LINPACKベンチマークの並列ベクトル処理

——並列計算機AP1000用数値演算アクセラレータによる実現——

上村和人[†] 清水俊幸[††]
石畑宏明[†] 堀江健志[††]

LINPACKベンチマークに並列化とベクトル化を施し，NCA付きAP 1000を用いて性能を評価した．LINPACKの並列化では負荷が均等でかつデータパラレル実行が可能となるようにデータを分散させ，ブロッキングを施すことによって行列積演算や外積演算といったベクトル化が容易でかつ高い性能が得られる演算に帰結させた．理論ピーク性能に対して，単体実行時で61%，並列実行時で47%の性能が得られた．また，1000×1000の行列を解く場合，NCAを付加した16セルで，NCAを付加しない128セルのAP 1000とほぼ同等の性能が得られた．

Parallel Vector Processing of the LINPACK Benchmark on the AP1000 with Numerical Computation Accelerators

KAZUTO KAMIMURA,[†] TOSHIYUKI SHIMIZU,[††] HIROAKI ISHIHATA [†] and TAKESHI HORIE [††]

This paper evaluates the performance of the LINPACK benchmark on the AP1000 with the Numerical Computation Accelerator (NCA) option. When parallelizing the program, matrix data were distributed among cells for load balancing and data-parallel execution. The blocking technique was applied to the code in order to utilize matrix multiplication and outer product routines, which were easily vectorized and could attain high performance on NCA. We obtained 61% of the theoretical peak performance for single-cell execution, and 47% for multiple-cell execution. A 16-cell configuration with NCA can sustain a performance which is comparable to a 128-cell configuration without NCA for a 1000×1000 matrix.

1. はじめに

計算機の性能が向上するにつれて，大規模な問題の高速な処理が可能になってきている．同時に，扱う問題の規模がさらに大きくなってきており，計算機には一層の処理能力の向上が要求され続けている．大規模な問題の一つの例である自然科学分野でのシミュレーションのように，膨大な演算量をもつ科学技術計算を高速に処理する方法には，ベクトル処理と並列処理の2つの方法がある．

並列処理あるいはベクトル処理のどちらか一方だけの適用では，処理速度の向上には限界がある．高性能の妥当なコストでの実現，あるいは，超高性能の実現のためには，並列処理とベクトル処理の両者をバランス良く利用する必要がある．大規模な科学技術計算では，十分大きなベクトル演算，行列演算を個々のプロセッサ内で処理するような並列化を施せる．この場合，並列処理とベクトル処理を組み合わせて処理時間の短縮を図ることは有効だと考えられる．

NCA（Numerical Computation Accelerator）は，並列計算機AP 1000の並列処理要素（セル）に，ベクトル処理機能を付加するためのオプションである．このオプションを使うことで，AP 1000のセルの計算能力を高めることができる．

本稿では，ベクトル化と並列化が相互に及ぼす影響を，LINPACKベンチマーク*,[2]を通して検討する．2章でNCAの構成を述べる．3章でLINPACKの並列化とベクトル化について述べ，最適化とその効果につ

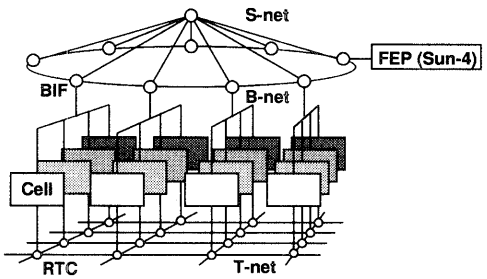
† 富士通（株）
Fujitsu Ltd.
†† （株）富士通研究所
Fujitsu Laboratories Ltd.

* 特異解を持たないn元連立1次方程式を部分ピボット選択つきガウスの消去法で解くプログラム．本論文では，文献2)のTPP（Toward Peak Performance）規定に従った性能（n=1000の場合）と，nを変化させた場合の性能を議論する．

いて検討する．4章で結果の考察と性能予測を行い，並列化によるオーバヘッドの割合の変化や，大規模システムでの性能について議論する．最後に本稿のまとめと今後の課題を述べる．

2. システム構成

AP 1000はセルごとにメモリをもつメッセージパッシング型並列計算機である[4]．図1にその全体構成を示す．通常のセルは，スカラー演算器であるSPARCプロセッサを持ち，1024台までネットワークで二次元トーラスに結合できる．ネットワークを構成するRTCには，列方向，行方向のセルにブロードキャストを行う機能がある．NCA[5]はセルにベクトル処理機構（μVP[6]）を付加するオプションで，それぞれのセルがもつ内部バス(LBUS)を介して接続する．図2にNCAを接続した時のセルの構成を示す．



Cell ：プロセッシングエレメント **T-net**：トーラスネットワーク
BIF ：**B-net** インターフェース **B-net**：放送ネットワーク
RTC：ルーティングコントローラ **S-net**：同期ネットワーク

図1 AP 1000の構成
Fig. 1 Configuration of the AP1000.

