École d'hiver iHPerf 2000: Applications Haute Performances

Analyse Conception et Utilisation des grappes homognes ou htrognes de calculateurs

Coordonnateurs

- J-L. Pazat, IRISA, Rennes
- S. Rajopadhye, IRISA, Rennes
- J. Roman, LaBRI, Bordeaux

Table des matières

Table des figures

Table des tableaux

Positionnement de lecole

Dans de nombreux domaines scientifiques, l'apport de la simulation numrique est aujourd'hui incontestable. Cette technique ncessite de hautes performances de calcul tout en manipulant de grandes masses de donnes; de plus, la simulation numrique prend une nouvelle dimension quand on considre ce que l'on appelle dornavant la simulation numrique distribue mettant en jeu des couplages de codes permettant de simuler globalement plusieurs phnomnes physiques. D'autre part, l'volution du paralllisme "conventionnel" conduit de plus en plus, pour des raisons de rapport prix/performances, vers un cadre o il faudra toujours "concevoir parallle" mais plutt "excuter en distribu". Les thmes de cette cole recouvrent intgralement cette problmatique et sont actuellement stratgiques; en particulier, ils font clairement partie des axes prioritaires du dpartement SPI du CNRS.

L'importance de ces enjeux trs actuels justifie pleinement la mise en place de cette cole o l'on traitera tous les domaines concernant le savoir-faire de la mise en œuvre du calcul hautes performances sur des plateformes d'excution paralllles et/ou distribues.

La formation offerte par cette cole sera coordonne et assure par des membres des groupes de recherche du GDR ARP (Architecture, Rseaux et systmes rpartis, Paralllisme) qui sont prsents ci-dessous; il s'agit donc d'une formation issue de la recherche et s'appuyant sur des comptences affirmes au niveau national et international. Elle fera aussi appel des industriels impliqus dans le dveloppement d'applications paralllles de grande taille utilisant aussi le concept de grappes de multiprocesseurs.

Centre sur la mise en œuvre d'applications ncessitant de hautes performances dans des contextes d'excution homognes et htrognes distribus, cette cole est organise conjointement par le Thme "iHPerf : initiative informatique pour les hautes performances" et par l'Action transversale thmatique "Grappes: architecture, systmes, outils et applications pour rseaux de stations de travail hautes performances" du GDR ARP.

Première partie

Titre Partie Journe

Chapitre 1

Algèbre Linéaire Creuse Hautes Performances : Application à la Mécanique des Structures.

D. Goudin, P. Hénon, F. Pellegrini, P. Ramet, J. Roman (Labri UMR 5800) et J-J. Pesqué (CEA/DAM/CESTA)

La résolution de grands systèmes linéaires creux est une brique de base essentielle pour le traitement de nombreux problèmes de simulation numérique en vraie grandeur. Pour de telles applications, par exemple basées sur une modélisation par éléments finis 3D, le recours au calcul parallèle hautes performances est incontournable. Dans ce cadre-là, on est souvent amené à utiliser des solveurs directs réputés pour leur robustesse numérique en particulier lorsqu'il y a des problèmes de conditionnement qui font que les solveurs itératifs avec préconditionnement se comportent de manière instable; c'est le cas de notre application cible de mécanique des structures 3D lorsque apparaissent des phénomènes de grands déplacements ou de plastification.

L'ensemble des étapes algorithmiques de la chaîne logicielle EMILIO [2] développée au LaBRI par l'équipe ALiENor est présenté à la figure 1.1 dans le cadre de son utilisation pour des problèmes de mécanique des structures. Elle comprend une partie séquentielle et une partie parallèle. Le prétraitement séquentiel des données regroupe les étapes suivantes :

- la renumérotation de la matrice, qui permet de minimiser le remplissage lors de la factorisation numérique et de maximiser l'indépendance dans les calculs ;
- la factorisation symbolique, qui permet de connaître à l'avance la structure

de la matrice factorisée. Afin d'économiser la place mémoire et d'accélérer le calcul de cette factorisation symbolique, celle-ci est s'effectuée par blocs;

- le placement des blocs de la matrice sur les processeurs de la machine parallèle en tenant compte de leurs dépendances mutuelles, et ce afin d'obtenir un parallélisme maximum lors de la factorisation numérique ;
- la distribution des éléments du maillage à partir de la distribution des blocs de la matrice. Cette distribution des éléments équilibre le calcul des matrices et/ou des seconds membres élémentaires ainsi que les mises à jour dans la matrice globale distribuée.

