Two-Stage In Situ Parallel Meshing for Large-Scale Atmospheric Fluid Simulation Over Complex Topography

Kenji Ono and Takanori Uchida

Abstract This paper describes a two-stage in situ mesh generation approach to improve the throughput of large-scale parallel simulations involving complex topography. This approach splits the mesh generation processes into two stages: in the first stage parameters are extracted which uniquely determine the mesh distribution, and in the second stage, a mesh system is generated in parallel by a proposed meshing algorithm based on the parameters obtained at the initialization of the simulation. To make the meshing parameter extraction efficient, we developed an easy-to-use visual application, which enables the user to interactively explore the optimal parameters. The proposed two-stage approach gives us more flexibility since we can select an arbitrary number of processes during run-time. The preliminary results for the meshing performance show excellent scalability in the strong scaling.

1 Introduction

Fluid simulations involving complex topography have many applications such as global climate prediction, local weather forecasting, and wind condition prediction for wind power generation, and these simulations can require large-scale computational meshes to obtain high fidelity results. These applications demand an efficient mesh generation process for parallel computation, as well as the ability to handle

Kenji Ono
Research Institute for Information Technology, Kyushu University, 744 Motooka Nishi-ku, Fukuoka, Japan
RIKEN Center for Computational Science, Kobe, Japan
e-mail: keno@cc.kyushu-u.ac.jp

Takanori Uchida
Research Institute for Applied Mechanics, Kyushu University, Japan
e-mail: takanori@riam.kyushu-u.ac.jp

complex topography and a high resolution mesh to resolve the boundary layer near the surface.

Typically, mesh generation for such complex configurations have utilized commercial software suites which have practical functions and user-friendly graphical user interfaces (GUI). Interactive mesh generation processes which can be run on a PC enable us to confirm the quality of the generated mesh. On the other hand, if the number of grid points increases, several issues arise: high memory usage, long processing times, large output files, and the transfer of large files from a PC to a computing server. In situ data processing is an effective approach for handling large datasets to avoid low-speed disk access, which is a key factor in determining the throughput. Cartesian mesh generation is suitable for this in situ approach [3].

In this paper, we propose a novel two-stage in situ meshing approach to address the aforementioned issues when generating a large-scale parallel mesh system for complex topography. This approach splits the mesh generation process into two stages: in the first stage parameters are extracted which uniquely determine the mesh distribution, and in the second stage, a mesh system is generated in parallel by a proposed meshing algorithm based on the parameters obtained at the initialization of the simulation. To assist the extraction of the optimal meshing parameters, a graphical interface is developed to enable us to interactively search for the optimal parameters in a heuristic manner. The proposed two-stage approach allows us to robustly generate a mesh system using the optimal parameters, and by avoiding the transfer of the mesh file our approach is also scalable and a high throughput.

2 In Situ Parallel Mesh Generation for Complex Terrains

2.1 Coordinate System and Application

The mesh generation of the terrain geometry for atmospheric fluid simulation has to consider the complex configuration of the surface topography, and have a suitable resolution to resolve the boundary layer. We consider a uniform Cartesian mesh on the XY-plane (horizontal direction), and a non-uniform distribution in the Z (vertical) direction, similar to the sigma coordinate system [2]. Equation (1) expresses the mapping between the physical and the computational space.

$$\begin{cases} x = x(\xi) \\ y = y(\eta) \\ z = z(\xi, \eta, \zeta) \end{cases} \leftrightarrow \begin{cases} \xi = \xi(x) \\ \eta = \eta(y) \\ \zeta = \zeta(x, y, z) \end{cases}, \tag{1}$$

Here, x, y, z and ξ, η, ζ indicate the physical and the computational space coordinates, respectively.

The RIAM-COMPACT wind simulator was introduced in [8]. This wind simulator exploits the collocated grids, in the boundary-fitted coordinate (BFC) system, and can be used for large-eddy simulation (LES) to model turbulence to predict the

local wind flow over complex topographies. RIAM-COMPACT simulator is used in this paper to assess the performance of our proposed method.

2.2 1st Stage: Extracting Meshing Parameters

In the first stage, we find the parameters which uniquely determine a mesh system. A graphical application, named *FXgen*, is used to help determine the optimal parameters which are used in the second stage. The essential parameters for determining the mesh are the size of the computational region, the number of grid points, and the mesh spacing for both ends of the line segments in the Z direction. These parameters can be interactively explored via a user-friendly GUI as illustrated in Fig. 1. A similar approach for generating a mesh is found in [6], but our approach does not use a mesh file.

