ScaLAPACK: une Librairie Logicielle d'Algèbre Linéaire pour Calculateurs Parallèles à Mémoire Distribuée

A. Petitet*

8 janvier 1998

Résumé

Ce papier présente le contenu et les performances de la librairie logicielle ScaLAPACK. Cette librairie contient une collection de programmes et d'algorithmes parallèles pour la résolution efficace de problèmes d'algèbre linéaire dense et creuse sur des calculateurs à mémoire distribuée. Ce document se limite toutefois à la présentation des solutions logicielles proposées pour la résolution des problèmes classiques d'algèbre linéaire dense telles que la résolution de systèmes linéaires et le calcul de valeurs et vecteurs propres.

Les librairies logicielles pour la résolution numérique des problèmes d'algèbre linéaire dense sont mises en perspective afin de permettre au lecteur de mieux comprendre la conception logicielle ainsi que le contenu de la librairie ScaLAPACK. Ces deux derniers aspects sont brièvement discutés. Les différentes composantes logicielles de la librairie sont également présentées. De plus, la manière dont les données sont distribuées, i.e. la décomposition bloc cyclique bi-dimensionelle, est détaillée. L'importance du développement de standards pour l'interface de routines de communication comme MPI (Message Passing Interface), et de calcul d'algèbre linéaire élémentaire comme les "de facto" standard BLAS (Basic Linear Algebra Subprograms) est soulignée. Finalement, un certain nombre de résultats de performance pour certains calculateurs modernes illustrent l'efficacité et la portabilité de la librairie ScaLAPACK.

1 Introduction et Motivation

La librairie logicielle ScaLAPACK contient des algorithmes parallèles d'algèbre linéaire spécifiquement conçus pour atteindre une grande efficacité sur des calculateurs à mémoire distribuée de type MIMD (Multiple Instruction Multiple Data). C'est une continuation du projet LAPACK, qui avait pour objectif la conception et la production d'une librairie logicielle pour la résolution numérique de problèmes d'algèbre linéaire sur des stations de travail, des super-calculateurs vectoriels ainsi que sur des calculateurs parallèles à mémoire partagée [1]. LAPACK et ScaLAPACK contiennent des routines pour la résolution de systèmes linéaires et de problèmes aux moindres carrés, ainsi que pour le calcul de valeurs et vecteurs propres.

Efficacité, fiabilité, flexibilité et facilité d'utilisation sont les principaux objectifs de LAPACK et ScaLAPACK:

- **Efficacité** – les routines de calcul intensif sont exécutées aussi rapidement que possible, et ce, de telle manière que le rendement de chaque processeur demeure constant lorsque le rapport de la taille du problème et du nombre de processeurs est fixé,

*C & C Research Laboratories, NEC Europe Ltd., Sankt Augustin, Germany

- **Fiabilité** – la valeur du résultat obtenu peut être quantifiée par des calculs d'erreurs afin d'estimer sa précision et sa justesse,
- **Flexibilité** – de nouvelles routines peuvent être facilement produites par l'utilisateur en partant de composantes bien conçues,
- **Facilité d'utilisation** – la librairie ScaLAPACK est construite de façon à ressembler de près à LAPACK du point de vue de l'utilisateur.

Le développement et la promotion de standards, tels la spécification de l'interface de routines de calcul et d'échange de messages, permettent d'atteindre la plupart de ces objectifs. Par exemple, l'efficacité de LAPACK repose sur un ensemble restreint de sous-programmes effectuant des opérations élémentaires d'algèbre linéaire – les BLAS (Basic Linear Algebra Subprograms) [12, 4, 5]. L'efficacité des communications effectuées au sein de ScaLAPACK [2] dépend d'un ensemble de routines appellées les BLACS (Basic Linear Algebra Communication Subprograms) [6]. En ce qui concerne les calculs, ScaLAPACK utilise une version parallèle des BLAS, appelés les PBLAS [3]. Ces derniers reposent directement sur les BLAS et les BLACS. Les programmes de LAPACK et ScaLAPACK peuvent donc être exécutés sur n'importe quel calculateur, où les BLAS et les BLACS sont disponibles. L'existence de versions des BLACS au dessus de PVM [10] et de MPI [8] garantit une grande portabilité de la librairie ScaLAPACK.

