AUTOMATIC UNSTRUCTURED MESH GENERATION ALGORITHM WITH MULTI-LEVEL PARALLELISM

Xiang Gao[*], Dali Li[†] and Chuanfu Xu[*]

[*]State Key Laboratory of High Performance Computing,
National University of Defense Technology, Changsha 410073, China

[†]College of Aerospace Science and Engineering, National University of Defense Technology,
Changsha 410073, China
e-mail: {gaoxiang12, achhx, xuchuanfu}@nudt.edu.cn

Key words: Unstructured mesh, Multi-level parallelism, Automatic generation, Tianhe-2 supercomputer

Abstract. Mesh generation is a very important pre-processing step in numerical simulation, and usually takes up most of the manual time of an entire simulation process. In this paper, a highly automatic multi-level parallel mesh generation algorithm is implemented. The improvement of automation is mainly achieved by optimizing the combination of multiple mesh optimization algorithms, and applying them to each sub-stage of mesh generation to improve the quality of the generated mesh. To further improve the efficiency of the algorithm, the domain decomposition method and MPI+OpenMP multi-level parallel technology are applied. The results tested on the Tianhe-2 supercomputer show that the grid quality is significantly improved by the combined grid optimization strategy; and with the multi-level parallelism, the speedup of the proposed algorithm in 24 processes is 21.6.

1 INTRODUCTION

At present, most of unstructured mesh generation softwares are mainly applied the advancing front method and the Delaunay algorithm. Since these algorithms are easy to generate low-quality cells near the boundary or constraint surface, which will bring problems such as calculation errors, time-consuming increase, and precision loss. There are two ways to improve the automation of mesh generation: one is to use the adaptive method to make the point selection more smart to improve the mesh quality during the process; the other is using the optimization algorithm to optimize the mesh after the mesh is generated. For sequential mesh generation algorithms, generating grids for large-scale numerical simulations is difficult due to the limited processor performance and memory of a single computing node. Parallel mesh generation algorithms can be divided into process-level and thread-level parallelism. The coarse-grained process-level parallelism generally decompose the domain and use multiple processes to mesh the parts. Fine-grained thread-level parallelism is generally based on the parallelism of the algorithm itself. In this paper, we based on the sequential mesh generation software NETGEN [1], and apply the MPI+OpenMP multi-level parallel technology to improve efficiency of the algorithm.

2 MESH OPTIMIZATION ALGORITHM

This work combines the advancing front method and the Delaunay algorithm to generate the initial mesh, and then different mesh optimization methods are utilized to optimize the mesh. However, other types of low-quality cells would generated when using certain method to address some types of lower-quality cells. In this paper, five mesh optimization schemes are implemented. They are edge swapping, face swapping, edge shrinkage, edge splitting, and grid smoothing. Edge swapping is performed by deleting one edge of the mesh and recombining the tetrahedron containing the edge, and is the reverse operation of the face swapping. Edge shrinkage is an operation that reduces low-quality cells by reducing two points to one point to reduce the mesh density to improve overall mesh quality. Edge splitting is performed by adding a new grid point on the edge and generating a corresponding tetrahedron around the edge of the segment, as shown in the left of Figure 1. Grid smoothing is to improve the quality of the mesh by moving the grid points, as shown in the right of Figure 1.

