Repartitioning Unstructured Meshes for the Parallel Solution of Transient Problems
(Extended Abstract)
PeiZong Lee, Chih-Ihsueh Yang, and Jeng-Renn Yang
Institute of Information Science
Academia Sinica
Taipei, Taiwan, R.O.C.
Internet:
{leepe,dickyang,yangjr}@iis.sinica.edu.tw
TEL: +886 (2) 2788-3799
FAX: +886 (2) 2782-4814

1 Introduction

Many practical applications are usually time-varying (transient) and have complex geometries. Therefore, more than one mesh can be adopted in numerical simulations. The internal combustion engine, which consists of chemical reactions, moving valves and pistons, and fuel injection, is typically a transient problem with changing shapes. The mesh should be regenerated if the valves and pistons move. The intermediate values also need to be transferred from the old mesh to the new one [4, 9].

Figure 1 shows a period of 128 frames for engine combustion, which involves the processes of fuel and air intake, compression of the fuel-air mixture, ignition and combustion of the charge, expansion of gases, and the removal of waste. For this type of transient application, in which shapes change with time, it is practical to generate a separate unstructured mesh for each frame (of an object geometry within a period of operation). Figure 2 shows parts of unstructured meshes for frame 1 and frame 2. The unstructured mesh is regenerated because the left intake valve moves. When simulating operations, we use interpolation techniques to transfer the status of variables from frame $i$ to frame $i + 1$ for $1 \leq i < M$, and from frame $M$ to frame 1, where we assume that a period of operation includes $M$ frames.

Unstructured meshes are becoming important as they can be generated automatically for applications with complex geometries or for those with dynamically moving boundaries [13]. For engine combustion applications, a fast approach might be to regenerate local meshes for places where boundaries change.

However, the quality of newly generated meshes in these places might be poor in terms of aspect ratio, area ratio, and edge ratio among the triangles (elements or cells of a mesh) [7]. The quality of a mesh influences the convergence rate of PDE solvers. Therefore, it is more suitable to generate a separate mesh for each of the frames which represent boundary geometries for a period of operations.

When dealing with parallel processing, we have to partition the computing domain such that each processing element (PE) contains a part of an unstructured mesh. Then, during parallel computation, each PE calculates its local data and occasionally exchanges data with other PEs. Although partitioning and repartitioning adaptive meshes are heavily studied [1, 2, 3, 5, 6, 10, 11, 12, 14], we did not see much work emphasizing partitioning and repartitioning unstructured meshes for transient problems.

2 An overview of the proposed methods

In this paper, we present three algorithms for partitioning and repartitioning unstructured meshes for parallel solutions of transient problems.

2.1 Algorithm 1: Partitioning each mesh independently

We use a static partitioning algorithm to decompose each mesh into partitions according to the weight of each PE [8]. Therefore, for each mesh partitioning, we can completely achieve load balance with as few cut edges between adjacent partitions as possible. However, each mesh partitioning is done independently, thus the volume of data migration between partitions of consecutive meshes may be very large.

2.2 Algorithm 2: Providing a static partitioning for all meshes

We use a regular background mesh, which can be treated as a complete quadtree, as a coarse graph. A node in the coarse graph represents a cell of the regular background mesh. The weight of each node represents the number of triangles for all unstructured meshes whose gravity centers fall within the territory of the corresponding cell in the regular background mesh. An edge connects two nodes if there exists at least one pair of adjacent triangles, whose gravity centers fall within each territory of these two corresponding cells in the regular background mesh. The

\footnote{This work was partially supported by the NSC under Grants NSC 91-2213-E-001-010 and NSC 91-2213-E-001-018.}
Figure 1: A period of 128 frames for the engine combustion.
weight of an edge represents the number of pairs of these adjacent triangles.

Based on the complete quadtree, we can apply a multi-level refinement algorithm to partition the coarse graph according to the weight of each PE [8]. Result partitioning is applied for all unstructured meshes. Static partitioning does not require to do data migration among partitions of consecutive meshes; however, it does not guarantee load balance for any mesh partitioning. In the engine combustion example, since the left intake valve and the right outtake valve do not appear at the same time, load imbalance is a crucial dilemma for static partitioning.

2.3 Algorithm 3: Refining consecutive partitionings

We use a regular background mesh as mentioned in Algorithm 2. We define a changing factor for each cell of the regular background mesh based on the number of triangles (falling within the cell) changed between two consecutive meshes.

If the shape of a consecutive mesh is changed so that the sum of all changing factors is larger than a threshold value, we repartition the new mesh based on a static algorithm [8]. Otherwise, we adopt multi-level refinement [8], dynamic diffusion [15], and Kernighan-Lin refinement to improve load balance and reduce cut edges between adjacent partitions.

3 Discussion

We have evaluated the effectiveness of our partitioning/repartitioning methods with the engine combustion benchmark application. A preliminary experimental result for studying the partitioning and repartitioning of 128 consecutive unstructured meshes for the engine combustion application will be reported in the workshop.

References