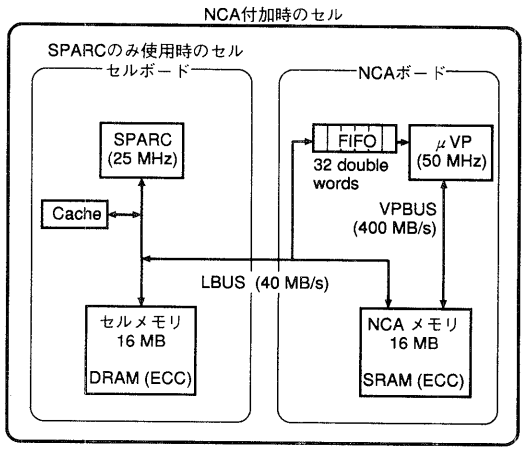


図2 NCAを付加したセルの構成
Fig. 2 Configuration of a cell with NCA.

2.1 NCA

NCAは16 Mバイトのメモリ(NCAメモリ)，ベクトル演算器μVP，それらをインタフェースするゲートアレイで構成されている．NCAメモリはSPARCとμVPの双方からアクセス可能である．セルボード上の16 Mバイトのメモリ（セルメモリ）はμVPからのアクセスはできない．μVPはNCAメモリ上のデータに対してのみ演算を行える．

SPARC，μVP間はダブルワード（64ビット）32段のコマンドFIFOによってインタフェースされている．この機構により，μVPがプログラムを実行中でも，SPARCからμVPへの命令の発行が可能となり，処理のオーバーラップとμVPのスタートアップ時間の短縮が図れる．

2.2 μVP

μVPは1チップのベクトル演算器で，8 Kバイトのベクトルレジスタ，128バイトのスカラレジスタ，そしてADD, MLT, DIV, LOAD/STOREの4本のベクトルパイプラインで構成されている．それぞれのパイプラインの理論ピーク性能は，ADDおよびMLTパイプラインは倍精度で50 MFLOPS，単精度で100 MFLOPS, DIVパイプラインは倍精度，単精度ともに6 MFLOPSである．これらのパイプラインは並列に演算を実行できるので，μVPの理論ピーク性能は，倍精度で106 MFLOPS，単精度で206 MFLOPSとなる．

3. LINPACKの実装

LINPACKベンチマークプログラムは，特異解を持たないn元連立1次方程式

$$Ax=b \qquad (1)$$

を，部分ピボット選択つきガウスの消去法によって解くものである．ここに，Aは$n\times n$の係数行列，xおよびbはそれぞれ要素数nの未知ベクトルおよび定数ベクトルである．

本プログラムにはガウスの消去法と等価な処理であるLU分解法（外積形式）を用いた．処理はLU分解部と後退代入部に分かれる．時間測定はベンチマークのTPP（Toward Peak Performance）規定に従い，LU分解部および後退代入部を合わせて評価した．

LU分解部では，行列Aを上三角行列Uと対角要素が1の下三角行列Lに変換する．これは，Pを並べ換え行列とすると，

$$PA=LU$$

となる行列LとUを求めることである．LU分解を拡大行列$\bar{A}=[A|b]$に対して適用すると，

$$P\bar{A}=L\bar{U},\ \bar{U}=[U|\bar{b}]$$

が得られ，式（1）は

$$Ux=\bar{b}$$

に変換される．

後退代入部では，$Ux=\bar{b}$ から x を求める．

プログラミング言語は，SPARC による処理部分には C を，NCA による処理部分には μVP の命令列を記述するアセンブラを用いた．

3.1 並列化

行列 A の i 行 j 列要素を a_{ij} で表すことにする．要素番号は 0 から付けることにする．N_x, N_y を AP 1000 の x 方向，y 方向のセル数，P_x, P_y を x 方向および y 方向のセル番号とする．$P_x=0, 1, \cdots, N_x-1, P_y=0, 1, \cdots, N_y-1$ である．

プログラムの並列化は，$N_x=N_y$ とし，$N_x \times N_x$ の正方形状に配置されたセルを想定して行った．また，問題の大きさ n は N_x の倍数とした．

拡大行列 $[A|b]$ の分散には二次元サイクリック分割を用いた．つまり，係数行列の要素 a_{ij} は $(P_x, P_y)=(j \bmod N_x, i \bmod N_y)$ で表されるセルに，ローカルなインデクス $(\lfloor i/N_x \rfloor, \lfloor j/N_x \rfloor)$ で格納する．定数ベクトル b は，大きさが $n \times N_x$ の行列 B に拡張してから同様に分散する．行列 B の第 1 列は b に等しく，他の要素は 0 であり，$P_x=0$ のセルのみが定数ベクトル b の要素を持つ．

演算の大部分を外積または行列積演算で行わせて処理の効率化を図るために，複数の行，列をブロック化して処理する．まとめる行数および列数をブロッキングファクタと呼び，記号 ω で表すことにする．分散された行列を再合成して得られるグローバルな行列に対しては，1 ブロック中の行数および列数は ωN_x となる．

