高並列計算機 EM-X のリモートメモリ参照機構の評価

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

分散メモリ型高並列計算機において，自プロセッサ以外のメモリをいかに容易に，かつ，高速に参照できるかが並列処理性能に大きく影響を与える．高並列計算機 EM-X では，細粒度通信に基づくマルチスレッド機構によりレイテンシの小さいプロセッサ間通信を実現しているが，さらに，(1)優先度処理に基づくパケットスケジューリング，(2)パケットバッファ部により演算実行部とは独立に処理される直接メモリ参照機構などにより，リモートメモリ参照の実行時レイテンシを大きく低減し，共有メモリモデルに基づく並列処理プログラムの実行性能を向上させている．本稿では，簡単なプログラムにより，EM-X におけるリモートメモリ参照機構の有効性を示す．

An Evaluation of Remote Memory Access on a Highly Parallel Computer EM-X

YUETSU KODAMA,[†] HIROHUMI SAKANE,[†] MITSUHISA SATO,[†]
SHUICHI SAKAI[††] and YOSHINORI YAMAGUCHI[†]

Easy and efficient mechanisms for accessing remote memory is important for parallel computers with distributed memory. Highly parallel computer EM-X realizes inter-processor communication with low latency using multi-thread architecture based on fine grain packet communication. It also supports the remote memory access by following mechanisms; (1) packet scheduling based on priority control, (2)direct remote memory access facility which can be operated on an input packet buffer unit independent of thread execution. These mechanisms greatly reduce the run-time latency of remote memory access and fully supports shared memory programming. This paper shows its effectiveness using simple programs.

1. はじめに

高並列計算機では，多数のプロセッサが相互に通信を行って計算を進めていくため，その通信性能は全体の処理性能に大きな影響を与える．特に，データを分散配置し，それらを互いに参照し合う機構は，実行効率の面からも，また，プログラミング支援の面からも重要となる．

プログラミングモデルからデータ分散およびその参照を考えた場合，大きく分けて共有メモリ型プログラミングとメッセージ交換型プログラミングが考えられる．これらに対応するハードウェアのメモリモデルとして，システム内で一意に決められた共有アドレスに対するアクセスの手段のみを提供する共有メモリ型モデルと，局所メモリアクセスとは別に結合網に直接接続する手段を提供する分散メモリ型モデルがあり，それぞれのモデルに基づいた並列計算機が開発されている．しかし，この対応は絶対的なものではなく，プログラミングモデルとしてどちらを選択するかは，その対象となる処理に応じて選択されるべきであり，それを実行するハードウェアのメモリモデルとは独立していることが望ましい．そのため，並列計算機には，ハードウェアのメモリモデルによらず，種々のプログラミングモデルを効率良く実行することが求められている．

分散メモリ型計算機においても，プロセッサ番号と局所メモリアドレスを組み合わせ，システムで一意なグローバルアドレスを用いることにより，共有メモリ型プログラミングを実現することは比較的容易である．例えば，データパラレルプログラミングのようなまとまったデータを扱う場合には十分な性能を発揮できる．しかし，現在のスループット重視の並列計算機，すなわち数十あるいは数百ワードをまとめて転送する

† 電子技術総合研究所情報アーキテクチャ部
Electrotechnical Laboratory, Computer Science Division
†† 新情報処理開発機構つくば研究センタ
Real World Computing Partnership, Tsukuba Research Center

場合でなければ処理性能を引き出せないような計算機においては，より細かな単位でアクセスが必要なプログラムを実行する場合，その通信のオーバヘッドおよびプロセッサ介在のオーバヘッドが足枷となって十分な性能を発揮できない．分散メモリ型計算機における共有メモリプログラミングの適用範囲を広げるためには，レイテンシの少ない，より細かなレベルの通信を可能にするとともに，通信と命令実行のオーバラップを可能にすることが必要である．

リモートメモリ参照を効率良く行うためには，リモートメモリ参照を行う際のレイテンシをいかに軽減あるいは隠蔽するかが重要である．キャッシュメモリはレイテンシを軽減する一手法であるが，この場合，各キャッシュメモリ間のコヒーレンスをいかに保つかが重要な問題となる．このためにスタンフォード大のDASH[1]などスヌープキャッシュとディレクトリ方式を組み合わせた並列計算機の研究が行われている．しかし，すべてのメモリ参照をキャッシュおよびコヒーレンス制御で一律に管理する方式では，リモート参照の割合が増大するにつれ，処理のオーバヘッドが顕在化してくることが予想される．キャッシュミス時の効率低下を防ぐために，MITのALEWIFE[2]などでは以下に述べるマルチスレッドによるレイテンシ隠蔽の方式をキャッシュシステムと組み合わせる研究が行われている．しかし，コヒーレンス制御方式がメモリ参照パターンと一致していない場合には，キャッシュミスの割合を低減することができないため，コヒーレンス制御のオーバヘッドを隠蔽することは難しい．このような場合に対処するためには，メモリ参照のパターンに応じてコヒーレンス制御方式自体をよりきめ細かく制御することが必要になると考えられる．

