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
BP 207
06904 Sophia Antipolis cedex
France

Key words: Massively parallel computation, Anisotropic mesh adaptation, Multigrid

Abstract. In this paper, we present developments done to obtain efficient parallel computations on supercomputers up to 8192 cores. While most massively parallel computation are shown using regular grid it is less common to see massively parallel computation using anisotropic adapted unstructured meshes. We will present here two mains components done to reach very large scale calculation up to 10 billions unknowns using a muligrid method over unstructured mesh running on 8192 cores. We firstly focus on the strategy used to generate computational meshes and in particular anisotropic ones adapted to capture a quite complicated test function. Then we will briefly describe a parallel multigrid method. Performance test over a large range of cores from 512 to 8192 cores is then presented using the French national supercomputers Jade and Curie. The last section will present a calculation done on smallest number of cores on our own cluster, but using more realistic data obtain directly from experimentation. The goal is to be able to realize such kind of simulation on really complex micro structure obtain by tomography at a larger scale.

1 INTRODUCTION

From the last years computers power increase mainly by cores multiplication (rather than clock rate rise) inside each CPU (classically 8 or 16 cores in 2012) and also for the supercomputers of the top500 that contain several thousand to more than one million cores. this context impose to develop fully parallel softwares to at least be able to take advantage of new computer hardware. If we look at numerical simulation, computation times are always to important and has soon as they decrease the user want to have more precise computation by introducing more physical properties or by looking in computing lower scale calculation to improve the material behavior. One of the mains way to reduce

computation time is to use parallel computers and in the ideal case divide the CPU time by the number of cores used. For that it is then necessary to develop algorithms that could run on massively parallel computer containing hundred, thousand or even more cores like ones present in the top500 list of supercomputers. We present here some algorithms done to allow our application CimLib to run over massively parallel supercomputer using up to 8192 cores. The first section will focus on parallel mesh adaptation with anisotropic elements taht lead to generate smaller meshes for a given precision. A second part will briefly present parallel multigrid method implemented to reduce the complexity of the algorithm used to solve linear systems and fully utilize the power and memory given on a computer. We present parallel performance on solving incompressible Stokes equation using from 8 to 8192 cores and enable resolution of a linear system containing more that 10 billions unknowns by using 8192 cores. The last section will present more reasonable simulation but on a more realistically and complex case. It consists in computing a flow through a micro structure given directly from a real one using tomography.

2 parallel anisotropic mesh adaptation

2.1 parallel mesh adaptation strategy

We briefly present here the methodology used to parallelize our unstructured and non hierarchic mesher, MTC [1]. At the beginning of this work, the sequential mesher already exist and is likely to evolve (we recently add anisotropic mesh size [2]), so we want to keep it as it was, by not including parallel instructions inside. So, we have not parallelized directly it, but only give the oprtunity to use it inside a parallel context. The strategy is then to combine local remeshing (inside each processor domain, freezing the interfaces between the partitions) [3] and parallel mesh repartitioning [4] to move unremeshed interfaces inside domains for the next step, in order to be able to apply the remeshing procedure on theses zones during the next phase. Figure 1 illustrate this strategy on a simple geometry using seven processors.

2.1.1 optimization and performance

This strategy of parallelization leads to several iterations (depending of the space dimension) between the mesher and the repartitioner, but the work to be done at each iteration decreases quickly. For example, in a 2d space : the first remeshing step is proportional to a surface, the second to a line and the last one to a point. For the repartitionner, as we only need to move bad quality zones inside the domain in order to remesh it, the proportion of elements and nodes to be migrate across the processors decreases in the same way. For that reason the cost of using a multi steps strategy must be really close to a single remeshing step: the global work done by the mesher is approximately the same, and only some few mesh migrations are done during repartitioning steps. For that purpose an optimization of the update procedures in case of small changes has been done (the time spent to remesh or migrate 10 elements is then close to zero). This optimization