3.1.1 LU 分解部のアルゴリズム

図 3 は，拡大行列 $[A|b]$ をグローバルな行列として表したものである．k_0 から $k_1=k_0+\omega N_x-1$ までが現在のブロックの範囲である．現在，k 行に注目して処理を行っているとする．ここで，行番号 k から k_1，列番号 k から k_1 の範囲の上三角行列と下三角行列をそれぞれ U' と L'，行番号 k_1+1 から $n-1$，列番号 k から k_1 の範囲の部分行列を L''，行番号 k から k_1，列番号 k_1+1 から $n+N_x-1$ の範囲の部分行列を U''，行番号 k_1+1 から $n-1$，列番号 k_1+1 から $n+N_x-1$ の範囲の行列を M とする．また，第 k 列のうち L' あるいは L'' の範囲のベクトルをそれぞれ l' と l''，第 k 行のうち U' あるいは U'' の範囲のベクトルをそれぞれ u' と u'' とする．さらに，第 k 列で対角要素より下の部分のベクトルを l とする．

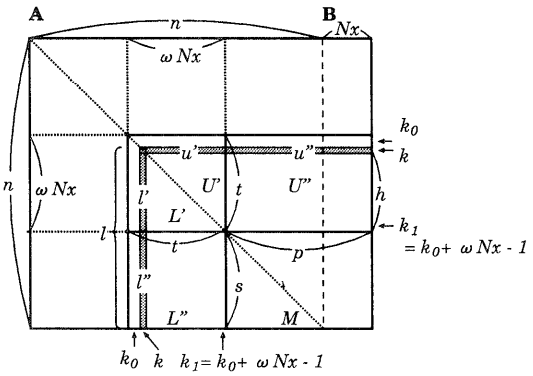


図 3 LU 分解
Fig. 3 LU decomposition.

LU 分解部では，パラメータ k を $k=0$ から $k=n-1$ まで変化させながら，以下の処理を実行する．

1. $k_0 \leq k < k_0+\omega N_x$ のとき，U', L', L'' の更新を行う．
   - (a) ベクトル l の中で絶対値が最大の要素を求め，ピボット行（第 i 行）を決定する．
   - (b) ピボット行を持つセルはピボット行を y 方向にブロードキャストし，同一列にあるすべてのセルで共有する．第 k 行が割り当てられているセルは第 k 行とピボット行を交換し，第 k 行をピボット行を持っていたセルに送り返す．ピボット行を持っていたセルは第 i 行を送られてきた第 k 行で置き換える．
   - (c) l をピボットで正規化する．l を x 方向にブロードキャストし，同一行にあるすべてのセルで共有する．
   - (d) l と u' により外積を計算し，U', L', L'' の部分を更新する．
2. $k=k_1+1$ のとき，U'' と M の更新を行う．
   - (a) k_0 から k_1 まで変化するパラメータ j を導入する．l'_j を行列 A の第 j 列のうち L' の要素からなる列ベクトル，U''_j を第 j 行のうち U'' の要素からなる行ベクトルとする．このとき，U''_j を外積 $U''_j \leftarrow U''_j + l'_j \times u''_j$ で更新する．
   - (b) 行列 M の部分を行列積 $M \leftarrow M + L'' \cdot U''$ により更新する．

このアルゴリズムは通信を含まない外積計算（ステップ 1-d, 2-a）や行列積計算（ステップ 2-b）が適用でき，ベクトル化に適している．

3.1.2 後退代入部のアルゴリズム

図 4 は，行列 A と未知ベクトル x をグローバルな行列として表したものである．$k_1=k_0-\omega N_x+1$ から

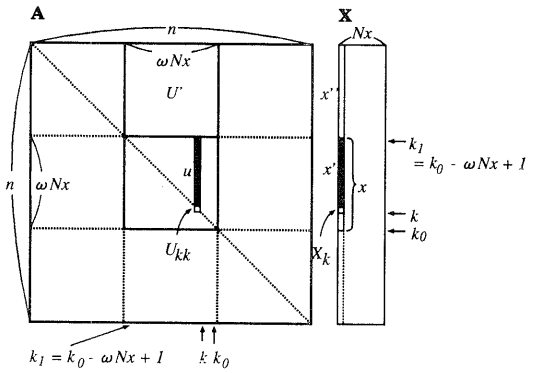


図4 後退代入
Fig. 4 Back substitution.

k_0 までが現在のブロックの範囲であり，現在，k 行あるいは k 列に注目して処理を行っている様子である．U_{kk} は上三角行列 U の第 k 行第 k 列の対角要素，X_k は未知ベクトル x の第 k 要素である．u と x はそれぞれ，第 k 列のベクトルと未知ベクトル x の要素番号 k_1 から $k-1$ の範囲の部分ベクトルである．x'' は未知ベクトル x の要素番号 0 から k_1-1 までの範囲の部分ベクトルである．

後退代入部のアルゴリズムは次の通りである．

1. 未知ベクトル x に定数ベクトル b を代入し，符合を変える．
2. パラメータ k を $k=n$ から $k=0$ まで変化させながら以下の処理を行う．
   (a) k がブロックの範囲にある場合
      i. k 列を持つセルは，k 列でブロックの範囲にある部分 (u) を x 方向にブロードキャストし，同一行にあるすべてのセルで u を共有する．
      ii. k 行を持つセルは，$X_k \leftarrow X_k \cdot (-1/U_{kk})$ を求め，X_k を y 方向にブロードキャストし，同一列にあるすべてのセルで X_k を共有する．
      iii. $x' \leftarrow u \times X_k$ により x' を更新する．
      iv. $k \leftarrow k-1$ として (a)-1 に戻る．
   (b) k がブロックの範囲を超えたら，次のブロックへ移る前に次の方法で x'' を更新する．
   $x'' \leftarrow U' \cdot x'$

