Hindawi Publishing Corporation
Journal of Engineering
Volume 2013, Article ID 850148, 6 pages
http://dx.doi.org/10.1155/2013/850148

Research Article

Large-Scale CFD Parallel Computing Dealing with Massive Mesh

Zhi Shang

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

Correspondence should be addressed to Zhi Shang; shangzhi@tsinghua.org.cn

Received 15 December 2012; Revised 15 March 2013; Accepted 18 March 2013

Academic Editor: Michael Fairweather

Copyright © 2013 Zhi Shang. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In order to run CFD codes more efficiently on large scales, the parallel computing has to be employed. For example, in industrial scales, it usually uses tens of thousands of mesh cells to capture the details of complex geometries. How to distribute these mesh cells among the multiprocessors for obtaining a good parallel computing performance (HPC) is really a challenge. Due to dealing with the massive mesh cells, it is difficult for the CFD codes without parallel optimizations to handle this kind of large-scale computing. Some of the open source mesh partitioning software packages, such as Metis, ParMetis, Scotch, PT-Scotch, and Zoltan, are able to deal with the distribution of large number of mesh cells. Therefore they were employed as the parallel optimization tools ported into Code_Saturne, an open source CFD code, for testing if they can solve the issue of dealing with massive mesh cells for CFD codes. Through the studies, it was found that the mesh partitioning optimization software packages can help CFD codes not only deal with massive mesh cells but also have a good HPC.

1. Introduction

Code_Saturne is a multipurpose computational fluid dynamics (CFD) software [1]. The code was originally designed for industrial applications and research activities in several fields related to energy production. They are including nuclear power thermal hydraulics, gas and coal combustion, turbomachinery, heating, ventilation, and air conditioning.

This code is based on a colocated finite volume approach that can cope with three-dimensional meshes built with any type of cell (tetrahedral, hexahedral, prismatic, pyramidal, and polyhedral) and with any type of grid structure (unstructured, block structured, and their hybrid). The code is able to simulate either incompressible or compressible flows, with or without heat transfer, and has a variety of models to account for turbulence [1].

It provided a mesh partitioning method, which is named as space-filling curve (SFC) [2], to deal with the optimization of parallel computing in Code_Saturne. For extending the parallel computing abilities, several open source mesh partitioning software packages, such as Metis [3], ParMetis [4], Scotch [5], PT-Scotch [6], and Zoltan [7], were ported into Code_Saturne 2.0.0-beta2 for the optimization on HPC in this paper. Through the tests using a DARPA submarine model [8, 9] with the mesh cells above 121 million, it was found that some of the software packages can perform the optimization for the CFD code, Code_Saturne, to deal with massive meshes on large-scale parallel computing.

2. Validation of CFD Code

Normally before the formal applications, the CFD code needs to be validated. It is because CFD simulations are usually dependent on the options of models, especially when the flow regime is controlled by turbulence. Generally, the validation of the CFD simulation can be carried out through the comparisons between the numerical results and experiments or theoretical results [10]. Due to the complex characteristics of the flows, it is generally impossible to get the theoretical analytical results. Therefore the validation can also be performed by the comparisons between the numerical results and experiments or the simulations by other software.

Before the studies of high-performance computing (HPC), the validation of the CFD simulation was carried out to ensure that Code_Saturne 2.0.0-beta2 is able to cope with the CFD calculation for the complex geometry under turbulence regime. In this paper, the simulations of the DARPA submarine [8, 9], shown in Figure 1, were chosen as the target of the validation. The detail geometry sizes of the

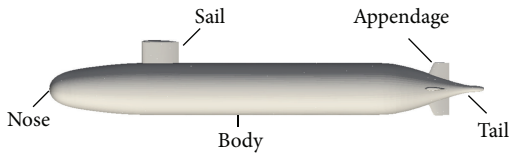


Figure 1: DARPA submarine model.

