5 D − 6

自動チューニング機能付き並列数値計算ライブラリ構築の試み — 対称疎行列用の連立一次方程式ソルバを例にして — *

黒田 久泰¶ 片桐 孝洋¶ 佃 良生¶ 金田 康正‖ †
¶東京大学大学院理学系研究科情報科学専攻‡ ‖東京大学大型計算機センター§

1 はじめに

近年分散メモリ型並列計算機用の数値計算ライブラリが数多く構築され，かつ自由に入手できるようになってきた．例えば，ScaLAPACK[1] がその一例である．ところが，これらのライブラリを利用するという観点から考えると，必ずしも利用者にとって使い易いインタフェースになっているとは言い難い．なぜならば，これらライブラリを利用するためには，非常に多くのパラメタを付加しなくてはいけないことが多いからである．インタフェースに多くのパラメタを必要とするライブラリは，多機能の電家製品のように，その使い方に手間取ることは明白である．

利便性を高めるために，これら性能に関するパラメタに人手で値を入力するのではなく，自動的に値を最適化して設定することが望ましい．しかもこのことは，高性能を達成することにもなる．この自動最適化の操作は，機種依存の要因を自動的に解決していることと同値で，いわば機種ごとにコードを自動生成することになるからである．

数値計算ソフトウエアにおいて，機種ごとにコードを自動生成する研究は，既になされている．例えば，階層メモリ構造やパイプライン処理を考慮した数学ソフトウエアのコード自動生成と最適化を目指すプロジェクトとして，ATLAS プロジェクト [2] が挙げられる．しかしながら，並列計算を考慮し，ライブラリ単位で数値計算ソフトウエアを最適化する試みはきわめて少ない．そこで本研究では，連立一次方程式のソルバを例として，ライブラリ構築を試みる．

連立一次方程式 $Ax = b$, $A \in \Re^{n\times n}$, $x, b \in \Re^n$ の解 x を求める並列数値計算ライブラリの場合，少なくとも以下のようなパラメタをもつ．(i) 使用する計算アルゴリズムの指定 (例えば，SOR 法，共役勾配法 (CG 法) など)，(ii) 演算性能に関するパラメタ (例えば，ブロック化アルゴリズムにおけるブロック幅やループ展開の段数など)，(iii) 並列処理方式の指定 (例えば，1 対 1 通信を使うか放送機能を使うか)．

ここで (i) のパラメタの自動決定は，一般に解きたい行列の性質を判定することは難しく，非常に困難であることに注意しておく．この問題を回避するため，本ライブラリの利用者は少なくとも解くべき問題の性質を良く知っていることとする．いまここで利用者は，解くべき問題は正値実対称疎行列であり，そして解法として CG 法が有効であることを知っていると仮定する．

2 自動チューニング機能

正値実対称疎行列を係数行列 A にもつ連立一次方程式を，分散メモリ型並列計算機を用いて (すなわち，メッセージ交換機能を用いて) CG 法で解く場合，以下の項目を指定する必要がある．

(a) 非零要素の位置に特化した計算方式の選択
(b) 疎行列 − ベクトル積 ($Ap, p \in \Re^n$) におけるループ展開段数の選択
(c) 通信方式の選択

[(a) に関して]: CG 法は正値実対称疎行列を係数行列にもつ連立一次方程式の効率的な解法の一つとして広く知られている [3] が，疎行列の非零要素の性質によって，より効果的な CG 法の計算方式を構築できることがある．そこで本ライブラリでは，特定の非零要素の型を検索することで，以下のように計算方式を自動選択する．(1) **三重対角**：三重対角用 LU 分解法 (逐次ルーチン)，(2) **5 点差分**：5 点差分に特化した不完全コレスキー分解前処理付き CG 法 (並列ルーチン)，(3) **その他**：前処理付き CG 法 (並列ルーチン)．ここで，(3) の前処理には 2-step Jacobi 前処理を用いている．具体的には，A がスケーリング済みの行列とすると，$M^{-1} = (I - L - U)$, (ここで L は A の対角要素を除く下三角行列，同様に U は上三角部分) を前処理行列とする．

[(b) に関して]: 疎行列とベクトルの積を行う場合において，一般的に `A[i] * p[ind[i]]` のような間接参照がなされる．この場合，i-ループ に関して n 段展開すると

```
A[i+1] * p[ind[i+1]] + A[i+2] * p[ind[i+2]]
+ ··· + A[i+n] * p[ind[i+n]]
```

---

*Constructing Automatically Tuned Parallel Numerical Calculation Library — A Case of Symmetric Sparse Linear Equations Solver —

†KURODA Hisayasu, KATAGIRI Takahiro, TSUKUDA Yoshio and KANADA Yasumasa
‡Department of Information Science, Graduate School of Science, the University of Tokyo
§Computer Centre, the University of Tokyo