後退代入部は逐次処理であるが，計算量は LU 分解部よりも少ない*．

* 計算量は，LU 分解部は $O(n^3)$，後退代入部は $O(n^2)$ である．4×4 セルで $n=2000$ の問題を与えた場合の演算時間は，LU 分解部は 10.9 秒，後退代入部は 0.39 秒であり，LU 分解部に対する後退代入部の演算時間の割合は 3.6%であった．

表 1 matmul のメモリ参照率 ($p=128$ の時) と NCA 使用時の単体性能 (倍精度)
Table 1 Memory reference for $p=128$ and single-processor performance of matmul (double precision).

アルゴリズム	R	性能 MFLOPS
Loop rolled	0.504	65.9
Loop unrolled	0.129	80.3

3.2 ベクトル化

プログラムのベクトル化は，上記アルゴリズムに基づくスカラプログラムを基礎として，行列積，外積，ベクトルのコピー，行列のコピー，ベクトルのスカラによる除算の各モジュールを NCA による処理関数で置き換えた．

ベクトル化に際し，次の 3 点に注意した．

- μVP へのデータのロード/ストアと算術演算をスケジュールし，ベクトルパイプラインの処理効率を高く保つこと．
- ループアンローリングし，メモリ参照を減らすこと．
- ベクトル長をできるだけ長く保ち，パイプラインの初期化オーバヘッドの影響を小さくすること．

3.2.1 行列積演算の最適化

行列積を求めるルーチン $M \leftarrow M + L'' \times U''$ には，ループアンローリングを施し最適化を行った．ここに，L'', U'', M は大きさがそれぞれ $s \times t$, $t \times p$, $s \times p$ の行列であるとする (図 3)．以下の説明で，$A_{i.}$ と $A_{.j}$ は行列 A の第 i 行と第 j 列をそれぞれ表すものとする．

1 回のループで行列 M の 1 列を更新するアルゴリズム $M_{i.} \leftarrow M_{i.} + \sum_{k=0}^{n-1} L''_{i.k} \cdot U''_{k.}$ には，$(1+p)st$ 個の浮動小数点データのロードと，$2stp$ 個の算術演算が含まれるので，メモリ参照率 R は $R=(1+p)/2p$ となる．

これにループアンローリングを施し，1 回のループで行列 M の l_{max} 行を更新できるようなアルゴリズムに変更する．

$$M_{i+l,.} \leftarrow M_{i+l,.} + \sum_{k=0}^{n-1} L''_{i,k} \cdot U''_{k.} \qquad (l=0, \cdots, l_{max}-1)$$

ベクトルレジスタの容量を最大限に使うと，$l_{max}=4$ となる．$s'=\lfloor s/4 \rfloor$ とすると，$(4+p)s't$ 個のデータロードと $8s'tp$ 個の算術演算が含まれるので，$R=(4+p)/8p$ となる．

表 1 に，$s=t=p=128$ とした時の行列積演算モジュール matmul のメモリ参照率と NCA 併用時の単体性能を示した．ベクトル演算器は一般に，ベクトル長が長いほど高い演算性能を示すが，μVP の倍精度浮動小数点データのベクトル長は最長で 128 である．このア

ルゴリズムでは行列 M と U'' の各行をベクトルレジスタに格納するので，p が最長のベクトル長と等しくなる $p=128$ のときに行列積演算の最大実効性能が得られる．ループアンローリングによって R は約 1/4 となり，NCA を使っての行列積演算ルーチンの性能は22%改善された．

3.2.2 外積演算とベンチマーク性能の向上

LINPACK の外積計算には，(1) 列ベクトルが行ベクトルより長い場合（LU 分解部ステップ 1-d），(2) 行ベクトルが列ベクトルより長い場合（同ステップ 2-a）の 2 種類がある．ベクトル長を長く保つためには，列ベクトルと行ベクトルのうちベクトル長の長い方をベクトルレジスタにロードするのが良い．2 つの場合はほぼ同じ回数だけ現れるので，(1)，(2) に対応する 2 つのルーチンを用意した．

LINPACK ベンチマークで，$n=500$ の問題を単体で解いた場合，2 つのルーチンを併用した場合の性能は，(2) に対応するルーチンだけを用いた場合の性能に対して，5%の向上が見られた．

3.2.3 ブロッキングの効果

一般に，外積計算は，行列積計算に比べて 1 浮動小数点演算あたりのデータロード回数が多いため，高性能が得られにくい*．そこで，LINPACK のインプリメンテーションでは，行列積と外積が分担する演算量の比をブロッキングファクタ ω で制御し，NCA を効率良く使うことにより，ベンチマーク性能の向上を図った．

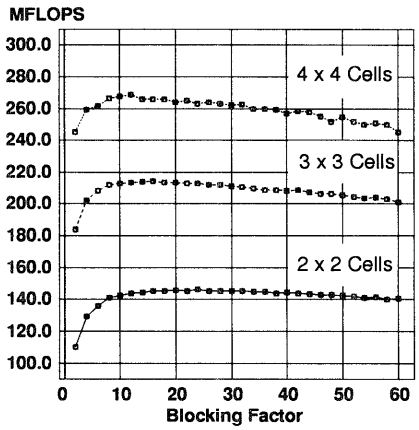


図5 ブロッキングファクタ ω と LINPACK (TPP)性能との関係（NCA 使用，問題サイズ $n=1000$）

Fig. 5 Relationship between blocking factor and LINPACK (TPP) performance (using NCA, matrix size $n=1000$).

