Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 1956852, 11 pages
https://doi.org/10.1155/2020/1956852

Research Article

Study and Verification of Large-Scale Parallel Mesh Generation Algorithm for Centrifugal Pump

Liang Dong[1], **Yuhang Zhang**[1], **Zhipeng Ge**[1], **Cui. Dai**[2] **and Jinnan Guo**[1]

[1]*Research Center of Fluid Machinery Engineering and Technology, Jiangsu University, Zhenjiang 212013, China*
[2]*School of Energy and Power Engineering, Jiangsu University, Zhenjiang 212013, China*

Correspondence should be addressed to Cui. Dai; daicui@ujs.edu.cn

Received 21 December 2019; Revised 8 March 2020; Accepted 12 March 2020; Published 14 April 2020

Academic Editor: Xesús Nogueira

In order to reveal the details of the internal flow in a centrifugal pump, a large-scale mesh is needed. However, the mesh generated by the serial grid algorithm cannot meet the calculation requirements due to the huge amount of time. A large-scale parallel mesh generation algorithm of a centrifugal pump for high-performance computers is presented in this paper. First, a grid point set for the 3D Delaunay triangular mesh on the surface of the centrifugal pump is generated. Then, the S-H (Sutherland–Hodgman) algorithm for cropping and segmenting these grid point sets on the surface is employed. A uniform boundary mesh is generated and is divided into different subregions. In addition, in order to ensure the consistency of the interface mesh and to avoid the boundary mesh intersection overlap error, a parallel constrained Delaunay mesh generation algorithm based on region numbering is proposed, which can improve the quality and efficiency of the generated parallel mesh. Finally, the centrifugal pump is tested for verifying the parallel mesh algorithm in the Tianhe-2 supercomputer. PIV (particle image velocimetry) internal flow experiment is comparatively analyzed with the numerical simulation of large-scale mesh. The results show that the algorithm can generate 10^8 3D unstructured grid elements in 5 minutes, and the parallel efficiency can achieve 80%. The proposed algorithm not only ensures high grid quality with the serial grid algorithm but also accurately simulates the flow law in the centrifugal pump. The double-vortex structure which is obtained by PIV experiment is captured by the large-scale mesh.

1. Introduction

Centrifugal pump has strong rotation and large curvature, which usually requires a large-scale mesh of hundreds of millions of scales to reveal the complex turbulent flow. However, there are serious performance bottlenecks in time and CPU storage to generate such a large-scale complex mesh on a single machine. In addition, the design and manufacturing technology of high-performance computer hardware in China has been greatly developed, but there are still few software and applications dedicated to high-performance computers. Therefore, in order to make better use of high-performance computers, developing the high-efficiency parallel mesh generation algorithm is an important way to solve the performance bottleneck in CPU storage and time when generating large-scale grid on a single machine [1] and enable high-performance computers to be used more effectively. Parallel generation of mesh is a cross-cutting field of parallel computing and computational geometry. It is a complex technique of tightly coupled computational geometry, data structure, load balancing, and mesh partitioning. The CFD method is an effective and convenient method for numerical simulation, which provides intuition for further experiments.[2]. In terms of parallel logic, parallel mesh generation algorithm can be divided into two categories: algorithm parallel and problem parallel [3]. Algorithm parallelism is used to find out the degree of parallelism from the running process of serial algorithm, which can obtain multiple subtasks that can be executed concurrently. Algorithm parallel is generally fine-grained, and there are usually dependencies between each subtask, or data competition among each other. So, the parallel algorithm produces a lot of communication and synchronization costs, which reduces the parallel efficiency of the algorithm. The

problem parallel algorithm is the application of parallel algorithm design through the divide-and-conquer strategy. Compared with algorithm parallel, it is usually a coarse-grained hierarchy. The generation process of a single clipped surface mesh is regarded as an inseparable parallel subtask in the problem parallel. The problem parallel algorithm includes data decomposition and data mapping, which decompose the problem into subproblems and assign subproblems to different processors [4], respectively. The problem parallel algorithm only considers the perspective of data and does not need to be familiar with and master all aspects of the serialization algorithm [5].

The problem parallel algorithm can also reuse and inherit the serial algorithm very well compared with the algorithm parallel. The problem parallel algorithm does not require more complicated expansion and design. Therefore, problem parallel algorithm is the mainstream of current parallel mesh generation algorithm research. S.X. Li et al.[6] decomposed the mesh-generated area into several subareas according to the number of workstations, and the subarea mesh is generated. Then, the subarea mesh is transmitted back to the server via LAN to form the final mesh. In order to improve the parallel computing efficiency, the algorithm reduces the number of nodes on the boundary of the subregion and improves the mesh generation efficiency. However, the algorithm does not consider the problem of low mesh quality generated by the boundary of each subregion. Gatltier [7] proposed a method for directly decomposing geometric regions based on the RDT (repartitioning Delaunay triangulation) theory. The description of the subregion boundary is obtained by inputting the boundary of the region. The 3D mesh of the subregion is generated by the description of the boundary of each subregion. This method uses only the region boundary description instead of other conditions, so that the mesh generation can be paralleled under the general framework. The generated division plane has higher quality, and the mesh of the division plane can be generated in the mesh division process. In some cases, it will be redivided from zero when the Delaunay triangulation cannot find the original boundary. The increasing number of attempts will inevitably lead to an increase in CPU time, which will affect the efficiency of mesh generation. X.Q. Wang et al. [8] proposed a parallel generation method of unstructured tetrahedral mesh by using the advancing front technique [9] and the graph partition method of graph theory. The initial number of mesh was controlled by the relative volume ratio and the optimal partition, and the graph partition method of graph theory was used to decompose the initial mesh. Finally, parallel mesh was generated by the splitting method, and the parallel efficiency could reach 70% in the case of 512 cores. However, the mesh quality is poor. Some other processing must be considered after mesh merges [10] for some smaller-scale problems.

