Computers and Electrical Engineering 67 (2018) 70–84

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Improved grid partitioning algorithms for load-balancing high-order structured aerodynamics simulations☆

Min Xiong[a], Chuanfu Xu[a,*], Xiang Gao[a], Dali Li[a], Dandan Qu[c], Zhenghua Wang[a,b], Xiaogang Deng[a]

[a] *College of Computer, National University of Defense Technology, Changsha, Hunan 410073, China*
[b] *State Key Laboratory of Aerodynamics, Mianyang, Sichuan 621000, China*
[c] *Beijing Aerospace Times Laser Inertial Technology Company, Ltd., Beijing 100143, China*

A R T I C L E I N F O

Article history:
Received 28 June 2017
Revised 9 March 2018
Accepted 9 March 2018

Keywords:
Structured grid
Grid partitioning
Computational fluid dynamics
Load balancing

A B S T R A C T

Grid partitioning is very important to achieving the load balancing of parallel structured aerodynamics simulations. For the large-scale parallel computing, the greedy partitioning (Greedy) algorithm may degrade the load balancing performance and even be infeasible for complex grids. In this paper, we propose two structured partitioning algorithms, the multi-dimensional greedy (MG) algorithm and the grid-first multi-dimensional (GFM) algorithm, to mitigate the issue. The new methods satisfy the specific requirement of stencil width in high-order aerodynamics applications. Moreover, the partitioning rules we presented are also satisfied in the new methods for better connectivity among sub-blocks. We validate and evaluate our methods using three realistic and complex three-dimensional structured configurations. Our experimental results with our in-house high-order aerodynamics software show that MG can achieve a speedup of 1.27x, 1.38x, and 1.21x for the three complex configurations, compared to Greedy, while the number for GFM is 1.36x, 1.24x, and 1.22, respectively.

1. Introduction

Computational fluid dynamics (CFD) is routinely used in the fields of aerodynamics, turbo-machinery, astrophysics, and aircraft design. CFD involves with the solution of the governing equations (Euler or Navier-Stokes), which are solved on a domain divided into a number of grid cells. Basically, there exist two types of grids: structured and unstructured grids, as Fig. 1 shows. Cells in structured grid are quadrilaterals/hexahedra in two/three dimension, and the neighboring cells can be directly identified by their indices [1]. Due to these unique features, structured grids are the most suitable grids to be used for the finite difference CFD applications. In a nutshell, grid partitioning is often used to further split the original multi-blocks into more fine-grained sub-blocks for parallel computing. By doing so, we can obtain better load balancing performance and parallel efficiency. Thus, grid partitioning is a necessary step in the high-order aerodynamics simulation. However, structured partitioning is very difficult, since the features of the structured grids should be maintained in the sub-blocks, and the requirement of stencil width should be satisfied in high-order schemes. Stencil width stands for the

☆ Reviews processed and recommended for publication to the Editor-in-Chief by Associate Editor Dr. Hong Shen.
* Corresponding author.
E-mail address: xuchuanfu@nudt.edu.cn (C. Xu).

https://doi.org/10.1016/j.compeleceng.2018.03.016
0045-7906/© 2018 Elsevier Ltd. All rights reserved.

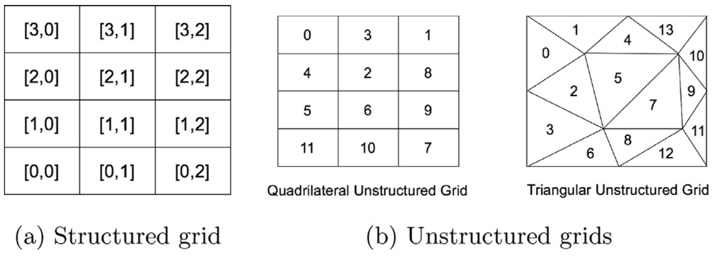


Fig. 1. Examples of two-dimensional structured and unstructured grids. Noted that the quadrilateral grid in (b) is an unstructured grid, even though it looks like a structured one, since the neighboring cells can not be directly identified by the indices.

number of those cell-centroids involved in the computation of residual, gradient, etc [1]. Therefore, the side lengths of sub-blocks from partitioning should be no smaller that of the stencil. As the number of processes grows, the existing partitioning methods may suffer from degraded performance and fail to satisfy the aforementioned restrictions. Hence, it is valuable and significant to improve the structured partitioning methods for large-scale high-order structured CFD simulations.

Aiming at splitting complex structured grids for large-scale high-order finite difference CFD applications, we make the following contributions:

1. The problem of load balancing in CFD simulations is analyzed by quantifying the workloads, and three rules are proposed to guide the partitioning.
2. We propose two structured grid partitioning algorithms, multi-dimensional greedy (MG) and grid-first multi-dimensional (GFM), to solve the potential problems in the traditional greedy (Greedy) algorithm. MG improves Greedy in three aspects and GFM employs a new strategy to generate several balanced sub-blocks from a single original block simultaneously.
3. We partition three realistic and complex configurations on TianHe-2 [2] platform, and test the simulation performance on our in-house high-order CFD software [3]. Theoretically and practically, MG and GFM are able to obtain at least 20% enhancement for all the configurations.

The rest of this paper is organized as follows. Section 2 introduces the related work. Section 3 discusses the partitioning rules and the evaluation metrics. Section 4 and Section 5 describe MG and GFM in detail. Section 6 presents the experimental results and discussions. Finally, conclusions and future work are drawn in Section 7.

2. Related work

In literature, grid partitioning is categorized into two main classes [4]: geometric and connectivity-based methods. Geometric methods rely on a recursive decomposition of the domain, or space filling curves [5]. Connectivity-based methods are graph or hyper-graph algorithms [6,7]. There exist various open-source partitioning tools to supply these methods. For example, graph methods are supported in Metis, ParMetis [8], and Scotch [9], and hyper-graph methods are in PaToh [10], and Zoltan [11]. Geometric methods are implemented in Zoltan2 [12] partitioning packages. However, most of the open-source tools are developed for unstructured grids. For example, the Zoltan library is clearly defined as a collection of data management services for unstructured applications [11]. Therefore, they could not be used for structured partitioning. Only the recursive decomposition methods are suitable for structured partitioning, such as Greedy [13]. Greedy is a one-dimensional partitioning algorithm, which selects the longest dimension and determines a cutting plane based on the computational capabilities of the processes. We used to employ Greedy and have obtained good performance in high-order CFD simulations [14–16]. However, as the number of processes grows, several problems have been found for Greedy, as Fig. 2 shows.

Compared with Greedy, MG and GFM use a multi-partitioning strategy to automatically select a one-dimensional (1D), two-dimensional (2D) or three-dimensional (3D) partitioning. They satisfy the requirement of stencil width flexibly and can extend well with the number of processes. Moreover, the communication is considered in both MG and GFM for better load balancing performance. Therefore, MG and GFM can be used to solve the particular problems suffered by Greedy.

