Adaptation de maillage en parallèle, application à la simulation de la mise en forme des matériaux

Hugues Digonnet, Marc Bernacki, Luisa Silva, Thierry Coupez

Adaptation de maillage en parallèle, application à la simulation de la mise en forme des matériaux

Hugues Digonnet, Marc Bernacki, Luisa Silva & Thierry Coupez

Ecole Nationale Supérieure des Mines de Paris
Centre de Mise en Forme des Matériaux,
1, rue Claude Daunesse, 06904 Sophia-Antipolis Cedex, France
Hugues.Digonnet@ensmp.fr

Résumé :

Dans ce papier, nous présentons une stratégie d'adaptation de maillage parallèle, qui consiste à paralléliser le mailleur, ainsi que le transport des données du maillage initial au maillage final. Cette stratégie d'adaptation consiste a remailler indépendamment chaque sous domaine sous la contrainte de ne pas toucher aux interfaces, puis de repartitionner en parallèle le domaine pour déplacer ces interfaces, ensuite il suffit d'itérer. Une méthode de transport hiérarchique parallèle est également présentée. Finalement, deux exemples d'utilisation de ces techniques sont montrés.

Abstract :

In this paper, we present a strategy to perform adaptative meshing in parallel. This parallelisation needs parallel remeshing but also parallel transport of the data from the initial mesh to the adapted one. This strategy use independent sub-domain remeshing under the constraint of blocking interfaces, then we repartition the mesh in parallel in order to move the interface. After this, the two phases are iterated. A parallel hierarchic transport algorithm over the mesh is also shown. Two examples using these methodologies are considered.

Mots-clefs : calcul parallèle ; adaptation de maillage ; éléments finis

1 Introduction

La simulation numérique a toujours été et reste encore aujourd'hui limitée par la puissance des calculateurs utilisés. Les calculateurs les plus puissants sont, depuis plusieurs années, des calculateurs parallèles. Ils peuvent, en effet, utiliser plusieurs unités de calcul simultanément, ce qui engendre une augmentation de leur puissance théorique. Il est à noter également que les derniers processeurs grand public doivent eux aussi être considérés comme des calculateurs parallèles. Ils sont, en effet, composés de plusieurs cœurs (core), chacun d'eux pouvant être assimilé à un microprocesseur. La parallélisation des codes de simulation est donc primordiale afin de pouvoir utiliser cette puissance de calcul d'une façon quasi optimale.

L'utilisation de clusters de PCs à plusieurs dizaines, centaines de processeurs représente une utilisation de calculateurs massivement parallèles et nécessite une parallélisation totale de la chaîne permettant de réaliser une simulation. Dans ce contexte, ce papier appréhende la parallélisation des outils de remaillage, que ce soit la construction du maillage ou son adaptation au cours du calcul. Cette tache est une des premières à effectuer avant de pouvoir lancer une simulation, le maillage généré doit, de plus, être suffisamment fin pour pouvoir capturer les phénomènes physiques mise en jeux. La génération d'un maillage fin peut vite devenir irréalisable sur une machine séquentielle. Il devient alors impératif de pouvoir construire ce maillage en parallèle. Cet aspect est d'autant plus important dans le cas où l'on utilise une

adaptation de maillage au cours de l'exécution. La simulation étant exécutée sur plusieurs processeurs, il devient naturel de considérer la phase de remaillage comme les autres et d'utiliser l'ensemble des processeurs réservés.

2 Stratégie de parallélisation

Nous nous intéressons à la parallélisation de l'adaptation de maillage, que ce soit pour la génération du domaine de calcul, ou pour les adaptations successives exécutées durant le calcul. Cette adaptation de maillage dynamique nécessite la parallélisation de la génération du maillage, mais également du transport des données du maillage initial sur le maillage adapté. En effet, il faut pouvoir reconstruire la configuration calculée sur l'ancien maillage sur le nouveau.