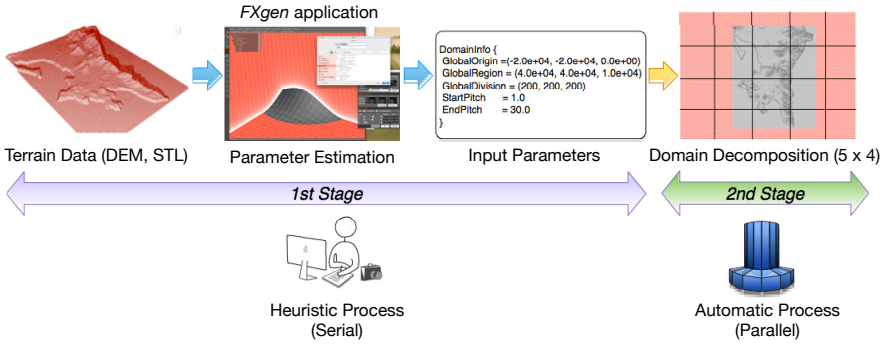


Fig. 1 Overview of the two-stage meshing process. The first stage is heuristic, while the second stage is automatic and parallelized

A meshing algorithm is required to determine the mesh distribution along the Z-direction using the commonly used Vinokur's stretching function [9], which requires the number of divisions in the Z direction, and the mesh spacing for both ends of the line segments. The Vinokur's one-dimensional stretching function was implemented *FXgen* to optimize the mesh distribution in the Z direction. As noted in [9], the stretching function is iterated to satisfy the slopes (gradients) of both ends of the line segment, although the spacing parameters may not be satisfied. Therefore, we implemented an additional outer loop so that the stretching process is repeated until the specified spacing constraints are satisfied, and we set the maximum number of iterations to 20. The number of iterations mainly depends on the number of points to divide, the length of the line segments, and the spacing at both ends. In other words, the number of iterations strongly depends on the local topography. Therefore, considering the domain decomposition approach, the number of iterations may vary according to the conditions of each domain, and the load balance may be affected

and degrade. Thus, we first need to calculate the length of each line segment in the Z direction, for which the lengths differ depending on the height of the local topography. The well-established intersection algorithm [4] is used to determine the intersections on the surface and calculate the length of each line segment.

The users are required to repeat this procedure until they obtain the optimal parameters. The obtained optimal parameters are then exported in a JSON like format, and can be passed to the simulator.

2.3 2nd Stage: In Situ Parallel Meshing in a Simulator

The obtained parameters are used to generate the same mesh in the first stage during the initialization of the simulation. In the second stage, first, the entire computational domain is divided by the number of processes given in the control file or with command line arguments during the invocation of the parallel simulation program. The domain decomposition is performed using a function from CPMlib [5] or CBrick [1] libraries, which are designed to easily construct MPI applications, and optimize their performance. A domain decomposition pattern is automatically calculated by the algorithm. In the same manner as other traditional atmospheric simulation applications, the entire computational domain is decomposed only in the XY direction, while remaining intact in the Z direction. By implementing the same algorithm for the mesh generation in both stages, it becomes possible to generate the desired mesh based on the obtained parameters. The proposed two-stage approach gives us more flexibility since we can choose an arbitrary number of processes during run-time thanks to on-the-fly meshing.

3 Experimental Results and Discussion

3.1 Intrinsic Mathematical Formula: Isolated Peak

Figure 2 shows a velocity field for the special case when the mesh is generated using a formula to express a simple configuration, which can be embedded in the source code without a geometry shape file. In this single isolated peak, the calculation of the intersection can be avoided and is replaced by a cosine function, and we can obtain the z coordinate on the surface directly. It was verified that the fine mesh surrounding the surface has sufficient resolution to correctly reproduce the boundary layer, and as a result, to reproduce the structure of the flow separation at the ridge.

