Outils et méthodes du calcul parallèle pour matrices creuses

Serge G. PETITON

Projet MAP
Méthodologie et Algorithmique Parallèles pour le calcul scientifique

Laboratoire d'Informatique Fondamentale de Lille (LIFL),
URA CNRS 369,
Université des Sciences et Technologies de Lille (Lille I)

et

Laboratoire d'Applications Scientifiques pour le Calcul Intensif (ASCI),
UPR CNRS 9029.

`petiton@lifl.fr` ou `petiton@asci.fr`

`http:\\www.lifl.fr\~petiton`
`http:\\www.asci.fr`

ORganisation Associative du Parallélisme, **Clamart le 29 Janvier 1998**

Résumé

Dans une première partie nous présentons un outil permettant de *paralléliser* des indirections dans des boucles. Cet outil est basé sur la séparation de ces boucles en deux phases : l'une d'*inspection* des communications et/ou vagues de calcul engendrées, l'autre d'exécution des calculs des nids de boucle. Cette *compilation à l'exécution* permet d'obtenir de bons résultats, en particulier lors du calcul parallèle d'un produit d'une matrice creuse par un vecteur. Nous présentons les principales caractéristiques de cet outil, PARTI devenu CHAOS depuis ; repris dans plusieurs compilateurs.

Dans la seconde partie nous abordons le problème des bibliothèques d'algèbre linéaire parallèles pour matrices creuses. Les méthodes utilisées sont pour l'instant principalement itératives. Après un court historique des bibliothèques d'algèbre linéaire pour matrices creuses et la mise en évidence de leurs intérêts pour le calcul numérique, nous décrivons les bibliothèques PETSc et P-SPARLIB.

Nous mettons ensuite en évidence quelques problèmes de paramétrisation des méthodes sur lesquelles repose la quasi-totalité des noyaux de calcul de ces bibliothèques. En particulier, la difficulté de choisir la taille du sous-espace pour la méthode d'Arnodli est discutée dans le cas de l'utilisation d'une machine parallèle. Ce choix ne peut être "optimisé" qu'à posteriori et montre bien le besoin d'expertise nécessaire pour pouvoir utiliser ces bibliothèques.

Nous concluons en listant quelques points qui, nous pensons, doivent être développés.

Principales pages Web sur le sujet:

- PARTI et CHAOS : `www.cs.umd.edu/projects/hpsl.html`

- SPARKIT, P-SPARLIB :`www-users.cs.umn.edu/~saad/software.htm`

- PETSc : `http://www.mcs.anl.gov/petsc/petsc.html`

Outils et méthodes du calcul parallèle pour matrices creuses

PLAN

I - Introduction

Des bibliothèques d'algèbre linéaire pour matrices creuses existent, parallèles ou non.

Il existe aussi des bibliothèques parallèles plus spécialisées : mailleurs, PDE, CFD,... mais en bien plus petit nombre et moins portables en général.

Le problème majeur est, du point de vue de la manipulation de la matrice, le produit Ax, ou A est très grande et creuse - dans le cas des méthodes itératives, calcul du type $A(Ax + f(x)) + g(x)$.

Les méthodes directes remplissent les matrices mais possèdent d'importantes propriétés numériques.

Des méthodes pour *creuser* de très grandes matrices denses sont proposées pour certains types d'applications ; ce qui permet de réduire la taille mémoire disponible au prix d'une *complication* a priori des calculs.

Les problèmes d'indirections liés aux calculs creux sont donc très stratégiques en calcul scientifique.

Quels techniques et outils pour ces calculs?

Quelles bibliothèques et méthodes?

II - PARTY, PARTI et CHAOS

1987 : première décomposition d'une boucle parallèle entre un **inspector** et un **executor**. Dans :

An experimental study of methods for parallel preconditioned Krylov methods, D. BAXTER, J. SALTZ, M. SCHULTZ, S. EISENSTAT, and CROWLEY. In Proceedings of the 1988 Hypercube Multiprocessor Conference, Pasadena, CA.

1989 : Parallel Automated Runtime Toolkit at Yale (PARTY)

Début à Yale du projet PARTY (J. Saltz).

Le premier communication compiler(CC) pour machine parallèle (la CM-2) est proposé par Denny Dahl.

1990 : Parallel Automated Runtime Toolkit at ICASE (PARTI)

Première utilisation du CC pour faire du calcul "creux" permettant d'obtenir enfin des performances acceptables là où la communauté pensait la chose irréaliste.