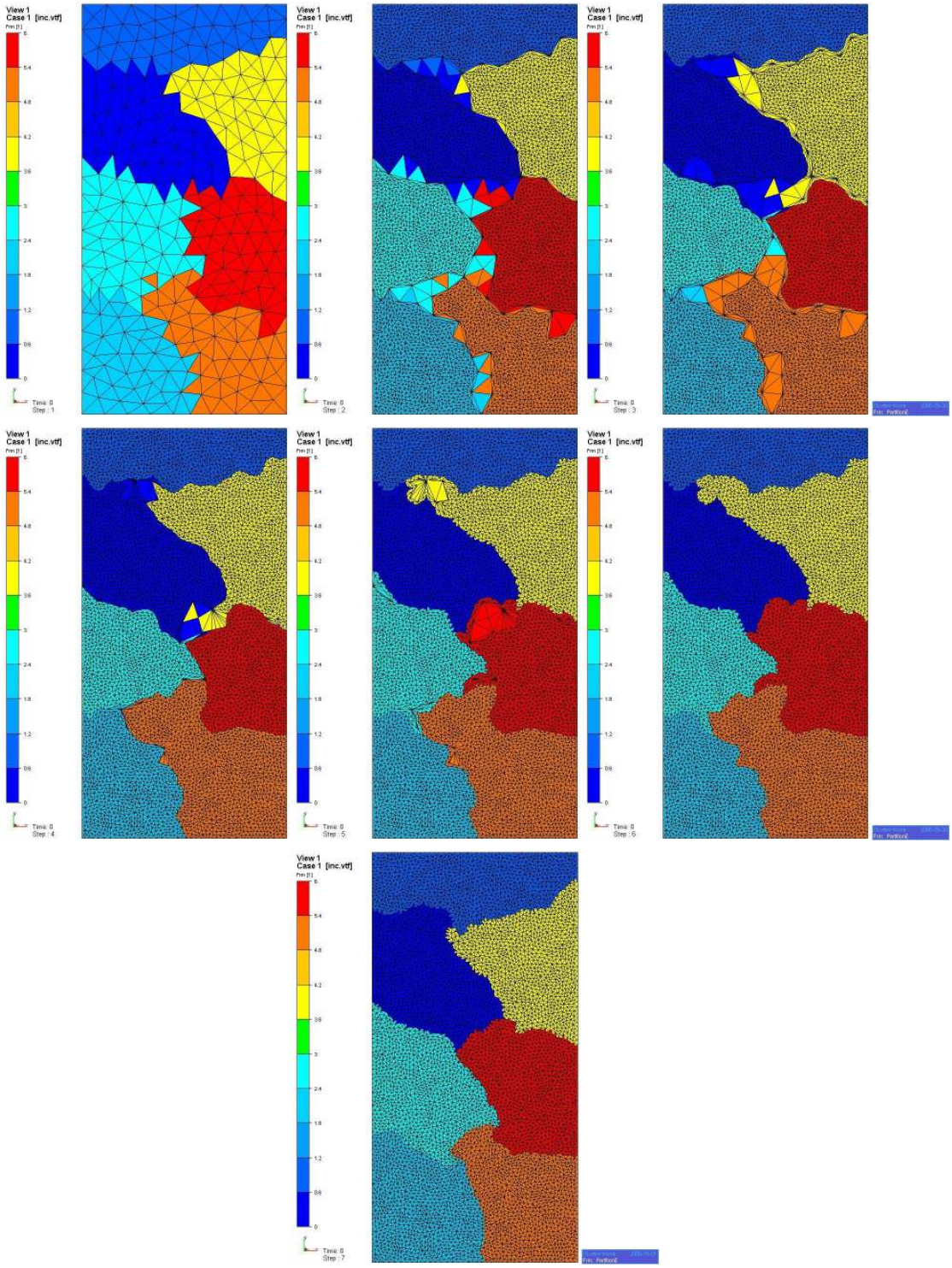

Figure 1: Illustration of the parallelization strategy used for the mesher in a 2d case with a refinement by 4, over 7 processors. Each image represent one step of the iterative methodology consisting in successively call parallel remeshing with frozen interfaces and parallel repartitioning, until convergence to good quality mesh. The last repartitionning phase is done to load balance the work by considering finite element resolution to come.

is based on a permute-copy-past algorithm that leads to reduce the complexity from N (the data size) to m (the moving size) with $m \ll N$. Permutation removes any copy of all the data by making instead some few permutations. After that, work is performed only in the cutting zone and pasted back after being done. This optimization was essential to maintain the iterative strategy costless, and so provide a good parallel efficiency to the mesher.

2.2 anisotropic error estimator

Anisotropic meshes are used to keep the same accuracy of isotropic meshes but with a smaller number of nodes and elements by allowing different mesh size depending of the direction. The use of such anisotropic meshes could reduce drastically the size of the mesh and in particular in 3d where a uniform refinement of a factor 2 increase the number of nodes by a factor 8 that has to be compared to only a factor 2 if the refinement is only needed in one direction. For anisotropic mesh adaptation we need use a mesh size definied by a scalar for isotropic mesh and a symmetric positive defined tensor (named metric) for anisotropic one. In order to enable mesh adaptation we need to provide to the mesher a continuous field of mesh size one simple solution is to give a P1 interpolate field where a local mesh size is defined at each mesh nodes. The way to compute this field could be a average of the elements metrics containing this node with the difficulty of choosing a good average value for tensors. A more direct way to defined this metric consist in using the distribution tensor describe in [2] and given by :

$$\mathbb{X}_i = \frac{1}{|\Gamma(i)|} \sum_{j \in \Gamma(i)} X^{ij} \otimes X^{ij}$$

where $X^{ij}$is the vector $\overrightarrow{X_i X_j}$ between two neighbor nodes i and j and $\Gamma(i)$ the neighbors nodes of i.