のようになる．このループ展開により，演算が高速化される場合がある．しかし，この段数はレジスタ数などの機種依存の要因で決まることから，同一段数の展開コードは全ての計算機で最適ではない．本ライブラリでは，1行当たりの非零要素数がある数以下の場合，列方向に 1，2，3，4，8 段の展開済のコード (プリフェッチ向きコードを含む二種) を用意する．そして CG 法反復に入る前に数回，あるループ長で展開コードを用いた行列-ベクトル積時間を測定することで，最速のループ展開コードを選択する．

[(c) に関して]:　疎行列-ベクトル積を並列に行う場合，メッセージ交換による通信が必要になる．この通信処理は疎行列の非零要素の位置に依存する．ところが，対象の疎行列と使用する PE 数を固定した場合でも，その通信の実装方法により通信時間が増加することがある．例えば通信ライブラリの標準規格の1つである MPI(Message Passing Interface) を用いて実装するとする．このときベクトルの要素を集める操作をする場合，(I)MPI が提供する高機能なルーチンを用いるか，(II) 低機能な 1 対 1 通信を用いて自分で実装するか，選択の余地がある．もしベンダが十分高性能な MPI ルーチンを提供している場合は (I) の実装法が有効となるが，そうでないなら (II) の実装法が有効になるだろう．本ライブラリでは，あらかじめ (I)(II) の実装法の実行時間をチェックして有効な実装法を自動的に選択する．このとき，(I) の実行時間と比較して，(II) の実行時間が一度に何回 1 対 1 通信が発生する場合まで有効か調べ，その値を保存する．この値により，疎行列の非零要の位置が変化しても，反復に入る前に通信回数を数えることで，高速な実装方式の選択が可能となる．

3　実験結果

ディリクレ境界条件を持つ 2 次元単位領域におけるポアソン方程式 $-\nabla(k\nabla u) = f$ in $\Omega = (0,1) \times (0,1)$ with $u = 0$ on $\partial\Omega$ において，領域を 512×512 のメッシュで離散化を行う．この問題に対して，日立 SR2201（16PE 使用）を用いて実験を行った．なお，この実験での本ライブラリのコンパイルには，以下に示す二種のコンパイラオプションを用いる．

[オプション -W0,'opt(o(0))'：非最適化版]

Unroll=(α, β) は 列方向 α段，行方向 β段のループ展開段数を示す．：の後の数字は時間で単位は秒である．なお計算方式の自動判定では，(3) その他の計算方式を自動選択した．

```
----- How To Unroll - Auto Tuning
Unroll=(1,1): 0.028979  Unroll=(1,5): 0.011536
Unroll=(2,5): 0.011182  Unroll=(3,5): 0.011747
Unroll=(4,5): 0.012301  Unroll=(8,5): 0.012612
Prefetch Unroll=(1,5): 0.015810
Prefetch Unroll=(2,5): 0.012946
Prefetch Unroll=(3,5): 0.013595
Prefetch Unroll=(4,5): 0.013683
Use Unroll = (2,5)
----- How To Communicate - Auto Tuning
MPI_Allgather :0.025872  One -> All    :0.068234
Isend -> Irecv:0.001735  Irecv -> Isend:0.001160
Use Irecv -> Isend
iteration=886, |b - Ax|_inf = 9.25385e-06
Elapsed time:   66.984 sec
```

[オプション -W0,'pvec(diag(1)),opt(o(s))'：最適化版]

```
----- How To Unroll - Auto Tuning
Unroll=(1,1): 0.009183  Unroll=(1,5): 0.002914
Unroll=(2,5): 0.003558  Unroll=(3,5): 0.004161
Unroll=(4,5): 0.004656  Unroll=(8,5): 0.005819
Prefetch Unroll=(1,5): 0.002748
Prefetch Unroll=(2,5): 0.009551
Prefetch Unroll=(3,5): 0.008503
Prefetch Unroll=(4,5): 0.011034
Use Prefetch Unroll = (1,5)
----- How To Communicate - Auto Tuning
MPI_Allgather :0.025955 One -> All    :0.068980
Isend -> Irecv:0.001770 Irecv -> Isend:0.000981
Use Irecv -> Isend
iteration=886, |b - Ax|_inf = 9.25851e-06
Elapsed time:   43.116 sec
```

以上の実験結果から，我々のアプローチによるループ展開段数や通信実装方式などの自動選択は効果があることがわかる．このことは，コンパイラや通信の実装方式の違いから生じる機種依存の要因を，柔軟に解決していることを意味している．

今後の課題として，CG 法以外の計算アルゴリズムの自動選択がどこまで可能なのか検討することなどが挙げられる．

謝辞　本研究の一部は，文部省科学研究費特定領域研究「発見科学」（課題番号 10143103）の支援により行った．

参考文献

[1] : ScaLAPACK Project
http://www.netlib.org/scalapack/index.html.

[2] : ATLAS project
http://www.netlib.org/atlas/index.html.

[3] Barrett, R., Berry, M., Chan, T. F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C. and van der Vorst, H.: *Templates for the Solution of Linear Systems : Building Blocks for Iterative Methods*, SIAM (1994).