A massivelly parallel multigrid solver using PETSc for unstructured meshes on Tier0 supercomputer.

H. Digonnet, T. Coupez, L. Silva

École Centrale de Nantes (ECN)
Institut de Calcul Intensif (ICI)
Email : hugues.digonnet@ec-nantes.fr
Web site : http://ici.ec-nantes.fr/

Tier0 supercomputers (top continental supercomputers)

Curie :

- 80 640 cores Intel Xeon 2.7 GHz with 322TB of RAM
- Rmax : **1,359 Pflops**, built in 2012

JuQUEEN :

- 458 752 cores PowerPC 1.6 GHz with 448TB of RAM
- Rmax : **5,0 Pflops**, built in 2012

Liger : (Ecole Centrale de Nantes Tier2)

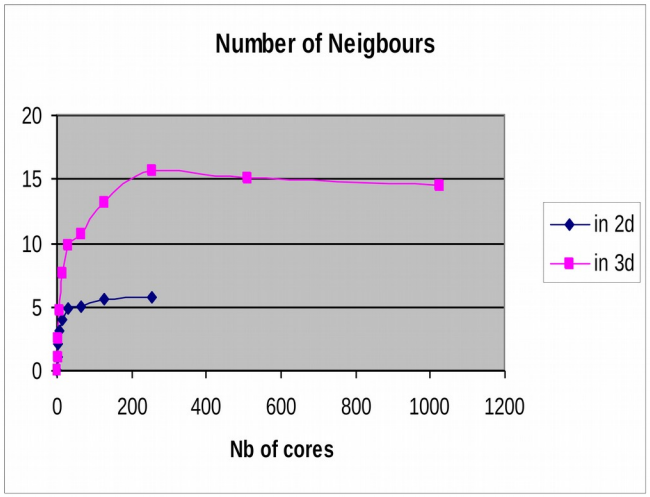
- 6 048 cores Intel Xeon 2.4 GHz with 32TB of RAM
- Rmax : **189 Tflops**, built in 2016

What is massively parallel computation ?

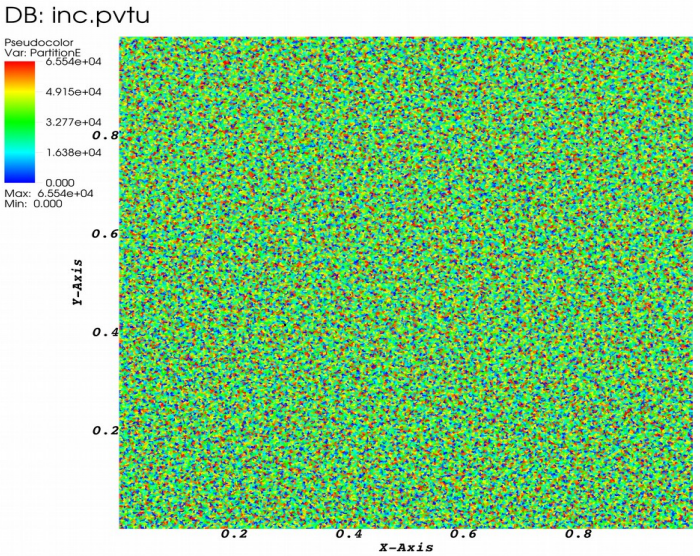
- Hardware : containing a large amount of cores
  Curie : 80 640 cores 2.7 GHz with 4GB/core
  JuQUEEN : 458 752 cores 1.6 GHz with 1GB/core

- Software 1 : when the number of neighbors reach a steady state.

- Software 2 : when the number of cores is similar to the local data size stored on one core.



Computer	# cores	#unknowns	#unknowns/core
Curie	65 536	100 billions	1 500 000
JuQUEEN	262 144	100 billions	375 000

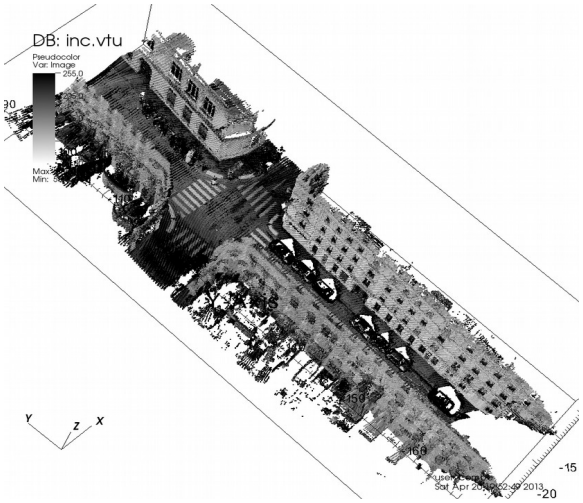


[16384x16384 px]

There is also massive real data !

3d X-Ray tomography image containing several million of voxels

[Solvay]



3d scatter plot with several million of points

[EMP-CAOR]

Surface triangulation with several million of 3d faces?

[ECN-IRSTV]

Plan :

- The context
- Parallel mesh adaptation
- Parallel Multigrid solver
- Unsteady computations
- Conclusions and future works.

Mesh adaptation : Goals

- Use an iterative procedure

  as the mesher strategy (topological improvement)

- Not being intrusive

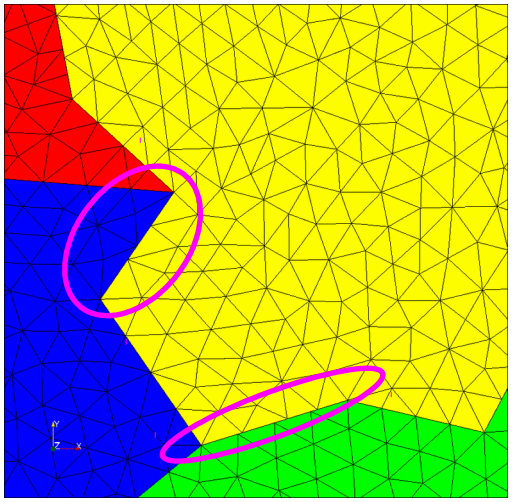
  keep most of the developments sequential

- Deal with isotropic and anisotropic mesh size.
- Use unstructured and unhierarchical simplex meshes

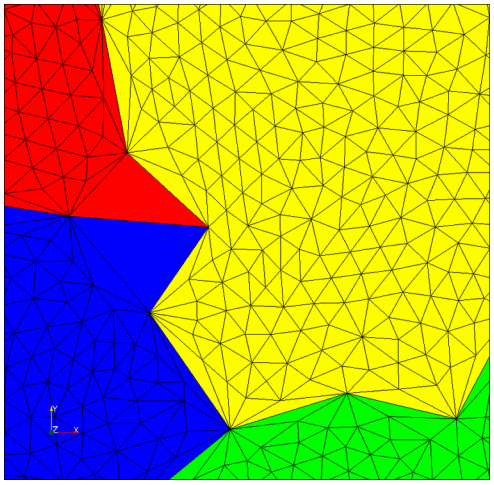
We don't parallelize directly the mesher, but we use it in a parallel context coupled with a parallel mesh repartitioner.

Mesh adaptation : Parallelization strategy

Remesh independently each sub-domain under the constraint of frozen interfaces to keep a conform mesh.



Without constraint : we don't have a global mesh !

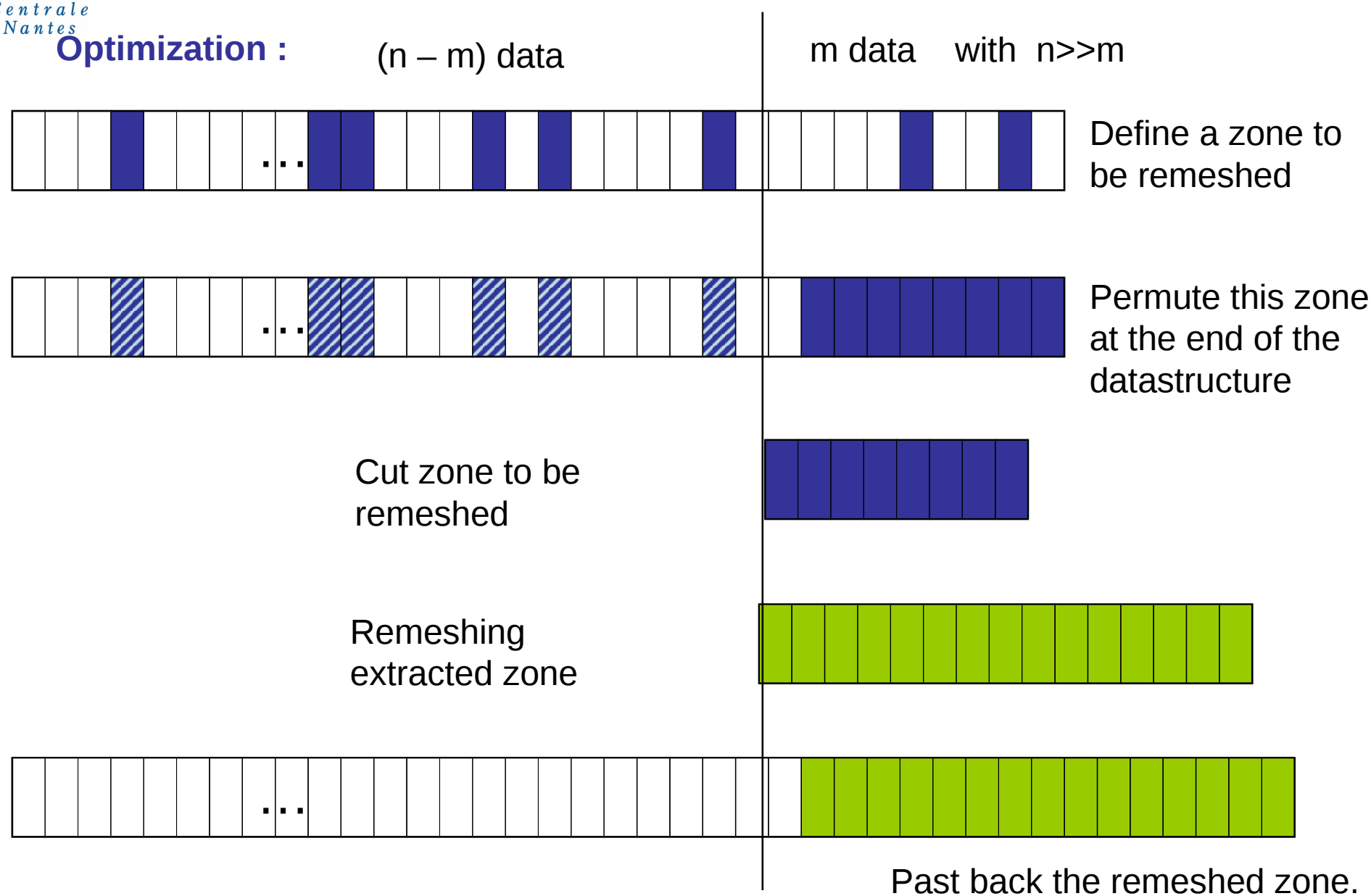


Constrained (frozen interfaces) : we have a global but not perfect mesh

Then move interfaces and iterate.