* NCA で得られる最高性能は，倍精度演算時に，外積が 43.7 MFLOPS，行列積が 80.3 MFLOPS（表 1）である．

図5に，並列実行時の ω とベンチマーク性能との相関を示した．ω が大きいと大部分の演算を外積で行うため，また ω が小さいと行列積演算 1 回あたりの演算量が不十分なため，全体としての高性能が得られない．セル数が多いほどこの傾向が強くなり，ピーク性能を得られる ω の範囲が狭められる．性能がピークとなる ω の値は，セル数が増えるにしたがって小さくなっている．ブロッキングを施したプログラムでは，ω の値を注意深く制御することで適正な演算量を NCA に供給でき，最適な状態でプログラムを実行することが可能になる．

4. 結果と考察

ベンチマークの結果を表現するうえで，以下の記号を用る．

$Cells$ ：セル数 $(=N_x \times N_y)$．
r ：ベンチマークの性能．
n_{half} ：r の半分の性能が得られる問題サイズ．
r_{peak} ：理論ピーク性能．浮動小数点演算ユニットの性能のみを考慮する．

4.1 ベンチマークの結果

まず，NCA を付加したセルの単体性能を示す（図6）．これは問題サイズ n に対する性能をプロットしたものである．$0<n\leq 500$ の範囲では，n の増加に伴い μVP に十分な演算量が供給されてゆくので，良い立ち上がりを見せている．$n>500$ では性能が飽和して行く．表 1 に示した NCA による行列積演算の最大性能 80.3 MFLOPS に押えられるためである．この測定で得られたベンチマークの単体性能の最大値は，$n=1400$ の時の $r=68.2$ MFLOPS であり，NCA のピーク

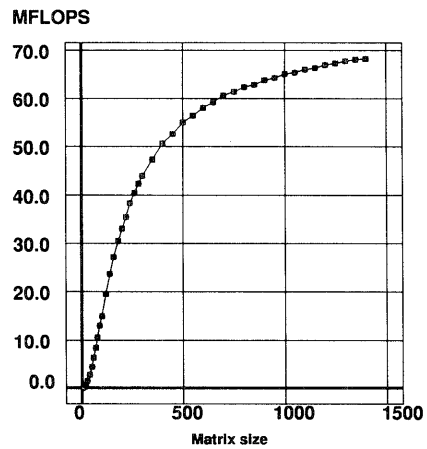


図6 ベンチマークの単体性能

Fig. 6 Benchmark performance of single processor.

表 2 LINPACK ベンチマークの結果
Table 2 LINPACK benchmark results.

(a) $n=1000$

Cells	Time [s]	r [MFLOPS]	r_{max}/r_{peak}
16	2.51	266.2	0.16
9	3.12	215.4	0.23
4	4.58	145.6	0.34
1	10.3	65.0	0.61

(b) $n=500\ N_x$

Cells	r_{500} [MFLOPS]	n [order]	n_{half} [order]	r_{peak} [MFLOPS]	r_{500}/r_{peak}
16	624	2000	800	1696	0.37
9	321	1500	620	954	0.34
4	145	1000	480	424	0.34
1	54.4	500	164	106	0.51

(c) $n=1000\ N_x$

Cells	r_{1000} [MFLOPS]	n_{max} [order]	n_{half} [order]	r_{peak} [MFLOPS]	r_{1000}/r_{peak}
16	804	4000	1200	1696	0.47
9	453	3000	900	954	0.47
4	201	2000	600	424	0.47
1	65.0	1000	200	106	0.61

r_{500}：$n=500\ N_x$ の時の性能.
r_{1000}：$n=1000\ N_x$ の時の性能.

性能 106 MFLOPS の 64.3%にあたる.

次に，並列処理時の性能を示す(**表 2**)．測定は，$N_x=2,3,4$ の場合について，4×4 のメッシュ上に配置された NCA を付加したセル 16 台を用いて行った．セルを正方形格子状に配置することでハードウェアによるブロードキャスト機構が使用できるため，最適な通信処理が行える.

$n=500N_x$ とした場合，セル数の増加につれて性能はリニアに向上する (表 2-b)．複数セルでの実行時には，通信のオーバヘッドによって，単体での実行時に比べて効率 r_{500}/r_{peak} が低下する．しかし，セル数が増加しても，効率はほぼ一定に保たれている.

$n=1000N_x$ とした場合，セル数の増加に対する性能と効率の変化の様子に，$n=500N_x$ の場合と同様の傾向が見られる (表 2-c)．ただし，性能と効率のどちらも先ほどと比べて高い値を示している．問題のサイズが大きくなったことで，NCA に充分な演算量が供給されるようになったためだと考えられる.

