The hierarchical domain decomposition pre-processing module with the parallel mesh refinement function without communication

Kohei Murotani, Shin-ichiro Sugimoto, Hiroshi Kawai, and Shinobu Yoshimura

The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
[muro kawai]@save.sys.t.u-tokyo.ac.jp, [sugimoto yoshi]@sys.t.u-tokyo.ac.jp

Summary. This paper describes a parallel fast generation method of large-scale meshes for a hierarchical domain decomposition method implemented in the open source parallel finite element software ADVENTURE. The present authors have newly developed a parallel algorithm such that communication table is updated and inherited in parallel during mesh refinement after the initial communication table is generated from an initial mesh. As a result, the generation of a refined mesh model over billions degrees of freedom (DOFs) from an initial medium-size mesh model of about a million DOFs can be performed in a parallel computer in a short time.

Key words: Parallel mesh refinement, Hierarchical domain decomposition tool, Parallel finite element analysis, ADVENTURE system, METIS, ParMetis.

1 Introduction

In a hierarchical domain decomposition method (HDDM), an analyzed domain is decomposed in two steps by ParMETIS and METIS [1]. The large decomposed unit of the first hierarchy level is called a "Part", and a smaller unit in the decomposed "Part" (the second hierarchy level) is called a "Subdomain". In a parallel computer, one "Part" must be respectively assigned to one computational node, and each "Part" is further partitioned into a number of "Subdomains". Parallel finite element solvers developed in the ADVENTURE project [2] employ the HDDM, but do not have a function of hierarchically decomposing domain of mesh. Instead, we developed ADVENTURE_Metis, which converts an input mesh into a hierarchically domain decomposed mesh for the HDDM.

Using the parallel solid solver ADVENTURE_Solid, various problems with hundreds of millions of DOFs have already been solved [3] in the Earth Simulator (ES), which was the fastest supercomputer in the world from 2002 to 2004. ADVENTURE_Metis executed the domain decomposition with low parallel performance and many integer operations in advance. The same situation would be true in Japan's Petaflops Supercomputer, nicknamed the "K computer", which was recognized to attain the world fastest performance of supercomputer, that is, a peak performance of 8.774 Petaflops in June 2011. To fully use the performance of the "K computer", large-scale meshes of more than tens of billions of DOFs are required. However, it is very difficult to generate such large-scale meshes from scratch and partition the generated meshes.

To overcome the above mentioned problems when using the HDDM solver efficiently in the Petaflops computer, we newly developed the hierarchical domain decomposition tool named ADENTURE_Metis Ver.2, to which a parallel mesh refinement function and a communication table generation function without communication were newly implemented. The strategy employed in the tool is in the following. An adaptively controlled mesh of medium-size is generated at the initial stage of mesh generation. Next the generated medium-sized mesh is uniformly refined. The first reason for employing such processes is that the time for generating the initial mesh should be as short as possible. The second reason is that the mesh should be finely refined in the entire domain, because nonlinear dynamic analyses are our final target.

High speed parallel mesh refinement has been actively investigated so far. Most of the investigation has been in edge-based adaptive refinements. Ruprecht and Muller [4] developed the streaming adaptive refinement using the tetrahedral edge-based subdivision, and Pebay et al. [5] extended both methods.

The generated mesh must have the hierarchical domain decomposition structure of two layers for the use of HDDM. In the algorithm where tetrahedra are subdivided as [4, 5], a communication table for sending and receiving data among computational nodes must be generated after mesh refinement. However, the generation cost of the communication table tends to be expensive. To overcome such difficulty, the present authors have developed a parallel algorithm such that the communication table is updated and inherited during mesh refinement after the initial communication table is generated with an initial mesh.

2 Eight-subdivision of a tetrahedron

In ADVENTURE_Metis Ver.2, a tetrahedron is subdivided into eight tetrahedra as shown in Fig. 1.

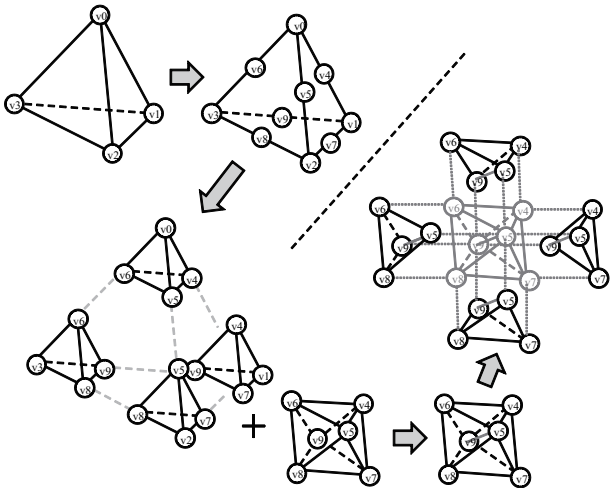


Fig. 1. Subdivision of a tetrahedron into eight tetrahedrons.

