ウェーブフロント型並列処理における分散メモリ型並列計算機の通信機構の評価

坂根広史[†,††] 児玉祐悦[†] 建部修見[†]
小池汎平[†] 山名早人[†]
山口喜教[†,†††] 弓場敏嗣[††]

本論文では，分散メモリ型並列計算機において，同期・通信の支援機構が行列問題の並列処理性能に与える影響について議論し，それらが有効となる条件・要因を，モデルと実験によって定量的に明らかにする．LU分解法の代入部に現れる三角方程式の求解では，互いに依存性のない計算要素がイテレーション間にまたがっており，その並列性はウェーブフロント状に抽出できる．この問題を，並列性を自然に利用する細粒度アルゴリズムと，ブロック化による粗粒度アルゴリズムで表し，並列計算機EM-XとAP1000+に実装した．最初に予備実験によって，これらの計算機が持つ同期・通信機構の特徴をパラメータによって表した．次に，アルゴリズムの性質をモデル化し，通信オーバヘッドに起因する性能上限と，並列度の制限による有効PE台数を理論的に示した．問題サイズが小さい場合，あるいは十分なPE台数が利用できる場合は，高い並列度が得られる細粒度アルゴリズムが有望である．ただし細粒度アルゴリズムで高い性能を得るには，通信起動のオーバヘッドが十分小さいことが必要であり，EM-Xがこの要件を満たす．逆に，問題サイズが十分大きいか，比較的少ないPE台数しか与えられない場合は粗粒度アルゴリズムの方が良い．この場合は通信性能より逐次演算性能が重要となり，AP1000+が優位性を示す．

Evaluation of Communication Mechanisms for Distributed Memory Parallel Computers in Wavefront Computation

Hirofumi Sakane,[†,††] Yuetsu Kodama,[†] Osamu Tatebe,[†]
Hanpei Koike,[†] Hayato Yamana,[†] Yoshinori Yamaguchi[†,†††]
and Toshitsugu Yuba[††]

In this paper, we discuss efficient parallel execution of a dense-matrix problem considering trade-offs between fine-grain and coarse-grain communication in distributed memory machines. The solution of the triangular system of equations involves data dependencies between consecutive iterations in the outer-loop. The dependencies can be naturally solved and processed in parallel by wavefront computation. Two ways of parallelizing are presented; the element-wise fine-grain approach and the coarse-grain approach. We implemented these algorithms on both EM-X and AP1000+. Fine-grain support mechanisms of the EM-X had a great effect on the performance of the element-wise method for relatively small problem size, while employed RISC processors of the AP1000+ brought high performance of the coarse-grain method for larger size.

1. はじめに

分散メモリ型並列計算機の重要な設計要素として，通信・同期の支援機構の方式およびパラメータの決定がある．このパラメータとは，通信・同期の起動時間，レイテンシ，スループット等であり，並列処理の性能に大きく影響する．本論文では，密行列計算のように通信パターンが定型で送受信の位置と順番が静的に決定できる問題において，分散メモリ型並列計算機の通信・同期パラメータが性能に与える影響を議論し，その支援機構の有効性を定量的に評価する．これにより設計者は，必要な性能を得るのに最適なパラメータを選択することができる．

† 電子技術総合研究所情報アーキテクチャ部
Computer Science Division, Electrotechnical Laboratory
†† 電気通信大学大学院情報システム学研究科
Graduate School of Information Systems, The University of Electro-Communications
††† 筑波大学電子・情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba

LU分解法の代入部に現れる三角方程式の求解では，二重ループ内の依存性のない要素計算を，イテレーション間にまたがったウェーブフロント状に抽出することができる．これを自然な形で並列処理すると要素計算ごとに細粒度の通信・同期が必要となり，そのオーバヘッドが問題となる．これらのオーバヘッドを減らすためには，行列のブロック化により粗粒度の処理に変換することが有効であるが，ブロックサイズを大きくしすぎると負荷分散の悪化と並列度の低下を招く．分散メモリ型並列計算機では，通信・同期のコストが粒度に関するトレードオフに大きく影響する．この性質を明らかにするため，通信・同期を要素ごとに行う細粒度アルゴリズムとブロックごとに行う粗粒度アルゴリズムによって求解プログラムを作成し，並列計算機EM-XとAP1000+に実装する．

EM-Xは専用に設計された要素プロセッサによって，低オーバヘッドの細粒度通信が可能である．一方，AP1000+では，汎用RISCプロセッサと外部通信機構の組合せのため通信オーバヘッドはEM-Xより大きいが，逐次計算性能は高く，さらに，ブロックデータ転送を演算とオーバラップできる特長がある．これらのことからEM-Xは細粒度に適し，AP1000+は粗粒度に適していることが予測できるが，それぞれの通信・同期支援機構のどのパラメータが性能に重要な影響を及ぼしているか，高い性能が発揮できる問題領域はどこからどこまでかを示すことが重要である．また，新しい計算機の設計者にとっては，ハードウェア投入量を必要最小限にしつつ，想定する問題領域を最大の効率で解くための設計指針を与えることが重要である．

実験による性能測定に先立ち，通信モデルとアルゴリズムの計算性能モデルを立て，理論的な性能上限と並列度を求める．これにより，実験結果を理論的に考察することができ，パラメータが性能に与える影響を明らかにすることができる．

本論文の構成を次に示す．2章でまず三角方程式とその並列化の方法を述べ，細粒度アルゴリズムとブロックアルゴリズムを示す．3章でEM-XとAP1000+の通信モデルについて述べる．4章では各アルゴリズムの計算モデルについて論じ，理論的な性能上限を示す．5章では，性能の測定結果を示し，パラメータが性能に与える影響と，各アルゴリズムが有効となる条件を議論する．6章で総括を述べる．

2. 三角方程式

LU分解法による線形方程式 $\boldsymbol{Ax}=\boldsymbol{b}$ の求解アルゴリズムは，LU分解部と代入部に分けられる．係数行列をLU分解した結果得られた上下三角行列が，代入部で三角方程式として順に解かれ最終的な解が得られる．代入部は下三角行列 $\boldsymbol{L}$ を用いた三角方程式（式(1)）と上三角行列 $\boldsymbol{U}$ を用いた三角方程式（式(2)）に分かれている．式(3)，(4)は $n=5$ としてそれぞれを代入操作に展開したものである．式(3)の前進代入で中間解 $\boldsymbol{y}$ が得られ，それを用いて式(4)の後退代入で最終解 $\boldsymbol{x}$ が得られる．

$$\boldsymbol{Ly}=\boldsymbol{b} \tag{1}$$

$$\boldsymbol{Ux}=\boldsymbol{y} \tag{2}$$

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 - l_{10}y_0 \\ b_2 - l_{20}y_0 - l_{21}y_1 \\ b_3 - l_{30}y_0 - l_{31}y_1 - l_{32}y_2 \\ b_4 - l_{40}y_0 - l_{41}y_1 - l_{42}y_2 - l_{43}y_3 \end{bmatrix} \tag{3}$$

