細粒度通信機構を用いたRadixソートの実行

児玉祐悦[†] 坂根広史[†] 佐藤三久[††]
山名早人[†] 坂井修一[†,†††] 山口喜教[†]

EM-Xは，ワード単位の細粒度通信を命令実行パイプラインでサポートする分散メモリ型の高並列計算機である．リモートメモリアクセスをスレッド処理とオーバラップすることにより通信スループットを向上させるとともに，複数のスレッドを効率良く切り替えることにより通信レイテンシにも強いアーキテクチャとなっている．これにより並列処理の適用分野の拡大を目指している．現在80プロセッサから構成されるシステムが完成し，実機による評価を行っている．その一例として，radixソートを取り上げ，並列性能の評価を行うとともに，他の並列計算機との比較を行った．この結果，細粒度通信を効率良くサポートすることにより，粗粒度通信時にみられるプロセッサ台数を増加させた場合のネットワーク混雑による性能低下を解消し，非常に高いスケーラビリティを確認できた．

Parallel Execution of Radix Sort Programs Using Fine-grain Communication

YUETSU KODAMA,[†] HIROFUMI SAKANE,[†] MITSUHISA SATO,[††]
HAYATO YAMANA,[†] SHUICHI SAKAI[†,†††] and YOSHINORI YAMAGUCHI[†]

EM-X is a highly parallel computer with a distributed memory. It supports fine-grain communication, whose size is two-word fixed, on an instruction execution pipeline. It achieves high communication throughput by overlapping remote memory access with thread execution, and tolerates communication latency by rapid switching of threads. We developed an 80 processor system of EM-X, and are evaluating its architectural features on the system. In this paper, we execute radix sort programs to evaluate the parallel performance of EM-X and compare the results with other parallel computers. The results show that fine grain communication achieves very good scalability, while coarse grain message passing decrease the performance on a large number of processors because of contentions on a network.

1. はじめに

多数のプロセッサを結合する高並列計算機では，分散共有メモリをサポートするアーキテクチャが多くなっている．これは，分散メモリアーキテクチャの持つ，実装面の利点や局所処理時の性能向上とともに，共有メモリアーキテクチャの持つプログラミングの容易さや逐次プログラミングからの移行のしやすさの両方の利点を持っているためといえる．しかし，分散共有メモリといっても共有メモリから発展したものと，分散メモリから発展したものの2つに大きく分けられる．前者は，バス共有の共有メモリで主流となったスヌープキャッシュによるコヒーレント維持管理を発展させたものであり，クラスタ構造を持つものが多い．すなわち，クラスタ内はスヌープキャッシュ，クラスタ間はディレクトリ法などを用いることにより，徹底してコヒーレント維持をサポートする方式である．現在も，効率的なクラスタ間コヒーレント維持管理手法は研究が進められている[1),2)]．一方，後者は分散メモリのメッセージ通信型の並列計算機から発展したもので，単にグローバルなアドレス空間をサポートしたものから，メッセージのバッファリングを不要としたリモートメモリアクセスをサポートしたものまでいろいろな計算機が開発されている．これらの計算機はメッセージ通信を基本としているため，通信のセットアップオーバヘッドが大きく，効率の良い通信を行うためにはメッセージサイズを大きくする必要がある．しかし，並列処理の適用分野を広げるためには，少量の

† 電子技術総合研究所情報アーキテクチャ部
Computer Science Division, Electrotechnical Laboratory
†† 新情報処理開発機構
Real World Computing Partnarship
††† 筑波大学電子・情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba

データをレイテンシを少なくして，あるいは隠蔽して通信を行うことが重要となる．

現在我々は，従来のメッセージ通信方式と異なり，レイテンシを重視した細粒度通信を行う並列計算機EM-X[3),4)]を開発中である．EM-Xでは，プロセッサ間通信の単位を細粒度なワード単位のパケットに限ることにより，その機能を命令実行パイプラインと融合し低レイテンシと高スループットを両立させている．また，プログラムを通信レイテンシが現れないスレッドという単位に分割し，スレッド内をRISCプロセッサにより高速に実行するとともに，レイテンシが必要な場合にはスレッドを他のアクティブなスレッドに切り替えることにより，そのレイテンシを隠蔽するマルチスレッド実行を効率良くサポートしている．共有メモリ型のプログラミングにおいても，分散配置した配列データをワード単位で不規則にアクセスしたり，各プロセッサで同期をとるような状況はしばしば生じるが，そのような状況に対しても細粒度通信およびマルチスレッド実行により効率良く適応できるため，並列処理の適用分野を広げていくことが期待できる．

本論文ではこの高並列計算機EM-Xのアーキテクチャを，共有メモリプログラミングへの適用という観点から評価することを目的とする．共有メモリプログラミングとしてradixソートを用いる．radixソートはスタンフォード大学でまとめられている共有メモリ並列ベンチマークSPLASH-2[5)]にも取り上げられており，効率的な並列ソートアルゴリズムとして知られている．メモリアクセスの競合を抑えることができれば，適当なデータサイズに対しプロセッサ台数の増加に見合った非常にスケーラブルな性能向上を期待できる．また，共有メモリプログラミングにより細粒度なメモリアクセスを必要とする．今回の評価結果はメモリ書き込みを主な通信パターンとする他のアプリケーションにも適応できると考えられる．

まず，2章においてEM-Xのアーキテクチャについて述べ，3章でradixソートの並列化について述べる．4章においてEM-Xをはじめとして複数の並列計算機によりradixソートを実行し，比較を行うとともに，EM-Xのシステム性能について評価を行う．特に，EM-Xと同様のマルチスレッド機構を持つEM-4[6)]と比較し，EM-Xで取り入れた新たな分散共有メモリ向け機能について性能評価を行う．また，分散共有メモリサポート機構を持つ従来型のメッセージ通信型並列計算機AP1000+および複数の強力なメモリバスを持つ共有バス型並列計算機CS6400との比較により，共有メモリ性能のスケーラビリティについて評価を行う．最後に5章でまとめと今後の課題について述べる．