La conception et l'organisation logicielles de la librairie ScaLAPACK sont détaillées dans la première partie de ce document. Puis, après une brève discussion du BLAS et de LAPACK, la décomposition bloc cyclique bi-dimensionelle utilisée par ScaLAPACK, ainsi que le contenu de cette librairie sont présentés. Un certain nombre de graphes de performance obtenus sur différents calculateurs à mémoire distribuée viennent alors illustrer la portabilité, en terme notamment d'efficacité, des algorithmes adoptés par ScaLAPACK.

2 Conception de ScaLAPACK

2.1 Portabilité, Efficacité et Standards

Pour concevoir des algorithmes d'algèbre linéaire capables d'atteindre de hautes performances sur des calculateurs d'architecture avancée, il est fondamental de minimiser la fréquence du mouvement des données entre les différents niveaux de la hiérarchie mémoire. A cet effet, ScaLAPACK utilise autant que possible des algorithmes *bloqués*. Cette approche algorithmique permet d'exploiter au mieux les unités vectorielles et autres ressources matérielles en parallèle. Cette stratégie est d'autant plus efficace, qu'elle est mise en oeuvre à l'aide de routines de calcul élementaire, comme les BLAS, optimisées pour une architecture donnée. En général, les algorithmes bloqués, ou encore par bloc, requièrent le mouvement de sous-matrices au lieu de vecteurs ou scalaires. Cela réduit largement le coût dù à l'envoi des messages, car moins de messages sont ainsi échangés.

L'utilisateur peut finement régler la performance d'un algorithme pour une architecture spécifique en modifiant les paramètres définissant l'organisation des données en mémoire. Sur des calculateurs à mémoire partagée, cela signifie modifier la taille des blocs, tandis que pour des calculateurs à mémoire distribuée, il s'agit également de faire varier la forme et la taille de la grille virtuelle de processeurs.

Afin d'être véritablement portables, l'interface des routines élémentaires à la base des librairies doit être *standardisée*. La définition de sous-programmes standards de calcul et d'échange de messages [12, 4, 5, 8] fournit aux vendeurs un ensemble clairement défini de routines qu'ils peuvent optimiser. Du point de vue de l'utilisateur, ces standards garantissent la portabilité des programmes qui les utilisent.

Du point de vue du développeur de logiciels numériques, garantir la portabilité peut nécessiter un effort significatif. Les standards permettent à cet effort de développement et de maintenance de logiciels d'être réparti sur le plus grand nombre possible de calculateurs. Etant donné la diversité des architectures parallèles, les programmes ne peuvent être totalement portables, il est, cependant, ainsi possible d'isoler l'impact des différences architecturales des calculateurs à un nombre restreint de sous-programmes.

Un programme efficace se doit de l'être pour un grand nombre de calculateurs différents. L'efficacité des algorithmes parallèles sur une grande variété d'architectures et un grand nombre de processeurs nécessite de pouvoir ajuster la granularité des calculs. Dans ce but, des algorithmes bloqués avec une taille de bloc ajustable sont utilisés. Cependant, des poly-algorithmes, où un algorithme particulier est sélectionné au moment de l'exécution en fonction des données en entrée et des paramètres du calculateur, peuvent s'avérer nécessaires.

La mémoire des calculateurs à venir sera probablement physiquement distribuée. A long terme, les progrès du matériel, des réseaux d'interconnection, des systèmes opératoires, ainsi que des languages et leurs compilateurs, permettront peut-être aux utilisateurs de considérer de telles architectures distribuées comme ayant une mémoire partagée et un espace d'adressage global, et ce sans une perte significative d'efficacité. A court terme, cependant, la nature distribuée du matériel sous-jacent ne peut être ignorée par les programmes. Des procédures de communication explicites continueront donc d'être nécessaires. Cela étant dit, des standards pour les routines d'échange de messages fondamentales (envoyer/recevoir), ainsi que pour des fonctions de communication plus élaborées (sommation globale, diffusion, etc.) sont essentiels au développement de librairies logicielles portables et efficaces. De plus, établir des standards pour des fonctions de communication spécifiques à certains domaines, comme l'algèbre linéaire, pourrait très bien s'avérer avantageux.