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} (y_0 - u_{01}x_1 - u_{02}x_2 - u_{03}x_3 - u_{04}x_4)/u_{00} \\ (y_1 - u_{12}x_2 - u_{13}x_3 - u_{14}x_4)/u_{11} \\ (y_2 - u_{23}x_3 - u_{24}x_4)/u_{22} \\ (y_3 - u_{34}x_4)/u_{33} \\ (y_4)/u_{44} \end{bmatrix} \tag{4}$$

このように三角方程式の求解は代入計算の繰返しによって進められ，行列サイズを $n \times n$ とすると計算量は $O(n^2)$ である．この代入部は $O(n^3)$ のLU分解部に対して演算回数が少ないが，次に述べるようにその重要性は小さくない．

- 一般にLU分解部はPE数に応じた良好な並列処理が行えるが，LU分解部を高速化するほど代入計算時間の割合が増加し，代入部の重要性が顕在化する．最近の超並列計算機事情は n に近い台数（1000〜10000）のPE投入を現実的にしつつあり，代入部についても並列化による高速化を行わなければ性能ボトルネックになる．
- 実アプリケーションにおいては，LU分解を一度だけ行った後，異なる定数ベクトルを用いて代入計算を繰り返すような処理が多い[1),2)]．

2.1 並列化

前進代入では，ある y_j に依存する列の乗算は原理的にすべて同時に行うことができる[2),3)]．しかしながら，全体の計算時間を最短にする目的のためには以降の y_j の計算を優先すればよく，その制約は，各列の計算は上の要素から，各行の計算は左の要素から計算が進みさえすれば充足される．このアルゴリズムでは，依存性のある要素間でのみ局所同期がとられるため，分散メモリ型並列計算機においてコストのかかるブ

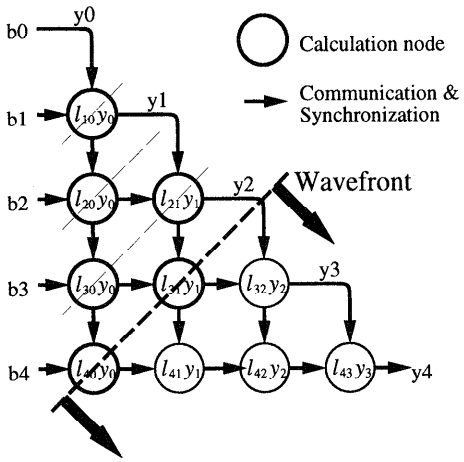


図1 ウェーブフロントの進行
Fig. 1 Wavefront propagation.

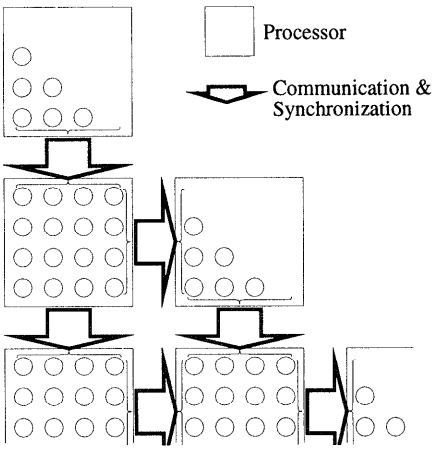


図2 ブロック化ウェーブフロント
Fig. 2 Block wavefront.

ロードキャストやグローバル演算を省略できる[4]．互いに依存性がなく並列計算可能な要素を表すウェーブフロントは，主対角線と直角に生じ，主対角線方向に伝搬する．

この様子を図1に示す．ウェーブフロントに沿って並列性が存在するため，並列度は対角線を通過するときに最大値 $\lfloor n/2 \rfloor$ をとり，対角線を離れるに従って抽出可能な並列度は減少し，処理の最初と最後においては並列度は1となる．

後退代入も同様の形式であるため，本論文では前進代入のみで議論を進める．

2.2 列サイクリック分割

行列の分割方法は，通信の効率と負荷分散の両方に影響を与えるため重要である．列方向の y_j の順送りの効率に焦点をあてると，レジスタ利用を前提とした列サイクリック分割が望ましい．列方向の通信がなくなり，PE内では saxpy 型演算となる．三角方程式問題は，計算が進行するに従って計算量が減っていくが，列サイクリックでは各PEへの適度な負荷分散が得られる．また，各列の上の要素から順に左側の列から必要なデータを待ち受け，データが揃うとともに積和計算を行い，右側の列へデータを送ることによりウェーブフロントを自然に進行させることが可能である．これにより，列間では要素ごとの細粒度な通信・同期が行われ，それらの起動オーバヘッドが性能に大きく影響するため，**細粒度アルゴリズム**として詳しい評価を行う．

2.3 ブロック化ウェーブフロント

行列問題を分散メモリ型並列計算機で解く場合，同期・通信のオーバヘッドを削減するために行列をブロック化して分割配置し，ブロックアルゴリズムを適用するのが一般的である．ループアンローリングの効果向上や，キャッシュ効率向上の効果もある．

三角方程式の解法も，ブロック単位でウェーブフロントを適用可能である．基本的な処理の流れを図2に示す．ブロックの形状としては正方形を用いる．ブロック内の行方向の計算が内積計算となり，多くの逐次プロセッサで高速化が期待できる．通信および同期の頻度はブロック単位となり，1回の通信量は境界の長さに比例する．

ブロックサイズが大きくなると通信量と通信頻度は少なくなるが，負荷分散は悪化する．このため行列を行方向および列方向に分割して負荷の偏りを低減する必要がある．実装するアルゴリズムでは，良好な負荷分散が可能なブロックサイクリック分割[5]を用いる．列方向にもPEの割当てが巡回するため，下側ブロックがすでに必要なデータを持っていることがあり，この場合は通信を省略できる．

このブロックアルゴリズムでは，ブロックサイズに応じて通信起動オーバヘッドと通信スループットの重要性が変化する．また，ブロックサイズが大きくなると，負荷分散の問題のほかに，ブロック化にともなう並列度の低下が深刻な問題となる．

3. 分散メモリ型並列計算機の通信モデル

分散メモリ型並列計算機の通信モデルとして，LogPモデル[6]に変更を加え，代入計算に適したモデルを組み立てる．LogPモデルについては，文献7)，8)で並列ループへの適用や拡張が試みられている．ハードウェアによる通信機構を備えた分散メモリ型並列計算機と

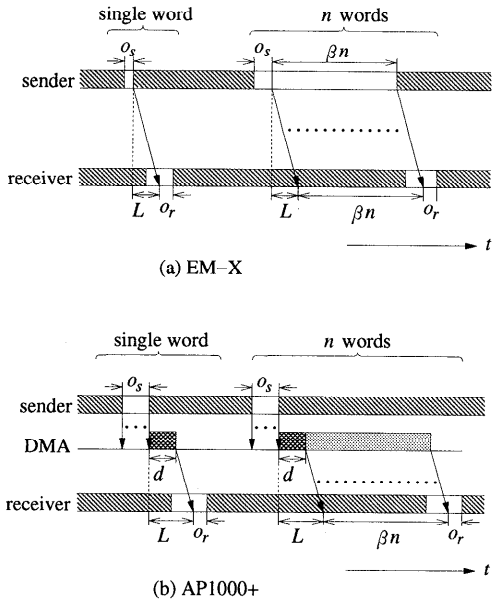


図3 EM-X と AP1000+の通信モデル
Fig. 3 Communication models of EM-X and AP1000+.

