DE LA RECHERCHE À L'INDUSTRIE

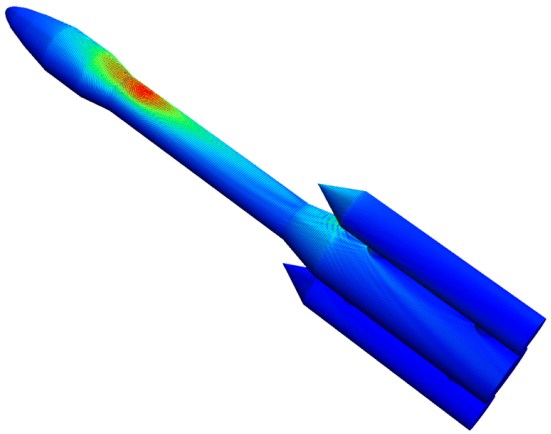
Équilibrage de charge pour solveur hiérarchique distribué

Directeur de thèse Raymond Namyst
Encadrants : Cédric Augonnet et Matthieu Kuhn
Équipe : CESTA/DSGA/SMMS/CHPC

| Paul BEZIAU
February 5, 2020

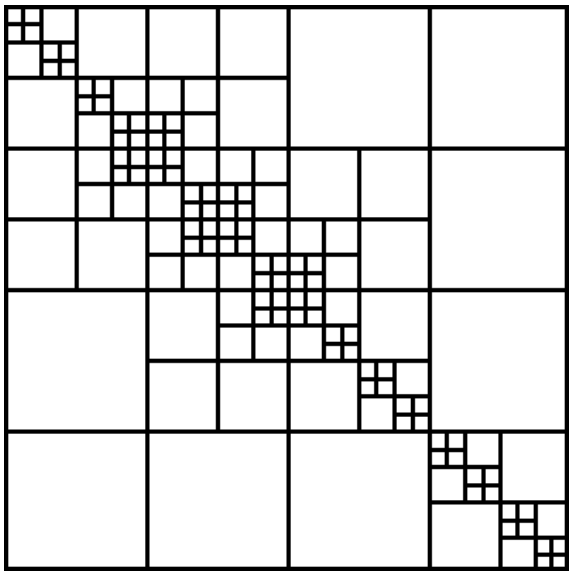
Introduction - Électromagnétisme

- On travaille sur des problèmes d'électromagnétisme
- Ceux-ci font intervenir des équations intégrales.
- Pour les résoudre on utilise un solveur dense direct.
- Ce solveur passe par une phase de factorisation de matrice
- Les matrices manipulées sont de très grandes tailles



Introduction - Compression

- Des méthodes de compression ont été mises en place
- D'abord du BLR puis des compressions hiérarchiques
- Le solveur continue d'utiliser des algorithmes tuilés



Introduction - Mémoire distribuée

- Les cas traités sont trop grands pour tenir sur un seul nœud de calcul
- Le solveur utilise donc plusieurs milliers de cœurs
- Les données sont distribuées selon un modèle 2D bloc cyclique
- Des optimisations importantes ont été développées ces dernières années
- Les problèmes d'équilibrage de charge deviennent limitant avec le passage à l'échelle

Simulation de l'exécution complète

Première solution : simulation au niveau des tuiles.

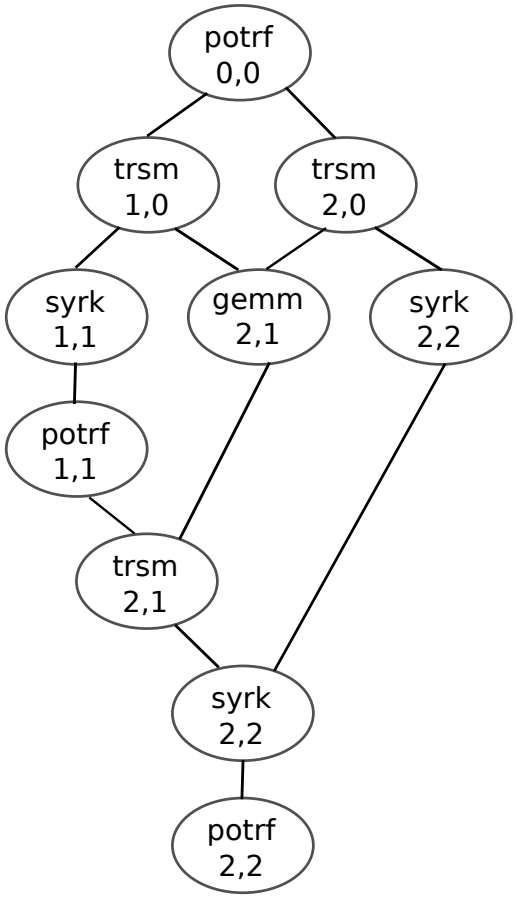
- On sauvegarde le temps de calcul nécessaire pour chaque tuile à chaque étape.
- Problème : comment exprimer le parallélisme au sein d'une tuile ?

