Le multigrille pour exploiter pleinement les calculateurs parallèles : expérimentation sur les supercalculateurs Tier0.

H. Digonnet, L. Silva et T. Coupez

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

(Anciennement groupe CIM au CEMEF laboratoire de MINES-ParisTech)

-École thématique sur le multigrille, 21 novembre 2014, Fréjus-

Adéquation méthodes multigrille et calcul parallèle :

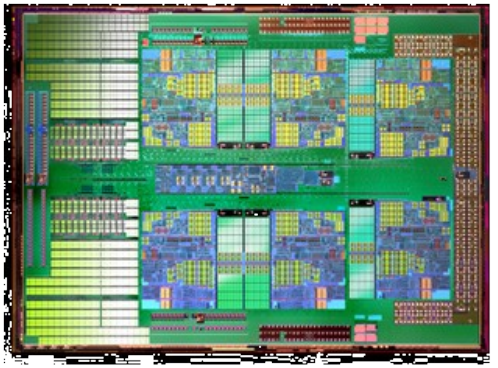
- Complexité algorithmique asymptotique quasi linéaire voir linéaire pour les méthodes multigrille (O(n log n) ou O(n))

  => favorable pour des grands systèmes linéaires

- Calcul parallèle voir massivement parallèle

  => augmentation importante de la taille des problèmes traités.

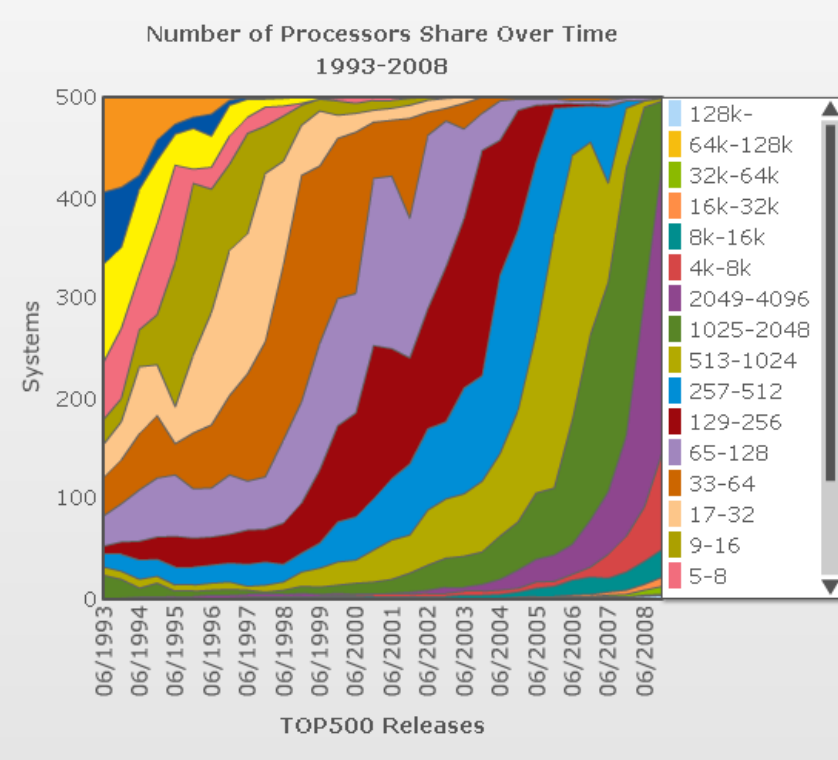
Pourquoi du calcul parallèle ?



- Plus de processeurs séquentielles
  - la puissance des CPU augmente plus par l'addition de nouveaux cœurs de calcul
  - Aujourd'hui 2,4 et même 18 cœurs par CPU.

- Cluster à plusieurs centaines de milliers de cœurs
  - En Novembre 2014 dans la liste top500: le nombre de cœurs allait de 2992 à plus de 3 000 000.

- Pour le calcul scientifique :
  - Nous devons utiliser le calcul parallèle
  - Nous devons penser au massivement parallèle (plus de N cœurs)



Calculateurs Tier0 (top continental supercomputer)

Curie :

- 80 640 cœurs Intel Xeon 2.7 GHz pour 322To de mémoire RAM
- construit en 2012
- puissance de **1,359 PFlops** soit $1{,}359*10^{15}$ Flops!

JuQUEEN :

- 458 752 cœurs PowerPC 1.6 Ghz pour 448To de mémoire RAM
- construit en 2012
- puissance de **5,0 PFlops** soit $5{,}0*10^{15}$ Flops!

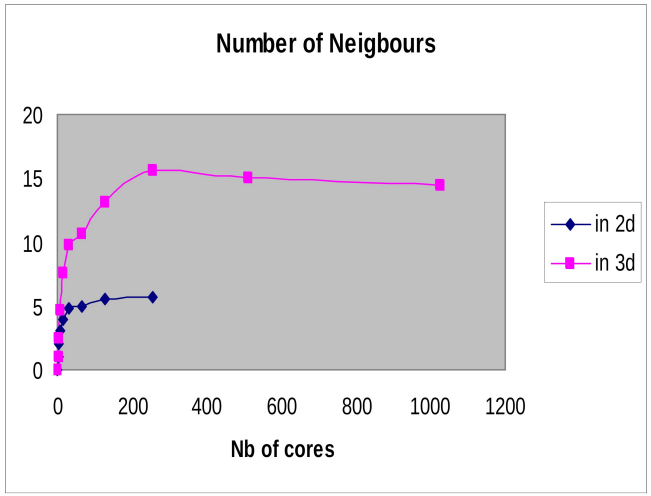
A titre de comparaison : un cœur standard **40 GFlops** et le supercalculateur le plus puissant : **33,8 PFlops.**

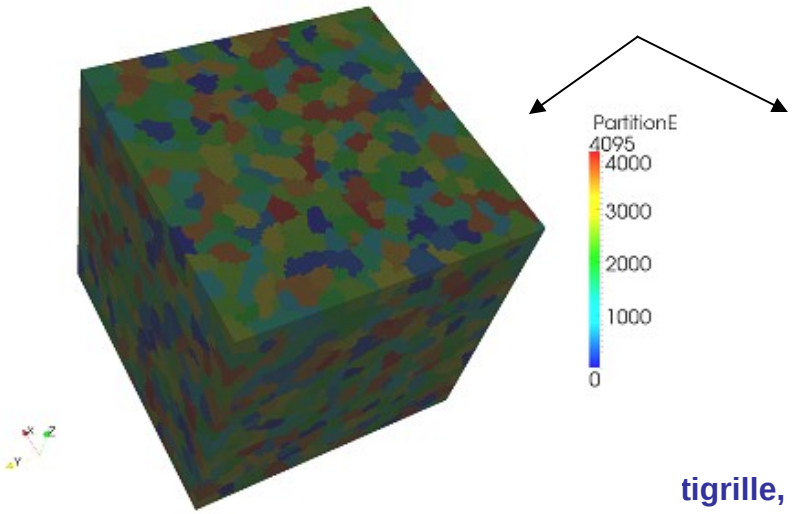
Qu'es que le massivement parallèle ?

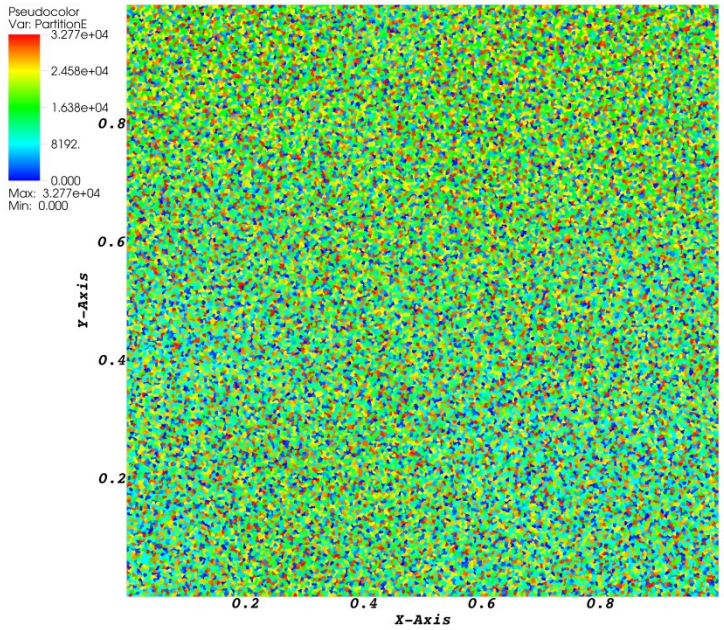
- Hardware : avec un grand nombre de cœurs
  Curie : 80 640 cœurs 2.7 GHz avec 4GB/cœur
  JuQUEEN : 458 752 cœurs 1.6 GHz avec 1GB/cœur

- Software 1 : le nombre de voisins d'un domaine ne varie plus.

- Software 2 : le nombre de cœurs est du même ordre de grandeur que le nombre de donnée locale à a un cœur







Plan :

- Brève introduction au calcul parallèle
- Adaptation de maillage en parallèle
- Implémentation parallèle d'une méthode multigrille
- Performances obtenus sur des supercalculateurs Tier0
- Exemples de simulations réalisées.

Brève introduction au calcul parallèle

Résolution d'un système linéaire Ax= b ;

- Méthodes directes

  Très difficilement parallélisable :

  ordre des instructions important : triangularisation par Gauss, descente puis remontée)

- Méthodes Itératives

  Parallélisation beaucoup plus facile :

  se réduit à des calculs de produits scalaires, de produits matrice vecteur et quelques opérations arithmétiques sur des vecteurs

Méthode du gradient conjugué

Initialisation:

X_0 , $r_0 = b - AX_0$, $p_0 = r_0$

Itérations

Pour i = 0,1, 2.....jusqu'à convergence

$$\alpha_i = \frac{(r_i, r_i)}{(p_i, Ap_i)} ,$$

$X_{i+1} = X_i + \alpha_i p_i$,

$r_{i+1} = r_i - \alpha_i A p_i$,

si $(r_{i+1}, r_{i+1}) < \varepsilon$ **stopper les itérations**

$$\beta_i = \frac{(r_{i+1}, r_{i+1})}{(r_i, r_i)} ,$$

$p_{i+1} = r_{i+1} + \beta_i p_i$.

Produit scalaire : séquentiel

$$\sum_{i=0}^{n} x_i y_i$$

Entrée : 2 vecteurs de taille n

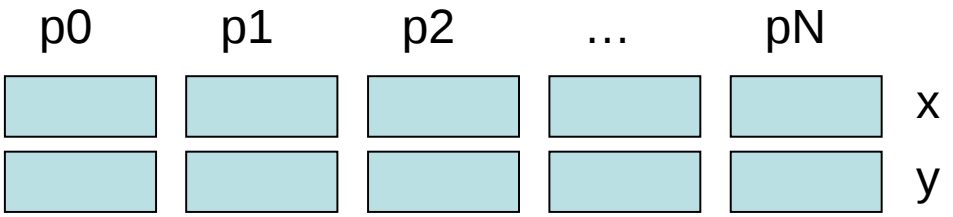
Sortie : un scalaire

Algo :

```
double ps = 0 ;
for(i=0;i<n;i++)
        ps += x[i]*y[i] ;
```

Produit scalaire : parallèle (répartition des taches)

$$\sum_{p=0}^{np} \sum_{i=0}^{n_p} x_i^p \, y_i^p$$



Entrée : 2 vecteurs distribués de taille n

Sortie : un scalaire

Quelle taille pour n_p ?

Idéal : $n_p = n/p$ mais peux ne pas être un entier

Solution 1 : $\begin{cases} n/p + n\%p & \text{Si p = 0} \\ n/p & \text{Sinon} \end{cases}$

Produit scalaire : parallèle (répartition des taches)

Cette solution est convenable si $n/p >> p$ (**!= massivement parallèle**)

Sinon ceci crée un gros déséquilibre de charge !!!

ex : n = 1 000 999 et p = 1000

n/p =1000 et n/p + n%p = 1999

le processeur 0 à le double de travail.

Solution 2 : $\begin{cases} n/p+1 \\ n/p \end{cases}$ si p < n%p

sinon

ex : n = 1 000 999 et p = 1000

999 processeurs avec 1001 valeurs

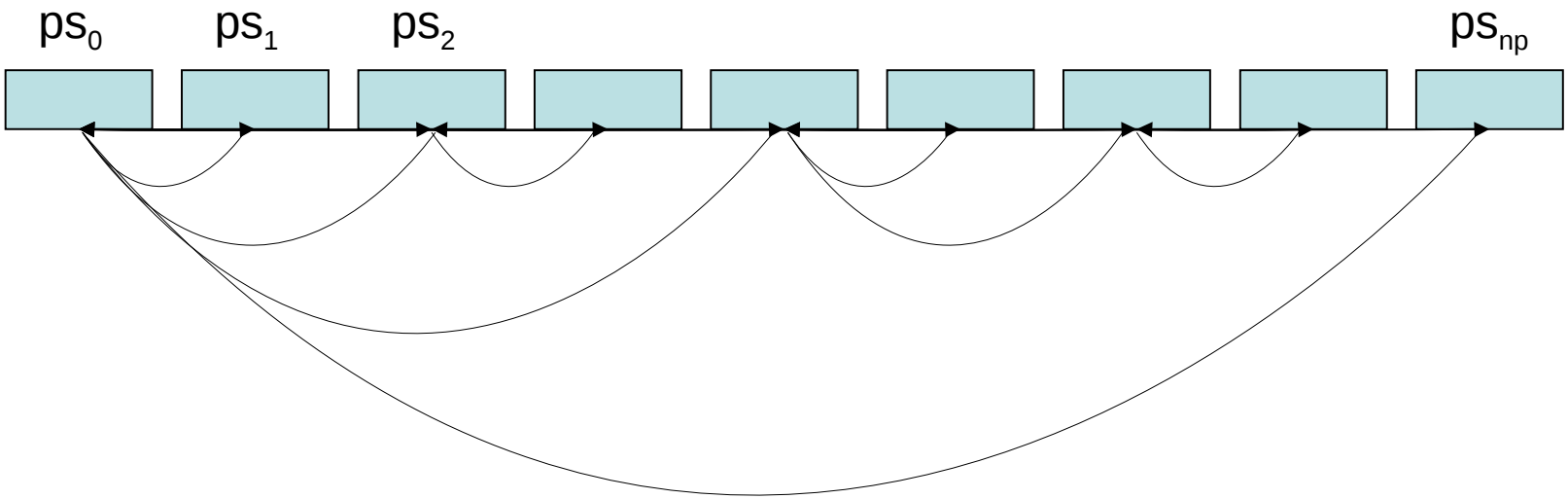
et le dernier avec 1000 valeurs.

Produit scalaire : parallèle (les communications)

$$\sum_{p=0}^{np} \sum_{i=0}^{n_p} x_i^p \, y_i^p$$

Chaque processeur a un scalaire qui représente le produit scalaire local, il faut maintenant les sommer !!!

Solution 3 :

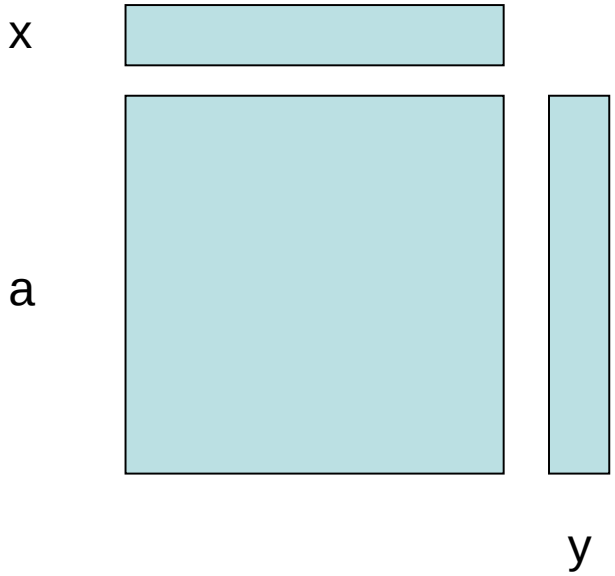


Chaque processeur impaire envoie sa valeur au processeur pair inférieur qui effectue la somme et on itère le processus.

Globalement : 2 (np-1) messages et (np-1) additions et profondeur en $\log_2(np)$

Produit matrice vecteur : séquentiel

$$y_i = \sum_{j=0}^{n} a_{ij} x_j$$



Entrée : une matrice n x n et un vecteur de taille n

Sortie : un vecteur de taille n

Algo :

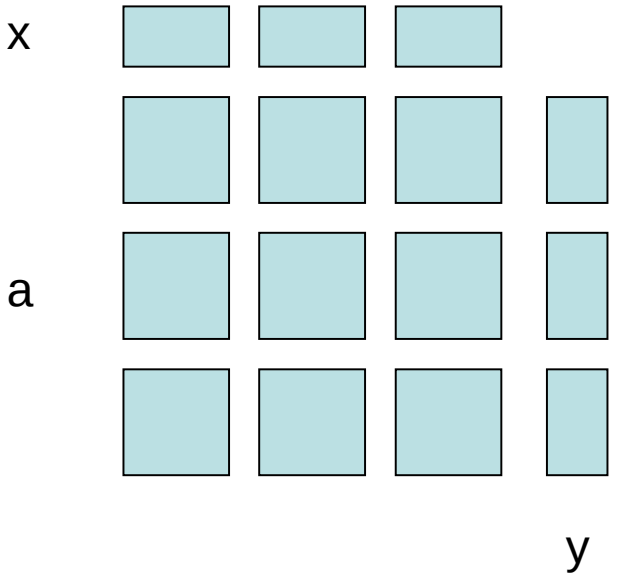
```
for(i=0;i<n;i++)
    y[i] = 0 ;
for(i=0;i<n;i++)
   for(j=0;j<n;j++)
      y[i] += a[i][j] * x[j]
```

Produit matrice vecteur : parallèle (matrice creuse)

$$y_i^p = \sum_{p=0}^{np} \sum_{j=0}^{n/np} a_{ij}^p x_j$$

$$y_i^p = \sum_{j=0}^{n/np} a_{ij}^p x_j^p + \sum_{q \neq p} \sum_{j=0}^{n/np} a_{ij}^q x_j^q$$

$Y^{p,p}$ $Y^{p,q}$



Remarques :

- si a est diagonale ou bloc diagonale $Y^{p,q}$ est nulle

Conclusions :

Le produit matrice vecteur parallèle sera d'autant plus efficace que les blocs extra diagonaux seront vides (pas de communication) ou creux (minimisation des communications sur le vecteur x).