2.1 Parallélisation du mailleur

Dans cette partie, nous présentons uniquement la méthode de parallélisation de la phase de remaillage. On ne considère pas le transport des variables et on se place dans une phase initiale avant tout calcul. Le but est simplement d'obtenir un maillage suffisamment fin qui sera ensuite utilisé lors d'une simulation complète parallèle.

Nous considérons la parallélisation du mailleur MTC Coupez (2000) qui génère des maillages non structurés et non hiérarchiques. Ceci rend très difficile une parallélisation direct de ce dernier. Pour le paralléliser, nous avons préféré conserver le mailleur séquentiel, mais en l'utilisant de manière indépendante sur chaque sous-domaine associé à chaque processeur. Cette utilisation indépendante du mailleur permet son exécution en simultanée sur chaque processeur et donc sa parallélisation. Cependant, cette utilisation indépendante nécessite l'application d'une contrainte supplémentaire au mailleur pour conserver un maillage global cohérent (voir FIG. 1). La contrainte nécessaire et suffisante est d'avoir une discrétisation des interfaces identique entre les sous-domaines. D'un point de vue parallèle, cette contrainte peut demander une très forte interaction entre les différents processeurs ce qui est à l'opposé de la stratégie choisie. Nous avons donc choisi une contrainte plus forte qui consiste à ne pas toucher aux interfaces, de façon à garantir un travail indépendant sur chaque sous-domaine et une cohérence globale sur le maillage comme le montre la figure suivante (FIG. 1).

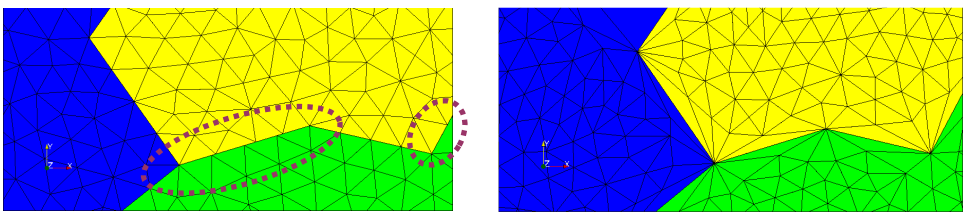

FIG. 1 – Zoom sur une partie du maillage remaillé indépendamment sur chaque processeur : à gauche sans contrainte, à droite avec la contrainte de ne pas toucher aux interfaces. Les parties entourées soulignent des défauts dans la topologie du maillage.

Cependant, l'ajout de cette contrainte abouti à un non respect des consignes données au mailleur au niveau de ces interfaces (FIG. 1 : la taille de maille des éléments proche de l'interface n'est pas celle demandée au mailleur, mais est identique à celle du maillage initial). Pour pouvoir remailler ces interfaces, la solution choisie consiste simplement à modifier la partition du maillage à l'aide d'un repartitionneur parallèle Basemann *et al.* (2000) en le contraignant cette fois ci à déplacer les interfaces vers le centre d'un des domaines. De cette manière, on autorise le mailleur à travailler à ces endroits lors d'une prochaine phase de remaillage.

La parallélisation complète du mailleur se fait donc aisément par la succession de phases de repartitionnement-remaillage qui ont pour but respectif : d'équilibrer la charge de travail et de déplacer les interfaces ; de remailler les parties pas encore effectuées. La figure suivante (FIG. 2) illustre parfaitement ce procédé dans un cas simple en 2d sur 6 processeurs, ou l'on souhaite raffiner le maillage initial en réduisant sa taille de maille.

