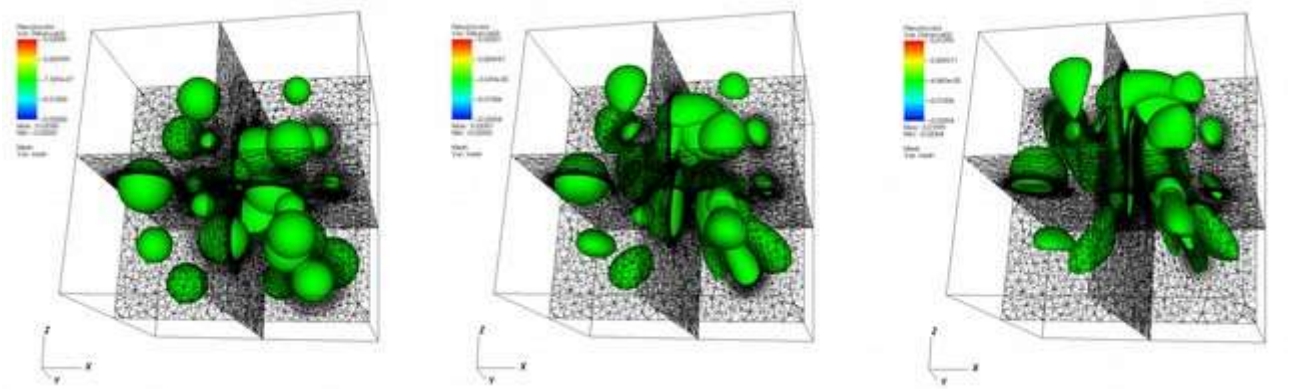
Task-based parallelisation of a 3D anisotropic mesher

Hugues Digonnet, Luisa Silva
Institut de Calcul Intensif – École Centrale de Nantes

I) Context

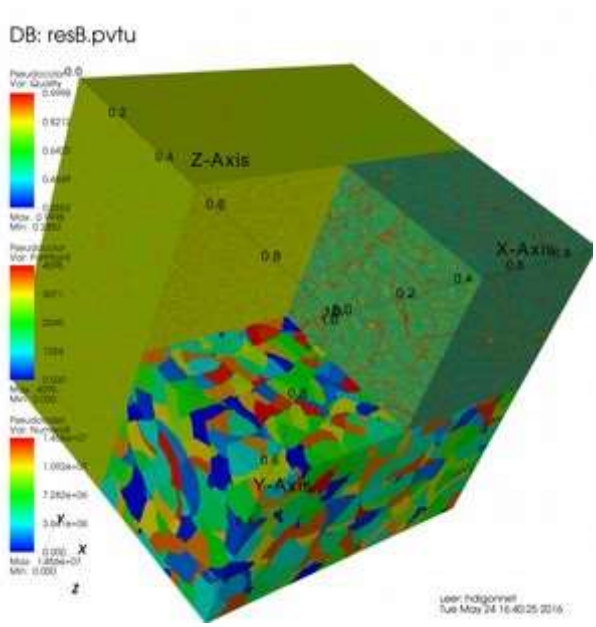
Mesh adaptation is one of the fundamental numerical tools in finite element numerical simulations to considerably reduce the size of the problems to be treated (and therefore the computation time) for a given precision.



IciMesh is a mesh generator, under Open Source license, based on topology optimization and quality criteria that generates good quality simplex type elements (2d triangles and 3d tetrahedra), in an unstructured and non-hierarchical way. Mesh adaptation consists of local successive optimizations (a good point for parallelization) that end with the global optimization of the quality of the mesh, given a wanted mesh size field. The concerned mesh generator generates both isotropic and anisotropic meshes adapted to a heterogeneous mesh size field (or a metrics one in the anisotropic case).

A parallel version of this mesh generator already exists and uses the MPI parallelization library. This parallel mesh generator allowed us to use the power delivered by cluster-type supercomputers and to be scalable up to several hundred thousand cores on several architectures: IntelXeon (Curie, Occigen), BlueGene / Q (Turing / JuQUEEN)) but also ARM (Prototype MontBlanc). The performances obtained are very good ont the execution on the whole of JuQUEEN (458,752 cores) and generation of meshes with several billion nodes and elements. Using 16,384 cores from Occigen's supercomputer, we were able to generate:

- a 2d mesh of 75 billion nodes and 150 billion elements.
- a 3d mesh of 20 billion nodes and 120 billion elements.



Nevertheless, current and future supercomputers increasingly include co-processors or GPU type accelerators. Likewise, CPUs have more and more cores and the strategy of one MPI process for each core may no longer be the most optimal, especially inside a compute node. The objective of this project is therefore to be able to develop and carry out a task-based parallelization strategy of

the sequential "motor" core of our parallel mesh generator with the objective of being able to choose the parallelization granularity by combining a distributed parallelization of MPI type with parallelization with shared memory on CPU, co-processors or GPU.

II) Strategy

In IciMesh, the remeshing operators are very local: there is a topology optimization around a node or an edge. The order in which these operators are executed is not important, so the algorithm is nicely parallelized.
The current parallelization via MPI consists in cutting the mesh into subdomains and applying to each of them the sequential mesh generator under the constraint of not touching the interfaces between them. A parallel repartitioning phase moves these interfaces and, by iteration of the previous steps, we get an adapted mesh over the entire domain. From a global point of view, this can be seen as a reordering of the main operators so as to be able to exploit the locality of the data and, consequently, the parallelism.

In the case of parallelization by spots, we will follow an identical strategy but in a context of shared memory. In the currently sequential engine of the mesh generator, it will be a question of over-partitioning the domain so as to define different zones which can be treated by the atomic operators independently. Two strategies are possible:
- over-partition and define fairly compact areas which will be assigned to a thread and re-mesh under the constraint of not touching the interfaces between the areas, then finish by processing the interfaces (identical to the pure MPI approach but in a context shared memory). This strategy seems, however, limited to a fairly low number of threads (CPUs);
- color each node and edge of the mesh so as to be able to execute on each node or edge of the same color the main operators without exchange between them. If each color contains a large enough number of nodes or edges (> 1000, 10,000), operators can be performed on different XeonPhi or GPU computation units. The final algorithm will be to carry out a loop on the different colors and, at each iteration, to launch in parallel the remeshing kernel adapted to the desired hardware.

III) Expected results

Mains results concern upscaling on the computation capabilities : this will allow mesh tools to be used on Exascale supercomputers, composed of CPUs, but also of GPUs.

Task based parallelization also makes it possible to consider the implementation of a wide variety of mesh kernels on different architectures and programming paradigms: CPU thread, XeonPhi, ARM, Cuda GPU, OpenCL GPU and to be able to make the dynamic choice of the latter depending on the most suitable resource.

IV) Bibliography

H. Digonnet, T. Coupez, P. Laure L. Silva, « Massively parallel anisotropic mesh adaptation », International Journal of High Performance Computing Applications, 2020

N. Möller, E. Petit, L. Thébault, Q. Dinh, « A Case Study on Using a Proto-Application as a Proxy for Code Modernization », Procedia Computer Science, Volume 51, 2015, Pages 1433-1442

H. Digonnet, « Extreme Scaling of IciPlayer with Components: IciMesh and IciSolve », JUQUEEN Extreme Scaling Workshop 2016, JSC Internal Report. 2016, Pages 31-36