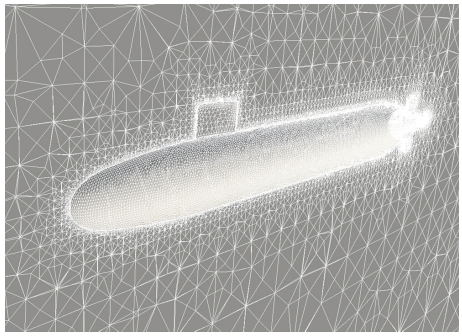


Figure 2: Tetrahedral mesh around the DARPA submarine.

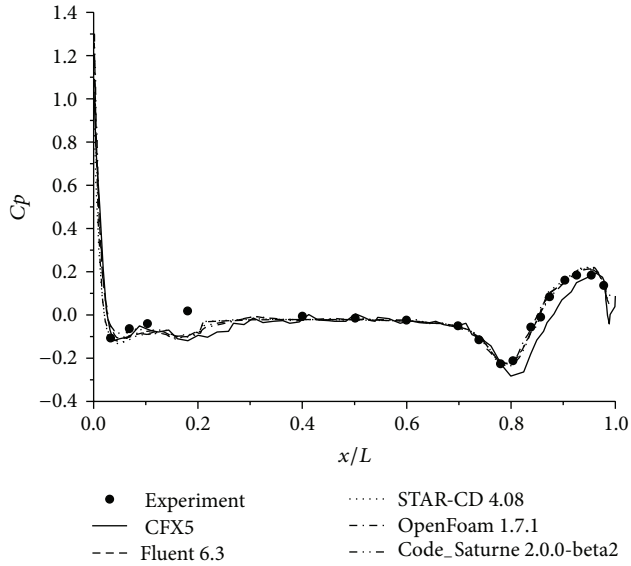


Figure 3: Comparisons of pressure coefficients.

DARPA submarine [8, 9] are of the geometry length, 4.355 m; cylinder body diameter, 0.507 m; tail diameter, 0.0075 m and sail height, 0.206 m.

The flow parameter for the simulation is the same as the experiments of the DARPA submarine [8, 9]. The flow is with a 9 m/s flowing speed attacking at the submarine's nose with a zero angle. The corresponding Reynolds number reaches 3.89×10^7 high based on the length of the submarine.

The simulation modeling can use the models provided by Code_Saturne. The RANS model (k-ε model) was employed for modeling the turbulence. The standard wall function was chosen as the near-wall treatment [11]. The stretched prism mesh with growth rate 1.2 near wall was adopted, which keeps the near wall y^+ value as 30 averagely and within the range of 25 to 70 in this paper. The unstructured tetrahedral mesh around the submarine is shown in Figure 2.

Figure 3 shows the comparisons of the pressure coefficients (C_p) at different cross-sections along the submarine's body with experiment and several other CFD software packages.

From the comparisons in Figure 3, it can be seen that the simulation results by Code_Saturne 2.0.0-beta2 are coincident with experiments. The curve shows a quite good agreement with experiment results. Code_Saturne 2.0.0-beta2 is able to get the simulation results equivalent to the famous commercial software packages Fluent [12] and STAR-CD [13]. Within the whole region, the simulation of Code_Saturne is near the benchmark results by OpenFOAM [14].

After the validation, the case with 121, 989, and 150 (121 M) tetrahedral cells was built up for the HPC tests. The tests are aimed to measure if these mesh partitioning software packages are able to optimize the CFD code, Code_Saturne, for obtaining a good HPC through dealing with the massive meshes for the large-scale parallel computing.

3. Porting Mesh Partitioning Software Packages into CFD Code

Different software code has its own characters and code structures. Before the utilization of the code, the user must know the structure of the code and find out where is the place of the interface for linking the extra libraries to the code. Through the studies, it was found that the mesh partitioning software codes can be ported into Code_Saturne through the connection of the extra libraries.