The ID numbers are stored as the following recurrence formulas (1) to (5) :

$$\begin{cases} v_i^{k+1} = v_i^k & (0 \le i < vn^k) \\ v_{vn^k+i}^{k+1} = vn^k + e_i^k & (0 \le i < en^k) \end{cases} \tag{1}$$

$$\begin{cases} e_{2i+j}^{k+1} = 2e_i^k + j & (0 \le i < en^k, 0 \le j < 2) \\ e_{2en^k+3i+j}^{k+1} = 2en^k + 3f_i^k + j & (0 \le i < fn^k, 0 \le j < 3) \\ e_{2en^k+3fn^k+i}^{k+1} = 2en^k + 3fn^k + t_i^k & (0 \le i < tn^k) \end{cases} \tag{2}$$

$$\begin{cases} f_{4i+j}^{k+1} = 4f_i^k + j & (0 \le i < fn^k, 0 \le j < 4) \\ f_{4fn^k+8i+j}^{k+1} = 4fn^k + 8t_i^k + j & (0 \le i < tn^k, 0 \le j < 8) \end{cases} \tag{3}$$

$$\{ t_{8i+j}^{k+1} = 8t_i^k + j (0 \le i < tn^k, 0 \le j < 8) \tag{4}$$

$$\begin{cases} vn^{k+1} = vn^k + en^k \\ en^{k+1} = 2en^k + 3fn^k + tn^k \\ fn^{k+1} = 4fn^k + 8tn^k \\ tn^{k+1} = 8tn^k \end{cases} \tag{5}$$

where k is the level of mesh refinements, v_i^k,e_i^k,f_i^k and t_i^k are the ID numbers of the vertex, tetrahedral element, face and edge, respectively, for the k-th level refined mesh, and vn^k,en^k,fn^k and tn^k are the

total numbers of vertices, tetrahedral elements, faces and edges, respectively, for the k-th level refined mesh.

3 Communication table generation without communication

3.1 Algorithm for generating communication table using global ID number

It is ideal that communication among computational nodes is not performed to create large-scale data in a parallel computer. This section describes an algorithm for generating the communication table without communication. The communication table here is a correspondence relationship to send and receive data among "Parts".

An ID number assigned to each simplex before "Part" decomposition is defined as a "global ID number". An ID number assigned to each simplex in each "Part" after "Part" decomposition is defined as a "local ID number". A mapping from the local ID numbers in all "Parts" to the global ID numbers is defined as the map "g". The local ID numbers on boundaries among neighboring "Parts" are redundantly assigned to the global ID numbers by the map "g". Using the fact that the global ID numbers on boundaries among neighboring "Parts" are necessarily overlapped, the communication table among "Parts" is generated.

$v_i^{p\to q}$ is defined as the ith local vertex ID number adjacent to "Part" q in "Part" p. Similarly, $v_i^{q\to p}$ is defined as the ith local vertex ID number adjacent to "Part" p in "Part" q. If the number of vertices on the boundary between "Part" p and "Part" q is $n_{p,q}$, let the set of the local vertex ID numbers adjacent to "Part" q in "Part" p be $\{v_i^{p\to q} \mid 0 \le i < n_{p,q}\}$. Similarly, let the set of the local vertex ID numbers adjacent to "Part" p in "Part" q be $\{v_i^{q\to p} \mid 0 \le i < n_{p,q}\}$. Here, if $\{v_i^{p\to q} \mid 0 \le i < n_{p,q}\}$ is sorted by the global vertex ID numbers, $\{v^{p\to q}_{\sigma_{p\to q}(i)} \mid 0 \le i < n_{p,q}\}$ is obtained. $\sigma_{p\to q}(i)$ is the sorting permutation using the global vertex ID number in an ascending order. Similarly, $\{v^{q\to p}_{\sigma_{q\to p}(i)} \mid 0 \le i < n_{p,q}\}$ is also obtained. Since the set of the global vertex ID numbers of the vertices adjacent to "Part" q in "Part" p is equal to the set of the global vertex ID numbers of the vertices adjacent to "Part" p in "Part" q,

$$g(v^{p\to q}_{\sigma_{p\to q}(i)}) = g(v^{q\to p}_{\sigma_{q\to p}(i)}) \tag{6}$$