A typical recursive decomposition method can iterate by processes or original blocks, corresponding to the Process First (PF) and Grid First (GF) methods. In PF method like MG, the required volume $vol(p)$ for a process is calculated by distributing the computation amount proportional to the process computational power. In GF method, such as GFM, the required number of processes $p(b)$ for an original block is calculated by distributing the processes proportional to the block volume. $vol(p)$ and $p(b)$ can be represented as:

$$vol(p) = \frac{f(p)}{\sum_{k=1}^{N_p} f(p_k)} \cdot \sum_{i=1}^{N_b} vol_i \tag{1}$$

$$p(b) = \frac{vol_b}{vol(p)} \tag{2}$$

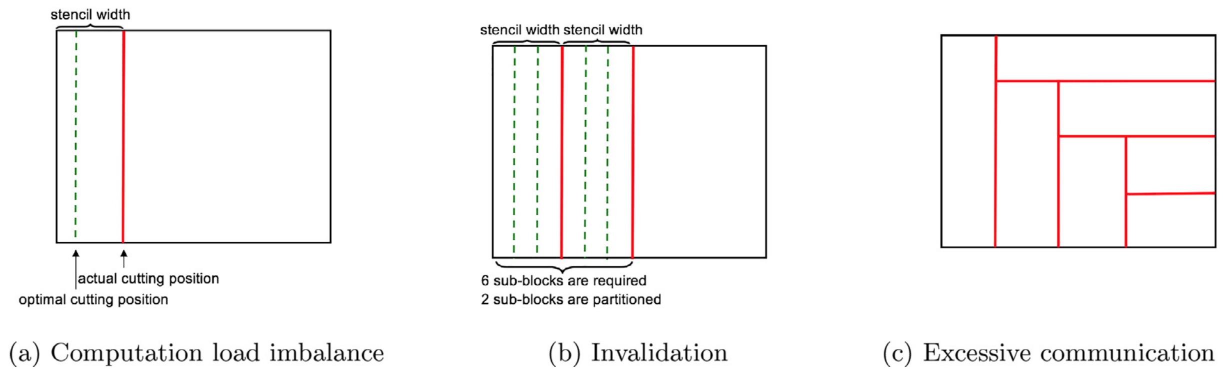

stencil width
actual cutting position
optimal cutting position
(a) Computation load imbalance
stencil width stencil width
6 sub-blocks are required
2 sub-blocks are partitioned
(b) Invalidation
(c) Excessive communication

Fig. 2. Problems of Greedy in high-order large-scale parallel CFD applications. (a) To satisfy the requirement of stencil width, Greedy may modify the calculated cutting planes, leading to computational imbalance. (b) Modifying the cutting planes may reduce the number of processes, and Greedy may become invalid when the processes could not extend any more. (c) Strip sub-blocks in Greedy may lead to excessive communication amounts.

where $f(p)$ is the computational power, N_p and N_b are the numbers of processes and original blocks respectively, and vol_b is the block volume.

3. Analysis of load balancing

3.1. Rules for structured partitioning

Achieving load balancing enables that all the processes accomplish their workloads (both computation and communication) at the same time. For finite difference simulations, the computation and communication workloads relate to the volume and the surface area of the sub-blocks, respectively. Correspondingly, the computation time $T_{comp}(p)$ and communication time $T_{comm}(p)$ for a 3D structured solver can be expressed as:

$$T_{comp}(p) = \frac{\bar{w} \cdot \sum_{s\in p} vol_s}{f(p)} \tag{3}$$

$$T_{comm}(p) = \frac{Num \cdot \sum_{s\in p} comma_s}{r(p)} \tag{4}$$

where the numerators are the computation and communication workloads respectively. $f(p)$ is the floating-point performance. $r(p)$ is the network transfer rate which depends on how the process communicates with the target process. Specifically, $\bar{w}/Num$ is a constant representing the computation/communication workloads per cell. Sub-block s, distributed to process p, has vol_s cells for computation, and $comma_s$ cells for communication. Based on the equations, we can conclude **Rule 1** for structured partitioning: the distributed computation (or communication) workloads of a process should be proportional to its floating-point performance (or network transfer rate).

Moreover, the total inter-process communication overhead should also be minimized in a partitioner, as implemented in PaToH [17] and Zoltan [11]. For a given sub-block volume, the cubic shape has the smallest surface area, or a minimum surface-to-volume ratio, than other shapes. Hence, we can deduce **Rule 2** for structured partitioning: the sub-blocks should have cube/square shape to minimize the total communication overhead.

At last, the good connectivity among sub-blocks can further reduce the communication cost. A good connectivity is a one-to-one mapping that one process communicates with only one neighbor process per surface. Hence, **Rule 3** is: the one-to-one mapping should be maintained during partitioning.

3.2. Evaluation metrics for partitioning

The load balancing performance is applied to evaluate the grid partitioning algorithms in this paper. The performing time of the partitioning algorithm is not concerned, since it runs only once in the preprocessing phase, and its running time is negligible compared with that of the solver. The in-house CFD solver uses 5th-order WCNS-E-5 [3] with stencil width of 11 for spatial discretization. Specifically, the performance metrics include the theoretical **volume/surface percent imbalance** and practical measured data. The volume/surface percent imbalance [18] reflects the gap between the maximum and the average load of the sub-blocks, and can be expressed as:

$$\begin{aligned} \sigma_V &= \frac{Vol_{max} - Vol_{ave}}{Vol_{ave}} \times 100\% \\ \sigma_S &= \frac{Surf_{max} - Surf_{ave}}{Surf_{ave}} \times 100\% \end{aligned} \tag{5}$$

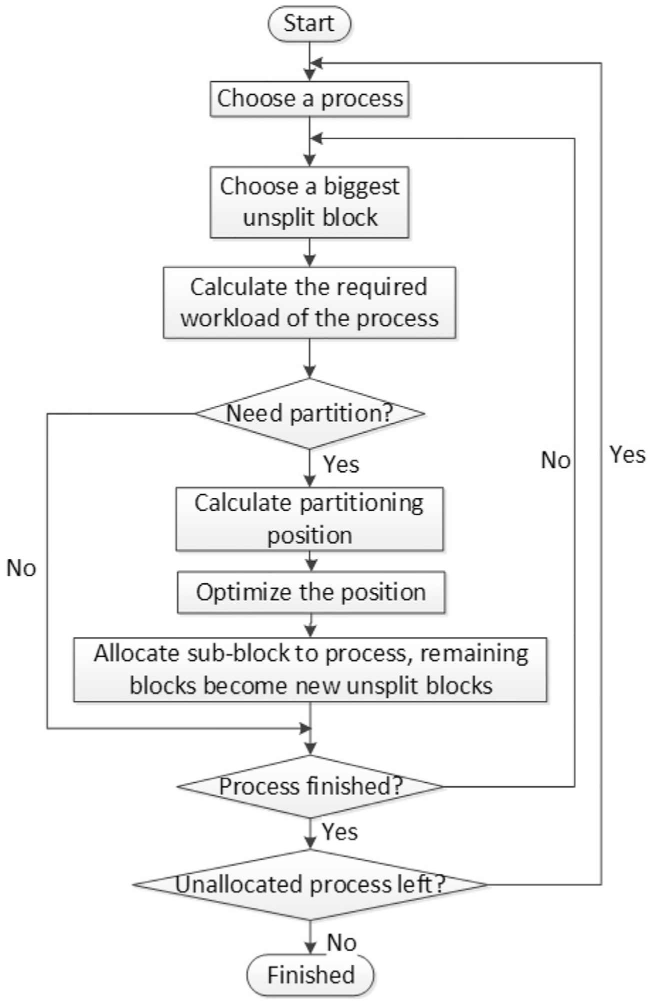

Start
Choose a process
Choose a biggest unsplit block
Calculate the required workload of the process
Need partition?
Yes
No
Calculate partitioning position
Optimize the position
Allocate sub-block to process, remaining blocks become new unsplit blocks
Process finished?
No
Yes
Unallocated process left?
Yes
No
Finished

Fig. 3. The flow chart of MG. As a PF method, MG iterates by processes.

where Vol_{ave}/Vol_{max} and $Surf_{ave}/Surf_{max}$ are the average/maximum computation and communication workloads. Lower values of σ_V and σ_S means better load balancing performance in computation and communication.

