Computers and Structures 154 (2015) 177–191

Contents lists available at ScienceDirect

Computers and Structures

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

Fine-grained parallel algorithm for unstructured surface mesh generation

Dawei Zhao [a], Jianjun Chen [a,b,*], Yao Zheng [a], Zhengge Huang [a], Jianjing Zheng [a]

[a] Center for Engineering and Scientific Computation, and School of Aeronautics and Astronautics, Zhejiang University, Hangzhou 310027, China
[b] Zienkiewicz Centre for Computational Engineering, Swansea University, Swansea SA2 8PP, Wales, UK

ARTICLE INFO

Article history:
Received 14 January 2014
Accepted 4 April 2015
Available online 21 April 2015

Keywords:
Mesh generation
Parallel algorithm
Surface mesh
Domain decomposition
Graph partitioning

ABSTRACT

A parallel surface meshing algorithm is proposed by exploiting the parallelism within the meshing process of a surface, which is more efficient than the conventional scheme that meshes surfaces individually. One integral part is the domain decomposition approach adopted, which ensures no small inter-domain angles; therefore, the parallel meshing procedure can fix the inter-domain boundary without compromising mesh quality. Combining the parallel surface mesher with the parallel tetrahedral mesher and improver developed previously, a parallel preprocessing pipeline for large-scale simulations is set up. It only consumes minutes to prepare a mesh containing hundreds of millions of elements.

1. Introduction

Thanks to the rapid advance of high performance computing (HPC) technologies, parallel machines are more and more cost-effective. In both the academic and industry communities, various CFD codes have been parallelised for many years to exploit the huge computing power of parallel machines efficiently. It was reported that some academic parallel CFD codes were able to exploit hundreds of thousands of computer cores to efficiently solve a problem containing billions of elements [1]. In the aerodynamics industry, CFD simulations usually involve thousands of computer cores, and the simulation time ranges from hours to days.

However, the wall-clock time to finish a complex simulation is far more than that consumed by a parallel simulation code, large portion of which accounts for mesh generation. For instance, it usually takes weeks or more to prepare a block-structured mesh in the exterior flow simulation of a complete aircraft model, even when the engineer is an expert user of a state-of-the-art commercial or in-house meshing tool [2]. Unstructured mesh generation does not require a painful process to decompose a complex domain into blocks; hence, it is more automatic. However, the process of generating an unstructured mesh composed of ten million mesh points may consume about an hour CPU time if executed sequentially [3–6]. In practice, when the input geometry is very complex, or a high standard is set for mesh quality, mesh generation is a trial-and-error process, and may need to be repeated many times. Therefore, the wall-clock time for preparing a large-scale unstructured mesh is usually comparable with or more than the time for conducting parallel simulations.

Parallelisation is a feasible way to speed up the meshing process. The studies on parallel mesh generation have started since the 1990s. In the early stage, the principal motivation was to overcome the memory bottleneck to generate a large-scale mesh [7,8]. Nowadays, a 64-bit desktop computer may be configured with a large amount of memory at an affordable price. Hence, the memory issue is no longer prominent, although to break it is still beneficial. However, the other motivation of developing parallel meshers, i.e., to overcome the time bottleneck of generating a large-scale mesh, is still meaningful. To fully exploit the computing power of ever emerging parallel computers, a simulation environment where both the simulation code and its pre and post processing codes are parallelised is now highly demanded [6,9].

For a typical 3D simulation, surface mesh generation, volume mesh generation, and volume mesh improvement are three major steps involved in the preparation of an unstructured mesh. We have presented parallel schemes for volume mesh generation and improvement recently [10]. In this study, we attempt to parallelise the surface meshing step in order to develop a complete parallel pipeline for the generation of large-scale unstructured meshes.

* Corresponding author at: Center for Engineering and Scientific Computation, and School of Aeronautics and Astronautics, Zhejiang University, Hangzhou 310027, China. Tel.: +86 571 8795 1883; fax: +86 571 8795 3167.
E-mail address: chenjj@zju.edu.cn (J. Chen).

http://dx.doi.org/10.1016/j.compstruc.2015.04.004
0045-7949/

Parallel volume meshing study abounds in the literature, whereas its surface counterpart was discussed rarely. In [11], a simple scheme was proposed for CAD models composed of many surfaces. The sequential process meshes each surface individually. So, considering the process of meshing each surface as a task, the meshing tasks of all surfaces are distributed on different processors and conducted in parallel. This scheme is essentially not scalable and its efficiency highly depends on geometric natures of the surface model. If the number of surfaces (M) is far more than the number of computer cores (N), and the meshing time of each surface (referred to as the load of a surface hereafter) is roughly equal, a high efficiency is possible; otherwise, the efficiency may be very low.

