データ並列型言語VPP Fortranによる線形計算ライブラリScaLAPACKの実現

浅岡香枝[†] 平野彰雄[†] 稲荷淳[††,☆]
岡部寿男[†††] 金澤正憲[†]

分散メモリ型並列計算機VPP上に，並列線形計算ライブラリScaLAPACKをデータ並列型言語VPP Fortranを用いて実現した．プロセッサ間通信にVPP Fortranのデータ転送機能を利用し，オリジナル版ScaLAPACKの機種依存の基本線形計算ライブラリPBLASおよびメッセージパッシングライブラリBLACSの機能をあわせ持つVPP-BLASを新たに開発した．VPP Fortranでは，全プロセッサが並列にブロードキャスト通信を行うunify転送が高速であることに着目し，unify転送に基づく行列積計算のアルゴリズムを設計した．行列積計算や密行列LU分解において，メーカから製品として提供されているMPIベースのScaLAPACKを凌ぐ高い並列性能が得られていることを確認した．

Implementing ScaLAPACK Linear Algebra Library in the Data Parallel Language VPP Fortran

Kae Asaoka,[†] Akio Hirano,[†] Kiyoshi Inari,[††,☆] Yasuo Okabe[†††] and Masanori Kanazawa[†]

We have implemented a data-parallel version of the ScaLAPACK parallel linear algebra library written in the VPP Fortran data parallel language on Fujitsu VPP distributed-memory parallel vector supercomputers. We have newly designed a basic linear algebra subroutines, called VPP-BLAS, instead of the PBLAS machine dependent basic linear algebra subroutines and the BLACS basic communication subroutines. The algorithms for matrix multiplication are especially designed and tuned so that the UNIFY high-speed data-transfer mode (where broadcast transfers are done by all processors simultaneously) is fully utilized. The performance of our implementation is better than that of the MPI-based code provided by the vender.

1. はじめに

分散メモリ型並列計算機におけるユーザのプログラミングの負担を軽減し，かつさまざまなアーキテクチャ間でのソフトウェアの流通を促すために，よく用いられるルーチンを集めた標準的な数値計算ライブラリの開発が重要視されている．行列積や連立一次方程式，固有値計算などの利用度の高い基本的な線形計算に対しては，たいていの商用並列計算機で高効率な並列アルゴリズムに基づく専用のルーチンが提供されているが，ユーザの負担の軽減のためには，絶対的な性能もさることながら，さまざまなプラットフォームにおいてインタフェースを共通化し互換性を高めることが必要である．本研究では，分散メモリ型並列計算機用線形計算ライブラリとして広く用いられ公開されているScaLAPACK[1]を，並列ベクトル計算機Fujitsu VPP[2]を対象とし，データ並列型言語VPP Fortran[3]を用いて実装した．

オリジナル版ScaLAPACKの各ルーチンの実行は，基本線形計算ライブラリPBLAS[4]とメッセージパッシングライブラリBLACS[5]を呼び出すことによって行われている．PBLAS，BLACSにおいてのみ機種依存の最適化を施すことにより，ScaLAPACK本体ルーチンは機種に依存しないままその高性能が得られるよ

† 京都大学大型計算機センター
Data Processing Center, Kyoto University
†† 京都大学工学研究科応用システム科学専攻
Department of Applied Systems Science, Kyoto University
☆ 現在，石川県庁
Presently with Ishikawa Prefectural Office
††† 京都大学情報学研究科
Graduate School of Informatics, Kyoto University

う設計されている．本研究で実装する VPP Fortran 版 ScaLAPACK（以下，VPP-ScaLAPACK と呼ぶ）においても，ScaLAPACK 本体ルーチンには並列化指示行を除いては手を加えず，機種に依存しないものとした．一方，プロセッサ間通信は，BLACS を用いずデータ並列型言語である VPP Fortran のデータ転送機能を利用することとし，PBLAS と BLACS の機能をあわせ持つ基本線形計算ライブラリ VPP-BLAS を VPP に特化して新たに開発した．

VPP Fortran では，データ並列型言語における通常の配列参照によるデータ転送機能のほかに，高速なデータ転送を行うためのいくつかの特別なデータ転送機能が提供されている．VPP では，このうち全プロセッサが並列にブロードキャスト通信を行う unify 転送が特に高速であることに着目し，unify 転送に基づく並列アルゴリズムを設計して VPP-BLAS において採用した．また，オリジナル版 ScaLAPACK では 2 次元ブロックサイクリック分割が採用されているが，VPP-ScaLAPACK においては 1 次元サイクリック分割を採用し，列方向のベクトル実行と行方向の並列実行の両方の並列性を生かせるようにした．