2. EM-Xアーキテクチャ

EM-Xでは，プログラムを通信レイテンシが現れない逐次処理可能なスレッドという単位に分割し，このスレッドを順次切り替えながら処理をするマルチスレッド実行を行う．ここで，通信レイテンシとは，リモートメモリ参照やリモート関数呼び出しの結果待ちのような遠隔処理を行う際に現れるレイテンシのことであり，一般にネットワークでのパケット衝突や各要素プロセッサ（PE）における負荷の変動のため静的に見積もることは難しい．各スレッドの実行はRISCパイプラインにより高速に実行される．各スレッドの実行状況を表すには，実行しようとする命令の開始アドレスと，その実行に必要な引数や途中結果などを格納する実行環境のベースアドレスのペアがあればよい．EM-Xではこれをcontinuationと呼ぶ1つのパケットとしてまとめている．このcontinuationパケットがプロセッサに到着すると，それに対応したスレッドがハードウェアにより自動的に起動される．このように各スレッド間をcontinuationパケットで結合することにより並列プログラムの実行が行われる．

EM-Xでは，continuationを含めてすべてのプロセッサ間通信を2ワードからなる細粒度のパケットを用いて行っている．これにより，パケット出力命令を$<send\ reg_{data}, reg_{addr}>$のように2入力オペランドの通常のレジスタ演算命令と同様に実装することが可能となり，命令実行パイプラインとの融合が容易となる．このような単純な*send*命令を備えることにより，余分なセットアップ時間を必要とせずパケット出力は1クロックで終了する．他の並列計算機では，通信レイテンシの多くの時間を通信のためのセットアップやパケット変換などに費やしているのに比べ，EM-Xではパイプラインレベルでパケットを生成しているため，ほとんどハード的なネットワーク転送時間だけでパケットを転送することが可能であり，低レイテンシを実現している．*send*命令ではパケットアドレスとして，PE番号（10 bit）とPE内局所アドレス（22 bit）からなる1ワード（32 bit）のグローバルアドレスを用いることができ，局所アドレスのポインタ参照と同様にリモートアドレスのポインタ参照を効率良く行える．また，*send*命令でのアドレス指定にはメモリ参照命令と同様に，$reg_{base} + imm_{disp}$というベースアドレッシングを指定できるため，柔軟なリモートメモリアクセスが可能となり，高スループットを実現している．このようにEM-Xでは細粒度通信を用いるこ

とにより，低レイテンシと高スループットを両立させている．

ネットワークからやってきたパケットは通常プロセッサ内部の入力パケットバッファへ格納され，順番に処理されていく．この入力パケットバッファは，プロセッサ内部の小容量バッファとメモリ上のバッファから構成されているが，待避と復帰のパケット管理はハードウェアによって行われており，チップ内バッファのオーバフロー処理のためにスレッド実行が中断されることはない．このため，仮想的な大容量バッファが存在すると考えることができる．このようにパケットのFIFO管理をハードウェア化することにより受信時の割込みを不要とし，実行中のスレッドはパケット受信のために中断されることはない．

しかし，この方式ではリモートメモリ読み出しパケットがやってきたとしても，現在実行中のスレッドが終了しない限りその処理は待たされる．共有メモリ型プログラミングでは分散配置された配列へのリモートアクセスレイテンシが重要であるため，このような状況は特に問題となる．これを解決するため，EM-Xではリモートメモリの書き込みおよび読み出しに対しては特別に，スレッド実行とは独立に処理できる機構を備えている．これにより，連続するリモートメモリ書き込みを行ってもパケットバッファが溢れることはなく，また，リモートメモリ読み出しのレイテンシもかなりの程度予測可能となり，プリフェッチの技巧などが行いやすくなる．このようなスレッド実行とは独立に書き込み，読み出しを行う機構を直接リモートメモリアクセス機構と呼ぶ．一方，これらのリモートメモリアクセスはスレッド実行とは独立に処理されるため，そのメモリに対するアクセスの同期を保証することはできない．そのため，スレッド起動によりメモリ参照を行うリモートメモリ書き込みおよび読み出しパケットも用意されている．

我々は現在80プロセッサからなるEM-Xを開発し，現在実機による評価を行っている．**図1**にその全体写真とプロセッサボードの写真を示す．1枚のボードに5プロセッサを搭載し，コンパクトな筐体に仕上げている．これはネットワークノードを含めて要素プロセッサ（PE）をシングルチップに納めていることによるもので，各PEはプロセッサチップとメモリSIMMおよび若干のバッファICのみからなっている．EM-Xはネットワークスループットを確保するため，ネットワークの各ポートは38 bitのデータ幅を持っている．ネットワークトポロジーはサーキュラオメガ網[7]であり，各ネットワークノードは3入力3出力のクロスバースイッチとなっており，そのうちの1対がプロセッサに接続されている．ボード内の各PEはリング接続されており，残り5入力5出力のうち，3/5を上下に別れたサブ筐体内のバックプレーンで，2/5を上下のサブ筐体を結ぶケーブルで接続している．クロック周波数は20 MHzとして設計されている☆．

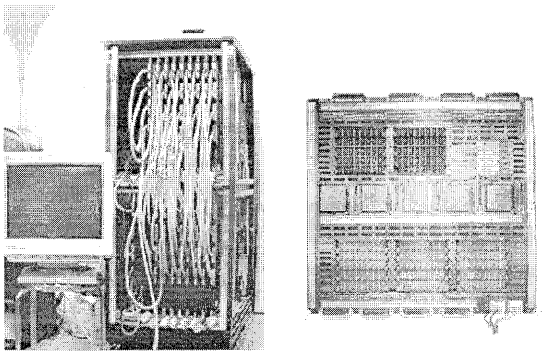


図1 EM-X 80プロセッサシステムとプロセッサボード
Fig. 1 EM-X 80 PE system and a PE board.