して，EM-X（80 PE）と AP1000+（256 PE）を取り上げる．これらは，ともにユーザレベルでリモートメモリに対する直接書き込みをサポートしている．通信モデルの違いとしては，通信オーバヘッドパラメータのほかに，ブロックデータの送出方法がある．それぞれの通信モデルを**図3**に示す．1ワード通信と n ワード通信は区別される．図中の記号を以下に説明する．

- L：通信レイテンシ
- o_s：送信オーバヘッド
- o_r：受信オーバヘッド
- β：1ワード（4 bytes）を転送するのにかかる時間（スループットの逆数）
- n：連続転送ワード数
- d：DMA セットアップ時間

ただし，LogP モデルではプロセッサ間通信レイテンシとして最大値で定義しているのに対し，ここでの L は平均値とする．AP1000+の L は DMA のセットアップ時間を含み，また，256 PE システムでの測定値である．o_r は受信 PE の受信処理開始よりも先にデータが到着している場合のオーバヘッドとする．スループットを表すパラメータは，LogP モデルの1ワード通信における最小送出間隔 g の代わりに，ブロックデータ転送における1ワードあたりの転送時間 β を用いる．EM-X，AP1000+とも，それぞれの特性に合わせて3種類の通信モデルを設定し，実機のパラメータを調べた．**表1**にそれぞれのパラメータを示す．* 印は測定により得られた値であることを表し，それ以外は各システムの仕様から求めたものである．

表1 EM-X と AP1000+の通信パラメータ
Table 1 Communication parameters of EM-X and AP1000+.

		L	o_s	o_r	β
EM-X (80 PE)	thread	0.46	0.05	0.05	0.10
	spin	0.46	0.05	0.20	0.10
	n_words	0.46	1.50	0.70	0.10
AP1000+ (256 PE)	inline0	5.21*	1.16*	0*	0.16
	inline	5.21*	4.10*	0*	0.16
	put()	5.21*	2.39*	0*	0.16

(μsec)

以下に，EM-X と AP1000+のアーキテクチャの特徴と通信モデルについて説明する．

3.1 EM-X

EM-X[9]は，演算ユニットと通信機構を1チップに内蔵した EMC-Y（20 MHz☆）を PE として備えている．パケット送出命令により，演算パイプラインから1クロックでパケットを直接送出できるため，送信オーバヘッドがきわめて小さい．パケットレベルで通信レイテンシと演算のオーバラップが可能であるが，パケットが持つことができるデータは1ワード固定であるため，大量のデータを送るときにはデータ数分のパケットを送出する必要があり，その間は他の計算を行うことはできない．ネットワークのデータ転送能力は 40 MBytes/sec である．メモリのデータは 32 bit に加えタグフィールドを 6 bit 備えており，同期フラグとして用いることができる．

本論文で扱う EM-X の通信モデルの実装は，受信方式の異なる2種類の1ワード通信と，1種類の連続 n ワード通信の3種類である．

thread EM-X では直接待合せ機構により，パケットの到着後1クロックでスレッドを起動させることができ，通信から演算へのデータ授受が低オーバヘッドで実現できる．**thread** ではデータ待ちのスレッドを用意し，必要なデータが到着次第計算を開始する．計算後は計算結果を必要とするスレッドへパケットを送出し，自スレッドを終了する．

spin 送受信の組合せと通信順序が静的に定まるならば，spin wait を用いて受信フラグを待ち合わせる方法も有効である．**spin** では，spin wait の命

☆ 安定動作のため現在は 16 MHz で運用されているが，本論文では設計仕様の 20 MHz を用いる．

令列をアセンブリ言語レベルでインライン挿入する．最初の3命令でフラグをチェックし，データが到着していなければスピンする．すでに到着していれば，その後フラグをクリアする命令が1個だけ実行されるため，合計4命令のオーバヘッドで済む．

送信は **thread**，**spin** ともインラインのパケット出力命令を用いることにより，$o_s = 1$ clock である．

n_words 連続した n ワードの送信はリモートブロック転送を行うライブラリルーチンの呼び出しで行い，受信同期は I-structure と呼ばれる同期付き局所通信機構を用いる．1ワード通信に比べて起動オーバヘッドは若干大きくなるが，ネットワークの最大スループットに近い性能を利用することができる．

なお，EM-X では8パケット分の出力バッファが一杯になるまでは $g = o_s$ であるが，定常的には $g = \beta$ となる．

3.2 AP1000+

AP1000+ [10)~12)]は，PE として SuperSPARC（50 MHz）を用い，通信支援機構として独自の MSC+を備えている．通信は，DMA を利用した PUT 機構をベースとして行う．MSC+のキューに対して，ソースとなるローカルメモリのアドレスや送り先となるリモートメモリのアドレス等の指定された8ワードを書き込むことにより，DMA 起動とデータ転送が自動的に行われる．MSC+へのパラメータ書き込み時間がオーバヘッドとなるが，その時間はデータサイズにかかわらず一定であり，その後次の計算を進めることにより通信のオーバラップが可能である．ネットワークのデータ転送能力は 25 MBytes/sec である．データの受信は同期フラグをチェックし，他 PE によって書き込まれたローカルメモリを直接読むことによって行われる．AP1000+ではこのほかリングバッファを用いたメッセージ通信が行える．

本論文では，AP1000+における通信モデルは PUT を基本とし，送信オーバヘッドの異なる3つの実装について調べた．

inline 細粒度で問題となるオーバヘッド (o_s) を削減するため，PUT を発行する8ワードの書き込みを，ライブラリ呼び出しではなく直接インライン展開したものである．5章で述べる細粒度アルゴリズムの実装と同一のループコードを用いてオーバヘッドを測定した．

inline0 参考のため，PUT のインライン展開コードのオーバヘッド (o_s) を十分長い送信間隔を置いて測定したものである．実際の性能評価には **inline** を用いる．

put() ライブラリ関数 put() を用いた PUT 通信である．十分長い送信間隔を置いて測定した．連続 n ワード転送にはこれを用いる．

AP1000+では $o_s > \beta$ であるため $g > \beta$ である．また，高頻度で PUT を発行すると MSC+のキュー溢れ処理によってオーバヘッドが増加する．**inline** のように連続する1ワード通信等で g が問題となる場合は，送信オーバヘッド o_s に含めて考える．**inline0** はキュー溢れが起きないため，**inline** に比べて o_s は小さい．