As for the practical measured data, the **wall time** and the **communication time** of the CFD solver are measured, applying the sub-blocks generated by three partitioning algorithms, respectively. To avoid contingency, the average values of 10 testing results are used to represent these metrics. Moreover, the **total number of communications** and the **communications percent imbalance** are also measured, considering that the communication is as important as the computation load balancing in large-scale parallel simulations.

4. The MG algorithm

4.1. Partitioning strategy in MG

The flow chart of MG is shown in Fig. 3. The computation complexity of MG is $O(N_p \cdot N_b logN_b)$, equalling to that of Greedy. Within each process iteration, the biggest unsplit block is selected and partitioned when its volume exceeds $vol(p)$, with a distribution shown in Fig. 4. Or it is distributed to the process directly, followed by a modification of $vol(p)$.

We regard the partitioning, which has n cutting plane(s) ($n = 1, 2, 3$) perpendicular to n coordinate axis(es), as a type of the n-dimensional partitioning, as Fig. 5 shows. For simplicity, a n-dimensional partitioning is uniformly represented by a cutting vector, the term in which represents the point number of partitioned side lengths. The cutting vectors, presenting the partitioning types in Fig. 5, are (x_1, 0, 0), (x_2, y_2, 0), and (x_3, y_3, z_3) respectively. The number of non-zero terms in a cutting vector reflects the partitioning type.

The strategy of MG is to calculate the ideal cutting vectors for three partitioning types, and then select the relatively optimal type. Assuming that the biggest unsplit block is ib^*jb^*kb with ($ib \geq jb \geq kb$). For a 3D partitioning, the ideal cutting

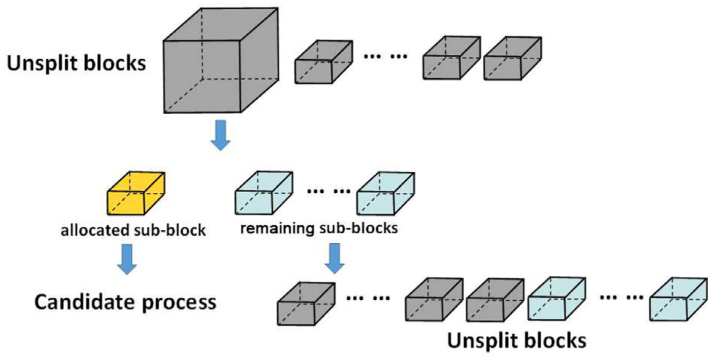


Fig. 4. Distribution after a partitioning in MG. If the partitioning happens, the partitioned sub-block is distributed while the others become new unsplit blocks.

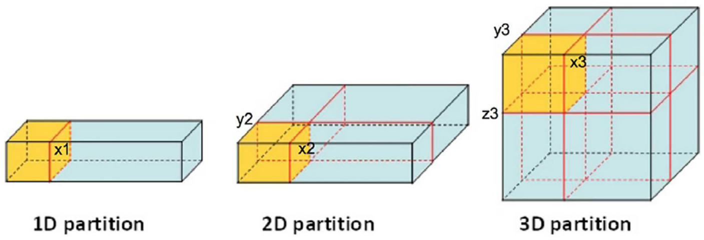


Fig. 5. Three types of partitioning. 1D partitioning has one cutting plane perpendicular to the longest side length. 2D partitioning has two perpendicular to the longest and the second longest side lengths. 3D partitioning has three cutting planes.

Table 1
The partitioning position and the side lengths.

Types	Possible cutting vector	Side lengths
1D	$(\frac{vol(p)}{jb\cdot kb}, 0, 0)$	$(\frac{vol(p)}{jb\cdot kb}, jb, kb)$
2D	$\sqrt{\frac{vol(p)}{kb}} \cdot (1, 1, 0)$	$(\sqrt{\frac{vol(p)}{kb}}, \sqrt{\frac{vol(p)}{kb}}, kb)$
3D	$\sqrt[3]{vol(p)} \cdot (1, 1, 1)$	$\sqrt[3]{vol(p)} \cdot (1, 1, 1)$

vector should be $\sqrt[3]{vol(p)} \cdot (1, 1, 1)$ to obtain a cubic partitioned sub-block, based on **Rule 2**. Similarly, the ideal cutting vector can be obtained for 1D and 2D partitioning, as shown in Tab. 1. These ideal cutting vectors guarantee a cubic partitioned sub-block, and MG further selects one from them to guarantee that the remaining sub-blocks in Fig. 4 will also be cubic.

The evaluation function, $f(type), type = 1, 2, 3$, is used to select the proper cutting vector from the candidate ones in Tab. 1. $f(type)$ reflects the cubic extent of the remaining blocks for a specific partitioning type, and a smaller value of $f(type)$ means a higher cubic extent. Take the 3D partitioning as an example, the side lengths of seven remaining blocks are calculated by $f(3)$. Then, for each remaining block, $f(3)$ recursively calculates the deviation between three side lengths and the cubic root of the block volume. At last, $f(3)$ returns to the maximum deviation among 7*3 deviations. The values of $f(1)$ and $f(2)$ can be obtained similarly, and MG selects a minimum $f(type)$ and its corresponding cutting vector. Quantitatively, $f(type)$ can be expressed as:

$$\begin{aligned} f(1) &= \max_{k=1,\dots,3} |e_{1,k} - \sqrt[3]{vol_1}| \\ f(2) &= \max_{n=1,\dots,3} \max_{k=1,\dots,3} |e_{n,k} - \sqrt[3]{vol_n}| \\ f(3) &= \max_{n=1,2,\dots,7} \max_{k=1,\dots,3} |e_{n,k} - \sqrt[3]{vol_n}| \end{aligned} \tag{6}$$

where n is the number of remaining blocks for a specific type. vol_n is the volume of the nth remaining block, and $e_{n,\,k}$ represents the $k^{th}(k = 1, 2, 3)$ side length of the nth remaining block.

4.2. Optimizations

Besides the automatic multi-dimensional partitioning, MG is further optimized in two aspects of communication and computation load balancing, respectively. One optimization is to modify the cutting plane adaptively for a better connectiv-

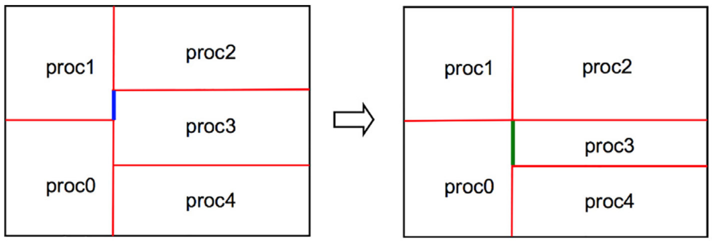


Fig. 6. The modification of the cutting plane. The cutting plane of proc2 and proc3 is moved to an existing plane of proc0 and proc1 for one-to-one mapping. The modification reduces the number of communications for both proc1 and proc3.