80PEシステムには16枚のPEボードのほか，ホスト計算機と接続するためのインタフェース基板，計算結果を直接表示するためのフレームバッファ基板が搭載されている．これらの基板は，ケーブル接続の途中に自由に追加可能である．ホスト計算機にはSparcStation2を用いており，S-BUSによりインタフェース基板と接続されており，メモリマッピング機構により直接パケットデータの入出力を行える．現在EM-X本体にはdiskは搭載していないためファイルへのアクセスはホスト経由のアクセスとなるが，1〜2 MB/s程度のスループットを確保している．また，フレームバッファ基板は1024 × 768の24 bitカラーのフレームメモリを4面装備しており，高速な表示が可能である．またビデオ信号の実時間取込みも可能となっている．

3. Radixソートの並列化

radix（基数）ソートとは，各データをn-進数で表し，各桁ごとのソートを繰り返すことにより全体のソートを行う方法である．効率化のために2の巾乗をnとすることが多い．各桁はたかだかn通りしかないので，その各値の個数を数え上げることによりソート後の位置を計算することができる．クイックソートなどのようにデータどうしの比較を行わないため実行時

☆ 現在プロセッサボードの調整中のため，16 MHzで動作させている．しかし，本論文では20 MHzで動作しているものとして速度を求めている．

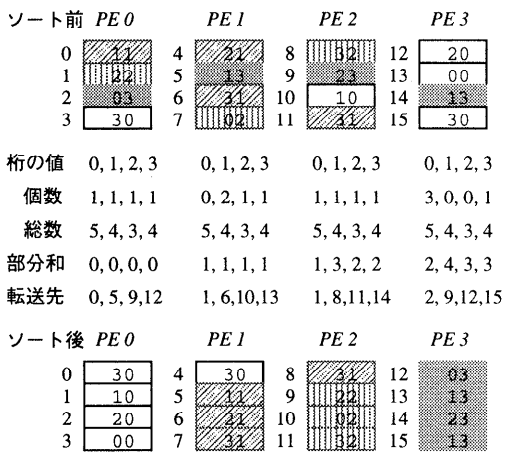


図2 並列radixソートアルゴリズム
Fig. 2 parallel radix sort algorithm.

間の見積もりが容易であり，固定長のデータに対する非常に効率的なソートとして知られている．ただし，radixソートではデータと同じサイズのワークエリアを必要とし，かつランダムなワード単位の書き込みが生じるためメモリサイズやキャッシュ性能の影響を受けやすい．

データを各要素プロセッサ（PE）に分散配置しておくことにより，分散共有メモリの並列計算機でradixソートを並列化することは容易であり，かつ，メモリアクセス競合がなく十分なデータ数があればプロセッサ台数に対して非常にスケーラブルな性能を期待できる．また，radixソートはスタンフォード大学でまとめられている共有メモリ並列ベンチマークSPLASH-2[5]にも取り上げられている．**図2**にプロセッサ数を4，データ数を16，radixを4とした場合の最下位桁のソートの様子を示す．ソートの手順は以下のとおり．

(1) 各PEでローカルに最下位桁が0，1，2，3である要素数を数える．
(2) (1)の結果の全PEの総和を求める（総和）．
(3) (1)の結果の自分よりPE番号の小さいPEまでの部分和を求める（部分和）．
(4) 上の2つの値から各値の転送開始アドレスを求める（転送先）．
(5) 各データを上の転送アドレスから転送する．

たとえば，PE2の先頭要素（index=8）の転送開始アドレスを求めるには次のようにすればよい．1）ソートキーとなる桁の値は2．2）ステップ(2)で求めた各要素数の総和から最下位桁が0，1である要素が全体で5+4=9個あることが分かる．3）ステップ(3)で求めた部分和から最下位桁が2である要素がPE0，PE1に2個あることが分かる．3）したがってPE2の最下位桁が2である要素は転送先バッファの9+2=11番目の要素から転送開始すればよい．また，データ転送時に各PEのデータ数が同じになるように負荷分散を行うことにより，効率の良い負荷分散が可能である．

全PEでの総和は，ハードウェアのバリア同期機構で求めることのできる並列計算機もあるが，部分和を求める機構まで備えた計算機はない．そのため，バタフライネットワークなどをソフト的にエミュレートする方法が一般的であり，PE台数のlogオーダで処理が行える．

データ転送はワード単位で行われ，転送アドレスも不規則である．このため，セットアップに時間がかかるようなメッセージ通信機構の場合は非常に効率が低下する．そのような計算機で通信を効率良く行うためには，ワード単位に分かれているデータをいったんブロックデータにまとめる必要があるが，これらの処理に余分なメモリアクセスが必要となり，性能低下の原因となる．

radixソートの処理時間は，通信の衝突などを無視した場合，データの内容には依存せず，データ数とPE台数およびradixの大きさにより定式化できる．ローカルな要素数の数え上げの1ワードあたりの処理時間を Tl，総和と部分和を求めるグローバルな処理に要する1データあたりの時間を Tg，1ワードあたりの転送時間を Tw，総データ数を N，PE台数を P，データのbit長を s，radixを r とする．1PEあたりのデータ数は $n=N/P$，全体をソートするのにかかる繰返しの数は $R=s/\log r$ であるので，全体の処理時間は次のとおり．

$$\begin{aligned} T &= R\times(nTl+rTg+nTw) \\ &= N/(P\log r)\times s(Tl+Tw) \\ &\quad +r/\log r\times(sTg) \end{aligned}$$