movie

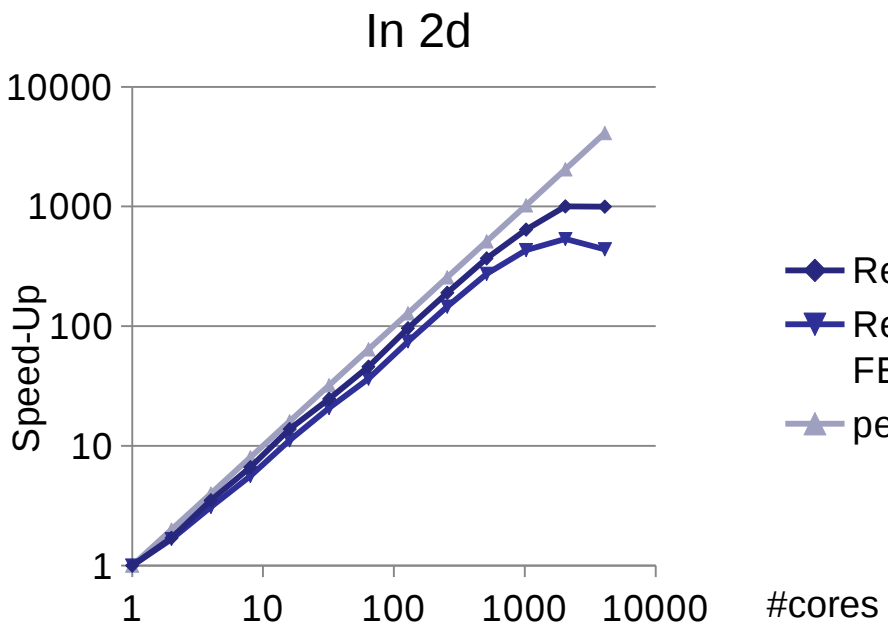
Optimization :

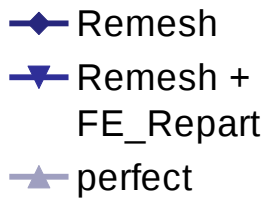


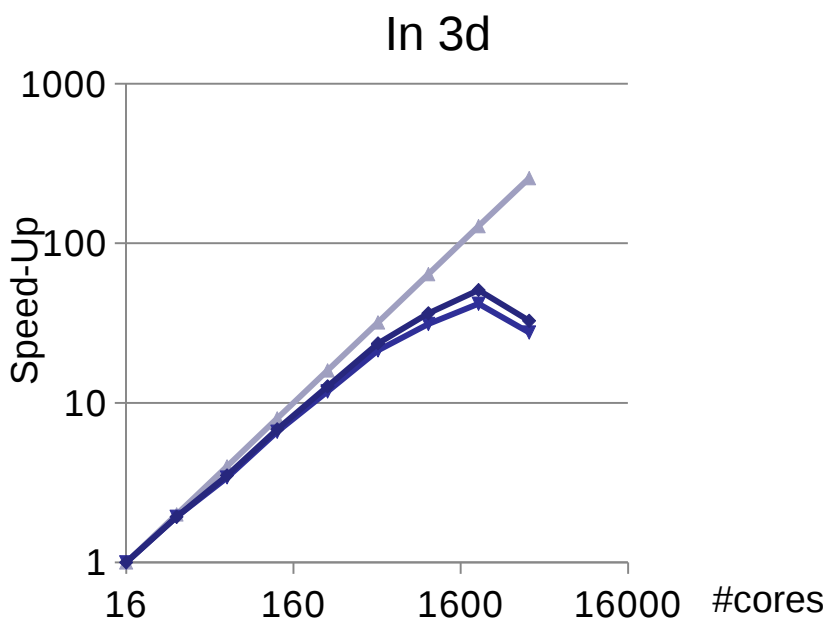
Parallel performance : Strong Speed-Up

Uniform mesh refinement by a factor 2

Space dimension	#cores	Initial mesh #nodes	Final mesh #nodes	Times (s)
2d	1 - 4096	5 million	21 million	3300 to 3,3 (6,6)
3d	16 - 4096	3.6 million	30 million	6800 to 122 (151)







Parallel meshing : Weak speed-up in 2d

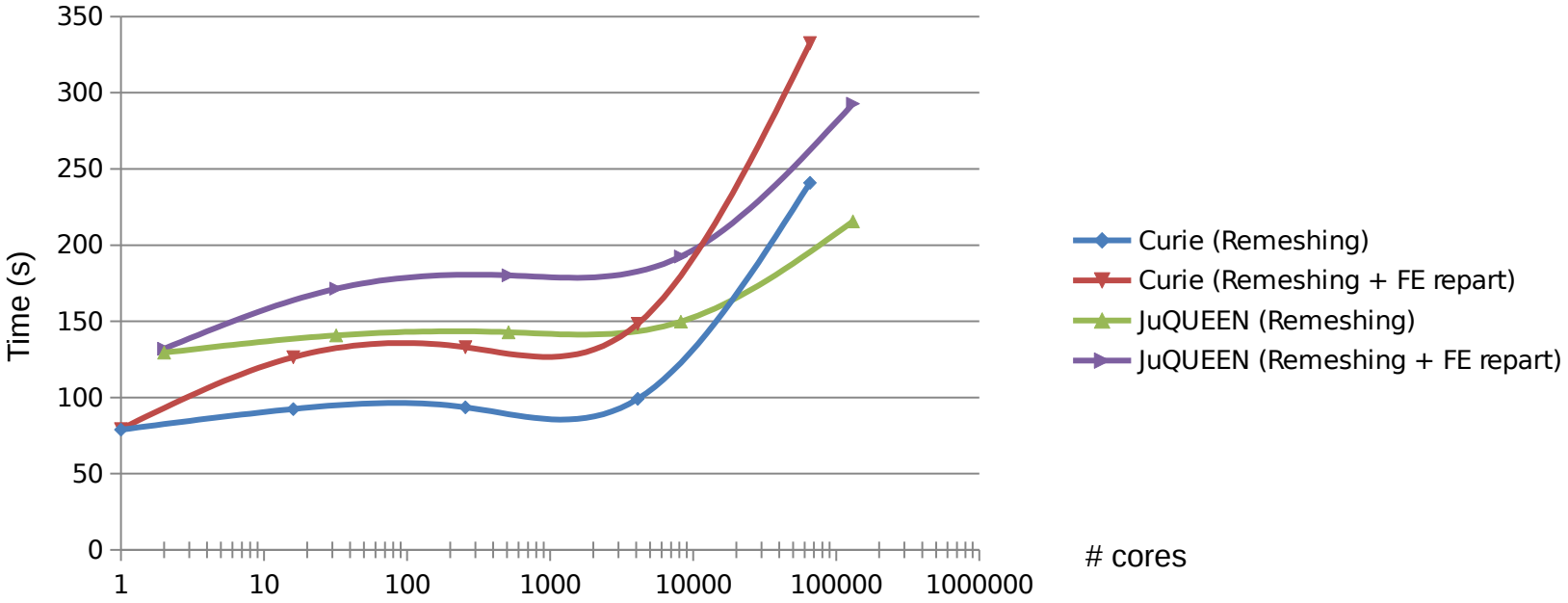
Run from 1 to 131 072 cores, uniform mesh refinement by a factor 4

Constant work load per core :

500 000 nodes on Curie and 125 000 (x2) on JuQUEEN.

Final mesh with 33.3 billion nodes and 67 billion of elements.

Excellent performance up-to 8192 cores, worsening beyond.



Illustrations :

A 3d mesh cube with :

10 billion of nodes

60 billion of elements

Done with IciMesh using 4096 cores of Liger in 1h30m

2.5 million of nodes and 15 million of elements per core.

Quality (shape and size) : min 0.2852, avg 0.7954

Image of 16384x16384 pixels done using Visit over 1024 cores

[H. Digonnet, “Extreme Scaling of IciPlayer with Components: IciMesh and IciSolve”, JUQUEEN Extreme Scaling Workshop, 2016]

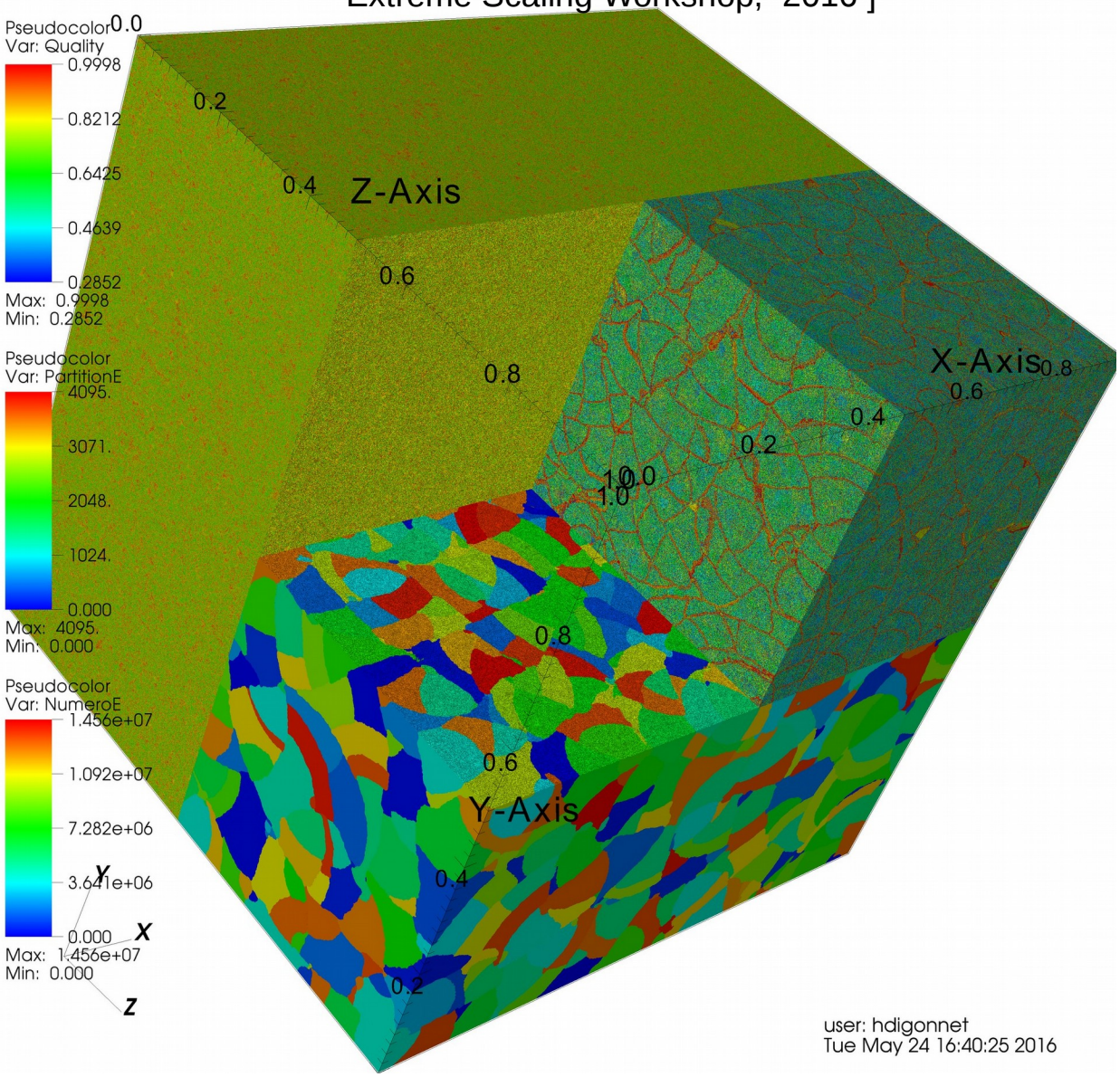


Illustration : static adaptation (with anisotropic error estimator)