受信同期では，spin wait の命令列をインライン展開した．spin wait はアセンブリ命令4個となるが，実測ではオーバヘッドはほとんど認められなかった．これは，SuperSPARC のスーパスカラ動作によって前後の命令グループに取り込まれたためと考えられる．AP1000+における実装の受信方式は，すべて spin wait をインライン展開したものを用いた．

4. 性能モデル

2章で述べた細粒度アルゴリズムおよびブロックアルゴリズムを，EM-X ならびに AP1000+に実装した．性能評価に先立ち，本章ではそれらの通信パラメータと，理論的な性能上限を与える性能モデルについて論ずる．ウェーブフロント問題には原理的にグローバル通信や共有資源が存在しないため，並列度が十分存在すれば，使用 PE 台数に従ってほぼ線形に性能向上すると考えられる．これにより，通信オーバヘッドの増加に基づく性能上限と，問題が持つ並列度による性能上限が問題となる．

4.1 ブロックアルゴリズムの性能

ブロック化した前進代入では，ブロック間の依存性から導かれる理想的なクリティカルパスは対角ブロックとその1段下のブロックを通る．その様子を図4に示す．灰色のブロック（濃淡とも）がクリティカルパスを表す．図の下部は，クリティカルパス中のある連続するブロック（a，b，c とする）を処理する PE（PEa，PEb，PEc）の時間変化を表しており，太線がクリティカルパスを表す．図では通信と演算はオーバラップできるものとしている．理想クリティカルパスに含まれる計算時間と通信時間の総和を求めれば，それが実行時間の下限となる．なお，このパスを1個の PE で逐次処理させるとその長さは短くなるが，クリティカルパスが他のブロックへ移り，モデルが複雑

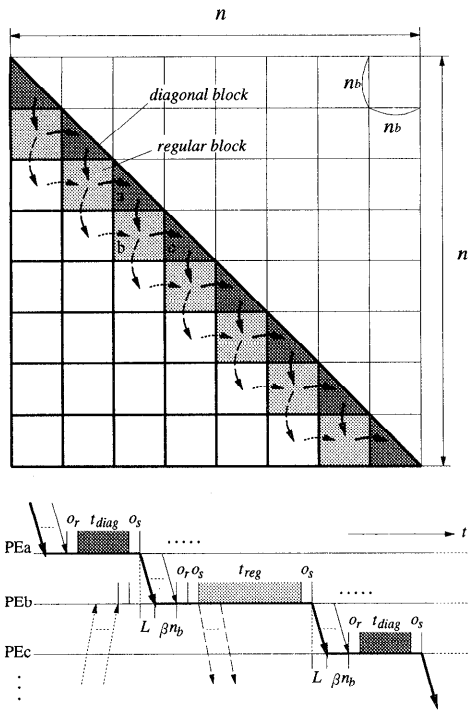


図4 ブロックアルゴリズムのクリティカルパス
Fig. 4 Critical path in the block algorithm.

になるため本論文では追求しない．

クリティカルパスの実行時間をモデル化するにあたり，以下のように記号を定義する．

n：係数行列のサイズ．
n_b：ブロックの1辺のサイズ．
N_B：係数行列の1辺のブロックの個数．
T_{comm}：クリティカルパスの通信時間の総和．
T_{comp}：クリティカルパスの計算時間の総和．
T_{cb}：クリティカルパスの全実行時間．
t_{reg}：非対角ブロックの計算時間（*regular*）．
t_{diag}：対角ブロックの計算時間（*diagonal*）．
α：通信を起動してから最初のデータが送り先PEに到着するまでの時間．
β：1要素分のデータを送るのにかかる時間．
o_s：送信オーバヘッド．
o_r：受信オーバヘッド．
L：通信路レイテンシ．
ただし $N_B = n / n_b$，$\alpha = o_s + L + o_r$ である．

クリティカルパスの実行時間 T_{cb} は通信時間 T_{comm} と計算時間 T_{comp} の和で与えられる．

$$T_{cb} = T_{comm} + T_{comp} \tag{5}$$

クリティカルパス中の通信時間は次の式で表される．

$$T_{comm} = 2(N_B - 1)(\alpha + \beta n_b) + (N_B - 2)o_s \tag{6}$$

右辺第1項は，クリティカルパスのブロックどうしの通信時間（図中実線矢印）であり，送受信オーバヘッドとデータ転送時間が含まれる．第2項は，クリティカルパスの非対角ブロックが対角ブロックから受け取った部分解ベクトルを下のブロックへ順送り（図中破線矢印）するためのオーバヘッドである．通信と計算とのオーバラップが可能であれば，下ブロックへのデータを転送する時間は含まれない．オーバラップ不可能であれば次の式となる．

$$T'_{comm} = 2(N_B - 1)(\alpha + \beta n_b) + (N_B - 2)(o_s + \beta n_b) \tag{7}$$

リティカルパス中の計算時間は次の式で表される．

$$T_{comp} = (N_B - 1)t_{reg} + N_B t_{diag} \tag{8}$$

t_{reg} および t_{diag} は，測定あるいは推定により n_b の関数としてあらかじめ求めておく．これらは一般に n_b に関する2次式となる．

全実行時間の下限（性能の上限）は，式(6)あるいは(7)と式(8)を式(5)に代入すれば求まる．

一方，並列度の観点からは有効な最大PE台数を見積もることができる．全体の計算量の総和とクリティカルパスの部分の総和の比は，通信オーバヘッドがないと仮定した場合の理論的な平均並列度と見なすことができ，性能向上比の上限となる．これは，その並列度を超えるPE数を与えても効果がないことを意味する．以下に示すように，平均並列度はブロック数の関数として表すことができる．

全体の計算時間の総和を T_{comp_total} とすると以下の式で表される．

$$T_{comp\_total} = \left(\frac{N_B^2}{2} - \frac{N_B}{2}\right) t_{reg} + N_B t_{diag} \tag{9}$$