第2項は総データ数 N とは独立であるため，N を十分大きくすることにより第2項を無視することが可能であり，非常にスケーラブルな並列化が行えることが分かる．また，radixが大きいほど第1項は小さくなるが，第2項は大きくなるため適切な値をとることが必要となる．この値は計算機の各並列プリミティブ性能やデータサイズなどにより変化する．たとえばローカル処理や転送処理に比べてグローバルな処理が遅い場合はradixは小さな値に抑える必要があるが，グローバルな処理が速いほどradixを大きくとることが可能となり，全体の速度向上に役立つ．

4. 並列 Radix ソートの実行と評価

前章の並列 radix ソートを，我々の開発した細粒度通信機構を持つ並列計算機 EM-X および他の並列計算機で実行し評価を行う．他の並列計算機としては EM-X と同様のマルチスレッド機構を持つ EM-4，リモートメモリアクセスをサポートしたメッセージ通信型並列計算機である AP1000+，強力なメモリバスを複数有する共有バス型並列計算機である CS6400 である．それぞれ同様のアルゴリズムでプログラムを作成した後，最適化を行い，実機を用いた実行を行っている．本稿で用いた各プログラムリストおよび実行結果の詳細データは EM-X のホームページ（http://www.etl.go.jp/etl/comparc/EM-X）で公開している．

最初に各並列計算機における逐次処理性能を逐次 radix ソートにより比較し，radix および問題サイズによる影響を述べる．次に各並列計算機での並列プログラムの生成および実行についてまとめる．その中で特に EM-X と EM-4 との比較を行い，EM-X で改良された分散共有メモリ機構の評価を行う．最後に，各並列計算機において，プロセッサあたりのデータサイズを一定にして，プロセッサ台数を増やした場合のスケーラビリティについて比較を行う．

4.1 各並列計算機における逐次 Radix ソートの実行

各計算機のプロセッサ単体性能を**表 1** に，各並列計算機の 1 プロセッサを用いた場合の逐次 radix ソートの実行結果を**図 3** に示す．x 軸はデータ量，y 軸は 1 秒あたりのソートデータ量である．radix を 64 と 256 の場合を示した．各計算機で最大データ量が異なるのは搭載しているメモリ量が異なるためである．EM-X，EM-4 ではキャッシュがないため非常になめらかな結果となった．データは 32 bit 整数なので，radix が 64 で 6 回，256 で 4 回のステップ実行が必要であり，十分大きなデータに対しての実行結果もそれを裏付けている．

一方，SuperSPARC をプロセッサとする計算機では，キャッシュミスおよび TLB ミスの影響が現れる．radix ソートでは主に 3 種類のメモリデータを扱う．1 つ目がソートすべき元データ（soc）で，連続的に読み出されるため空間的局所性の効果が期待できる．2 つ目が各 radix に対応した書き込みポインタからなる配列（wp）で，データサイズは radix の大きさと同じであり他のデータに比べて十分小さく，繰り返しアクセスされるため，時間的局所性の効果が期待できる．3 つ目がソートした結果を格納するデータ（dst）で，ワード単位にランダムにアクセスされるが，各 radix ごとに見ると空間的には連続して書き込みが行われるため，空間的局所性の効果が期待できる．ただし，SuperSPARC では write-alloc を行わないため 1 次キャッシュから溢れた段階でキャッシュ効果はなくなり，ライトバッファの効果のみとなる．データサイズが 4 K ワードから 64 K ワードまでの AP1000+ と CS6400 の差が動作周波数以上に大きいのは 2 次キャッシュの効果である．

データサイズが 64 K ワードを超えると AP1000+ で性能が低下するのは TLB ミスのためである．SuperSPARC は 64 本の TLB をキャッシュしており，AP1000+では TLB は 4 K バイトのページサイズである．radix が 256 の場合，データサイズが 64 K ワードを超えると，書き込みを行うページ数が 64 を超え，TLB のスラッシングが始まり性能低下となる．データがすでにソートされていると，radix が 64 でもこの TLB のスラッシングの影響が見られることは確認したが，データがランダムであるため図 3 ではスラッシングの影響はあまり見られない．CS6400 ではデータサイズが 64 K ワードを超えると，この TLB のスラッシングの影響が現れ，さらに 256 K ワードを超えると，2 次キャッシュの溢れの影響が現れる．そのため

表 1 要素プロセッサの性能諸元
Table 1 Spec. of processing element.

	プロセッサ	動作周波数	データキャッシュ
EM-X	EMC-Y	20 MHz	なし
EM-4	EMC-R	12.5 MHz	なし
AP1000+	SuperSPARC	50 MHz	L1:16 K, L2:なし
CS6400	SuperSPARC	60 MHz	L1:16 K, L2:2 MB

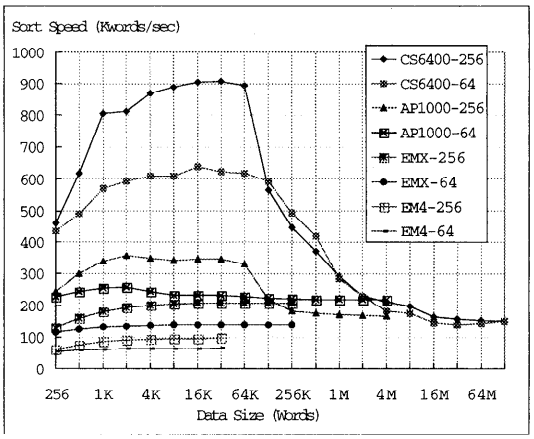


図 3 逐次 radix ソート
Fig. 3 Sequential radix sort.