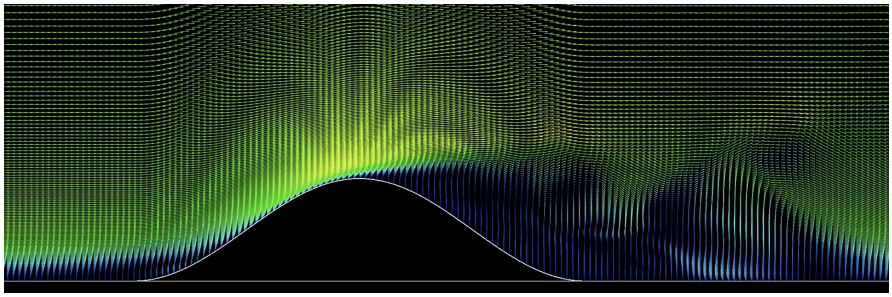
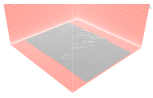
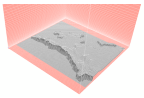
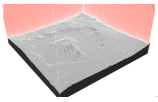
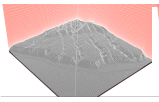
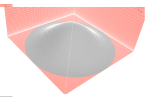


Fig. 2 Closeup of the computed vector field at the center of the isolated peak example. The mesh system has a resolution of $801 \times 181 \times 181$, and the Reynolds number based on the height of the peak is 10^4. The fine mesh well captures the boundary layer and the separation at the ridge. The color indicates the magnitude of the velocity vectors

3.2 Performance Evaluation

Table 1 shows the measured timings and required memory for the first stage of the proposed method when using an Apple MacBook Pro with Intel Core-i7@3.3 GHz, and 16GB of main memory. We evaluated five geometries with different complexities of shape in terms of the mesh sizes: $100 \times 100 \times 50$; $200 \times 200 \times 50$; and $400 \times 400 \times 50$ in which only the sizes in the X and Y directions vary. We verified that as the mesh size is decreased, the processing time also decreases in all cases. In addition, when the mesh contains fewer triangles, the processing time decreases. However, case (d) was an outlier where the number of iterations was much larger than for other cases. Figure 3 shows the mesh generated for case (d).

Table 1 Processing time and the required memory for different polygonal datasets

	(a)	(b)	(c)	(d)	(e)
					
Number of triangles	5,840	43,623	133,110	642,970	1,071,836
File size	292 KB	2.2 MB	6.7 MB	121 MB	53.6 MB
Used memory[a]	205 MB	283 MB	231 MB	660 MB	445 MB
100x100x50 (0.5×10^6)	0.3 s	< 1.0 s	40.0 s	216.0 s	33.0 s
200x200x50 (2×10^6)	0.8 s	< 1.0 s	182.0 s	890.0 s	128.0 s
400x400x50 (8×10^6)	2.0 s	1.6 s	756.0 s	3323.0 s	499.0 s

[a] Maximum memory for a $400 \times 400 \times 50$ grid

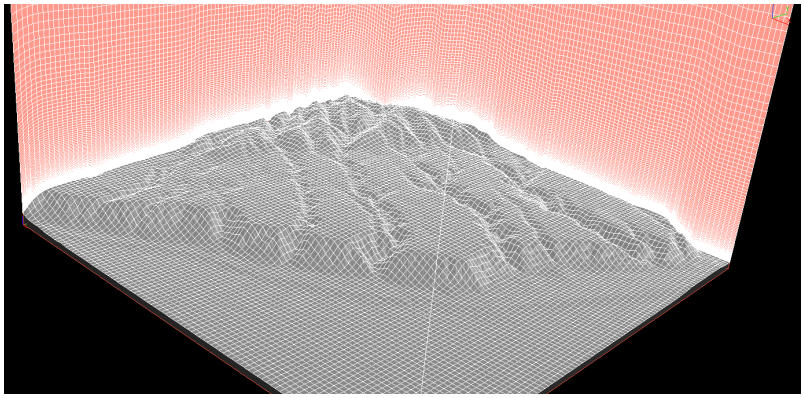


Fig. 3 Example of a mesh generated for actual topography. The location is a coastal area in Kochi-city, Japan

The second stage of the proposed mesh generation method was evaluated by measuring the time required to run RIAM-COMPACT simulation on the supercomputer "ITO", which compose of Intel Xeon Gold 6154 processors (3.0GHz, Skylake-SP), with 18-core×2-CPU and 192GB of main memory [7]. The generated mesh size was 4001×721×721 (approximately 2 billion points), and the required memory was 132GB. The measurement was carried out by FlatMPI execution mode. Figure 4 shows the measured mesh generation time for the case of the isolated peak explained in section 3.1. We obtained excellent scalability up to 4,096 processes in the strong scaling test.