For the problem parallel algorithm, how to decompose the solved region into suitable subregions is a very important problem. The obtained subregions need not only to ensure similar workload but also ensure that sharp edges cannot be generated when the boundary is introduced. Otherwise, the sharp edges may reduce the quality of the mesh near the boundary. Based on the region decomposition method, Larwood et al. [11] took the largest coordinate direction as the normal of the division plane and avoided the narrow angle between the division plane and the original surface when selecting the segmentation position. However, this method is not as good for load balancing of large-scale mesh as for smaller-scale mesh. Buoffet et al. [12] used the recursive rigidity discomposing method to determine the direction and position of the division plane, but the disadvantage was that the division plane would affect the shape of the nearby volume elements. Polygon clipping algorithm is one of the most basic and widely used algorithms in computer graphics and image processing and has been widely used in many aspects. Sutherland–Hodgman polygon clipping algorithm is an early line clipping algorithm, which is simple, easy to implement, and widely used. For the rectangular clipping window, the clipping of polygons is decomposed into the clipping of the polygon one by one with the window boundary line (including extension lines). H. Liu et al. [13] proposed an algorithm suitable for clipping arbitrary polygons. First, the received vertices are input into the computer. Then, the vertices are rearranged and processed, and the final vertex sequence is output. The algorithm implements clipping of arbitrary polygons at the cost of the spatial complexity of the algorithm. C. Su et al. [14] improved the classical Sutherland-Hodgman clipping algorithm. The algorithm effectively avoids the generation of duplicate graph elements by adding necessary judgment and exit conditions in the process of dual-loop. How to get the interface between the division plane and the divided area is the key problem. In the process of domain decomposition, the poor-quality mesh elements are easy to be appeared at the mesh interface, and the mesh quality directly affects the accuracy and efficiency of CFD numerical calculation [15]. In order to ensure the stability of the parallel algorithm, it is necessary to avoid the influence of geometric features on the final mesh quality as much as possible. It is a good method to generate the interface mesh by finding the internal geometric description of the subarea and the description on the interface and combining the two subregions. In conclusion, the existing parallel grid generation algorithms have the following problems. A sharp angle is easy to appear in the boundary region. The mesh quality is poor at the artificial interface, and the parallel efficiency is low.

The paper proposes a massive parallel mesh generation algorithm to solve those problems. The proposed algorithm first introduces the basic framework of parallel mesh generation. Then, it expounds the subregion decomposition method and proposes an improved method based on Sutherland–Hodgman polygon clipping algorithm. In order to ensure the quality of the generated interface mesh and effectively avoid the intersection of the interfaces, an algorithm for generating the interface and parallel mesh is proposed. Finally, the parallel mesh algorithm is tested in the Tianhe-2 supercomputer.

2. Parallel Mesh Generation Algorithm

The overall framework of the parallel mesh generation method is shown as flowing. The mesh generation algorithm

based on the problem parallel first generates the triangular grid point set on the surface of the model through the Delaunay algorithm. Then, the boundary grid of the subregion is divided into each node to ensure that the boundary nodes of the adjacent subregion are consistent. At last, the Delaunay algorithm was used to divide the meshes. The subregions are combined, and the mesh quality improvement algorithm is employed.

2.1. Subregion-Solving Algorithm Based on Domain Decomposition. Domain decomposition is the key problem of parallel mesh generation algorithm, which directly affects mesh quality and parallel efficiency. The domain decomposition method decomposes the problem and solves each subproblem. So, the method has good applicability to the parallel environment [16–18]. The region decomposition is completed. Each processor region is assigned to a different processor core parallel computation in order to generate the required mesh and refine it. Because artificial interface will be generated in the process of region decomposition, it is easy to generate mesh intersection error and poor mesh quality at the interface. Therefore, the mesh without meeting the requirements needs to be redivided when the mesh quality inspection is carried out. This process mainly adopts the methods of step back and redivides the division plane, moving the division plane or meshing repair at the interface. So, the quality of parallel mesh requirements can be met.

The existing domain decomposition methods mainly include discrete domain decomposition and continuous domain decomposition [19]. The discrete domain decomposition method divides the sparse background grid covering the solved region and decomposes the background meshes into different subregions. In the process of discrete decomposition, when the background mesh is used for more complex models, it is easy to generate long and narrow faces and fragmented partitions, which affect the quality of the mesh. The continuous region decomposition method consists in to directly decompose the solution region into multiple subregions based on the geometric surface [20,21]. It does not need to generate sparse background mesh, and the adaptability to the complex model is better than discrete domain decomposition method. In addition, the subregions obtained by the continuous region decomposition method will not generate any new features other than the original region, such as small sharp corners.

The continuous region decomposition method is employed for better adaptability to complex models. The focus of domain decomposition problem is how to find the description of subregion interior and region boundary. In this paper, the S-H polygon clipping algorithm is used for region decomposition and boundary division. The efficient algorithm proposed by Sutherland and Hodgman for clipping convex polygons is to pass the vertices of polygons to each clipping stage, and the vertices can be immediately transferred to the next stage after each clipping stage [22]. The final output is the vertex queue that describes the clipped polygon boundary.

The process of S-H polygon clipping algorithm is shown as follows:

(1) The 3D Delaunay triangulation grid point set of the model is generated by the algorithm

(2) The S-H polygon clipping algorithm is used to segment the generated grid point set, and the boundary surface mesh with correspondent node can be formed

(3) The algorithm clipped and segmented the grid point set and obtained different subregions

The idea of finding the internal description of the subregion is as follows:

(1) The triangulated mesh of the model surface is input

(2) The internal description of the subregion is obtained by polygon clipping algorithm

(3) The triangular faces of each face are clipped to obtain a geometric description of the interior of the subregion

There are four situations (see Figure 1) to consider for cropping the edges of a polygon with clipping boundaries, and the processing method is as follows:

(1) The first endpoint of the polygon is outside the clipping boundary and the second endpoint is inside, which is sent to the intersection of the polygon edge and the window edge, and the second vertex is sent to the next clipper

(2) Both endpoints are inside the clipping boundary, and only the second vertex is sent to the next clipper

(3) The first endpoint is inside the clipping boundary and the second endpoint is outside, and only the intersection of the polygon edge and the window edge is sent to the next clipper

(4) Both endpoints are outside the cropped edge, and no vertices are passed to the next clipper

Figure 1 shows the main step of the polygon cutting algorithm. In order to achieve the transfer of vertices from one clipping phase to the next, the output of each clipper needs to be arranged in the format shown in Figure 2. When each pair of endpoints passes one of the four clippers, the following test results are used to generate an output for the next clipper.