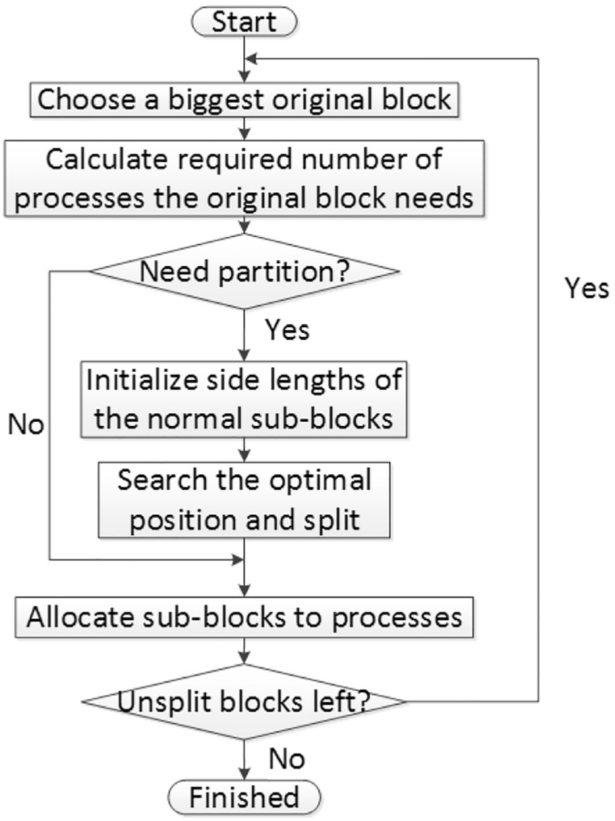

Start
Choose a biggest original block
Calculate required number of processes the original block needs
Need partition?
Yes
No
Initialize side lengths of the normal sub-blocks
Search the optimal position and split
Allocate sub-blocks to processes
Unsplit blocks left?
Yes
No
Finished

Fig. 7. The flow chart of GFM. As a GF method, GFM iterates by original blocks.

ity, based on **Rule 3**. In MG, the new cutting planes, close enough to the existing cutting planes, are moved to the existing ones for one-to-one mappings, as Fig. 6 shows. During the modification, the requirement of stencil is checked to be satisfied. The modification can improve the connectivity in MG, but could not totally solve the problem of one-to-one mapping. The other optimization is to solve the rounding problem, caused by division or cubic root operations for the cutting vectors. In MG, the terms in the cutting vectors are rounded off in order to become integrals. Specifically, 2^n rounding combinations for a $n-$dimensional partitioning are calculated to select a best one, which has the minimum volume deviation from the ideal volume. For example, for a 3D partitioning with the cutting vector (15.5,15.5,15.5) and an ideal volume of 3723, the combination (16,16,15) has the minimum load imbalance of 3.14%, while the combination (16,16,16) has the maximum one of 10.02%, among 8 possible combinations. In practice, the rounding problem and the plane modification are dealt together, and 2^n combinations are not redundant.

In conclusion, the MG algorithm is able to partition multiple dimensions automatically, modify the cutting planes adaptively, and solve the rounding problem. However, MG is essentially a variant of Greedy, which cannot totally solve the problems presented in Fig. 2. For example, the remaining sub-blocks become new unsplit blocks in MG, leading to many small chunks allocated to the last several processes. As the number of processes rises, the possible small chunks may deteriorate the communication performance. For those larger-scale situations, we further propose the GFM algorithm to remedy the possible deficiency of MG.

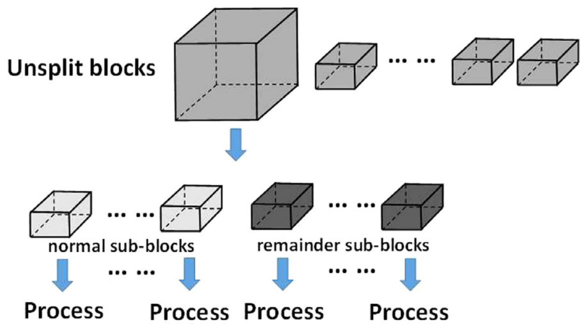


Fig. 8. Distribution in GFM. In GFM, all the sub-blocks are distributed.

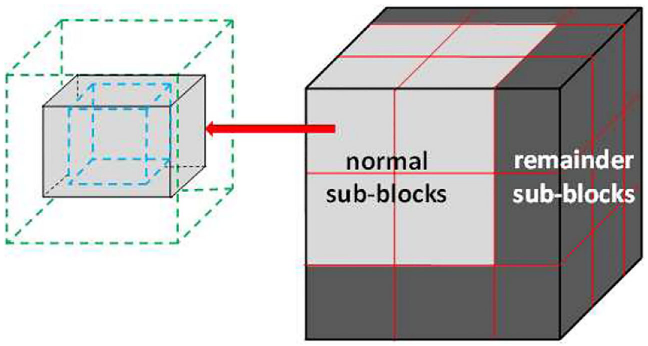


Fig. 9. The searching scope for the normal sub-blocks. The three side lengths of a normal sub-block change within the scope of $[\sqrt[3]{vol(p)} - sten, \sqrt[3]{vol(p)} + sten]$. Hence, the possible shape of a normal sub-block will change between the two dashed cubes.

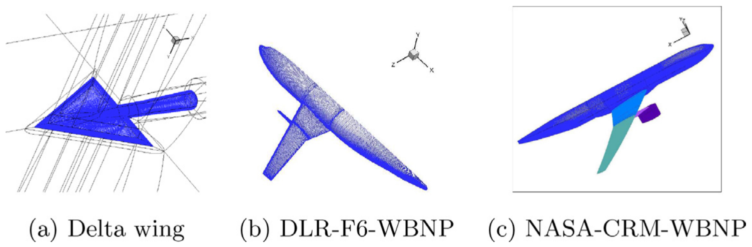


Fig. 10. 3D realistic and complex configurations.

5. The GFM algorithm

5.1. Characteristic of GFM

The GFM algorithm is proposed further to handle the problems of MG especially for large-scale cases. Its flow chart is shown in Fig. 7. The computation complexity of GFM is $O(N_p \cdot N_b + N_b logN_b)$.

Within each block iteration of GFM, the biggest original block is selected and $p(b)$ is calculated. Then the selected original block is totally partitioned into $p(b)$ pieces, and each piece is distributed to a process separately, as shown in Fig. 8. However, $p(b)$ cannot be arbitrary and may be automatically modified by the partitioning strategy. For example, if the calculated value of $p(b)$ is 7.5, it may be modified to 8 for a 3D partitioning with a cutting vector of (2, 2, 2). For large-scale simulations, two assumptions are reasonable in GFM: (1) the modification of $p(b)$ has little influence on the total performance; (2) the original block, whose volume is smaller than $vol(p)$, can be distributed directly to a process with small influence on the total performance of load balancing.

In GFM, a single original block is split into two kinds of sub-blocks, the normal sub-blocks and the remainder ones. There must be several normal sub-blocks, whose side lengths are calculated based on **Rule 2**. When the side lengths of the original block is divided by those of the normal sub-blocks, the remainder sub-blocks are generated by the division remainders, as Fig. 9 shows. The unique characteristic of GFM is that all the normal and the remainder sub-blocks are distributed to the processes immediately after their generation.

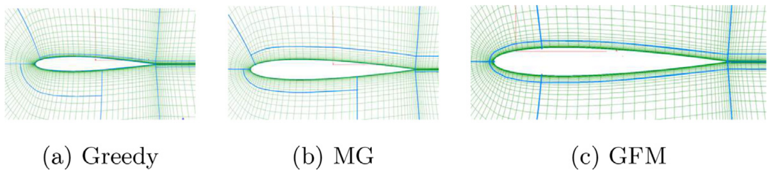


Fig. 11. Sub-blocks of NACA 0012.