Exécution réelle : 10.9s
Exécution simulée : 2.6s

Simulation de l'exécution complète

Deuxième solution : simulation au niveau des tâches

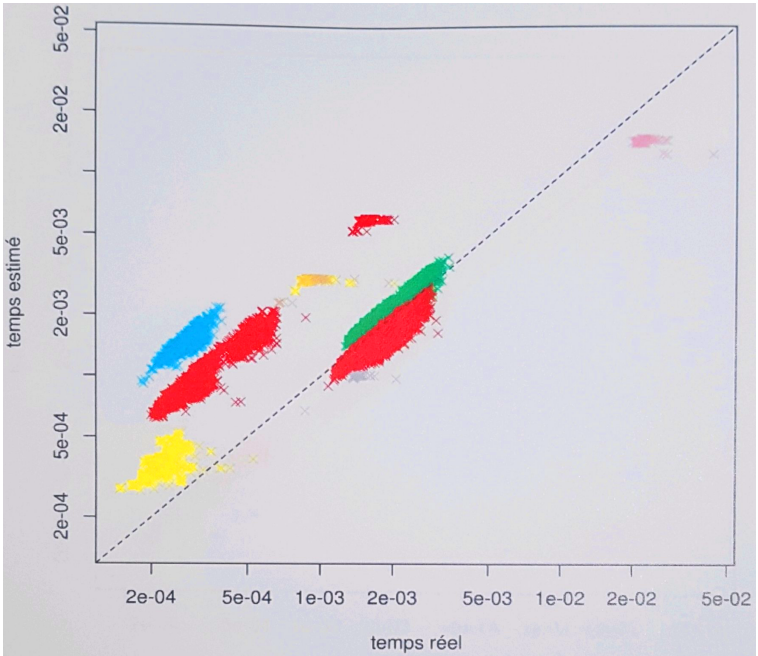
- On sauvegarde une trace de toutes les tâches
- On les rejoue ensuite en mode offline
- Mais c'est potentiellement plus coûteux

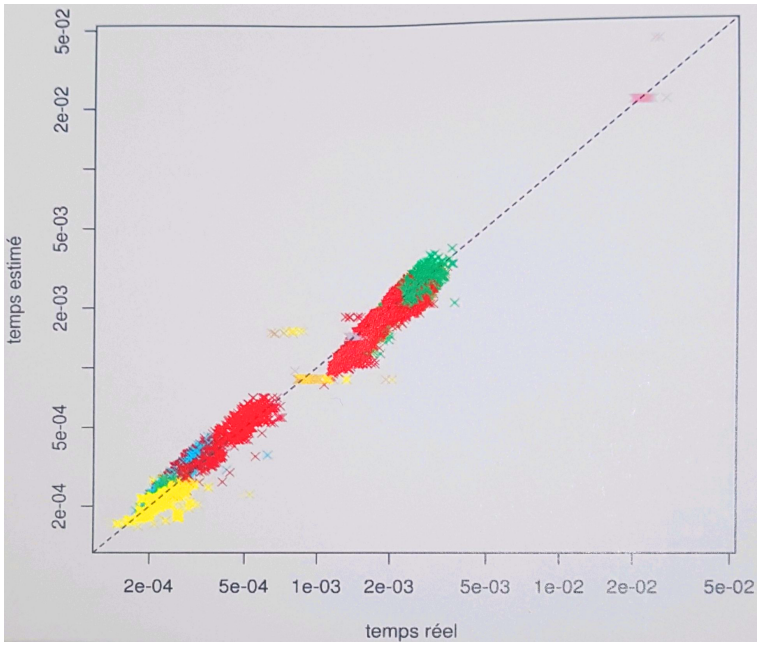


Exécution réelle : 10.9s
Exécution simulée : 13s

Modèle de performance

- Dans chaque tâche plusieurs noyaux sont appelés
- Le temps d'exécution ne suit pas une règle simple en fonction de la taille des données
- On crée alors un modèle pour chaque noyau
- En rejouant une tâche à vide on peut donc correctement prévoir son temps d'exécution

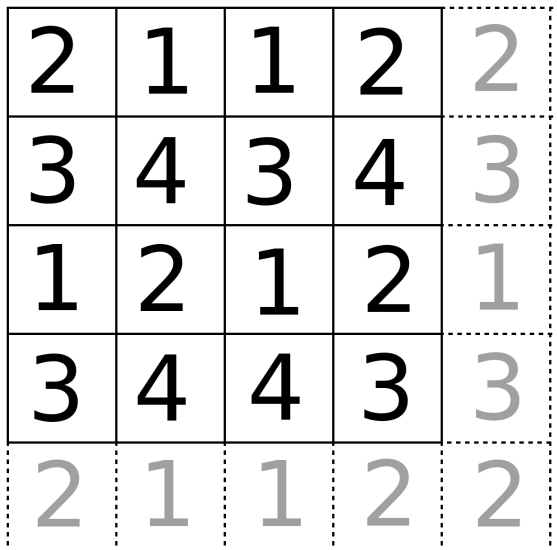




Processus virtuels

- On veut garder les bonnes propriétés de la répartition 2D bloc cyclique.
- Cette dernière est cependant trop rigide dans le cas des hmat

- On crée alors des processus virtuels, qui sont répartis en 2D cyclique
- Chacun de ces processus virtuel est ensuite assigné a un processus réel
- Cette méthode a été implémentée dans notre code.