2.2 Composantes Logicielles de ScaLAPACK

Dans ScaLAPACK, les algorithmes sont présentés en termes de processus plutôt que de processeurs. En général, plus d'un processus peuvent être exécuté par un seul processeur, auquel cas il est supposé que le système opératoire est responsable du séquencement des processus. En l'absence d'un tel système, ScaLAPACK suppose qu'un seul processus est exécuté par chaque processeur.

La Figure 1 décrit la hiérarchie de la librairie logicielle ScaLAPACK. Les routines des composantes locales au-dessous de la ligne pointillée sont appellées par un seul processus avec des arguments résidant en mémoire locale. Les routines des composantes globales au-dessus de la ligne sont synchrones et parallèles. Ces routines sont appellées avec des arguments qui sont des matrices ou des vecteurs distribués sur l'ensemble des processus selon la décomposition bloc cyclique bi-dimensionelle. Ces composantes sont décrites tour à tour dans les paragraphes suivants.

2.3 Composantes Locales

Les BLAS (Basic Linear Algebra Subprograms) [12, 4, 5] incluent des sous-programmes effectuant des calculs usuels d'algèbre linéaire, comme le produit scalaire de deux vecteurs, la multiplication d'un vecteur par une matrice, ou encore plus généralement la multiplication

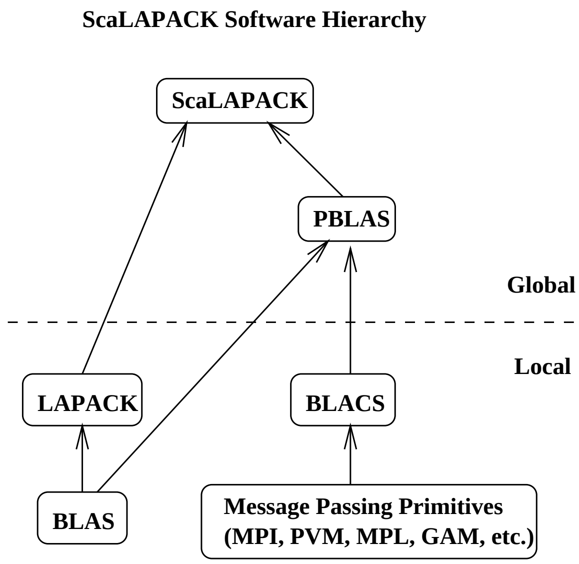


Figure 1: Hierarchie du Logiciel ScaLAPACK

de matrices entre elles. Il est bien connu que l'utilisation d'une routine de produit de matrices optimisée pour une architecture donnée peut effectivement masquer les effets négatifs de la hiérarchie mémoire comme les défauts de cache ou de TLB par exemple. Cela permet également d'atteindre le débit d'opérations flottantes théorique maximal pour un processeur donné.

Comme mentionné plus haut, LAPACK (Linear Algebra PACKage) [1] est une collection de routines résolvant des systèmes linéaires, des problèmes aux moindres carrés, et calculant les valeurs et vecteurs propres d'une matrice. Cette librairie atteint de hautes performances grâce à l'utilisation d'algorithmes qui effectuent la plupart de leur calculs au sein des BLAS, et en particulier du produit de matrices. Chaque routine possède un ou plusieurs paramètres de réglage fin des performances, comme par exemple les tailles des blocs manipulés par les BLAS. Les valeurs optimales de ces paramètres dépendent du calculateur cible, et sont lus dans un tableau à l'exécution.

Les sous-programmes de LAPACK sont conçus pour un seul processeur. Cependant, LAPACK peut accomoder les calculateurs à mémoire partagée ayant plusieurs processeurs à la condition qu'une version parallèle des BLAS soit disponible. Autrement dit, la seule forme de parallélisme exploité est implicite et confinée au sein des appels aux BLAS. La seconde édition du guide de l'utilisateur de LAPACK [1] contient un grand nombre de résultats de performance et ce pour une grande variété de stations de travail et de calculateurs à mémoire partagée.

