ハイパフォーマンス
コンピューティング 68－3
（１９９７．１０．１７）

ScaLAPACKのVPP500への移植

稲荷 淳[1] 岡部 寿男[2] 金澤 正憲[2]

[1] 京都大学大学院工学研究科

[2] 京都大学大型計算機センター

分散メモリ型並列コンピュータ用の線形計算ライブラリとして知られているScaLAPACKをVPP500へ移植している。オリジナル版のScaLAPACKは、基本線形計算を行うPBLAS、プロセッサ間通信を行うBLACS、ScaLAPACK本体の各部分から構成されるが、開発しているVPP版においては、ハードウェア依存の最適化はPBLASに集約しScaLAPACK本体には手を加えない、という方針を立てた。言語はVPP Fortranを用い、配列は原則として列方向サイクリック分割された分割ローカル配列を用いることとした。

以上の基本方針に基づいて、行列積計算、三角方程式の解法、LU分解の各ルーチンがVPP500上で最適に動作するよう、そのアルゴリズムについて考察し、作成したルーチンについて性能を評価する。

Implementing ScaLAPACK on VPP500

Kiyoshi Inari[1] Yasuo Okabe[2] Masanori Kanazawa[2]

[1] Faculty of Engineering, Kyoto University

[2] Data Processing Center, Kyoto University

We have transplanted a distributed parallel version of the famous linear algebra library ScaLAPACK to VPP500. Original ScaLAPACK includes PBLAS, BLACS and ScaLAPACK routine. As to ScaLAPACK for VPP500, we decided that only PBLAS routine is dependant on the VPP500 hardware, and ScaLAPACK routine must not be modified. When we implement the routines in VPP Fortran, we almost use column-directed cyclic distributed local array.

In this report, we consider the algorithm for optimal calculation of multiplication, multiple triangular systems of equations and LU factorization on VPP500, and evaluate its performance.

1 はじめに

今日、科学技術計算や数値シミュレーションといった数値演算が大規模になるにつれ、スーパーコンピュータが果たすべき役割は大きくなってきたと言える。なかでも、マルチプロセッサ型並列計算機の一層の活躍が期待される。

しかし、過去のプログラムを使うにしても大幅な手直しが必要であること、並列プログラミングが複雑であること、並列化することはできるものの、なかなか速くなってくれないことなどの要因から、普及には至っていない。

本報告では、MIMD 型分散メモリコンピュータ用線形計算ライブラリとして公開されている ScaLAPACK を VPP500 に移植する試みについて述べる。以下、ScaLAPACK の紹介、ScaLAPACK の VPP500 への移植にあたっての検討事項、作成したルーチンの性能評価について、順次述べる。

2 ScaLAPACK

netlib(`http://www.netlib.org/`) は、MIMD 型分散メモリコンピュータや、PVM、MPI をサポートしているワークステーション用の線形計算ライブラリとして、ScaLAPACK(Scalable Linear Algebra PACKage) を公開している。これは、ベクトルスーパーコンピュータや共有メモリ型並列コンピュータ用の線形計算ライブラリ LAPACK(Linear Algebra PACKage) の拡張として開発された。

LAPACK や ScaLAPACK には、連立一次方程式、線形最小 2 乗法と固有値問題の解析と解法のためのルーチンが含まれている。

LAPACK は、機種依存部分を基本線形計算サブルーチン群 BLAS (Basic Linear Algebra Subprograms) に集約し、LAPACK 自身は機種に依存しない。ScaLAPACK でも同様に、機種依存部分を並列版基本線形計算サブルーチン群 PBLAS(Parallel BLAS) とメッセージパッシングライブラリ BLACS(Basic Linear Algebra Communication Subprograms) に集約している。

ScaLAPACK で用いられる変数は、図 1(a) のように複数プロセッサに 2 次元ブロックサイクリック分割されている。図は、2 × 3 格子へのブロック分割を示し、各格子の中の数字は割り付けられるプロセッサ番号を示している。