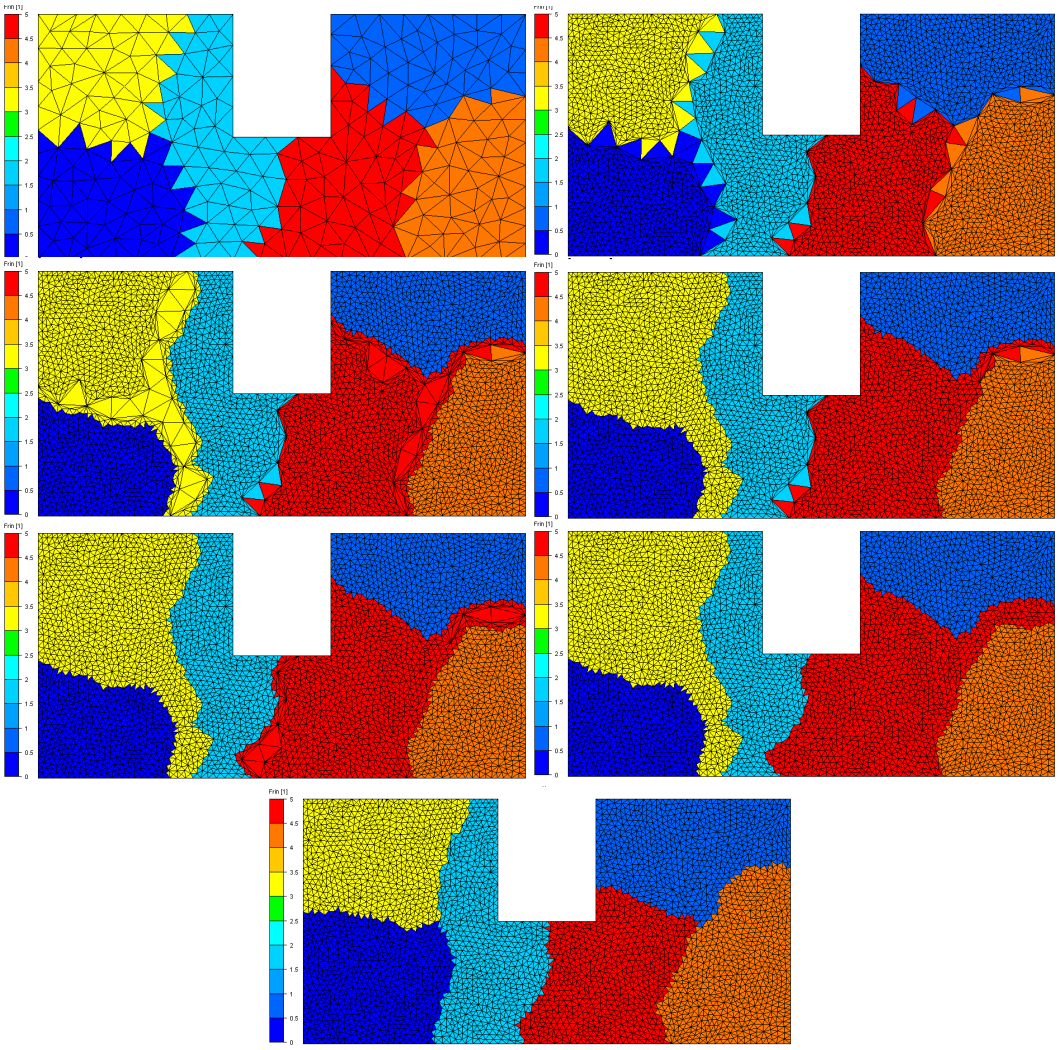

FIG. 2 – Illustration de la stratégie de parallélisation du mailleur sur 6 processeurs en 2d. Stratégie qui consiste à itérer des phases de repartitionnement - remaillage jusqu'à obtenir un « bon » maillage et une « bonne » partition.

Remarque : Il est important de noter que l'utilisation du mailleur dans un contexte parallèle, plutôt que sa parallélisation directe, améliore également l'encapsulation du code. Cette caractéristique des programmes orientés objets facilite l'évolutivité des développements faits. Dans le cas présent, il est clair que le développement d'un mailleur est une tache séquentielle qui de surcroît pressente une certaine évolutivité (prise en compte d'une adaptation anisotrope…). Cette bonne encapsulation permet à un développeur classique, sans connaissance particulière en programmation parallèle, de continuer à enrichir la version séquentielle du mailleur qui pourra immédiatement être utilisée dans un contexte parallèle. Ces notions de programmation orientée objet constituent un atout supplémentaire à la stratégie proposée.