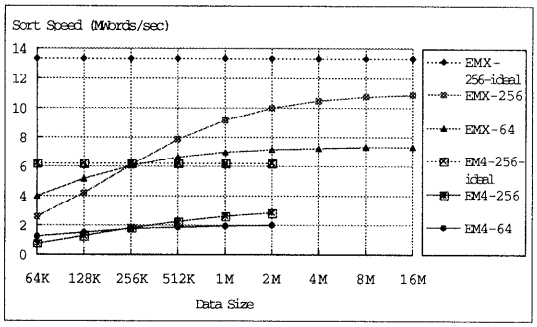


図4 EM-XおよびEM-4での並列radixソート（64PE）
Fig. 4 Parallel radix sort on EM-X and EM-4 (64PE).

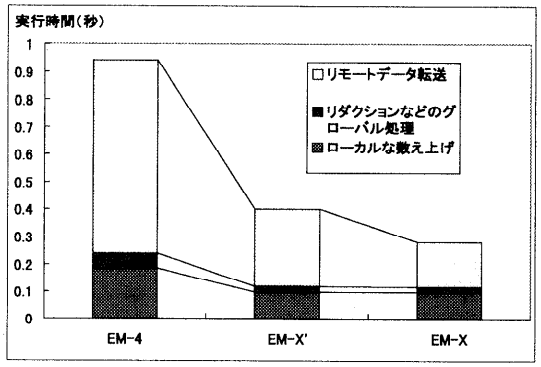


図5 EM-Xでの並列radixソートの処理内訳（64PE, 2.5Mword）
Fig. 5 Comparison of Parallel radix sort on EM-X and EM-4 (64PE, 2.5 Mword).

radixが64のときも性能が低下する．2次キャッシュはダイレクトマップであるためキャッシュ衝突の影響が大きい．

4.2 各並列計算機における並列Radixソートの実行

4.2.1 EM-XおよびEM-4

前章の並列アルゴリズムを，EM-X上のCコンパイラであるEM-C[8]を用いて実装を行った．逐次版から並列版へは，以下に示す変更が必要であった．EM-Cではリモートメモリを参照するグローバルポインタを利用できるため，カーネルループ部は(3)を除き変更が不要であった．すなわち，逐次のメモリアクセスと同様にワード単位のデータ転送をポインタ変数への代入という同一の記述で行える．また，共有メモリ上のindexをグローバルアドレスに変換するオーバヘッドも，転送開始アドレスを示す各グローバルポインタを求めるときのみであり，データ数に対してほぼ無視できる程度に削減できる．

(1) ローカルな要素数の数え上げのあとに，グローバルな操作により全体の要素数などを求める．
(2) 共有メモリ上のindexを分散メモリ上のアドレスに変換する．
(3) データ転送アドレスの更新の際に，境界に達したかどうかのチェックを行う．境界に達していたら転送アドレスを次のPEの先頭に設定する．
(4) 転送終了の同期を行う．

また，EM-4においてもほぼ同じプログラムがそのまま実行できる．ただし，搭載メモリサイズの違いなどにより最大データ量がEM-Xの1/8と小さい点と，2章で述べた直接リモートメモリアクセス機構がEM-4にはないため，パケットバッファが溢れないようにスレッドを明示的に中断することが必要である．

PE台数64台で問題サイズを変化させて実行した結果が図4である．x軸が総データ量，y軸が1秒あたりのソートデータ量である．図には64，256の2種類のradixについてEM-X，EM-4の両計算機の結果を示すとともに，radixが256の場合の逐次処理性能から理想的台数効果を得られるとした場合の値を示している．全体のヒストグラムを生成する際のグローバル処理のオーバヘッドにより，EM-X，EM-4ともに総データ量が256Kワード以下ではradixが64の場合が高速であるが，それ以上になるとradix256の場合が高速となる．また，EM-4では逐次処理からの理想処理速度に対して実際の並列性能が半分程度であるのに対し，EM-Xでは8割以上の処理速度を実現している．これは2章で述べたEM-Xの直接メモリ参照機構の効果である．この効果をより詳しく比較するため，各処理ごとの実行時間を比較する．

総データ数を2.5Mワード，PE台数を64，radixを256としたときのEM-4およびEM-Xの実行時間を図5に示す．全実行時間を各PE内でのローカルな数え上げ，全PEでの総和や部分和などのグローバル処理，リモートデータ転送に分けて示している．また，EM-Xの持つ分散共有メモリ機構の評価のため，EM-XでEM-4と同様にスレッド起動によるリモートメモリ書き込みを用いた場合をEM-X'として示した．EM-X'をEM-4と比べると約2.5倍速い．クロックスピードの改良は1.6倍であるので，それ以外は命令セットアーキテクチャおよびメモリ参照機構の改良によるものである．メモリ参照機構の改良による効果としては，局所的な数え上げのループ実行があげられる．各ループはEM-4，EM-Xともに10命令からなるが，3回のメモリ参照を含むため，EM-4では13サイクルかかるのに対し，EM-Xでは10サイクルと3割ほど高速になっている．また，命令セットアーキテ

クチャの改良の効果としては，ブロック転送があげられる．総和を求める場合には配列に対するリダクション処理が行われており，このブロック転送性能が重要となる．EM-4，EM-X ともブロック転送機構を持っておらず，1 ワードの転送を繰り返すことが必要であり，ループアンローリングを用いて最適化を行ったライブラリを用意している．EM-4 ではメモリおよびパケットのアドレス更新を別途行うため，1 ワードにつき 5 クロック必要であるが，EM-X では自動更新型のメモリアドレッシングやレジスタ相対型のパケットアドレッシングにより 1 ワードにつき 2 クロックと 2.5 倍のスループットを実現した．これはネットワークのハードウェア転送性能と同じスループットである．