Partitionnement : qualité d'une partition

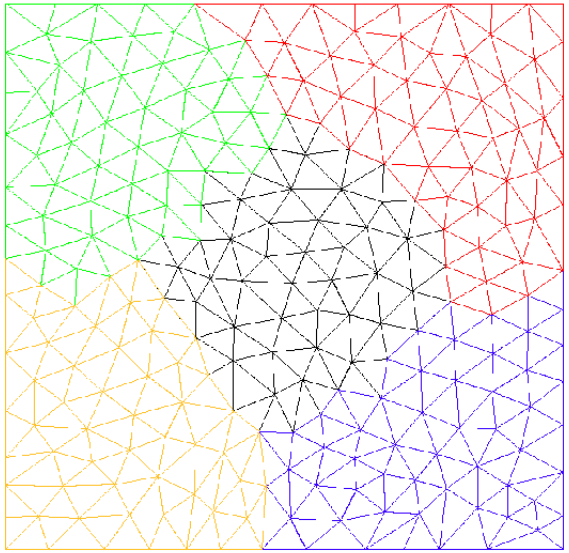
Qualité d'une partition : une optimisation multicritères

- elle doit être équilibrée : le volume de calcul effectué par chaque processeur doit être sensiblement le même.

- elle doit minimiser les interfaces entre les sous-domaines de façon à minimiser le temps passé dans les communications.

Partitionnement : exemple 2d

La partition



Les communications

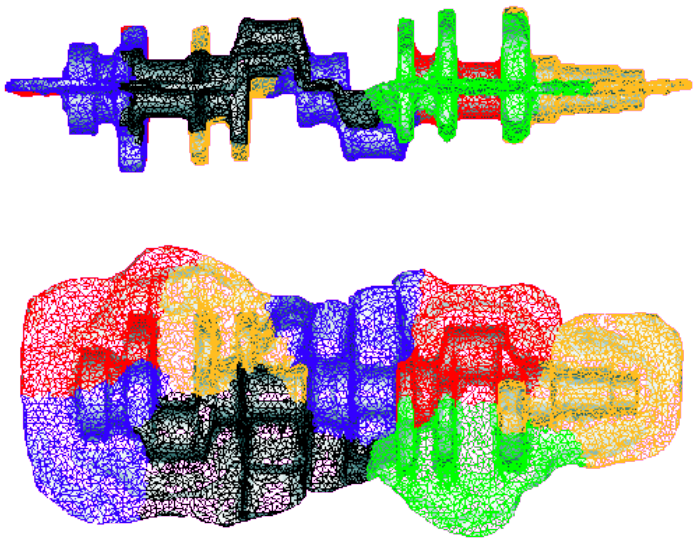
$p \setminus q$	1	2	3	4	5
1			3	5	6
2			4	4	8
3	3	4			8
4	5	4			8
5	6	8	8	8	

La charge des processeurs

domaine	1	2	3	4	5
charge	57	58	58	57	58

Partitionnement : exemple 3d

La partition



Les communications

$p \setminus q$	1	2	3	4	5	6	7	8
1				176			216	
2						82		387
3				50	212	131	159	
4	176		50		28		358	
5			212	28		354		294
6		82	131		354			10
7	216		159	358				
8		387			294	10		

La charge des processeurs

domaine	1	2	3	4	5	6	7	8
charge	5049	5223	6019	6055	6055	5677	5806	5217

Définitions : Accélération (speed-up) et efficacité parallèle

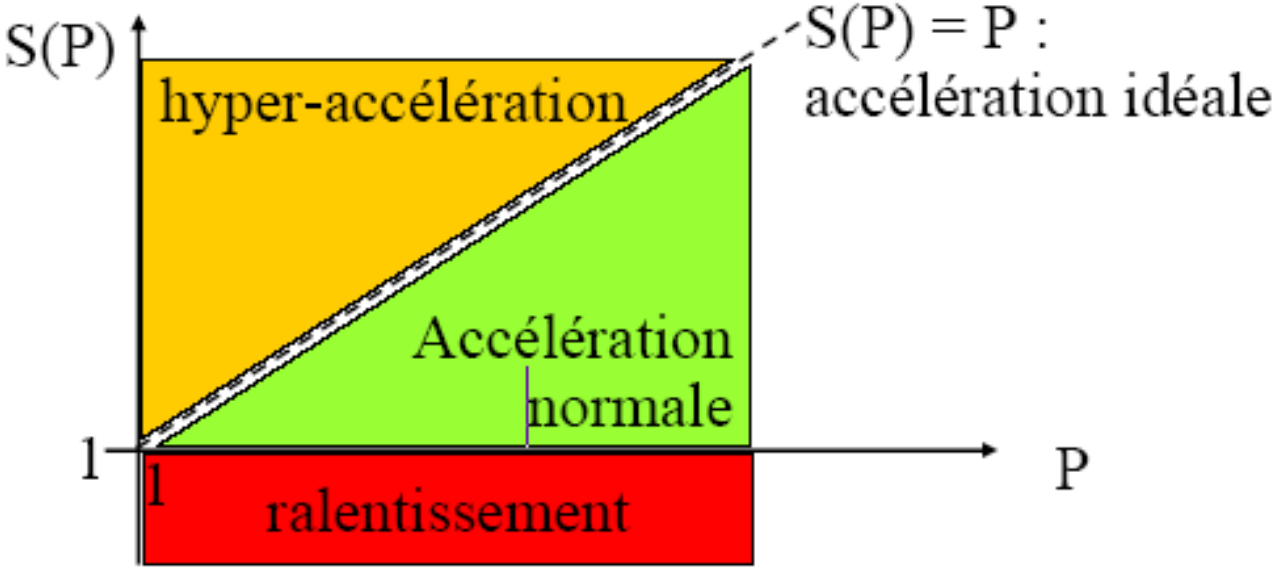
$$S(p) = \frac{t(1)}{t(p)}$$

$$E(p) = \frac{S(p)}{p}$$

- S(P) < 1 : on ralentit ; très mauvaise parallélisation

- 1 < S(P) < P : accélération “normal”. Plus on est près de P, meilleure est la parallélisation

- P < S(P) : hyper accélération. Analyser & justifier



Adaptation de maillage en parallèle

Remaillage : le contexte et la stratégie de parallélisation

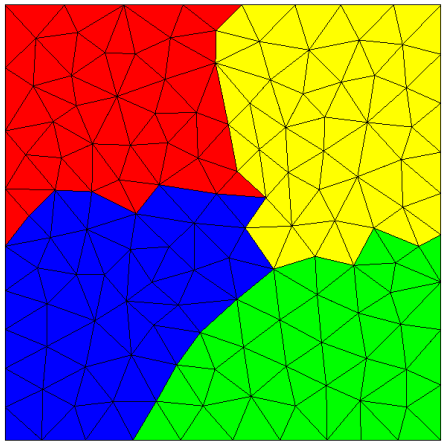
Le moteur de remaillage :

- il existe un mailleur séquentiel « mtc » qui reste en cours de développement
- mailleur tétraédrique non structuré et non hiérarchique
- taille de maille isotrope ou anisotrope

Parallélisation :

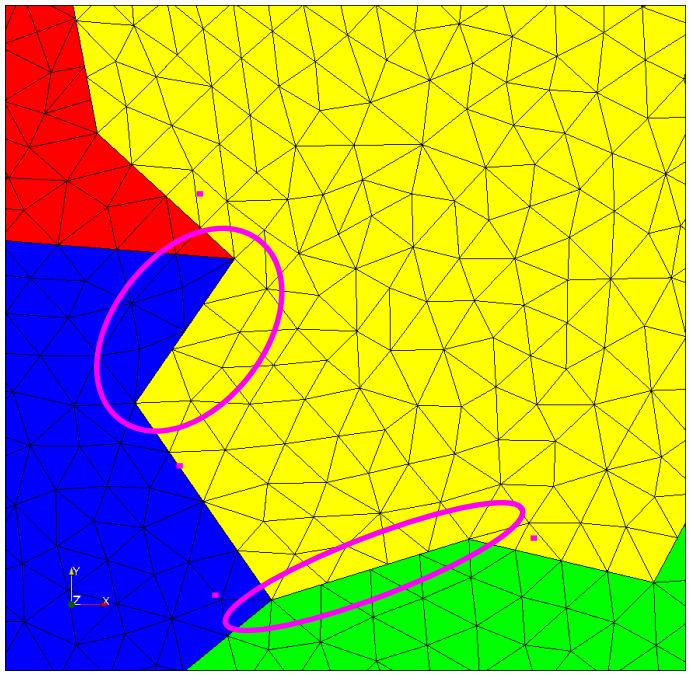
- une parallélisation directe est très intrusive
- nous avons un repartitionneur parallèle
- le repartitionneur et le remailleur utilisent tout les deux une stratégie d'amélioration itérative

Stratégie : Pas de parallélisation direct du mailleur mais utilisation de ce dernier dans un contexte parallèle.



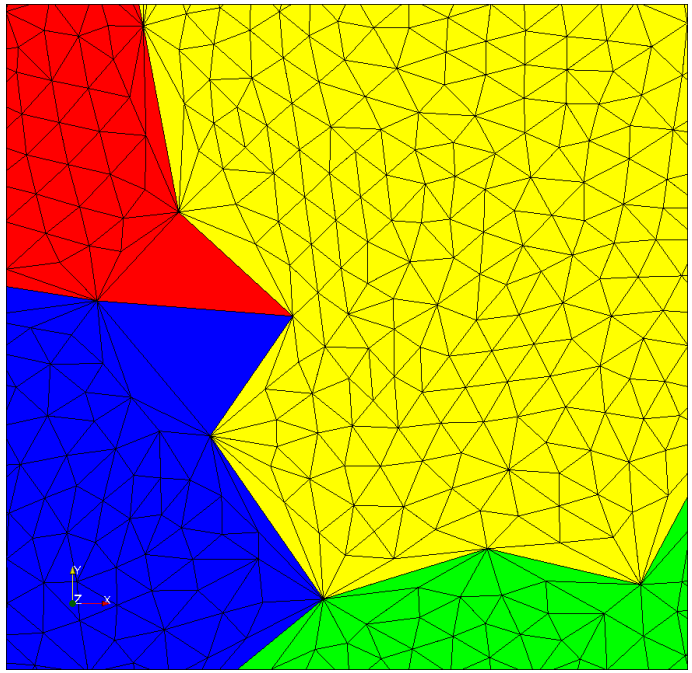
Remaillage : ajout d'une contrainte aux interfaces

1) Remaillage indépendant des sous domaines



Sans contrainte : nous n'obtenons pas un maillage globale conforme !

Avec la contrainte de bloquer les interfaces : nous obtenons un maillage globale conforme mais de bonne qualité.



2) Repartitionnement et itérer

Partition initiale du domaine sur 7 processeurs

Illustration dans un cas 2d :

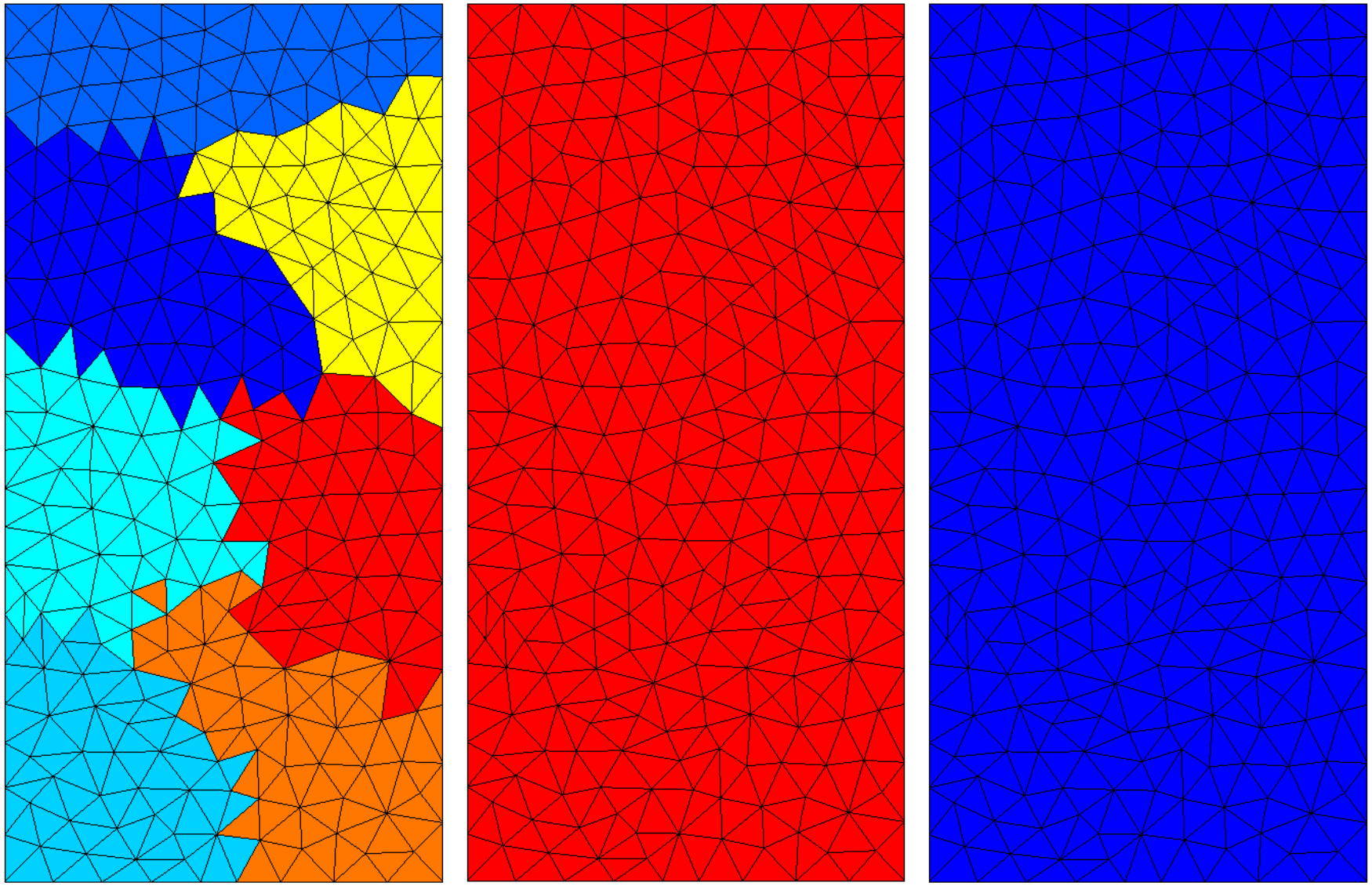


Illustration dans un cas 2d :

1er remaillage avec interfaces bloquées

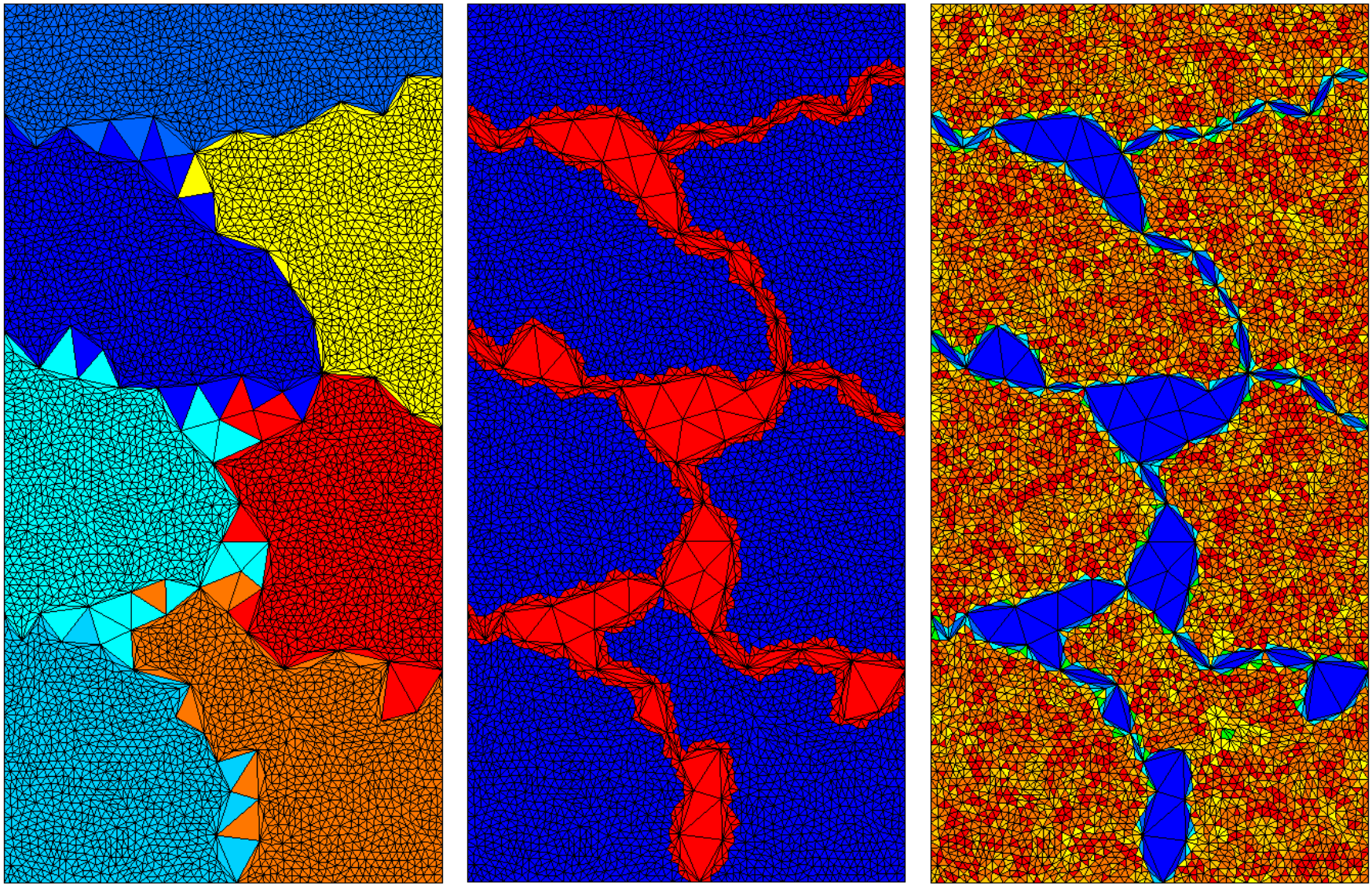
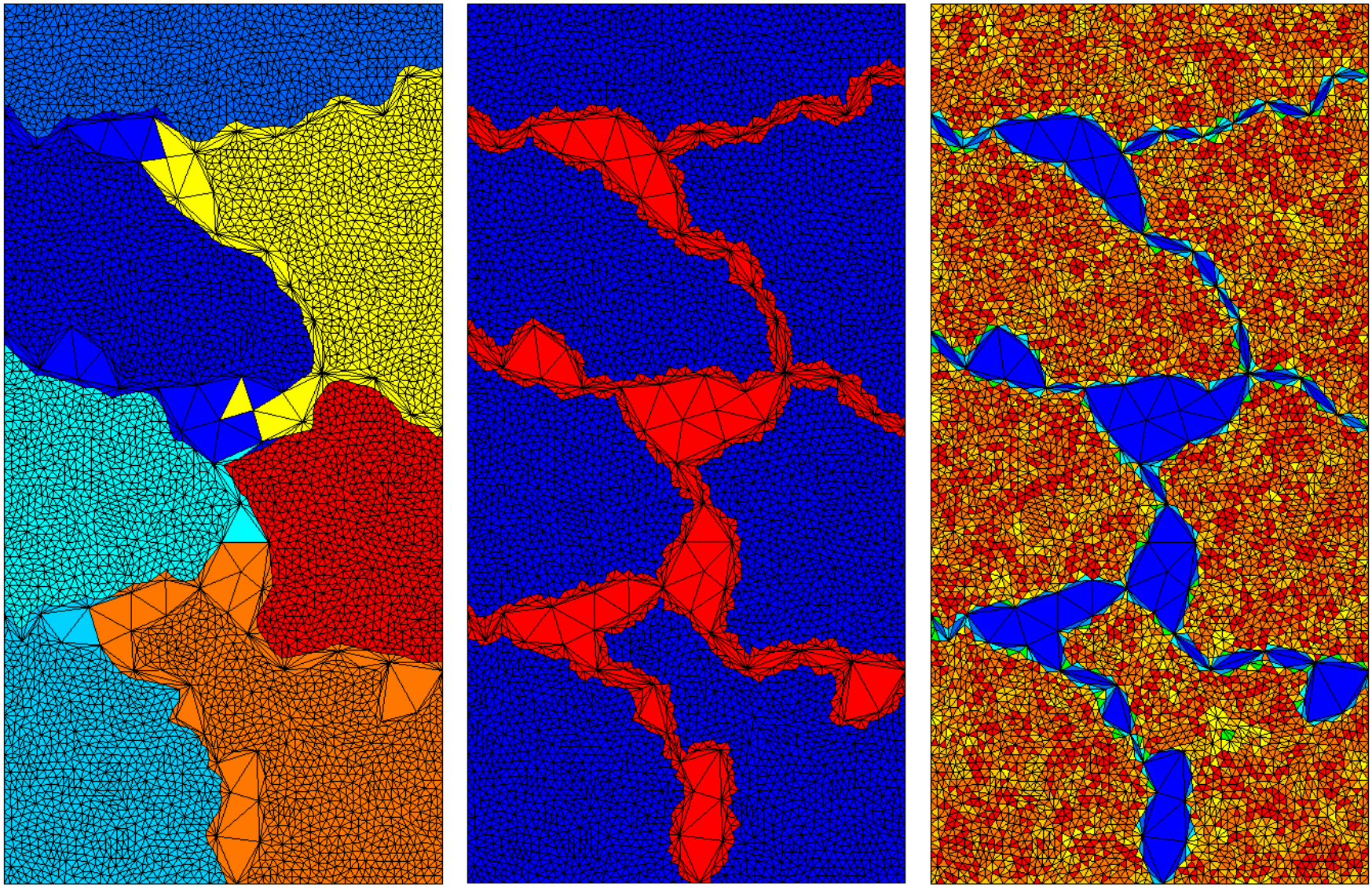