This tensor give the anisotropic size of the envelope containing of the elements belonging to node i, and the natural metric for the mesh is then simply defined by $\mathbb{M}_i = \mathbb{X}_i^{-1}$ to insure that using this metric this envelope has an unit length.

The anisotropic error estimator is well presented in [2] and we will here only present the mains ideas. To be able to build a anisotropic error estimator we first need to be able to define an anisotropic error and for that we will use the same technique of the distribution tensor. For a given Level Set we will compute the error over nodes edges as :

$$e_{ij} = \left| G^{ij} X^{ij} \right|$$

where $G^{ij}$is the gradient variation of the level set function across the edge ij.

With this anisotropic error and given a number of nodes mesh we could define (for demonstration refer to paper [2]) a metric tensor to generate an adapted mesh that will uniformly distribute the error across its edges. The metric is given by :

$$\mathbb{M}_i = \frac{1}{|\Gamma(i)|} \left(\sum_{j \in \Gamma(i)} s_{ij}^2 X^{ij} \otimes X^{ij} \right)^{-1}$$

with

$$p \in [1, d]$$

$$\lambda = \left(\frac{\left(\sum_i \sum_{j \in \Gamma(i)} e_{ij}^{\frac{p}{p+2}} \right)}{A} \right)^{\frac{p+2}{2}}$$

where d is the space dimension, A is the global number of edges in the mesh and p is the power of the power law estimation for the number of edges generated in the mesh when we want to divide one edge. For example: for a one direction anisotropic mesh divide an edges in that direction by a factor 2 will only multiply by 2 the global number of edges in the mesh and then $p = 1$; in case of an isotropic mesh divide an edge by two will generate eight times more edges so $p = 3$.

2.3 application test case

We present in this section an application test case adapted to massively parallel computation. It consist in computing a adapted mesh to allow a good representation of a complicated function defined by :

$$f(x) = a \circ a(x - x_0) + a \circ a(x - x_1)$$

where

$$a(x) = tanh\left(Esin\left(\frac{4 * N + 1}{2} \Pi \|x\| \right) \right)$$

and

$$x_0 = (0, .., 0) \qquad x = (1, ..., 1)$$

E and N are two parameters that can be adjusted. They respectively influence the thickness and the complexity of the details present in the function: if E increase the size of the detail decrease and if N increase then each detail contain become more and more complex. The figure 2 represent the test function on an uniform square mesh containing around 50000 nodes for E,N parameters equal to 1, 2 and 4. We notice that for E=4

and N=4 the uniform mesh start to be to small to obtain a good representation of the function f.

As soon as these parameters increase the function become really difficult to capture and we need to activate both the anisotropic mesh adaptation and parallel computation to generate meshes fine enough to represent the test function. The bottom of figure 3represent the 23 millions nodes anisotropic mesh generate to represent the test function with E=16 and n=6 on the unit square. It clearly illustrate the benefit of being able to execute anisotropic mesh adaptation in parallel, here using 256 cores.

The figure 4 represent the same test case in a 3d context using smaller parameters $N = 1$ and $E = 4$. E and N parameters. The 3d adapted mesh contain 2 millions nodes and have been generate using 50 mesh adaptation iterations using 64 cores of Jade supercomputer in 15.5 hours .

3 Multigrid solver

Using powerful computers containing a large numbers of cores could lead using very big meshes with several millions nodes. Solving physical equation using for example the finite element method will ends in solving very large linear systems. Using iterative methods like Krylov ones will parallel preconditioner give good parallel performance but the algorithm complexity (around $\mathcal{O}(n^{3/2})$ in 3d) become a bottleneck as soon as the number of unknowns start to be important (taking a problem 2 times bigger and using 2 times more cores the time spent to solve will be greater than 2). To over come this difficulty we have implemented a parallel multigrid solver using the framework give by PETSc [5]. To do this we need to provide for each level the system to be solve and also the interpolation/restriction matrix operator. Of course all these matrices must be build and store in a distributed way to allow using a large number of cores. More details are presented in [6] but a particular attention have been given to reduce the communication during the construction of the interpolation operator using a parallel octree localization algorithm and pixel mask filter distributed over processors to limit false positive external nodes detection. The figure 5 present parallel performance of the multigrid resolution for Stokes equation over a 2d square mesh of 800 millions nodes and using 512 to 4096 cores. Time spent to solve the system decrease from 96.7 seconds with 512 cores to 17.7 seconds with 4096 cores, in the same time the number of multigrid iteration only increase from 11 to 13 and leads to a speed-up of 5.46.