また，EM-X を EM-X' と比較すると，データ転送以外は等しく，データ転送では 1.7 倍の速度向上が見られる．これはリモートメモリ書き込みの処理がスレッド処理とオーバラップされるという直接リモートメモリ参照機構によるものである．オーバラップされるリモートメモリ書き込み自体の処理量は，高速化により削減された処理量の 2 割程度にすぎない．この直接のオーバラップ効果の他に，リモートメモリ書き込みのスケジューリングのためにスレッドを中断する必要がないため，スレッド環境待避のオーバヘッドが削減できる効果が大きい．命令コードの静的な解析によると，スレッド中断時には 4 変数の待避/復帰に 8 クロックサイクルが必要であったが，これが不要となり，カーネルループの実行が約 1.5 倍に高速化された．さらに，直接リモートメモリ参照機構の効果は性能向上ばかりではなく，パケットバッファの溢れをほぼ気にしなくてもよいことによりプログラミングの容易さにもつながっている．

4.2.2 AP1000+

AP1000+ではメッセージバッファを経由しない PUT/GET 機構を用いてユーザレベルで直接リモートメモリアクセスができる．そこで，EM-X と同様に，ワード単位のリモートメモリ書き込みを用いた場合（fine）と，転送データをいったんまとめてからそのブロックをリモートメモリ書き込みで行った場合（coarse）のそれぞれについて調べた．

ワード単位で転送する方式（fine）では PUT の DMA の終了をチェックすることによりコマンドキューにとどまっているリクエストの最大個数の管理を行っている．これは PUT のリクエストを連続して与えるとコマンドキューが溢れて，コマンドのリストアのソフトウェア処理により遅くなってしまうためである．実験の結果では 7 個までのリクエストを先行受付けする方式が最適であった．

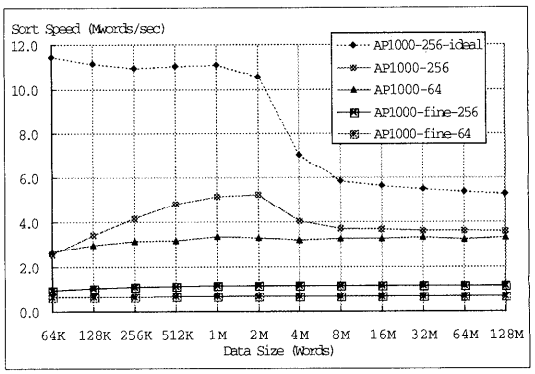


図 6 AP1000+での並列 radix ソート（32PE）
Fig. 6 Parallel radix sort on AP1000+ (32PE).

また，いったんまとめてから転送する方式（coarse）では，各 PE でのデータ転送がほぼいっせいに始まるため，転送先の受信処理の衝突が起きないようデータ転送の順番をスケジューリングしている．方式 fine では元から転送先がランダムになっているのでその必要はない．

AP1000+では全 PE での総和を求めるためのライブラリはサポートされているが，部分和を求めることはできず，また対象となるデータはスカラに限られてしまう．このため本プログラムでは通常のメッセージ通信を用いてバタフライネットワークを実現することにより配列データに対する総和および部分和の計算を行っている．これによりスカラデータに対する総和をデータ数分繰り返す方式に比べ radix が 256 の場合で 30 倍以上の高速化を実現した．

PE 台数 32 台で問題サイズを変化させて実行した結果が図 **6** である．x 軸が総データ量，y 軸が 1 秒あたりのソートデータ量である．radix は 64 と 256 の場合を示すとともに，radix が 256 の場合の逐次処理性能からの理想的台数効果を得られるとした場合の値を示した．ワード単位の細粒度通信を行う方式（fine）ではメッセージ生成処理は必要ないが，それ以上にデータ転送処理に時間がかかっている．これは AP1000+では，PUT/GET 機構を用いて直接リモートメモリアクセスが可能であるが，1 回の起動のオーバヘッドが 5.1 マイクロ秒[9]とワード単位の転送を行うには大きすぎるためである．細粒度通信を効率的に実行するには，送信側のレイテンシを削減する機構が必要となることが分かる．一方，いったんデータをまとめてから転送する方式（coarce）では，グローバル処理のオーバヘッドを相対的に減らすためにデータ量を大きくしないといけないが，データ量を大きくすると TLB ミ

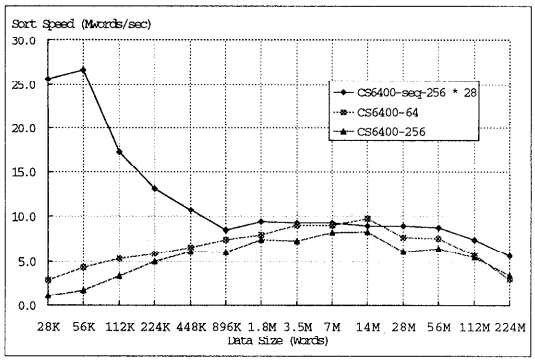


図7 CS6400での並列radixソート（28PE）
Fig. 7 Parallel radix sort on CS6400 (28PE).

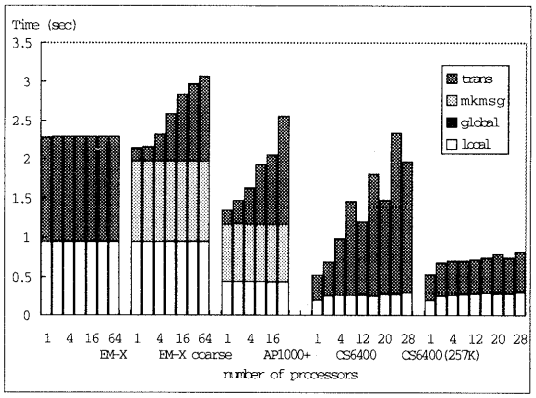


図8 PEあたりのデータ量を一定（256K）にした場合
Fig. 8 Scalability with same load (256 K) on a PE.