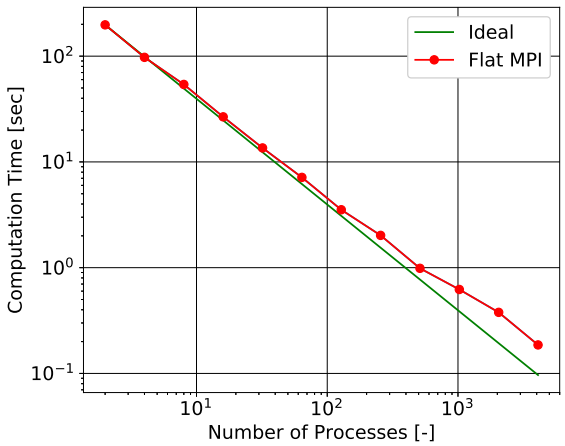


Fig. 4 Measured time in the second stage (isolated peak example). The triangles and the red circles indicate the ideal performance and the measured strong scaling performance, respectively

4 Conclusion

A two-stage in situ mesh generation method was proposed to reduce the computational cost of the meshing process for a large-scale parallel atmospheric fluid simulation with complex topography. The proposed approach splits the mesh generation process into two parts: One is the determination of the parameters which uniquely determine the mesh distribution with a heuristic approach; and the other is the generation of a mesh with a meshing algorithm based on the obtained parameters in parallel at the initialization of the simulation. The proposed two-stage approach gives us more flexibility since we can choose an arbitrary number of processes during runtime thanks to the in situ meshing. From the preliminary performance evaluation, it was found that our approach had excellent scalability in the strong scaling.

Acknowledgements This research is partially supported by the Japanese Ministry of Education, Culture, Sports, Science and Technology (MEXT) as a social and scientific priority issue to be tackled using the post-K computer (Project ID:hp170238 for Development of Innovative Design and Production Processes that Lead the Way for the Manufacturing Industry in the Near Future, and hp170270 for Accelerated Development of Innovative Clean Energy Systems).

References

1. CBrick. https://github.com/RIIT-KyushuUniv/CBrick [Cited 22 May 2018]
2. Chu, Peter C. and Fan, C.: Hydrostatic correction for sigma coordinate ocean models. Journal of Geophysical Research (2003) Vol. 108, No. C6, 3206, doi:10.1029/2002JC001668
3. Lintermann, A., Schlimpert, S., Grimmen, J.H., Günther, C., Meinke, M., Schröder, W.:Massively parallel grid generation on HPC systems. Computer Methods in Applied Mechanics and Engineering, Vol.277, pp.131–153, (2014), doi:10.1016/j.cma.2014.04.009
4. Möller, T. and Trumbore, B.:Fast, Minimum Storage Ray-Triangle Intersection. Journal of Graphics Tools, No.2, pp..21–28 (1997)
5. Ono, K., Kawashima, Y., and Kawanabe, T.:Data Centric Framework for Large-scale High-performance Parallel Computation. Procedia Computer Science, Vol.29, pp.2336–2350 (2014), doi:10.1016/j.procs.2014.05.218, https://github.com/avr-aics-riken/CPMlib
6. Evetts, S.: Glyph Scripting Automates Terrain Meshing for Wind Turbine Siting. http://www.pointwise.com/theconnector/2014-September/Glyph-Scripting-Automates-Terrain-Meshing-Wind-Turbine-Siting.html [Cited 2 Aug. 2018]
7. Research Institute for Information Technology, Kyushu University: Supercomputer System ITO, https://www.cc.kyushu-u.ac.jp/scp/eng/system/01_into.html [Online; accessed 15-May-2018]
8. Uchida, T.:CFD Prediction of the Airflow at a Large-Scale Wind Farm above a Steep,Three-Dimensional Escarpment. Energy and Power Engineering, Vol.09, pp.829–842, (2017), doi:10.4236/epe.2017.913052
9. Vinokur, M.: On One-dimensional Stretching Functions for Finite-difference Calculations. Journal of Computational Physics, Vol.50, pp.215–234, (1983)