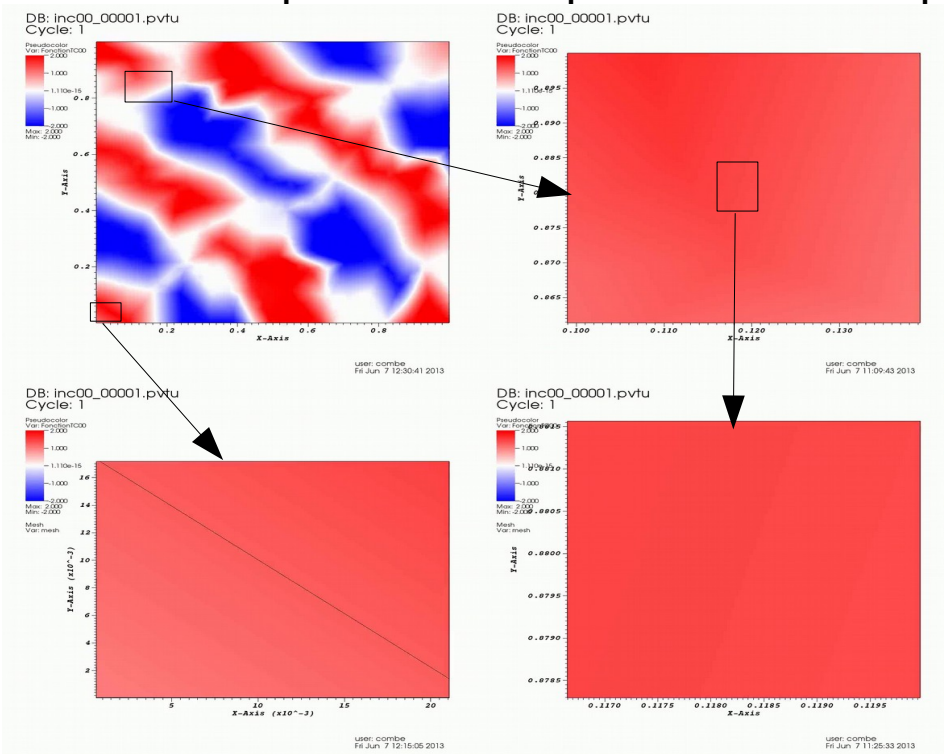
Capturing a complicated test function :

- Almost constant everywhere
- Locally very high variation

$$g(x) = \tanh\left(E \sin\left(\frac{4N+1}{2}\pi x\right)\right)$$

$$f(x) = g \circ g\left(\|x-0\|\right) + g \circ g\left(\|x-1\|\right)$$

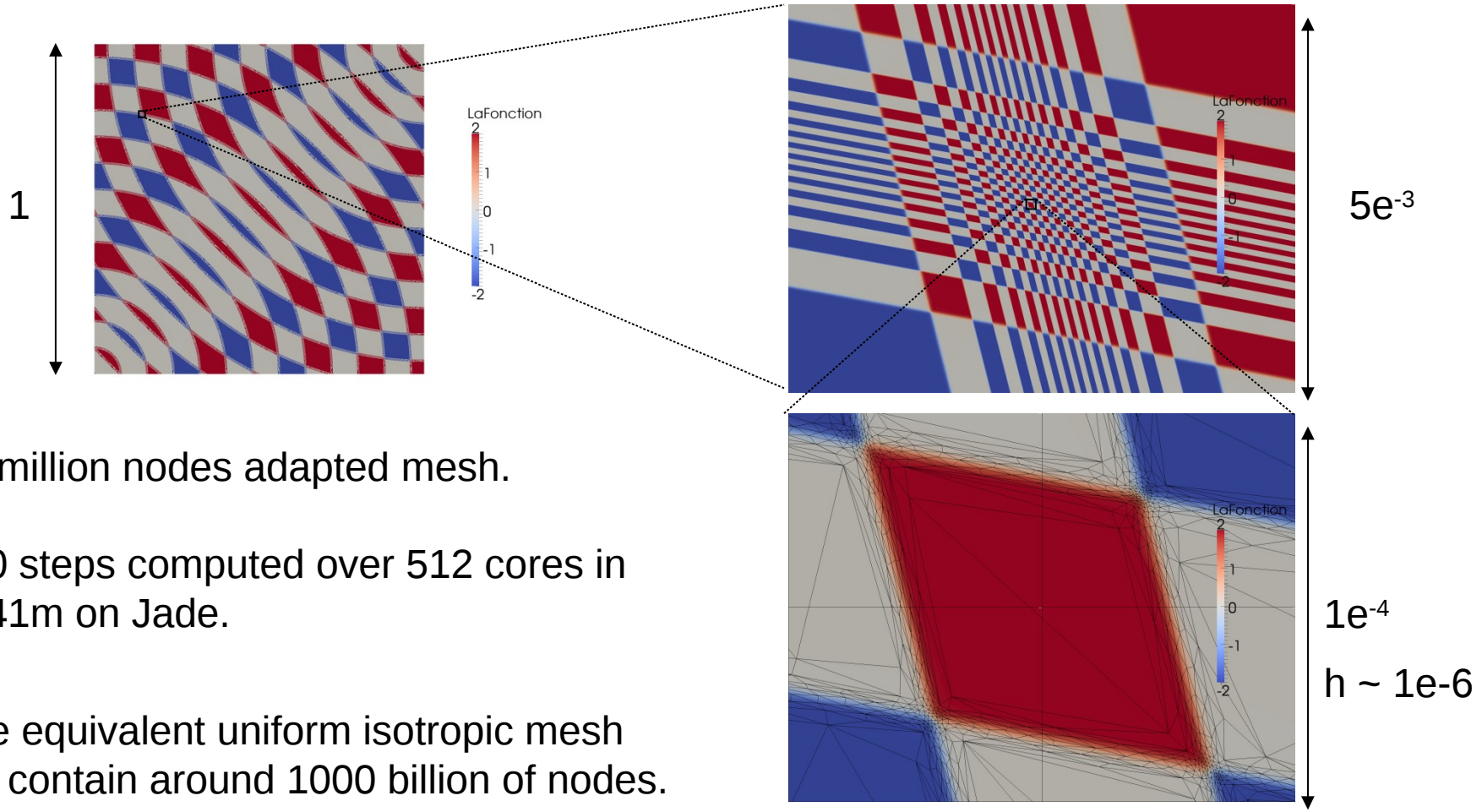
Anisotropic mesh adaptation : 200 steps with a 10 000 nodes mesh



Vidéo (avec E=2 et N=8)

Illustration : static adaptation (with anisotropic error estimator)

Same function with E=16, N=6.



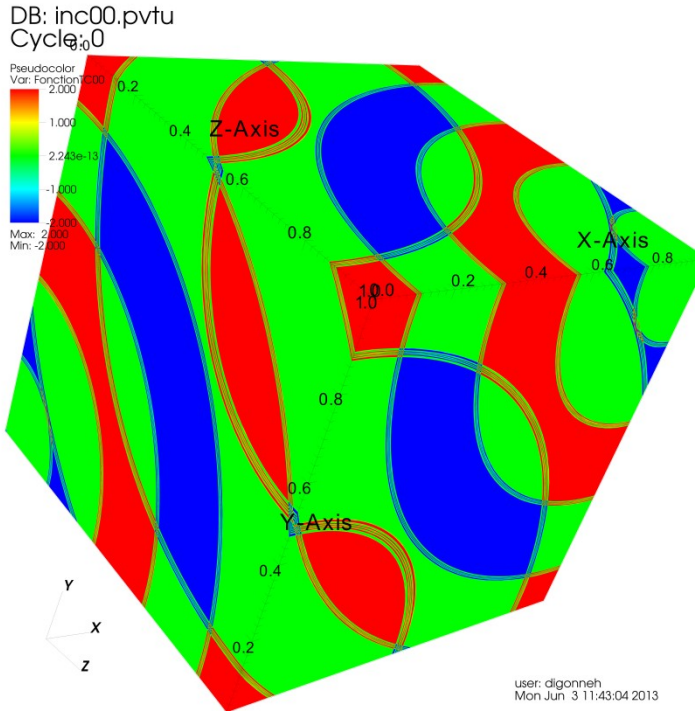
25 million nodes adapted mesh.

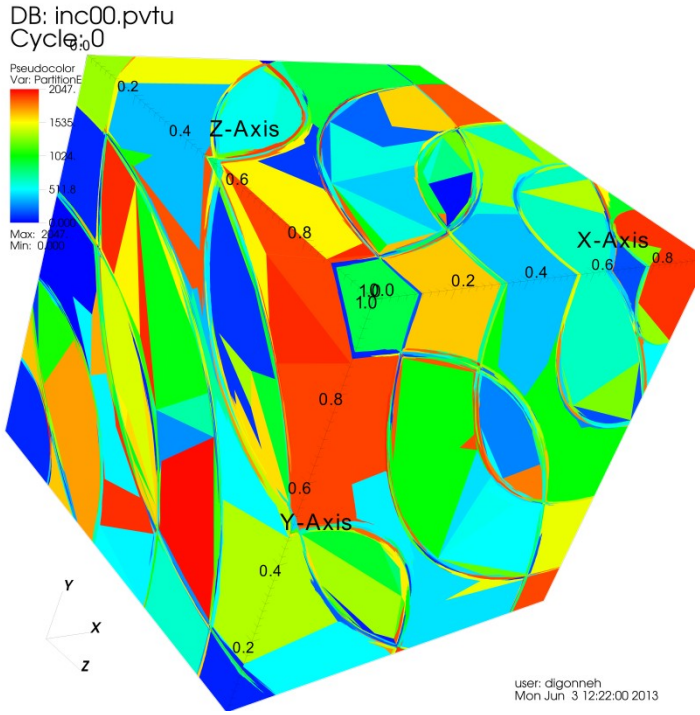
150 steps computed over 512 cores in 1h41m on Jade.

The equivalent uniform isotropic mesh will contain around 1000 billion of nodes.