Les BLACS (Basic Linear Algebra Communication Subprograms) [6] constituent une librairie de sous-programmes d'échange de messages spécifiquement conçue pour les logiciels numériques effectuant des calculs d'algèbre linéaire. Le modèle de programmation consiste en une grille mono- ou bi-dimensionelle de processus, où la mémoire de chaque processus contient des matrices ou des vecteurs. Les BLACS incluent des routines synchrones d'envoi et de réception de matrices d'un processus à un autre, des sous-programmes de diffusion de sous-matrices à un ensemble de processus, ou encore des routines pour des opérations globales de réduction, comme la sommation, et les maxi- et minimisation. De plus, les BLACS contiennent aussi des routines pour construire, modifier, ou bien encore obtenir des informations concernant une grille de processus. Les BLACS permettent également à un processus de devenir membre de plusieurs grilles disjointes ou non, parceque certains algorithmes de ScaLAPACK effectuent des diffusions ou des réductions sur différents sous-ensembles de processus. Un tel sous-ensemble de processus est identifié par ce qu'on appelle un *contexte*. Certains systèmes d'echange de messages comme MPI [8] possèdent également ce type de mécanisme. MPI appelle cela un *communicator*. Finalement, les BLACS permettent aussi aux contextes systèmes d'interopérer avec les contextes définis par les BLACS, et ce sans risque possible de confusion.

2.4 La Distribution Bloc Cyclique Bi-dimensionelle

L'utilisateur d'un calculateur à mémoire distribuée se doit de décomposer ses données sur l'ensemble des processus. La manière avec laquelle une matrice est ainsi distribuée influence beaucoup la répartition des charges de travail, ainsi que les charactéristiques de communication d'un algorithme parallèle. Ainsi, la performance et l'efficacité d'un tel algorithme sont largement déterminées par le schéma de distribution des données adopté. ScaLAPACK suppose que les matrices opérandes doivent être distribuées suivant la décomposition bloc cyclique bi-dimensionelle. Cette méthode est une façon simple et suffisament générale de distribuer une matrice partitionnée par blocs sur une grille bi-dimensionelle de processus. De plus, cette méthode de distribution des données fait partie de la définition du language de programmation High Performance Fortran [11, 7].

a_{11} a_{12}	a_{13} a_{14}	a_{15} a_{16}	a_{17} a_{18}	a_{19}
a_{21} a_{22}	a_{23} a_{24}	a_{25} a_{26}	a_{27} a_{28}	a_{29}
a_{31} a_{32}	a_{33} a_{34}	a_{35} a_{36}	a_{37} a_{38}	a_{39}
a_{41} a_{42}	a_{43} a_{44}	a_{45} a_{46}	a_{47} a_{48}	a_{49}
a_{51} a_{52}	a_{53} a_{54}	a_{55} a_{56}	a_{57} a_{58}	a_{59}
a_{61} a_{62}	a_{63} a_{64}	a_{65} a_{66}	a_{67} a_{68}	a_{69}
a_{71} a_{72}	a_{73} a_{74}	a_{75} a_{76}	a_{77} a_{78}	a_{79}
a_{81} a_{82}	a_{83} a_{84}	a_{85} a_{86}	a_{87} a_{88}	a_{89}
a_{91} a_{92}	a_{93} a_{94}	a_{95} a_{96}	a_{97} a_{98}	a_{99}

Partitionnement d'une matrice 9x9 par bloc 2x2

Figure 2: Partitionnement d'une matrice par blocs

Considérons par exemple une matrice de taille `M` par `N` partitionnée en blocs de taille `MB` par `NB`. Ces blocs sont alors distribués sur la grille de processus de façon uniforme comme illustré par la Figure 3. Chaque processus contient donc un ensemble de blocs, qui sont localement stockés de façon contiguë dans un tableau Fortran 77 à deux dimensions.