	0	1	2	3	4	5	6	7	8	9	10	11
0	0	1	2	0	1	2	0	1	2	0	1	2
1	3	4	5	3	4	5	3	4	5	3	4	5
2	0	1	2	0	1	2	0	1	2	0	1	2
3	3	4	5	3	4	5	3	4	5	3	4	5
4	0	1	2	0	1	2	0	1	2	0	1	2
5	3	4	5	3	4	5	3	4	5	3	4	5
6	0	1	2	0	1	2	0	1	2	0	1	2
7	3	4	5	3	4	5	3	4	5	3	4	5
8	0	1	2	0	1	2	0	1	2	0	1	2
9	3	4	5	3	4	5	3	4	5	3	4	5
10	0	1	2	0	1	2	0	1	2	0	1	2
11	3	4	5	3	4	5	3	4	5	3	4	5

(a)2×3 グリッドへのブロック分割

	0	1	2	3	4	5	6	7	8	9	10	11
0	0	1	2	3	0	1	2	3	0	1	2	3
1	0	1	2	3	0	1	2	3	0	1	2	3
2	0	1	2	3	0	1	2	3	0	1	2	3
3	0	1	2	3	0	1	2	3	0	1	2	3
4	0	1	2	3	0	1	2	3	0	1	2	3
5	0	1	2	3	0	1	2	3	0	1	2	3
6	0	1	2	3	0	1	2	3	0	1	2	3
7	0	1	2	3	0	1	2	3	0	1	2	3
8	0	1	2	3	0	1	2	3	0	1	2	3
9	0	1	2	3	0	1	2	3	0	1	2	3
10	0	1	2	3	0	1	2	3	0	1	2	3
11	0	1	2	3	0	1	2	3	0	1	2	3

(b)4 プロセッサへの列方向分割

図 1: サイクリック分割

3 VPP500 への移植

3.1 基本方針

ScaLAPACK の VPP500 への移植にあたっては、

- ScaLAPACK 本体ルーチンには、並列化指示行の挿入を除き手を入れない。
- PBLAS に相当する部分を、original 版 BLAS をもとに作成し[1]、VPP に特化した最適化や VPP Fortran に依存した並列化は、この部分にのみ適用する。
- BLACS ルーチンは用いない。プロセッサ間通信は PBLAS ルーチン内で行う。
- ScaLAPACK、PBLAS はそれぞれ分割コンパイル可能にしてライブラリ化できるようにする。(かつ可能ならば) プロセッサ数非依存とする。

のような基本方針をたてた。

配列[2]については次のような方針をたてた。

- 原則としてグローバル配列は使わない。
- 配列の分割は、行列の列方向次元に沿ったサイクリック分割(図 1(b))とする。これは、各プロセッサエレメントが持つベクトルプロセッサの性能を発揮させるためである。

また、プロセッサ間で通信を行うには次の二つの方法を用いるものとする。

BROADCAST 文による方法 通信用のバッファとして別途重複ローカル配列を確保しておき、特定プロセッサがこの配列にデータを書き込んだ後 BROADCAST (図 2) する。

UNIFY 文による方法 通信用のバッファとして別途重複ローカル配列を確保しておき、各プロセッサがこの配列の割り当てられた部分を書き込んだ後 UNIFY(図 3) する。

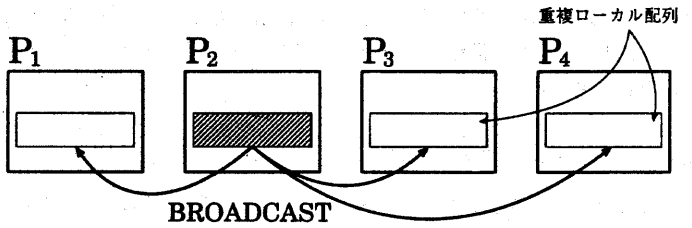


図 2: BROADCAST 文

3.2 アルゴリズム

この節では、LU 分解に必要となるルーチン群のうち、行列積計算ルーチン及び三角方程式の解法ルーチンのアルゴリズムについて述べる。

[1] ここで PBLAS をもとに作成しないのは、ブロックサイクリック分割を用いた original 版に手を加えて列方向サイクリック分割を用いた VPP 版を作り出すよりも、BLAS を並列化するほうが用意であると考えられるからである。