The source code of Code_Saturne contains four parts as kernel, preprocessor, and opt and src. The solver is within the kernel. The preprocessor performs reading the mesh in and checking the mesh quality. The opt includes the libraries for dealing with the numerical procedures and the src contains all the basic mathematic and finite volume source codes. Therefore, the extra libraries have two ways to be embedded into Code_Saturne. One is through kernel directly linked to the solver. Another one is using the preprocessor to perform the mesh reading in and checking together with a mesh partitioning. Actually these two methods were used in this paper. It is that Metis and Scotch libraries were linked to preprocessor as a serial preprocessing, and ParMetis, PT-Scotch, and Zoltan libraries were linked to kernel to do the parallel mesh partitioning in the solver.

ParMetis is a parallel version of Metis [3]. Both of Metis and ParMetis use a graph mesh partitioning method to realize the parallel computing optimization. During the mesh partitioning, a coarsening graph is abstracted from the original mesh firstly. Then the partitioning based on the coarsening graph is carried out through minimizing the edge-cut and optimizing the load balance by a multilevel k-way graph partitioning [4]. Finally through the multilevel refinement the whole mesh partitioning is recovered on the original mesh. After the mesh partitioning, the original mesh cells are distributed into a number of subdomains. The number

of subdomains equals the number of processors. If the optimization performs well, the later parallel computing will be easy to have a good HPC.

PT-Scotch is a parallel version of Scotch [5]. During the mesh partitioning, Scotch and PT-Scotch use a dual recursive bi-partitioning algorithm to perform the mesh partitioning. It is the same as ParMetis, starting from a coarsening phase that constructs a coarse graph. The coarsening process reduces the size of the graph to bipartition through collapse vertices and edges from the original mesh. The initial partitioning is carried out on the coarse graph. Then a multilevel partitioning process which is in conjunction with the banded diffusion method to refine the projected partitions until the whole mesh partitioning is built up on the original mesh.

Zoltan is a direct parallel version for the mesh partitioning [7]. Both of graph partitioning and geometry partitioning can be realized in Zoltan. Owing to the robust geometry partitioning, Zoltan was ported into Code_Saturne 2.0.0-beta2 only with its geometry partitioning method in this paper. There are three geometry mesh partitioning methods provided by Zoltan. They are recursive coordinate geometric bisection (RCB) [15], recursive inertial bisection (RIB) [16], and Hilbert space-filling curve partitioning (HSFC) [17]. After the tests, HSFC and RIB were employed in this paper for the studies.

4. Influence of Mesh Partitioning on HPC

Code_Saturne has its own mesh partitioning tools for the parallel computing. They are simple mesh partitioning [1] and space-filling curve (SFC) method [2]. The simple mesh partitioning actually did not perform any optimization on the mesh distributions.

Figure 4 shows the 3D mesh partitioning results on 4 processors for the DARPA submarine. It can be seen that Metis, ParMetis, Scotch, PT-Scotch, and Zoltan (RIB) can produce the neat inner boundaries among the processors. Therefore, they can have fewer neighbor processors. It will be in merit for the data communication during the parallel computing.

Table 1 shows the comparisons of CPU time and speedup at 512 processors for 121 M case. The CPU time and speedup were calculated by the average value from the numbers of iterations. In Table 1, all the HPC tests were carried out on HECToR Phase2a Cray XT4, which was a cluster supercomputer as a high-end computing resource in the UK [18].

Due to the failure of initialization of the iterations, simple method of Code_Saturne produced nothing, therefore it is not including the result of simple method in Table 1. From Table 1, it can be seen that ParMetis can produce the best HPC and have a speedup value above 30 times compared with SFC. All the neat inner boundaries generation software, such as Metis, ParMetis, Scotch, PT-Scotch and Zoltan (RIB), was able to have higher speedup, that is, good performances on HPC. Overall, the graph mesh partitioning methods (Metis, ParMetis, Scotch, and PT-Scotch) are better than the geometry mesh partitioning methods (SFC, Zoltan (RIB), and Zoltan (HSFC)) on the performance of HPC.