これらの方針に基づいて，VPP-ScaLAPACK を実装している．まずは，最も利用度の高い密行列連立一次方程式解法に必要なルーチン群を実現した．性能的には不利とされるデータ並列型言語を用いての実装でありながら，行列積や LU 分解に関しては，メーカから製品として提供されている MPI ベースの ScaLAPACK[6] や，メーカ独自のライブラリ SSL II/VPP[7] を凌ぐ高い並列性能が得られた．これにより，それほど性能が重視されない入力データの準備と出力データの処理の部分を平易なデータ並列型言語である VPP Fortran で記述し，性能が必要な部分で ScaLAPACK のルーチンを呼び出す形で，並列処理をほとんど意識しないプログラミングで並列計算機の高性能を享受することが可能となった．

以下，2 章では準備として線形計算ライブラリ ScaLAPACK の概略，対象コンピュータである VPP とデータ並列型言語 VPP Fortran の特徴について述べる．次いで 3 章では ScaLAPACK を VPP 上で実現するにあたっての検討事項について述べる．4 章では実装した LU 分解ルーチンで用いられるサブルーチン群のうち，行列積計算のアルゴリズムを示す．5 章で実装したルーチンの性能の実測結果について示す．

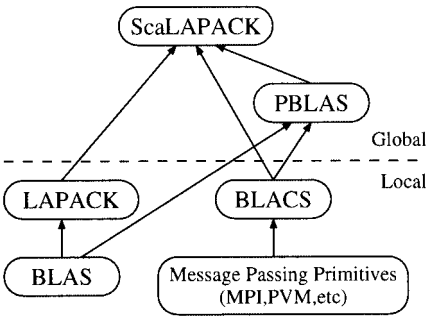


図 1 ScaLAPACK ソフトウェア階層
Fig. 1 ScaLAPACK software hierarchy.

2. 準　　備

2.1 ScaLAPACK

ScaLAPACK[1]（Scalable Linear Algebra PACKage）とは，分散メモリ型並列計算機やワークステーションクラスタを対象とした線形計算ライブラリである．これは，逐次計算機や共有メモリ型並列計算機を対象とした線形計算ライブラリ LAPACK[8]（Linear Algebra PACKage）の拡張として開発が行われている．ScaLAPACK や LAPACK は，Netlib（`http://www.netlib.org/`）において，ソースコードなどが公開されている．ScaLAPACK などには，連立一次方程式や線形最小二乗問題，固有値問題の解析などのためのルーチンが含まれている．

図 1 は，ScaLAPACK とその実行の際に用いられるソフトウェアの階層図を示している．点線より下，“Local” に分類されるルーチンは，1 つのプロセッサから呼び出され，また，変数は個々のプロセッサ上のもののみ参照および定義可能である．点線より上，“Global” に分類されているのは，複数プロセッサで並列処理を行うことができる並列ルーチンである．

基本線形計算サブルーチン群 BLAS[9]（Basic Linear Algebra Subprograms）には，行列とベクトルの乗算，行列どうしの乗算といった，一般的な線形計算を行うサブルーチンが含まれる．ハードウェアに依存した最適化はこの BLAS ルーチンで行い，BLAS ルーチンを呼び出す側（たとえば LAPACK ルーチン）では，機種に依存しないコードを書くことが可能である．

BLACS[5]（Basic Linear Algebra Communication Subprograms）は，線形計算用に作られたメッセージパッシングライブラリである．BLACS には，PVM や MPI を用いた実装が提供されているほか，計算機固有の通信プリミティブを用いた実装も可能になっている．PBLAS[4]（Parallel BLAS）は BLAS の並列版で

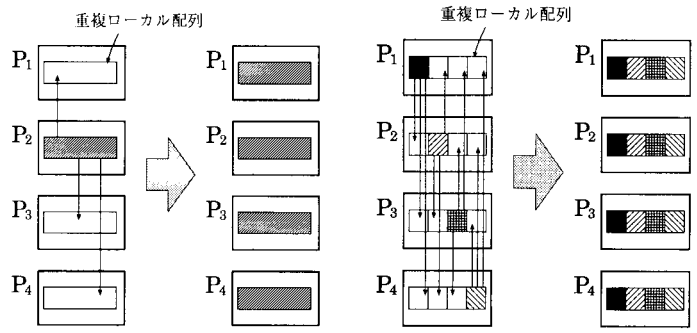


図 2 ブロードキャストと unify 転送
Fig. 2 Broadcast and unify data transfer.