また，レイテンシを積極的に隠蔽しようとする方式としてマルチスレッドがある．これはあるスレッドがリモートメモリ参照などによりアイドルになると，他のスレッドの実行を開始することにより全体の効率を低下させない方式である．この方式ではスレッドの切替えオーバヘッドがレイテンシに比べて小さいことが必要となる．また，各スレッドの実行順序は実行時に変化するため，スレッド間の同期が効率良く行える必要がある．これらの要件を満たす計算機としてデータ駆動計算機[3),4)]が知られており，さらに逐次処理の効率的実行を目指したマルチスレッドマシンが研究されてきた[5)~7)]．各スレッドの実行は自由にプログラムすることができるため，メモリアクセスパターンに応じた制御を行うことが可能である．

我々はさらに汎用で高性能なマルチスレッドマシンを目指してEM-Xの開発を行っている．EM-Xはマルチスレッドによるレイテンシ隠蔽と，ネットワークおよび通信インタフェースの高速化によるレイテンシの低減を基本アーキテクチャとすることにより，レイテンシに対する耐性の強い計算機となっており，メッセージ通信モデルのプログラムを効率良く実行することができる．これに加え，リモートメモリ参照機構を最適化することにより，共有メモリプログラミングに対しても効率良く支援することが可能となっている．本論文で述べる最適化手法であるパケット実行の優先度制御およびネットワークからの直接メモリ参照機構は，他のメッセージ通信型並列計算機に対しても適用可能であるが，パケット通信やスレッド切替をハードウェアレベルでサポートしているEM-Xアーキテクチャにおいては，その効率的実現が重要となる．

本論文では，まずEM-Xのマルチスレッドアーキテクチャについて述べ(第2章)，さらにリモートメモリ参照のための最適化について詳しく述べる（第3章)．次に，レジスタ転送レベルシミュレータを用いた性能評価を行い，その有効性を考察する(第4章)．最後に今後の予定と課題について述べる（第5章)．

2. EM-X アーキテクチャ

EM-Xは分散メモリ型の高並列計算機であり，プロセッサ間通信のオーバヘッドを低減するためのネットワーク，プロセッサおよび両者のインタフェースを総合的にとらえた並列アーキテクチャを提唱している．すなわち，EM-Xではプロセッサ間通信をI/Oとしてではなく，メモリ参照と同等のリソース処理ととらえ，通信と命令実行の融合を積極的に押し進めている．この融合の鍵となるのは細粒度通信とマルチスレッドアーキテクチャである．

細粒度通信とは，2ワード程度からなる細粒度パケットを通信の単位とすることであるが，これにより，パケットの送信，転送，受信の各ステージを簡略化・高速化することを容易にするとともに，ステージ間のインタフェースのオーバヘッドを低減することが可能となる．**図1**に要素プロセッサEMC-Y内の各ステージを示す．

例えば，パケット生成回路は，細粒度通信に限ることにより，他の演算ユニットと同様に演算パイプラインの1ステージとして実現することが可能となり，通信のためのセットアップ時間を不要としている．これは，通常のRISCパイプラインでは入力オペランドは2つまでであり，これに命令内のデータを加えた限られたリソースから生成できる程度にパケットを簡略化

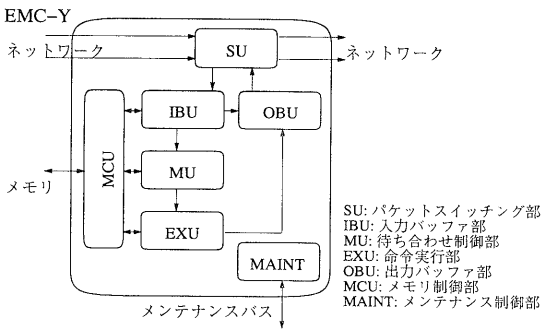

図1 要素プロセッサ EMC-Y
Fig. 1 Processing element EMC-Y.

しているためである.

ネットワーク上のパケット転送は,EM-X ではネットワークトポロジーとしてサーキュラオメガ網[8]を用いており,PE 間距離が PE 台数の log オーダに抑えられている.本ネットワークでは,ストア-アンド-フォワード-デッドロックを防ぐため,3 バンクのバッファを各入力ポートに備えているが,細粒度パケットを用いることにより,バッファ量を抑え,パケットスイッチをプロセッサに内蔵することを可能にしている.さらにパケット内の宛先 PE 番号のみにより最短ホップ数を保証した経路を選択できるセルフルーティング方式[9]や,レイテンシを抑えたバーチャルカットスルー転送の制御も容易になっている.