2.2 Parallélisation du transport

Si on considère maintenant l'adaptation de maillage au cours d'une simulation, il faut considérer le transport des grandeurs calculées au cours de la simulation de l'ancien maillage sur le nouveau. Ce transport consiste à évaluer, pour chaque entité du nouveau maillage, la valeur de la donnée associée sur l'ancien maillage.

Lors d'un remaillage, cette donnée doit alors être représentée sur le nouveau maillage et il nous faut donc calculer sa valeur pour chaque entité du nouveau maillage en l'interpolant sur l'ancien maillage. Un exemple d'interpolation consiste, pour un champ P1 (linéaire par élément et continu), à trouver à quel élément de l'ancien maillage appartient un nouveau noeud, et ceci pour tous les noeuds du nouveau maillage. Cet algorithme initialement en $O(n^2)$ est réduit à une complexité en O(n log(n)) par l'utilisation d'une structure arborescente de boîtes permettant de localiser rapidement l'élément de l'ancien maillage qui contient le noeud. On teste récursivement l'appartenance du point aux boîtes de l'arbre hiérarchique de façon à réduire la quantité des éléments à tester comme le montre la figure suivante (FIG. 3).

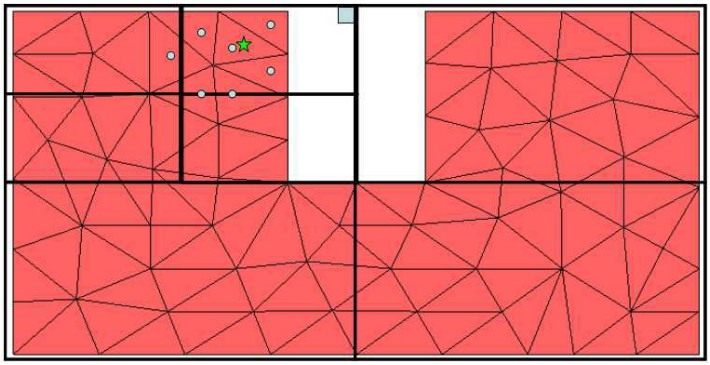

FIG. 3 – Localisation hiérarchique en O(n log(n)) d'un nouveau nœud (l'étoile verte) dans l'ancien maillage. Seul les éléments contenant le rond bleu sont testés au lieu de leur totalité.

La version parallèle de ce transport est réalisée simplement en ne considérant plus une unique boite globale initiale mais autant que de sous-domaines associés à chaque processeur. Pour chaque nœud du nouveau maillage, on cherche, dans un premier temps, s'il appartient à un des éléments de la partition locale du maillage. En cas de succès, le processus est terminé ; sinon, on utilise la première génération de boîtes qui est distribuée à l'ensemble des processeurs comme un filtre, afin de savoir sur quel processeur le nouveau nœud est susceptible d'appartenir dans l'ancienne partition du maillage. Dans ce cas, une requête est envoyée aux processeurs potentiellement clients sous la forme des coordonnées de ce nœud, ces processeurs exécutent alors la même méthode de recherche. En cas de succès, le processeur client renvoie la valeur du champ interpolé au processeur d'origine. Donc, chaque processeur traite l'ensemble de ses nœuds locaux, plus un certain nombre de nœuds extérieurs.

L'efficacité d'une telle parallélisation repose sur le fait que les deux partitions de maillage (l'ancienne et la nouvelle) ne sont pas trop éloignées l'une de l'autre : l'intersection entre deux sous-domaines hébergés par le même processeur doit être maximale. Cette condition n'est pas nécessaire pour obtenir un transport correct, mais réduit de façon importante le temps de calcul nécessaire (voir FIG. 4). Il est à noter que cette contrainte est en accord avec la stratégie de parallélisation du mailleur, qui procède par des optimisations locales du maillage et de la partition.