5. Performance Dealing with Massive Mesh on HPC

Through the comparisons of mesh partitioning in Section 4, it can be seen that the different mesh partitioning methods can produce different mesh distribution results. They will affect the high performance parallel computing. Usually the high performance parallel computing can be estimated by the load imbalance, which is the reciprocal of load balance normally larger than 1.0, before the parallel computing [19]. In this paper the load imbalance is defined as the number of processors multiplying the maximum number of cells among processors, and then the multiplied result was divided by the whole mesh cells.

Table 2 shows the comparisons of the load imbalance of different mesh partitioning methods at different processors (subdomains) on the case of 121 M (121, 989, 150 tetrahedral cells). The symbolic in the first row is of Processor (processors), Simple (simple method), SFC (space-filling curve), Metis (Metis 5.0pres2), Scotch (Scotch 5.1), ParMetis (ParMetis 3.1.1), PT-Scotch (PT-Scotch 5.1), Z_RIB (Zoltan (RIB) 3.0), and Z_HSFC (Zoltan (HSFC) 3.0), respectively. Due to the failure of initialization, the mesh partitioning was failed for SFC at 4096 processors.

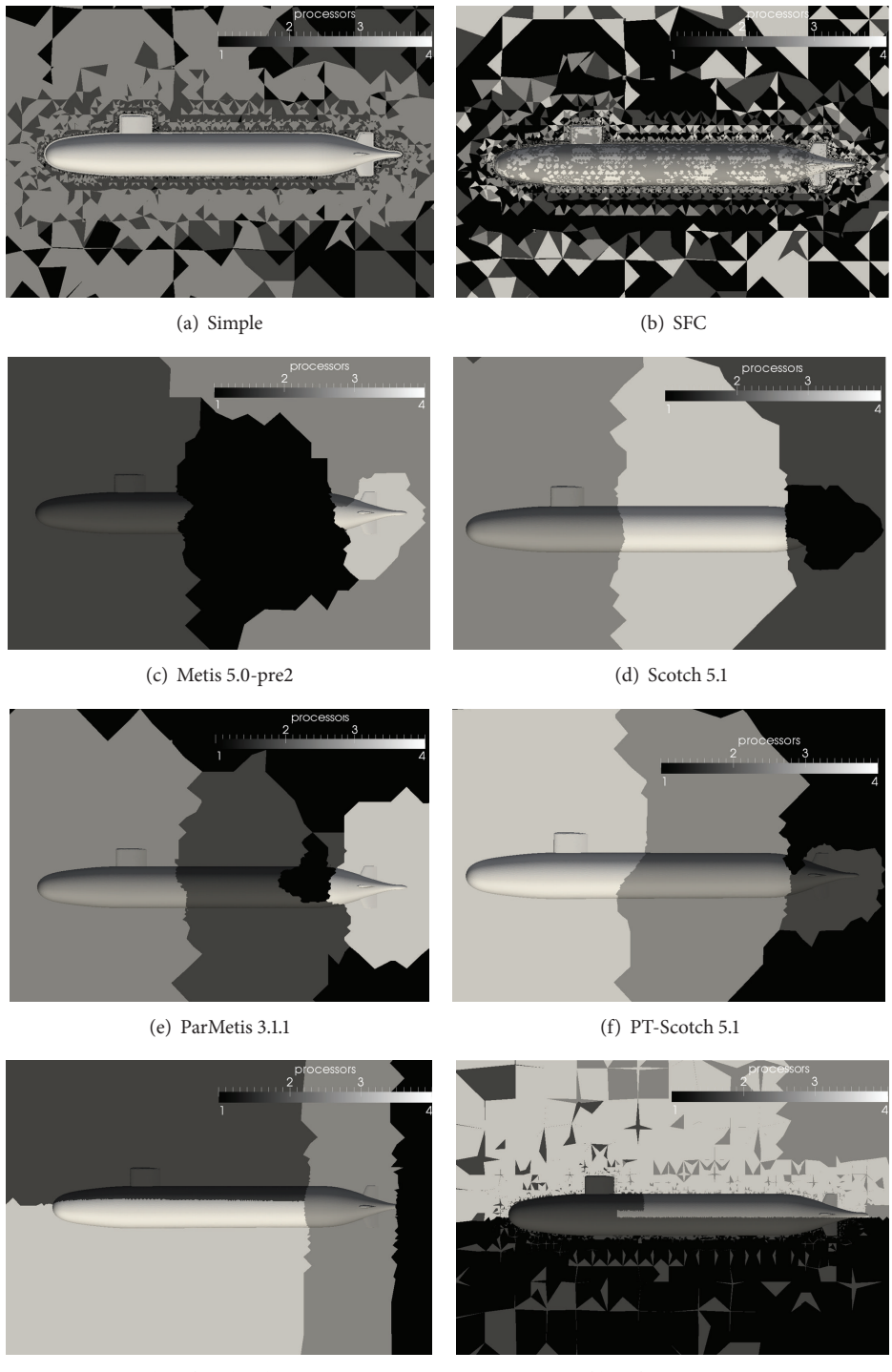
The mesh partitioning by Scotch, which can only be executed sequentially due to that it is a serial code, employed lots of memories. Through the tests it was found that when the number of processors (subdomains) is greater than 1024 even on a computer with 250 Gb memories it was not enough for Scotch to do the mesh partitioning. Therefore the statistics of Scotch do not include the results when the subdomains are greater than 1024 in Table 2.