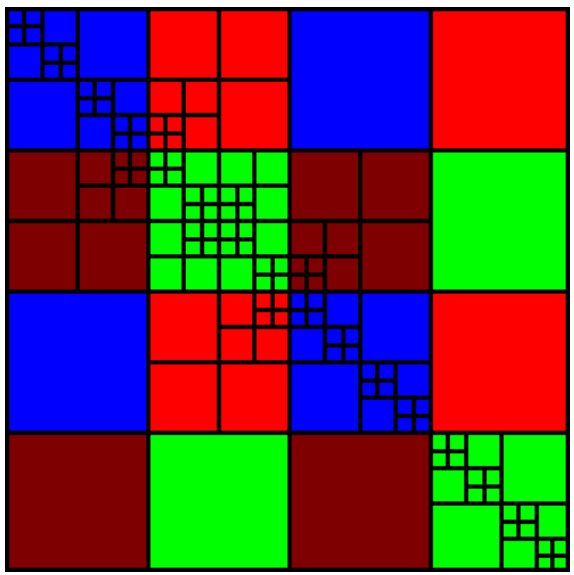
Mapping Hiérarchique

- Des tuiles plus grosses permettent une meilleure compression
- Des tuiles plus petites permettent une meilleure distribution
- Les matrices fortement compressées sont peu coûteuses

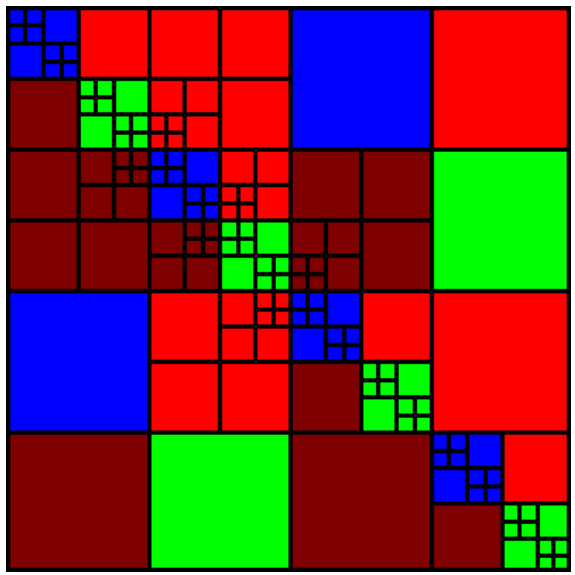
- Pour avoir les deux avantages il faut garder des grandes tuiles quand la compression est possible et des petites quand elle ne l'est pas
- Le problème existe toujours avec le modèle des processus virtuels
- Il faut donc un nouveau modèle de distribution.

Mapping hiérarchique

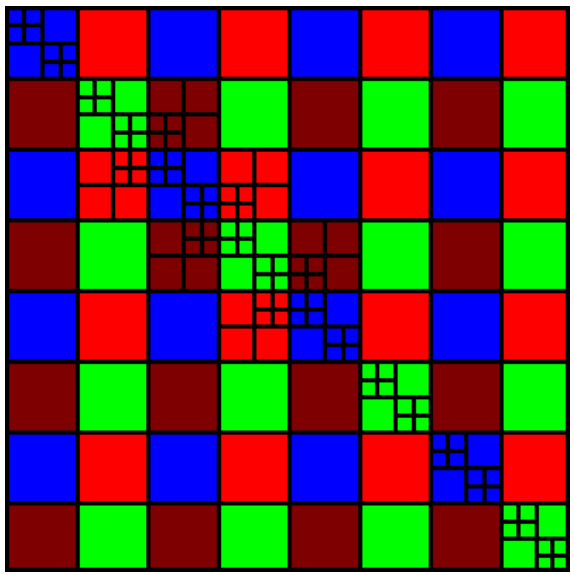
- On veut garder un modèle de mapping simple, qui passe à l'échelle
- Chaque niveau de hmat a son propre mapping



niveau 1



récursion



niveau 2

- owner(col, row) devient owner(col, row, level)

Conclusion

- On a vu comment nous pouvons prévoir le temps d'exécution d'une factorisation en H-mat.
- Cette prédiction permet de prédire et de corriger le déséquilibre de calcul.
- On a également vu qu'une amélioration du mapping 2D bloc cyclique permettra de mieux tirer parti de la compression tout en gardant une bonne répartition des calculs.

- Colaboration en cours avec l'Inria
- À venir : prédiction on-line