[2] VPP500 で使用できる配列は、グローバル配列 (各プロセッサからアクセス可)、分割ローカル配列 (各プロセッサのローカル空間に分割して割り付け)、重複ローカル配列 (各プロセッサに重複して割り付け) の 3 種類。

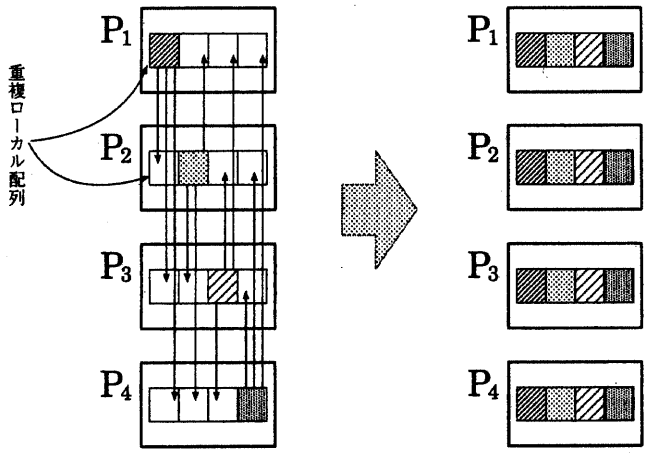


図 3: UNIFY 文

3.2.1 行列積計算 (PBLAS ルーチン)

このルーチンでは、$C = C + A \times B$ の計算を行う。行列 A, B, Cはそれぞれ列方向サイクリック分割された分割ローカル配列として宣言されていることとする。A、Bは転置行列であってもよいが、ここでは簡単のため、いずれも転置でない場合を考える。

$$c(i,j) = c(i,j) + \sum a(i,l)b(l,j)$$

の計算において、A は $m \times k$ 行列、Bは $k \times n$ 行列、Cは $m \times n$ 行列である。ループは内側から i, j, lという順で計算する。このルーチンにおいて計算を行うための条件は、Bと Cの分割が揃っていること、すなわち Bと Cの第 1 列が同じプロセッサに割り当てられていることである。

以下に、このルーチンのアルゴリズムを示す (図 4)。使用できるプロセッサ数を npe とし、各プロセッサを $P_1, P_2, \cdots, P_{npe}$と書く。また、通信バッファとして用いる重複ローカル配列を $mem()$ とし、その大きさ (要素数) を $msize$ とする。

以下の操作を $lpe = 1$ から npe まで npe 回繰り返す

1. プロセッサ P_{lpe}において、P_{lpe}に割り当てられている A の要素を列単位でバッファ $mem()$ に書き込む。このとき、$rows = msize/m$ 列の要素がバッファに書き込まれる (第 r_1列, 第 r_2列,$\cdots$, 第 r_{rows}列とする)。

2. $mem()$ を BROADCAST する。

3. [lループ] $l = 1$ から $rows$ までのループ

   [jループ] $j = 1$ から n までのループ。各プロセッサは割り当てられている B, Cの要素に対して以下の操作を行う。

   [i ループ] $i = 1$ から m まで、
   $c(i,j) = c(i,j) + a(i,r_l)b(r_l,j)$
   を計算する。A の要素は、バッファから読み込む。

4. 1. から 3. の操作を、P_{lpe}に割り当てられた A の要素すべてに対して行うまで繰り返す。

3. の操作において、jループは各プロセッサが並列に行うことができ、さらにiループにおいてはベクトル化可能である。

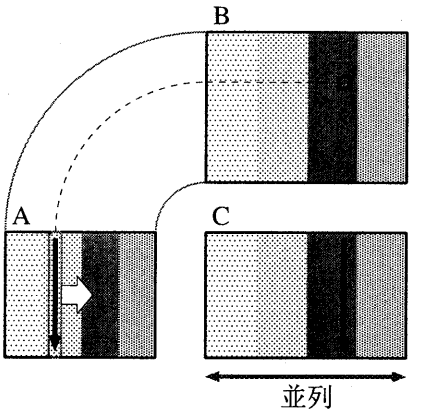
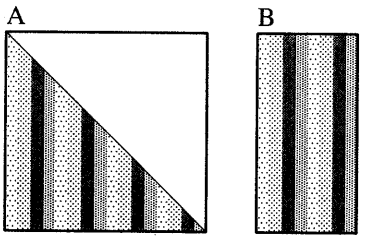