ブロックサイズ n_b が大きいときは $t_{reg} \simeq 2t_{diag}$ と近似できるため，T_{comp_total}，T_{comp} はそれぞれ，

$$T_{comp\_total} \simeq \frac{N_B^2}{2} t_{reg} \tag{10}$$

$$T_{comp} \simeq \left(\frac{3}{2}N_B - 1\right) t_{reg} \tag{11}$$

となる．これらの比は平均並列度を表し，これを R とおくと，

$$R = \frac{T_{comp\_total}}{T_{comp}} \simeq \frac{N_B^2}{3N_B - 2} \tag{12}$$

となる．

4.2 細粒度アルゴリズムの性能

細粒度アルゴリズムでは，要素計算間の依存性から

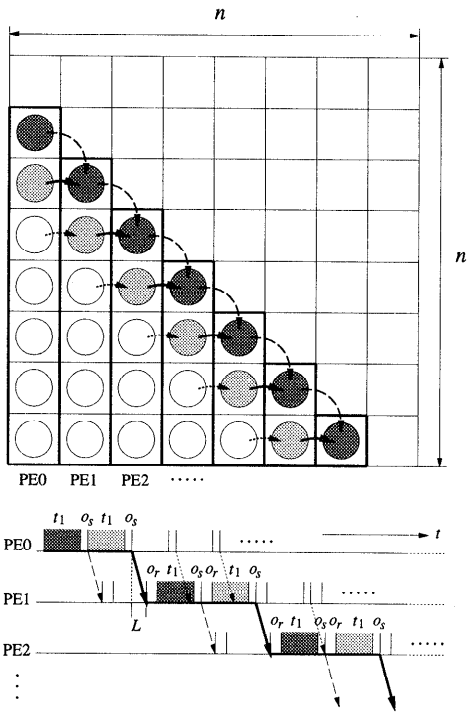


図5 細粒度アルゴリズムのクリティカルパス
Fig. 5 Critical path in the fine-grain algorithm.

導かれるクリティカルパスは，図5の灰色の円（濃淡とも）で示される．円は1回の積和演算を表す．下部は時間変化を表し，太線がクリティカルパスを表す．PE数は十分あるものとする．前節と同様の記号を用いると，クリティカルパス中の通信時間は次の式で表される．

$$\begin{aligned} T_{comm} &= (n-2)o_s+(n-2)(\alpha+\beta)+(n-3)o_r \\ &= n(2o_s+2o_r+L)-4o_s-5o_r-2L \end{aligned} \tag{13}$$

計算時間の基本量として次の記号を定義する．

t_1：1回の積和演算とその前後処理を含めた時間．

クリティカルパス中の計算時間は次の式で表される．

$$T_{comp} = (2n-3)t_1 \tag{14}$$

クリティカルパスの実行時間 T_{cf} は，ブロックアルゴリズムと同様に T_{comm} と T_{comp} の和となる．

$$\begin{aligned} T_{cf} &= T_{comm}+T_{comp} \\ &= n(2o_s+2o_r+L+2t_1) \\ &\quad -(4o_s+5o_r+2L+3t_1) \end{aligned} \tag{15}$$

平均並列度 R も，ブロックアルゴリズムと同様に T_{comp_total} を求めたうえで計算することができる．

$$T_{comp\_total} = \left(\frac{n^2}{2}-\frac{n}{2}\right)t_1 \tag{16}$$

$$R = \frac{T_{comp\_total}}{T_{comp}} = \frac{n^2-n}{4n-6} \tag{17}$$

5. 性能評価

5.1 求解アルゴリズムと通信モデルの実装

EM-X上での通信モデルの実装として，細粒度アルゴリズムには `thread` と `spin` の両方を用いて比較し，ブロックアルゴリズムには `n_words` を用いた．AP1000+上では，細粒度アルゴリズムに `inline` を用い，ブロックアルゴリズムには `put()` を用いた．これらは，細粒度アルゴリズムではチューニングによりオーバヘッドを最大限削減して性能を追求し，ブロックアルゴリズムではユーザアプリケーションからライブラリ関数を使用するという想定に基づいている．以下では，細粒度アルゴリズムを `fine-grain` と表し，ブロックアルゴリズムを `block` と表す．EM-Xの `fine-grain` で `thread` と `spin` を区別する場合は，それぞれの表記を用いる．

各アルゴリズムの実装はC言語を使用して記述した．EM-Xでは `fine-grain` の最内ループと，`block` の内積計算の部分について，EM-Cコンパイラ[13]のアセンブリ言語出力に手を入れて最適化した．AP1000+ではすべてC言語レベルで最適化した．`block` における内積計算は，両マシンとも8重のアンローリングを施した．計算はすべて単精度（32 bit）で行い，データ転送も32 bitのワード単位で行った．

実行時間と計算量（n^2）から算出されるMFlopsを性能測定の主な指標とした．

5.2 測定結果

本節では，実際の性能測定値と，性能モデルから計算できる各理論値を示す．

図6と図7に，EM-XおよびAP1000+の問題サイズ n と性能の関係を示す．それぞれに `fine-grain` と `block` の両方が示されている．PE数は16 PE，64 PEに加え，AP1000+では256 PEでも測定した．EM-Xの64 PEの場合では，$n < 2048$ の範囲で `fine-grain` の性能が高く，$n \geq 2048$ では逆転して `block` が上回っている．AP1000+ではすべての問題サイズにおいて `block` が良い．

5.2.1 細粒度アルゴリズム (fine-grain)

EM-Xの `thread` と `spin` の1回の積和計算あたりのイテレーションの命令数は，アドレス計算，通信オーバヘッドを含め前者が11命令，後者が12命令となった．`thread` はスレッド起動に1クロックかかるため，両者の1イテレーションの実行クロック数（$t_1+o_s+o_r$ に相当）はともに12クロック（0.6 μsec）

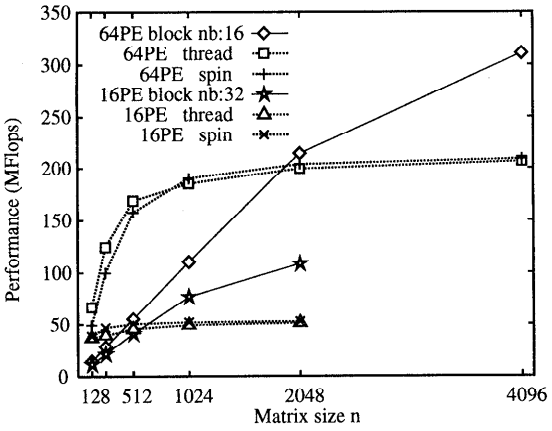


図6 EM-Xにおける問題サイズと性能の関係
Fig. 6 Relation between problem size and performance on EM-X.

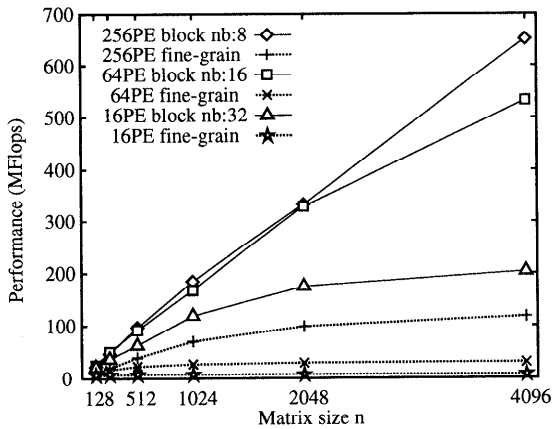


図7 AP1000+における問題サイズと性能の関係
Fig. 7 Relation between problem size and performance on AP1000+.