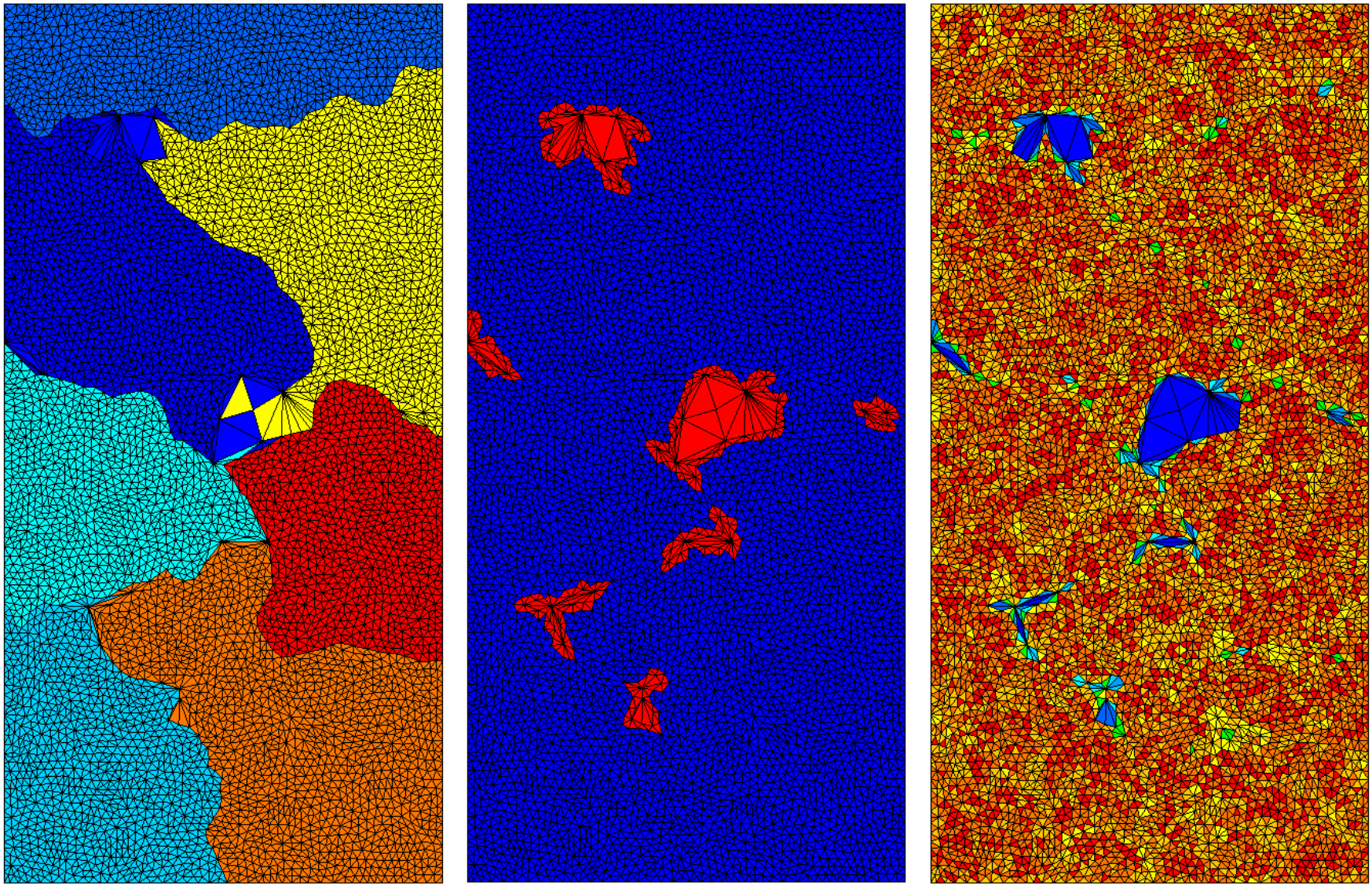
Illustration dans un cas 2d :

1er repartitionnement pour déplacer les interfaces à l'intérieur



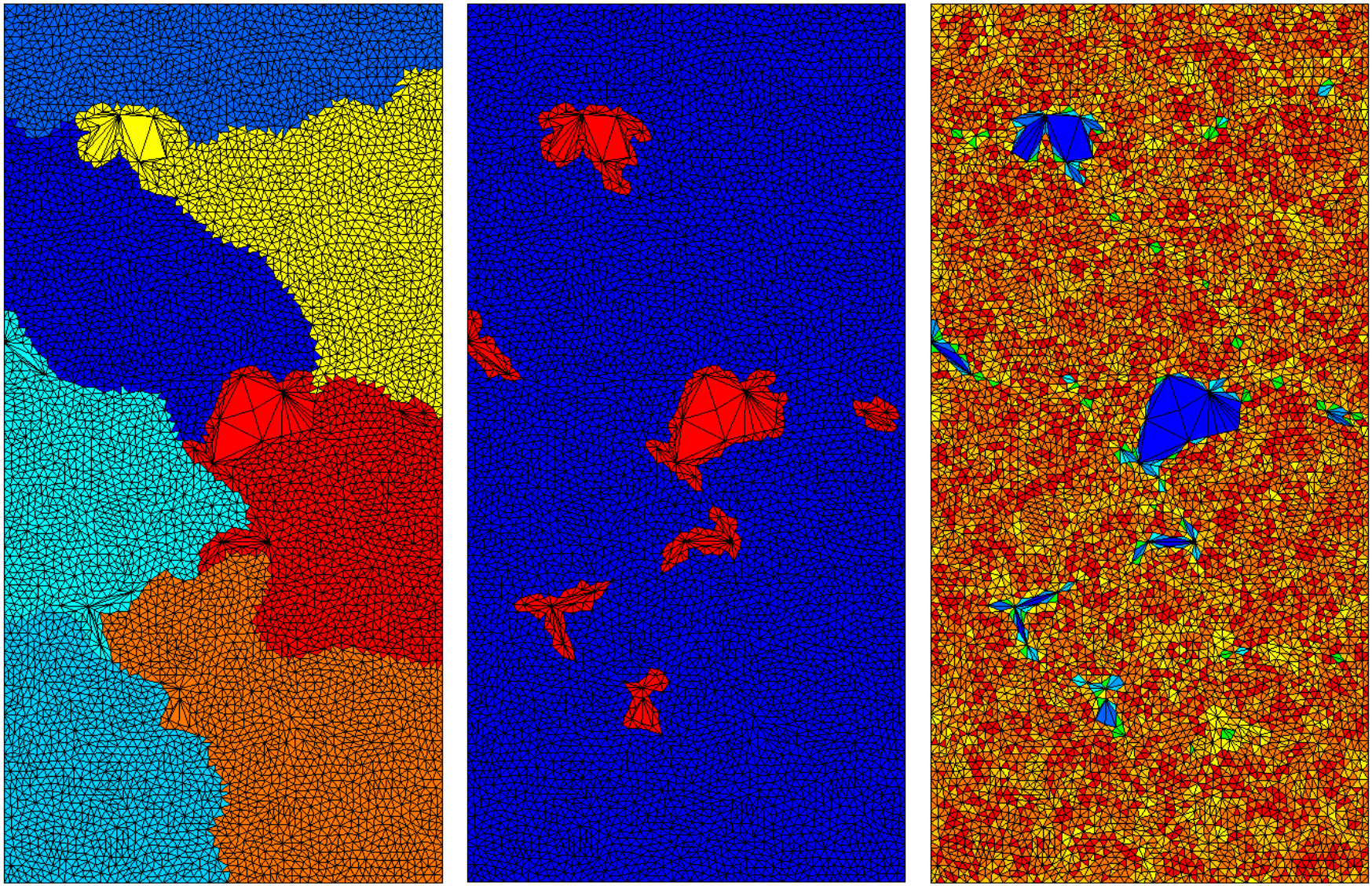
2ème remaillage

Illustration dans un cas 2d :



2ème repartitionnement

Illustration dans un cas 2d :



3ème remaillage

Illustration dans un cas 2d :

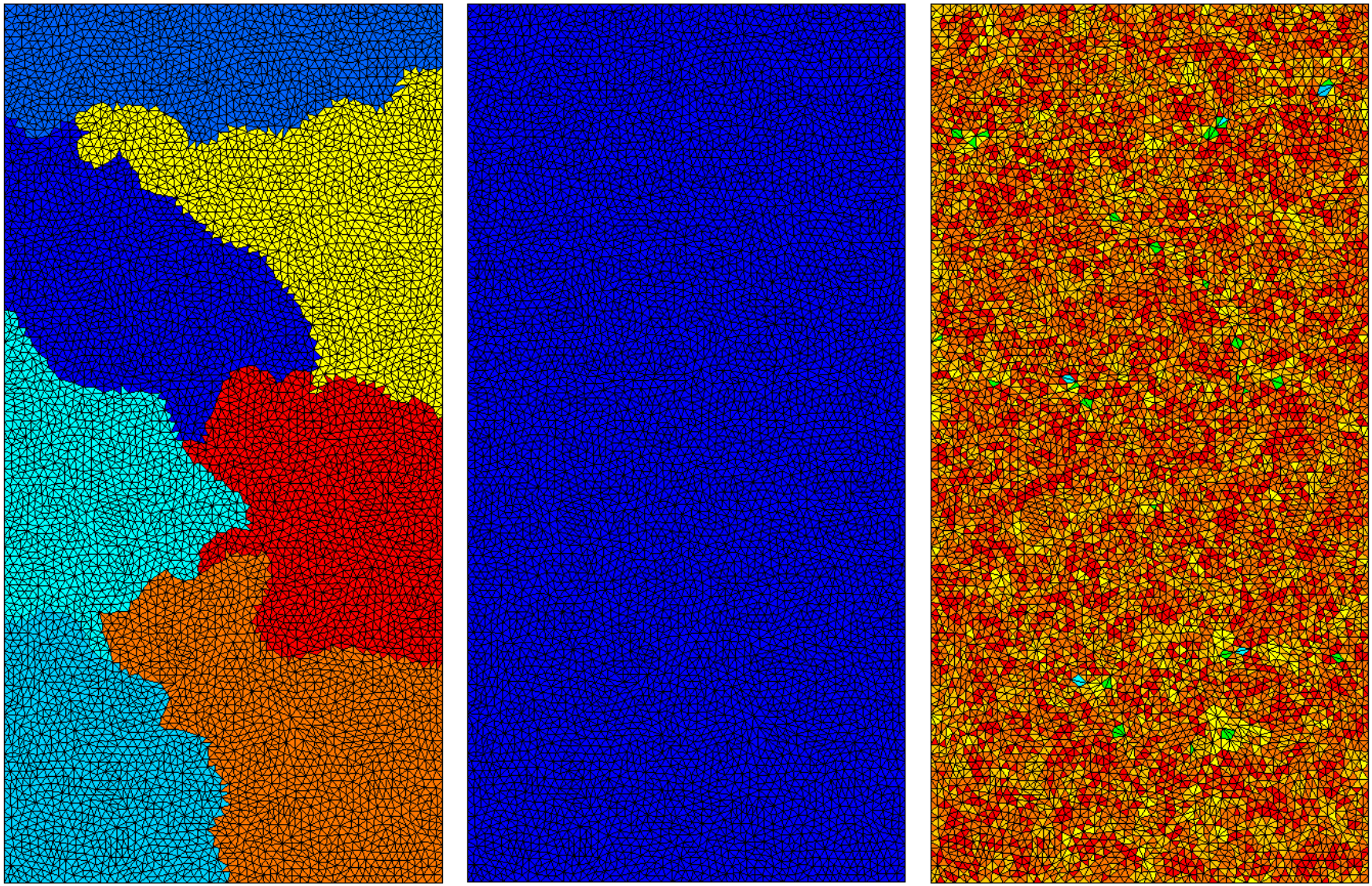
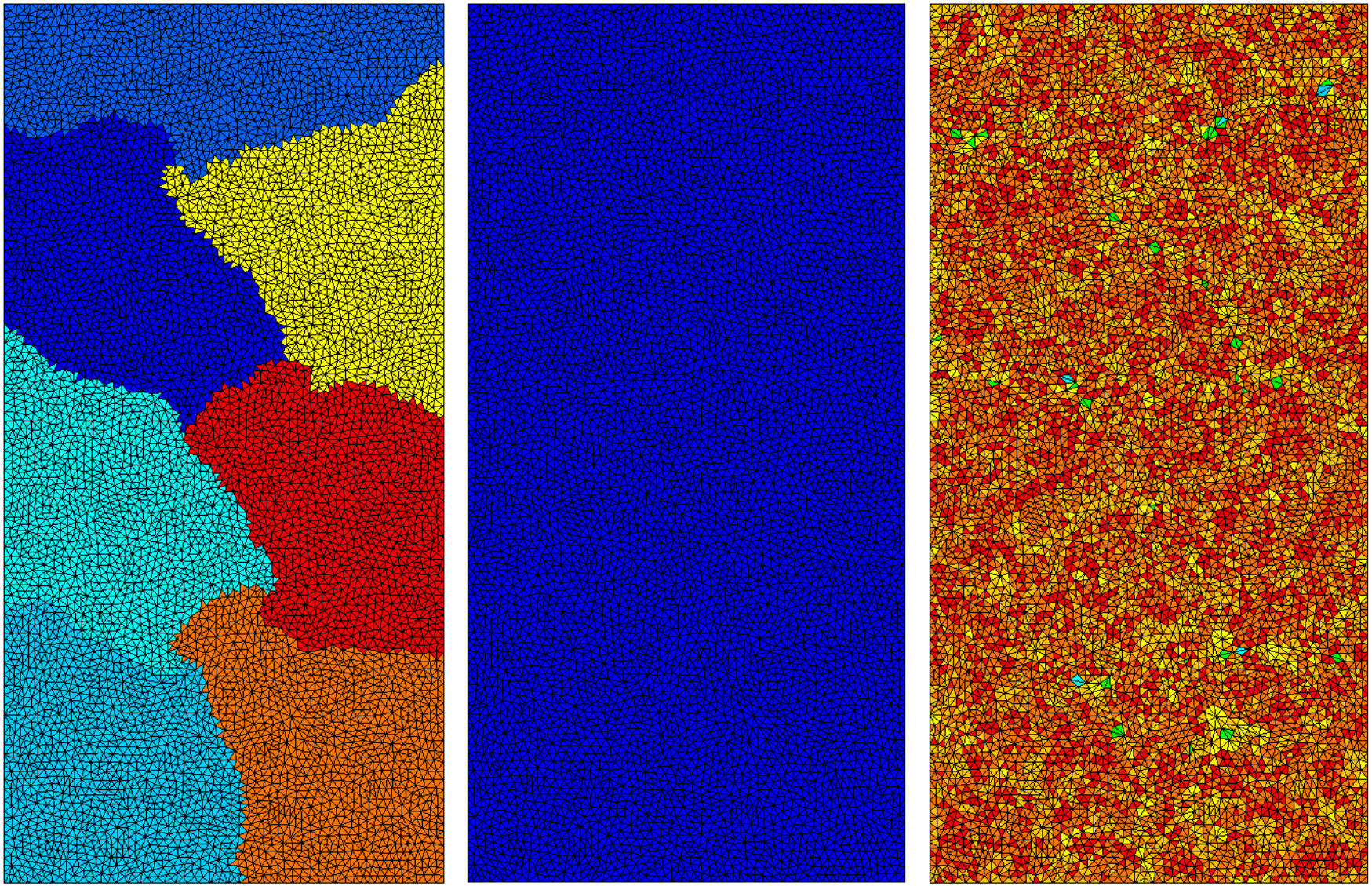


Illustration dans un cas 2d :

Dernier repartitionnement pour équilibrer la charge de travail pour les calculs EF



Commentaires :

L'étape de remaillage est de moins en moins coûteuse :

- en 2d, le premier remaillage est proportionnel à une surface, le second à une ligne et le dernier à un point.