From Table 2, it can be seen that the Simple and SFC provided by Code_Saturne have the load imbalance larger than the extra mesh partitioning software packages. The large load imbalance means that the mesh distribution is seriously nonuniform among the processors. Therefore the processor, which has the maximum number of cells, will spend a lot of time on the iteration computing, and the other processors have to wait for it with a long idle time. It is unacceptable especially for the large-scale high-performance parallel computing [20].

ParMetis has the lowest load imbalance value among the mesh partitioning methods when the number of processors is less than 512. When the number of processors is greater than 1024, Metis has the lowest load imbalance value.

Due to the memory limits, all the mesh partitioning of Metis, which is a serial code, was carried out at SGI machine at Daresbury Laboratory in the UK, which has 96 Gb memories at one processor. The peak value of memory used by Metis on the mesh partitioning of 121 M case is around 30 Gb for all the mesh partitioning.

Figure 5 shows comparisons of the CPU time. It can be seen that two groups are separated. One is composed by Zoltan (RIB) and Zoltan (HSFC). Another one is composed of Metis, ParMetis, and PT-Scotch. Following the increase of the processors, the CPU time spent by Zoltan (RIB) and Zoltan (HSFC) is averagely higher than others about 200% when the number of processors is greater than 1024. Metis



(a) Simple (b) SFC

(c) Metis 5.0-pre2 (d) Scotch 5.1

(e) ParMetis 3.1.1 (f) PT-Scotch 5.1

(g) Zoltan (RIB) 3.0 (h) Zoltan (HSFC) 3.0

Figure 4: Comparisons of mesh partitioning.

Table 1: Comparisons of CPU time and speed up on 512 processors.

	SFC	Metis	Scotch	ParMetis	PT-Scotch	Zoltan (RIB)	Zoltan (HSFC)
CPU time (s)	8007.8	294.3	283.7	263.7	293.9	298.1	306.5
Speedup	1.00	27.21	28.23	30.37	27.25	26.86	26.13

Table 2: Load imbalance under different processors.