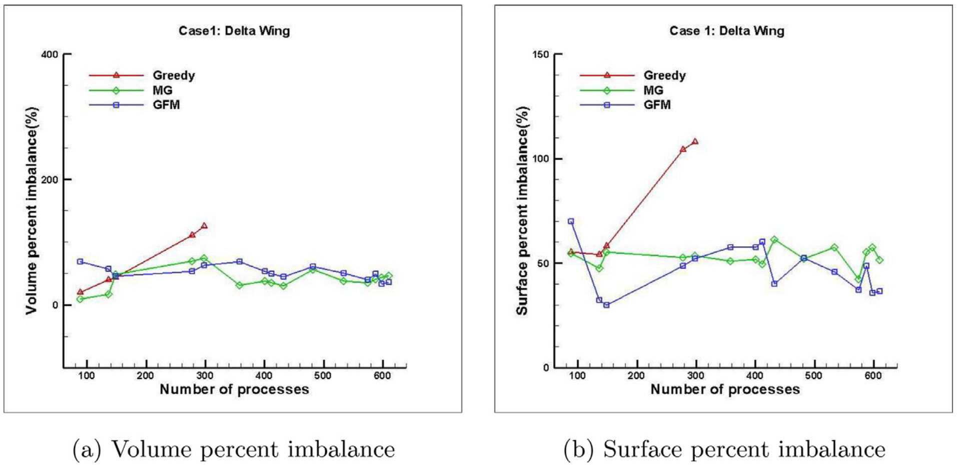


Fig. 12. Theoretical metrics of delta wing.

5.2. Partitioning strategy in GFM

To partition an original block, the cutting vector of the normal sub-blocks should be determined in GFM. Specifically, the cutting vector is firstly limited within a suitable searching scope, then a relatively optimal one is selected by an evaluation function among the possible candidates. A suitable searching scope should be large enough to find a relatively optimal vector, but should not be too large for efficiency. Hence, we set the searching scope to the value of the stencil width *sten* for two reasons: the stencil width of a high-order scheme is basically large enough, and the value of the optimal cutting vector is usually close to $\sqrt[3]{vol(p)} \cdot (1, 1, 1)$ due to **Rule 2**. Then, the possible cutting vector is limited between $(\sqrt[3]{vol(p)} - sten) \cdot (1, 1, 1)$ and $(\sqrt[3]{vol(p)} + sten) \cdot (1, 1, 1)$, and limited by the requirement of stencil width at the same time. Correspondingly, the normal sub-block is changeable between two dashed cubes illustrated in Fig. 9.

The evaluation function $g(v)$ returns to the maximum deviations between the volume of the sub-blocks (normal or remainder) and the required volume of a process, for a specific cutting vector in the searching scope. The function could be represented as:

$$g(v) = \max_{s_i} |vol_{s_i} - vol_p| \tag{7}$$

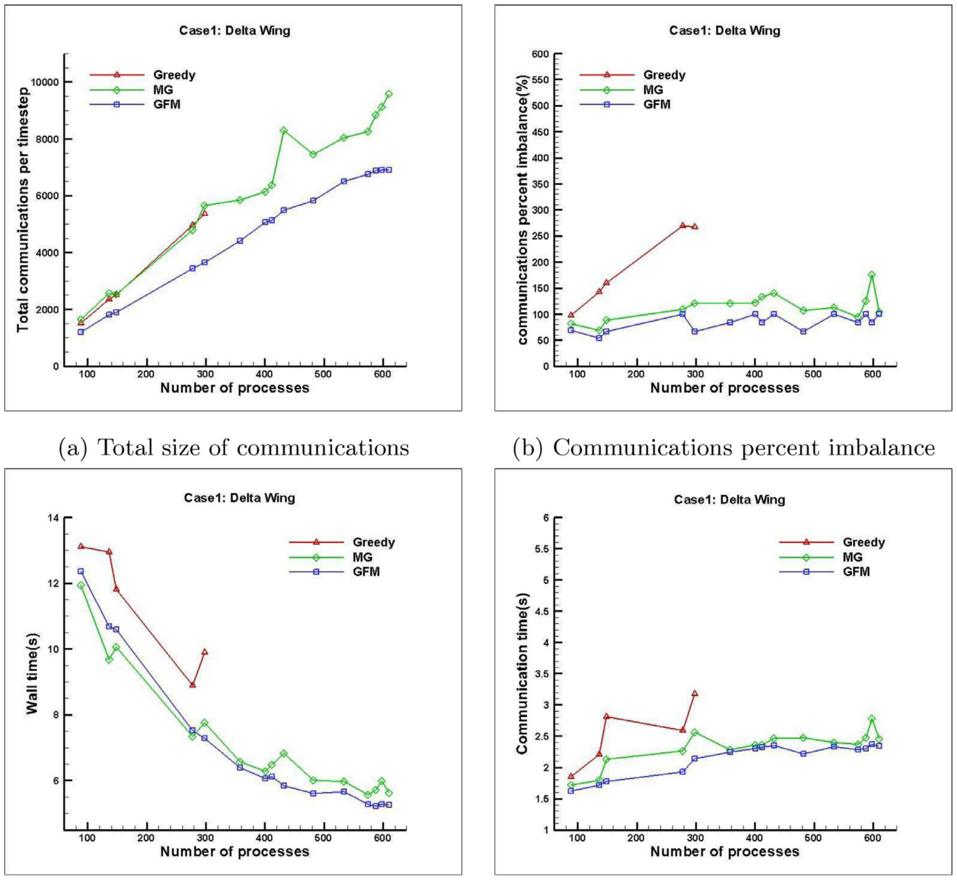
where v is a possible cutting vector in the searching scope. s_i is a normal or remainder sub-block when v is employed. vol_{s_i} is the volume of the sub-block, and vol_p is the required volume of a process. The relatively optimal cutting vector is selected by finding the minimum value of $g(v)$.

The partitioning strategy in GFM implicitly settles the rounding problem or connectivity problem in MG. On one hand, the rounding problem is tackled implicitly since all the possible vectors have already been compared by $g(v)$. On the other hand, the one-to-one mapping is naturally satisfied within the same original block. Hence, only the one-to-one mappings among different original blocks should be handled by modifying the cutting vectors.

In conclusion, GFM can solve the problems of MG in larger-scale cases. GFM partitions an original block into several similar and balanced sub-blocks simultaneously. All the generated sub-blocks are distributed to the corresponding processes. In GFM, the requirement of stencil width and the one-to-one mapping are satisfied naturally. Moreover, GFM can produce high-quality sub-blocks with regularity, similarity and good connectivity, making it more suitable for extremely large-scale cases.

6. Results and discussions

In this section, a simple 2D NACA 0012 airfoil (100 thousand cells) and three complex 3D configurations are used to evaluate the performance of Greedy, MG and GFM theoretically and practically. The 3D configurations are delta-wing [19] (15*M*



(a) Total size of communications (b) Communications percent imbalance

(c) Total running time (d) Total communication time

Fig. 13. Practical communication and time metrics of delta wing.

cells), DLR-F6-WBNP [20] (108*M* cells), and NASA-CRM-WBNP [21] (218*M* cells), as Fig. 10 shows. Considering each process should have 10,000 –500,000 grid cells for efficiency, the number of processes is 1600 at most in this paper. For all the considered metrics, the smaller the better.