Illustration : in 3d

Same function with E=16 E=2



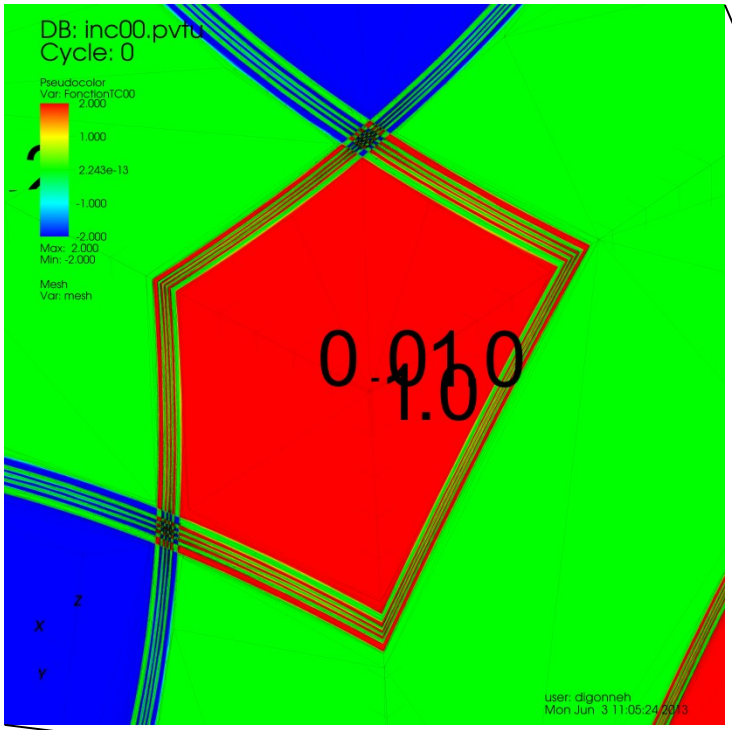


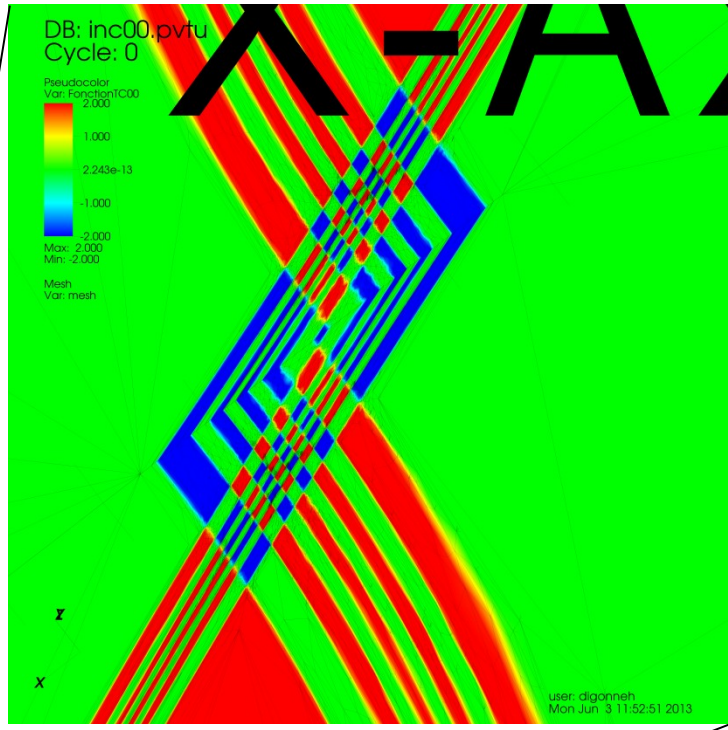
The 60 millions nodes adapted mesh.

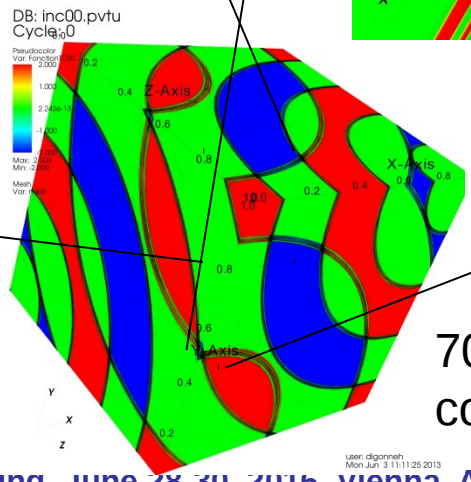
The partition of the mesh over 2048 cores.

Illustration : in 3d

Same function with E=16 E=2







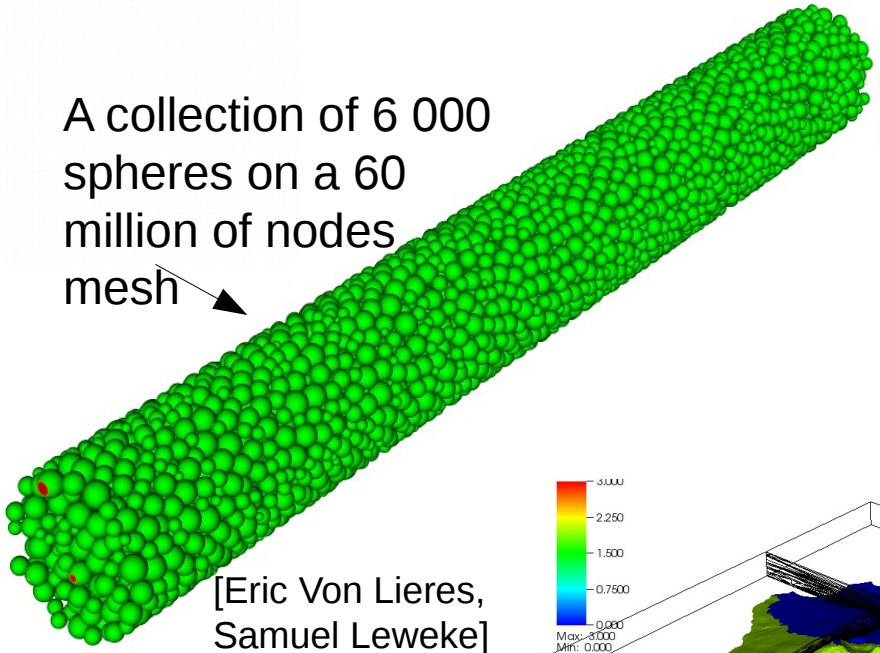
Adapted mesh with 60 millions nodes.
Smallest mesh size less than 1e-4

70 steps done using 2 048 cores of Curie in (10h)

Application : From real to virtual

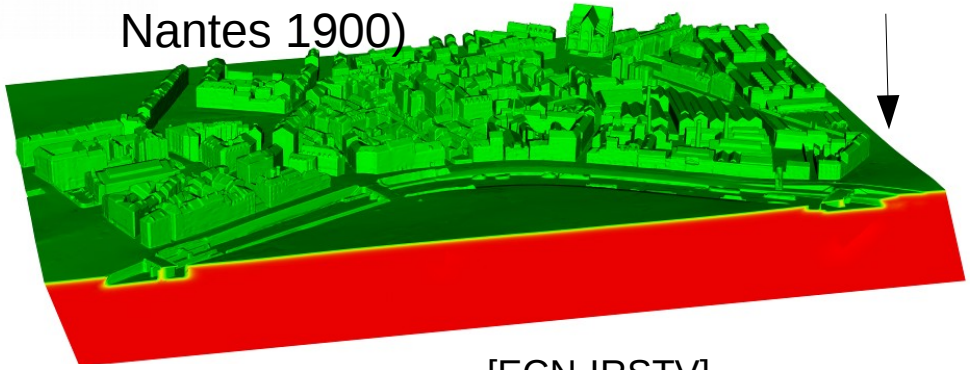
The goal is to incorporate real data into our simulation by combining anisotropic mesh adaptation and immersed domain simulations.

A collection of 6 000 spheres on a 60 million of nodes mesh

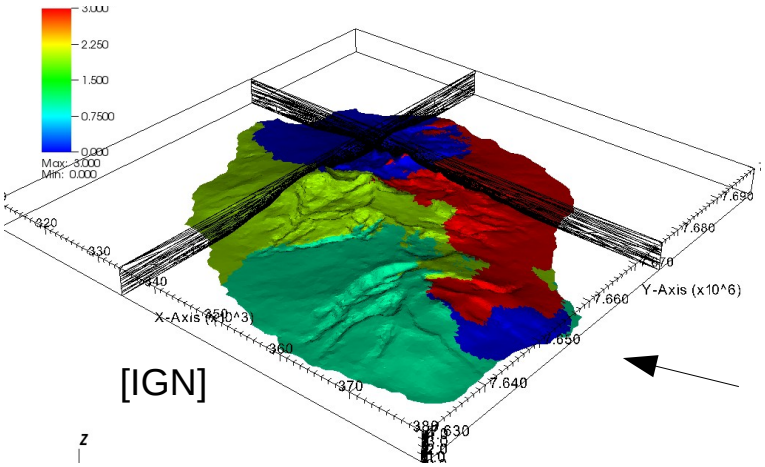


[Eric Von Lieres, Samuel Leweke]

A 5 millions of nodes mesh used to represent a view of Nantes (project Nantes 1900)



[ECN-IRSTV]



[IGN]

Reunion island : anisotropic mesh generated on 4 cores

Multigrid solver : Goals

Issue : nonlinear complexity O($n^{3/2}$ in 2d ; $n^{4/3}$ in 3d) of iterative methods represents an obstacle for solving very large systems.

Stokes resolution using meshes.

# nodes	8 073	32 205	128 354	512 661
# iterations	191	534	1381	3866
Assembly (s)	0.064	0.263	1.14	4.30
Solve (s)	0.90	9.02	102	1221

# nodes	2 070	14 775	112 664	878 443
# iterations	55	137	348	931
Assembly (s)	0.112	0.971	7.737	61.68
Solve (s)	0.0768	1.467	36.52	836

changes in terms of number of iterations, assembling and solving times based on the number of mesh nodes in 2d and 3d cases.

Multigrid solver : Goals

- We could generate and dynamically adapt meshes with several billions of nodes and elements. So we aim in solving EDPs on these meshes.

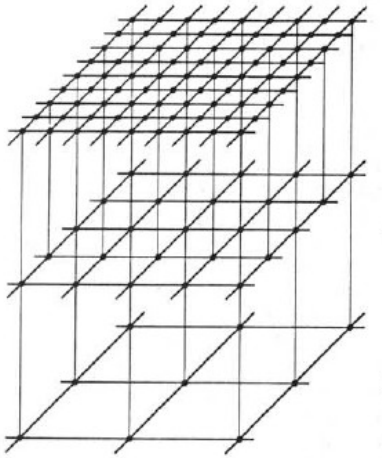
- Complexity of iterative methods, such as conjugate gradient, is a breakdown to deal with such large meshes.

- We want to keep the method versatile and robust
  - not using hierarchical refinement
  - mesh partitioning may change between grid levels
  - only the finest mesh is given

Multigrid solver : PETSc

To perform simulations on such big meshes (containing several millions or billions of nodes) we need to be able to solve very large linear systems.

Traditional preconditioned iterative methods (Krylov) have non linear complexity. To overcome this, we must implement a multigrid method to reduce the complexity and so think about scalability.