La Figure 2 présente le partitionnement d'une matrice d'ordre neuf par blocs carrés de taille deux. La Figure 3 montre comment ces blocs sont alors distribués sur une grille 2×3 de processus, i.e., $M = N = 9$ and $MB = NB = 2$. Les éléments locaux de chaque colonne

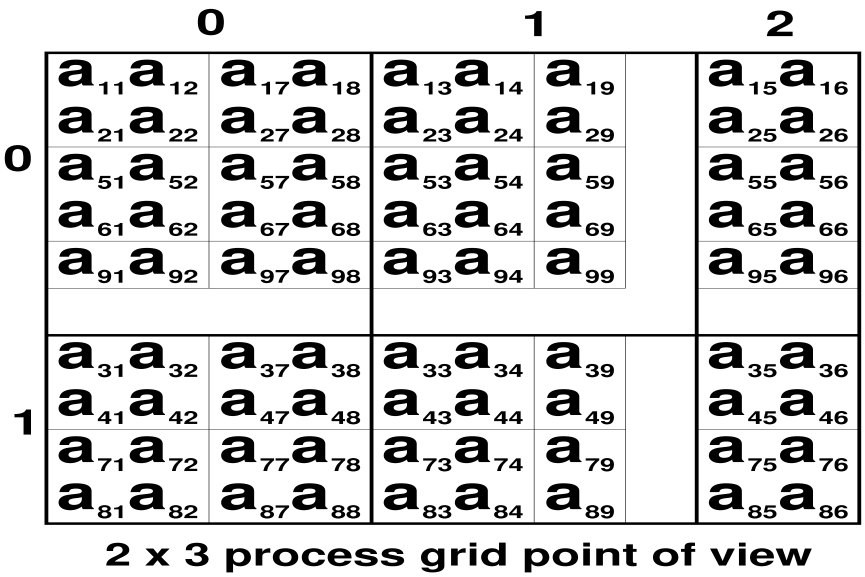


Figure 3: Distribution des blocs d'une matrices sur une grille 2×3 de processus

de la matrice sont stockés de façon contiguë dans la mémoire de chaque processus. De plus amples détails concernant les méthodes de distribution de données sont fournis dans [2].

3 Contenu de la Librairie Logicielle ScaLAPACK

La librairie logicielle ScaLAPACK contient des routines pour la résolution de systèmes linéaires pour des matrices denses, bandes ou tridigonales, générales ou symétriques définies positives. Pour les matrices denses, il existe également des routines estimant leur conditionnement, calculant leur inverse, et effectuant le raffinement itératif des vecteurs solutions du système. De plus, ScaLAPACK contient des sous-programmes résolvant numériquement les problèmes suivants: problèmes aux moindres carrés pour des matrices de rang maximal, factorisations orthogonales simples et généralisées, application et génération de transformations orthogonales, réductions sous forme de Hessenberg, bidiagonale et tridigonale, réduction sous forme standard du probème du calcul de valeurs et vecteurs propres généralisé symétrique défini positif, décomposition en valeurs singulières, calcul des valeurs et vecteurs propres pour les problm̀es symétriques, symétriques généralisés et non symétriques. Enfin, la librairie logicielle ScaLAPACK contient un certain nombre de sous-programmes classés dans une catégorie "prototype", et ce parcequ'ils concernent des domaines de recherche actuels et font encore l'objet de réflexions et travaux. On peut trouver par exemple dans cette catégorie des routines utilisant l'agorithme de la fonction signe pour le calcul des valeurs et vecteurs propres d'une matrice non symétrique, la résolution d'un système linéaire creux utilisant la méthode super nodale pour des calculateurs à mémoire partagée et/ou distribuée. Cette catégorie contient également des routines pour la résolution de systèmes linéaires denses, lorsque la matrice de coefficients réside sur disque, ou pour le cas symétrique défini positif utilisant une méthode de stockage économique par bloc de colonnes. Enfin, un ensemble de routines d'interface directement appelables d'un programme HPF [7, 11] (High Performance Fortran) et appelant les routines principales de ScaLAPACK font aussi partie de cette catégorie "prototype". Tous ces sous-programmes sont disponibles pour des matrices de réels ou de complexes, simple ou double précision. Les sous-programmes inclus dans ScaLAPACK sont classées comme suit:

- **drivers:** – Chacune de ces routines résoud complètement un problème donné, comme par exemple le calcul de la solution d'un système d'équations linéaires, ou encore le calcul des valeurs propres d'une matrice symétrique. Il est recommandé aux utilisateurs d'utiliser ces drivers, lorsqu'il en existe un correspondant à la résolution du problème souhaité. Ces routines vérifient en entrée la cohérence globale et locale des arguments qui leur sont passés.