In this study, an enhanced algorithm is proposed to overcome the bottleneck of the above scheme in terms of scalability and efficiency. In the enhanced algorithm, a surface with large loads is split into many smaller subdomains, and these subdomains are meshed individually. Therefore, the proposed algorithm is *fine-grained* because it takes advantage of the parallelism within the meshing process of a surface. The total number of subdomains is determined in the run time and scalable with N rather than being fixed as M. Therefore, the load difference between different meshing tasks no longer depends on the surface loads, but is controllable by the user. By subdividing large surfaces properly, it is possible to avoid the case where the time of meshing a single surface dominates the overall meshing time. Moreover, each subdomain has a similar representation like the original surface, and the sequential mesher can handle it without modification. This full reuse capability is very desirable because it minimises the cost of parallelising a sequential code, in particular if the sequential code may need constant improvement.

The decomposition of a surface model is essential to the success of the proposed parallel scheme. The domain decomposition tools that prevail in parallel solutions of partial differential equations (PDEs) mainly focus on reducing load imbalances and interface communications, and are incompetent to avoid the generation of a poorly shaped inter-domain boundary, which is harmful in the context of parallel mesh generation:

(1) Some mesh generation schemes require that the boundary angles be within certain bounds in order to guarantee the termination and to achieve a provably good element quality. When these schemes are employed on subdomain mesh generation, the artificial features such as small angles are prohibitive [12–14].
(2) A poorly shaped boundary is troublesome because low-quality elements may form in its neighbourhood. To prepare a qualified mesh for simulations, a time-consuming step to improve these elements is indispensable, e.g., assimilating the submeshes (if the memory allows) and then improving the entire mesh sequentially [15,16], or improving the distributed mesh concurrently at the cost of inter-processor operations [17].

In this study, a novel approach for domain decomposition that features its ability to produce an inter-domain boundary having good geometric properties is examined. The domain is filled up with a non-overlapping mesh first (referred to as a background mesh hereafter). This mesh is generated under a coarsened size map; hence, it is much coarser than the final mesh. Instead of directly sending the background mesh to a graph partitioner [18,19], an intermediate procedure is proposed to simplify the mesh based on some local operations defined on the dual graphs of the mesh. Partitioning the simplified mesh rather than the initial mesh produces a distributed mesh that not only fulfils the dual goals of load balancing and minimisation of communications, but also contains desirable geometric properties in the inter-domain boundary. Therefore, the subsequent parallel meshing procedure can fix the inter-domain boundary without compromising the quality of elements around this boundary, and any sequential mesher that respects domain boundary can be reused. In the mean time, because the meshing procedure involves no communications, its parallel efficiency is very high.

To demonstrate the effectiveness of the proposed domain decomposition approach, a parallel surface mesher is developed by integrating a sequential mesher based on the advancing front technique (AFT) and employing the message passing interface (MPI) standard for parallel implementation. Combining the parallel surface mesher with the parallel tetrahedral mesher and improver we developed previously, a parallel preprocessing pipeline for large-scale simulations has been set up on the distributed computing environment. For typical computational aerodynamics configurations, it only consumes minutes to prepare a mesh containing hundreds of millions of elements; however, a traditional sequential pipeline may take several hours to prepare this mesh.

2. Related works

Domain decomposition approaches have been applied in various disciplines of parallel computing. Instead of presenting a complete range of references, our review only pertains to those related to parallel mesh generation. In accordance with how the inter-domain interfaces are treated in parallel mesh generation, three types of domain decomposition approaches are summarised [7]:

(1) Those for the parallel mesh generation approaches that mesh interfaces as they mesh subdomains;
(2) Those for the parallel mesh generation approaches that post-mesh the interfaces;
(3) Those for the parallel mesh generation approaches that pre-mesh the interfaces.

Chrisochoides and Nave [20] and Okusanya and Peraire [21] examined the first type of parallel mesh generation approach. A boundary conforming background mesh is partitioned into the same number of submeshes as that of processors. Each submeshes are refined on a single processor individually by using a parallel version of Delaunay point insertion kernel. If the cavity formed in the insertion of a new point does not cross the inter-domain boundary, the insertion operation runs locally; otherwise, a remote data gathering operation is required to enforce the mesh conformity. The submeshes are redistributed at intervals to balance the loads, and their boundaries are changed accordingly.

de Cougny and Shephard [7,22] adopted the second domain decomposition approach to parallelise their tetrahedral meshes. Firstly, the octant cells that cover (interior cells) or intersect (boundary cells) the problem domain are distributed evenly on the processors, and then the majority of interior cells is tetrahedralised using the mesh templates in parallel. Next, a cavity zone is formed by combing the boundary cells and their neighbouring interior cells on each processor. Between the adjacent cavities, three types of buffer zones are set up around the inter-processor faces, inter-processor lines and inter-processor points, respectively. The cavity zones are filled in an order of meshing non-buffer zones first, and then meshing face-related, line-related and point-related buffer zones successively. In the interim of these meshing steps, meshes are redistributed and the subdomain boundaries are changed as well.

In an early parallel advancing front algorithm proposed by Löhner [23], the meshing stage of subdomain interiors also takes the precedence to that for interfaces. An octree that covers the