Le traitement parallèle des données, regroupe les étapes suivantes :

- l'assemblage parallèle de la matrice de rigidité globale et/ou du second membre qui calcule les coefficients du système à résoudre;
- la factorisation numérique parallèle LDL^t basée sur un algorithme supernodal par bloc où les contributions distantes sont localement agrégées;
- la résolution des systèmes triangulaires ainsi construits.

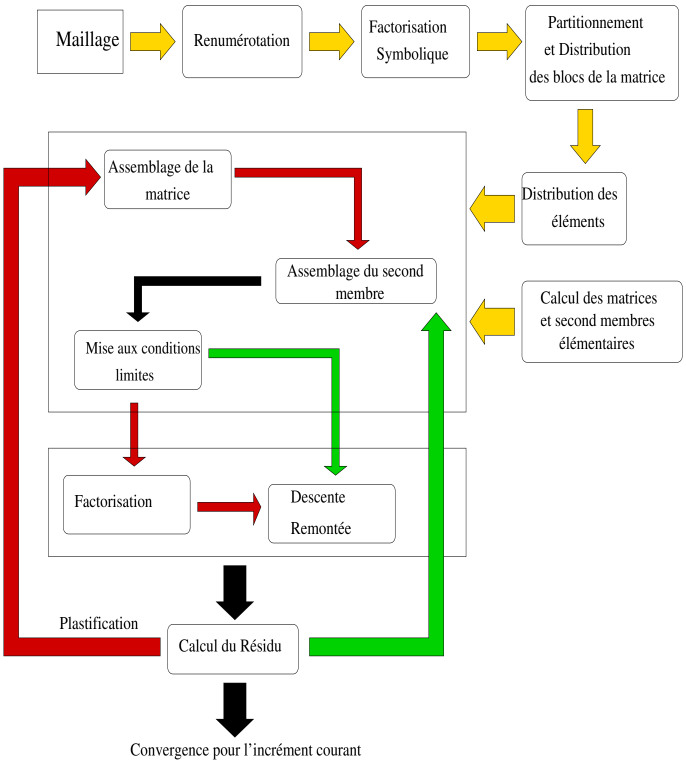

Figure 1.1 : L'enchaînement des différentes étapes

1.1 Renumérotation et Factorisation Symbolique

Les deux méthodes les plus utilisées à l'heure actuelle pour la renumérotation sont celles du degré minimum et des dissections emboîtées. L'idée de l'algorithme du degré minimum est de choisir à chaque étape le pivot correspondant au sommet de plus petit degré du graphe associé à la matrice. L'algorithme de dissections emboîtées [1] consiste à déterminer un ensemble de sommets S qui sépare le graphe en deux parties A et B non voisines, numéroter S avec les plus grands numéros disponibles, et recommencer récursivement sur les sous-graphes A et B jusqu'à ce que leurs tailles deviennent inférieures à une valeur limite. Cette numérotation garantit qu'aucun terme non nul ne peut apparaître lors de la factorisation entre des inconnues de A et des inconnues de B. Nous utilisons un “couplage fin” entre les deux méthodes : une fois que les premiers niveaux de récursion de la dissection emboîtée sont effectués, les sous-graphes restants sont alors numérotés en utilisant l'algorithme du “Halo Approximate Minimum Degree” [6].

En complément du calcul de la renumérotation des inconnues, cet algorithme construit une partition des colonnes en blocs colonnes. Cette partition est réalisée en fusionnant le partitionnement des séparateurs calculés par l'algorithme de dissections emboîtées et les super-variables amalgamées pour chaque sous-graphe renuméroté.

L'algorithme de factorisation symbolique opère sur la matrice renumérotée et partitionnée en blocs-colonnes par la phase de renumérotation, et produit en sortie une nouvelle matrice décomposée par blocs permettant de stocker l'ensemble des termes de remplissage qui seront produits au cours de la factorisation numérique.