となる☆．データはすでに到着しているものとする．これに対し，AP1000+における **fine-grain** では1イテレーションあたり24命令であり，実行時間は測定により4.37 μsecであった．EM-Xに比べて命令数が多いのは，フラグアドレスの計算とPUTのワード書き込みによるものである．ループの構成および測定により求めた内側ループ実行時間と t_1 を，**表2** に示す．t_1 は，ループ実行時間から通信オーバヘッドを除いたものであるが，インデクス処理のためのアドレス計算等は含む．式(15)に表1の値と t_1 を代入することにより，理論的な最小実行時間 T_{cf} が求まる．T_{cf} と

☆ threadとspinのスループットがちょうど等しくなったことには大きな意味はなく，アーキテクチャの実装方法次第で大小関係は変化すると考えられる．

表2 細粒度アルゴリズムの内側ループ実行時間
Table 2 Execution time of the most inner-loop of the fine-grain algorithm.

		1 iteration	t_1
EM-X	thread	0.60	0.50
	spin	0.60	0.35
AP1000+	inline	4.37	0.27

(μsec)

表3 細粒度アルゴリズムの平均並列度と理論的性能上限
Table 3 Theoritical parallel gain and performance limit of the fine-grain algorithm.

n	R	T_{cf} (MFlops) EM-X thread	EM-X spin	AP1000+ inline
256	64.1	155.3	155.4	18.5
1024	256.1	575.6	575.7	73.5
4096	1024.1	2125.8	2125.9	213.9

問題サイズからMFlops値を性能上限として求め，式(17)の平均並列度 R とあわせて**表3** に示した．T_{cf} の算出では，平均並列度に対応してPE台数を増加させると仮定し，その場合にネットワークトポロジが L に与える影響も考慮した☆☆．

図6と図7の **fine-grain**（EM-X は **thread** と **spin**）について同一PE数で比較すると，すべてのサイズにおいてEM-Xの方が性能が高く，64PE，$n = 4096$ ときにAP1000+に対して約6.8倍となっている．一方，EM-Xの **thread** と **spin** の比較では，$n = 128$ の場合に **thread** が **spin** に対して36%高い性能となっている．n が大きくなると，その関係はわずかに逆転する．

5.2.2 ブロックアルゴリズム（block）

ブロックサイクリック分割におけるPE配置（プロセスグリッド）は，縦横同数の構成（$P = p \times p$）とした．$p \times p$ 以外の構成も可能であるが，$p \times p$ あるいはそれに近い構成（PE数が N^2 でない場合）の場合に最も適度な分散が行われ，高い性能が得られたため，測定値として採用した．ブロックサイズも最も高い性能を示すものを選んだ．式(5)から最適なブロックサイズを理論的に求めることは可能であるが，導出は省略する．

図6と図7の **block** についてEM-XとAP1000+を比較すると，すべての問題サイズにおいてAP1000+の方が性能が高く，64PE，$n = 4096$ 時にEM-Xに対して約1.7倍となっている．

図8 と**図9** に，ブロックサイズと性能の関係を示

☆☆ EM-X は $O(\log P)$，AP1000+は $O(P^{\frac{1}{2}})$ を適用した．

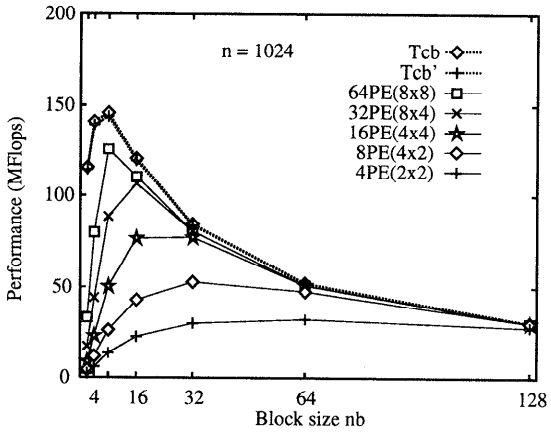


図8 EM-Xにおけるブロックサイズと性能の関係
Fig. 8 Relation between block size and performance on EM-X.

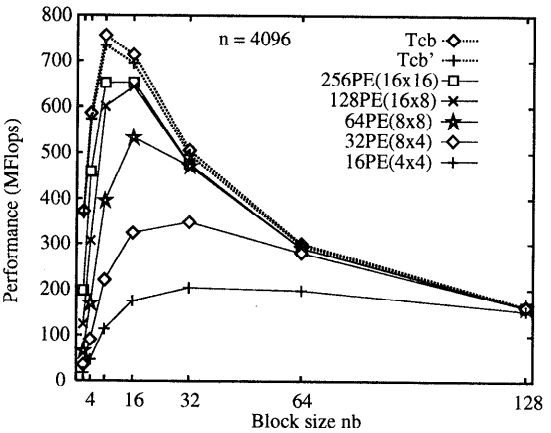


図9 AP1000+におけるブロックサイズと性能の関係
Fig. 9 Relation between block size and performance on AP1000+.

す．EM-Xでは $n = 1024$ とし，PE数は4PEから64PEまでの結果を示した．AP1000+では $n = 4096$ とし，PE数は16PEから256PEとした．また，式(5)に通信パラメータを与えて得た性能上限曲線（T_{cb}, T_{cb}'）も示した．T_{cb} は演算と通信がオーバラップできる場合，T_{cb}' はオーバラップできない場合を示す．AP1000+では，図の条件において，オーバラップできない場合に対して最大約2.9%の性能向上が得られていることを示している．EM-Xでは，オーバラップできるとすると最大約1.4%の性能向上となる．同式に含まれる t_{reg}, t_{diag} は，あらかじめブロックの部分だけ測定したもの（**表4**）を用いた．表4のAP1000+での測定の際，キャッシュの振舞いが **block** と同様になるよう注意した．

表4 1ブロックの計算時間
Table 4 Execution time of a block.

	EM-X		AP1000+	
n_b	t_{reg}	t_{diag}	t_{reg}	t_{diag}
2	5.70	4.85	2.12	1.64
4	12.0	9.50	5.25	3.87
8	26.2	21.8	12.2	11.0
16	72.6	54.7	39.8	29.5
32	230.2	152.9	142.4	90.4
64	804.6	478.9	537.5	307.7
128	2990.2	1649.3	2086.8	1122.4

(μsec)

表5 ブロックアルゴリズムの平均並列度
Table 5 Theoritical parallel gain of the block algorithm.