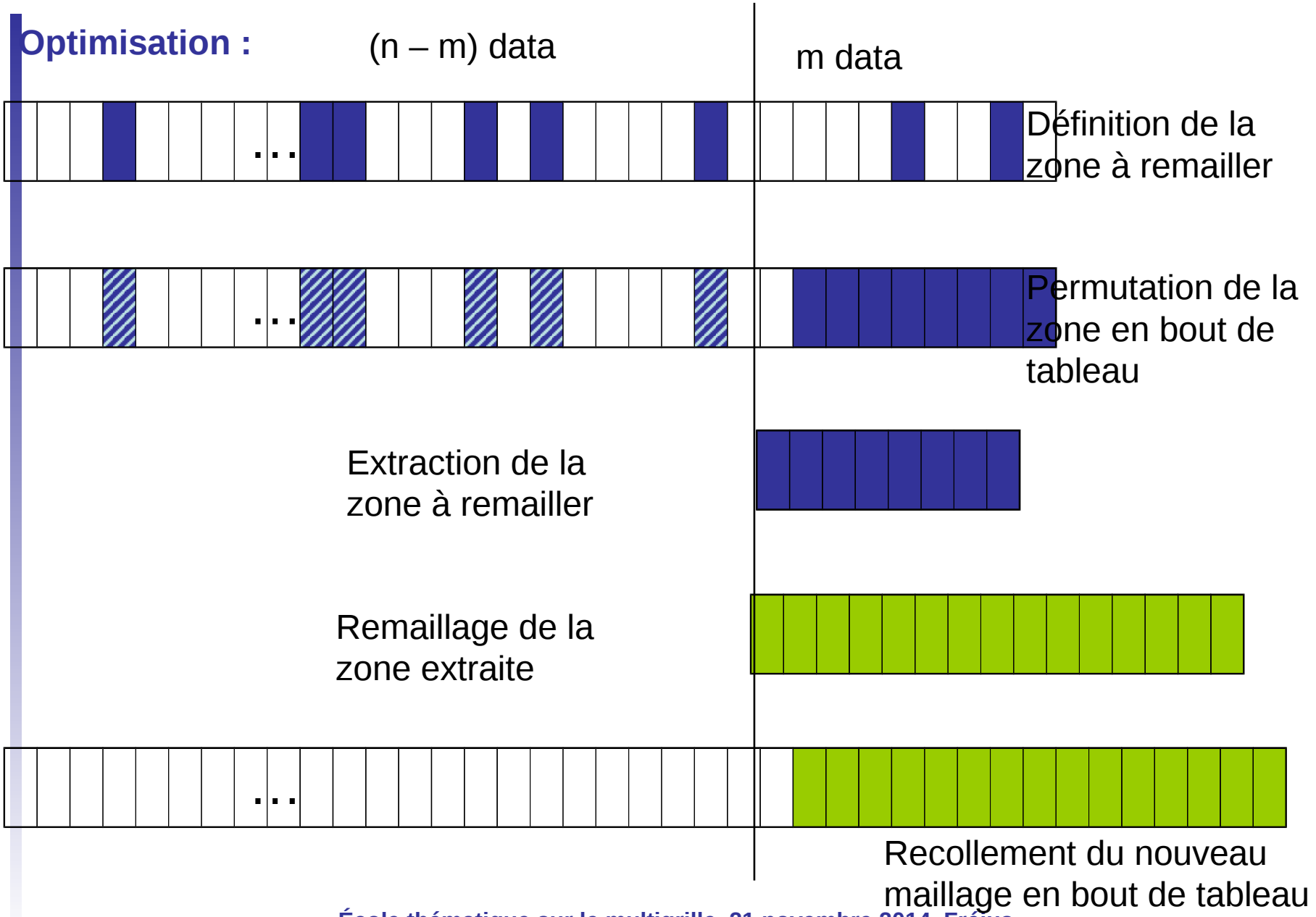
- en 3d, il y a une étape de plus : on part d'un remaillage volumique jusqu'au ponctuel.

Lien entre la dimension spatiale et le nombre d'itérations :

- 2d -> 3 itérations
- 3d -> 4 itérations
- Nd -> n+1 itérations "ceci marche également en 4d ;-)"

En théorie, le coût CPU de chaque itération diminue très rapidement et le coût total reste le même aux étapes de repartitionnement près. Numériquement, ceci est également vrai après avoir implémenté une optimisation par "permutation - couper - coller".

Optimisation :



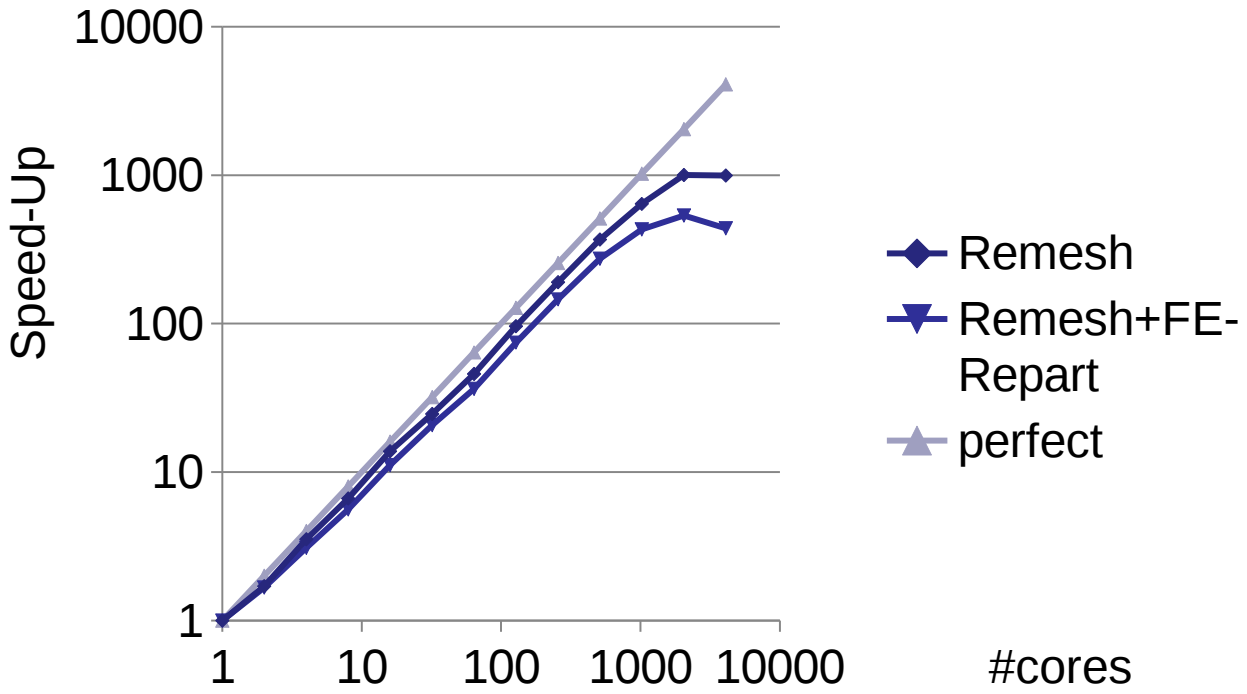
Performances parallèles :

- cas 2d avec un raffinement uniforme d'un facteur 2
- maillage initial de 800 000 nœuds et final de 3 200 000 nœuds
- exécuté sur 1 à 32 processeurs (cluster 512 cœurs AMD Opteron avec un réseau à faible latence infiniband)

	1p	2p	4p	8p	16p	32p
Durée Old	3071	1599	798	414	203	96
Accélération Old/réf	1.0/.7	1.9/1.3	3.8/2.5	7.4/4.9	15.1/10.	32.0/21.0
Durée New	2199	1077	487	285	135	63
Accélération New/réf	1.0/.9	2.0/1.9	4.5/4.2	7.7/7.1	16.3/15.	34.9/32.1
Durée Réf	2020	-	-	-	-	-

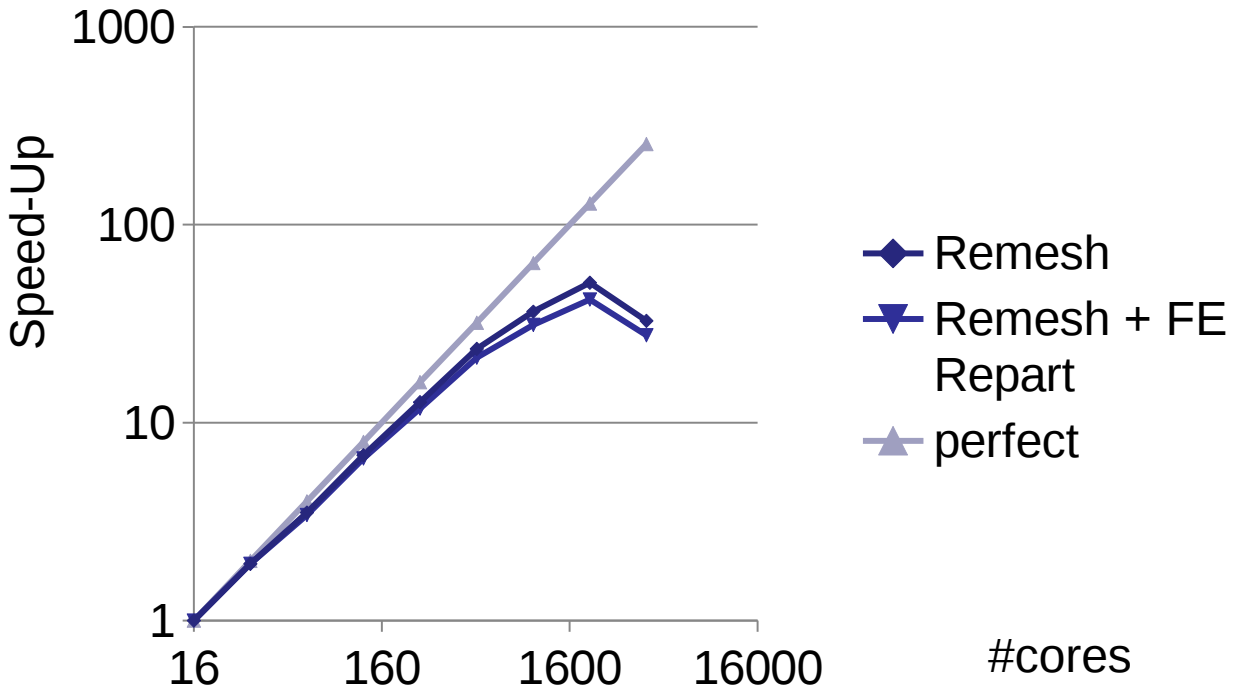
Performances parallèle : Hard Speed-Up

- Cas test 2d avec un raffinement uniforme du maillage d'un facteur 2
- Maillage initial de 5 000 000 de nœuds pour un final à 21 000 000
- Exécute en utilisant de 1 à 4096 cœurs de Curie.
- Le temps de remaillage passe de 3300s sur 1 cœur à 3.3s/6.3s sur 2048 cœurs



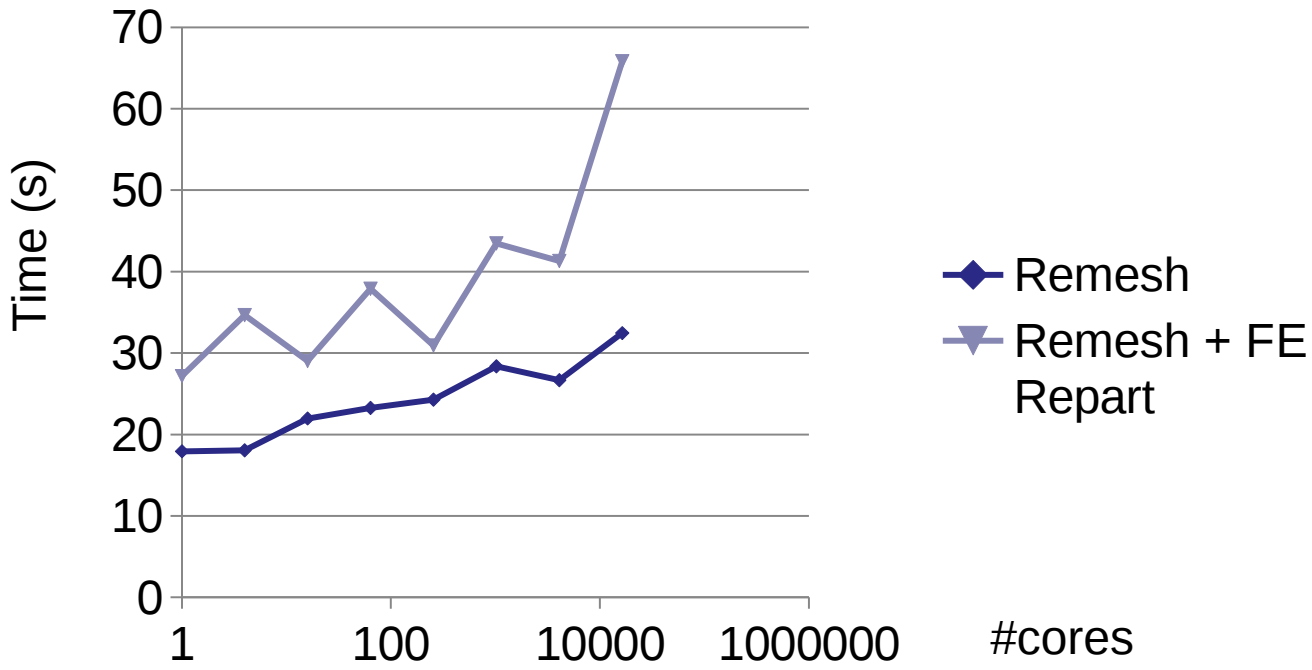
Performances parallèle : Hard Speed-Up

- Cas test 3d avec un raffinement homogène d'un facteur 2
- Maillage initial de 3,6 millions de nœuds pour un final de 30 millions.
- Exécutée en utilisant de 16 à 4096 cœurs
- Le temps de remaillage passe de 6800s sur 16 cours à 162s sur 2048 cœurs



Performances parallèle: Weak Speed-Up

- Cas test 2d avec un raffinent homogène d'un facteur 2
- Charge constante par cœur : maillage final avec 125 000 nœuds par cœur
- Exécuté de 1 à 16384 cœurs de Curie : maillage finale de 125 000 nœuds sur 1 cœur à 2 milliards sur 16 384 cœurs



Solveur multigrille parallèle

Pourquoi un solveur multigrille

- Nous somme capable de générer/adapter des maillages à plusieurs milliards de nœuds en utilisant plusieurs dizaines de milliers de cœurs

- Ces maillages ont pour objet principal de discrétiser un domaine de calcul pour résoudre des EDP.

- Même si les méthodes itératives se parallélisent très bien avec d'excellentes performances (Accélération quasi parfaite voir même super linéaire avec les effets de cache)

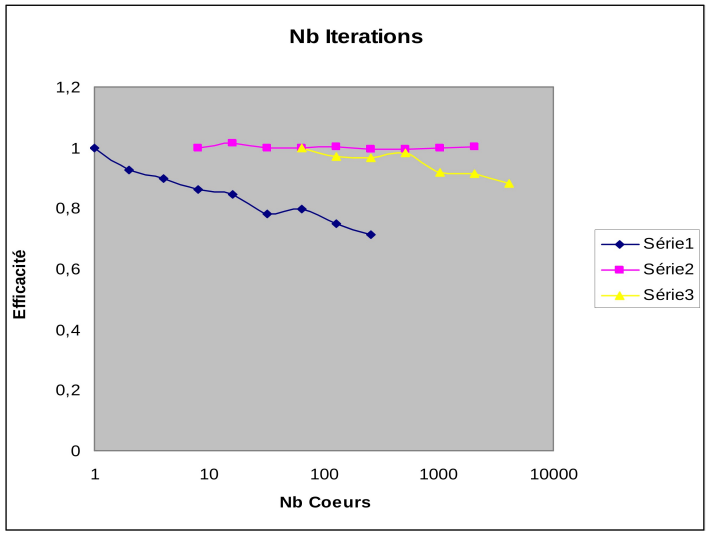
- La complexité algorithmique de ces dernières est telles qu'il est impossible d'envisager la résolution de systèmes linéaires à plusieurs milliards de degré de liberté

Multigrille : performance méthodes itératives

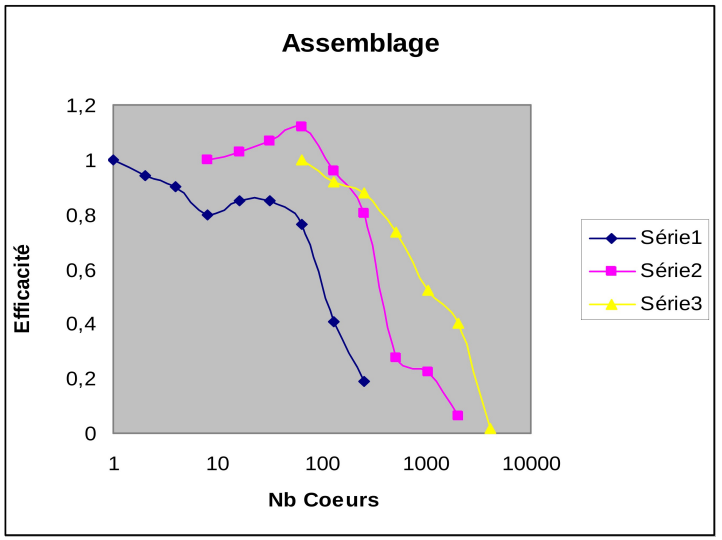
Efficacité parallèle des méthodes itératives pour un problème de Stokes incompressible avec une formulation éléments finis mixte vitesse/pression en P1+P1

3 séries de hard speed-up sur des maillages de tailles : 5 000, 500 000 et 5 millions de nœuds exécuté de 1 à 8192 cœurs sur Jade

Performance du préconditionneur ILU



Performance de l'assemblage



Multigrille : performance méthodes itératives

Performance de la résolution

Commentaires :

- Assez mauvaise performances de 1 à 8 cœurs
- Légère hyper accélération puis une nette hyper accélération (effet de cache)
- Avant un chute brutale d'efficacité

Multigrille : motivation

Problématique : la complexité non linéaire $O(n^{3/2}$ en 2d ou $n^{4/3}$ en 3d) des méthodes itératives représente un obstacle à la résolution de très grands systèmes.