is obtained. Equation (6) is called the communication table. It is important to create $\{v^{p\to q}_{\sigma_{p\to q}(i)} \mid 0 \le i < n_{p,q}\}$ in "Part" p and $\{v^{q\to p}_{\sigma_{q\to p}(i)} \mid 0 \le i < n_{p,q}\}$ in "Part" q without knowing the local vertex ID numbers in the other "Parts". Next, consistent communication as follows using the relationship given in equation (6) is performed. First, send data in the order corresponding to $\{v^{p\to q}_{\sigma_{p\to q}(i)} \mid 0 \le i < n_{p,q}\}$ in "Part" p, and send data in the order corresponding to $\{v^{q\to p}_{\sigma_{q\to p}(i)} \mid 0 \le i < n_{p,q}\}$ in "Part" q. Second, each set of sent data is exchanged for the other. Finally, each set of received data is respectively assigned to the vertices corresponding to $\{v^{p\to q}_{\sigma_{p\to q}(i)} \mid 0 \le i < n_{p,q}\}$ in "Part" p and $\{v^{q\to p}_{\sigma_{q\to p}(i)} \mid 0 \le i < n_{p,q}\}$ in "Part" q.

3.2 Consistent updating algorithm of global ID number after mesh refinement

In ADVENTURE_Metis Ver.2, since mesh refinement is performed after "Part" decomposition, a consistent set of global vertex ID numbers must be assigned to newly generated vertices in all of "Parts". Once the consistent global vertex ID numbers are assigned, the communication table can be generated by the method as described in Subsection 3.1. This subsection describes how to independently assign the consistent global ID numbers in each "Part" after mesh refinement.

The local ID numbers after mesh refinement are updated as per equations (1) to (5). The updating procedure of the global ID numbers after mesh refinement is defined as follows :

$$\begin{cases} g(v_i^{k+1}) = g(v_i^k) & (0 \le i < vn^k) \\ g(v_{vn^k+i}^{k+1}) = g(vn^k) + g(e_i^k) & (0 \le i < en^k) \end{cases} \tag{7}$$

$$\begin{cases} g(e_{2i+j}^{k+1}) = 2g(e_i^k) + j & (0 \le i < en^k, 0 \le j < 2) \\ g(e_{2en^k+3i+j}^{k+1}) = 2g(en^k) + 3g(f_i^k) + j & (0 \le i < fn^k, 0 \le j < 3) \end{cases} \tag{8}$$

$$\{ g(f_{4i+j}^{k+1}) = 4g(f_i^k) + j (0 \le i < fn^k, 0 \le j < 4) \tag{9}$$

$$\begin{cases} g(vn^{k+1}) = g(vn^k) + g(en^k) \\ g(en^{k+1}) = 2g(en^k) + 3g(fn^k) + g(tn^k) \\ g(fn^{k+1}) = 4g(fn^k) + 8g(tn^k) \\ g(tn^{k+1}) = 8g(tn^k) \end{cases} \tag{10}$$

where v_i^k,e_i^k and f_i^k are the local vertex, edge and face ID numbers in each "Part", respectively, $g(v_i^k)$,$g(e_i^k)$ and $g(f_i^k)$ are the global vertex, edge and face ID numbers in each "Part" , respectively, and $g(vn^k)$,$g(en^k)$,$g(fn^k)$ and $g(tn^k)$ are the total numbers of vertices, edges, faces and tetrahedral elements in the entire domain. If the global ID numbers are updated using equations (7) to (10), the communication method using the global ID number described in Subsection 3.1 is directly applied because the consistent global ID numbers are determined independently in each computational node.

4 Large-scale mesh refinement for Pantheon model and Its Analysis

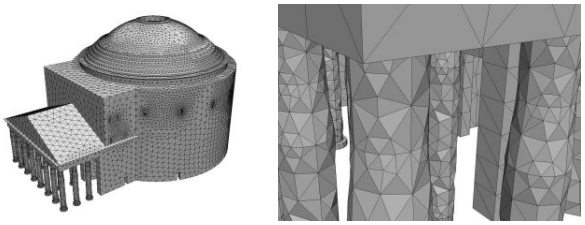
The parallel computer used in this section is the T2K in the Information Technology Center of the University of Tokyo. As for the T2K, the cores of up to 8,192 cores of 2.3 GHz and 32 GB memory can be used. For the purpose of comparison with results on the T2K, a single PC machine with 4 cores (Intel Core i7-930, 2.8 GH) and with a memory of 24 GB is used as well.

In this section, a mesh with 1,506,447 DOFs and 323,804 elements generated from the CAD data of the Pantheon released in the ADVENTURE project [2] is used as an initial mesh model. Figure 2 shows the initial mesh, and Fig. 3 shows its fourth-level refined mesh. Table 1 is a comparison of the calculation conditions of the HDDM and the calculation times in rank 0 for the different numbers of computational nodes. The finite element models are the tetrahedral quadratic element models using the surface fitting function based on finite element shape functions.