The surface mesh in the example in this paper is composed of triangular facets. As long as the S-H polygon clipping algorithm is used for each triangular facets, the internal description of the subregion can be obtained, and then the description of the subregion interface is solved so that the subregion after subdivision can be reconstructed.

Figure 1 shows an example of the S-H polygon clipping algorithm for processing triangular facets defined by a point set {1, 2, 3}. In this example, the output vertex sequence is $\{2', 3, 3', 3''\}$.

According to the rules of the S-H polygon clipping algorithm, four steps are performed separately:

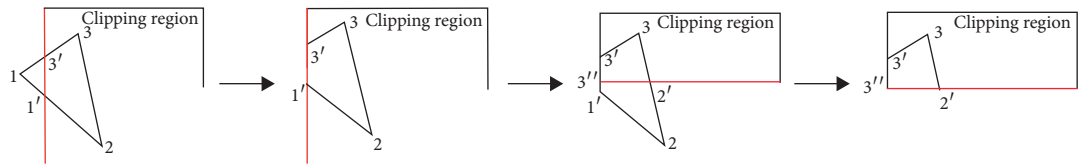


Figure 1: Uses of the S-H polygon clipping algorithm to process the set of vertices {1, 2, 3} near the interface of the subregion.

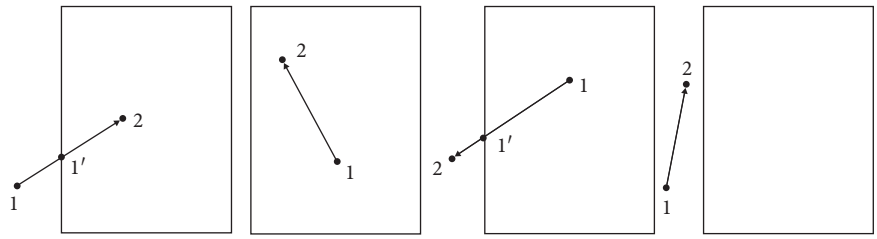


Figure 2: Four outputs generated by a pair of endpoints relative to the left edge of the cropping window.

①Crop the left part, Input boundary {2,3} (inside-inside)—Output {3},

Input boundary {3,1} (inside-outside)—Output {3′},
Input boundary {1,2}(outside-inside) —Output {1′,2};

②Crop the right part, Input boundary {3, 3′}(inside-inside)—Output {3′};

Input boundary {3′,1′}(inside-inside)—Output {1′};
Input boundary {1′,2}(inside-inside)—Output {2};
Input boundary {2,3}(inside-inside)—Output {3};

③Crop the lower part, Input boundary {3′,1′}(inside-outside)—Output {3″};

Input boundary {1′,2}(outside-outside—Output {NULL};
Input boundary {2,3}(outside-inside)—Output {2′,3};
Input boundary {3,3′}(inside-inside)—Output {3′};

④Crop the upper part, Input boundary {3″,2′}(inside-inside)—Output {2′};

Input boundary {2′,3}(inside-inside)—Output {3};
Input boundary {3, 3′}(inside-inside)—Output {3′};
nput boundary {3′, 3″}(inside-inside)—Output {3″};

The uniform boundary can be obtained by clipping the nodes near the interface.

2.2. Interface Mesh Generation Algorithm. Finding the interface between the dividing plane and the region to be divided is another key problem. In general, the interface should be described by a ring of edges that are connected end to end. The algorithm for solving the subregion interface is described in detail below. The algorithm to find the interface between the dividing plane and the region to be divided is as follows:

(1) The S-H polygon clipping algorithm is used to crop each triangular facet in the segmentation region and mark all edges on the segmentation plane as true

(2) All the edges labeled $e_i(i = 1,2, \ldots, n)$ are found and defined as E

(3) All associated edges in E are extracted to form a new set called G

(4) All simple loops in the set G are found to form a polygon set called P

(5) In the polygon set P, the smallest polygon subset p is found (the smallest polygon subset means that the polygons in the subset cover the entire graph and there is no inclusion relationship between them)

(6) The graph G that descripts as Φ of the interface is combined of all the polygons in the smallest polygon subset p

The interface of the subregion is found after the description of the internal geometric, and the two are combined to form different subregions called ψ.

2.3. 3D Mesh Generation and Subregional Merging. Delaunay triangulation is a popular automatic unstructured mesh generation method. Delaunay triangulation has also been well applied in the field of parallel mesh generation [23,24]. After domain decomposition, each subregion is distributed to different processor cores, and finally the volume mesh is generated for different subregions in parallel. When tetrahedral mesh is generated in the subregion, the existing open-source tetrahedral mesh generation software Gmsh is used to generate and refine the mesh. For a given 3D region or 3D point set, the software can be used to generate a 3D unstructured mesh.

Some mesh generation software can generate Delaunay volume mesh independently in different subregions. However, the existing serial algorithm cannot guarantee the consistency of meshes near the interface for subregions, and it is easy to induce mesh intersection error and poor mesh quality of the interface. When the parallel mesh generation algorithm generates intersection errors at the boundary, the

method of redividing the plane, moving the division plane or meshing repair at the interface is adopted to solve it. However, the disadvantage of this approach is that the process of backtracking and mesh repair will affect the overall running efficiency of the program [25]. Chew et al. [26] proposed a parallel constrained meshing algorithm based on the constrained Delaunay triangulation (CDT), which used the method of splitting data on the transmission edge of the adjacent subregion to ensure the consistency of the interface mesh and ensure parallel efficiency. In the CDT algorithm, the boundary surface mesh and the 3D mesh elements are generated, respectively, which inevitably leads to the problem of incorrect boundary mesh generation (poor mesh quality and intersection errors near the interface). Moreover, because the boundary mesh does not need to be fixed, in order to ensure the quality of the mesh near the boundary, it is easy to oversubdivide near the boundary. In order to solve the problem, a parallel constrained Delaunay mesh generation algorithm based on region numbering is proposed in this paper. In the proposed algorithm, the artificial boundary is introduced into the region to be meshed, and a subregion is created in each processor. Since the points on one side of the boundary have no effect on the other side, synchronization is not required during generating mesh elements. In addition, because the constraint edges are precisely divided, there is no need to transmit additional information. The process of the proposed algorithm implementation is described as below:

(1) For the generated interface mesh, two subregions near the interface are denoted as i and $i+1$, and the two subregions are named ψ_{i} and $\psi_{\mathrm{i+1}}$.