スの影響が現れて逆に実行時間が増大してしまう．また，逐次処理からの理想処理速度と比較して，たとえば，データサイズ2Mの場合，およそ半分の処理速度となっている．これについては，3.3節でプロセッサ台数を変化させた場合のスケーラビリティとして詳しく述べる．

4.2.3 CS6400

CS6400でのプログラムはSolarisのthreadライブラリを用いて並列化している．CS6400の2次キャッシュは2Mバイトと大容量であるが，ダイレクトマップなので共有データを2次キャッシュ境界に合わせるとともに，2次キャッシュでキャッシュラインの衝突が起きにくくなるよう，データ量を2の巾乗個より若干多くして実行している．また，グローバルヒストグラムを生成するためのリダクション処理は$O(N)$（Nは PE台数）のアルゴリズムを用いているためデータ数が少ない場合はその影響が見られるが，バリア処理を文献10)に基づく共有メモリ向きの高速な処理を行うことにより，大きなデータ量に対してはグローバル処理は無視できるようにしている．

スレッド数を28とし，問題サイズを変化させて実行した結果を図**7**に示す．x軸が総データ量，y軸が1秒あたりのソートデータ量である．使用したマシンがサーバ機であり，他のユーザの影響を避けるため，32プロセッサ中，使用するプロセッサを28プロセッサまでとし，10回繰り返した中で1番良い結果を表示している．radixは64と256とし，比較のためradixが256の場合の逐次処理の結果をスレッド数（28）倍した値を示す．radixの違いによる差は，データ数が少ないときのグローバル処理のオーバヘッドの差を除き，小さい．また，逐次処理との比較では，プロセッサ台数が増えることにより2次キャッシュが増加する効果で，896Kから14Mの範囲ではほぼ逐次処理と同等の実行効率を示しているが，ソートした結果を格納するデータに関しては2次キャッシュの衝突を引き起こしやすいため，スーパリニアの効果を示すところまではいかない．

4.3 PE台数に対するスケーラビリティの評価

図**8**にPEあたりのデータ数を一定として，PE台数を増加させた場合の実行時間とデータ転送時間を示す．TLBの影響を受けないようradixは64とし，PEあたりのデータ量を256Kワードとした．PEあたりのデータ量が一定なら，radixソートのPEあたりの処理量およびデータ転送量はグローバル処理を除くと一定である．図8ではグローバル処理の実行時間はほとんど現れていない．また，EM-Xでは実行時間およびデータ転送時間ともにPE台数によらず一定である．一方，AP1000+の場合は，データ転送以外は一定であるが，PE台数が増加するとともにデータ転送時間が増大し，結果として実行時間が増加している．元々プロセッサどうしを直接接続する直接網ではバイセクションバンド幅がプロセッサ台数に比例しては増加しないため，全対全通信時にネットワークがボトルネックとなることが指摘されている．EM-Xでも若干の性能の差はあるがほぼ同様のことがいえる．ところがAP1000+の場合のみにデータ転送時間の増大の現象が見られたのは，AP1000+の場合，データをいったんまとめてから転送を開始するため，大量のデータ転送がほぼ同時に各PEで開始されることにより，ネットワーク上での衝突を引き起こしているためであると考えられる．

このことを確認するため，EM-Xでもいったんメッセージにまとめてから転送を行う方式を実装し，実行した結果がEM-X coarseである．この場合EM-XでもAP1000+と同様にPE台数の増加とともにデータ

転送時間の増加が確認された．このように，メッセージにまとめず細粒度のままで転送を行う場合，データを処理しつつ出力するため出力頻度が均等化され，ネットワークへの負荷が均等化されるため，PE台数の増加の影響を受けにくくなっている．

たとえば，EM-Xの細粒度転送の場合，最内ループは16命令からなりこの16サイクルごとに1パケットの転送を行えばよいのでネットワークに対する負荷は低い．これに対し，粗粒度転送ではメッセージ生成およびデータ転送の最内ループがそれぞれ13命令，2命令であるが，メッセージ生成とデータ転送が分離しているためネットワークへの負荷としては2サイクルに1パケットと非常に高い．

細粒度処理の効果は，共有バス型並列計算機であるCS6400にもいえる．1PEあたり257Kワードとした場合は非常にスケーラブルな結果となっている．プログラミング上は細粒度なデータアクセスを行う一方で，2次キャッシュ上でデータのプリフェッチやブロックライトなどにより共有バスへの負荷が抑えられている．しかし，このようなキャッシュの効果はつねにうまくいくわけではない．CS6400の場合，2次キャッシュはダイレクトマップのため，キャッシュの衝突が生じる可能性が高く，図8によると，1PEあたり256Kワードとした場合にはデータ転送時間が3倍近くかかる場合があることが分かる．

5. まとめ

本稿では，高並列計算機EM-Xのアーキテクチャについて述べるとともに，その特徴である細粒度パケットによる共有メモリプログラムの評価として，radixソートを取り上げ，実行結果を示すとともに各アーキテクチャ機構の考察を行った．また，他の並列計算機EM-4，AP1000+，CS6400との実行結果を比較することにより，EM-Xの細粒度共有メモリアクセスの有効性を確認した．また，実際のプログラミングを通じて，逐次プログラムからの書換えの容易さなども確認した．

