core counts produces the minimum run-time. Even at 8192 cores, it is only just below Metis 5.0pre2 which gives the best performance at the highest number of cores tested on HECToR Phase 2a.

3. Conclusions

Mesh partition is a key component of solving grid-based problems using unstructured meshes and the advent of petascale and, before 2020, exascale systems has highlighted the need to revisit this "*solved*" problem. To test the available partitioning software for its suitability of creating very large-scale mesh partitions, we have used *Code_Saturne*, an open-source CFD package that is used extensively in industry and Europe. The geometric problem is based on the DARPA submarine which was meshed with 121M tetrahedral elements to reflect the scale of the problem sizes expected on a petascale architecture.

The partitioning software considered was Metis 5.0pre2, ParMetis 3.1.1, PT-Scotch 5.1 and Zoltan (RIB) 3.0 which are all available as open-source packages. As partitioning a graph is considered to be an NP-hard problem, all packages use heuristics in their solution strategy.

This naturally leads to differences in the algorithms employed and their implementation with the corresponding result that each software package produces a different partition. As Metis 5.0pre2 is sequential, there are natural memory limitations that impact the total number of sub-domains the package can create.

For PT-Scotch 5.1, we found that we could generate 131072 sub-domains using just 16 cores. In contrast, ParMetis 3.1.1 required a minimum of 32 cores to partition the DARPA submarine and the number of cores grew with the 131072 partition requiring at least 512 cores. However, ParMetis 3.1.1 was consistently faster than PT-Scotch but, in practice, the amount of time required to partition the mesh was very modest. Although the time is not a major issue, it is clear that memory constraints could make an impact on deciding which package to use.

If we consider the metrics presented in Tables 5 and 6, the indications are that PT-Scotch 5.1 might provide the better solution. In practice, this was not the case. In contrast to the statistics produced, ParMetis 3.1.1 provided the minimum run times up to 1024 cores whereas Metis 5.0pre2 delivered the best performance above 1024 cores and up to the limit of 8192 cores tested on HECToR phase 2a. We did find PT-Scotch 5.1 performing better than ParMetis 3.1.1 on 2048 and 4096 cores. In general, however, it is not possible to identify specific trends that would lead to one package being clearly superior to the others.

The results presented for Zoltan are very preliminary. Although this package indicates it provides the worst performance, the limited time available to investigate this package means that it would be unfair to take this as a definitive result. However, one aspect that became apparent was that it was necessary to allocate one core to each sub-domain being created. This could be attributed to a limited understanding of how the software works but it would be undesirable feature.

As a final comment and observation, all current packages considered performed well and are very similar at low core counts. We should also note that these results are only valid for the DARPA test case running with *Code_Saturne* on HECToR phase 2a and other codes and problems could perform in a different way, although we anticipate that the results and observations are fairly general.

Acknowledgements
The authors would like to thank the Engineering and Physical Sciences Research Council (EPSRC) for their support of Collaborative Computational Project 12 (CCP12) and Dr. Ming Jiang of STFC who assisted with the partitioning libraries of *Code_Saturne*.

This project was funded under the HECToR Distributed Computational Science and Engineering (CSE) Service operated by NAG Ltd. HECToR - A Research Councils UK High End Computing Service - is the UK's national supercomputing service, managed by EPSRC on behalf of the participating Research Councils. Its mission is to support capability science and engineering in UK academia. The HECToR supercomputers are managed by UoE HPCx Ltd and the CSE Support Service is provided by NAG Ltd.

References

[1] 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, 1(1), 2004.

[2] *Code_Saturne* open source: http://www.code-saturne.org.

[3] M. Sohaib, M. Ayub, S. Bilal, S. Zahir, M.A. Khan. Calculation of flows over underwater bodies with hull, sail and appendages. Technical Report of National Engineering and Scientific Commission, Islamabad, 2001.

[4] Cindy C. Whitfield. Steady and Unsteady Force and Moment Data on a DARPA2 Submarine. Master Thesis of the Faculty of the Virginia Polytechnic Institute and State University, August 1999, USA.

[5] www.fluent.co.uk.

[6] www.cd-adapco.com/products/STAR-CD.

[7] www.ansys.com/products/fluid-dynamics/cfx.

[8] www.openfoam.com.

[9] EDF R&D. *Code_Saturne* version 1.3.2 practical user's guide. April 2008.

[10] G. Karypis, V. Kumar. Metis: A Software Package for Partitioning Unstructured Graph, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices. Version 5.0, 2007.

[11] http://glaros.dtc.umn.edu/gkhome/views/metis/.

[12] http://www.labri.fr/perso/pelegrin/scotch/.

[13] http://www.cs.sandia.gov/zoltan/Zoltan.html.

[14] http://www.cse.scitech.ac.uk/sog/.

[15] http://www.win.tue.nl/~hermanh/stack/dagstuhl08-talk.pdf.

[16] http://www.hector.ac.uk.