- **routines de calcul:** – Chacune de ces routines effectue une tâche calculatoire distincte, comme par exemple le calcul de la factorisation LU d'une matrice dense générale, ou encore le calcul de la réduction d'une matrice symétrique sous forme tridiagonale. Une routine "driver" est essentiellement une séquence d'appels à des routines de calcul. Les utilisateurs, et plus particulièrement les développeurs de logiciels peuvent avoir besoin d'appeler directement ces routines de calcul, lorsqu'il n'existe pas de driver approprié. Ces routines vérifient aussi en entrée la cohérence globale et locale des arguments qui leur sont passés.

- **routines auxilliaires:** – Ces routines sont à leur tour classées comme suit:

  - les sous-tâches des algorithmes bloqués — en particulier les versions non bloquées de ces algorithmes;
  - les calculs de bas-niveau comme par exemple la multiplication d'une matrice par un scalaire, ou encore le calcul de la norme d'une matrice, ou enfin la génération d'une matrice de Householder élémentaire; Certaines de ces routines présentent un intérêt certain pour le numéricien et/ou le dévelopeur de logiciels numériques, et peuvent être considérées comme des candidats potentiels à ajouter aux futures versions des PBLAS;
  - un certain nombre d'extensions aux PBLAS, comme par exemple des opérations faisant intervenir une matrice complexe symétrique et un vecteur (les PBLAS ne font pas partis de ScaLAPACK au sens strict).

4 Performance

Une métrique de performance importante est *l'efficacité parallèle*. L'efficacité parallèle, $E(N,P)$, pour un probème de taille N sur P processeurs est définie de manière usuelle [9] comme suit:

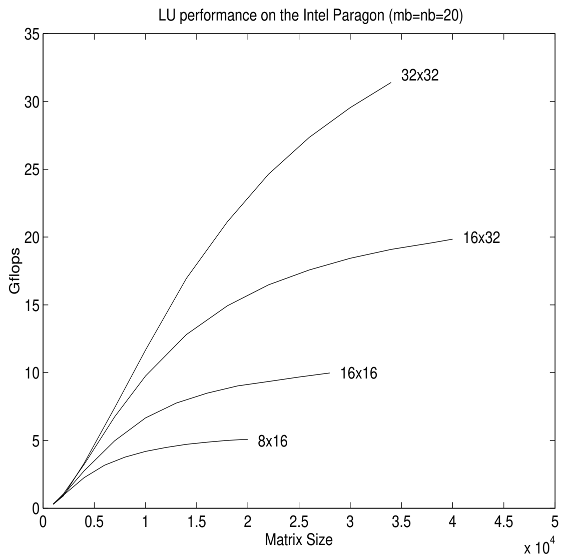
$$E(N,P) = \frac{1}{P}\frac{T_{\text{seq}}(N)}{T(N,P)}, \tag{1}$$

où $T(N,P)$ est le temps d'exécution de l'algorithme parallèle, et $T_{\text{seq}}(N)$ est le temps d'exécution du meilleur algorithme séquentiel. Un programme réalisant cette opération est dit efficace, lorsque l'efficacité parallèle est une fonction croissante de N/P, i.e. la taille du problème par processeur. Dans le cas du calcul matriciel dense, N est le nombre de coefficients de la matrice.

La performance d'un algorithme est également mesurée en Megaflops (ou Gigaflops) par seconde. Cette unité de mesure est appropriée aux calculs d'algèbre linéaire dense, puisque le coût des opérations arithmétiques flottantes domine celui des communications.

Un algorithme est qualifié d'efficace, lorsque cette mesure de performance est proportionelle à P avec N/P fixé.