1.2 Repartitionnement et distribution des blocs

Certains solveurs numériques prennent entièrement à leur charge le séquencement dynamique des calculs de factorisation des blocs sur l'architecture parallèle, en se basant sur le placement des blocs préalablement calculé par un algorithme de prétraitement par exemple de type *subtree-to-subcube*.

L'approche que nous avons choisie consiste à décharger entièrement le solveur de cette tâche, et donc à pré-calculer totalement l'ordonnancement des calculs et des communications de manière statique, en simulant le plus précisément possible le comportement de l'architecture parallèle utilisée [2, 3]. Notre algorithme comprend deux phases distinctes.

La première est la phase de repartitionnement, dans laquelle les plus gros blocs sont redécoupés afin de pouvoir faire participer plusieurs processeurs à leur traitement. On exhibe ainsi un parallélisme de type *calcul parallèle sur des blocs pleins*, qui s'applique principalement aux plus gros séparateurs de l'arbre d'élimination à la fin de la factorisation. Au cours de cette phase, on parcourt l'arbre d'élimination de haut en bas et on calcule pour chaque bloc un ensemble de processeurs candidats susceptibles de le traiter; cette méthode est une variante améliorée de la technique de "proportional mapping" introduite par Pothen et Sun [7]. Cette notion de processeur candidat est indispensable pour préserver la bonne localité des communications; celle-ci s'obtient en suivant la logique de l'arbre d'élimination, car ne seront candidats pour un nœud de l'arbre d'élimination que les processeurs ayant participé à la factorisation du sous-arbre dont il est la racine.

La deuxième phase est la phase de distribution proprement dite, dans laquelle on attribue à chaque bloc le processeur qui le calculera, parmi ses processeurs candidats. Pour ce faire, on remonte l'arbre d'élimination des feuilles vers la racine et, pour distribuer chaque nœud de l'arbre sur les processeurs, on calcule le temps (estimé) pris par sa factorisation et l'envoi de ses contributions. Ainsi, le choix de placement d'un bloc sur l'un de ses processeurs candidats est décidé en connaissant la date à laquelle potentiellement le processeur aura terminé de le traiter. Le processeur élu sera alors celui qui pourra avoir fini de traiter le bloc le plus tôt. Cette étape de simulation permet un ordonnancement fin des calculs et des communications; elle nécessite le calibrage des opérations BLAS3 et des opérations de communication MPI pour l'architecture cible.

Comme il a été constaté que des distributions hybrides (1D en bas de l'arbre d'élimination et 2D en haut) [3] donnent de meilleurs résultats, l'algorithme de repartitionnement a été modifié pour que le choix du type de distribution soit réalisé lors de la descente récursive de l'arbre d'élimination par blocs. Le critère de changement du type de distribution est actuellement un critère de niveau (ou de seuil) dans l'arbre d'élimination. Des travaux sont actuellement en cours pour améliorer ce choix, et en particulier pour exprimer ce critère en fonction de la taille des blocs et du nombre de processeurs candidats.

1.3 Distribution des éléments

La distribution des éléments est directement induite par la distribution des blocs issue de l'étape précédente. Cette distribution permet de connaître à l'avance les différentes communications à effectuer dans le module d'assemblage.

Le traitement effectué se décompose en deux parties : on commence par attribuer un nombre précalculé d'éléments totalement locaux aux processeurs selon un critère d'équilibrage de charge en termes de nombre d'éléments (par élément local, on entend un élément dont tous les nœuds appartiennent à des blocs locaux à un processeur au sens de la distribution des blocs). Puis on effectue la répartition équilibrée des éléments restants (non totalement locaux) en suivant la logique du partionnement induit par la renumérotation de nœuds.

Les communications ne sont pas prises en compte pour l'équilibrage, car elles sont supposées *recouvertes* par les calculs. Par contre on minimise la distance entre les processeurs possédant des nœuds d'un élément et celui sur lequel il sera placé.