6.1. 2D NACA 0012 airfoil

For the NACA 0012 airfoil, the partitioning results of 12 sub-blocks are shown in Fig. 11 to compare the connectivity and one-to-one mapping. All the sub-blocks generated by GFM satisfy the one-to-one mapping, while some generated by Greedy and MG cannot. Hence, GFM shows better performance in connectivity and communication, and the sub-blocks it generated communicate only once per surface. This case demonstrates that GFM can generate high-qualified sub-blocks with less neighbors and better connectivity.

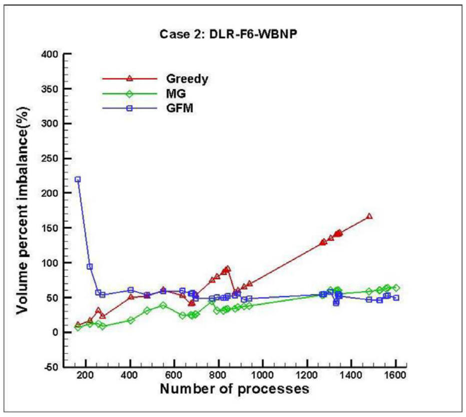
6.2. 3D delta wing

6.2.1. Theoretical metrics

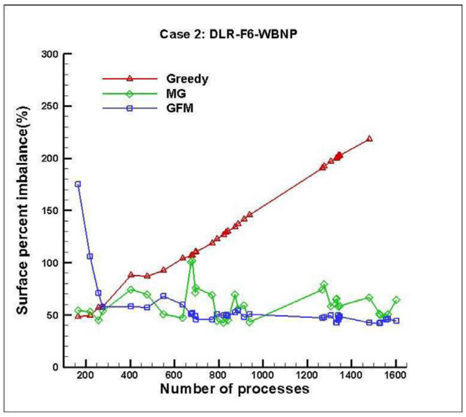
The volume/surface percent imbalance of delta wing is shown in Fig. 12. Greedy becomes invalid when the number of processes exceeds 300. Because Greedy has to modify the cutting planes to satisfy the requirement of stencil width, leading to some idle processes with no workloads. For valid cases, Greedy tends to deteriorate in communication load balancing due to excessive communication amounts. Both MG and GFM are capable of scaling to over 300 processes due to their abilities of multi-partitioning. MG has a more stable behavior in communication, since the biggest original block in this case is not large enough for GFM to show its advantage on communication.

6.2.2. Practical metrics

The total number of communications, as well as the communications percent imbalance, within a time-step is collected when the sub-blocks are utilized in the in-house solver, as Fig. 13 (a) and (b) show. It is obvious that GFM has the best

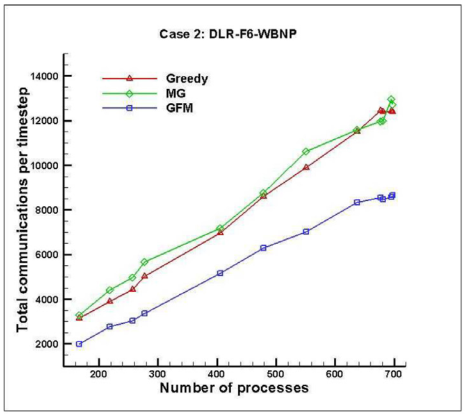


(a) Volume percent imbalance

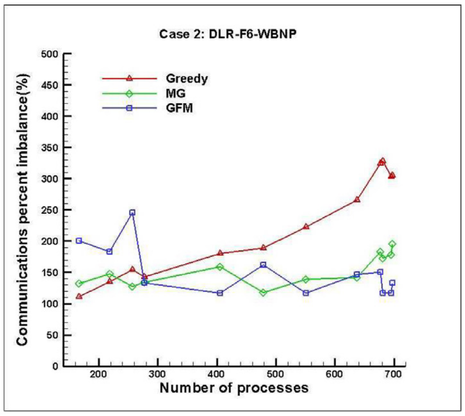


(b) Surface percent imbalance

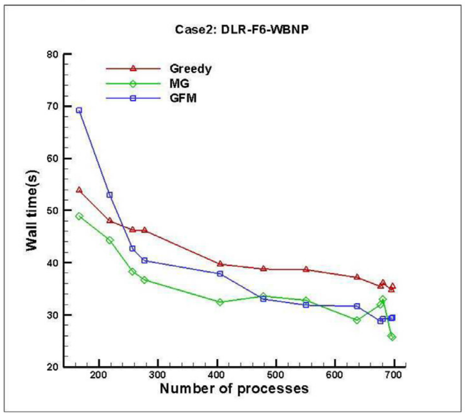
Fig. 14. Theoretical metrics of DLR-F6-WBNP.



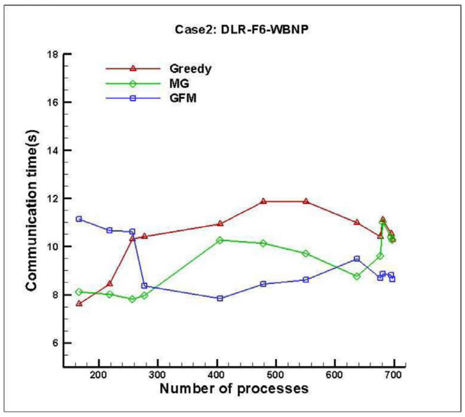
(a) Total size of communications



(b) Communications percent imbalance



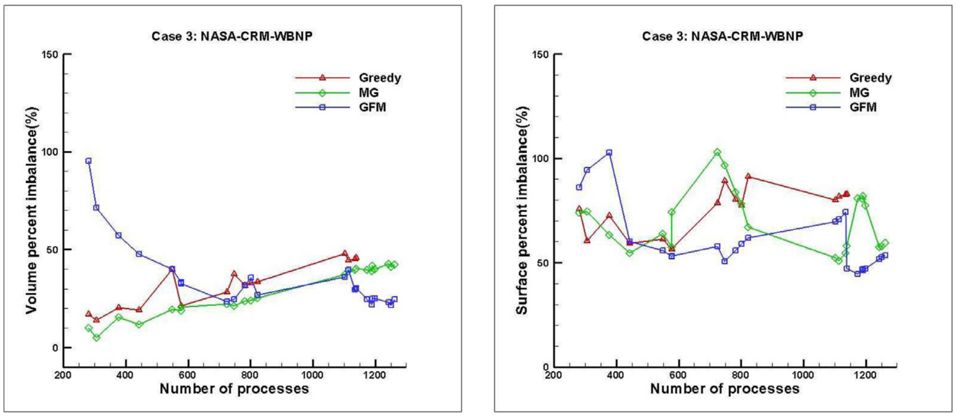
(c) Total running time



(d) Total communication time

Fig. 15. Practical communication and time metrics of DLR-F6-WBNP.

performance in communication due to its advantage on the one-to-one mapping. The total communication size of MG is not stable, and it increases unstably as the number of processes grows due to the possible small chunks. The total running time and the communication time within 10 time-steps of the solver, utilizing the partitioning results, are collected in Fig. 13 (c) and (d). As the number of processes increases, GFM has an advantage on these time metrics, which is possibly attributed to the minimum number of communications. The communication influences little in this case and MG has good total performance when the number of processes exceeds 500, even though it has a large size of communication.