Processor	Simple	SFC	Metis	Scotch	ParMetis	PT-Scotch	Z_RIB	Z_HSFC
32	6.46376	3.31521	1.05662	1.04363	1.01575	1.06292	1.02420	1.03958
64	7.24894	3.62416	1.03822	1.05928	1.05571	1.08747	1.03215	1.04492
128	7.24965	3.53683	1.06850	1.07096	1.06366	1.08627	1.04832	1.05739
512	7.80080	3.90431	1.09823	1.10278	1.09765	1.12977	1.09586	1.09930
1024	7.80855	4.17676	1.11929	1.13165	1.12763	1.23072	1.13053	1.17049
2048	7.80841	4.96599	1.13735	—	1.14698	1.15967	1.20171	1.20329
4096	7.91606	—	1.16461	—	1.18525	1.21051	1.37601	1.30728
8192	7.87476	5.51827	1.20426	—	1.22898	1.23925	1.63646	1.36711

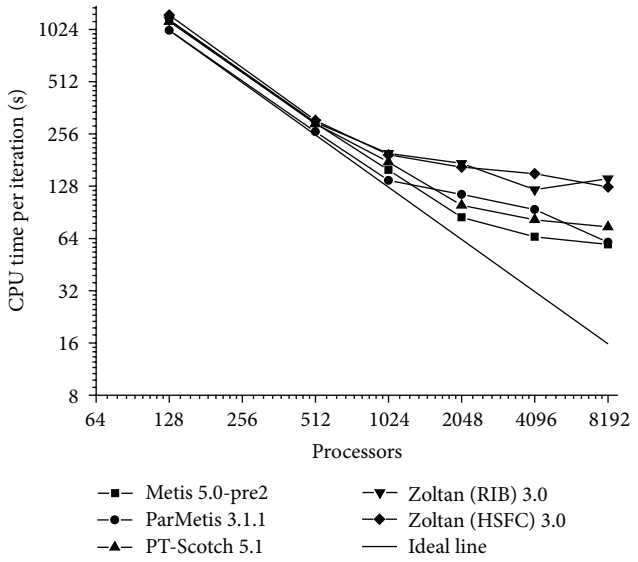


Figure 5: CPU time on different processors.

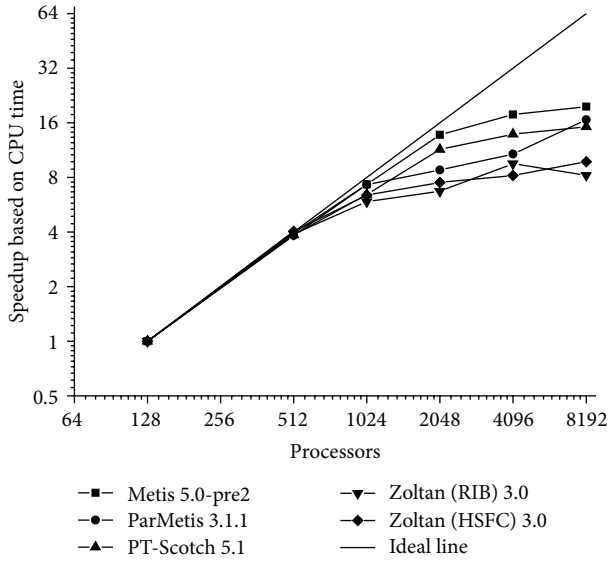


Figure 6: Speedup of CPU time on different processors.

has the outstanding falling curve. When the number of processors is less than 1024, ParMetis has the lowest CPU time. However, when the number of processors is greater than 1024, the CPU time spent by ParMetis increases higher than Metis about 100% and PT-Scotch about 50% averagely. PT-Scotch has the CPU time higher than Metis about 50%.

Figure 6 shows the curve of speedup based on the CPU time of per iteration. Metis has the fastest speedup following the increase of the number of processors higher than others when the number of processors is greater than 1024. PT-Scotch has the similar speedup to Metis. ParMetis has lower speedup than PT-Scotch but higher than Zoltan (RIB) and Zoltan (HSFC). When the number of processors reaches 8192, ParMetis has the speedup that is higher than PT-Scotch 28%.