1.4 Assemblage de la matrice

Cette phase utilise les informations qui lui sont données par la distribution des éléments. On commence par traiter les éléments qui ont été attribués au processeur courant mais dont un ou plusieurs nœuds sont dans des blocs appartenant à un autre processeur (éléments non totalement locaux). On prend en compte, en priorité, les éventuels messages de contributions distantes. On calcule la matrice élémentaire d'un élément non local, on prépare les messages à envoyer aux processeurs concernés, et on effectue les additions de contributions locales. Pour réduire le nombre de communications, les messages sont vectorisés au niveau d'un élément.

Les éléments totalement locaux sont traités en dernier pour que leurs calculs recouvrent les communications induites par les éléments non locaux. On calcule leur matrice élémentaire, puis on additionne les contributions (locales) dans la partie de matrice correspondante.

1.5 Le solveur parallèle

Le solveur actuellement utilisé dans EMILIO est basé sur une distribution 1D des blocs de la matrice. L'intégration dans la chaîne EMILIO du solveur basé sur une distribution mixte 1D/2D est en cours. Les calculs par blocs, considérés comme des *tâches*, sont totalement ordonnés lors de l'étape de distribution. Les émissions des blocs de contributions agrégées sont également ordonnées de façon compatible.

Nous présenterons lors de l'exposé les résultats obtenus pour un grand nombre de matrices, issues de problèmes industriels, certains dépassant le million d'inconnues pour des problèmes 3D, ainsi que des comparaisons avec le couple de référence MeTiS/PSPASES [4, 5].

1.6 Conclusion

Dans cette présentation, nous avons décrit l'intégralité des étapes algorithmiques pour la mise en œuvre de solveurs directs hautes performances par blocs.

Récemment nous avons intégré dans notre phase de régulation statique la possibilité de réduire le surcoût mémoire lié aux agrégations locales de contributions par l'anticipation l'envoi d'agrégats partiels.

De plus, nous avons étendu notre stratégie aux architectures hétérogènes et plus particulièrement celles basées sur des réseaux de nœuds SMP comme le IBM SP3. Pour cela nous avons étendu notre modèle d'évaluation des coûts de communications pour prendre en compte différents type de communications (via la mémoire partagée ou via le réseau). Des résultats numériques récents seront présentés.

Bibliographie

[1] A. George and J. W.-H. Liu. *Computer Solution of Large Sparse Positive Definite Systems.* Prentice-Hall, Englewood Cliffs, NJ, 1981.

[2] D. Goudin, P. Hénon, F. Pellegrini, P. Ramet, J. Roman, and J.-J. Pesqué. Parallel sparse linear algebra and application to structural mechanics. *accepté dans Numerical Algorithms, Baltzer Science Publisher*, 2000, à paraître.

[3] P. Hénon, P. Ramet, and J. Roman. PaStiX: A Parallel Sparse Direct Solver Based on a Static Scheduling for Mixed 1D/2D Block Distributions. In *Proccedings of Irregular'2000*, number 1800 in Lecture Notes in Computer Science, pages 519–525. Springer Verlag, May 2000.

[4] M. Joshi, G. Karypis, V. Kumar, A. Gupta, and Gustavson F. PSPASES : Scalable Parallel Direct Solver Library for Sparse Symmetric Positive Definite Linear Systems. Technical report, University of Minnesota and IBM Thomas J. Watson Research Center, May 1999.

[5] G. Karypis and V. Kumar. METIS – *A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices – Version 4.0.* University of Minnesota, September 1998.

[6] F. Pellegrini, J. Roman, and P. Amestoy. Hybridizing nested dissection and halo approximate minimum degree for efficient sparse matrix ordering. In *Proceedings of Irregular'99*, number 1586 in Lecture Notes in Computer Science, pages 986–995. Springer Verlag, April 1999. Extended paper appeared in Concurrency: Practice and Experience, 12:69-84, 2000.

[7] A. Pothen and C. Sun. A mapping algorithm for parallel sparse Cholesky factorization. *SIAM J. Sci. Comput.*, 14(5):1253–1257, September 1993.

Un autre *index* et *sous-index* ou *sous-index* ou encore *SOUS-index*.

Annexe

Index