Les hautes performances des algorithmes présents dans ScaLAPACK sont obtenues d'une part grâce à la réduction des coûts dus à une mauvaise répartition des charges et des mouvements de données, et d'autre part grâce à la restructuration bloquée des algorithmes. La façon dont les données sont distribuées sur la mémoire hiérarchique du calculateur est d'une importance fondamentale à la réduction de ces coûts.



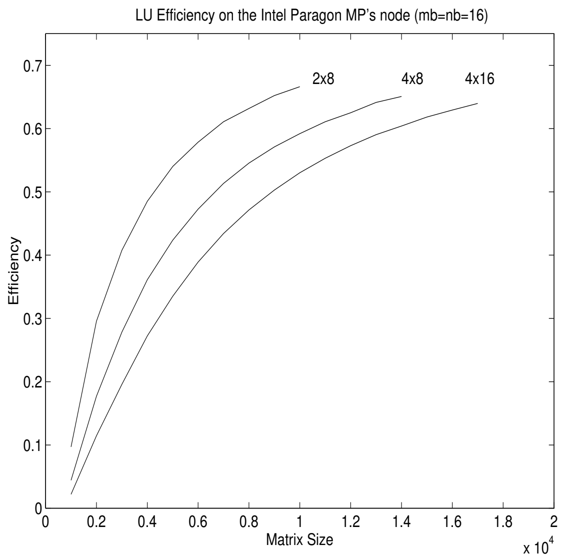


Figure 4: Performance des factorisations LU et QR sur un Intel Paragon

Cette section présente des résultats de performance sur différentes plateformes pour certaines routines de factorisations et de réductions. Les Figures 4 et 5 montrent la performance des factorisations LU, Cholesky et QR et réductions classiques pour différentes tailles de problèmes et configurations matérielles sur les calculateurs Intel Paragon et IBM SP-2. Le nombre de processeurs utilisés dans les expériences est spécifié par la taille de la grille de processeurs donnée pour chaque courbe du graphe. Ces figures montrent l'efficacité de ces algorithmes lorsque la taille du problème et le nombre de processeurs augmentent. Le guide de l'utilisateur de ScaLAPACK [2] contient des résultats de performance plus complets, ainsi que leur analyse pour une plus grande variété de calculateurs modernes et un plus grand nombre de routines présentes dans ScaLAPACK.

5 Conclusions

La librairie logicielle ScaLAPACK peut être utilisée sur un grand nombre de calculateurs à mémoire distribuée, comme par exemple la série des calculateurs IBM SP, les calculateurs Intel (Gamma, Delta, Paragon), la série des Cray T3, la CM-5 de Thinking Machines, les clusters de stations de travail, et de manière générale n'importe quel système pour lequel PVM [10] ou MPI [8] sont disponibles. Le développement et la promotion de standards, en particulier pour des routines de communication et de calcul de bas niveau contribuent au nombreux objectifs du projet ScaLAPACK, de la même façon que les BLAS pour LAPACK. Limiter et confiner les différences architecturales des calculateurs au sein de deux