Since 8,192 computational nodes in the T2K can be used only for 24 hours, the fourth-level refined mesh with 8,192 "Parts" and 2,000 "Subdomains" per "Part" was generated in advance on a single PC using ADVENTURE_Metis Ver.2. A self weight analysis using 8,192 computational nodes in the T2K was performed for the fourth-level refined mesh by the HDDM-based parallel structural solver, ADVENTURE_Solid. Figure 4 shows the equivalent stress distribution calculated with the initial small mesh, while Fig. 5 does that of the fourth-level refined mesh. Large-scale meshes like this can be easily prepared even on a single PC in a reasonable time by using ADVENTURE_Metis Ver.2.

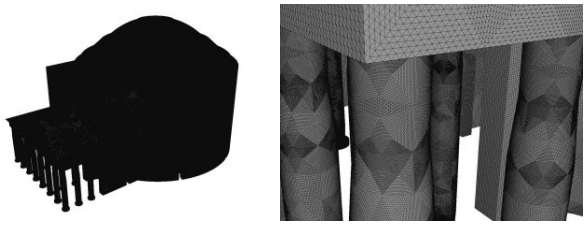
Table 1. Calculation conditions and costs of mesh refinement for the Pantheon model.

Number of processes	Parts	Number of refinments	Input DOFs	Output DOFs	Calculation times (sec)
4	8192	4	1,506,447	5,359,260,867	408.576
1024	8192	4	1,506,447	5,359,260,867	21.374
8192	8192	4	1,506,447	5,359,260,867	84.540



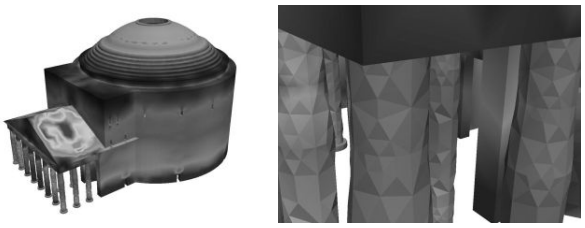
(a) (b)

Fig. 2. Initial mesh.



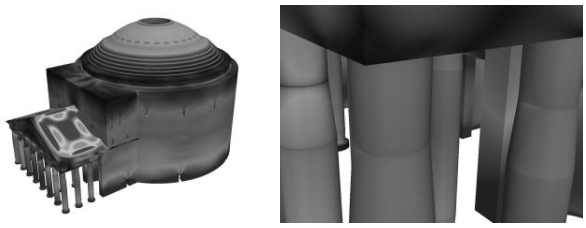
(a) (b)

Fig. 3. Fourth-level refined mesh.



(a) (b)

Fig. 4. Equivalent stress distributions of the initial mesh.



(a) (b)

Fig. 5. Equivalent stress distributions of the fourth-level refined mesh.

5 Conclusions and future work

ADVENTURE_Metis Ver.2 has already been used for HDDM-based parallel solvers such as ADVENTURE_Solid and ADVENTURE_Magnetic. ADVENTURE_Solid together with ADVENTURE_Metis Ver.2 has succeeded in solving the problem with 5.3 billion DOFs using T2K as shown in Section 4. Furthermore, since the number of "Parts" is flexibly set, irrespective of the number of computational nodes, data of a number of "Parts" to be solved on parallel computers with a number of computational nodes can be created by a single PC. That means, ADVENTURE_Metis Ver.2 enables to generate a mesh with hundreds of billions of DOFs on a single, normal-performance computer.

References

1. G. Karypis, and V. Kumar, "A Fast and Highly Quality Multilevel Scheme for Partitioning Irregular Graphs", *SIAM Journal on Scientific Computing*, vol.20, no.1, pp.359–392, 1999.
2. Home page of ADVENTURE Project. http://adventure.q.t.u-tokyo.ac.jp
3. M. Ogino, R. Shioya, H. Kawai, and S. Yoshimura, "Seismic Response Analysis of Full Scale Nuclear Vessel Model with ADVENTURE System on the Earth Simulator", *Journal of the Earth Simulator*, vol.2, pp.41–54, 2005.
4. D. Ruprecht, and H. Muller, "A Scheme for Edge-based Adaptive Tetrahedron Subdivision", *Mathematical Visualization*, H. C. Hege, K. Polthier, editors, Springer Verlag, Heidelberg, pp.61–70, 1998.
5. P. P. Pebay, and D. C. Thompson, "Parallel Mesh Refinement Without Communication", *Proceedings 13th International Meshing Roundtable*, Willimasburg, Virginia, USA, Sandia National Laboratories, pp.437–448, 2004 .