1993 : CHAOS (Saltz et al.)

1994 : MetaCHAOS (G. Edjlali et al.)

Idée de base de PARTI :

Beaucoup de nids de boucle sont exécutés plusieurs fois (cas d'un produit matrice vecteur dans une méthode itérative).

La boucle sera décomposée en deux.

Compilation à l'exécution : les boucles sont décomposées en une phase d'inspection des communications engendrées **(inspector)** et d'exécution des calculs **(executor)**

Inspecteur : pré-calcul définissant les vagues d'indices exécutables en parallèle (mémoire partagée) ou le graphe de communication à employer pour rechercher les données non locales nécessaires au calcul (mémoire distribuée).

PARTI fournit les fonctions nécessaires à l'implémentation de cette méthode pour des structures irrégulières utilisant des tableaux d'indirection entiers.

PARTI impose des synchronisations des communications. Des phases de calcul puis des phases de communication.

Exemple

```
do i = 1, N
   y(i) = a(i) * y(ia(i)) + b(i) * y(ib(i))
enddo
```

Inspecteur (mémoire partagée) :

Analyse les dépendances entre itérations dans le nid de boucle à l'exécution.

Construit des *vagues* d'indices de le boucle pouvant être exécutées en parallèle.

Génère **schedule()**, **nbvague**, et le tableau **count**.

Exécuteur

```
ynew(1:n) = y(1:n)
do phase = 1, nbvague
   do parallel i = count(phase),count(phase+1) - 1
      ii = schedule(i)
      if (ia(ii) < ii)
         then tmp1 = ynew(ia(i))  % y(ia(ii) deja modifie.
         else tmps = y(ia(i))
      endif
      if (ib(ii) < ii)
         then tmp2 = ynew(ib(i))
         else tmp2(ib(i))
      endif
      ynew(ii) = a(ii) * tmp1 + b(ii) * tmp2
   enddo parallel
enddo
y(1:n) = ynew(1:n)
```

Le surcoût engendré par l'inspecteur doit être acceptable comparé au gain obtenu lors de l'exécution.

Le résultat du travail de l'inspecteur peut être réutilisé plusieurs fois ; jusqu'à ce que les tableaux d'indirections soient modifiés dans le programme.

Dans le cas distribué, pour le calcul d'une boucle parallèle chaque processeur doit :

- obtenir les données extérieures dont il a besoin

- ranger les copies en local. Dans des *hash tables* par exemple (nommé *hashed cache* dans PARTI).

L'inspecteur étudie les communications engendrées et optimise les `send/receive` (vectorisation des communications par exemple).

Inspecteur

Un processeur P :

- cherchera les éléments dont la boucle initiale aura besoin
- déterminera quels sont les éléments hors processeurs
- déterminera quelles sont les émissions/réceptions nécessaires pour l'échange de données
- générera la liste de toutes les données non locales à rechercher

les autres processeurs l'informent des données dont ils auront besoin

Exécuteur

Un processeur P :

- Avant la phase de calcul
  - les données à envoyer sont envoyées
  - les copies des données non locales sont reçues
  - les données sont écrites dans le hash cache
- Calcul de la boucle
  - les données hors processeurs sont lues et écrites dans le hash-cache
- Après la phase de calcul
  - les données à stocker hors processeur sont lues à partir du hash-cache
  - les données modifiées sont envoyées
  - les données à écrire sont réceptionnées
  - les données reçues sont écrites dans le tableau distribué.

CHAOS

Principalement PARTI

Primitives de placement de données.

Primitives de redistribution

Outils (concepts) utilisés par le compilateur Fortran D (Rice,...)

Utilisés aussi par Zima pour HPF+

...

III - Vers les *toolkits* pour le calcul scientifique parallèle

Bibliothèques pour matrices creuses

Yale Sparse Package, 1977 (séquentiel)

Bibliothèque pour matrices creuses.

Yale Sparse format (futur CSR) comme format de base.

Yale University, DCS, Eisenstat, Gursky, Schultz et Sherman.

SPARSe KIT **(SPARSKIT)** (séquentiel)

Bibliothèque pour matrices creuses.

format CSR comme format de base.

Outils de manipulation de matrices, de visualisation.

NASA puis Université de Minneapolis, Y. Saad.

La bibliothèque travaille uniquement avec le format CSR.
Il existe des primitives de conversion de CSR vers de nombreux autres formats (COO, CRC, ELLPACK, profil,....).

Sparse BLAS (en cours de spécification, prochaine réunion en Avril 1998, à DC).