Thanks to PETSc, we have a framework to implement a preconditioned multigrid solver, developers “only” need to provide:

Systems to solve at each level:
Discretized physical problem on the level mesh (Geometric MG)
Recursive reduction of the fine problem $A_{n-1} = {}^{t}I_{n-1,n} A_n I_{n-1,n}$ (Algebraic MG)

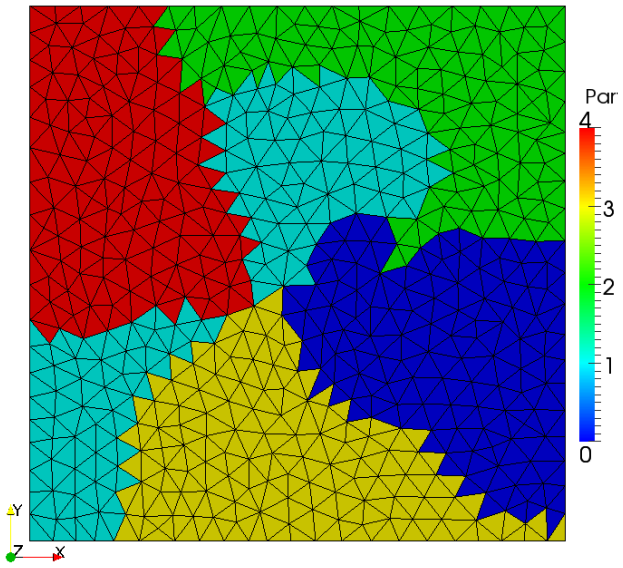
Interpolation $I_{n-1,n}$ or Restriction $R_{n,n-1}$ operators between two mesh levels

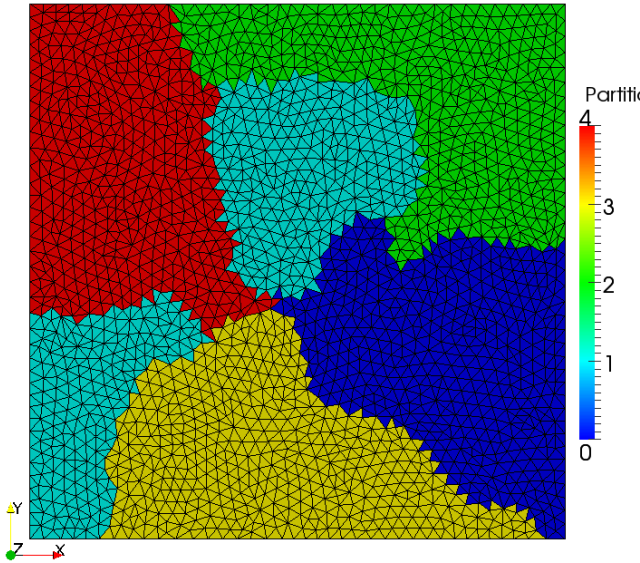
Multigrid Solver : Parallel interpolation/restriction operators

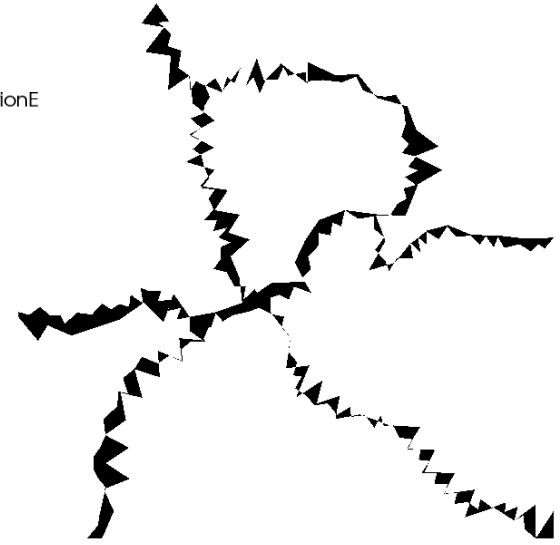
Interpolation operator consist in barycentric coordinates of the fine mesh nodes in the coarse mesh elements.

In parallel, we deal with distributed meshes over processors and so we have to perform both local and external element localization

Thanks to the parallel mesh adaptation strategy, the partition of the two meshes are close to each other and so minimize, by construction, the number of external localizations that represent only a few per cent of the mesh nodes.





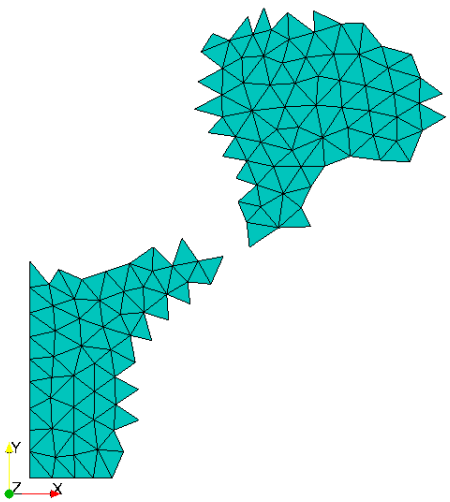


Multigrid Solver : Parallel interpolation/restriction operators with filters

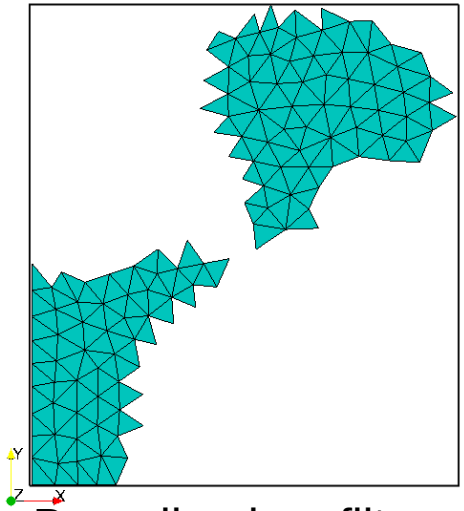
Massively parallel computation makes details become important !!!

For example, for only 5% of external nodes, their coordinates, which need to be sent to the other processors (to acquire external localisation) will represent 50 times local nodes when using 1000 cores, and will lead to memory breakdown .

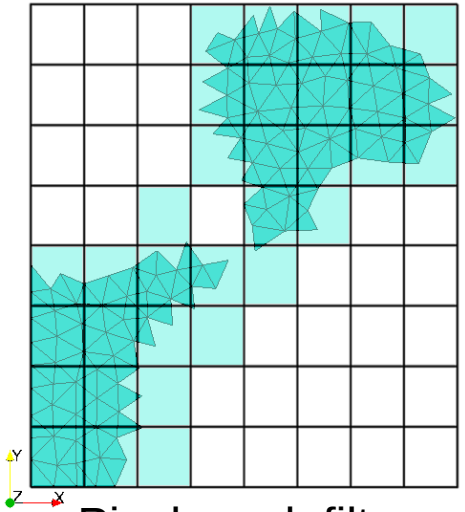
To minimize false detection and so the memory used per core, we introduce processor filters (distributed to all cores) to only send coordinates to potential owner cores and not to all.



The core subdomain



Bounding box filter



Pixel mask filter

Multigrid Solver : Strong speed-up performance

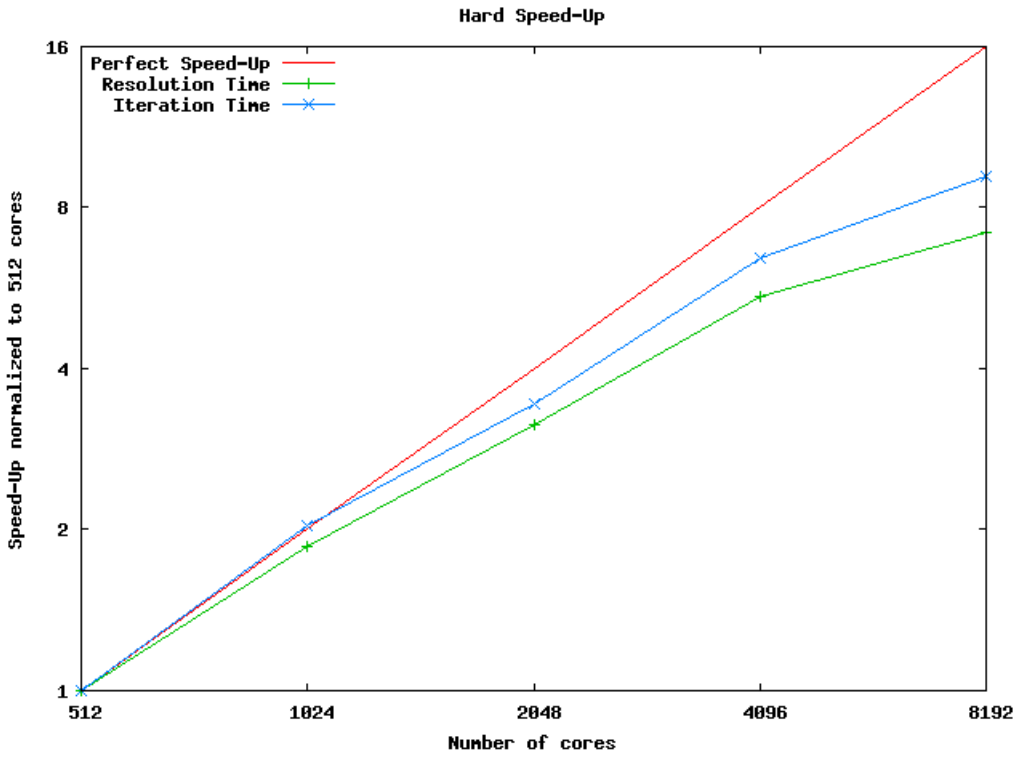
The test case consists in computing the velocity and pressure fields from the incompressible Stokes equation. Resolution using a P1+/P1 finite element formulation on a 2d mesh containing 216 million-nodes (650 million-unknowns).

Runs are executed from 512 to 8192 cores using a 8-level multigrid solver.

Time goes

from 96.7s in 11 iterations on 512 cores

to 13.4s in 14 iterations over 8192 cores.

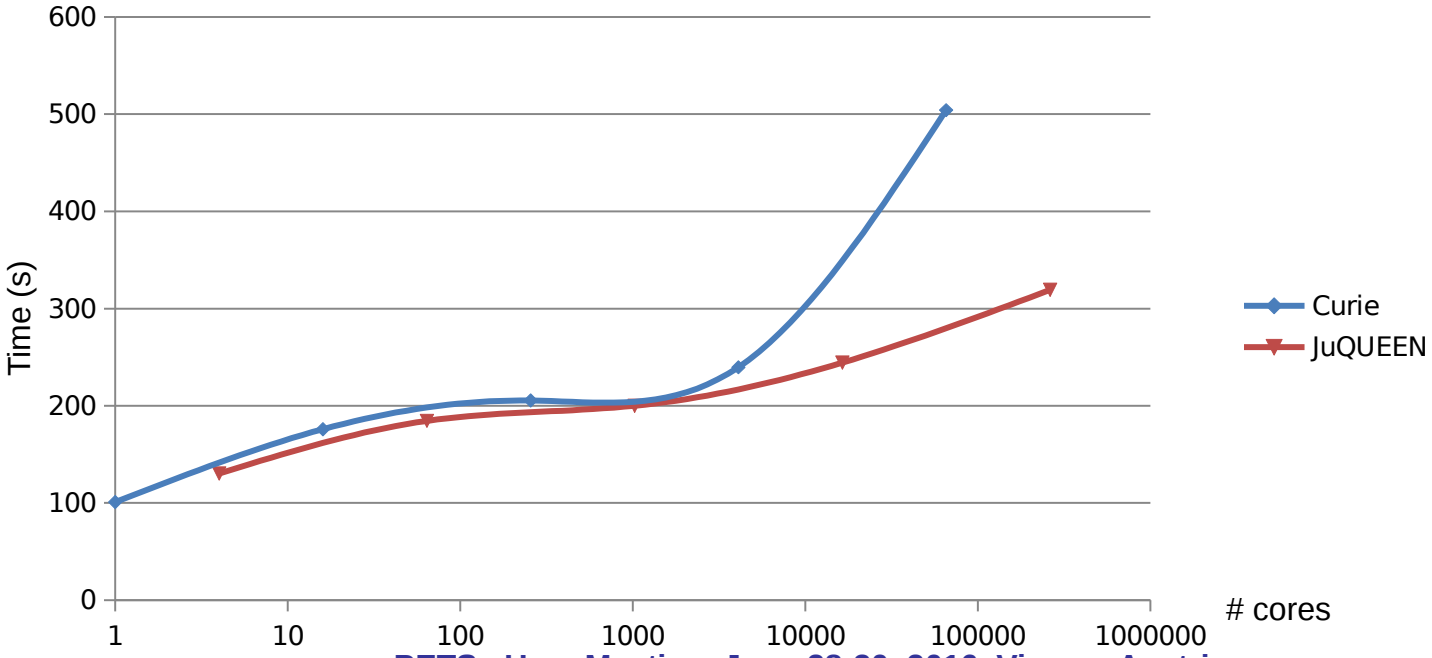


Multigrid Solver : Weak speed-up performance

The test case consists in computing the velocity and pressure fields from the incompressible Stokes equation resolution using a mixed P1+/P1 finite element

Runs from 1 to 262 144 cores for a relative error of 1e-9
Biggest run with 100 billion of unknowns (using around 200TB of RAM)

Really good performance up-to 10 000 cores, worsening beyond (specially on Curie).



