11th World Congress on Computational Mechanics (WCCM XI)
5th European Conference on Computational Mechanics (ECCM V)
6th European Conference on Computational Fluid Dynamics (ECFD VI)
July 20–25, 2014, Barcelona, Spain

PARALLEL MESH ADAPTATION USING PARALLEL GRAPH PARTITIONING

C. Lachat[1], C. Dobrzynski[2] and F. Pellegrini[3]

[1] Inria Bordeaux Sud-Ouest, 200 av. de la Vieille Tour, 33405 Talence, France
cedric.lachat@inria.fr
[2] Institut Polytechnique de Bordeaux & Inria Bordeaux Sud-Ouest, 351 cours de la Libération, 33405 Talence, France cecile.dobrzynski@inria.fr
[3] Université de Bordeaux, LaBRI & Inria Bordeaux Sud-Ouest, 351, cours de la Libération, 33405 Talence, France francois.pellegrini@inria.fr

Key words: *HPC, parallel remeshing, distributed memory, anisotropic mesh, mesh adaptation, parallel graph partitioning*

Today's large scale simulations can only be run in parallel, because the sizes of the meshes are too big to fit in the memory of a single computer. Since shared-memory architectures are subject to memory bottlenecks, scalability can only be achieved by using distributed-memory architectures such as workstation clusters. Therefore, a prerequisite for such simulations is to be able to generate huge meshes in a parallel, distributed-memory fashion. Moreover, in the case where users would like to perform mesh adaptation, the latter must also be performed in parallel. In this work, we propose a parallel mesh adaptation procedure that relies on existing sequential mesh adaptation software. The only requirement of this sequential software is that it should not modify prescribed mesh elements. The advantage of this approach is its ease of use through the ability to use existing sequential software to generate huge meshes in reasonable time.

Algorithm description. Our approach is based on an iterative parallel algorithm that applies to distributed meshes. First, we consider the submesh of all elements that are flagged as needing to be remeshed. We divide it into zones of a size suitable for the sequential remesher. Each of these zones, which may span across several processors, is centralized on a given host processor, so as to balance remeshing workload. Then, processors remesh their assigned zones in parallel, and send back the remeshed zone parts to their origin processors. Finally, remeshed parts are reintegrated into the original mesh, remeshed elements are no longer flagged, and the process is restarted until no flagged elements remain.

Experimental validation. Our method has been implemented in the PaMPA parallel library [3]. The purpose of this library is to relieve solver writers from the tedious and error-prone task of writing again and again service routines for mesh handling, data communication and exchange, remeshing, and data redistribution. PaMPA represents

meshes as graphs, whose data is distributed across the processors of the parallel machine. It allows users to define distributed meshes, to declare values attached to the entities of the meshes (e.g. temperature attached to elements, pressure to the faces, etc.), to exchange values between overlapping entities located at the boundaries of subdomains assigned to different processors, to iterate over the relations of entities (e.g. iterate over the faces of elements), to remesh the pieces of the mesh that need to be, and to redistribute evenly the remeshed mesh across the processors of the parallel architecture. PaMPA relies on the PT-Scotch library [5] for parallel graph (re)partitioning. We validated our method on several isotropic and anisotropic tetrahedral meshes. For that purpose, we combined PaMPA with the MMG3D sequential tetrahedral isotropic and anisotropic remesher [2]. Our test machine is Avakas, a cluster of Intel® Xeon® x5675's running at 3 GHz. On 240 processors, we have been able to remesh an isotropic mesh with 27 millions of elements into a refined mesh comprising more than 600 millions of elements, in less than 35 minutes. Also, we used this approach within an adaptation loop to solve a transsonic Eulerian flow around a M6 wing. The system of equations is discretized using residual distribution schemes [1]. Mesh adaptation is guided by a metric based on an *a posteriori* error estimate of the interpolation error [4].

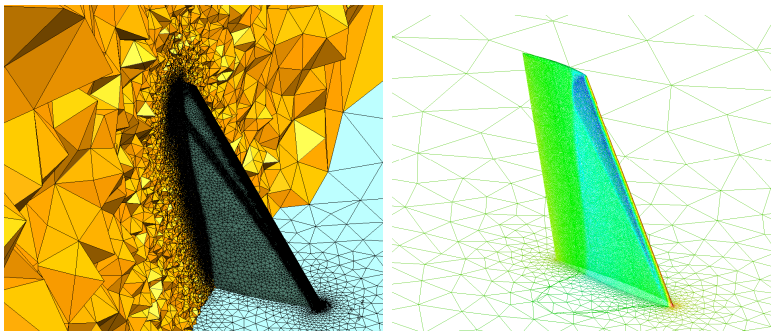

Figure 1: On the left: volumic cut on the adapted mesh; on the right: density field.

REFERENCES

[1] R. Abgrall and A. Larat and M. Ricchiuto. Construction of very high order residual distribution schemes for steady inviscid flow problems on hybrid unstructured meshes. *J. Comput. Phys.*, 230(11), p. 4103–4136, 2011.

[2] C. Dobrzynski and P. Frey. Anisotropic Delaunay mesh adaptation for unsteady simulations. Proc. 17th Int. Meshing Roundtable, Pittsburgh, USA, 2008.

[3] C. Lachat. *Conception et validation d'algorithmes de remaillage parallèles à mémoire distribuée basés sur un remailleur séquentiel*. PhD thesis, Université de Nice-Sophia Antipolis, December 2013.

[4] P. Frey and F. Alauzet. Anisotropic mesh adaptation for CFD simulations. *Comput. Meth. Appl. Mech. Engrg.*, 194(48-49), p. 5068–5082, 2005.

[5] F. Pellegrini. Scotch and PT-Scotch Graph Partitioning Software: An Overview. In *Combinatorial Scientific Computing*, Uwe Naumann and Olaf Schenk, eds., CRC Press, 2012, p. 373–406, ISBN 978-1-4398-2735-2.