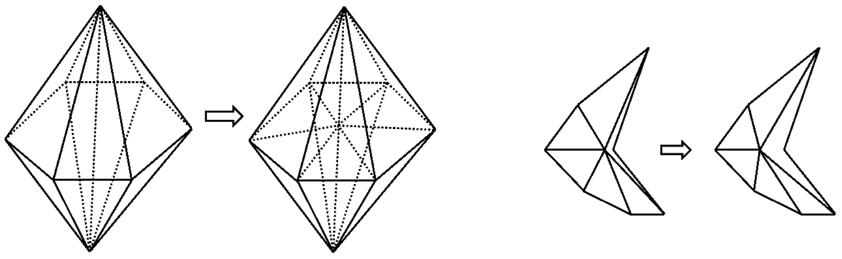

Figure 1: Edge splitting and grid smoothing

The five mesh optimization schemes are written into a standard function interface, and only parameter of the function is the mesh that needs to be optimized. While the mesh is passed in, it can complete the corresponding optimization operations, and the mesh quality statistics before and after optimization are output. During the scheme sequence determination stage, each optimization method will output the number of each type of cells and the overall mesh quality distribution. Users can select the next optimization scheme or replace the current optimization scheme based on these statistics. When the mesh quality meets a certain accuracy, the optimization stopped.

3 PARALLEL MESH GENERATION ALGORITHM

This work uses the MPI+OpenMP multi-level parallelism to speedup the mesh generation algorithm. The basic idea of process-level parallelism is to decompose the coarse mesh to each process, then each process generates the corresponding part. The difference between different methods is: whether each process mainly generates the volume mesh or the surface mesh, and whether the algorithm is to call the existing generation method or use the mesh refinement to generate the mesh. As shown in Figure 2, three different process-level parallel mesh generation algorithms are designed. The thin line box indicates that only the root process is executed, the bold line box indicates that each process is executed, and the bold red line box indicates that there are communication among each process at this stage. It can be seen that Algorithm 1 and Algorithm 2 mainly use mesh refinement to implement grid generation by using multiple processes. The Algorithm 1 refines the volume mesh, while the algorithm 2 refines the surface mesh and then generates the final mesh. Algorithm 3 generates the final fine surface mesh, and each process directly generates the volume mesh. Compared to Algorithm 3, the communication cost of Algorithm 1 and Algorithm 2 are increased. However, due to the use of mesh refinement, each process is load-balanced. In contrast, Algorithm 3 may have load imbalance in the process of generating fine-grained meshes, and mesh migration is needed. Thread-level parallel implementation is mainly to find functions with natural parallelism like multiple layers of for loops in the serial grid generator, and then add pragma parallel statements to optimize.