Résolution de Stokes sur différents maillages.

Nb nœuds	8 073	32 205	128 354	512 661
Nb itérations	191	534	1381	3866
Assemblage	0.064	0.263	1.14	4.30
Résolution	0.90	9.02	102	1221

Nb nœuds	2 070	14 775	112 664	878 443
Nb itérations	55	137	348	931
Assemblage	0.112	0.971	7.737	61.68
Résolution	0.0768	1.467	36.52	836

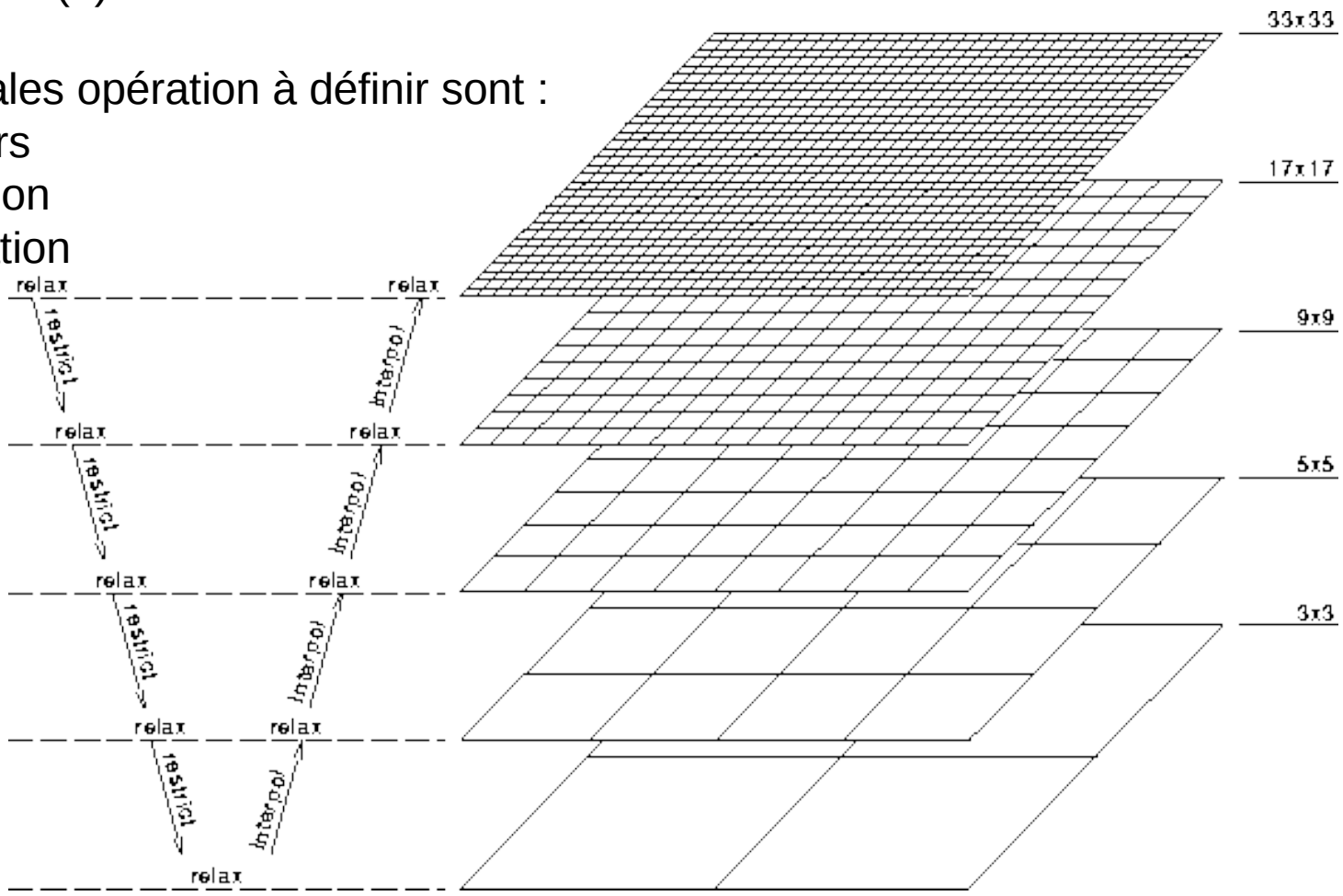
évolution du nombre d'itérations, des temps d'assemblage et de résolution en fonction du nombre de nœuds du maillage dans un cas 2d et 3d.

Solveur multigrille :

Les méthodes multigrille sont la réponse à l'extensibilité des solveurs au problème de très grande taille du faite d'une complexité quasi linéaire O(n log n) voir linéaire O(n).

Les principales opération à définir sont :

- les lisseurs
- la restriction
- l'interpolation



Solveur multigrille : utilisation de PETSc

PETSc est une bibliothèque comportant une large gamme de solveur itératifs parallèles ainsi qu'une large variété de préconditionneur parallèle.

Elle présente de très bonne performance parallèle et est largement utilisée dans les codes de simulation numérique utilisant une formulation implicite.

Elle inclus une interface de développement afin d'implémenter un préconditionneur de type multigrille PCMG. Le développeur doit cependant fournir les fonctionnalités suivantes :

- Les systèmes à résoudre sur chaque niveaux:
  - Discrétisation du problème physique (Géométrique MG)
  - Récursive réduction du problème fin (Algébrique MG)

$$A_{n-1} = {}^{t}I_{n-1,n} A_n I_{n-1}$$

- Opérateurs d'interpolation $I_{n-1,n}$ et/ou de restriction $R_{n,n-1}$ entre deux niveaux.

Solveur multigrille : contraintes

Mous voulons une solution qui soit **flexible** et **robuste :**

- qui fonctionne avec une formulation éléments finis
- sur des maillages indépendants (pas d'imbrication des éléments, ou des nœuds)
- sur des domaines discrétisés pas forcement identiques (cas des bords courbes)
- et avec des partitions de domaine pas forcement identiques
  - partitions non imbriquées
  - voir pas avec le même nombre de sous domaines

Solveur multigrille : opérateur de restriction/interpolation

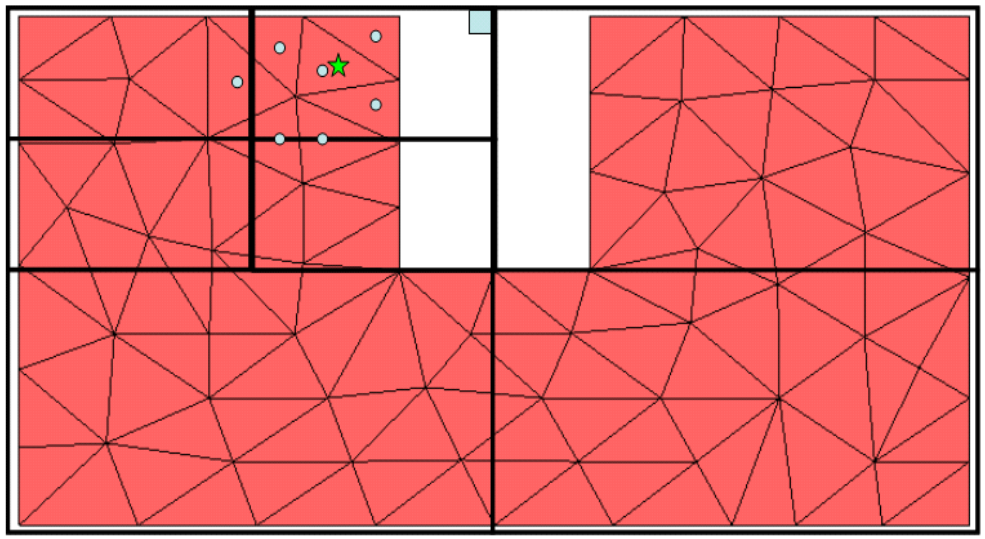
Ces opérateurs ont pour but de transporter des champs d'un maillage à un autre :

- Interpolation : du plus grossier au plus fin.
- Restriction : du plus fin au plus grossier.

Une façon simple de définir ces opérateurs est d'utiliser les coordonnées barycentriques (des nœuds d'un maillage sur les éléments de l'autre).

La construction de cet opérateur repose sur un algorithme de localisation rapide de l'élément contenant un nœud.

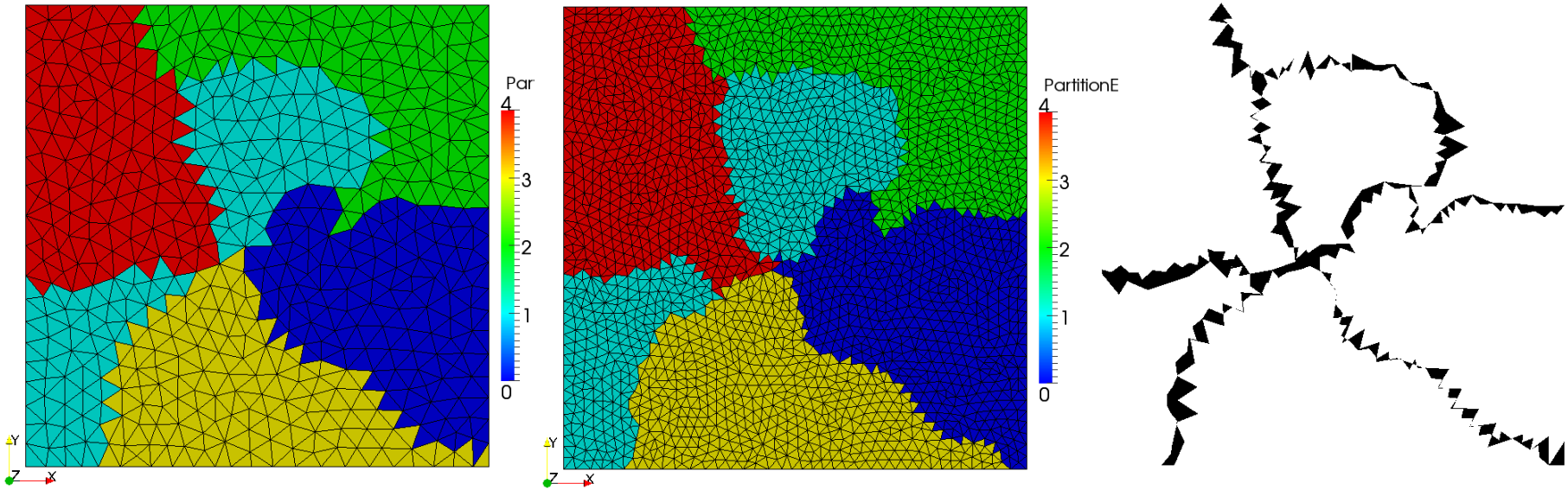
Pour cela nous utilisons une structure d'octree de façon à réduire la complexité algorithmique à O(n log n)



Solveur multigrille : opérateur parallèle de restriction/interpolation

Dans un contexte parallèle ces opérateurs s'appuient sur des maillages partitionnés. Il est donc nécessaire d'effectuer des localisations locales et d'autres externes.

Grâce à la technique de remaillage choisi, les partitions des deux maillages sont très proches (bien que différentes), et le nombre de localisations externes représente uniquement un petit pourcentage des nœuds.

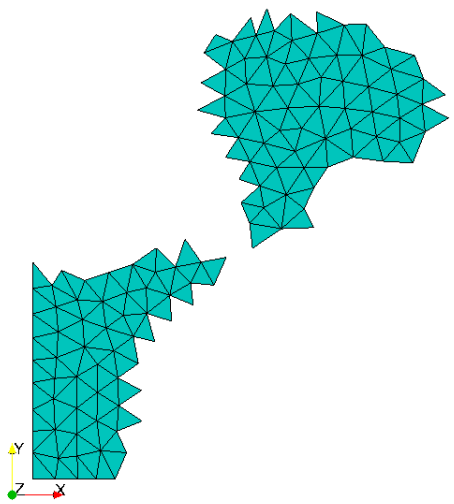


Solveur multigrille : opérateur parallèle de restriction/interpolation avec utilisation de filtre.

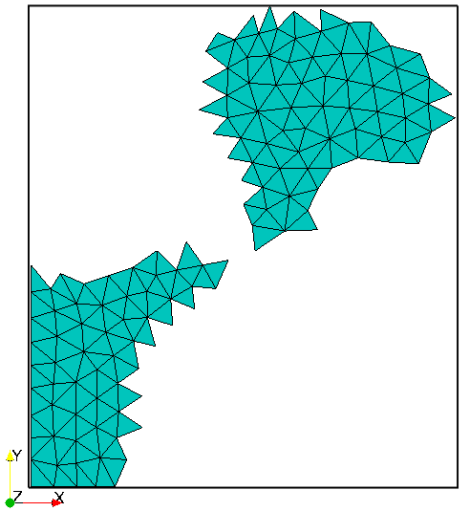
Le calcul parallèle rend les détails prépondérants !!!

Par exemple avec en moyenne de seulement 5% de nœuds externes mais en utilisant 1000 cœurs, on se retrouve à devoir localiser 5 fois plus de nœuds externes que de nœuds locaux. Ce qui entraîne un dépassement mémoire.

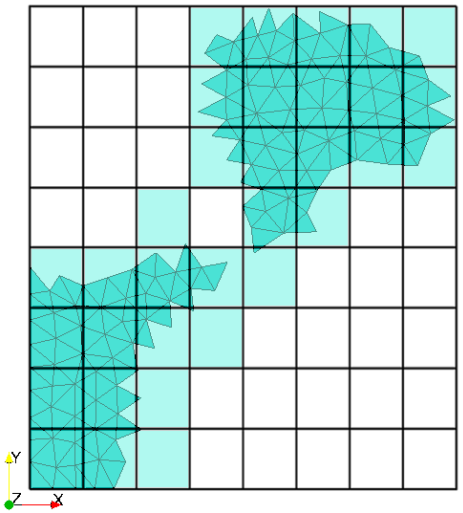
Pour lutter contre ceci on introduit des filtres qui limitent les faux positifs : boite englobante ou mieux un masque de pixel.



Le sous domaine



Boite englobante



Masque de pixel

Solveur multigrille : performance des opérateurs de restriction/interpolation

Performance des opérateurs de transport :

* Opt 0 : tous les nœuds externes sont envoyer aux autres cœurs
* Opt 1 : on utilise la boite englobante.
* Opt 2 : on utilise le masque de pixel ; lvl représente la résolution des pixels

Test réaliser sur 512 cœurs entre un maillage à 800 000 nœuds et un autre à 7.25 millions.

Nb nœuds	Calculés	Estimés-Envoyés	Temps (s)
Opt 0	6972995	6972995	3.31
Opt 1	30152	4314393	0.165
Opt 2 – lvl 1	30152	4323074	0.131
Opt 2 – lvl 2	30152	2438244	0.111
Opt 2– lvl 3	30152	665052	0.22
Opt 2– lvl 4	30152	203614	0.52

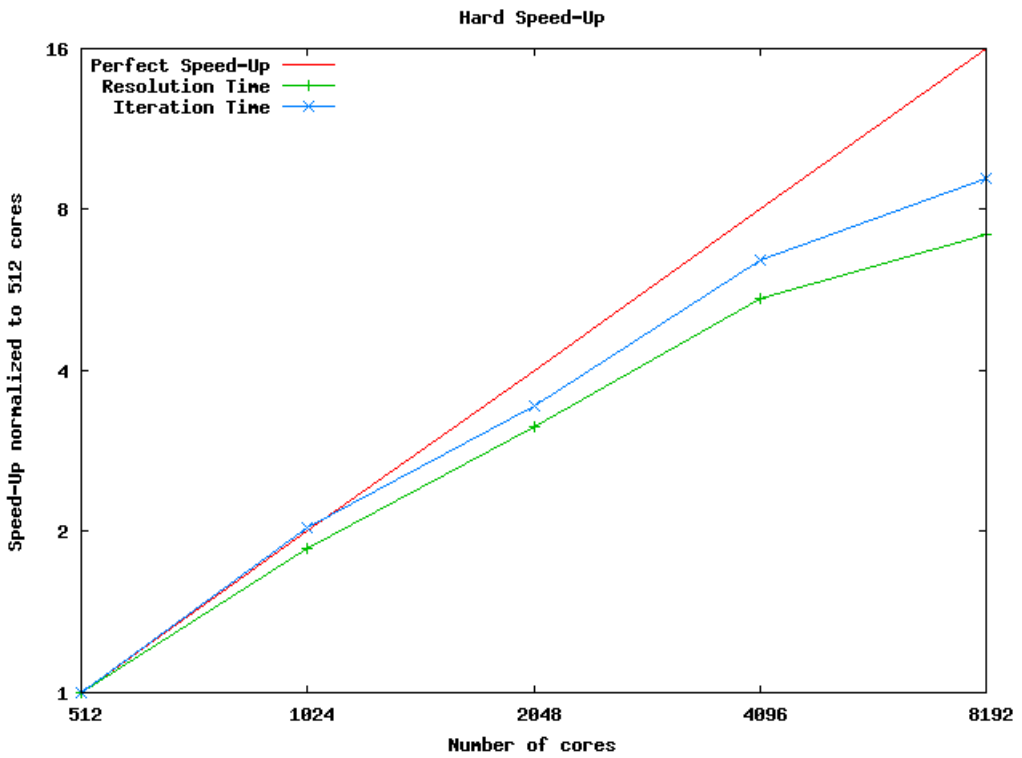
Solveur multigrille : performance parallèle (Hard speed-up)

Ce cas test consiste en la résolution des équations de Stokes incompressible en formulation éléments finis mixte (vitesse,pression) discrétisé par un élément P1+/P1. Maillage 2d de 216 millions de nœuds (650 millions de ddl). Exécuté sur 512 à 8192 cœurs avec un solveur multigrille à 8-niveaux.