(a) Volume percent imbalance (b) Surface percent imbalance

Fig. 16. Theoretical metrics of NASA-CRM-WBNP.

In summary, for delta wing, Greedy can not scale to over 300 processes, while both MG and GFM scale and behave well theoretically and practically. For communication, GFM is a little better than MG especially when the number of processes exceeds 300.

6.3. 3D DLR-F6-WBNP

6.3.1. Theoretical metrics

We partition DLR-F6-WBNP on 200 to 1600 processes for theoretical tests, while 200 to 700 for practical tests due to the limited available sources. The theoretical metrics for DLR-F6-WBNP are shown in Fig. 14. Greedy becomes invalid when the number of processes exceeds 1500, and the communication performance keeps decreasing with a large gradient, so is the computation load balancing performance when the number of processes exceeds 900. Both MG and GFM behave well as the number of processes grows. GFM is not suitable with a small number of processes since its simplifying assumptions are hard to maintain. MG is unstable in communication due to its possible small chunks.

6.3.2. Practical metrics

Limited by the accessible sources, the number of processes scales from 200 to 700 in this configuration. As Fig. 15 (a) and (b) show, the communications percent imbalance conforms to the surface percent imbalance basically. Due to the small chunks, MG behaves as similar as Greedy in total number of communications. GFM has the best performance in the total number of communications due to its advantage on the one-to-one mapping. The wall time and communication time of DLR-F6-WBNP are shown in Fig. 15 (c) and (d). Greedy is applicable in this scope of processes with the poorest performance, and the turning point of GFM is very obvious. The computation load balancing is the main influence of the total performance in this case, so MG still behaves well even though it communicates many times. The practical time metrics basically conform to the theoretical metrics.

In a word, for DLR-F6-WBNP, Greedy is invalid theoretically when the number of processes exceeds 1500. GFM only behaves well when the number of processes is large enough due to its simplifying assumptions. MG behaves unstable in communication and requires massive number of communications.

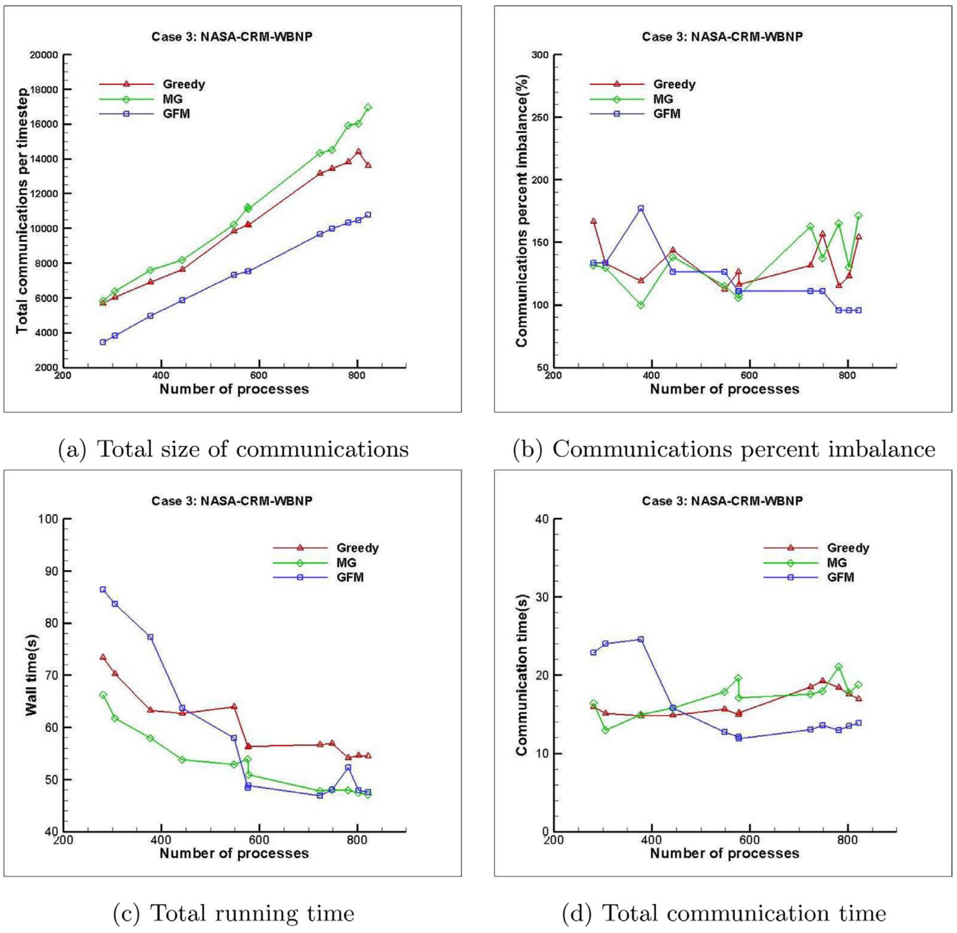
6.4. 3D NASA-CRM-WBNP

6.4.1. Theoretical metrics

NASA-CRM-WBNP is supplied by NASA in 2016 on the 6th AIAA AVIATION conference [21]. We partition it on 200 to 1300 processes for theoretical tests, and 200 to 800 for practical tests. The volume/surface percent imbalance for NASA-CRM-WBNP is shown in Fig. 16. Greedy becomes invalid when the number of processes exceeds 1100. GFM still behaves badly when the number of processes is small, and MG behaves unstable in communication. As for the theoretical metrics, GFM has relatively optimal performance in communication and computation load balancing when the number of processes is large enough.

6.4.2. Practical metrics

Fig. 17(a) and (b) show the total communications per time-step and the communications percent imbalance of NASA-CRM-WBNP. The number of processes scales from 200 to 800 for the reason mentioned before. GFM is able to obtain the best performance in these metrics when the number of processes is large enough. The performance in communication of MG are



(a) Total size of communications (b) Communications percent imbalance (c) Total running time (d) Total communication time

Fig. 17. Practical communication and time metrics of NASA-CRM-WBNP .

Table 2
The speedup of MG compared with Greedy.

Configurations	Theoretic speedup		Practical speedup		
	Comp	Comm	Comp	Comm	Total
Delta wing	2.41	2.02	1.36	1.24	1.27
DLR-F6-WBNP	2.90	2.21	1.63	1.32	1.38
NASA-CRM-WBNP	2.07	1.36	1.37	1.17	1.21

even worse than that of Greedy in this scope of processes, since the small chunks and one-to-one mapping are very severe in this very complex configuration. The wall time and communication time of NASA-CRM-WBNP are shown in Fig. 17 (c) and (d). MG performs well in the running time, since the total performance is also influenced by many other aspects. Greedy is applicable in this scope of process, but has a longest running time due to the poor computation performance. The turning point of GFM is obvious since it only performs well when the number of processes is large enough. The practical time metrics basically conform to the theoretical ones.

In summary, for NASA-CRM-WBNP, Greedy is invalid theoretically when the number of processes exceeds 1100. MG is preferable when the number of processes is smaller than 400. Both MG and GFM can extend to more than 400 processes, and GFM will be a better choice for a larger number of processes.

6.5. Conclusions for experiments

A 2D NACA 0012 airfoil and three 3D realistic and complex configurations are used to test the load balancing performance of Greedy, MG and GFM. Both MG and GFM are capable of solving the invalidity problem in Greedy, and obtain better load balancing performance theoretically and practically. The speedup of MG and GFM is shown in Tables 2 and 3, respectively. The theoretical speedup is calculated by the volume/surface percent imbalance, and the practical one is calculated by the

Table 3
The speedup of GFM compared with Greedy.