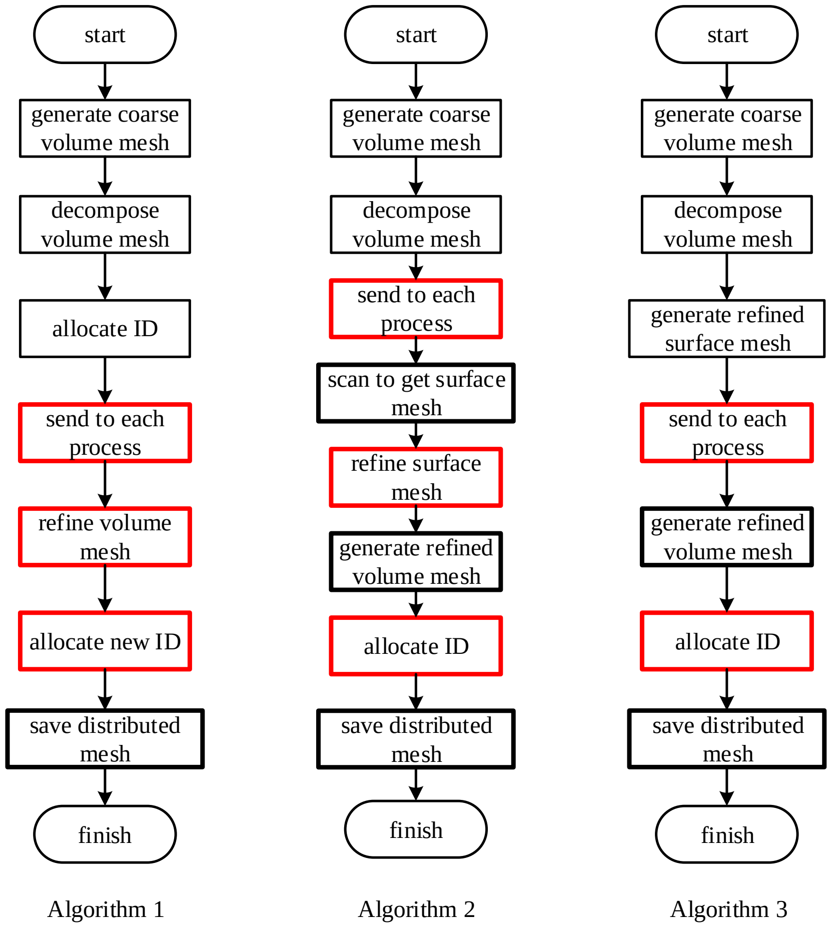

Figure 2: The flowcharts of three process-level parallel mesh generation algorithms

4 RESULTS

The case is tested on the Tianhe-2 supercomputer [2], each compute node is equipped with two Intel Xeon E5-2692V2 CPU and 64 GB memory, and each CPU has 12 cores. For the mechanical geometry case, the mesh quality (from 0 to 1, the higher the better) distribution before and after the mesh optimization strategy is shown in Figure 3.

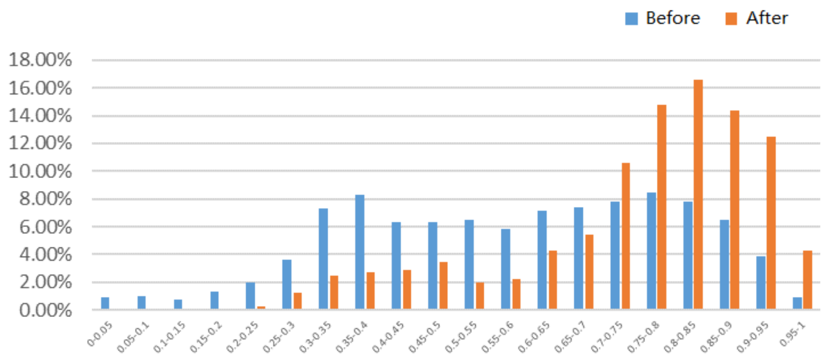

Figure 3: The distribution of mesh cell quality before and after optimization

Figure 4 shows the time cost for Algorithm 1 running from 1 to 24 processes with a grid size of 1 million. When the number of processes is 24, the parallel speedup of the algorithm is 21.6. 7 functions with high frequency of calls are paralleled in thread-level, and the average time cost is reduced by about 40%.

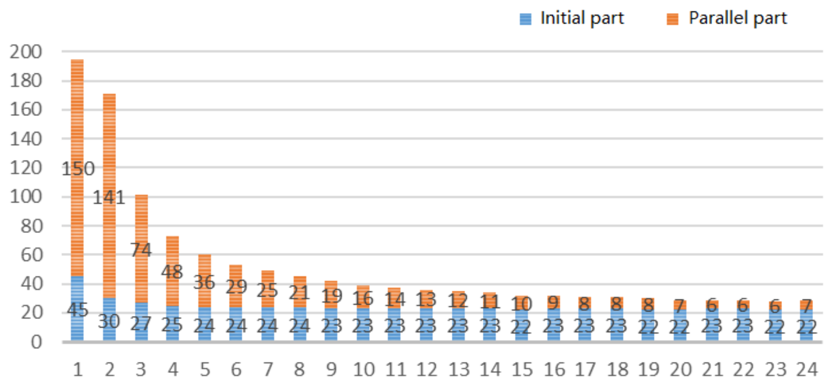

Figure 4: The time cost for Algorithm 1 running from 1 to 24 processes with a grid size of 1 million

REFERENCES

[1] Joachim, S. NETGEN An advancing front 2D/3D-mesh generator based on abstract rules. *Comput. Vis. Sci.* (1997) **1**:41–52.

[2] Liao X.K. and Xiao L.Q. and Yang C.Q. Milkyway-2 supercomputer: system and application. *Front. Comput. Sci.* (2014) **8(3)**:345–356.