Temps de résolution :

de 96.7s avec 11 itérations sur 512 cœurs

à 13.4s avec 14 itérations sur 8192 cœurs

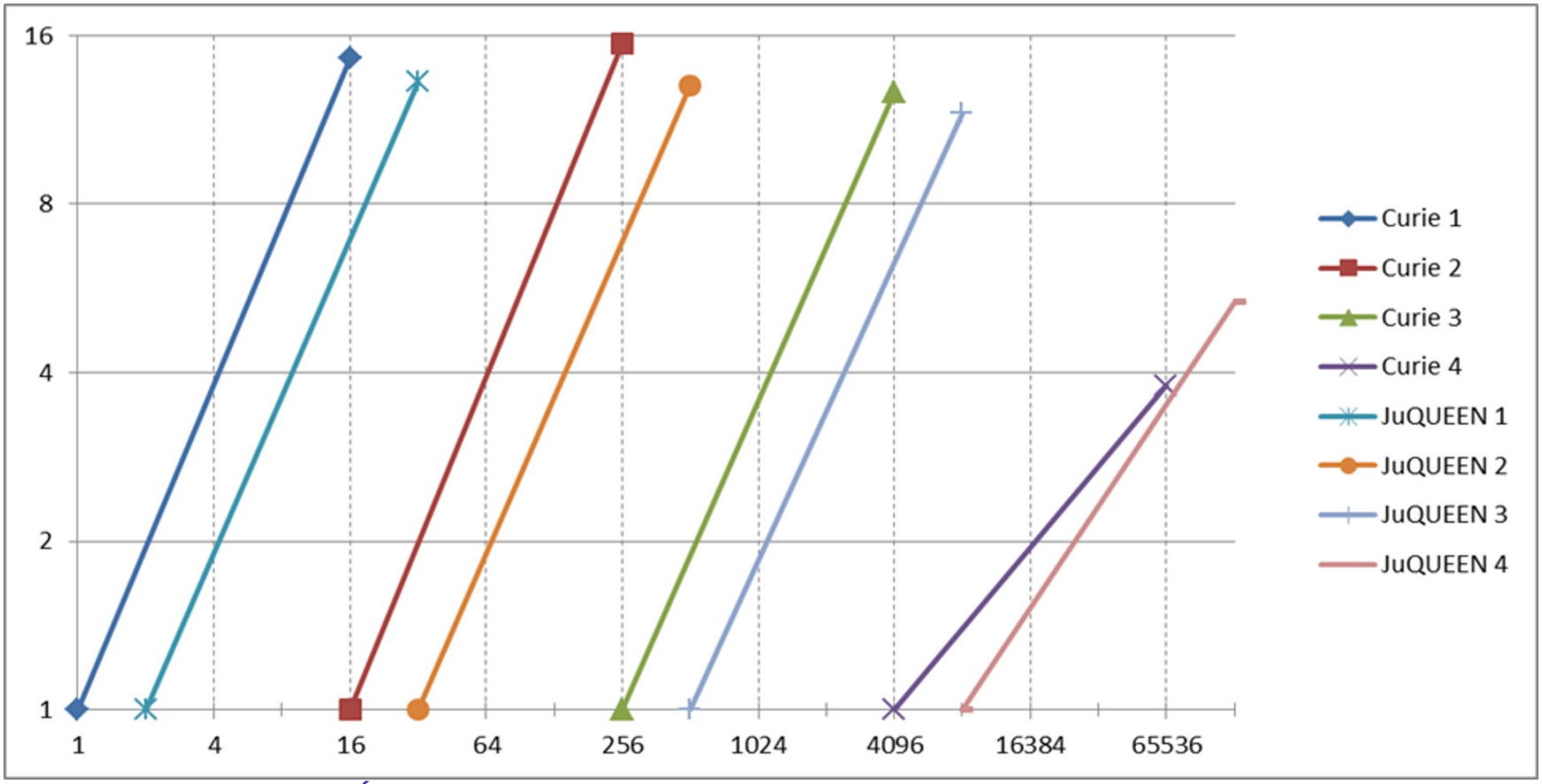


Performance à l’échelle des calculateurs Tie0

Maillage parallèle : hard speed-up pour un cas 2d

Accélérations obtenues pour un raffinement d'un facteur 4
en utilisant 16 fois plus de cœurs

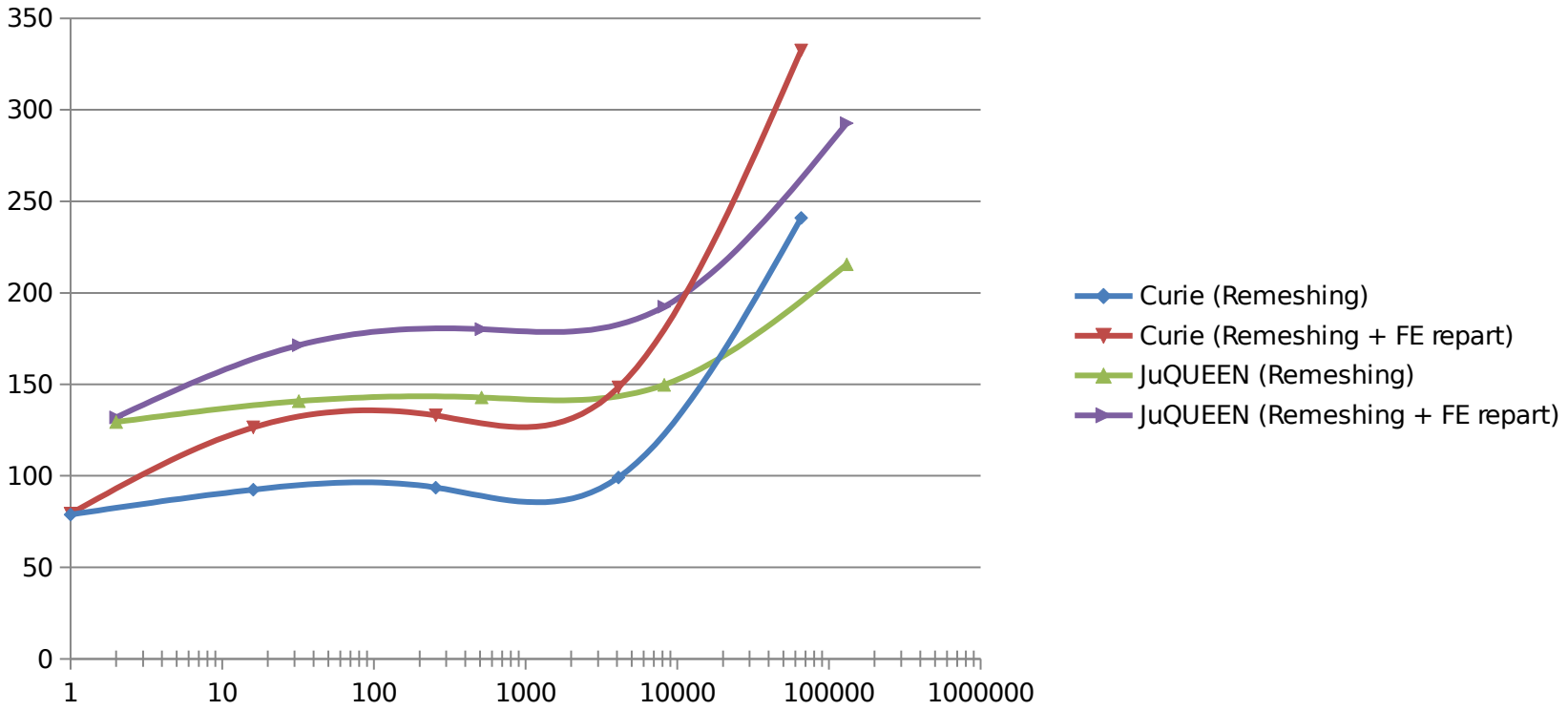
Bonnes performances jusqu'à 8192 cœurs légère dégradation au-delà.



Maillage parallèle : weak speed-up pour un cas 2d

Exécution de 1 à 131 072 cœurs, temps de remaillage pour un raffinement d'un facteur 4 avec une charge de travail constante par cœur : maillage adapté avec environ 500 000 nœuds sur Curie et 125 000 sur JuQUEEN.

La aussi d'excellentes performances jusqu'à 8192 cœurs et dégradation au-delà.



Solveur multigrille : weak speed-up pour un cas 2d sur Curie

Exécution de 16 à 65 536 cœurs pour une convergence relative à 1e-9

Cas « record » à 100 milliards de degrés de libertés

Temps d'assemblage et de résolution relativement stables
exceptés à 65 mille cœurs où l'on a une dégradation des performances

Nb cœurs	16	256	4 096	65 536
Nb niveaux	4	6	7	8
Nb Lissages	10	10	10	10
Nb nœuds	8 157 137	130 503 221	2 087 280 602	33 394 443 636
Nb éléments	16 314 272	261 006 440	4 174 561 202	66 788 887 270
Nb ddl	24 471 411	391 509 663	6 261 841 806	100 183 330 908
Nb itérations (s)	13	17	16	22
Assemblage (s)	10.09	10.41	11.84	61.43
Résolution (s)	218.8	274,3	263.9	431.6

Solveur multigrille : weak speed-up pour un cas 2d sur JuQUEEN

Exécution de 64 à 262 144 cœurs pour une convergence relative à 1e-9

Cas « record » à 100 milliards de degrés de libertés

Temps d'assemblage qui augmente plus rapidement que les temps de résolution à un grand nombre de cœurs

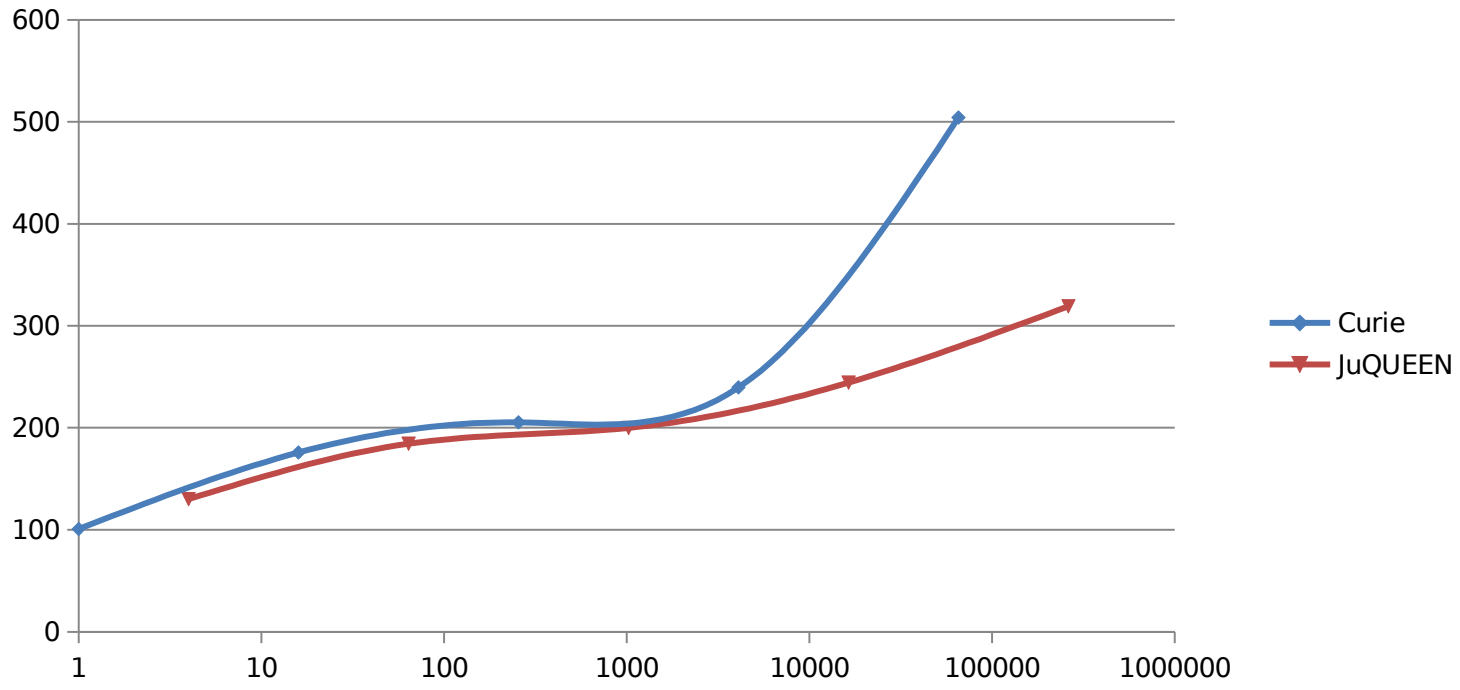
Nb cœurs	64	1 024	16 384	262 144
Nb niveaux	4	5	6	7
Nb Lissages	5	5	5	5
Nb nœuds	8 216 917	131 689 655	2 108 867 176	33 777 732 848
Nb éléments	16 433 832	263 379 308	4 217 734 350	67 555 465 694
Nb ddl	24 650 751	395 068 965	6 326 601 528	101 333 198 544
Nb itérations (s)	15	18	18	19
Assemblage (s)	28.27	29,1	34.2	88.9
Résolution (s)	164.5	197,6	207.3	226.0

Solveur multigrille : weak speed-up pour un cas 2d

Exécution de 1 à 262 144 cœurs pour une convergence relative à 1e-9

Cas final à 100 milliards de degrés de libertés

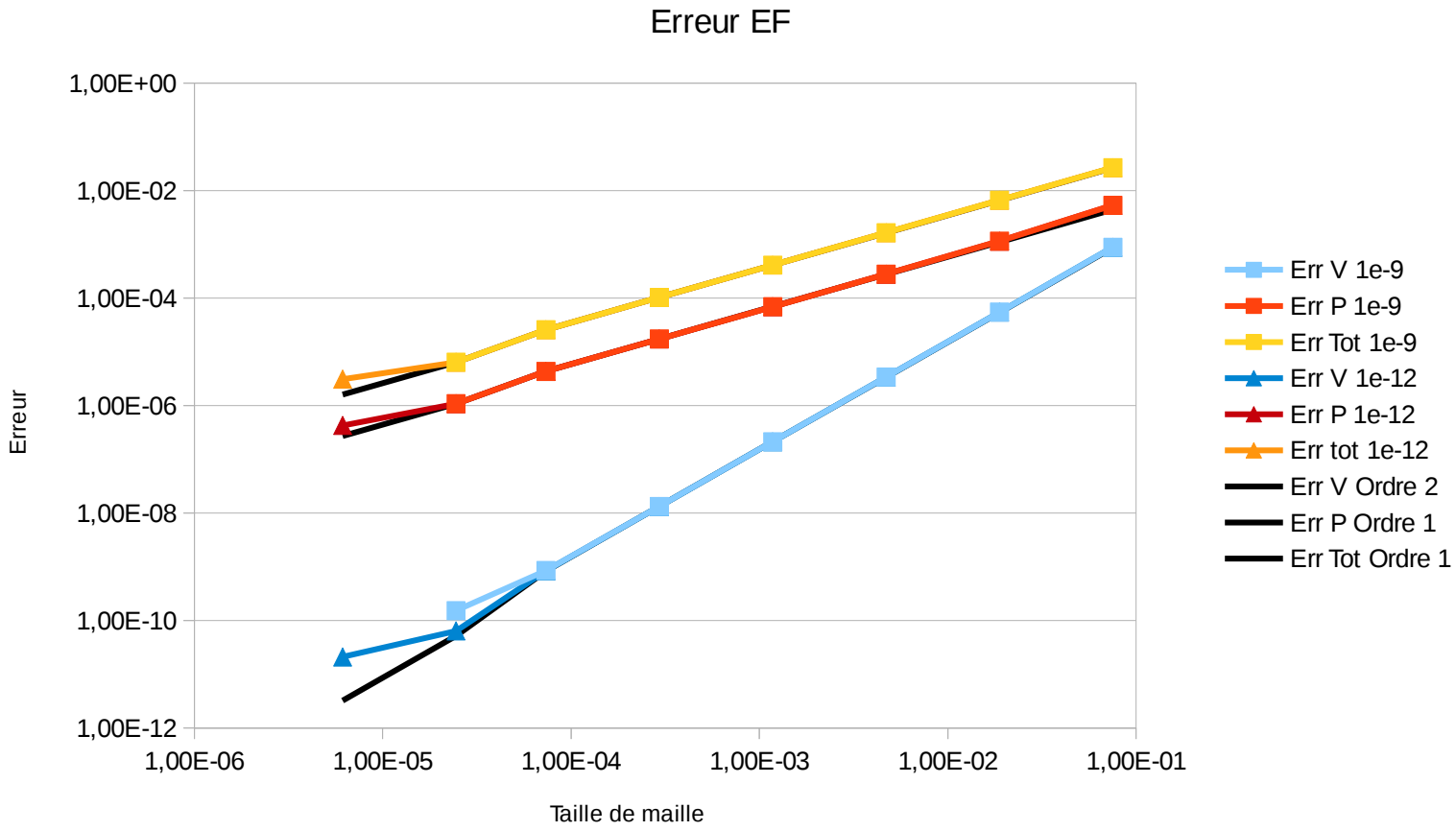
On observe de très bonne performance jusqu'à 10 000 cœurs et une dégradation au-delà (surtout sur Curie).



Solveur multigrille : évolution de l'erreur EF

Convergence linéaire pour la pression et quadratique en vitesse

Limite du réel « double précision » ($1e^{-14}$) sur le cas le plus fin ?