Even if a good parallel performance of the multigrid solver is important, the main interest is the scalability of this approach that make us able to solve very very large systems. An ideal scalability will be give by a algorithm that will use exactly the same computing resources for a given per core problem size independently of the number of cores uses. A scalability test have been done from 8 to 8192 cores on the multigrid solver. The results are very interesting : the memory used per core during the process stay nearly constant (from 1.94 to 2.14 GB) as well as the time spent to assemble all levels systems (from 9.7 to 11 seconds) and the multigrid iterations (from 13 to 11); only the resolution

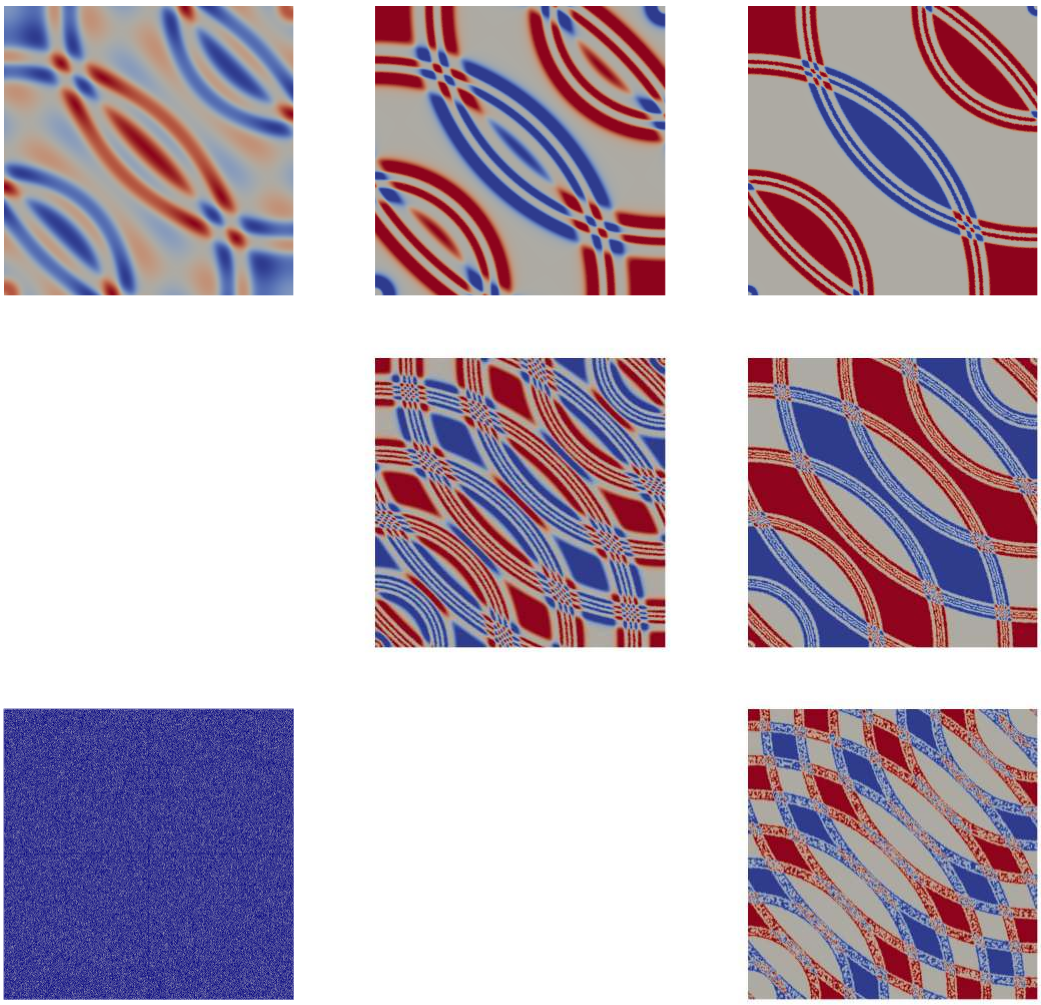

Figure 2: Influence of the two parameters N and E on the complexity of the test function. This test function is plot on an uniform 2d mesh containing 50 000 nodes. From the top to the bottom the parameter $N = 1, 2, 4$ and from the left to the right the parameter $E = 1, 2, 4$. On the left bottom the uniform mesh used to plot the test function.