Multigrid Solver : Weak speed-up details on Curie

Runs done over 16 to 65,536 cores for a relative convergence at 1e-9

Biggest run with 100 billion of unknowns

Assembling ans solving times almost constant except 65,536 cores (with a significant degradation)

# cores	16	256	4 096	65 536
# levels	4	6	7	8
# smoothing	10	10	10	10
# nodes	8 157 137	130 503 221	2 087 280 602	33 394 443 636
# elements	16 314 272	261 006 440	4 174 561 202	66 788 887 270
# dof	24 471 411	391 509 663	6 261 841 806	100 183 330 908
# mg iterations	13	17	16	22
Assembly (s)	10.09	10.41	11.84	61.43
Solve (s)	218.8	274,3	263.9	431.6

Multigrid Solver : Weak speed-up details on JuQUEEN

Runs done over 64 to 262,144 cores for a relative convergence at 1e-9

Biggest run with 100 billion of unknowns

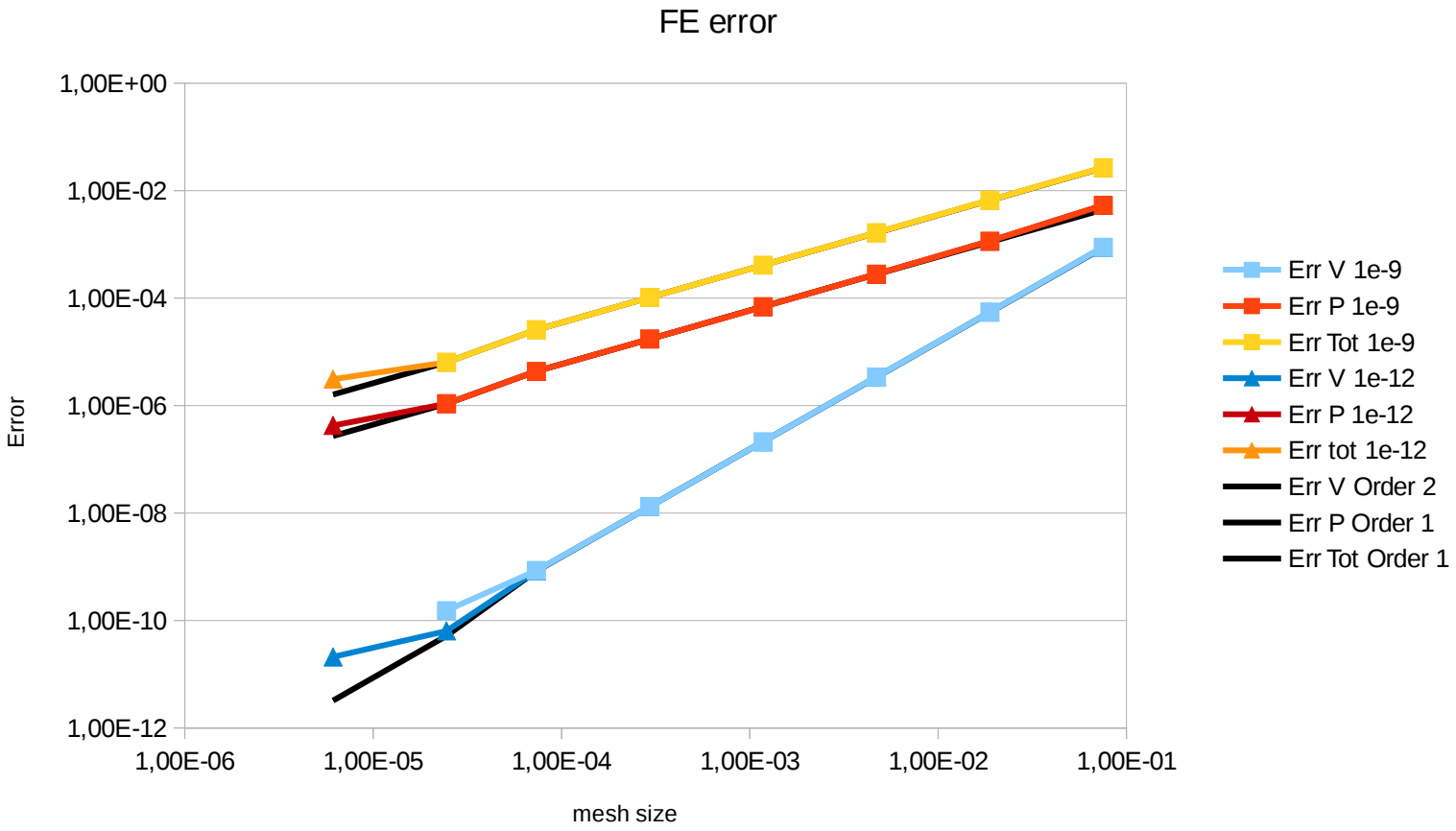
Solving times almost constant, Assembling time also except on 262,144 cores (with a significant degradation)

# cores	64	1 024	16 384	262 144
# levels	4	5	6	7
# smoothing	5	5	5	5
# nodes	8 216 917	131 689 655	2 108 867 176	33 777 732 848
# elements	16 433 832	263 379 308	4 217 734 350	67 555 465 694
# dof	24 650 751	395 068 965	6 326 601 528	101 333 198 544
# mg iterations	15	18	18	19
Assembly (s)	28.27	29,1	34.2	88.9
Solve (s)	164.5	197,6	207.3	226.0

Multigrid solver : FE error analysis

Linear convergence for the pressure and quadratic for the velocity

We reach the limit of the double precision float (1e-14) on the biggest run !



Multiphase flow

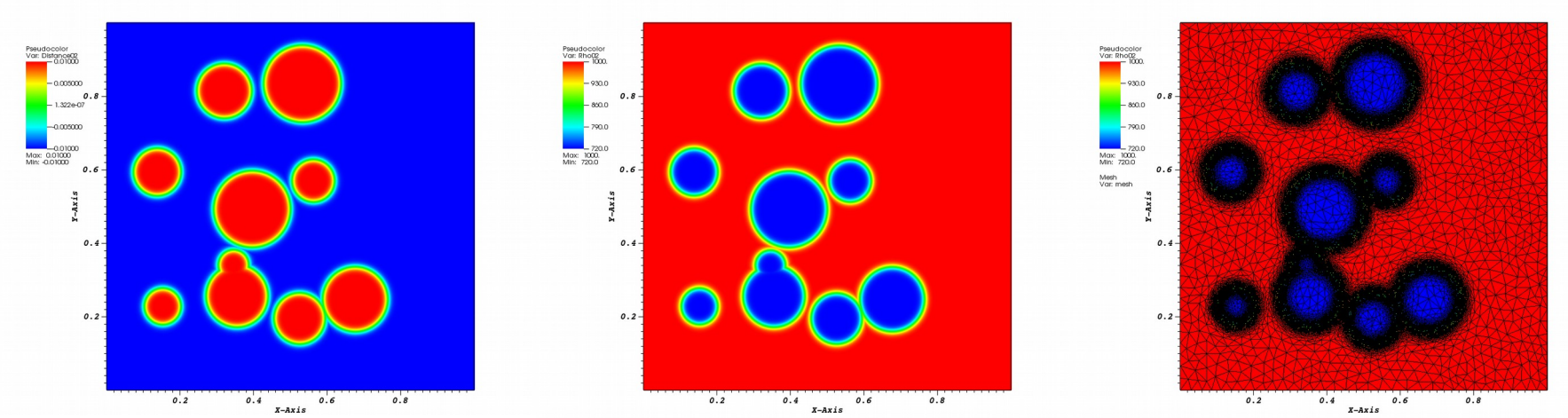
[T. Coupez, “Metric construction by length distribution tensor and edge based error for anisotropic adaptive meshing”, Journal of Computational Physics, 2011]

Phases are separated using a modified LevelSet function

$$u_\epsilon(x) = \epsilon * \tanh\left(\alpha(x)/\epsilon\right) \quad \text{with} \quad \alpha(x) = \begin{cases} -d(x) \text{ outside} \\ 0 \text{ boundary} \\ d(x) \text{ inside} \end{cases}$$

This LevelSet function is used both :

- to mix the physical properties $P_{AB} = \left(1 + \frac{u_\epsilon}{\epsilon}\right) P_A + \left(1 - \frac{u_\epsilon}{\epsilon}\right) P_B$
- to do mesh adaptation in order to well capture phases using an anisotropic error estimator [T. Coupez, 2011]



Unsteady multiphase flow

[L. Ville et al. , “Convected level set method for the numerical simulation of fluid buckling”, International Journal for numerical methods in fluids, 2011]

We iteratively :

- Solve incompressible Stokes equation to compute velocity and pressure fields using a mixed P1+/P1 finite element.

$$\begin{cases} 2\eta\epsilon(v) - \nabla p = \rho g \\ \nabla . v = 0 \end{cases}$$

- Solve the advection equation to update the position of the phases.

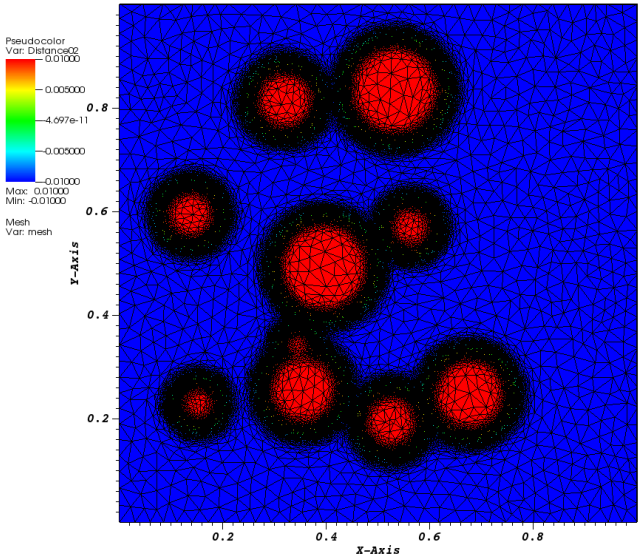
  This formulation is stabilized and uses redistancing to keep a gradient close to one, with a convective reinitialization method [L. Ville, 2011]

$$\frac{\partial u_\epsilon}{\partial t} + (v + \lambda v_r) . \nabla u_\epsilon = \lambda s(u_\epsilon) \left(1 - \left(\frac{u_\epsilon}{2} \right)^2 \right)$$

- Remesh to keep a good enough mesh (not each time step)

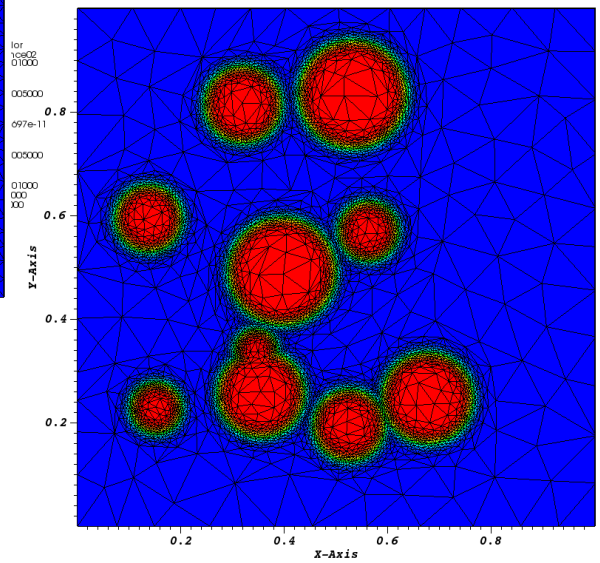
Multigrid solver for unsteady computation

Goals : a) Keep the finest level as the input.