Solveur multigrille : weak speed-up pour un cas 3d sur Curie

Exécution de 256 à 65 536 cœurs pour une convergence relative à 1e-9

Cas « record » à 55 milliards de degrés de libertés

Temps d'assemblage non négligeables vis-à-vis des temps de résolutions

Nb cœurs	256	1 024	16 384	131 072
Nb niveaux	4	4	5	6
Nb Lissages	3	3	3	3
Nb nœuds	56 495 278	215 328 185	1 721 643 234	13 757 688 739
Nb éléments	334 019 712	1 273 910 398	10 191 081 139	81 464 208 387
Nb ddl	225 981 112	861 312 740	6 886 572 936	55 030 754 956
Nb itérations (s)	10	9	12	13
Assemblage (s)	40.30	41,48	48.34	186.4
Résolution (s)	112.0	115,6	126.7	282.6

Solveur multigrille : weak speed-up pour un cas 3d sur JuQUEEN

Exécution de 256 à 131 072 cœurs pour une convergence relative à 1e-9

Cas « record » à 21 milliards de degrés de libertés

Temps d'assemblage étonnants et quasi prépondérant

Dégradation importante des performances sur 131 mille cœurs

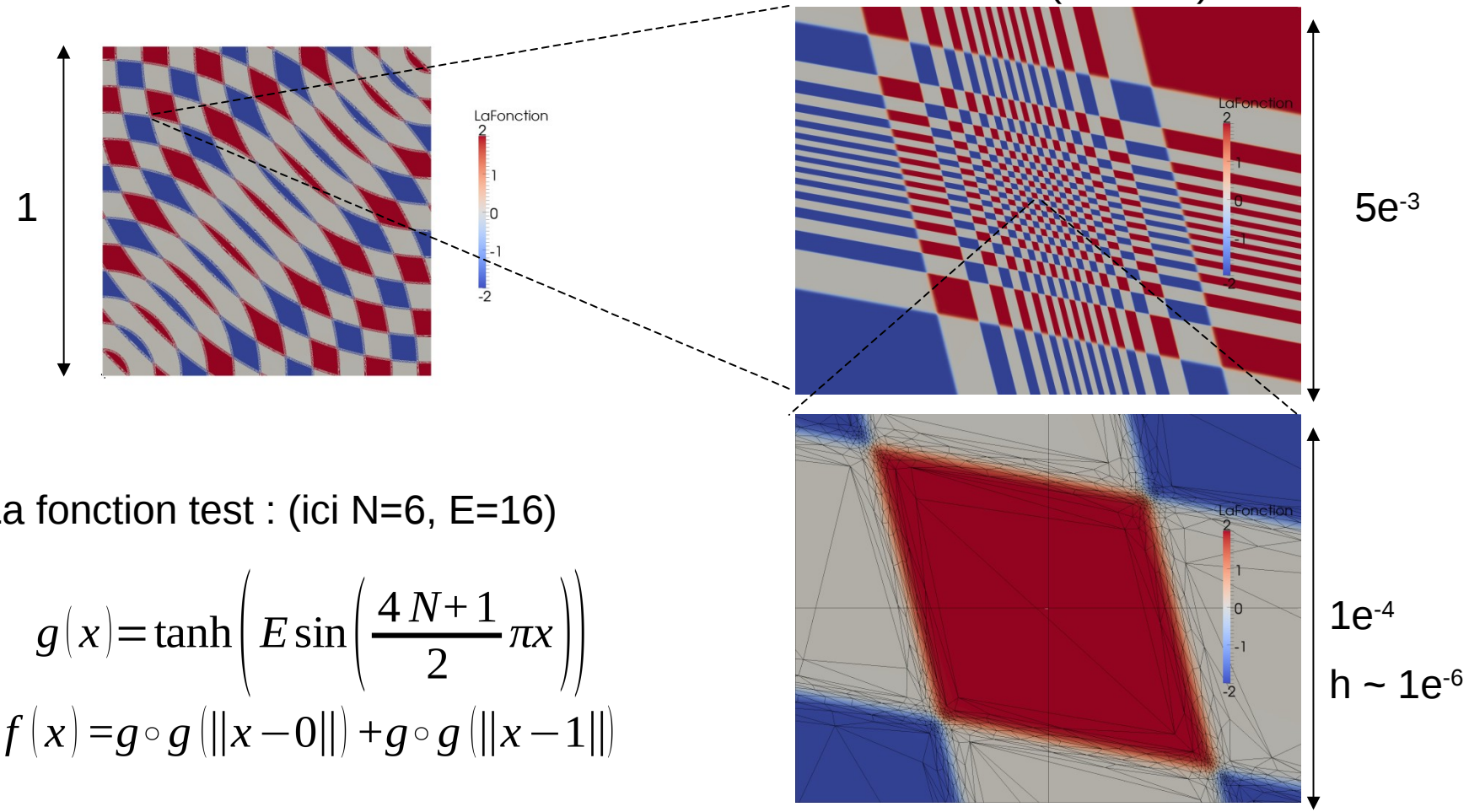
Nb cœurs	256	2 048	16 384	131 072
Nb niveaux	5	5	6	7
Nb Lissages	5	5	5	5
Nb nœuds	10 635 346	84 237 716	676 982 712	5 420 755 850
Nb éléments	62 707 274	497 667 191	4 001 133 874	32 045 959 130
Nb ddl	42 541 384	336 950 864	2 707 930 848	21 683 023 400
Nb itérations (s)	11	12	12	16
Assemblage (s)	57.59	62.23	112.4	634
Résolution (s)	109.4	136.3	121.8	824

Exemples d'applications

Application : adaptation statique (avec un estimateur d'erreur anisotrope)

Test 2d avec un maillage de 25 000 000 nœuds

150 itérations exécutées sur 512 cœurs en 6 059s (1h 41m) sur Jade



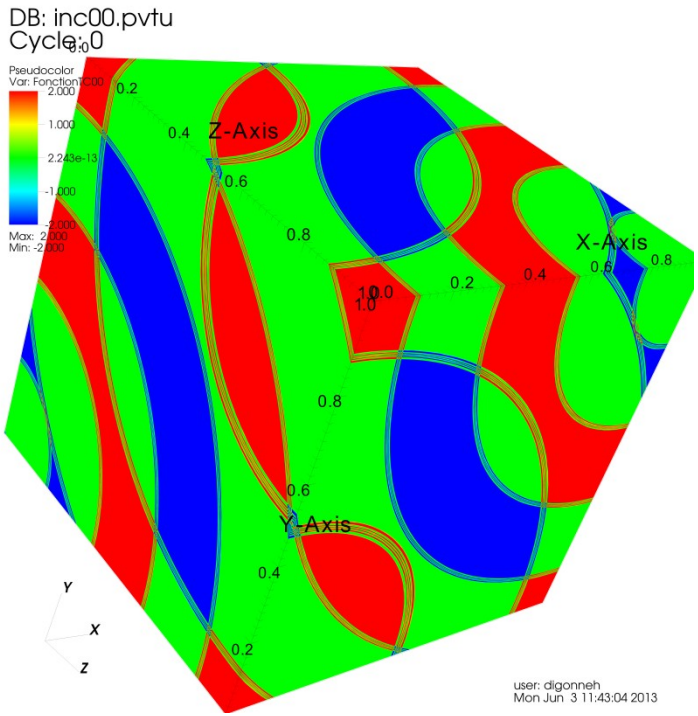
La fonction test : (ici N=6, E=16)

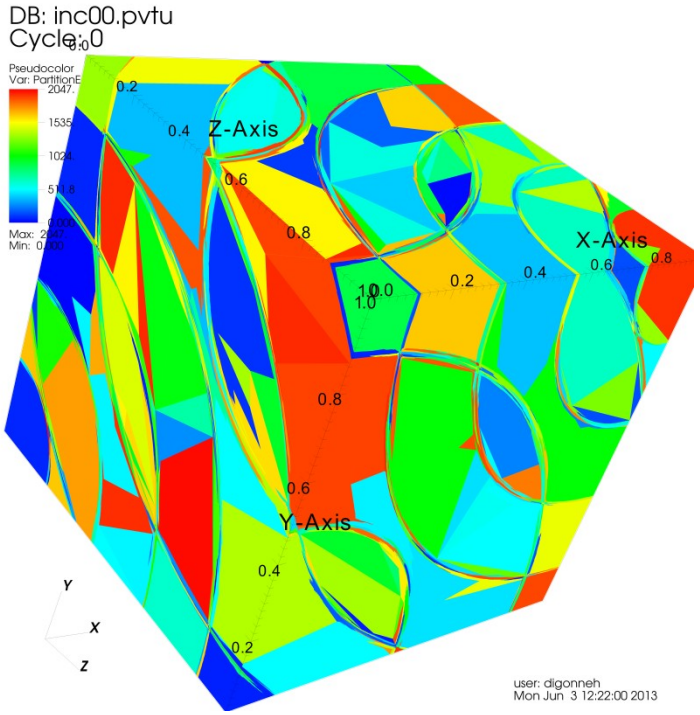
$$g(x) = \tanh\left(E \sin\left(\frac{4N+1}{2}\pi x\right)\right)$$

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

Application : adaptation statique (avec un estimateur d'erreur anisotrope)

Test 3d : la fonction avec N=2 E=16 et la partition du maillage



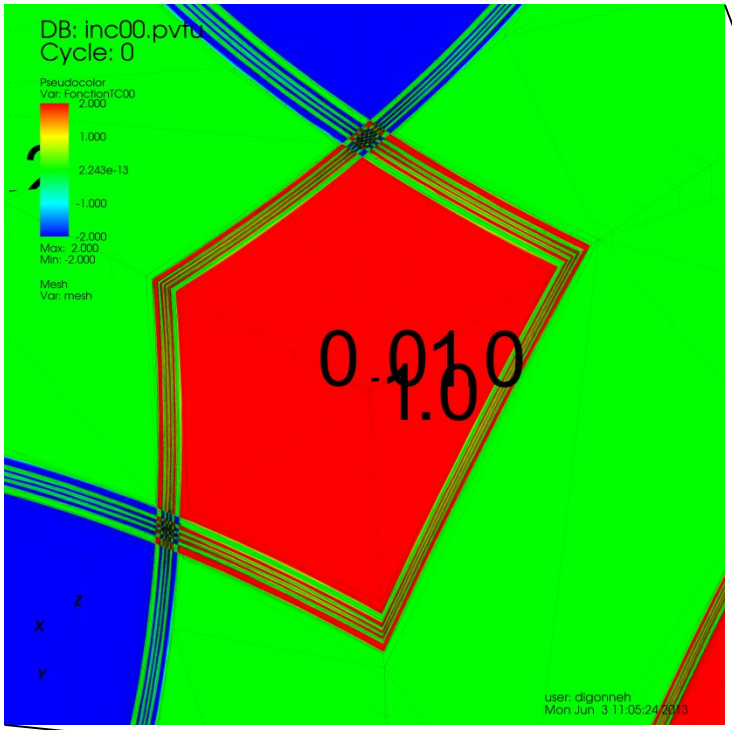


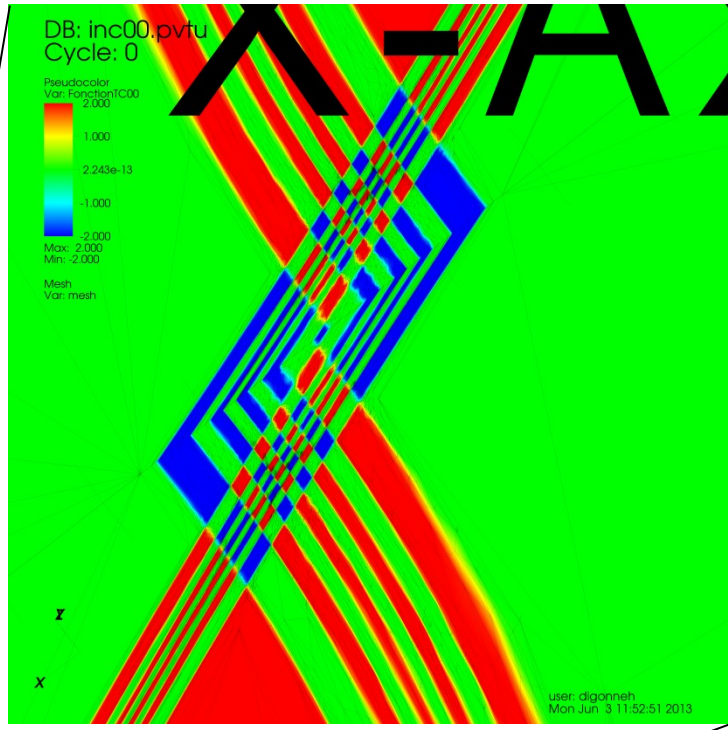
Maillage adapté sous la contrainte d'avoir 60 millions de nœuds

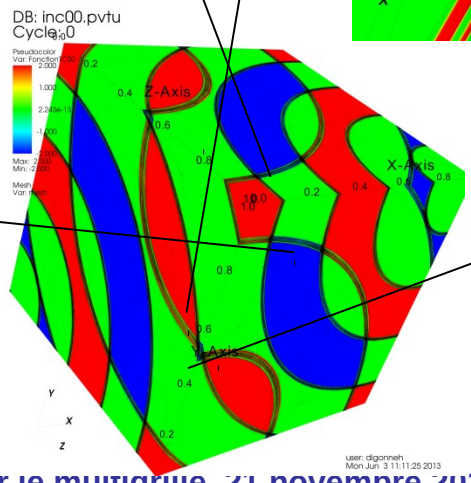
70 itérations effectuées sur 2048 cœurs de Curie in (10h)

Application : adaptation statique (avec un estimateur d'erreur anisotrope)

Test 3d : quelques détails



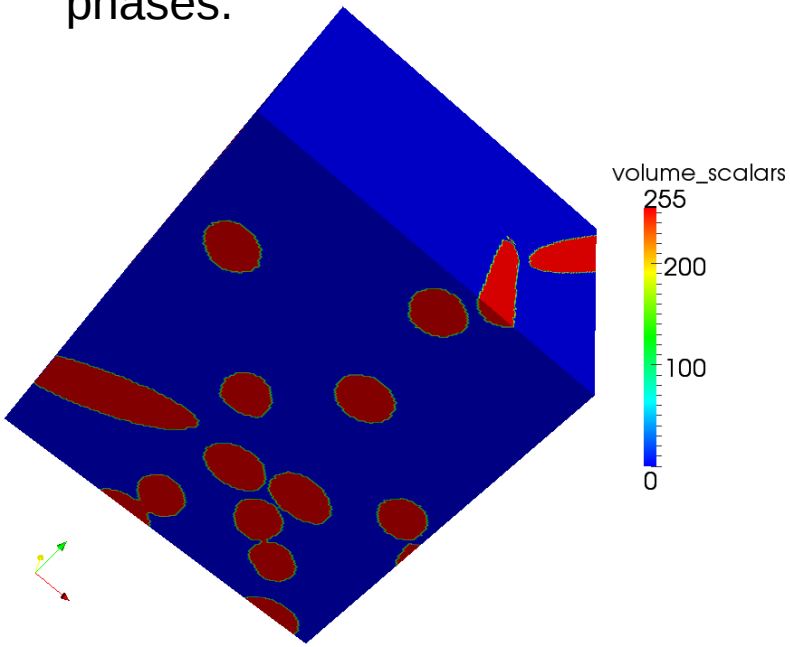




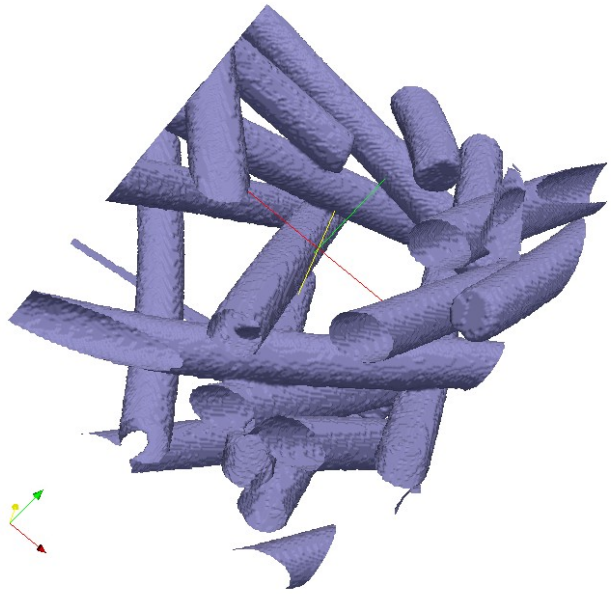
Du réel au virtuel :

Aujourd'hui les tomographies produisent des images 3d précises de microstructures (ici un exemple avec 200^3 voxels).

Cette image segmentée peut alors être utilisé pour déterminer les différentes phases.

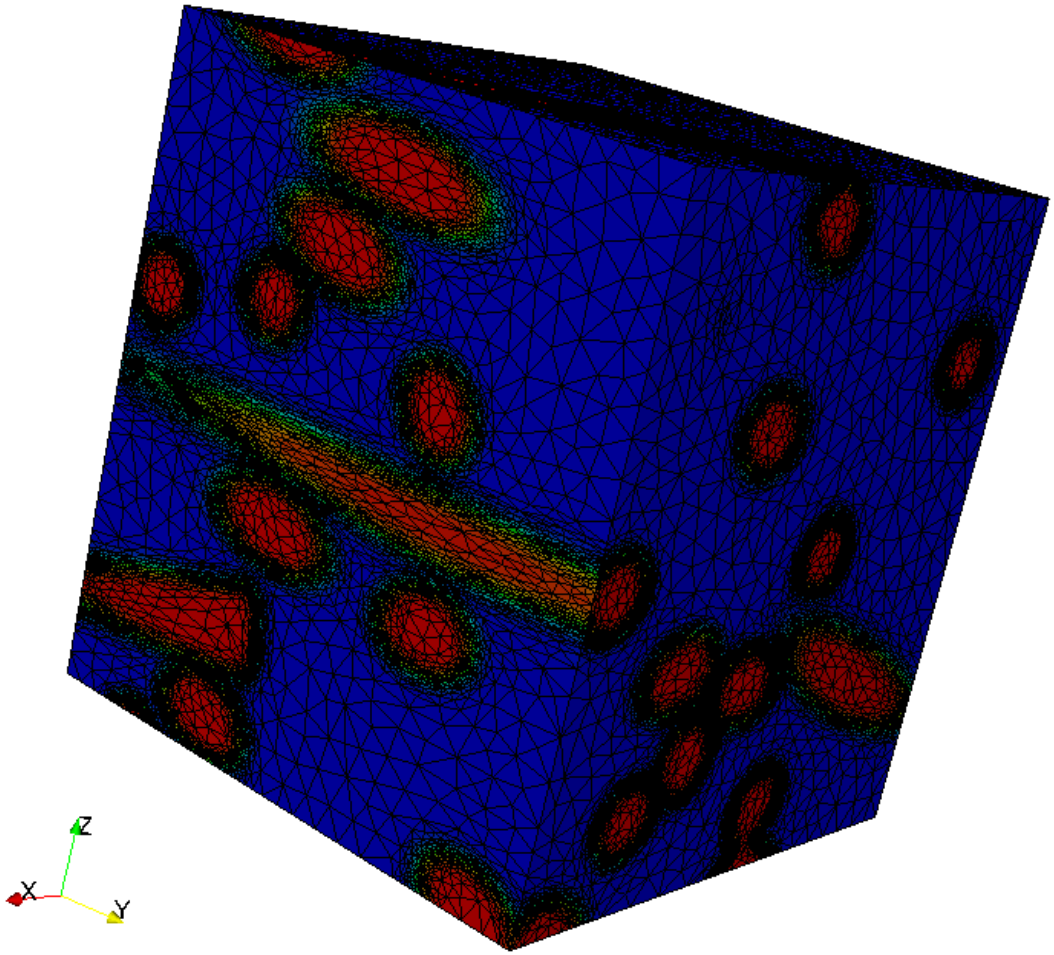


L'utilisation de cette image permet de définir une fonction LevelSet (distance) qui nous autorise à incorporer la microstructure dans le domaine de calcul.