ある．基本線形計算を並列に行うルーチンを持つ．その際，BLACS ルーチンを呼び出すことによってプロセッサ間のメッセージ通信を行う．ScaLAPACK ではハードウェアに依存した最適化は PBLAS ルーチンと BLACS ルーチンで行い，呼び出す側（ScaLAPACK ルーチン）は機種に依存しない．

2.2 VPP システム

富士通 VPP システム[2] は分散メモリ型並列計算機であり，複数台のプロセッサ（プロセッシングエレメント；PE）を要素とし，各プロセッサ間がクロスバネットワークで接続される構造になっている．各プロセッサは，ベクトルユニットおよびスカラユニットを有する CPU と主記憶から構成されている．また各プロセッサは，クロスバネットワークを通して他のプロセッサと通信することができる．

2.3 VPP Fortran

VPP Fortran[3] は，Fortran90 に並列処理を行うためのいくつかの機能を付加した，HPF[10]（High Performance Fortran）に類似のデータ並列型プログラミング言語である．VPP Fortran の詳細については文献 11)~14) に譲るが，ここでは次節以降でアルゴリズム設計の際に検討の対象となるデータの割付けおよびデータ転送について説明する．

VPP Fortran で使用できるデータは，各プロセッサのローカル空間に重複して割り付けられる**重複ローカル変数/配列**，各プロセッサのローカル空間に分割して割り付けられる**分割ローカル配列**，全プロセッサからアクセス可能な仮想グローバル空間に割り付けられる**グローバル変数/配列**に分類される．分割ローカル配列は，並列実行部分においてはそのプロセッサに割り付けられている配列要素だけがアクセス可能である．一方，グローバル配列はつねに全プロセッサから全データがアクセス可能であるが，アクセス速度が遅く，また配列参照の場合においてもベクトル実行の対象とならない．

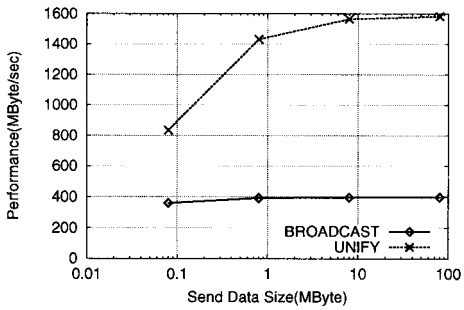


図 3 ブロードキャストと unify 転送におけるデータ受信速度
Fig. 3 Speed of receiving data in broadcast and unify data transfer.

重複ローカル配列に対しては，**ブロードキャスト**および **unify 転送**と呼ばれる 2 つの高速なデータ転送モードが提供されている．ブロードキャスト（BROADCAST 文による）では，あるプロセッサ上の重複ローカル配列の値をすべてのプロセッサに転送する．これに対し unify 転送（UNIFY 文による）では，各プロセッサに割り当てられた重複ローカル配列の一部分を全プロセッサが並列にブロードキャストする．ブロードキャストおよび unify 転送の振舞いを**図 2** に示す．

図 3 に，VPP800 10 PE におけるブロードキャストおよび unify 転送での 1 PE あたりのデータ受信速度の実測結果のグラフを示す．unify 転送の方がブロードキャストに比べて 10 PE において約 4 倍速く☆，1 MB 以上のデータの unify 転送では，VPP800 のプロセッ

☆ VPP Fortran におけるブロードキャスト転送の速度は PE 数の 2 を底とする対数に反比例する[11]．

サ間通信の公称の最大性能値である 1.6 GB/s にほぼ等しい性能が得られる．

3. ScaLAPACK の実現に関する検討項目

3.1 ScaLAPACK のルーチンの構成

ScaLAPACK を VPP 上に実現する方法としてまず考えられるのは，VPP 上で提供されている PVM や MPI の上に BLACS を介して ScaLAPACK をそのまま動作させる方法であり，実際，メーカ提供の製品版 ScaLAPACK は MPI 上で実現されている．しかし，現状 VPP 上のデータ並列型言語である VPP Fortran および HPF は MPI などと混在できないため，MPI 上に実現された ScaLAPACK に対しては，ScaLAPACK に対する入力データを準備する部分，および出力結果を行うべき処理の部分を，ユーザが MPI を用いて記述する必要が生じる．これでは線形計算ライブラリを提供することにより並列プログラミングが不要になった効果が半減する．そこで，本研究では，PVM や MPI などのメッセージパッシング系の通信ライブラリを用いず，データ並列言語 VPP Fortran のデータ転送機能を用いて ScaLAPACK を実現するという方針をとった．

これに基づき ScaLAPACK の各ルーチンについて以下のような基本方針を立てた（**図 4**）．まず，ScaLAPACK 本体ルーチンには並列化指示行☆の挿入を除き手を加えないものとする．これはオリジナル版 ScaLAPACK の「ScaLAPACK 本体ルーチンは機種に依存しない」という精神に基づいている．ScaLAPACK 本体ルーチンは，VPP Fortran で書かれたメインプログラム中の「パラレルリージョン」内で並列に呼び出される．一方，PBLAS に相当する VPP-BLAS を独自に開発し，VPP に特化した最適化は，この部分にのみ適用する．VPP-BLAS では BLACS ルーチンは用いず，BLACS が行うべきプロセッサの管理などは VPP Fortran に任せ，プロセッサ間の通信は VPP-BLAS ルーチン内で VPP Fortran のデータ転送機能を用いて行う．