$n=1000$ の問題を，NCA を付加した 16 セルを使って解くのに要する時間は 2.51 秒である (表 2-a)．同じ規模の問題を SPARC のみで解く場合は，128 セルを使って 2.42 秒かかることが報告されている(**表 3**)．したがって，NCA を付加した 16 セルの AP 1000 は，

表 3 LINPACK ベンチマークの結果 ($n=1000$) (文献 1) より抜粋)
Table 3 LINPACK benchmark results for $n=1000$ (reported in Ref.1)

Cells	Time [s]	Speedup	Efficiency
512	1.10	147	0.29
256	1.50	108	0.42
128	2.42	66.5	0.52
64	3.51	46.0	0.72
16	11.5	13.9	0.87
4	41.3	3.90	0.97

表 4 各部分の実行時間 (16 セル使用)
Table 4 Execution time of each function (using 16 cells).

(a) SPARC のみ使用

		問題サイズ n			
		500	1000	1500	2000
T_{all}	[s]	2.52	15.66	48.06	108.67
T_{calc}	[s]	1.79	13.24	42.72	99.44
T_{matmul}	[s]	0.96	9.82	34.75	84.92
T_{outerp}	[s]	0.65	2.69	6.31	11.54
T_{comm}	[s]	0.73	2.42	5.34	9.23
T_{comm}/T_{all}	[%]	0.29	0.15	0.11	0.08

(b) NCA を併用

		問題サイズ n			
		500	1000	1500	2000
T_{all}	[s]	0.86	2.73	6.42	11.74
T_{calc}	[s]	0.19	0.82	2.50	5.07
T_{matmul}	[s]	0.05	0.46	1.60	3.88
T_{outerp}	[s]	0.09	0.23	0.65	0.82
T_{comm}	[s]	0.67	1.91	3.92	6.67
T_{comm}/T_{all}	[%]	0.78	0.70	0.61	0.57

SPARC のみの 128 セルの AP 1000 に匹敵する性能をもっているといえる.

4.2 計算時間と通信時間の割合の変化

表 4 に，16 セルの AP 1000 で SPARC のみおよび NCA 併用でベンチマークを実行した場合の，ベンチマーク・プログラムを構成する各部分の実行時間を示す.

T_{all} はベンチマークの総実行時間，つまり LU 分解部と後退代入部の処理に要した時間を示している．T_{calc} は浮動小数点演算に費やした時間を示しており，T_{matmul}，T_{outerp} で示した行列積と外積の演算実行時間が含まれる．通信時間 T_{comm} は $T_{all}-T_{calc}$ で見積もった.

$n=2000$ の問題を解く場合の通信時間は，SPARC のみの場合の 9.2 秒から NCA を併用した場合の 6.7 秒へ減少する．NCA を使用するために浮動小数点演

算データをセルメモリからNCAメモリへ移行したので，SPARCが行う通信処理のキャシュヒット率が向上したことがその理由として考えられる．

通信時間と総実行時間の比率 T_{comm}/T_{all} は，SPARCのみの場合の8%からNCAを併用した場合の57%へ増加する．NCA併用時の総実行時間がSPARCのみ使用時の約1/10になり，ベクトル処理で加速されない通信処理部の比率が高まったため，通信処理のオーバヘッドが顕在化した．

4.3 ハードウェアによる放送の効果

NCAを使用すると演算時間が短縮されるため，処理時間を占める通信時間の割合が有意になる．通信のオーバヘッドが性能に与える影響を調べるために，8×2のメッシュ上に16台のNCA付きセルを配置して実験を行った．本アルゴリズムでは正方形のメッシュを仮定しており，実行時には8×2のメッシュを論理的に4×4メッシュにマッピングする．このとき，ブロードキャスト通信は1対1のセル間通信でエミュレートされるため，通信時間が増加する．この性質を利用して測定を行った．SPARCのみを使った場合とNCAを併用した場合について，ベンチマーク性能と実行時間を測定した（**表5**）．

表5-(a) はベクトル化していないLINPACKの実行結果である．通信時間の増加による性能の変化は2%に留まった．通信機構の差は性能にほとんど影響を与えなかったことがわかる．一方，表5-(b) はベクトル化したLINPACKの実行結果である．(a) と比較して，処理時間は約1/10となっている．実行時間の差（$t_{brd}-t_{1by1}$）が通信機構の変化によるオーバヘッドの増加分で，(a)，(b) ともほぼ同じである．演算時間の短縮が，相対的に通信のオーバヘッドの増加を起こし，通信機構の変化による性能の劣化が大きくなった．セル数が多いほどこの傾向が顕著である．

NCAを付加して処理の高速化をするには，通信時間を含めたスカラ処理部分の処理時間を減らすことが重要である．

表5 通信機構の性能への影響

Table 5 Effect of communication mechanism on performance.