N_B	n_b		R
	$n = 1024$	$n = 4096$	
2048		2	682.9
1024		4	341.6
512	2	8	170.9
256	4	16	85.6
128	8	32	42.9
64	16	64	21.6
32	32	128	10.9
16	64		5.57
8	128		2.91

これらの図ではさらに，式(12)で示される理論的な平均並列度との関係を見ることができる．同式から求めた平均並列度 R を**表5**に示す．たとえば，$n = 4096$, $n_b = 16$ 時に85.6，$n_b = 32$ 時に42.9である．これらの値は，それ以上PE数を増加させても効果はないことを意味し，図9においてそれぞれ128PE，64PEで性能が飽和していることを理論的に示している．図8でも同様の関係を見ることができる．

5.3 考　　察

5.3.1 細粒度アルゴリズム

fine-grain の性能は，式(15)の n の項に大きく影響される．同項を構成する o_s, o_r, t_1, L は，EM-Xではすべて0.5μsec以下であるのに対し，AP1000+では o_s と L が5μsec前後と大きい．このことがEM-XとAP1000+の性能差の原因となっており，計算時間 t_1 の影響は小さい．EM-Xでは細粒度支援機構が有効に働いているといえる．一方，AP1000+では，PUTの起動時間と，MSC+によるDMAのセットアップ時間が，これらのパラメータ値を大きくしていると考えられる．

spin と **thread** は，式(15)から導かれる性能としてはほとんど差がないはずであるが，実際の性能が異なるのは，**thread** のスレッド生成オーバヘッドと，**spin** の外側ループのオーバヘッドに起因する．それ

ぞれ n に対する感度が異なるため，性能に対する影響の大きさが n によって変化する．表3に示した性能上限と，それと同じ条件の測定値の差は，これらのオーバヘッドの存在が原因である．n が大きくなるほど全体の計算時間に占めるオーバヘッドの割合は減るため，その影響は少なくなり，両アルゴリズムは同じ飽和性能に漸近する．

thread に用いられているスレッド起動方式は，EM-Xのデータ駆動モデルに基づいており，本来，依存関係が実行時に動的に決まるような問題に対して特に有効に働くものである．しかしながら，密行列計算のようにそれが静的に決まる問題においても，受信オーバヘッド o_r を小さくできるため有用である．一方，**spin** は待合せ順を限定するが，本論文で扱う密行列計算においては問題ない．o_r がやや大きい代わりにスレッドの中断・起動のオーバヘッドがないため，性能は **thread** と同程度となった．これらの方式は，ともに送信オーバヘッド o_s および通信レイテンシ L が小さいことが高性能に結び付いている．

なおAP1000+の予備評価で示された受信オーバヘッドの隠蔽は，通信性能の観点におけるスーパスカラ方式の利点を示唆するものである．

5.3.2 ブロックアルゴリズム

block では，ブロックサイズ n_b がある程度大きいと，式(5)の中で t_{reg}，t_{diag}（表4）が支配的となる．EM-X と AP1000+の測定結果にも t_{reg}，t_{diag} の差が直接表れており，これはさらにPEの単体性能（EMC-Y 20 MHz 対 SuperSPARC 50 MHz）に起因している．通信性能はEM-Xの方が高いが，n_b が大きくなると演算時間の比率が高まるため，その影響は小さくなる．AP1000+では，演算と通信のオーバラップ効果が実際に存在することが理論的に示されるが，同様の理由によりその影響は顕著とはいえない．EM-Xにおいてさらにその効果が小さいのは，通信スループットが高く，β が小さいためである．

n_b を小さくしていくとある点を境に T_{cb} が逆に増加するのは，次の2つの理由による．1) ブロック内のループオーバヘッドが増加する．2) 通信オーバヘッド（式(6)あるいは式(7)の o_s および α の係数）が増加する．

T_{cb} あるいは T_{cb}' は，PE台数 $\geq R$ となった場合の，PE台数に対する飽和性能の理論値でもあるが，図8，図9によれば n_b が小さくなるに従い誤差が大きくなっている．これは，t_{reg}，t_{diag} の測定の際に，**block** におけるブロックの外側のループオーバヘッドが考慮されていないためである．これを考慮するためには，より緻密なモデルを立てる必要がある．また，式(12)の導出の際，$t_{reg} \simeq 2t_{diag}$ という近似を用いたが，実際には n_b が小さくなるとブロック内のループオーバヘッドのために両者の差は小さくなるため，R には誤差が含まれる．この誤差が問題となるような n_b では，T_{cb} による制限が強くなるため，実際に R を参照してPE台数を設定することはないと考えられる．

5.3.3 性能の外挿

fine-grain での実機を超えるPE台数の性能予測は，表3で与えられている．たとえば $n = 4096$ の場合，問題に内在する並列度として1024.1が得られ，1024台までのPEで有効な並列処理を行えることを示している．また同表により，1024 PE の EM-X で得られる性能は 2125.9 MFlops と与えられる．一方 **block** では，表5によって $n = 4096$ において1024台のPEは有効利用できないことが分かる．その場合の予測最大性能は，式(5)に $n_b = 8$ を与えたときの581.5 MFlopsにとどまり，**fine-grain** の方が高い最大性能を得られる．

6. おわりに

本論文では，三角方程式の求解問題を細粒度および粗粒度アプローチによって分散メモリ型並列計算機に実装し，それぞれのアルゴリズムの特性と通信パラメータの関係を，計算モデルと実験によって明らかにした．また，通信オーバヘッドに起因する性能の限界と，並列度の制限による有効PE数を理論的に示した．

与えられた計算機アーキテクチャが有効となる問題領域を決定するため，1) 通信パラメータ，2) 通信と演算のオーバラップ効果，について，性能に与える影響を調べた．その結果，以下の性質を定量的に示した．

- 低オーバヘッドの細粒度支援機構が有効となるのは次の場合である．
  - PE台数が一定であれば，問題サイズが小さい場合．
  - 問題サイズが一定であれば，十分多数のPE台数が利用できる場合．
- この条件を満たさない場合は，通信性能より逐次演算性能が重要となる．
- ブロックアルゴリズムでは，ブロックサイズが大きくなるほど通信時間の割合が小さくなるため，通信と演算のオーバラップ効果は顕著ではない．

細粒度アプローチが有効となる問題領域において，高い並列処理効率を得るためには，通信オーバヘッドの削減が大きな効果をもたらす．本論文で取り上げた

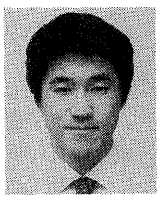
問題のように静的に通信順序が決まるものについては，受信同期方式として，特別なハードウェアを必要としないspin waitでよく，オーバヘッドも小さい．したがって，特に送信起動時間と通信レイテンシの削減が重要となる．それを実現する方法の1つが，EM-Xのように要素プロセッサ内で通信と演算が密に融合するアーキテクチャである．逆にそれ以外の領域では，細粒度支援機構は必ずしも必要ではなく，AP1000+のように粗粒度の通信サポートで十分といえ，高速な逐次実行能力を持つことがより重要である．