From Figure 6, it can be seen that the different mesh partitioning methods produce quite different results of the speedup performance mainly at the number of processors greater than 512. The reason can be analyzed from the load imbalance in Table 2. From the load imbalance results, it can be seen that since the number of processors is greater than 512, Metis has the smallest load imbalance values. The small load imbalance value will produce the uniform distribution of the mesh cells among the processors. Therefore during the iteration computing, all the processors can keep synchronous well. It has no extra idle time at waiting for the calculation of synchronization. Accordingly every single processor CPU time spending on the computing is saturated otherwise; the CPU time at some of processors will be oversaturated and at some of processors will be undersaturated. Needless to say that the oversaturated processors will spend more computing time than the saturated processors, it means that the large load imbalance will produce lower HPC than the small load imbalance. It can be seen that in Table 2 when the number of processors is greater than 512, Metis, ParMetis, and PT-Scotch produce smaller load imbalance than Zoltan (RIB) and Zoltan (HSFC). Therefore, Metis, ParMetis, and PT-Scotch have higher speedup performance than Zoltan (RIB) and Zoltan (HSFC), as shown in Figure 6.

However, the tendency of the speedup performance between ParMetis and PT-Scotch is against the analysis of load imbalance; that is, the small load imbalance produces higher speedup performance. It must have some other factors affecting the speedup performance. The reasons are the distributions of neighboring processors which can be referenced from Shang's researches [21].

6. Conclusions

From the performance of HPC, it can be seen that the mesh partitioning methods will affect the performance of HPC. The graph mesh partitioning method is able to obtain better HPC than geometry mesh partitioning method. The load imbalance is the key criterion to measure HPC. The lower load imbalance the better HPC can be obtained.

From the comparisons of HPC, it can be seen that Metis 5.0 has the highest high parallel performance synthetically. However, it has to employ large memories to perform the mesh partitioning for large-scale parallel CFD application due to Metis 5.0 is a sequential code that has to be carried out by a single processor.

The parallel mesh partitioning software packages can get rid of the memory limit. However the quality is slightly lower than serial version of Metis 5.0. Within the parallel versions, ParMetis 3.1.1 and PT-Scotch 5.1 have the similar high parallel performance. Zoltan (RIB) 3.0 and Zoltan (HSFC) 3.0 have worse HPC compared with others.

In case of ignoring the memory limit, Metis can be used for large-scale parallel CFD application. Among the parallel mesh partitioning software packages, ParMetis and PT-Scotch are recommended to CFD code for the large scale parallel CFD computing.

Acknowledgments

The author would like to thank Daresbury Laboratory of Science and Technology Facilities Council (STFC) for providing the facilities to support the research. They are grateful for the fund support from HECToR CSE, The Numerical Algorithms Group Ltd. (NAG). They are thankful for the help from Mr. Charles Moulinec (STFC Daresbury Laboratory).

References

[1] F. Archambeau, N. Mechitoua, and M. Sakiz, "Code_Saturne: a finite volume code for the computation of turbulent incompressible flows—industrial applications," *International Journal on Finite*, vol. 1, no. 1, pp. 1–62, 2004.

[2] G. Peano, "Sur une courbe, qui remplit toute une aire plane," *Mathematische Annalen*, vol. 36, no. 1, pp. 157–160, 1890.

[3] G. Karypis and V. Kumar, "A software package for partitioning unstructured graph, partitioning meshes, and computing fill-reducing orderings of sparse matrices," 1998, Version 4.0 and Version 5. 0pre2, pp. 1–44.

[4] G. Karupis, K. Schloegel, and V. Kumar, "ParMetis parallel graph partitioning and sparse matrix ordering library," 2003, Version 3. 1, pp. 1–29.

[5] F. Pellegrini, 2008, Scotch and libScotch 5.1. User's guide, pp. 1–127.

[6] F. Pellegrini, 2009, PT-Scotch and libScotch 5.1. User's guide, pp. 1–76.