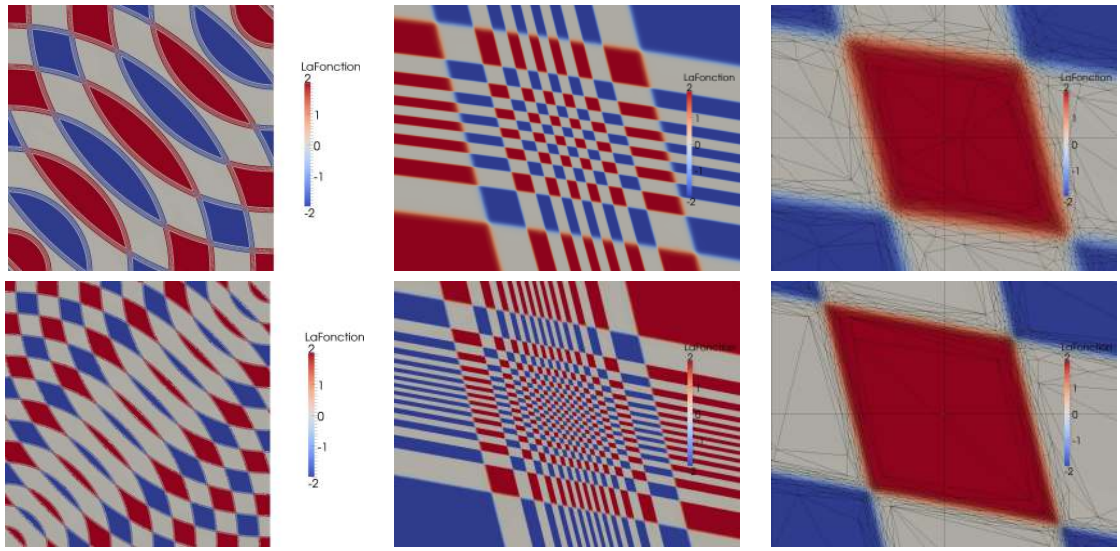

Figure 3: Anisotropic mesh adaptation around the test function $f(x) = g \circ g(\|x - 0\| + g \circ g(\|x - 1\|)$ with $g(x) = tanh(Esin(\frac{4*N+1}{2}\pi x))$ with 2 sets of parameters : on the top N=3, E=16 and at the bottom $N = 6$, $E = 16$. For these two sets we present : the function on the unit square, one cross detail and a zoom on the mesh for a deep detail inside the cross detail. The 2d adapted meshes containing respectively 1 million and 23 millions nodes. They where build in respectively 576 seconds over 128 cores and 3 hours on 256 cores of Jade supercomputer. The smallest mesh size is around receptively 10^{-5} and 10^{-6}.

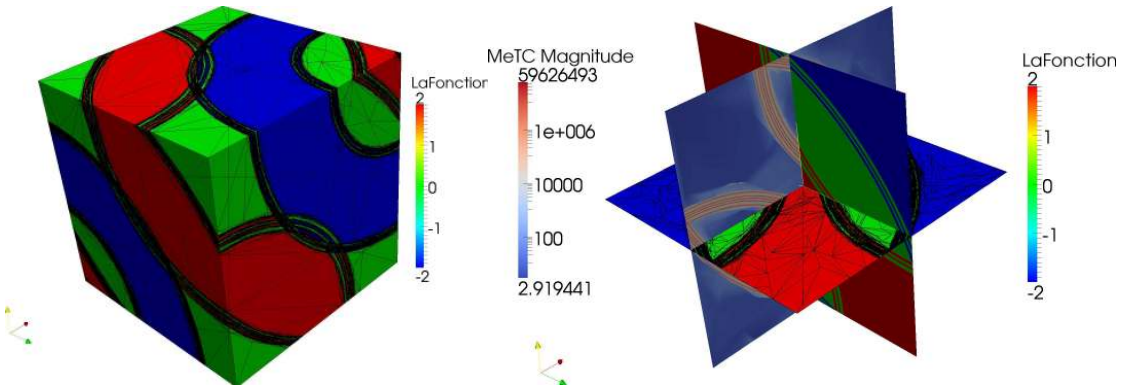

Figure 4: Anisotropic mesh adaptation in 3d around the test function with $N = 1$ and $E = 8$ for a final mesh containing 2 millions nodes.