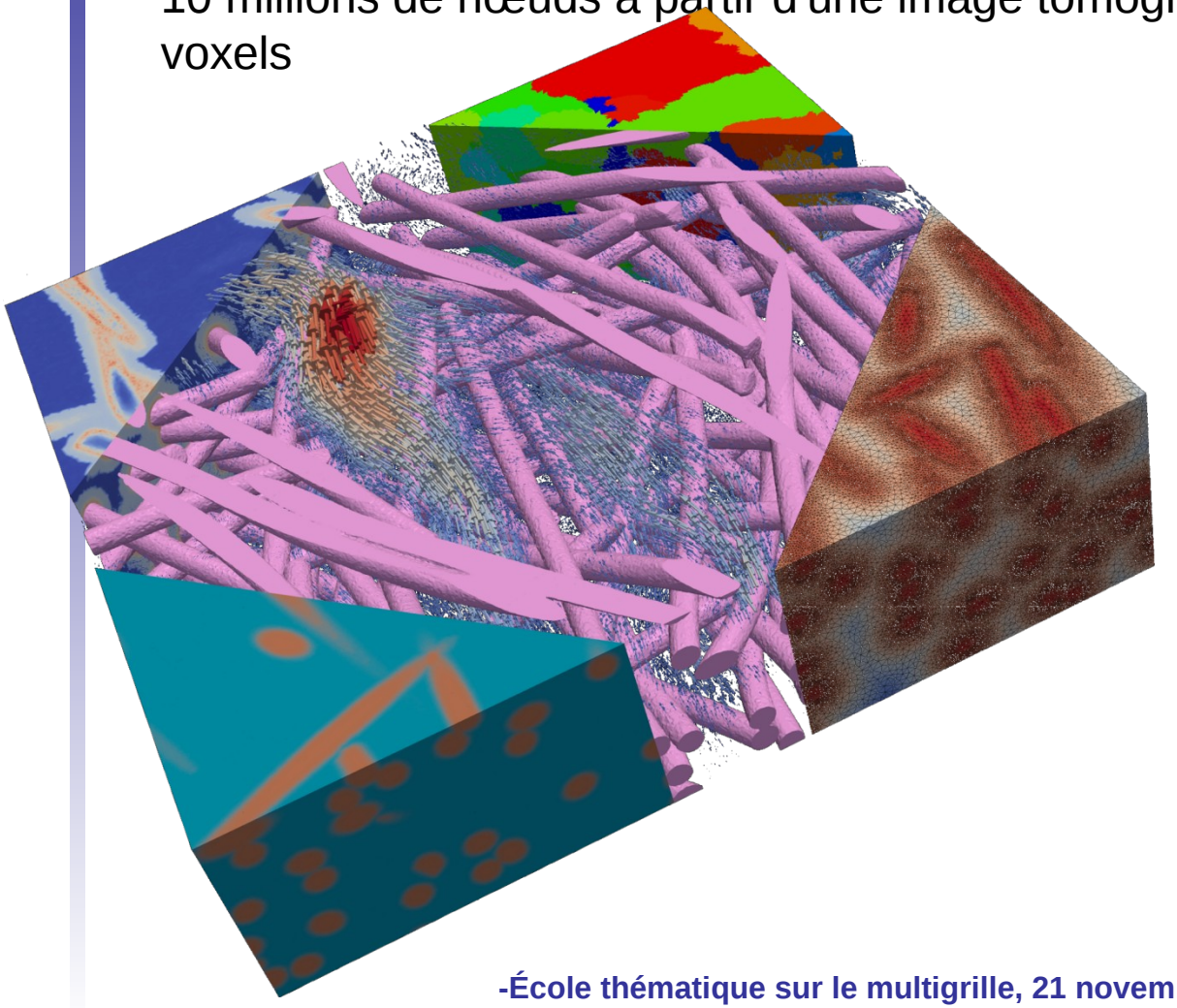
Du réel au virtuel :

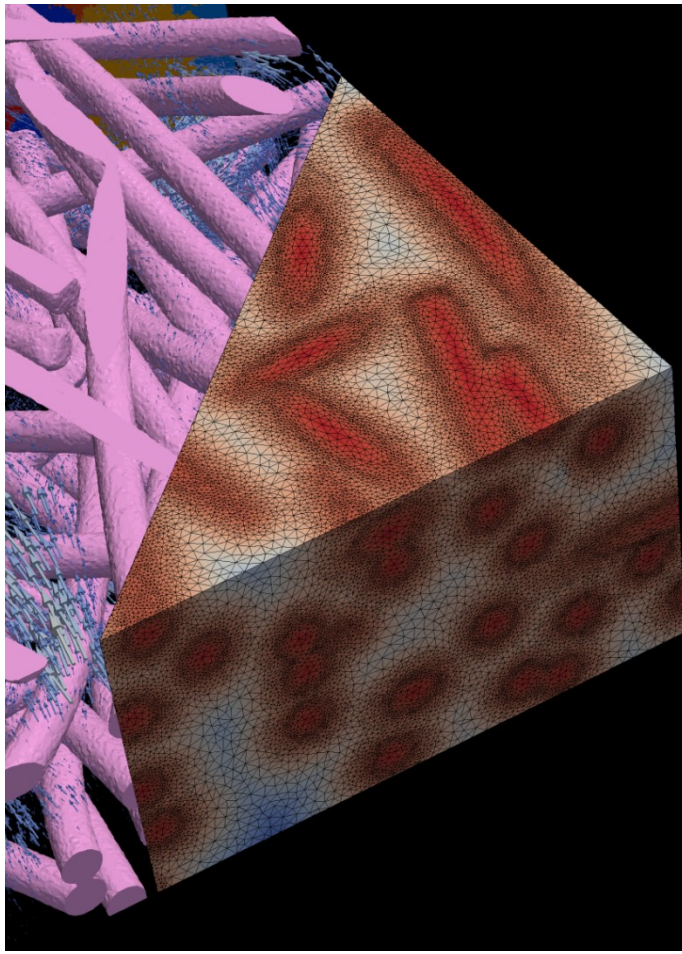
Le maillage de calcul est alors adapté de manière anisotrope et permet « à moindre coût » d'obtenir une bonne représentation de la microstructure.



Du réel au virtuel :

Exemple d'un calcul d'écoulement à travers une microstructure (pour déterminer une perméabilité). Exécuté sur 96 cœurs sur un maillage adapté comprenant 10 millions de nœuds à partir d'une image tomographique de 900x900x220-voxels





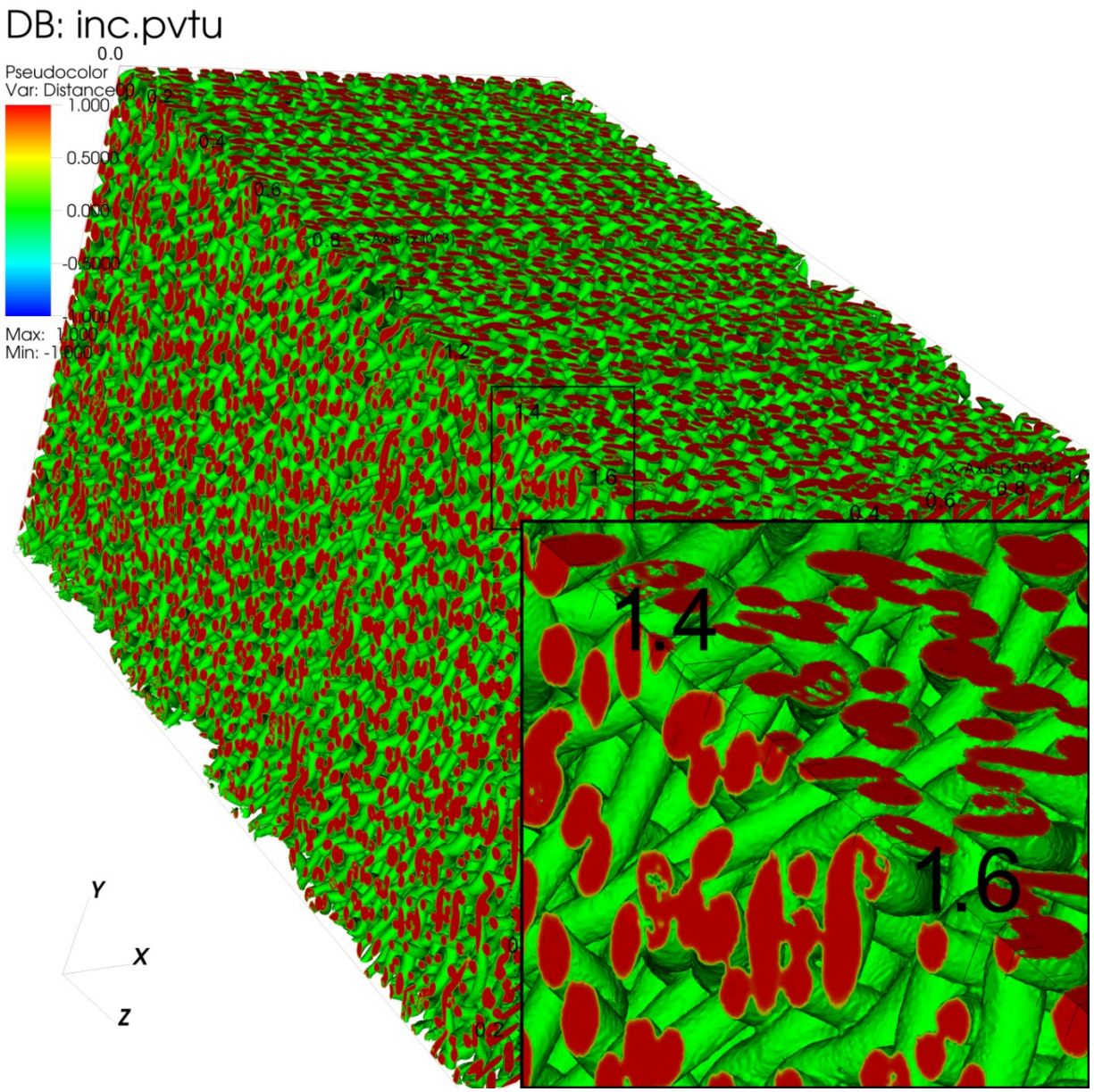
Du réel au virtuel :

Exemple :

Isosurface zéro de la LevelSet sur un maillage adapté de 400 millions de nœuds et 2.16 milliards éléments

Calculé à partir d'une image tomographique 3d de la microstructure de taille 1200x1200x1800 voxels.

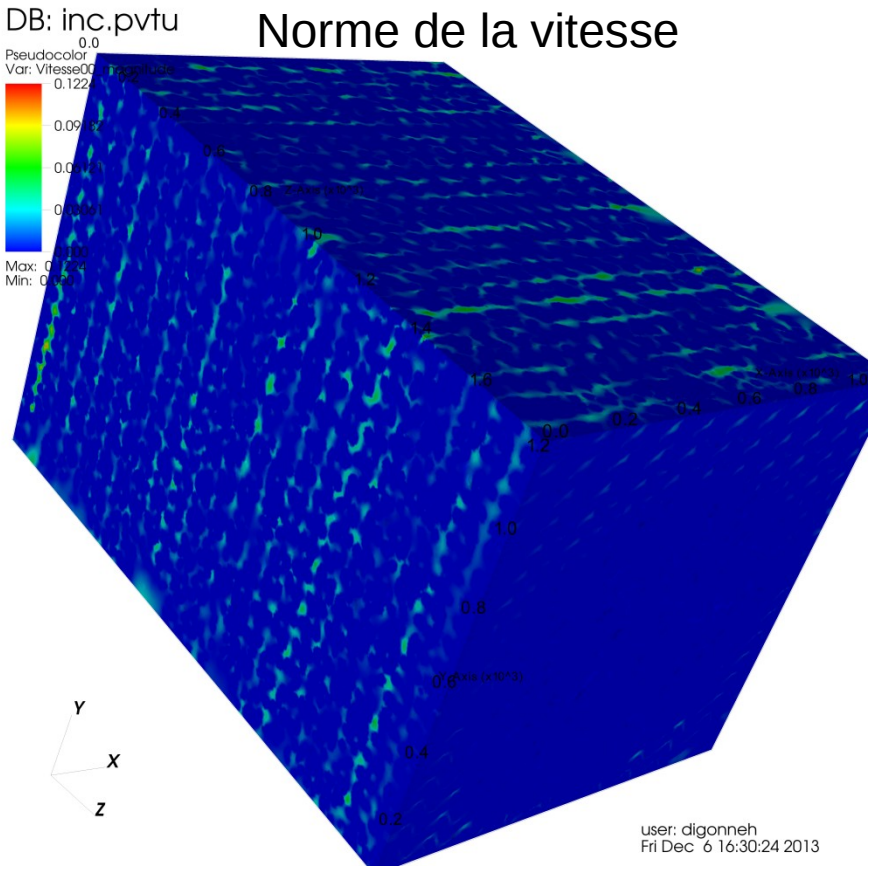
Exécuté sur 4096 cœurs en 5h pour une dizaines d'itérations



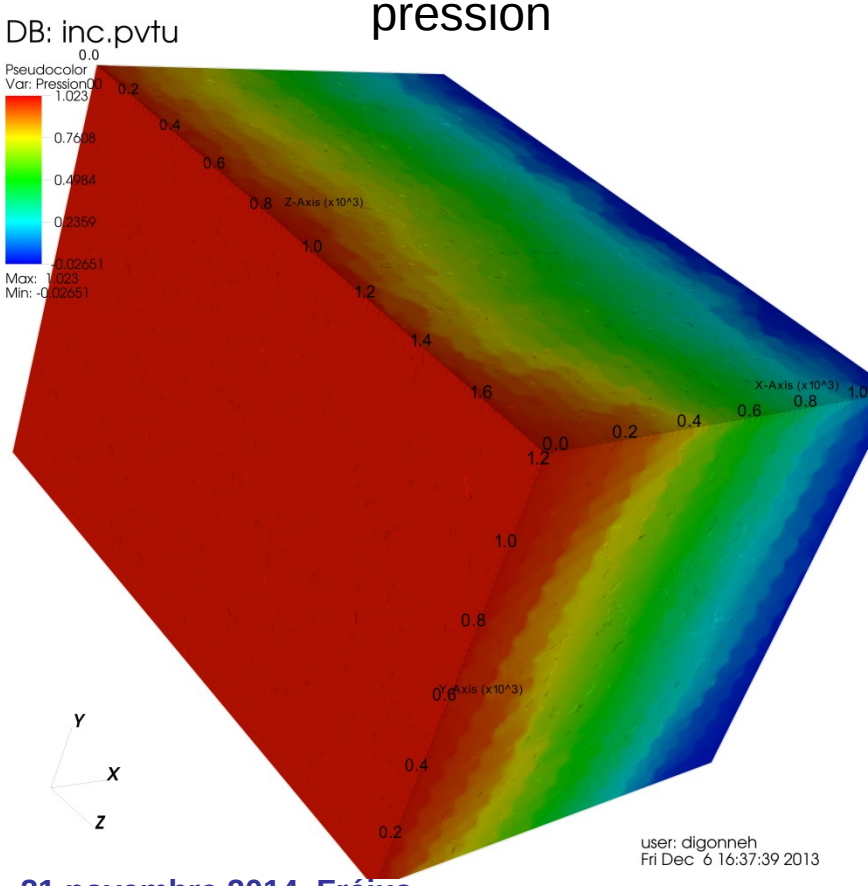
Du réel au virtuel :

Exemple : Calcul d'écoulement à travers cette microstructure. Équation de Stokes en formulation éléments finis mixte P1+/P1 aboutissant à la résolution d'un système à 1.5 millions de degrés de libertés effectuée en 1100 seconds.

Norme de la vitesse



pression



Conclusions

Nous somme capable d'adresser les supercalculateurs de type Tier0 avec une bonne efficacité parallèle et une extensibilité au-delà des 100 000 cœurs que ce soit :

- avec l'adaptation anisotrope de maillage
- mais également pour la résolution des système linéaire à l'aide d'une méthode multigrille

Une résolution « record » d'un système à **100 milliards de ddl** à été réalisée sur **65 536** (Curie) et **262 144** (JuQUEEN) cœurs en utilisant **200 To** de mémoire RAM.

Tout ceci nous permet de combiner les accélérations de chaque optimisation ($x10^{11}$):

- l'adaptation de maillage
  => réduit le nombre d'inconnues du problème (x1 000)
- le calcul massivement parallèle
  => accès à une puissance de calcul impressionnante (x 100 000)
- le solveur multigrille
  => réduit considérablement le nombre d'opérations nécessaire à la résolution de grands systèmes (x 1 000)

Visit ou Paraview nous permettent également de visualiser les résultats obtenues.

Perspectives

A court terme :

Améliorer les performances lors de l'utilisation de la quasi totalité des supercalculateurs (à 100 000 cœurs) par un meilleur placement des processus en fonction du hardware de ces derniers.

Valider les développements effectués dans d'autre contextes :

- solveur non linéaire
- problèmes instationnaires (exemple : évolution de surface libre)

Incorporer des données réelles de très grandes tailles : images tomographiques, bâtiments à l'échelle d'une ville, …

A plus long terme :

Étendre la méthode à des ordre d'éléments plus élevé P2, …

Construire une version en float128 bits pour autoriser des convergences au-delà des 10^{-14} du double précision (retrouver une erreur de convergence inférieur à L'erreur EF)

Remerciement

Je veux remercier les programmes

GENCI (Grand Équipement National de Calcul Intensif)

pour les accès aux supercalculateurs Tier1 français : Jade, Curie et Turing

PRACE (Partnership for Advanced Computing in Europe)

Pour les accès aux supercalculateurs Tier0 Curie et JuQUEEN.

Et merci pour votre attention