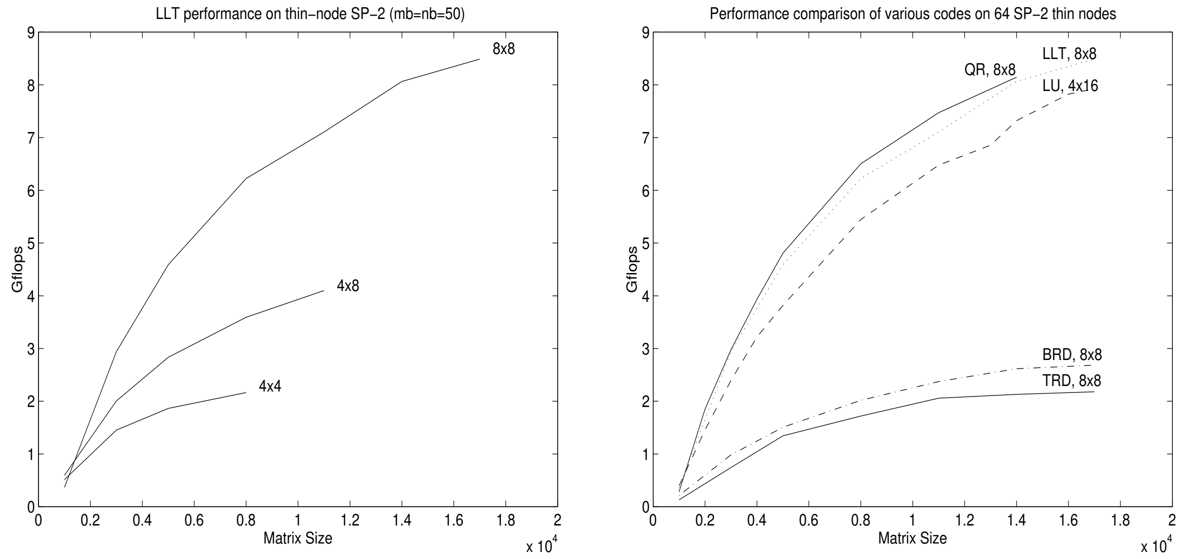


Figure 5: Performance de la factorisations Cholesky sur IBM SP-2 et Intel Paragon

librairies logicielles telles les BLAS (Basic Linear Algebra Subroutines) et les BLACS (Basic Linear Algebra Communication Subroutines) ont permis d'atteindre ces objectifs avec succès. En conséquence, ScaLAPACK peut être utilisé sur n'importe quel calculateur, où les BLAS et les BLACS sont disponibles. L'ensemble du logiciel ScaLAPACK ainsi qu'une large documentation sont publiquement disponibles sur *netlib* via l'URL:

http: //www.netlib.org/scalapack/index.html

Références

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK Users' Guide, Second Edition*. SIAM, Philadelphia, PA, 1995.

[2] L. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. *ScaLAPACK Users' Guide*. SIAM, Philadelphia, PA, 1997.

[3] J. Choi, J. Dongarra, S. Ostrouchov, A. Petitet, D. Walker, and R. C. Whaley. A Proposal for a Set of Parallel Basic Linear Algebra Subprograms. In J. Dongarra, K. Masden, and J. Waśniewski, editors, *Applied Parallel Computing*, pages 107–114. Springer Verlag, 1995. (also LAPACK Working Note No.100).

[4] J. Dongarra, J. Du Croz, S. Hammarling, and R. Hanson. Algorithm 656: An extended Set of Basic Linear Algebra Subprograms: Model Implementation and Test Programs. *ACM Transactions on Mathematical Software*, 14(1): 18–32, 1988.

[5] J. Dongarra, J. Du Croz, S. Hammarling, and R. Hanson. An extended Set of FORTRAN Basic Linear Algebra Subprograms. *ACM Transactions on Mathematical Software*, 14(1): 1–17, 1988.

[6] J. Dongarra and R. C. Whaley. A User's Guide to the BLACS v1.0. Technical Report UT CS-95-281, LAPACK Working Note No.94, University of Tennessee, 1995. (http://www.netlib.org/blacs/).

[7] High Performance Forum. High Performance Fortran Language Specification. Technical Report CRPC-TR92225, Center for Research on Parallel Computation, Rice University, Houston, TX, May 1993.

[8] Message Passing Interface Forum. MPI: A Message Passing Interface Standard. *International Journal of Supercomputer Applications and High Performance Computing*, 8(3–4), 1994.

[9] G. Fox, M. Johnson, G. Lyzenga, S. Otto, J. Salmon, and D. Walker. *Solving Problems on Concurrent Processors*, volume 1. Prentice Hall, Englewood Cliffs, N.J, 1988.

[10] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. *PVM: Parallel Virtual Machine. A User's Guide and Tutorial for Networked Parallel Computing.* The MIT Press, Cambridge, Massachusetts, 1994.

[11] C. Koebel, D. Loveman, R. Schreiber, G. Steele, and M. Zosel. *The High Performance Fortran Handbook.* The MIT Press, Cambridge, Massachusetts, 1994.

[12] C. Lawson, R. Hanson, D. Kincaid, and F. Krogh. Basic Linear Algebra Subprograms for Fortran Usage. *ACM Transactions on Mathematical Software*, 5(3): 308–323, 1979.