謝辞 本研究を遂行するにあたり，ご指導，ご討論いただいた大蒔和仁情報アーキテクチャ部長ならびに同僚諸氏，電気通信大学の本多弘樹助教授に感謝いたします．

参考文献

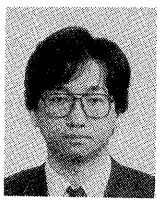
1) 今村：並列言語 ADETRAN4 での行列計算—VPP500 上の処理系実装とその応用，*JSPP '96*, pp.17–24 (1996).
2) 小国，村田，三好，ドンガラ，長谷川：行列計算ソフトウェア—WS，スーパーコン，並列計算機，6章，丸善 (1995).
3) Dongarra, J.J., Duff, I.S., Sorensen, D.C. and Vorst, H.A., 小国（訳）：コンピュータによる連立一次方程式の解法—ベクトル計算機と並列計算機，3章，丸善 (1992).
4) 坂根，児玉，小池，佐藤，山名，坂井，山口：EM-Xによる密行列計算の細粒度並列処理—ウェーブフロント型並列性の効率的実行，*JSPP '97*, pp.29–36 (1997).
5) Choi, J., Dongarra, J.J., Ostrouchov, L.S., Petitet, A.P., Walker, D.W. and Whaley, R.C.: The Design and Implementation of the ScaLAPACK LU, QR and Cholesky Factorization Routines, Oak Ridge National Laboratory, ORNL/TM-12470 (Sep. 1994).
6) Culler, D.E., Karp, R., Patterson, D., Sahay, A., Schauser, K., Santos, E., Subramonian, R. and Eicken, T.V.: LogP: Towards a Realistic Model of Parallel Computation, *Proc. 4th ACM SIGPLAN Symposium on Principles & Practice of Parallel Programming*, pp.1–12 (1993).
7) 高畠，大澤，弓場，佐藤，山口：細粒度並列アーキテクチャ用 SISAL コンパイラにおける並列粒度調整方式，情報処理学会論文誌，Vol.39, No.6, pp.1709 1717 (1998).
8) 當山，堀口：並列アルゴリズムの動作解析のための実用並列計算機モデル LogPQ，情報処理学会論文誌，Vol.39, No.6, pp.1766–1774 (1998).
9) Kodama, Y., Sakane, H., Sato, M., Yamana, H., Sakai, S. and Yamaguchi, Y.: The EM-X Parallel Computer: Architecture and Basic Performance, *Proc. 20th Int. Symp. on Computer Architecture*, pp.14–23 (1995).
10) 石畑，堀江，清水，林，小柳，今村，白木：AP1000+：デザインコンセプト，情報処理学会研究報告，94-ARC-107, pp.153–160 (1994).
11) 小柳，白木，今村，林，清水，堀江，石畑：AP1000+：メッセージハンドリング機構 (I)—ユーザレベルインターフェース，情報処理学会研究報告，94-ARC-107, pp.161–168 (1994).
12) 白木，小柳，今村，林，清水，堀江，石畑：AP1000+：メッセージハンドリング機構 (II)—システムレベルインターフェース，情報処理学会研究報告，94-ARC-107, pp.169–176 (1994).
13) 佐藤，児玉，坂井，山口：並列計算機 EM-4 の並列プログラミング言語 EM-C，*JSPP '93*, pp.183–190 (1993).

(平成10年9月9日受付)
(平成11年3月5日採録)



坂根 広史（正会員）

1966年生．1990年山口大学工学部電子工学科卒業．1992年電気通信大学大学院博士前期課程電子工学専攻修了．同年通商産業省工業技術院電子技術総合研究所入所．以来，並列計算機のアーキテクチャおよびその性能評価の研究に従事．1998年より，在職のまま電気通信大学大学院情報システム学研究科博士後期課程に在学．電子情報通信学会，神経回路学会各会員．



児玉 祐悦（正会員）

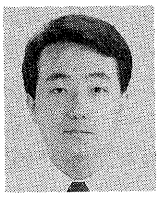
1962年生．1986年東京大学工学部計数工学科卒業．1988年同大学大学院情報工学専門課程修士課程修了．同年電子技術総合研究所入所．現在，同所情報アーキテクチャ部主任研究官．データ駆動型計算機等の並列計算機システムの研究に従事．特にプロセッサアーキテクチャ，並列性制御，動的負荷分散，並列探索問題等に興味あり．情報処理学会奨励賞，情報処理学会論文賞（1990年度），市村学術賞（1995年）等受賞．電子情報通信学会，IEEE 各会員．

建部 修見（正会員）

1969年生．1992年東京大学理学部情報科学科卒業．1997年同大学大学院理学系研究科情報科学専攻博士課程修了．同年電子技術総合研究所入所．並列数値アルゴリズム，並列計算機システム，広域分散計算の研究に従事．ハイパフォーマンスコンピューティングに興味を持つ．理学博士．日本応用数理学会，ACM各会員．

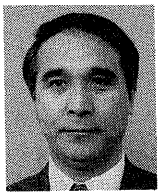
小池 汎平（正会員）

1961年生．1984年東京大学工学部電子工学科卒業．1986年同大学大学院情報工学専門課程修士課程修了．1989年同博士課程単位取得退学．同年同大学工学部電気工学科助手，1991年同講師，1995年同助教授．1996年電子技術総合研究所入所．この間1994年より1996年までマサチューセッツ工科大学コンピュータサイエンスラボラトリ客員研究員．工学博士．並列処理計算機の研究に従事．1992年度元岡賞受賞．



山名 早人（正会員）

1964年生．1987年早稲田大学理工学部電子通信学科卒業．1989年同大学大学院修士課程修了．1993年同大学院博士課程修了．1989年より同大学情報科学研究教育センター助手．工学博士．1993年より電子技術総合研究所に勤務．1993年情処研究奨励賞，1995年山下記念研究賞受賞．コンピュータアーキテクチャ，並列化コンパイラ，広域分散処理の研究に従事．著書「超並列コンピュータ入門」（共著，オーム社）等．電子情報通信学会，ACM，IEEE各会員．



山口 喜教（正会員）

1972年東京大学工学部電子工学科卒業．同年電子技術総合研究所入所，計算機方式研究室長等を経て，1999年筑波大学電子・情報工学系教授（電子技術総合研究所併任），工学博士．高級言語計算機，並列計算機アーキテクチャ，並列実時間システム等の研究に従事．1991年情報処理学会論文賞，1995年市村学術賞受賞．著書「データ駆動型並列計算機」（共著，オーム社）．IEEE Computer Society，ACM，電子情報通信学会各会員．

弓場 敏嗣（正会員）

1966年神戸大学大学院工学研究科修士課程修了．（株）野村総合研究所を経て，1967年通商産業省工業技術院電気試験所（現，電子技術総合研究所）に入所．以来，計算機のオペレーティングシステム，見出し探索アルゴリズム，データベースマシン，データ駆動型並列計算機等の研究に従事．その間，知能システム部長，情報アーキテクチャ部長等を歴任．1993年より，電気通信大学大学院情報システム学研究科教授．並列処理の科学技術一般に興味を持つ．工学博士．情報処理学会，電子情報通信学会，日本ソフトウェア科学会，日本ロボット学会，ACM，IEEE-CS各会員．