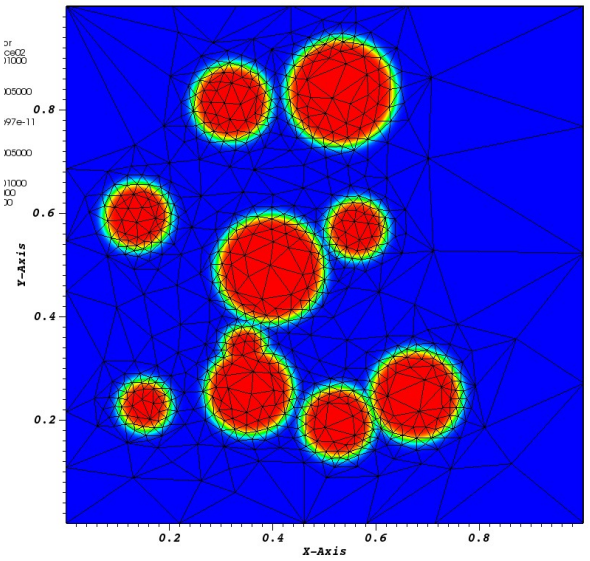
fine mesh :
35 898 nodes

Illustration with 3 levels mesh coarsening



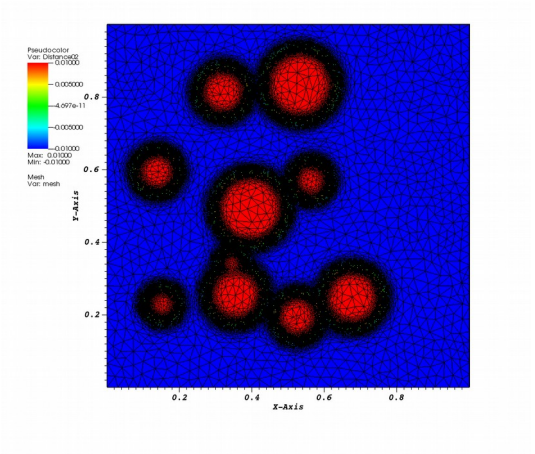
inter mesh :
5 850 nodes

coarse mesh :
815 nodes

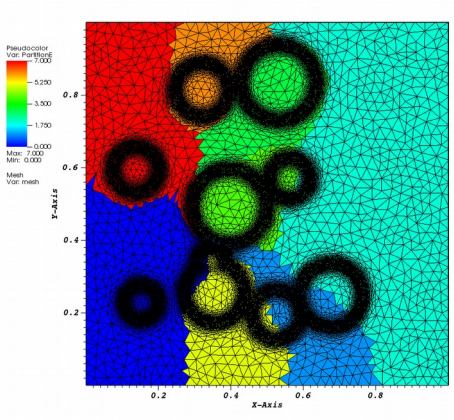


Parallel multigrid solver for unsteady computation

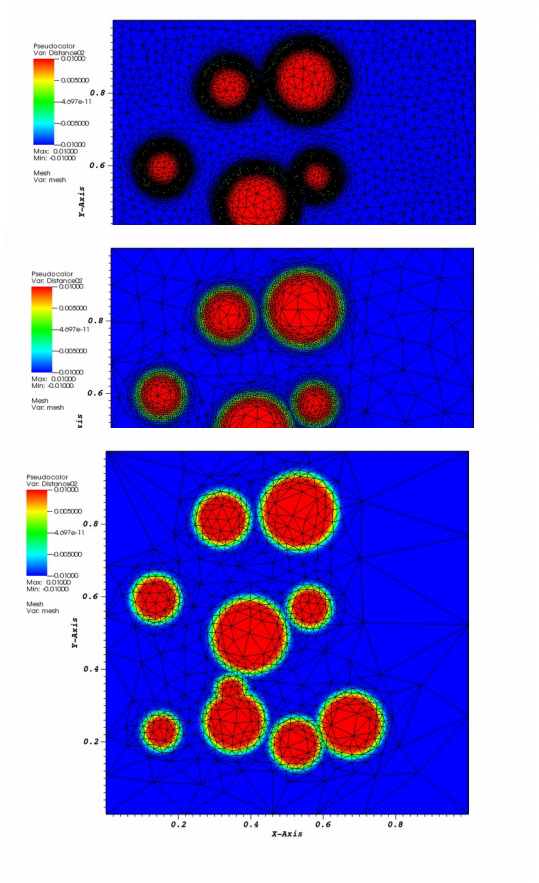
Goals : b) Combine optimizations



Mesh adaptation
H/A



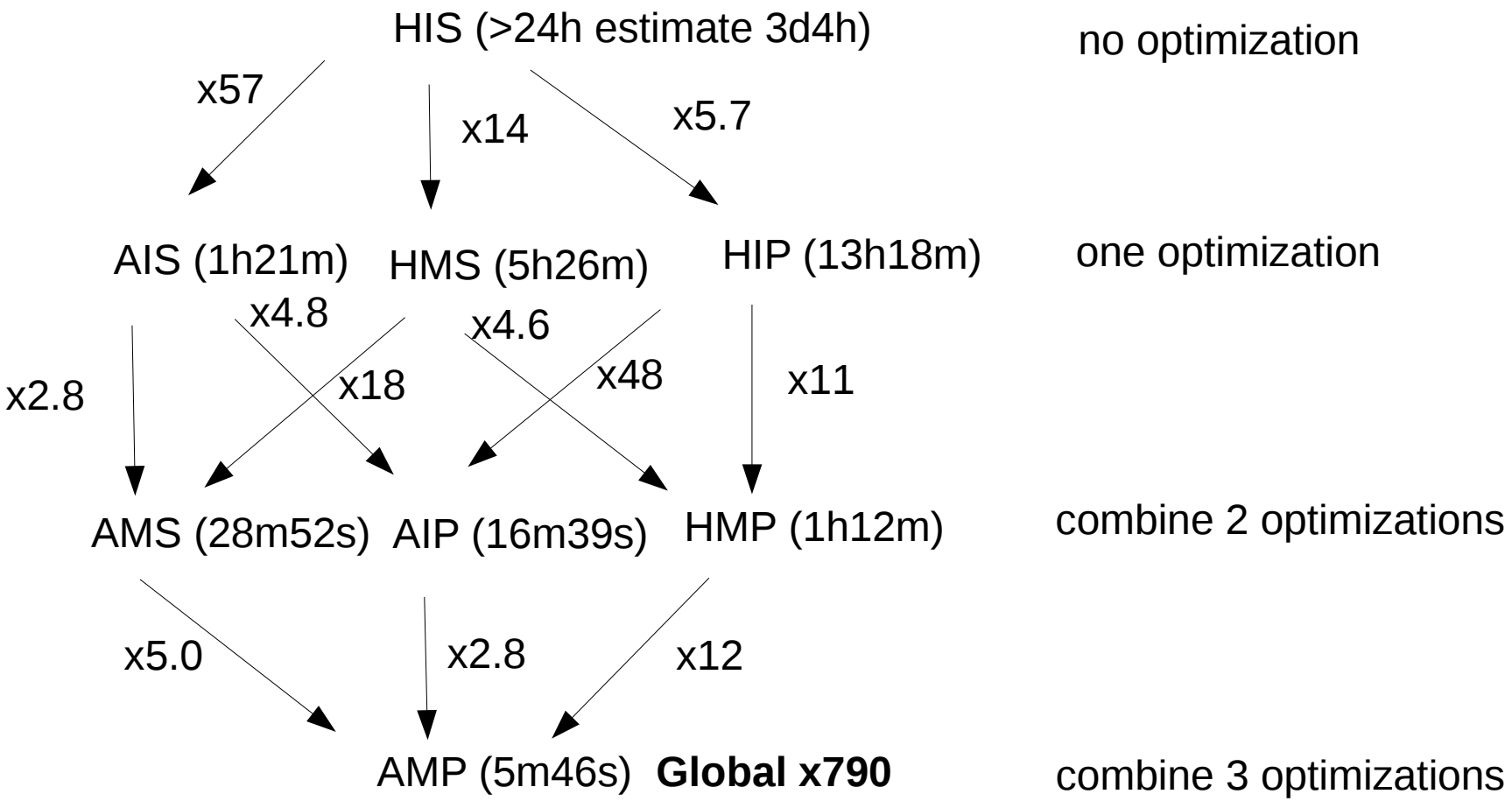
Parallel computing
S/P



Multigrid solver
I/M

Graph of gains combining the 3 optimizations

Global time for 40 increments for results equivalent to a 1.25 millions nodes homogeneous mesh (multigrid 3 levels, parallel using 8 cores).



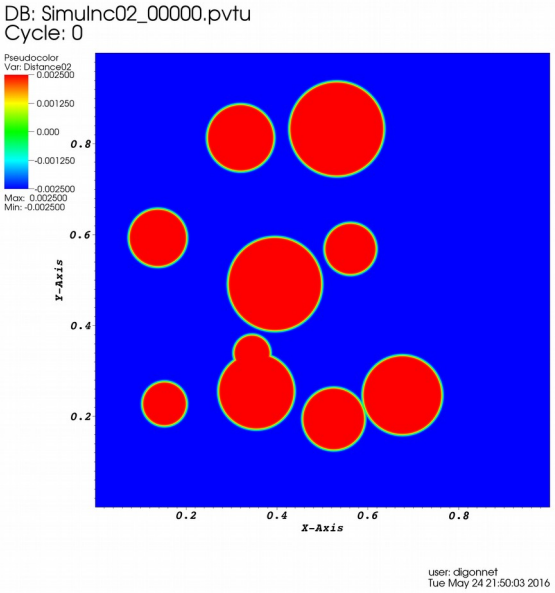
2d example : unit square with 10 bubbles

10 bubbles

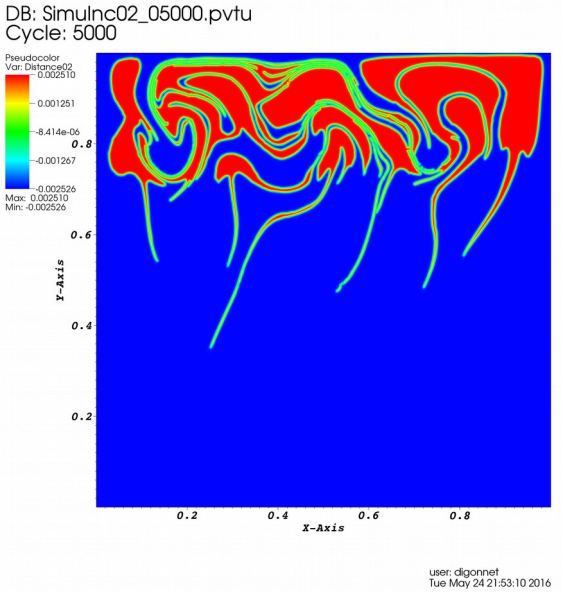
3 levels multigrid :
- fine : 135 000 nodes
- inter : 15 000 nodes
- coarse : 2 000 nodes