3.2 プロセッサ間通信

VPP Fortran はデータ並列型言語であり，2.3 節で述べたグローバル配列を用いれば，プロセッサ間通信を意識しないようなプログラミングが原理的には可能である．しかしながら，現在の VPP Fortran の実装では，グローバル配列に対する参照が連続すると粒度の細かい通信が頻発し，通信のオーバヘッドが大きくなって，十分な性能が得られない．

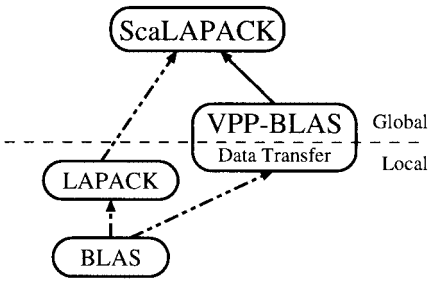


図 4 VPP 版 ScaLAPACK ソフトウェア階層
Fig. 4 VPP-ScaLAPACK software hierarchy.

そこで，ScaLAPACK 内では，プロセッサ間通信を，可能な限り高速なデータ転送モードである unify 転送によって行うことにした．すなわち VPP-BLAS ルーチン内において通信用のバッファとして重複ローカル配列を確保しておき，各プロセッサがバッファの割り当てられた部分にデータを書き込んだ後 unify 転送することでプロセッサ間の通信を行う．

3.3 配列の分割

行列計算では，さまざまな状況下での最適化や負荷均衡といった観点から，2 次元ブロックサイクリック分割が好都合である場合が多く，MPP（Massively Parallel Processor）システムを念頭においたオリジナル版 ScaLAPACK でも 2 次元ブロックサイクリック分割を採用している．しかし，VPP は各プロセッサがベクトルユニットを持っており，2 次元の分割を行うとベクトル長が短くなり，ベクトル性能を十分発揮できないおそれがある．本研究では，VPP のベクトル実行による並列性とマルチプロセッサによる並列性をともに生かすために，1 次元の分割を採用した．VPP Fortran ではブロックサイクリック分割の機能が提供されていないため，行列の列方向次元に沿った 1 次元サイクリック分割を採用した．

4. アルゴリズム

LU 分解に必要なルーチン群を**表 1** に示す．これらのルーチンのうち VPP-BLAS に属するものを，前章で述べた方針に基づいて新たに設計し，実装した．この章では，LU 分解の性能に最も影響を与える行列積計算ルーチンのアルゴリズムについて説明する．

行列積計算ルーチン（PDGEMM）では，$C = C + A \times B$ の計算を行う．A は $M \times L$ 行列，B は $L \times N$ 行列，C は $M \times N$ 行列であるとする．各行列において，各次元の添字が 1 から始まるものとし，$a(1{:}M, 1{:}L)$,

☆ プロセッサグループ大きさ引継ぎ手続きを表す「`!xocl subprocessor p(:)`」の 1 行のみ．

表 1 LU 分解に必要なルーチン一覧
Table 1 Routines used in LU factorization.

ルーチン名	所属	機能
PDGETRF	ScaLAPACK	ブロック化 LU 分解を行う
PDGETF2	ScaLAPACK	部分行列で LU 分解を適用
PDLASWP	ScaLAPACK	複数のベクトルを交換
PDGEMM	VPP-BLAS	行列積を計算
PDTRSM	VPP-BLAS	三角方程式を解く
PDGER	VPP-BLAS	ランク 1 の行列更新
PDSWAP	VPP-BLAS	ベクトルを交換
PDSCAL	VPP-BLAS	ベクトルをスカラ倍
PDAMAX	VPP-BLAS	絶対値最大の値とその位置

表 2 行列積のパターン
Table 2 Patterns in matrix multiplication.

番号	A	B	C，B の分割
1	転置でない	転置でない	揃っている
2	転置でない	転置でない	揃っていない
3	転置	転置でない	揃っている
4	転置	転置でない	揃っていない
5	転置でない	転置	
6	転置	転置	

$b(1{:}L, 1{:}N)$，$c(1{:}M, 1{:}N)$ のように宣言されているとする．行列 A，B は転置行列であることが許されている．たとえば A，B いずれも転置でない場合，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になる．さらに，PDGEMM では引数として部分配列を受け取ることが許されている．すなわち A，B，C はいずれもサイクリック分割であるが，それらの第 1 列が同一のプロセッサ上にあるとは限らない．以下，2 つの行列の第 1 列が同一のプロセッサ上に割り付けられている場合を，分割が揃っている，と呼ぶことにする．

上記の場合分けに対応して，**表 2** に示す 6 パターンに分類してアルゴリズムを設計した．設計にあたっては，3.2 節で述べた設計方針に従いプロセッサ間通信をすべて unify 転送を用いるようにした．以下，最も簡単な場合であるパターン 1 について説明する．

行列 C の (i, j) 要素の計算は，

$$c(i,j) = c(i,j) + \sum_{k=1}^{L} a(i,k)\, b(k,j)$$

と表される．この計算には，行列 A の第 i 行要素と行列 B の第 j 列要素が必要となる．行列 B と C の分割が揃っている場合を考えているので，$b(k, j)$ は $c(i, j)$ と同一のプロセッサ上にあり，参照に通信の必要はない．