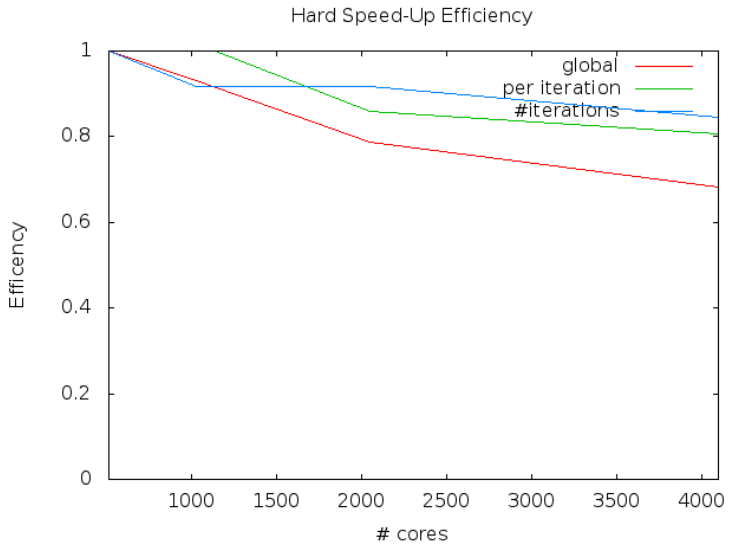

Figure 5: Parallel efficiency for the multigrid resolution using 512 to 4096 cores. The number of multigrid iteration needed to converge stay almost constant (efficiency close to one) has well as the parallel efficiency per iteration. At the global point of view the parallel efficiency is a bit worse due to the combination of per iteration parallel performance and increasing number of iteration but still very good (close to one) has the problem size per core decrease as number of cores used increase.

time increase a beat more from 90 to 148 seconds. This augmentation of 50% of the time spent to solve the problem between 8 and 8192 cores is still reasonable and may also be reduce using more multigrid level over 8192 cores as the coarse level system size may become to big (around 600 000 unknowns) to be solve in a efficient way. To conclude this multigrid solver implementation have been able to solve a global system (based on the Stokes equation) containing more than 10 billions unknowns using 8 levels in 158 seconds using 8192 cores and 17.5 PB of memory.

4 From real to virtual

In this section we present some work done to enable making simulation using real micro structure. Today tomography could produce some really nice (well defined but also heavy) image of real material as shown on the left of figure 6 [7]. This image is constituted of a large number of gray level voxel (equivalent to pixel in 2d), the back ones represent the polymer matrix and white one solid fibers. From this image we extract the an isosurface mesh (shown on the right of figure 6). Depending of the image definition the number of voxel and thus the number of faces in the isosurface mesh could become really important and may need to be executed in parallel using some small enough part of the whole domain using some similar technique as octree. Once we have the isosurface mesh, it could be immerse into the computational mesh by computing for all the mesh nodes the distance to the isosurface that give a level set representation of the micro structure [8]. To improve the

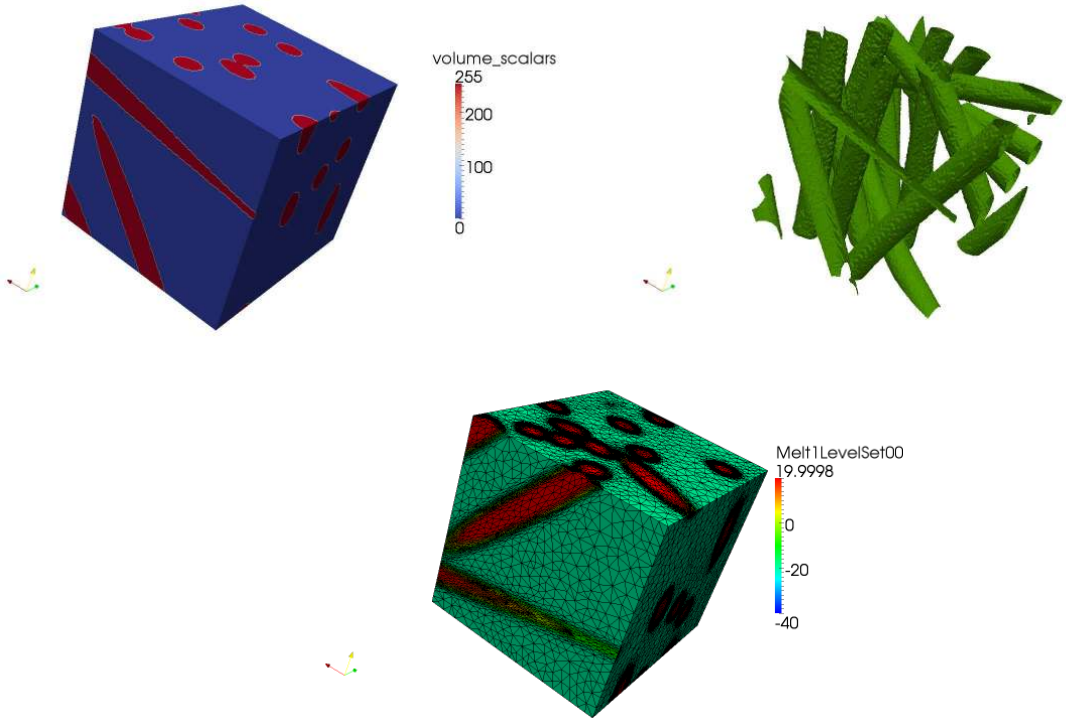

Figure 6: From real to virtual: a large 3d voxel image of the microstructure obtain by tomography and treated with a segmentation algorithm, its surface mesh given by the isosurface 127, and finally the computational domain with an anisotropic adapted mesh.

immersion of the microstrure we could lauch some anisotropic mesh adaptation procedures that will provide the computational adapted mesh as shown in bottom of figure 6.