Details per increment :
Time step : 1e-4 s
Stokes : 3.46 s
LevelSet : 0.59 s
Remeshing : 1.2 s (*)

7 000 increments (Stokes + LevelSet), 700 remeshing steps runs on 16 cores on Curie supercomputer 9h37m



movie



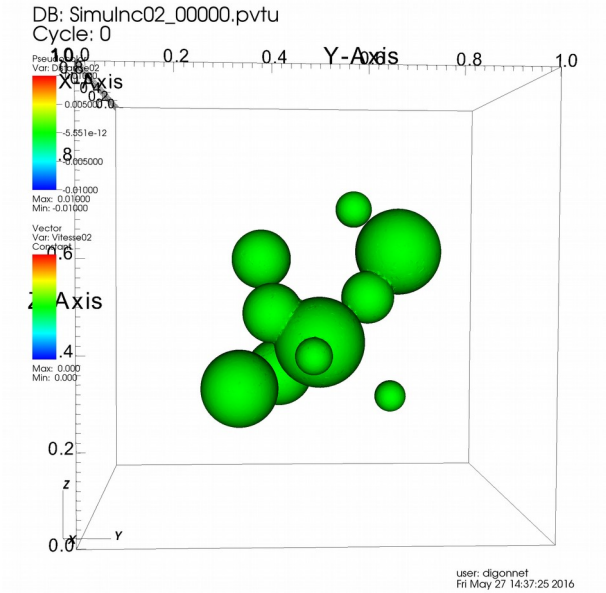
3d example : unit cube with 10 bubbles

10 bubbles

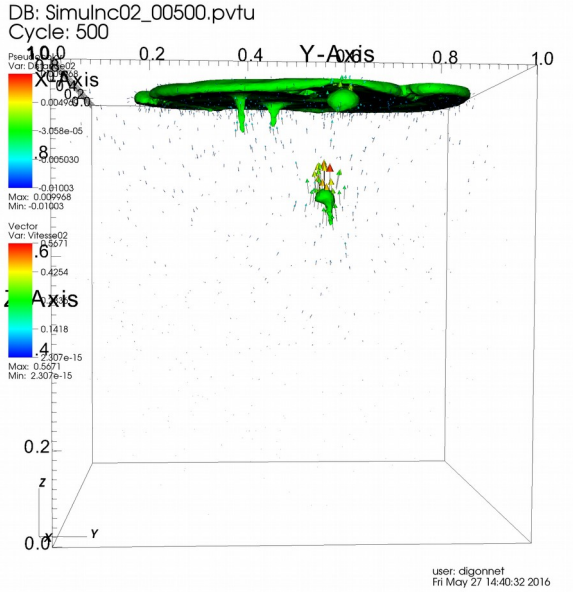
3 levels multigrid :
- fine : 100 000 nodes
- inter : 5 000 nodes
- coarse : 300 nodes

Details per increment :
- Time step : 1e-3 s
- Stokes : 4.22 s
- LevelSet : 0.57 s
- Remeshing : 9.5 s (*)

500 increments (Stokes + LevelSet), 100 remeshing steps runs on 16 cores on Curie supercomputer 1h51m



movie



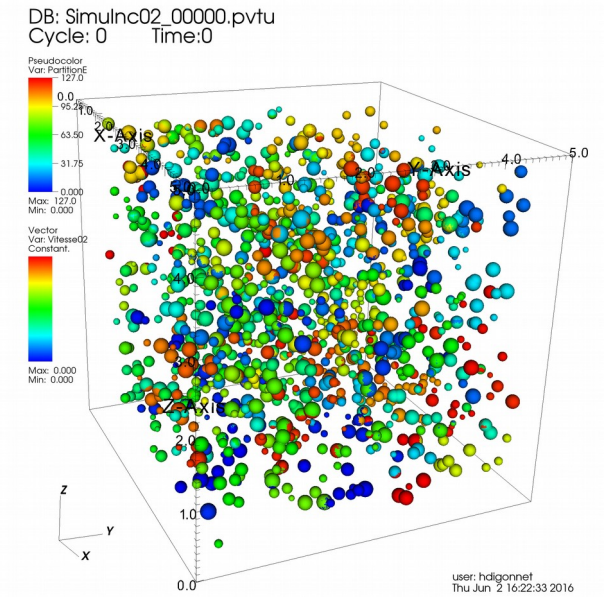
3d example : 5x5x5 cube with 1 250 bubbles

1 250 bubbles

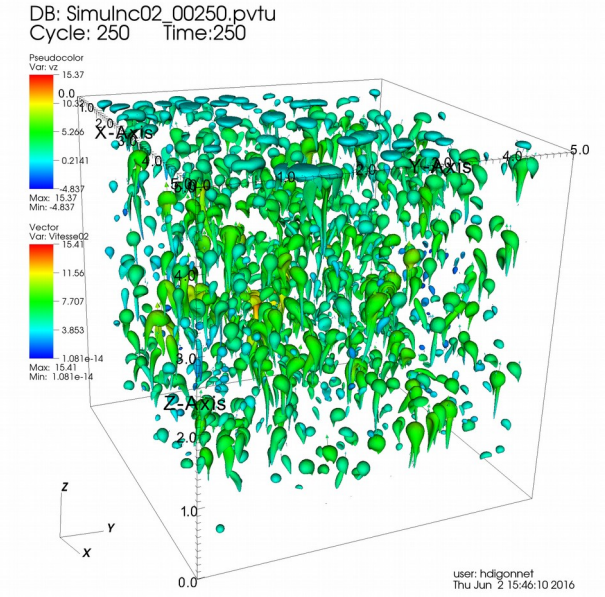
3 levels multigrid :
- fine : 10 000 000 nodes
- inter : 1 450 000 nodes
- coarse : 170 000 nodes

Details per increment :
Time step : 5e-4 s
Stokes : 84,2 s
LevelSet : 3.25 s
Remeshing : 402 s (*)

330 increments (Stokes + LevelSet), 33 remeshing steps runs on 128 cores on Ligar supercomputer in 15h41m



movie



3d example : 10x10x10 cube with 10 000 bubbles

10 000 bubbles

4 levels multigrid :

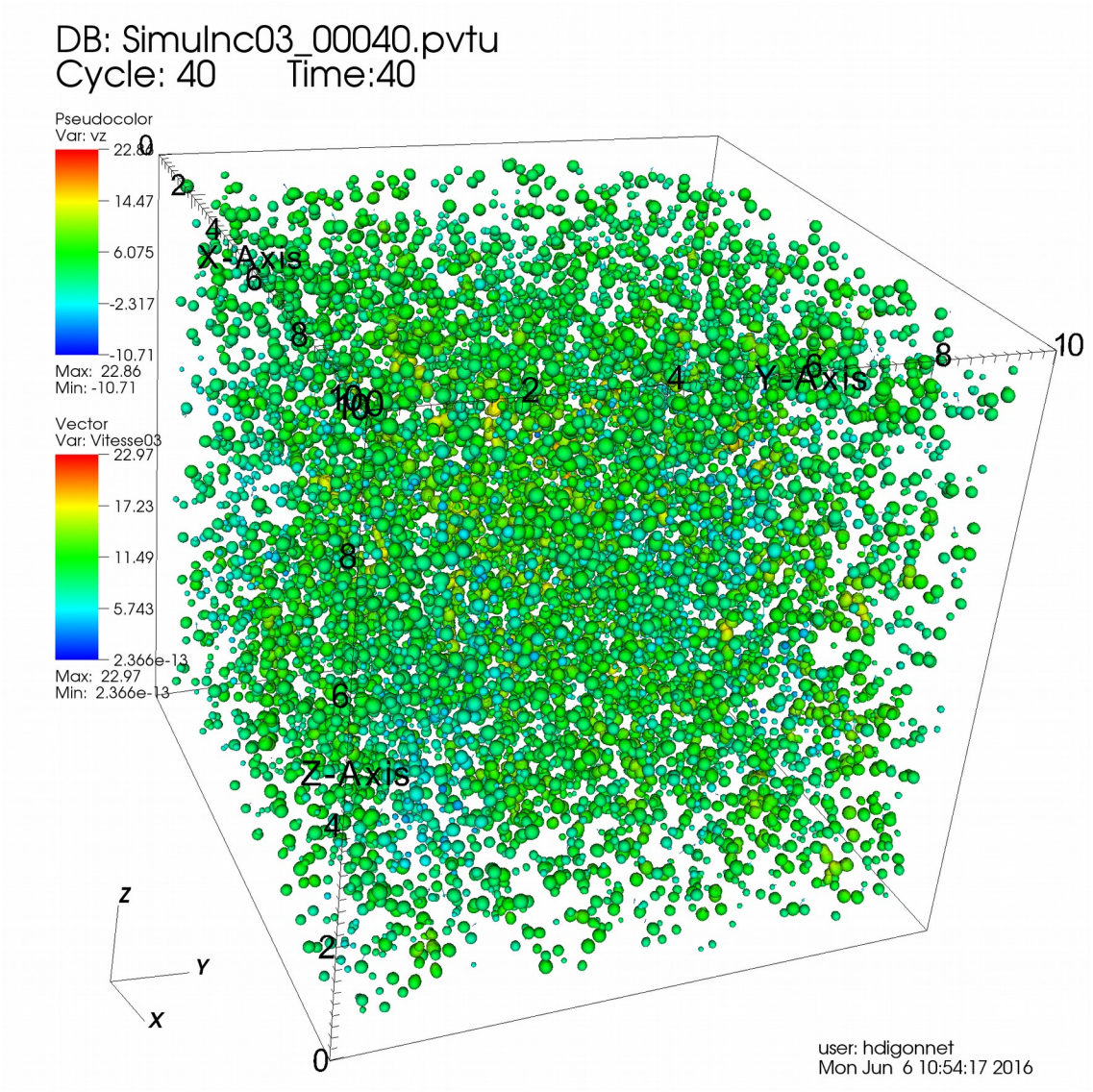
- fine : 100 000 000 nodes
- inter : 12 700 000 nodes
- inter 2 : 1 713 000 nodes
- coarse : 223 000 nodes

Details per increment :

Time step : 2.5e-4 s
Stokes : 118 s
LevelSet : 13.3 s
Remeshing : 805 s (*)

40 increments (Stokes + LevelSet), 2 remeshing steps runs on 1 024 cores on Liger supercomputer in 4h25m

3d example : 10x10x10 cube with 10 000 bulles



Conclusions and future works

We have been able to execute mesh adaptation and a multigrid solver at the full Tier0 supercomputer scale. The biggest linear system solved contained **100 billions unknowns** using **65 536** (Curie) and **262 144** (JuQUEEN) cores and **200 TB** of RAM.

All this work leads to combine accelerations ($x10^{11}$) given by each improvement:

- anisotropic mesh adaptation : reduce number of dof (x1 000)
- massively parallel computation : impressive computational power (x100 000)
- multigrid solver : reduce the number of floating operations needed (x1 000)

Visit and Paraview enabled us to visualize such results

Application to unsteady simulations seems very promising and will allow very large scale simulations.

To be done :
- continue analysis of the multigrid solver : time step stability, coarsening factor, reduce number of cores used on coarse levels.
- allow higher order elements P2/P3
- look to 128 bits float precision

Acknowledgment

I want to thanks

GENCI (Grand Equipement National de Calcul Intensif)

For access to French Tier1 supercomputers : Jade, Curie and Turing

PRACE (Partnership for Advanced Computing in Europe)

For access to Tier0 supercomputer : Curie and JuQUEEN.

Thanks for your attention