VI International Conference on Adaptive Modeling and Simulation
ADMOS 2013
J. P. Moitinho de Almeida, P. Díez, C. Tiago and N. Parés (Eds)

MASSIVELY PARALLEL COMPUTATION ON ANISOTROPIC MESHES

H. DIGONNET, L. SILVA AND T. COUPEZ

CEMEF - MINES ParisTech,
Rue Claude Daunesse CS 10207
06904 Sophia Antipolis cedex France

Keywords. massively parallel, anisotropic mesh adaptation, multigrid, finite element simulation

Abstract. In this paper, we present the work performed to enable massively parallel computations over anisotropic meshes and preserve a good efficiency, to take advantage of two main ways to speed-up an application or to improve the precision of the calculation. Firstly, today supercomputers provide access to a very large number of cores (several thousands), but the applications need to be adapted to fully benefit of such a large amount of power and memory. Such calculations can lead to the use of meshes of billion(s) of nodes. Another direction to improve finite element simulation consists in using anisotropic meshes that allow us to refine the mesh only in the direction of interest. In this way, the CPU cost to improve the precision of our simulations is reduced by increasing the number of nodes only in the direction needed and not in an isotropic way. This makes a huge difference, especially in 3d calculations: in the best case, we can improve by a factor two the accuracy using only 2 times much nodes rather than 8 times.
The first step has been to improve the mesher to enable it to generate very large distributed unstructured anisotropic meshes on thousands of cores. The second one concerned the implementation of a massively parallel multigrid solver to allow us to solve large linear systems generated by Finite Element formulations used to solve the Stokes/Navier Stokes equations. All these developments enabled our software, CimLib [1], to solve a 10 billion of unknowns system using 8192 cores in less than 200 seconds.
We first present an original parallelisation strategy for mesh adaptation [2]. It is based on an independent subdomain remeshing under the constraint of blocked interfaces. Furthermore, a new partition of the mesh is done in order to move the interfaces. Then, these two steps are iterated until we obtain a good mesh quality everywhere. The fact of using the original mesher in a parallel context rather than explicit parallelization provides easier anisotropic mesh adaptation to anisotropic mesh easier. Some examples with large distributed anisotropic meshes will be shown in 2d, as well as in 3d. In this work, we also describe the optimisation made to obtain a good parallel performance by reducing the complexity of the algorithm to a linear one only in the part where the mesher relaying needs to work and not on the whole mesh. This has been done by introducing a “permute-cut-and-paste” procedure: the ”bad quality” submeshes are extracted and then remeshed and pasted back to the complete mesh.
Then, we describe a parallel multigrid implementation for solving very large systems of equations. Even if the efficiency of our iterative parallel solver (GMRES,CR) with complex

parallel preconditionner ILU(k) inside the PETSc library [3] is good and close to the ideal Speed-Up , the non linear complexity of such algorithms fails to solve systems of billions of unknowns. Therefore, a multigrid method is needed. PETSc provides some facilities to implement multigrid algorithms, but for unstructured meshes the user still has to provide at each level the smoother operator, as well as the restriction/injection operators between each level. Building the smoother consists of the assembly step of the classical resolution and did not need any new developments, but building the restriction/injection operator for two unstructured distributed meshes revealed much more difficult. The sequential algorithm implemented was a simple one: for each node “n” of a mesh “A”, find the element of then mesh “B” containing “n” and compute its barycentric coordinates; for finding the element, an octree algorithm is used to speed-up de process. For the parallel version, the difficulty consisted in the fact that the node and the element may not be on the same core. An additional problem is due to the volume of data exchange: thanks to the mesh adaptation strategy chosen, most of the mapping of the two partitions is quite good and only a small percentage of nodes need to be communicated to other cores. Even with small exchange, using several thousands of cores was not straight forward : negligeable things on a small amount of cores can lead to a « blottleneck » on thousands of cores. Several optimisations will be presented and analysed over a large amount of cores.
Finally, parallel performance analysis is done with a massively parallel computer cores. Performances show a very good scalability of our library, including mesh adaptation and linear solver resolution. Multigrid implementation has
optimized the CPU time, as well as memory consuming of the supercomputer used during the numerical simulations. The use of the parallel visualisation software ParaView [4] allowed data processing directly on a end-user site [5].

REFERENCES

[1] Digonnet H, et al. “Cimlib: a fully parallel application for numerical simulations based on components assembly”. NUMIFORM 2007, Porto, 2007.
[2] Coupez, T., et al. “Parallel meshing and remeshing”. Appl. Math. Modelling 25(2), 83–98, 2000.
[3] Balay S, et al. “PETSc Users Manual ANL-95/11 - Revision 3.0.0”, Technical Report, Argonne National Laboratory, 2008.
[4] Henderson, A. “ParaView Guide, A Parallel Visualization Application”. Kitware Inc., 2007.
[5] Digonnet H. “Making Massively Parallel Computations Available for End Users”, PARENG 2011, Ajaccio, Corsica, France, 2011