(2) The Delaunay mesh refined generation process is performed, and new points are continuously inserted into the subregion to form new tetrahedral elements.

(3) When the newly inserted point is located in the outer catch of the triangular plane T on the interface Φ, the insertion point at the interface F needs to be refined. At this time, the corresponding region ψ_{i} with a small region number was inserted in the middle of the peripheral circle of T to generate a volume mesh, while the subregion $\psi i+1$ with a large region number was not inserted. After waiting for the refine grid generated by ψi's insertion point, the inserted node data were transmitted to $\psi i+1$. Figure 3 shows a schematic diagram of the mesh generation of the algorithm near the boundary surface. In the figure, ABCD forms the interface plane Φ, ψ_{i}, and $\psi_{\mathrm{i+1}}$ are subregions on both sides of the interface plane Φ and JMK is the triangular facet within the interface F. Points P and Q are points in the external sphere of the triangular facet JMK that require Delaunay refine insertion.

Through the parallel constrained Delaunay mesh generation algorithm based on region numbering, and the parallel mesh generation guarantees the consistency of the mesh near the interface. The algorithm can avoid the boundary mesh intersection overlap error, which can reduce the operation of grid repair or interface redivision when the mesh quality does not meet the requirements. The quality and efficiency of parallel mesh generation algorithm can be guaranteed by this algorithm.

3. Verification of Parallel Grid Generation Algorithm

In this paper, the parallel mesh generation algorithm is tested by the Tianhe-2 supercomputer platform of Guangzhou supercomputer center. The Tianhe-2 supercomputer has 64G memory for each blade server node and has two Intel Xeon E5-2692 V2 processors and 24 processor cores per node. The test example used in the parallel mesh algorithm is the centrifugal pump impeller that needs to capture enough flow details. In this paper, the algorithm is analyzed and evaluated from two aspects of mesh generation quality and parallel efficiency by testing the example.

3.1. Comparative Analysis on the Quality of Generated Meshes. The aspect ratio is used to evaluate the quality of tetrahedral elements. For tetrahedral unstructured meshes, the aspect ratio is the proportional value of the longest side and the shortest side. When aspect ratio is closer to 1, the quality of mesh is higher. Considering the efficiency and memory limitation of Gmsh serial meshing, use parallel algorithms to generate about 2.1×10^7 grid elements, which is compared with the quality of Gmsh serial mesh generation under the same conditions. The comparison and distribution of grid quality are shown in Table 1. It can be seen that the mesh quality obtained by using the proposed parallel algorithm is almost the same as the original serial algorithm of Gmsh. The mesh quality distribution is basically the same. Moreover, the mesh quality is mostly distributed in the area within the aspect ratio value of 3, and the mesh quality is relatively high. It shows that the proposed parallel algorithm can well inherit the advantages of the original serial algorithm and can obtain the mesh with higher overall quality. The quality of the mesh generated by the proposed method is better than Chew algorithm.

3.2. Analysis of Parallel Efficiency. The mesh quality of the proposed algorithm can be guaranteed, and the centrifugal pump model is meshed with a different number of processors on a single node in the Tianhe-2 supercomputer. The proposed algorithm is compared with the Gmsh serial algorithm to analyze the parallel efficiency of the algorithm. Due to memory limitation, Gmsh single-core serial cannot directly generate hundreds of millions of grid elements, so both of them generate about 2.1×10^7 grid elements. The parallel efficiency statistics of the mesh generation algorithm are shown in Table 1. The definition of speedup ratio and parallel efficiency is as follows:

$$S_N = \frac{T_1}{T_N}, \qquad E_N = \frac{S_N}{N}, \tag{1}$$

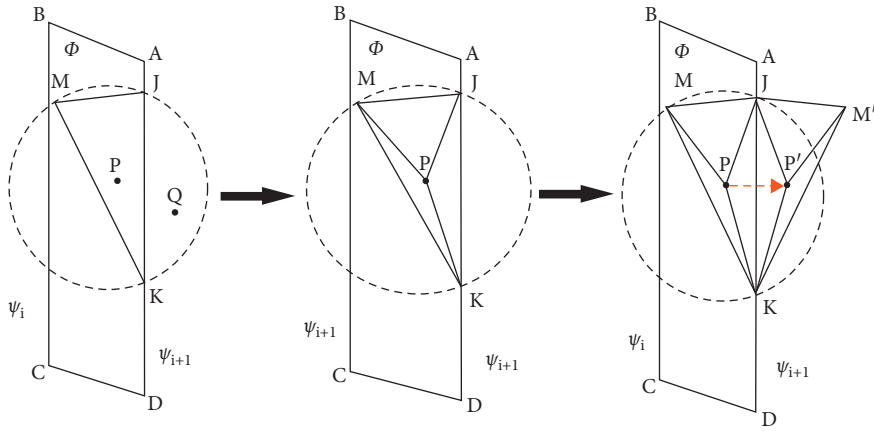


Figure 3: Schematic diagram of grid generation based on region numbering. (a) When subdividing and refining the mesh at the interface, the insertion point PQ is located in the JMK external ball. (b) Refining in the small numbered area ψ_i and no operation in the large numbered area ψ_{i+1}. (c) After the grid in the small numbered area ψ_i is generated, the node data are mirrored to ψ_{i+1}.

Table 1: Mesh element quality statistics.