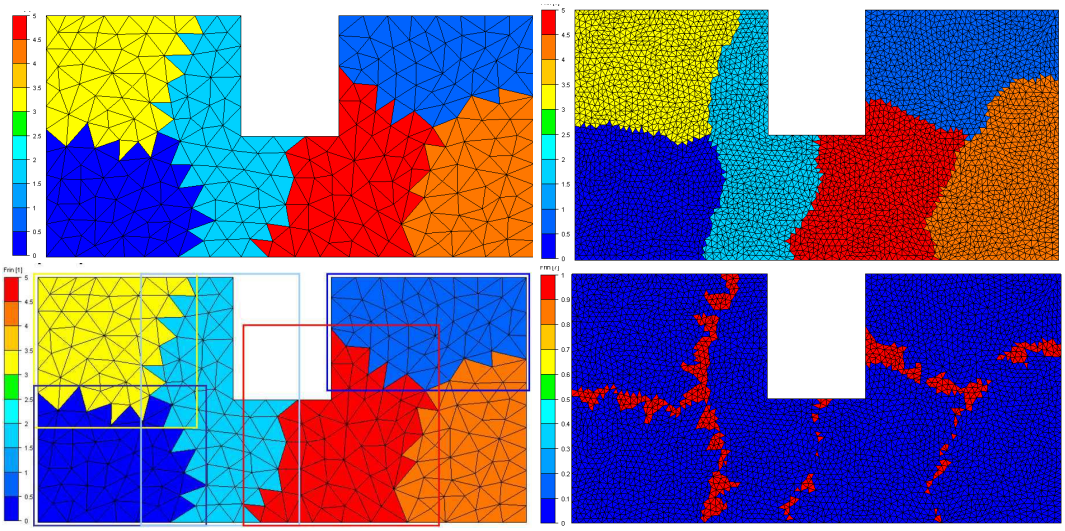

FIG. 4 – Illustration du transport parallèle : le maillage et la partition initiale (en haut, à gauche), le maillage raffiné et sa partition (en haut, à droite), les boîtes correspondantes au sous domaine de chaque processeur (en bas, à gauche) et en rouge les zones qui nécessitent des communications (en bas, à droite).

3 Applications et Performances

3.1 Matériau digital

Ces méthodes ont été appliquées dans la modélisation d'un matériau digital, Il s'agit de modéliser la déformation d'un volume élémentaire représentatif (VER) Bernacki *et al.* (2007) d'un alliage métallique composé de plusieurs grains. Ici, l'adaptation de maillage permet une discrétisation précise des grains, en raffinant de façon anisotrope autour des interfaces entre les grains. Au cours de la déformation, des remaillages sont nécessaires, d'où une utilisation fréquente des méthodes décrites précédemment. Le tableau ci-dessous (TAB. 1) présente les temps obtenus dans les différentes parties du code lors d'une simulation réalisée sur un maillage comportant environ 100 000 nœuds et 12 grains réalisée sur un cluster de 48 Opterons dual-core avec un réseau rapide Infiniband.

	Sorties	Assemblage	Résolution	Remaillage	Transport	Total
1 processeur	204	3773	19661	4614	486	28738
1 processeur //	201	3755	19026	8129	888	31999
2 processeurs	196	2013	11053	4070	506	17838
4 processeurs	204	1015	5660	1898	233	9010
8 processeurs	205	510	2249	850	106	3920
16 processeurs	205	256	1066	415	50	1992

TAB. 1 – Temps passé en secondes dans les différentes parties du code. Le cas 1 processeur // est réalisé en séquentiel, mais en utilisant le remailllage parallèle itératif.