Sparse BLAS + routines les utilisant

Matrices creuses sous de nombreux formats

Sparse BLAS Level 2 et 3 ; ex :

`CSR_MatMult_CATB_double()`

$$C \longleftarrow A^T B$$

De nombreux paramètres pour décrire les structures creuses (points et blocs).

De nombreuses routines à écrire (+docs) et à optimiser (constructeurs).

Plus de 10.000 routines évoquées (doc 1997)!?

À suivre!

Portable, Extensible Toolkit for Scientific computation : **PETSc**

Orienté objet, ciblé pour PDE.

Produit matrice-vecteur de l'utilisateur (*pgm objet du pauvre*)

Utilise **BLAS**, **LAPACK**, MPI

Possède aussi des *drivers*, *profilers* et outils de visualisation.

ANL, Bill Gropp et al.

Parallel SPARSe LIBrary : **P-SPARSLIB**

Méthodes itératives, matrices creuses

Préconditionnements parallèles

Partitionnement de graphe, décomposition de domaines

Univ. Minneapolis, Youcef Saad et al.

NAG Parallel Library

....

Voir le NA-DIGEST hebdomadaire pour les nouveautés....

P-SPARSLIB

Partitionnement de graphe

Matrices creuses distribuées (Ax)

Localement CSR, Block Sparse Row ou dense

Ax :

i/ multiplication des blocs diagonaux B_{loc} par variables locales
ii/ prendre les variables externes nécessaires (vecteur distribué)
iii/ multiplier B_{ext} par ces variables locales et sommer

i/ et ii/ possible en parallèle ; B_{loc} et B_{ext} en CSR.

Algorithmes de sous espace de Krylov préconditionnés : FGMRES.
avec GM modifié.

Structure de P-SPARSLIB :

- GMRES, FGMRES, CG, etc...
- outils de préprocessing
- préconditionnements : Jacobi par blocs, SOR multicouleurs
- outils de passage de messages

PETSc - 1 de 2 -

Portable, Extensive Toolkit for Scientific computation

Méthodes implicites pour PDE.

Financé par DoE et NSF

Code en C, orienté objet, Utilisateurs Fortran, C et C++ ciblés

Avant d'installer PETSC : avoir BLAS, LAPACK, MPI

Versions pour DEC alpha, RS6000, Intel Paragon, IBM SP2,...

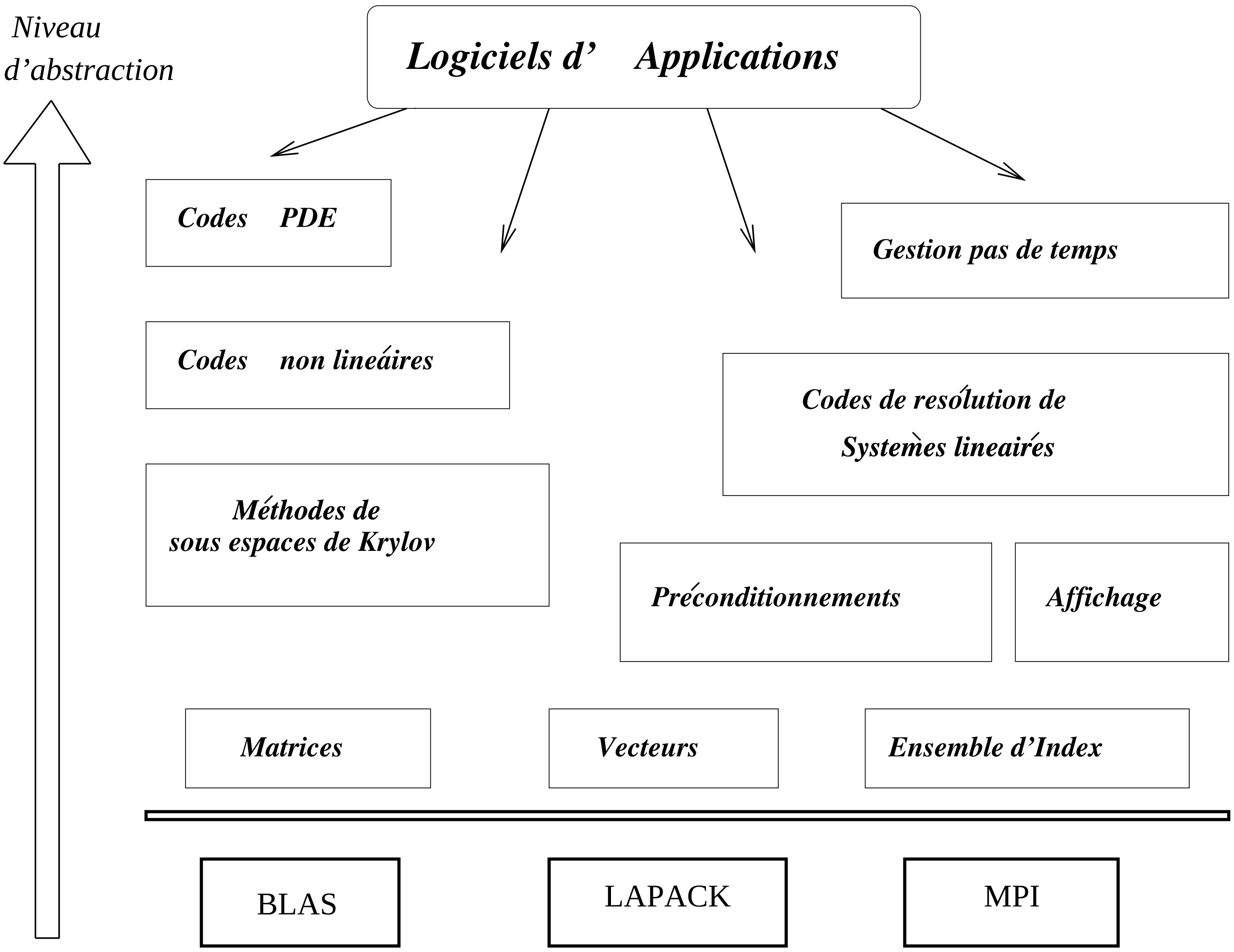
Permet de :
- développer des codes pour problèmes de PDE,
- prototyper rapidement des infrastructures d'algorithmes,
- accéder à un environnement de programmation parallèle.