	Aspect ratio							Overall mesh	
	1-1.5 (%)	1.5-2 (%)	2-3 (%)	3-4 (%)	4-5 (%)	5-7 (%)	7-10 (%)	Max	Avg
Serial algorithm	4.5	43.3	32.2	8.5	7.3	3.5	0.7	8.6	2.90
Proposed algorithm	4.0	43.1	30.5	10.1	8.2	3.0	1.1	8.8	2.98
Chew algorithm	3.0	40.1	31.2	11.5	7.6	4.1	1.5	9.4	3.12

where N is the number of processor used in parallel computing, S_N is the speedup ratio, T_1 is the time which takes for a single-core processor to mesh, T_N is the time which takes for N processor cores to mesh in parallel, and E_N is the parallel efficiency.

It can be seen from Table 2 that with the increase of the processor core, the parallel mesh generation time is significantly reduced compared with the Gmsh serial for the same scale. When the processor is used in parallel, the acceleration effect of the algorithm is obvious. When 12 core processes are used, the acceleration effect of nearly 10 times can be achieved, and the parallel efficiency is also high, which can reach about 75%. As the number of cores in parallel increases, the parallel efficiency of the mesh generation algorithm begins to decrease. When parallel processor core is increased to 24 core, the algorithm can achieve 12.25 times the acceleration effect. But the efficiency of parallelism drops to about 50% in this case. The extratime consumed by the process such as region decomposition and interface generation in the algorithm increases gradually, and the number of iterations between subregions increases, leading to the decrease of parallel efficiency.

The mesh diagram of the centrifugal pump impeller water body of 10^8 grid elements is generated in parallel by the algorithm under 24 processor cores at one node, which can be finished in 5 minutes. Therefore, in general, the proposed parallel mesh generation algorithm can effectively break through the limitation of existing serial programs on grid scale compared with existing serial mesh generation programs. The proposed parallel mesh generation algorithm can play a good acceleration effect and greatly reduces the generation time of large-scale mesh with the same number of meshes.

4. Numerical Simulation of Centrifugal Pump with Large-Scale Grid

4.1. Numerical Calculation Model and Method. The design parameters of the centrifugal pump model are as follows: flow rate $Q_d = 25\,m^3/h$, rated head $H = 10m$, rated speed $n = 1450r/min$, and specific speed $m = 78.4$. The structural parameters of the centrifugal pump model are shown in Table 3.

ANSYS CFX-Linux of the Tianhe-2 supercomputer center is used in this test. The k-ε turbulence model is employed for steady calculation 27. LES is employed for transient calculation. Smargorinsky is selected for a subgrid model. The impeller water body is set as the rotational flow field, and the static water bodies are set as the static flow field. The interface between the dynamic and static computational domains is used for data exchange. The steady calculation uses the frozen rotor interface. The transient calculation uses the transient rotor/stator interface. The wall of the calculation domain is set as the no-slip condition, and the near wall is set as automatic wall condition. Grids are GGI-compliant. The roughness to 25 μm according to the actual processing conditions is set. The time step is set to $\Delta T = 0.00011495\,s$, which is the time required for the impeller to rotate 1°. The convergence accuracy in the calculation process is set to 1×10^{-4}. The inlet is set to the fixed pressure of 1 atm. The export condition is set to the mass flow outlet.

4.2. Mesh Generation. From Figure 4, we can see that the comparison of the head calculation results under different

Table 2: Parallel efficiency statistics of mesh generation.

Items	Processor number				
	1	6	12	18	24
Time (s)	419.31	86.11	46.03	36.43	31.74
Speedup S_N	—	4.87	9.11	11.51	12.25
Parallel efficiency E_N (%)	—	81.13	75.92	60.94	51.04

Table 3: Main structural parameters of the model pump.

Description	Numerical value
Number of impeller blades z	6
Diameter of the impeller inlet D_1 (mm)	74
Diameter of the impeller outlet D_2 (mm)	200
Wrap angle of impeller blades φ (°)	71
Width of the impeller outlet b_2 (mm)	8
Diameter of the base of the volute D_3 (mm)	204
Width of the volute inlet b_3 (mm)	12

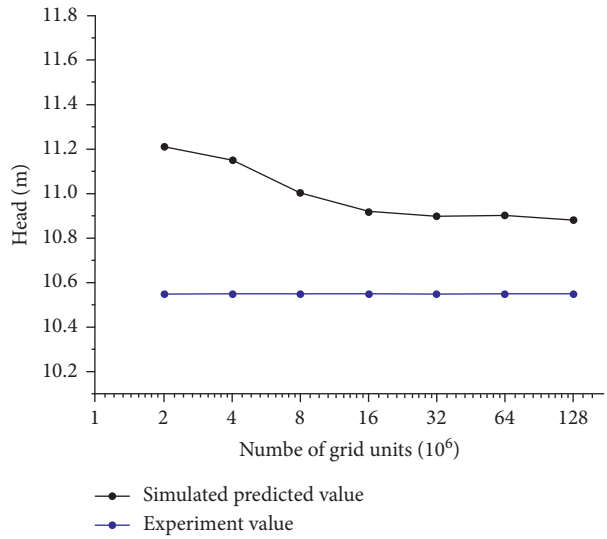


Figure 4: Calculation head under different grid cell numbers.