(a) SPARCのみ使用

Cells		4	9	16
n		1000	1500	2000
r_{brd}	[MFLOPS]	16.0	35.6	63.6
r_{1by1}	[MFLOPS]	16.0	35.0	62.2
r_{brd}/r_{1by1}		1.00	1.02	1.02
t_{brd}	[s]	41.9	63.3	84.0
t_{1by1}	[s]	42.0	64.5	85.9
$t_{brd}-t_{1by1}$	[s]	0.1	1.2	1.9

(b) NCAを併用

Cells		4	9	16
n		1000	1500	2000
r_{brd}	[MFLOPS]	145	321	624
r_{1by1}	[MFLOPS]	146	288	473
r_{brd}/r_{1by1}		0.99	1.11	1.32
t_{brd}	[s]	4.61	7.03	9.42
t_{1by1}	[s]	4.58	7.75	11.3
$t_{brd}-t_{1by1}$	[s]	0.03	0.72	1.88

r_{brd}, t_{brd}：ハードウェアによるブロードキャスト機構の使用時の性能と処理時間．

r_{1by1}, t_{1by1}：1対1通信によるブロードキャストのエミュレーション時の性能と処理時間．

4.4 大規模システムにおける性能の予測

NCAの実機を用いたプログラム実行は4×4セルのシステムを使って行った．並列計算機上での問題規模や台数を変化させた時の性能見積りは，演算性能ばかりでなく，通信パタンや通信量の変化による性能への影響を考慮しなければならない．大規模なシステムにおける性能を高い精度で予測するために，これらの要因を評価するメッセージ・レベル・シミュレータ（Message Level Simulator; MLSim）を用いて実験を行った．

MLSimはメッセージ通信型の並列計算機の性能をシミュレーションで評価するものである[3]．MLSimは実行時にパラメータを様々に変化させて，その影響を見ることができる．

例えばcomputation_factorというパラメータによってCPUの性能を変化させることができる．その値が1.0の時が，現在のAP 1000のCPU性能と同等であることを意味し，その値を0.5とすると，2倍の性能のCPUを仮定することができる．

今回の性能見積もりでは，NCAの付加がセルのCPUの性能向上と等価であるとみなしてシミュレーションを行った．つまり，NCAを付加した単体のセルで，様々な大きさの問題に対してプログラムのベクトル化によるスピードアップ（S）を求めておき，複数セルを使った場合には1つのセルが担当する行列の大きさを基にcomputation_factorを決定した．具体的には，単体で問題規模が500×500の時はスピードアップが18.6倍であったので，1/18.6=0.0538をcomputation_factorとして $N_x \times N_x$ セルで $500N_x \times 500N_x$ の問題を解く場合のシミュレーションを行った．なお，ベクトル化による通信時間の減少は考慮から外した．

表6にシミュレーションの結果を示す．表中，S_1 は問題規模が $n/\sqrt{\text{Cells}}$ の時のNCAを使用したことに

表 6 MLSim による台数効果のシミュレーション結果
Table 6 Result of simulation by MLSim.

Cells	n [order]	r [GFLOPS]	r_{peak} [GFLOPS]	r/r_{peak}	S	$r/r_{peak}/S$
256	8000	10.3	27.1	0.38	18.6	0.020
64	4000	2.55	6.78	0.38	18.6	0.020
16	2000	0.667	1.70	0.39	18.6	0.021
4	1000	0.176	0.42	0.41	18.6	0.022
256	1000	1.47	27.1	0.054	4.05	0.013
64	1000	0.702	6.78	0.10	7.47	0.014
16	1000	0.411	1.70	0.24	14.0	0.017
4	1000	0.176	0.42	0.41	18.6	0.023

S：単体で $n/\sqrt{\text{Cells}}$ の問題を解いた時のスピードアップ．

よる単体のスピードアップを示し，第7列目は r/r_{peak} と S の比 $(r/r_{peak})/S$ を示す．

$n=500N_x$ の場合，セル数が増えるにつれて r が直線的に増加する．r/r_{peak} はわずかずつ減少してはいるものの，ほぼ一定の値を保っている．したがって，$500N_x$ 程度の問題であれば，通信量，通信パタンの変化はほとんど影響せずに，高い性能が得られることがわかる．

一方で，$n=1000$ と固定した場合は，セル数の増加にともなって r/r_{peak} は大きく減少する．$(r/r_{peak})/S$ が減少していることから，単体のスピードアップの減少以上に性能が減少していることがわかる．セル数が多くなるにつれて1台のセルに割り当てられる問題が小さくなり，また演算量に対する通信量の割合も増加するため，効率の良い実行が行えない．

5. まとめ

本稿では，LINPACK ベンチマークを NCA 付き AP 1000 に向けて並列ベクトル化した．ブロッキングを施した並列化を行うことで，ベクトルプロセッサでの演算の単位を大きく保ち，高い性能を実現することができた．

ベクトル化における最適化手法とその効果については，次の2点について考察した．行列の積を求めるルーチンに対しては，ループアンローリングを適用することで，そのルーチンの性能を20%向上させた．今回のインプリメントでは，LINPACK の最大性能は行列積を求めるルーチンの最大性能によって制限されるので，この効果は大きかった．ベクトルの外積を求めるルーチンに対しては，LINPACK の実行中にベクトル長を長く保つために，ベクトルレジスタの使い方が違う2つのルーチンを用意することで，ベンチマーク性能を5%向上させた．