If the previous figures 6were done to illustrate the mesh adaptation to the real microstructure using anisotropic mesh, using bigger image for the microstructure generate very big mesh and justify using massively parallel computer for both anisotropic mesh adaptation but also for computing the solution. Figure 7 represent the computed flow using Stokes equation across the immerged microstucture. Computation was done using 96 cores on a 10 millions nodes adapted mesh to a microsuture image containing 900x900x220 voxels. This is this type of computation than we plan to do in a close future but over much more larger image with around 4000x4000x4000 voxels given by new tomographic acquisition. For being able to do that all the computational chain need to be run in parallel from the image generation to the visualization of the computed results.

5 CONCLUSIONS

In this paper, we have presented the parallelization strategy for the mesher that consist in iterate between a remeshing step with frozen interfaces and repartitionning step used to migrate interfaces. In that way the mesher engine stay sequential and do not contain any

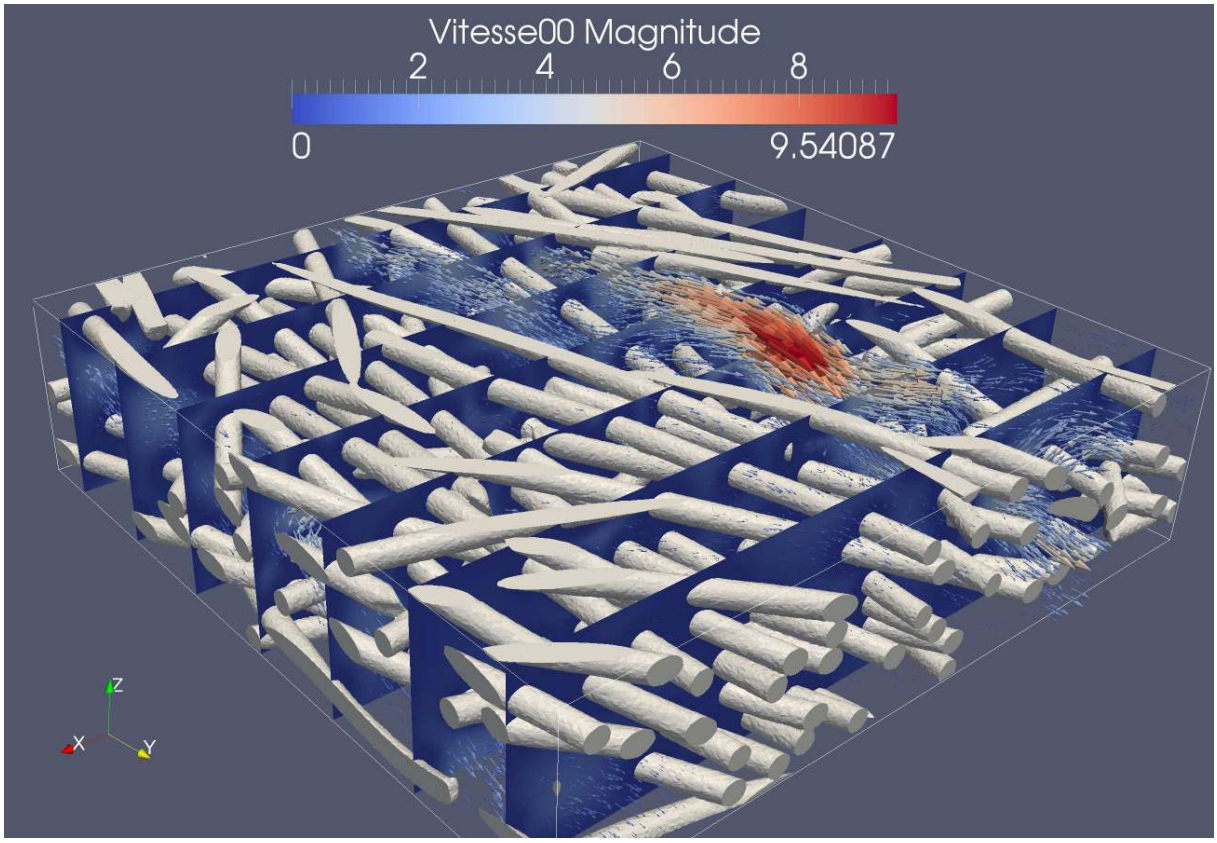

Figure 7: Large numerical simulation of the 3d flow across a microstructure done using a big 900x900x220 tomographic image. Flow is compute over a 10 millions nodes adapted mesh using 96 cores. The isosurface represent fibers microstructure, vector the velocity field.

parallel instruction so it could still be developed (introduce anisotropic mesh) without any specific knowledge. The implementation of a permutation cut and past optimization gives excellent parallel performance as well as a better quality control for meshes generated. Some examples of anisotropic distributed meshes adapted to well represent a complicated test function are give and clearly show the interest of using anisotropic mesh rather than isotropic ones : a 23 millions nodes anisotropic mesh adapted using 128 cores can capture the test function with parameters N=6 and E=16 leading to a very complex function, a equivalent representation will need a picture containing around one thousand billions pixels.