ここで A の第 1 列の要素 $a(i, 1)$ をブロードキャストしたとすると，$a(i, 1)$ が全プロセッサで参照可能になり，すべての j について $c(i, j) = c(i, j) + a(i, 1) * b(1, j)$ の計算ができる．これを同様に $a(i, 2)$ 以下すべての列の要素に対し行えば C の第 i 行 $c(i, j)$，$j = 1, \ldots, N$ の計算が完了する．$a(i, k)$ の代わりに A の 1 列 $a(1{:}M, k)$ をまとめてブロードキャストしたとすると，C のすべての列に対して $c(1{:}M, j) = c(1{:}M, j) + a(1{:}M, k) * b(k, j)$，$j = 1, \ldots, N$ の更新操作が可能となる（$1 \le k \le L$）．実際には，列のブロードキャストを unify 転送で行うために，A の左側の列から順に，通信用のバッファに載せられる限り複数列まとめ，unify 転送する．転送後，通信バッファ上の A のコピーと対応する B の部分を用いて C を更新する（**図 5**）．

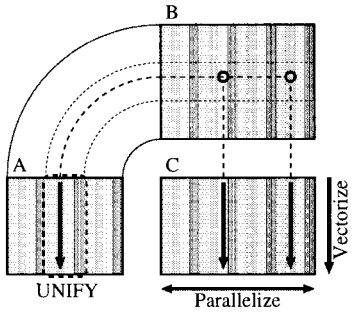


図 5 行列積アルゴリズム（B と C の分割が揃っている場合）
Fig. 5 The algorithm for matrix multiplication (B and C are aligned).

```
      do KO=0, L-1, KWIDTH
!xocl   spread do /(p, index=KO+1:KO+KWIDTH, part=cyclic)
        do K=KO+1, KO+KWIDTH
          K1=K-KO-1
          mem( 1 + K1*M : M + K1*M ) = a ( 1:M , K )
        enddo
!xocl   end spread
!xocl   barrier
!xocl   unify ( mem(/(p, index=1:M*KWIDTH, part=cyclic ) ) )(TID)
!xocl   movewait(TID)
!xocl   spread do /(p, index=1:N, part=cyclic)
        do J = 1, N
          do K=KO+1, KO+KWIDTH
            K1=K-KO-1
            do I = 1, M
             c( I, J ) = c( I, J ) + mem( I + K1*M ) * b( K, J )
            end do
          end do
        end do
!xocl   end spread
      end do
```

図 6 行列積計算の VPP Fortran コードの概略
Fig. 6 Outline of the VPP Fortran codes for matrix multiplication.

このアルゴリズムを VPP Fortran で記述したコードの概略を，**図 6** に示す．配列 mem は通信用のバッファであり，変数 KWIDTH は，通信バッファに一度に載せられる A の列の最大数を示す．前半の K のループで，図 5 の A 上の破線枠に対応する配列 a の部分を，それぞれのプロセッサが当該プロセッサ上に割り付けられている要素について通信バッファにコ

ピーする．次に通信バッファの内容を unify 転送で全プロセッサに送る．これで，図の破線枠の部分が，通信バッファ上のデータとして全プロセッサから直接参照可能となったことになり，C の更新が通信なしで計算できる（図の J のループ）．

以上のように，行列が一次元サイクリック分割されていることにより，行列積計算において必要となる部分行列のブロードキャスト型転送を unify 転送により行うことができる．B と C の分割が揃っていない場合や，B が転置されている場合には，A の転送の前に B の転送を行っておく必要があるが，これも unify 転送で行える[15]．VPP-BLAS の実装においては，行列積以外のほとんどのルーチンにおいても，必要となるすべての通信パターンを効率の良い unify 転送のみを用いて実装した．

5. 評価と考察

実装した VPP-ScaLAPACK において行列積ルーチンおよび LU 分解ルーチンについて速度測定を行った．測定に用いたシステムは，京都大学大型計算機センターの VPP800/63 システムで，コンパイラは，Fujitsu Fortran90/VPP Compiler L99051 である．京都大学の VPP800 システムは 63 PE を有し，通常運用時は 1 並列ジョブあたり最大 40 PE まで利用できる．1 PE あたり 8 GB の主記憶（うちユーザ領域は 7 GB）を有し，理論最大性能は 1 PE あたり 8 Gflops，40 PE を用いると 320 Gflops である．

5.1 行列積ルーチンの単体テスト結果

行列積計算ルーチンで，$N \times N$ の正方行列の行列積 $C = C + A \times B$ について測定を行った．転置の有無と C と B の分割の状態から考えられるすべてのパターン（表 2 参照）について測定を行った．

測定は PE 10 台で行った．10 台の理論最大性能は，80 Gflops である．また通信バッファには 32 MB を用意した．この大きさは使用可能メモリ 7 GB の 0.5%未満であり，プログラミング上問題のない大きさであると思われる．