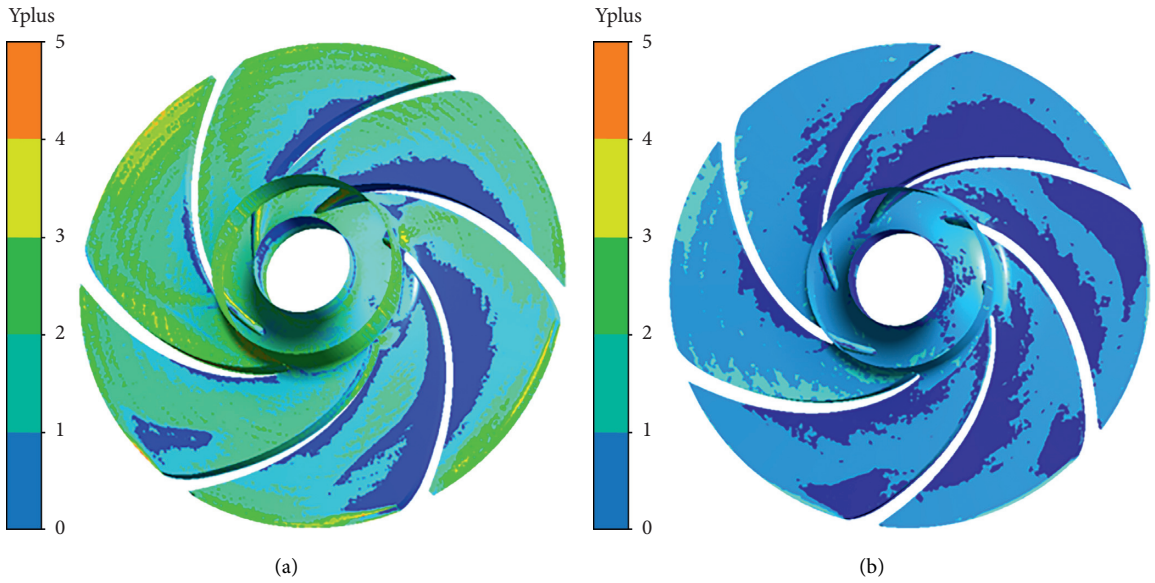
mesh scale. When the grid is about 8 million, the error between the predicted results and the test is less than 5%. When the number of grid cells reaches about 16 million, t the calculated heads are basically not changed greatly, and the relative errors of the two predictions are within 1%. And the error between the calculated head and the test head is only 3.2%. Therefore, the reliability of the calculated results can be guaranteed by selecting grids with 16 million or more grid cells.

The requirements of large eddy simulation on wall mesh are very strict. The grid node of the first layer is the viscous sublayer of the wall. The wall should meet $y^+<5$, and it is better to reach less than 1. It is necessary to continuously refine the mesh on the wall surface and verify it through calculation. Finally, two sets of meshes meeting the requirements of large eddy simulation were selected for comparative analysis. A smaller set of grids S (20 million grid cells) and a set of large-scale grids L (145 million grid cells) were selected for comparative analysis.

For the impeller mesh L generated by the proposed parallel mesh algorithm, $y+$ analysis is performed on its numerical calculation results at the wall surface, and the results are shown in Figure 5. It can be seen from Figure 5 that the $y+$ of the shroud and hub of the impeller is mostly within 3, while the $y+$ of the refined blade suction surface and pressure surface is mostly controlled within 1. Therefore, the grid generated by the parallel algorithm can meet the requirement that the first layer of the wall mesh is arranged in the viscous sublayer in the large eddy simulation.

4.3. Comparative Analysis with PIV Experiment Results. Numbering the centrifugal pump model flow channel used in the experiment, as shown in Figure 6.

In order to verify the accuracy of large-scale parallel meshes in numerical simulation, the calculation results of the two meshes are compared with the PIV experiment results. During the comparison, the intermediate section of



(a) (b)

Figure 5: y^+ distribution at the impeller wall surface calculated by (a) Mesh L and (b) Mesh S.

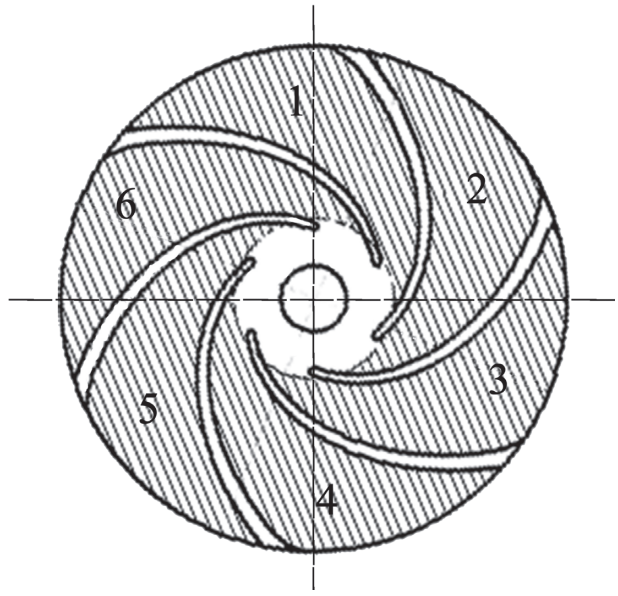


Figure 6: Flow channel numbering schematic diagram.

the impeller at the initial moment of the last unsteady calculation period was selected as the reference surface, where the scale of velocily was the same as the shot section of PIV experiment.

It can be seen from Figure 7 that the overall distribution trend of the flow lines of the two sets of grids is very close to the test results, and both of them generate the vortex structure of unsteady flow at the working face of the impeller. The numerical simulation cloud pictures of the two sets of grids have a relatively high velocity fluid area at the flow channel 6. The reason is that the flow channel 6 is located near the volute tongue and has an effect on the velocity distribution at this location.

The large-scale grid L has a greater advantage in the flow details. It is shown in the following aspects: at the large flow rate of $1.2Q_d$, the distribution of the streamline is more uniform and closer to the condition captured by the test. Especially, the flow line at flow channel 2 is more smooth than the mesh S, while the vortex cluster of unsteady flow appears in multiple flow channels with a large flow rate. At a small flow rate of 0.8Qd, the mesh L captures the double vortex structure reflected in the test in the flow channel 4