On remarque une bonne accélération globale de 14,4 sur 16 processeurs en considérant l'écriture de fichiers résultats qui est exécutée en séquentiel (logiciel de visualisation séquentiel). Pour la phase de remaillage (remaillage + transport), l'accélération est inférieur à 11 sur 16 processeurs, mais reste importante compte tenu du processus itératif mis en place. Il reste

cependant des améliorations à apporter, principalement du point de vue informatique, en exploitant pleinement la diminution importante du nombre de changements effectués au cours des itérations : le temps passé dans la phase de remaillage devrait, à terme, être sensiblement le même entre 1 processeur et 1 processeur //).

3.2 Injection assistée eau

Nous présentons ici un exemple de calcul qui utilise la quasi-totalité des ressources de calcul du cluster existant au CEMEF. Ce cas a nécessité la construction d'un maillage de 25 millions de nœuds, maillage présentant une légère anisotropie dans le canal d'alimentation. La construction d'un tel maillage n'a été possible qu'en parallèle et la résolution du problème de Stokes sur ce maillage a duré 8230 secondes, en utilisant 88 processeurs pour un total de 100 millions d'inconnues. Le champ de vitesse déterminée a été utilisé pour convecter le transport de l'interface eau-polymère, par résolution d'une équation de transport et en utilisant la méthode Level-Set Zerguine *et al.* (2007). Pour cela, 1250 incréments ont été réalisés en 17 heures. Ce cas test est plus à considérer comme un « benchmark ». La simulation réalisée n'est en effet pas assez réaliste, mais considère les difficultés liées au calcul massivement parallèle où l'ensemble de la chaîne doit être parallélisé, de la génération du maillage à la visualisation des résultats.

6 Conclusions

Dans ce papier nous présentons une stratégie de parallélisation d'un mailleur originale par déplacement d'interfaces. Cette méthode consiste à coupler un mailleur séquentiel avec un partitionneur parallèle. L'efficacité de cette approche repose sur le caractère d'optimisation locale, utilisé à la fois par le mailleur MTC et par le repartitionneur. Un autre avantage très important est la très faible interaction entre le mailleur et sa parallélisation. En effet, le mailleur est simplement utilisé dans un contexte parallèle. Il peut donc être développé et maintenu par un développeur sans connaissance en parallélisme et ces améliorations sont immédiatement utilisables dans sa version parallèle.

La parallélisation d'un l'algorithme de transport utilisé après chaque remaillage est également présentée. Elle repose sur une localisation hiérarchique des éléments et l'envoi de requêtes d'interpolation d'un processeur vers des processeurs candidats.

Finalement, un très bon comportement parallèle (proche de l'optimal) de l'application est montré lors de simulations réalisées sur un cluster, avec de 1 à 16 processeurs, et ceci malgré les améliorations informatiques qui restent à faire. Un dernier exemple montre le type de simulations envisageables sur un cluster de 96 processeurs avec l'ensemble des outils parallélisés.

Références

Coupez, T. 2000 Génération et Adaptation de maillage par optimisation locale. *Revue européenne des éléments finis.* **9(4)**, 403-423

Basemann, A., Clinckemaillie, J., Coupez, T., Fingberg. J., Digonnet, H., Ducloux, R., Gratien, J.M., Lonsdale, G., Maerten, B. Roose, D. & Walshaw, C. 2000 Dynamic load balancing of finite element applications with drama library. *Appl Math Modeling.* **25(2)**, 83-98

Bernacki, M., Chastel, Y., Digonnet, H., Heba, R., Coupez, T., & Logé, R.E. 2007 Development of numerical tools for the multiscale modelling of recrystallisation in metals, based on a digital material framework. *Computer methods in materials science.* **7(1)**, 142-149

Zerguine, W., Silva, L., Coupez, T. Digonnet, H. & Rodriguez-Villa, A. 2007 Capture d'interface et application au procédé d'injection assistée eau. Congrès Français de Mécanique. Grenoble