A massively parallel multigrid method working on unstructured meshes have been also presented. The construction of interpolation/restriction operators between very large distributed meshes and using a large number of cores need a powerful localization algorithm used to determine which element contain a node. For that we use a parallel octree localization method improve by a domain filter define by a pixel mask to reduce false detection of non local nodes. By using this filter we have a important reduction of communication even when domains are unrelated???. Very good parallel performance have been presented for the multigrid approach use to solve Stokes equation on very fine meshes over a large range of cores from 8 up to 8192 cores. The multigrid method developed allow to better use all the resources of massively parallel computers (CPU and memory). Over 8192 cores a global system containing more than 10 billions nodes have been solved with an 8 levels multigrid solver in 148 seconds, using a total of 17,5Po of memory. It is important to notice that for such a resolution we have been oblige to remove the 32 bits integer limit in our code.

Finally we presented some more realistic simulation done on real micro structure obtained by tomography. Tomography of a real material gives a 3d image (voxels) segmented to represent the micro structure. This image is then used to build a Level-Set function. A anisotropic adapted computational mesh is build in parallel to fit the micro structure. Once this adaptation is done we could compute the flow through the micro structure. We have presented here one simulation done over a 10 millions nodes 3d adapted mesh using a micro structure image with 900x900x200 voxels. Visualization of the results has been done using a parallel version of the Paraview [9]software, coupled with distributed results files and using python scripts [10].

Future work we be done to improve and validate our algorithms and be able to scale over tens to one hundred of thousands cores as the 32 bits integer limits have already been removed. We will also look to simulate non linear materials on a larger VER given by new tomographic acquisition leading to 4000x4000x4000 voxels images.

6 Acknowledgments

This work was granted access to the HPC resources of [CCRT/TGCC/CINES/IDRIS] under the allocation 2012-[x2012066109] made by GENCI (Grand Equipement National de Calcul Intensif)

REFERENCES

[1] T. Coupez, Génération de maillage et adaptation de maillage par optimisation locale, *Revue Européenne des Éléments* **9**, 403 (2000).

[2] T. Coupez, Metric construction by length distribution tensor and edge based error for anisotropic adaptive meshing, *Journal of Computational Physics"* **230**, 2391(apr 2011).

[3] T. Coupez, H. Digonnet and R. Ducloux, Parallel meshing and remeshing, *Applied Mathematical Modelling* **25**, 153 (2000).

[4] A. Basermann, J. Clinckemaillie, T. Coupez, J. Fingberg, H. Digonnet, R. Ducloux, J.-M. Gratien, U. Hartmann, G. Lonsdale, B. Maerten, D. Roose and C. Walshaw, Dynamic load-balancing of finite element applications with the DRAMA library, *Applied Mathematical Modelling* **25**, 83 (2000).

[5] S. Balay, J. Brown, , K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith and H. Zhang, *PETSc Users Manual*, Tech. Rep. ANL-95/11 - Revision 3.3, Argonne National Laboratory (2012).

[6] H. Digonnet, Multigrid using Adaptive Unstructured Meshes for Massively Parallel Computation, in *Proceedings of the Third International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering*, eds. P. Iványi and B. ToppingCivil-Comp Proceedings (Civil-Comp Press, Stirlingshire, UK, Pécs, Hongary, 2013).

[7] L. Orgéas, P. Dumont, J.-P. Vassal, O. Guiraud, V. Michaud and D. Favier, In-plane conduction of polymer composite plates reinforced with architectured networks of copper fibres, *Journal of Materials Science* **47**, 2932 (2012).

[8] P. Laure, G. Puaux, L. Silva and M. Vincent, Permeability computation on a REV with an immersed finite element method, *AIP Conference Proceedings* **1353**, Pages 978(May 2011), The 14th International Esaform Conference on Material Forming: ESAFORM 2011- Queen's University Belfast, UK, 27-29 April, 2011.

[9] A. Henderson, *The ParaView Guide: A Parallel Visualization Application* (Kitware, November 2004).

[10] H. Digonnet, Making Massively Parallel Computations Available for End Users, in *Proceedings of the Second International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering*, eds. P. Iványi and B. Topping, Civil-Comp Proceedings, Vol. 95 (Civil-Comp Press, Stirlingshire, UK, Paper 61, Ajaccio, France, 2011).