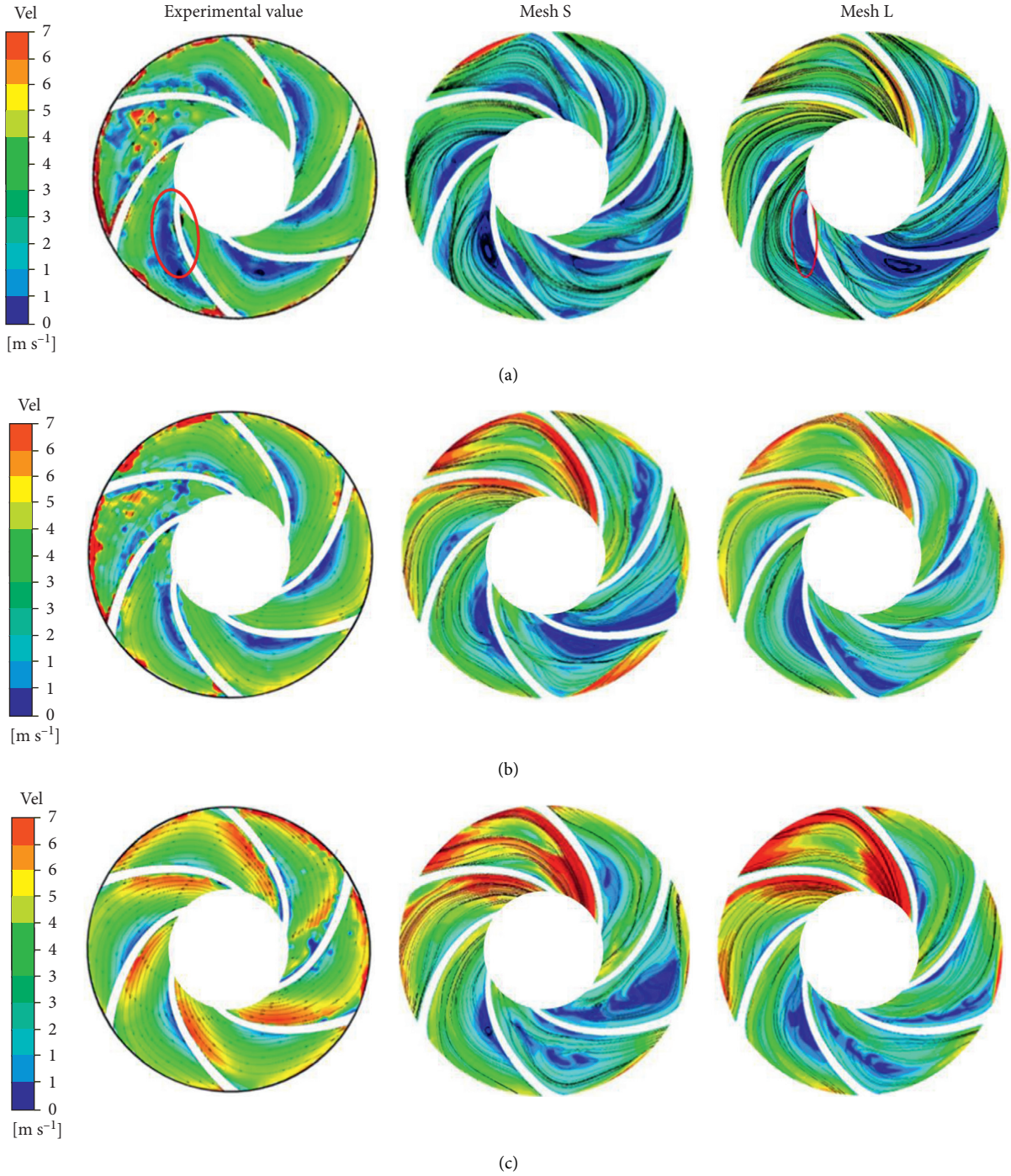


FIGURE 7: Comparison of relative velocity distribution of the intermediate section of the impeller. (a) $Q = 20\,m^3/h$ $(0.8Q_d)$. (b) $Q = 25\,m^3/h$ $(1.0\ Q_d)$. (c) $Q = 30\,m^3/h$ $(1.2\ Q_d)$.

(the position of the red ellipses in Figure 6), while the grid S does not capture this detail.

5. Conclusion

Serial mesh generation algorithm cannot solve the problem of memory and time bottleneck for large-scale mesh. A large-scale parallel mesh generation algorithm is proposed based on continuous domain decomposition, and the large-scale meshing and numerical calculation of the centrifugal pump in the Tianhe-2 supercomputer is carried out to illustrate the efficiency and quality of the generated mesh. The conclusion is as follows:

(1) The proposed algorithm is based on continuous domain decomposition. The mesh on the bounding surface with the same node data is obtained, and

different subregions are divided through the S-H polygon clipping algorithm. The problem of sharp angle and long and narrow face, which are generated when complex geometry is decomposed into discrete region, can be solved by the proposed algorithm.

(2) A parallel mesh generation algorithm for constrained Delaunay is proposed based on region numbering theory, which ensures to generate uniform volume mesh near the interface. The high-quality mesh is generated at the artificial interface when generating meshes in the coarse-grained problem parallel method, which improves the efficiency and stability of the parallel mesh generation process.

(3) The proposed parallel mesh algorithm is tested on "Tianhe-2". The quality of the grid obtained by the parallel algorithm is basically the same as that obtained by the original serial algorithm. Most of the aspect ratios of the parallel grids are less than 3, which guarantee quality of the grid. The algorithm can effectively break through the memory and time limit of a single machine, which can generate three-dimensional unstructured mesh with 10^8 units in 5 minutes. The parallel efficiency can reach about 80% when 12 cores running in parallel.

(4) Two sets of girds with different sizes are selected and compared with the experimental results. The flow in the centrifugal pump is simulated accurately through the large-scale parallel meshes generated by the proposed parallel mesh generation algorithm. The y^+ of the front and back cover board surface of impeller is mostly within 3, and the working and back surface of the blades is mostly within 1. Moreover, the double vortex structure is better captured though the set of girds L, which is closer to the results obtained by PIV.

Data Availability

All the data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (Grant no. 2016YFB0200901), National Natural Science Foundation of China (Grant nos. 51879122 , 51579117 , and 51779106), Association Innovation Fund of Production, Learning, and Research (Grant no. BY2016072-01), Zhenjiang Key Research and Development Plan (Grant nos. GY2017001 and GY2018025), Open Research Subject of Key Laboratory of Fluid and Power Machinery, Ministry of Education, Xihua University (Grant nos. szjj2015-017, szjj2017-094, and szjj2016-068), Sichuan Provincial Key Lab of Process Equipment and Control (Grant no. GK201614 and GK201816), Advanced Talent Foundation of Jiangsu University (Grant no. 15JDG052), and a project funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD) Jiangsu Top Six Talent Summit Project (Grant no. GDZB-017).