測定結果を**図 7** に示す．図 7 のグラフ番号 1 から 6 は表 2 の番号に対応している．グラフの横軸は N の大きさを表し，縦軸は得られた計算時間から演算量を $2N^3$ として求めた Gflops 値を表す．

図 7 に示すように，1 から 6 の全パターンにおいて $N = 5{,}000$ で理論最大性能の 90%以上の性能が得られ，$N = 20{,}000$ ではパターン 1 および 3 が 77.7 Gflops（理論最大性能比 97.1%），他の場合でも 76.7 Gflops（理論最大性能比 95.8%）以上の性能が得られている．

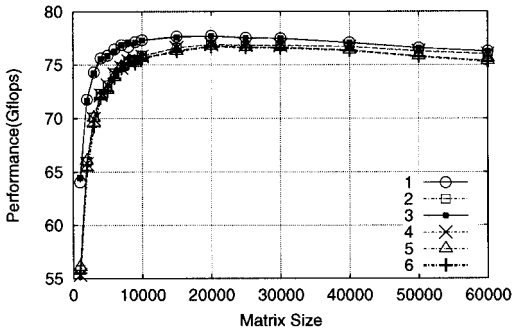


図 7 行列積性能測定結果
Fig. 7 Performance of matrix multiplication.

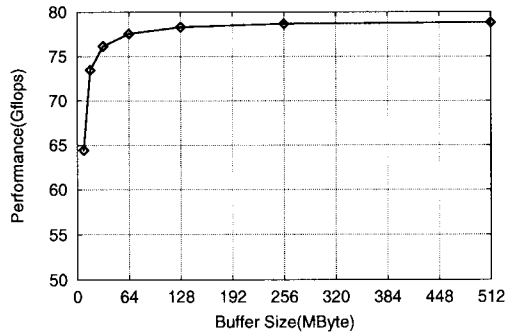


図 8 通信バッファ容量と行列積性能の関係
Fig. 8 Performance of matrix multiplication with respect to the size of communication buffer.

また，バッファの容量を 8 MB から 512 MB と変化させて，プロセッサ数 10 で，65660×65660 の行列積の場合で測定した結果を，**図 8** に示す．バッファ容量が 8 MB の場合，1 回の unify 転送で転送されるのは各プロセッサあたり 1 列ずつであり，十分な性能が得られていないが，バッファ容量が 32 MB の場合，つまり 1 回の転送に 1 PE あたり 6 列程度の転送が行われれば，バッファがより大きい場合に比べて遜色ない性能が得られることが分かる．

比較のために，メーカから製品として提供されている MPI 版 ScaLAPACK[6]（UXP/V ScaLAPACK V20L10 L00000）および VPP Fortran 用の数値計算ライブラリである SSL II/VPP[7]（UXP/V SSL II/VPP V20L10 L00000）についても行列積計算の性能測定を行った．行列が転置でなく分割が揃っている場合を対象とした．結果を**表 3** に示す．測定は，40 PE を用いて $N = 100{,}000$ について行った．VPP-ScaLAPACK では通信バッファ 512 MB とし，MPI

表 3 行列積計算の性能比較
Table 3 Comparison of the performance of matrix multiplication.

ライブラリ	性能（Gflops）	理論最大性能比
VPP-ScaLAPACK	313.9	98.0%
MPI 版 ScaLAPACK	256.9	80.3%
SSL II/VPP	296.3	92.6%

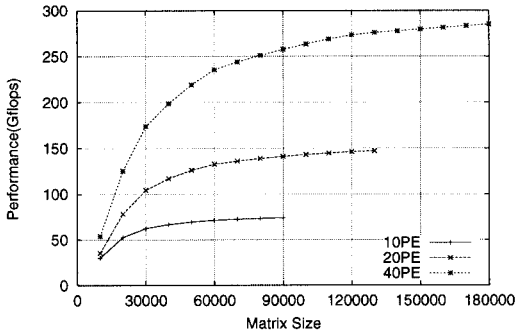


図 9 LU 分解性能測定結果
Fig. 9 Performance of LU factorization.

版 ScaLAPACK のプロセッサ配置は 5×8 とした結果である．我々の unify 転送をベースとする行列積ルーチンの実装はほぼハードウェアの限界性能を引き出しており，既存の製品レベルのライブラリと比べてもかなり優れている．

5.2 LU 分解ルーチンの速度測定結果

LU 分解ルーチン PDGETRF 全体としての速度測定を行った．対象とする行列を転置ではない $M \times M$ の正方行列とし，M を 10,000 から 180,000 まで変化させた．プロセッサ数は 10 PE（ブロックサイズ[8]NB=400），20 PE（NB=400），40 PE（NB=2,000）と変化させ，通信用バッファとして 512 MB の重複ローカル配列を用意した．測定結果を**図 9** に示す．40 PE においては $N = 180,000$ で 285.1 Gflops（理論最大性能比 89.1%）の性能が得られている．