Encapsulation des données : code indépendant de la représentation des matrices creuses, par exemple.

PETSc - 2 de 2 -

Contient :

- Noyau autour des méthodes de Krylov ; pour matrices creuses.
- Assemblage des matrices
- Préconditionnement
- *Profilling*
- Visualisation des calculs
- Calculs avec matrices creuses



Méthodes de Krylov

Trois parties : projection, calculs de Ritz, test d'arrêt et redémarrage

La phase de projection est composée d'une boucle sur j séquentielle comme suit, (avec v_0 donné et m fixé) :

$$v_j = Av_{j-1} - \sum_{i=0}^{j-1} h_{i,j-1} v_i, \tag{1}$$

$$\text{où } h_{i,j-1} = (Av_{j-1}, v_i); i = 0, j-1, \tag{2}$$

$$h_{j,j-1} = \|v_j\|_2, \tag{3}$$

$$v_j = v_j / \|v_j\|_2. \tag{4}$$

Les opérations parallèles :

$w_j = Av_{j-1}$, **(Sparse PBLAS 2)**

$(w_j, v_i) = h_{i,j}$ pour $i = 1, j-1$, **(PBLAS 1)**

$z_j = \sum_{i=1}^{j-1} h_{i,j} v_i$, **(PBLAS 1)**

$v_j = w_j - z_j$, **PBLAS**

Gram Schmidt **classique**; avec techniques de réorthogonalisation.

Lanczos implicitement redémarré dans le cas Hermitien.

Programmation objet *du pauvre* : le produit matrice vecteur est quelconque. L'utilisateur peut utiliser celui qui souhaite.

IV - Paramétrisation des méthodes et expertises

Choix de certains paramètres, comme la taille des sous-espaces pour les méthodes de type Krylov très utilisées dans les bibliothèques numériques parallèles.

Expression de l'hybridation et de la répartition des calculs selon les modèles de programmation.

Méthodes hybrides et bibliothèques.

Tests de convergences, arithmétiques différentes entre processeurs

La structure d'une matrice creuse joue sur les performances du produit matrice-vecteur (communications et placement). L'optimisation des codes parallèles est donc fonction de ces structures.