性能予測においては，通信方式のおよぼす影響についてブロードキャストの効果を調べた．また，シミュレーションにより演算性能だけでなく通信量や通信パタンの変化も考慮することで，大規模システムでの性能を高い精度で得ることができた．その結果，処理要素数が多い場合，それに見合う大規模な問題を与えることで NCA を効果的に使用できることを確認した．

NCA を使うことで算術演算に要する時間が大幅に短縮され，通信処理時間が顕在化するため，NCA を使った時の通信時間が処理速度に及ぼす影響は非常に高くなる．ブロードキャストなどの数の増加に対するインパクトの小さなプリミティブを利用し，単体が分担する問題規模を十分大きく保てば，リニアな性能向上が得られることを示した．

今回は 16 台までの NCA を用いて実験を行った．今後セル数を増やして性能評価を行い，ベクトル化と並列化が相互に及ぼす影響についてさらに詳細な検討を進める予定である．

本論文では，文献 1) に基づいて，二次元サイクリック分割によるデータ分割法を用いた．他のデータ分割法としては，列単位（行単位）サイクリック分割，ブロック分割が考えられる．その中で列単位サイクリック分割は，二次元サイクリック分割と比べて通信量が増える，ロードバランスがやや劣るといった欠点はあるものの，ピボット選択時に通信が不要，ベクトル長が長くとれるといった利点もある．この分割法でのインプリメントを行い，検討する必要があると考える．

並列ベクトル化のアプリケーションとして LINPACK ベンチマークを使用した．これには科学技術計算の並列処理に一般的に現れる算術演算と通信処理が盛り込まれているので，並列・ベクトル処理の特徴を知る良い指標だといえる．並列ベクトル処理の詳細な特徴を調べるために，今後さらに多くのアプリケーションに対して検討を進める．

また，NCA を利用するアプリケーションを，数多く，効率的に開発するためには，BLAS ライブラリなどの，基本的な処理を行うプログラムを用意するだけではなく，ベクトルコードを自動生成するコンパイラ等の開発も進めて行く．

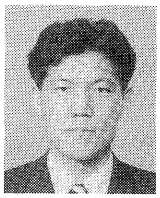
謝辞 日頃より御指導，御助言をいただいている，富士通(株)HPC 本部石井本部長代理，白石開発部長，佐藤プロジェクト課長，池坂プロジェクト課長，ならびに同僚諸氏に感謝いたします．

参考文献

1) Brent, R. P.: The LINPACK Benchmark on the AP1000, *Proc. Frontiers'92*, IEEE, Virginia,

pp. 128-135 (1992).
2) Dongarra, J. J.: Performance of Various Computers Using Standard Linear Equations Software in a Fortran Environment, 4998 LINPACK REPORT Dec. 27, HPCwire (1995).
3) Horie, T., Hayashi, K., Shimizu, T. and Ishihata, H.: Improving AP1000 Parallel Computer Performance with Message Communication, *The 20th Annual Int. Symp. Computer Architecture,* pp. 314-325 (1993).
4) 石畑宏明, 稲野 聡, 堀江健志, 清水俊幸, 池坂守夫：高並列計算機AP 1000のアーキテクチャ, 電子情報通信学会論文誌DI, Vol. J75-D-I, No. 8, pp. 637-645 (1992).
5) 清水俊幸, 石畑宏明, 飯野秀之, 木村雅春：並列計算機AP 1000用数値演算アクセラレータの構成と評価, SWoPP鞆の浦'93ハイパフォーマンスコンピューティング研究会 (HPC), pp. 17-24 (1993).
6) 富士通 (株)：マイクロベクトルプロセッサ：μVPシリーズMB 92831ユーザマニュアル, ハードウェア編 (TM 61-10101-1), ソフトウェア編 (TM 61-10102-1).

(平成6年9月16日受付)
(平成7年5月12日採録)



上村 和人

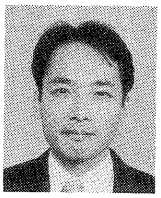
1969年生. 1991年群馬大学工学部電子工学科卒業. 1993年同大学院電気電子工学専攻博士前期課程修了. 同年 (株) 富士通研究所入社. 並列計算機に関する研究開発に従事. 現在, 富士通株式会社HPC本部に勤務. 電子情報通信学会会員.

清水 俊幸 (正会員)

1964年生. 1986年東京工業大学工学部電子物理工学科卒業. 1988年同大学院理工学研究科情報工学修士課程修了. 同年 (株) 富士通研究所入社, 現在に至る. 並列計算機に関する研究開発に従事. 1992年電子情報通信学会論文賞受賞. 電子情報通信学会会員.

石畑 宏明

1957年生. 1980年, 早稲田大学理工学部電子通信学科卒業. 同年 (株) 富士通研究所入社. 画像処理システムの研究, 並列コンピュータアーキテクチャの研究に従事. 現在, 富士通株式会社HPC本部に勤務. 元岡賞, 電子情報通信学会論文賞受賞.



堀江 健志 (正会員)

1962年生. 1984年東京大学工学部電気工学科卒業. 1986年同大学院修士課程修了. 同年 (株) 富士通研究所入社, 現在に至る. 並列計算機に関する研究開発に従事. 1992年電子情報通信学会論文賞受賞.