References

[1] N. Chrisochoides, "Parallel Mesh Generation," in *Numerical Solution of Partial Differential Equations on Parallel Computers. Lecture Notes in Computational Science and Engineering*, A. M. Bruaset and A. Tveito, Eds., Vol. 51, Springer, Berlin, Germany, 2006.

[2] Y. Liu, X. Lu, and W.-J. Cai, "Hydrodynamic performance of marine current turbine with diffuser based on overlap grid method," *Journal of Drainage and Irrigation Machinery Engineering*, vol. 37, no. 7, pp. 606–611, 2019.

[3] C. D. Antonopoulos, F. Blagojevic, A. Chernikov et al., "A multigrain Delaunay mesh generation method for multicore SMT-based architectures," *Journal of Parallel and Distributed Computing*, vol. 69, no. 7, pp. 589–600, 2009.

[4] D.-W. Zhao, *Parallel Unstructured Mesh Generation Approaches for Large-Scale Numerical Simulations*, Zhejiang University, Hangzhou, China, 2016.

[5] N. Chrisochoides and D. M. Nave, "Parallel Delaunay mesh generation kernel," *International Journal for Numerical Methods in Engineering*, vol. 58, no. 2, pp. 161–176, 2003.

[6] S.-X. Li, Y.-P. Wang, and Y.-Q. Chen, "Distributed parallel FEM mesh generation on LAN," *Computer Engineering and Design*, vol. 26, no. 12, pp. 3165–3193, 2005.

[7] J Galtier and P.-L. George, "Prepositioning as a way to mesh subdomains in parallel," in *Proceedings of the 5th international meshing roundtable*, Pittsburgh, PA, USA, October 1996.

[8] X. Q. Wang and X. L. Jin, "A method for large-scale parallel tetrahedral mesh generation," *Journal of Vibration and Shock*, vol. 33, no. 21, pp. 102–107, 2014.

[9] J. Schöberl, "NETGEN: An advancing front 2d/3d-mesh-generator based on abstract rules," *Computing and Visualization in Science*, vol. 1, pp. 41–52, 1997.

[10] X. –Q. Wang and X. –L. Jin, "Parallel finite element mesh generation method and its application," *Chinese Journal of Computational Mechanics*, vol. 32, no. 2, pp. 256–261, 2015.

[11] B. G. Larwood, N. P. Weatherill, O. Hassan, and K. Morgan, "Domain decomposition approach for parallel unstructured mesh generation," *International Journal for Numerical Methods in Engineering*, vol. 58, pp. 177–188, 2003.

[12] Boufflet, "A modular design for a parallel multifrontal mesh generator," in *Proceedings of the 8th international euro-par conference on parallel processing*, Springer, Paderborn, Germany, August, 2002.

[13] H. Liu, Z. Tian, X.-Y. Li, and J. Chen, "Design and implementation of IPC arithmetic used in plane clipping," *Computer Technology and Development*, vol. 24, no. 2, pp. 224–233, 2014.

[14] C. Su and J.-G. Han, "An advanced Sutherland-Hodgman algorithm," *Journal of Xi'an University of Posts and Telecommunications*, vol. 18, no. 3, pp. 80–82, 2013.

[15] L. Dong, C. Dai, Y. –M. Zhang, J. –W. Xiao, X. –C. Tang, and Y. –F. Liu, "An optimization algorithm for improving tetrahedral mesh quality," *Journal of Drainage and Irrigation Machinery Engineering*, vol. 33, no. 5, pp. 397–401, 2015.

[16] A. Toselli and O. B. Widlund, *Domain Decomposition Methods: Algorithms and Theory*, Springer, Berlin, Germany, 2005.

[17] J. –F. Huang, S. Yang, L. –Y. Long, and L. –X. Zhang, "Numerical study on dynamic characteristics of flow around vane in channel based on overset meshe," *Journal of Drainage and Irrigation Machinery Engineering*, vol. 37, no. 4, pp. 319–324, 2019.

[18] W. –C. Xu and J. Hu, *Computational Fluid Dynamics*, Beijing Institute Of Technology Press, Beijing, China, 2011.

[19] Z. –Q. Guan, J. –L. Shan, and Y. –X. Gu, "Surface mesh generation based on riemannian metric," *Chinese Journal of Computers*, vol. 29, no. 10, pp. 1823–1833, 2006.

[20] J. –J. Chen, *Unstructured Mesh Generation and its Parallelization*, Zhejiang University, Zhejiang, China, 2006.

[21] E. C. Sherbrooke, *3-D shape interrogation by medial axial transform*, PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1995.

[22] D. Hearn and M. P. Baker, "Computer graphics with OpenGL," pp. 273–277, Publishing House of Electronics Industry, Beijing, China, 3rd ed edition, 2010.

[23] J. –J. Zheng, *Some Issues on Dynamic Unstructured Grid Generation for Parallel CFD Simulation*, Zhejiang University, Zhejiang, China,, 2016.

[24] R. Said, N. P. Weatherill, K. Morgan, and N. A. Verhoeven, "Distributed parallel delaunary mesh generation," *Computer Methods in Engineering*, vol. 58, no. 2, pp. 161–176, 2003.

[25] Y. Liang, J. –J. Chen, L. –G. Chen et al., "Parallel planar Delaunay mesh generation," *Journal of Zhejiang University(Engineering Science)*, vol. 42, no. 4, pp. 558–564, 2008.

[26] L. P. Chew, N. Chrisochoides, and F. Sukup, "Parallel constrained Delaunay meshing," *New*, American Society of Mechanical Engineers (ASME) Press, in *Proceedings of Trends in Unstructured Mesh Generation*, pp. 89–96, York, NY, USA, 1997.

[27] W. Zhang, Y. –C. Yu, and H. –X. Chen, "Numerical simulation of flow in centrifugal pump impeller at off-design conditions," *Journal of Drainage and Irrigation Machinery Engineering*, vol. 28, no. 1, pp. 38–42, 2010.