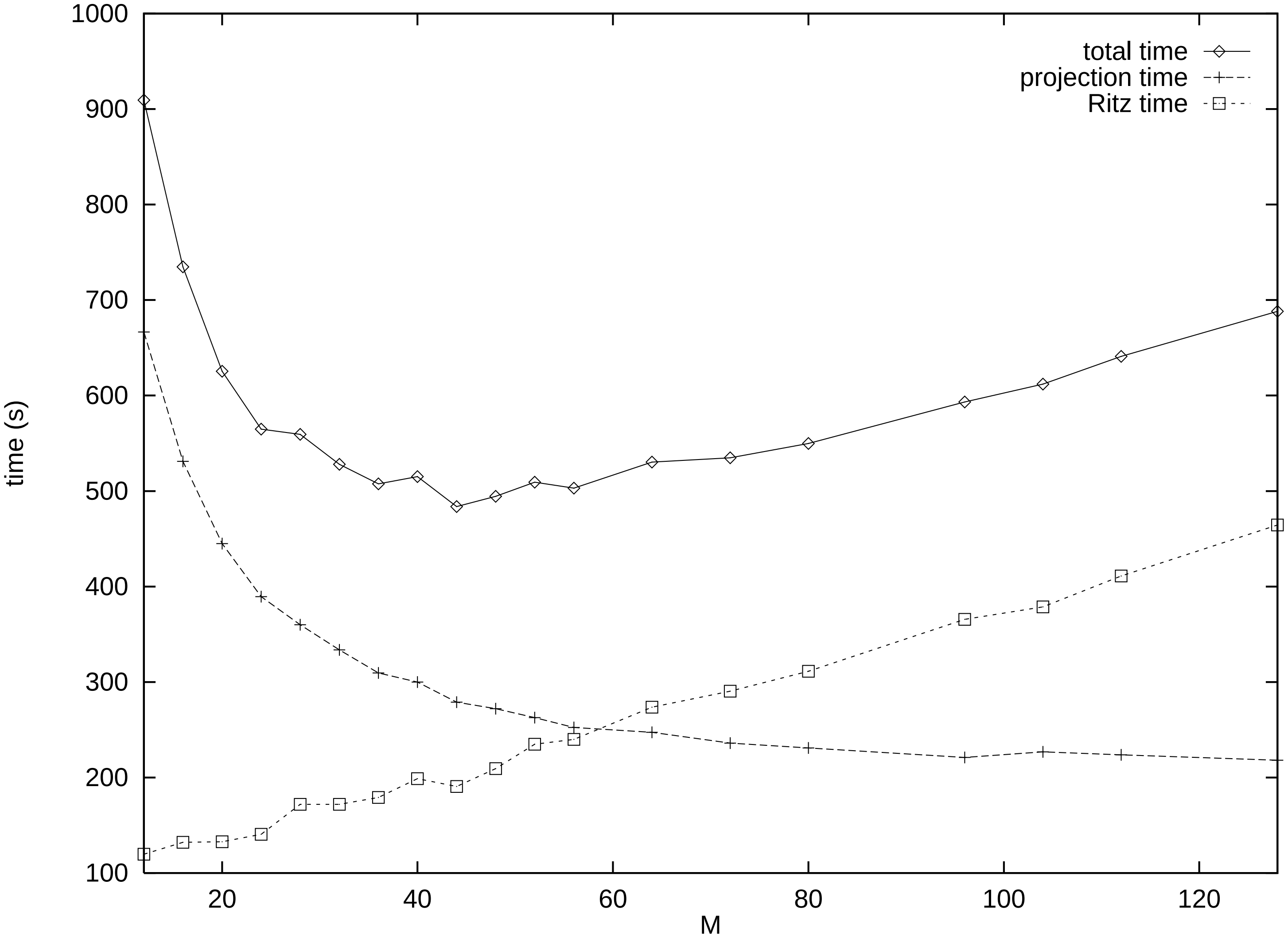


FIG. 1 – *Temps de calcul cumulé pour les diverses parties de la méthode d'Arnoldi jusqu'à convergence, exemple avec* $r = 1$

Il existe une taille optimale pour une matrice et une machine fixées.

V Conclusion

Il existe des outils et techniques pour les calculs *creux* parallèles.

Il existe des bibliothèques proposant des méthodes bien adaptées pour de nombreux calculs scientifiques actuels.

Le choix de certains paramètres et de certaines méthodes reste très délicat et d'importants problèmes demandent de *nouvelles* méthodes parallèles efficaces.

Vers des ateliers de programmation scientifique parallèle assistés d'experts.

- Orienté objet.
- Matrices creuses et maillages adaptatifs
- Out of core
- Modèle de programmation comme *objet*
- Outils de visualisation, y compris interactifs.

Liens nécessaires entre développeurs des bibliothèques et les équipes de recherches (numériques, parallèles et objet).

Drivers parallèles

Vrai MIMD.