[7] E. Boman, K. Devine, R. Heaphy et al., "Zoltan: parallel partitioning, load balancing and data management seivices," 2007, User's guide Version 3.0, pp. 1–173.

[8] C. C. Whitfield, *Steady and unsteady force and moment data on a DARPA2 submarine [M.S. thesis]*, Faculty of the Virginia Polytechnic Institute and State University, Blacksburg, Va, USA, 1999.

[9] M. Sohaib, M. Ayub, S. Bilal, S. Zahir, and M. A. Khan, "Calculation of flows over underwater bodies with hull, sail and appendages," Tech. Rep., National Engineering and Scientific Commission, Islamabad, Pakistan, 2001.

[10] Z. Shang, "CFD investigation of vertical rod bundles of supercritical water-cooled nuclear reactor," *Nuclear Engineering and Design*, vol. 239, no. 11, pp. 2562–2572, 2009.

[11] B. E. Launder and D. B. Spalding, "The numerical computation of turbulent flows," *Computer Methods in Applied Mechanics and Engineering*, vol. 3, no. 2, pp. 269–289, 1974.

[12] ANSYS Inc, *ANSYS Fluent 6.3*, ANSYS, Canonsburg, Pa, USA, 2006.

[13] CD-adapco Inc, *STAR-CD V4.08*, CD-adapco, London, UK, 2008.

[14] OpenCFD Ltd, *OpenFoam Version 1. 7. 1*, OpenCFD, Berkshire, UK, 2010.

[15] M. J. Berger and S. H. Bokhari, "A partitioning strategy for nonuniform problems on multiprocessors," *IEEE Transactions on Computers*, vol. C-36, no. 5, pp. 570–580, 1987.

[16] V. E. Taylor and B. Nour-Omid, "Study of the factorization fill-in for a parallel implementation of the finite element method," *International Journal for Numerical Methods in Engineering*, vol. 37, no. 22, pp. 3809–3823, 1994.

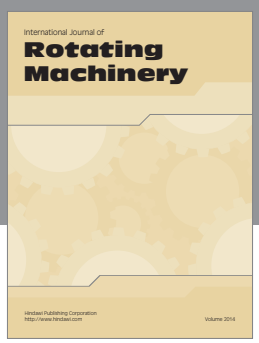
[17] D. Hilbert, "Ueber die stetige Abbildung einer Line auf ein Flachenstuck," *Mathematische Annalen*, vol. 38, no. 3, pp. 459–460, 1891.

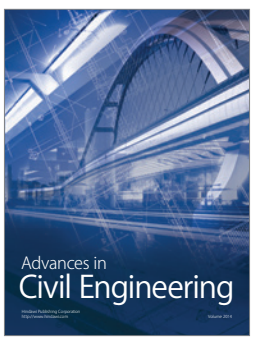
[18] HECToR, 2010, UK National Supercomputing Service, http://www.hector.ac.uk/.

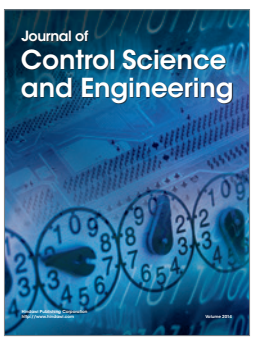
[19] A. Sunderland, C. Moulinec, and Z. Shang, *Developing Code_Saturne for Computing at the Petascale in the PRACE Project*, Frontiers, pp. 59–61, Computational Science and Engineering Department, Science and Technology Facilities Council, Warrington, UK, 2010.

[20] M. S. Fisher, M. Mani, and D. Stookesberry, "Parallel processing with the wind CFD code at Boeing," *Parallel Computing*, vol. 27, no. 4, pp. 441–456, 2001.

[21] Z. Shang, "Performance analysis of large scale parallel CFD computing based on Code_Saturne," *Computer Physics Communications*, vol. 184, pp. 381–386, 2013.







Hindawi
Submit your manuscripts at
http://www.hindawi.com