メーカから提供されているアナライザであるサンプラを用いて，LU 分解ルーチン実行時の各ルーチンにおける演算時間の割合を調べたところでは，180,000×180,000 行列の LU 分解を 40 PE で計算した場合において，全体の演算時間に対し，行列積計算ルーチン PDGEMM で 89.5%，ランク 1 の行列更新ルーチン PDGER で 6.7%，三角方程式の解法ルーチン PDTRSM で 3.3%，最大値とその場所を求めるルーチン PDAMAX で 0.4%の時間がかかっている．PDGER は，LU 分解の 1 列ごとの進行ごとに 1 回呼び出されており，我々の実装では毎回ほとんど同じ領域が unify 転送されている．ScaLAPACK 本体ルーチンの修正がともなうため今回の実装では採用しなかったが，この冗長な転送を避けるように工夫すればさらなる高速化が期待できる．

表 4 LU 分解の性能比較
Table 4 Comparison of the performance of LU factorization.

（単位は Gflops）

ライブラリ	$N = 130,000$	$N = 180,000$
VPP-ScaLAPACK	277.4 (86.7%)	286.3 (89.5%)
MPI 版 ScaLAPACK	278.1 (86.9%)	281.1 (87.9%)
SSL II/VPP	258.8 (80.9%)	——

MPI 版 ScaLAPACK および SSL II/VPP についても LU 分解の性能を測定し比較を行った結果を，**表 4** に示す．測定は，40 PE を用いて $N = 130,000$, $N = 180,000$ について行い，VPP- ScaLAPACK では通信バッファ 512 MB，NB=1,000 とし，MPI 版 ScaLAPACK のプロセッサ配置は 5×8，NB=64，SSL II/VPP では NB=50 とした結果である．VPP-ScaLAPACK は $N = 180,000$（40 PE，PE あたりの主記憶 7 GB において，主記憶に格納できる行列のほぼ最大サイズ）において最も高い性能を得ている．なお，SSL II/VPP では，LU 分解の対象となる行列と同サイズの作業領域を別途必要とするため，$N = 180,000$ は計算できなかった．VPP-ScaLAPACK は，対象行列のサイズの十分の 1 以下の通信バッファで十分な性能を得ており，かつ必要に応じて通信バッファの容量をさらに小さくすることもできる点でも，SSL II/VPP より優れている．

6. おわりに

本論文では，データ並列型言語による線形計算ライブラリの実現について考察し，分散メモリ型並列計算機用線形計算ライブラリ ScaLAPACK の VPP Fortran による実装について述べた．負荷分散や VPP の各プロセッサが持つベクトルユニットの性能，VPP Fortran に用意されている unify 転送の高速性を生かすよう，配列の分割として 1 次元サイクリック分割を採用し，それらを前提に行列積などの基本演算の高速なアルゴリズムを設計した．さらに，LU 分解に必要なルーチン群の実装を行い，その性能を評価した．京都大学大型計算機センターの VPP800 上で速度を測定した結果，40 PE を用いた場合で，行列積計算ルーチンで理論最大性能の 98%，LU 分解ルーチンで 90%という高い性能が得られていることが確認できた．

VPP-ScaLAPACK のような高性能の数値計算ライ

ブラリが提供されるという前提のもとでは，ユーザが明示的に行う並列プログラミングは，比較的性能が重視されない初期データの準備や計算結果の出力処理が中心になる．メッセージパッシングライブラリに比べてユーザ親和性の高いデータ並列型言語である VPP Fortran でこの部分のコーディングが行えることが，MPI をベースにした ScaLAPACK に対する優位点である．

一般に，データ並列型言語による実装はメッセージパッシングライブラリを用いた実装に比べて性能的に劣ることが多いといわれる．それに対し本研究では，データ並列型言語において，通常の配列参照によるプロセッサ間通信とは別に，VPP Fortran の unify 転送のような高速な転送モードが提供されていれば，それを計算の核となる部分で利用することで，メッセージパッシングライブラリを用いた実装と同等以上の高性能が達成できることを明らかにした．VPP Fortran における unify 転送は HPF においても HPF/JA 拡張[16] におけるフル SHADOW 属性を持つ分散された配列の REFLECT 操作として表現できることから，VPP 上の HPF による同様の実装も検討中である．

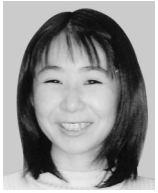
今後の課題としては，QR 分解やコレスキー分解ルーチンなどを実装し VPP-ScaLAPACK を完成させることがあげられる．行列積計算ルーチン，三角方程式の解法ルーチンなど主要な基本線形計算サブルーチンは LU 分解と共通であるので，他のルーチンの作成は比較的容易に行えると考えられる．また，実装したルーチン群のソースコードは公開しているが[17]，ドキュメントの整備などさらに充実させていきたい．