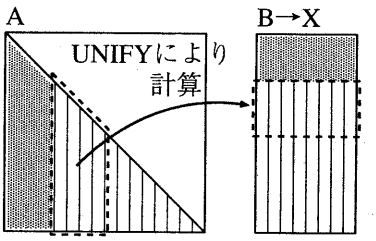
図 4: 行列積計算

3.2.2 三角方程式の解 (PBLAS ルーチン)

このルーチンでは、$AX = B$ の解法、つまり $X = A^{-1}B$の計算を行う (図 5(a))。A は $m \times m$ の上三角行列または下三角行列、Bは $m \times n$ 行列で、列方向サイクリック分割された分割ローカル配列として宣言されている。A は転置行列であってもよい。行列 Bは、このルーチンの終了時には解 Xとして上書きされる。



(a) 三角方程式



：計算済

(b) 三角方程式の解法アルゴリズム

図 5: 三角方程式の解法

使用できるプロセッサ数を npe とし、各プロセッサを $P_1, P_2, \ldots, P_{npe}$ と書く。通信バッファとして用いる重複ローカル配列を $mem()$ とし、その大きさを $msize$ とする。$mem()$ はプロセッサ数 npe で均等に割り当てておく。各プロセッサに割り当てられるバッファの大きさは、$bsize = msize/npe$ 。

このルーチンのアルゴリズム (図 5(b)) を以下に示す。簡単のため、A は転置でなく下三角行列の場合を示す。

1. 次の操作をプロセッサ P_{lpe} $(1 \leq lpe \leq npe)$ で行う。

   - P_{lpe} に割り当てられているバッファ領域は
     $mem\Big((lpe-1)*bsize+1 : lpe*bsize\Big)$
     ここに P_{lpe} にある A の要素を列単位で書き込む。

   全体としては、行列 A の連続する $rows$ 列 (各プロセッサ $rows/npe$ 列ずつ) の要素が $mem()$ に書き込まれたことになる。

2. $mem()$ を UNIFY する。

3. 三重ループ
   [jループ] $j=1$ から N までのループ。
   各プロセッサは割り付けられている Bの要素に対して以下の操作を行う。

   [kループ] $k=1$ から $rows$ までのループ
   $b(k,j) = b(k,j)/a(k,k)$.

   [i ループ] $i=k+1$ から m まで、
   $b(i,j) = b(i,j) - b(k,j)a(i,k)$
   を計算する。$a()$ は、対応する $mem()$ の要素を用いる。

4. 1. から 3. までの操作を、A の要素すべてに対して行うまで繰り返す。

3. の操作において、jループは各プロセッサが並列に行うことができ、さらに i ループにおいてはベクトル化可能である。

4 評価

現在単体でテストできるのは行列積計算のルーチンだけであり、これに関しては次のような結果が出ている。10 プロセッサで、通信バッファに 8MB(各プロセッサが使用できるメモリサイズは 200MB) の領域を割り当て、10000×10000 行列 (800MB) の積を計算した場合、およそ 10Gflops の性能が確かめられた。なお、1 プロセッサのピーク性能は 1.6Gflops である。これは、列方向サイクリック分割を用いることによって、各プロセッサの持つベクトル性能が十分に発揮できることを示している。

5 おわりに

ScaLAPACK を VPP500 へ移植するにあたっての方針、検討事項、動いている部分の評価などについて述べた。現在、三角方程式の解法ルーチンの性能評価、LU 分解ルーチンのデバッグ等を行っている。当面の目標は、各プロセッサの持つベクトル性能をできるだけ下げないで、LU 分解ルーチンをはじめ主要なルーチンの VPP500 上での並列化を完成させることである。

参考文献

[1] L. S. Blackford, J. Choi, A. Cleary, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. *ScaLAPACK Users' Guide*, preliminary draft edition (1997).

[2] 富士通 (株). UXP/V Fortran90/VPP プログラミングハンドブック V10 用 (1997).

[3] 富士通 (株)HPC 事業本部編. PE 台数無依存並列手続きの作成方法 第 1 版 (1995).