特に，radixソートプログラムでは効率的な細粒度通信を用いることにより，計算と通信を融合し，メッセージ生成などの余分な処理を減らすとともに，通信頻度の平滑化によりネットワーク負荷を軽減し，プロセッサ台数を増加させたときのオーバヘッドを大幅に減らすことが可能であった．今回評価に用いたradixソートプログラムは，リモートメモリ書き込みが主な通信であり，そのようなアプリケーションに対しては本稿のradixソートのような細粒度通信を適用することによりスケーラビリティの向上を図ることが可能であると考えられる．この他のリモートメモリ読み出しや，リダクションなどのグローバル通信が重要となるプログラムについても同様に，細粒度通信を実装した実機を用いた検証を行い，総合的な評価を行っていく予定である．

謝辞 本研究を遂行するにあたりご指導，ご検討いただいた太田公廣前情報アーキテクチャ部長，大蒔和仁情報アーキテクチャ部長ならびに研究室の同僚諸氏に感謝いたします．特に小池汎平氏には共有バス上での同期機構などについて有用な助言をいただきました．

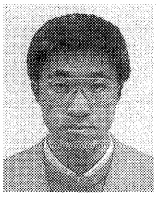
参考文献

1) Heinlein, J., Gharachorloo, K., Dresser, S. and Gupta, A.: Integration of Message Passing and Shared Memory in the Stanford FLASH Multiprocessor, *Proc. ASPLOS '94*, pp.38–50 (1994).
2) 平木，天野，久我，末吉，工藤，中島，中條，松田，松本，森：超並列プロトタイプ計算機JUMP-1の構想，計算機アーキテクチャ研究会，102-10，pp.73–84 (1993).
3) 児玉，坂根，佐藤，坂井，山口：高並列計算機EM-Xのリモートメモリ参照機構の評価，情報処理学会論文誌，Vol.36，No.7，pp.1691–1699 (1995).
4) Kodama, Y., Sakane, H., Sato, M., Yamana, H., Sakai, S. and Yamaguchi, Y.: The EM-X Parallel Computer: Architecture and Basic Performance, *Proc. ISCA '95*, pp.14–23 (1995).
5) Woo, S.C., Ohara, M., Torrie, E., Singh, J.P. and Gupta, A.: The SPLASH-2 Programs: Characterization and Methodological Considerations, *Proc. ISCA '95*, pp.24–36 (1995).
6) Sakai, S., Yamaguchi, Y., Hiraki, K., Kodama, Y. and Yuba, T.: An Architecture of a Dataflow Single Chip Processor, *Proc. ISCA '89*, pp.46–53 (1989).
7) Sakai, S., Kodama, Y. and Yamaguchi, Y.: Design and Implementation of a Circular Omega network in the EM-4, *Parallel Computing*, Vol.19, No.2, pp.125–142 (1993).
8) Sato, M., Kodama, Y., Sakai, S., Yamaguchi, Y. and Koumura, Y.: Thread-based Programming for the EM-4 Hybrid Dataflow Machine, *Proc. ISCA '92*, pp.146 155 (1992).
9) 白木，小柳，今村，林，清水，堀江，石畑：高並列計算機AP1000+のメッセージハンドリング機構，並列処理シンポジウムJSPP'95，pp.233–240 (1995).
10) Crummey, J.M. and Scott, M.L.: Algorithms for Scalable Synchronization on Shared-memory Multiprocessors, *ACM Trans. Com-*

puter Systems, Vol.9, No.1, pp.21–65 (1991).
(平成8年9月18日受付)
(平成9年7月 1 日採録)

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．同年通産省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など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，並列探索問題などに興味あり．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）など受賞．電子情報通信学会，IEEE各会員．

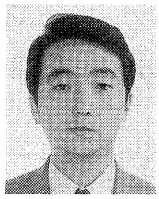


坂根 広史（正会員）

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．同年通産省電子技術総合研究所入所．以来，超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．電子情報通信学会，神経回路学会会員．

佐藤 三久（正会員）

1959年生．1982年東京大学理学部情報科学科卒業．1986年同大学院理学系研究科博士課程中退．同年新技術事業団後藤磁束量子情報プロジェクトに参加．1991年，通産省電子技術総合研究所入所．1996年より，新情報処理開発機構つくば研究センターに出向．現在，同機構並列分散システムパフォーマンス研究室室長．理学博士．並列処理アーキテクチャ，言語およびコンパイラ，計算機性能評価技術などの研究に従事．日本応用数理学会会員．

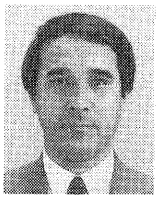


山名 早人（正会員）

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年同大学院博士後期課程修了．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年情報処理学会第46回全国大会奨励賞受賞．1993年より電子技術総合研究所に勤務．並列処理システム，並列化コンパイラ，WWW情報検索の研究に従事．著書「超並列コンピュータ入門」（共著）．電子情報通信学会，ACM，IEEE各会員．

坂井 修一（正会員）

1958年生．1981年東京大学理学部情報科学科卒業．1986年同大学大学院情報工学専門課程修了．工学博士．同年，電子技術総合研究所入所．1991年4月より1年間米国MIT招聘研究員．1993年3月より1996年2月までRWC超並列アーキテクチャ研究室室長．1996年10月より筑波大学助教授（電子・情報工学系）．システム一般，特にアーキテクチャ，並列処理，スケジューリング問題などの研究に従事．情報処理学会論文賞（1990年度），日本IBM科学賞（1991年），市村学術賞（1995年），ICCD Outstanding Paper Award（1995年）など受賞．IEEE，電子情報通信学会会員．



山口 喜教（正会員）

1972年東京大学工学部電子工学科卒業．同年通商産業省工業技術院電子技術総合研究所入所．以来，高級言語計算機，並列計算機アーキテクチャ，特にデータフロー計算機や実時間並列処理システムなどの研究に従事．現在，情報アーキテクチャ部・並列アーキテクチャラボ・リーダー，筑波大学連携大学院教授．工学博士．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）．電子情報通信学会，IEEE各会員．