謝辞 ご討論いただいた本学大型計算機センター研究開発部の諸氏に深謝する．

参 考 文 献

1) Blackford, L.S., Choi, J., Cleary, A., Demmel, J., Dhillon, I., Dongarra, J., Hammarling, S., Henry, G., Petitet, A., Stanley, K., Walker, D. and Whaley, R.C.: *ScaLAPACK Users' Guide*, preliminary draft edition (1997).
2) Utsumi, T., Ikeda, M. and Takamura, M.: Architecture of the VPP 500 Parallel Supercomputer, *Proc. Supercomputing '94*, pp.478–486 (1994).
3) 富士通：UXP/V Fortran/VPP 使用手引書 V20 用，J2U5-0430-01 (1999).
4) Choi, J., Dongarra, J., Ostrouchov, S., Petitet, A., Walker, D. and Whaley, R.C.（著），安斎宏幸，川守田和男，小松崎浩司，酒井健作，村瀬芳生，萩原俊彦，石田栄美，長谷川秀彦（訳）：並列線形代数基本サブプログラム PBLAS の提案，LAPACK Working Note #100 (1996).
5) Dongarra, J.J. and Whaley, R.C.: *A User's Guide to the BLACS v1.1*, LAPACK Working Note #94 (1997).
6) 富士通：UXP/V BLAS/VP LAPACK/VP ScaLAPACK 使用手引書 V20 用，J2U5-0480-01 (1999).
7) 富士通：SSLII/VPP 使用手引書，J2X0-1372-01 (1998).
8) Anderson, E., Bai, Z., Bischof, C., Demmel, J., Dongarra, J., Croz, J.D., Dreenbaum, A., Hammarling, S., McKenney, A., Ostrouchov, S. and Sorensen, D.（著），小国 力（訳）：行列計算パッケージ LAPACK 利用の手引，丸善 (1995).
9) Dongarra, J.J., Duff, I.S., Sorensen, D.C. and van derVorst, H.A.（著），小国 力（訳）：コンピュータによる連立一次方程式の解法，丸善 (1993).
10) High Performance Fortran Forum: High Performance Fortran Language Specification Version 2.0 (1996).
11) 岡田 信，坂本喜則，浅井 登：VPP500 システムのソフトウェア，電子情報通信学会論文誌，Vol.J78-D-I, No.2, pp.149–161 (1995).
12) 岩下英俊：HPF からみた VPP Fortran，情報処理，Vol.38, No.2, pp.114–120 (1997).
13) 平野彰雄：並列プログラミング入門，京都大学大型計算機センター広報，Vol.28, No.3, pp.116–136 (1995).
14) 永井 亨：VPP Fortran 入門，名古屋大学大型計算機センターニュース，Vol.26, No.4, pp.315–339 (1995).
15) 稲荷 淳，岡部寿男，金澤正憲：並列ベクトル計算機における線形計算ライブラリ ScaLAPACK の実現，並列処理シンポジウム JSPP'99, pp.183–190 (1999).
16) Japan Association for High Performance Fortran: HPF/JA 言語仕様書 Version 1.0 (1999).
17) `http://www.kudpc.kyoto-u.ac.jp/HPC-WG/`.

(平成 11 年 8 月 30 日受付)
(平成 12 年 2 月 4 日採録)



浅岡 香枝（正会員）

1974 年生．1997 年名古屋工業大学電気情報工学科卒業．同年京都大学大型計算機センター技官，現在に至る．ベクトル計算機，並列計算機に興味を持っている．

平野 彰雄（正会員）

1953 年生．1977 年京都工芸繊維大学工業短期大学部電気工学科卒業．1974 年京都大学大型計算機センター技官．大学間ネットワーク，システム運用管理システム等の開発に従事．コンピュータネットワーク，分散処理システム，並列プログラミングに興味を持っている．

稲荷 淳

1996 年京都大学工学部情報工学教室卒業．1998 年同大学大学院工学研究科応用システム科学専攻修士課程修了．現在石川県庁勤務．在学中，スーパーコンピュータ用基本ソフトウェアの研究開発に従事．

岡部 寿男（正会員）

1964 年生．1988 年京都大学大学院工学研究科情報工学専攻修士課程修了．同年同大学工学部助手．同大学大型計算機センター助教授を経て，現在同大学大学院情報学研究科助教授．博士（工学）．並列アルゴリズム等の研究に従事．電子情報通信学会，システム制御情報学会，日本ソフトウェア科学会，IEEE，ACM，EATCS 各会員．

金澤 正憲（正会員）

1946 年生．1971 年京都大学大学院工学研究科数理工学専攻修士課程修了．1972 年同大学大型計算機センター助手．同助教授を経て，1995 年同教授，現在に至る．工学博士．計算機システム・オペレーティングシステムの方式と性能評価，およびコンピュータネットワーク・分散システムに興味を持っている．電子情報通信学会，ACM，日本応用数理学会各会員．