Configurations	Theoretic speedup		Practical speedup		
	Comp	Comm	Comp	Comm	Total
Delta wing	2.08	2.14	1.31	1.48	1.36
DLR-F6-WBNP	1.11	2.42	1.25	1.40	1.24
NASA-CRM-WBNP	1.52	1.77	1.14	1.42	1.22

corresponding running time. All of them are measured within the valid process scopes of Greedy (300 for delta wing, 700 for DLR-F6-WBNP, and 800 for NASA-CRM-WBNP). The practical speedup is lower than the theoretical one since other factors may also influence the total performance during the running procedure. In conclusion, for realistic and complex configurations, both MG and GFM are capable of cutting the configuration for a large range of the number of processes and obtain well balanced workloads. GFM is more suitable than MG with extremely large number of processes.

7. Conclusion and future work

We propose two structured partitioning algorithms, namely the multi-dimensional greedy (MG) algorithm and the grid-first multi-dimensional (GFM) algorithm. They solve the problems of the common used traditional greedy (Greedy) algorithm and obtain better load balancing performance for large-scale high-order structured aerodynamics simulations. Specifically, MG determines the required workload for a process by distributing the computation amount proportional to the process computational power. The multi-dimensional partitioning is obtained by selecting the cutting planes with the best load balancing performance. The requirement of stencil is also satisfied during each partitioning. Moreover, both the computational and the communicational load balancing are optimized based on the partitioning rules for better performance. GFM determines the required number of processes for an original block by distributing the processes proportional to the block volume. GFM compares the qualities of the possible sub-block shapes within a searching scope. Several sub-blocks can be generated simultaneously during each partitioning procedure. Charactered with balanced and regularly ordered sub-blocks, as well as a good connectivity, GFM is more suitable than MG for extremely large-scale simulations.

The partitioning methods are used for three realistic and complex configurations. The new methods can extend to a large number of processes without destroying the load balancing performance. Moreover, MG and GFM can also obtain up to 1.27x/1.36x, 1.38x/1.24x, and 1.21x/1.22x enhancements for the three configurations, compared with the Greedy algorithm.

In the future, GFM will be ported on a heterogeneous platform as an inter-node partitioner. Then, the number of processes will be scaled to a larger value by a collaborating computation on TianHe-2. Moreover, MG and GFM should be employed in more complex configurations in practice.

Acknowledgment

This paper was supported by the National Natural Science Foundation of China under Grant No. 11502296, No. 61772542 and No. 61561146395; the Foundation of National University of Defense Technology under Grant No. ZDYYJCYJ20140101; the Open Research Program of China State Key Laboratory of Aerodynamics under Grant No. SKLA20160104; and the Defense Industrial Technology Development Program under Grant No.C1520110002.

References

[1] Blazek J. Computational fluid dynamics: principles and applications. Butterworth-Heinemann; 2015.
[2] Top500 list. http://www.top500.org/.
[3] Deng X, Zhang H. Developing high-order weighted compact nonlinear schemes. J Comput Phys 2000;165(1):22–44.
[4] Saule E, Baş EÖ, Çatalyürek ÜV. Load-balancing spatially located computations using rectangular partitions. J Parallel Distrib Comput 2012;72(10):1201–14.
[5] Aftosmis MJ, Berger MJ, Murman SM. Applications of space-filling curves to cartesian methods for cfd. AIAA Pap. 2004;1232:2004.
[6] Deveci M, Kaya K, Uçar B, Çatalyürek ÜV. Hypergraph partitioning for multiple communication cost metrics: model and methods. J Parallel Distrib Comput 2015;77:69–83.
[7] Acer S, Selvitopi O, Aykanat C. Improving performance of sparse matrix dense matrix multiplication on large-scale parallel systems. Parallel Comput 2016;59:71–96.
[8] Karypis G, Schloegel K, Kumar V. Parmetis: Parallel graph partitioning and sparse matrix ordering library, version 3.1[r], Version 10,. Dept of Computer Science, University of Minnesota; 2003.
[9] Pellegrini F, Roman J. Scotch: A software package for static mapping by dual recursive bipartitioning of process and architecture graphs. In: High-performance computing and networking. Springer; 1996. p. 493–8.
[10] Çatalyürek Ü, Aykanat C. Patoh (partitioning tool for hypergraphs). In: Encyclopedia of parallel computing. Springer; 2011. p. 1479–87.
[11] Devine K, Boman E, Heaphy R, Hendrickson B, Vaughan C. Zoltan data management service for parallel dynamic applications. Comput. Sci. Eng. 2002;4(2):90–7.
[12] Boman EG, Devine KD, Leung VJ, Rajamanickam S, Riesen LA, Deveci M, et al. Zoltan2: Next-generation combinatorial toolkit. Tech. Rep.. Sandia National Laboratories; 2012.
[13] Ytterström A. A tool for partitioning structured multiblock meshes for parallel computational mechanics. Int J Supercomput Appl High Perform Comput 1997;11(4):336–43.

[14] Xu C, Deng X, Zhang L, Fang J, Wang G, Jiang Y, et al. Collaborating cpu and gpu for large-scale high-order cfd simulations with complex grids on the tianhe-1a supercomputer. J Comput Phys 2014;278:275–97.
[15] Wang Y, Zhang L, Liu W, Che Y, Xu C, Wang Z, et al. Efficient parallel implementation of large scale 3d structured grid cfd applications on the tianhe-1a supercomputer. Comput Fluids 2013;80:244–50.
[16] Che Y, Zhang L, Xu C, Wang Y, Liu W, Wang Z. Optimization of a parallel cfd code and its performance evaluation on tianhe-1a. Comput Inf 2015;33(6):1377–99.
[17] Catalyürek UV, Aykanat C. Patoh: a multilevel hypergraph partitioning tool, version 3.0, 6533. Bilkent University, Department of Computer Engineering, Ankara; 1999.
[18] Pearce O, Gamblin T, De Supinski BR, Schulz M, Amato NM. Quantifying the effectiveness of load balance algorithms. In: Proceedings of the 26th ACM international conference on Supercomputing. ACM; 2012. p. 185–94.
[19] Delta wing model. http://vfe2.dlr.de/Site/index.php?id=18.
[20] Dpw-3. https://aiaa-dpw.larc.nasa.gov/Workshop3/workshop3.html.
[21] Dpw-6. https://aiaa-dpw.larc.nasa.gov/.

Min Xiong, born in 1990. PhD candidate in National University of Defense Technology. Her main research interests include high performance computing and application.

Chuanfu Xu, born in 1980. PhD and associate professor in National University of Defense Technology. Member of China Computer Federation. His main research interests include high performance computing and application.

Xiang Gao, born in 1990. PhD candidate in National University of Defense Technology. His main research interest is high performance computing.

Dali Li, born in 1987. PhD and assistant professor in National University of Defense Technology. His main research interest is high performance computing.

Dandan Qu, born in 1981. PhD in Beijing Institute of Technology. Engineer in Beijing Aerospace Times Laser Inertial Technology Company, Ltd.

Zhenghua Wang, born in 1962. PhD and professor in National University of Defense Technology. His research interests are large-scale engineering and scientific computing, and big data processing.

Xiaogang Deng, born in 1960. PhD and professor in National University of Defense Technology. His main research interests are computational aerodynamics, and research and application of high order precision format.