プロセッサとネットワーク間のパケット送受信は,RISC パイプラインによる命令実行とは独立に処理できるように,送信・受信の双方に 8 パケット程度のバッファをプロセッサ内部に備えている.このバッファは FIFO (First In First Out) により管理されているが,細粒度の固定長パケットにより管理は容易である.出力バッファ (OBU) がいっぱいになった時には,ネットワークの飽和を抑え,かつ,制御を簡略化するため,パケット出力命令の実行に限り命令実行パイプラインを中断する.入力バッファがいっぱいになった時には,ネットワークの飽和を抑えるため,溢れたパケットをメモリ上のパケットバッファに退避する.入力バッファに空きが生じると,到着順序を保ちつつ自動的に復帰される.このメモリパケットバッファの管理は,命令実行パイプラインとは独立に動作できるようになっている.

EM-X のパケットは,単にプロセッサ間で通信されるデータではなく,パケット自身が実行命令列 (スレッド) を直接起動する continuation として働く.すなわち,パケットのヘッダ部には実行を開始する命令アドレス,実行に必要な作業領域などのベースアドレス,2 入力の待ち合わせを行う待ち合わせメモリアドレスなどがパックされた形で収められている.パケットによるスレッドの起動は,待ち合わせ処理部 (MU) により先行制御されるため,このためのオーバヘッドの多くは命令実行から隠蔽することが可能である.

パケットにより起動されたスレッドは,自分自身がスレッドを中断しない限り,エラー例外ハンドラを除いて,他のスレッドにより割り込まれることはない.これによりスレッド内は静的にスケジューリング可能で,かつ,キュー操作などでもロックなどの必要はない.

入力バッファ部におけるプロセッサへの到着順によるパケットの FIFO 制御,パケットによるスレッドの直接起動,および,スレッド実行の自律排他的制御により,EM-X の実行モデルは非常に簡略化され,また,マルチスレッドに適したモデルとなっている.すなわち,他 PE への関数呼び出しやメモリ参照などでスレッドの実行がアイドルになるような場合は,単にスレッドを中断するだけで,入力バッファの先頭のパケットにより次のスレッドが起動される.この起動も待ち合わせ処理部での先行制御により,通常 1-2 クロックサイクルで行うことが可能であり,スレッド切替のオーバヘッドは非常に小さく抑えられている.他 PE へ処理を要求する際に,スレッド再開のための continuation を一緒に送っておき,リモート側で処理が終った際には,その continuation をヘッダ部として結果を出力することにより,スレッドの再起動が行われる.continuation は 1 ワードに自動的にパックされ,また,この生成も 1 命令で行うことができるため,オーバヘッドは小さい.このように,プログラムはいくつかのスレッドからなり,それらをパケット continuation により結び付けているととらえることができる.このため,各スレッドが複数の PE に分散していても効率を落すことなく処理を進めることが可能である.

3. リモートメモリ参照機構

EM-X は分散メモリ型の並列計算機であり,前節でも述べた通りプロセッサ (PE) 間の通信は細粒度パケットを用いて行われる.EM-X のパケットは,2 ワード構成の非常に短いパケットであり,他のパケット通信型並列計算機のような多大なセットアップ時間を必要としない.このため,パケットの生成・出力は,局所メモリへの参照と同様に命令実行パイプラインレベルで行うことが可能である.この細粒度パケットを用いて EM-X では種々の方法でリモートメモリへのアク

セスが可能となっている．

3.1 関数呼び出しの利用

リモートメモリに対してアクセスする最も一般的な方法は，そのメモリの存在するプロセッサに対してリモート関数呼び出しを行い，その関数内で局所メモリに対してアクセスする方法である．EM-X では1つの関数インスタンスに対してオペランドセグメントと呼ぶ関数フレームを割り当てる必要がある．このオペランドセグメントは引数の受渡し，局所変数の退避，待ち合わせメモリなどに使用される．また，このオペランドセグメントの先頭には，この関数の命令コード領域へのベースポインタ（TTOP）が置かれている．これにより，パケット continuation はオペランドセグメントへのポインタと，実行を開始する命令の TTOP からの変位を持つだけで良いことになり，1ワードに収めることが可能となっている．リモート関数呼び出しを行う際には，まずリモート PE に対してオペランドセグメントの割当を要求し，それへのポインタを受けとる．次に，そのポインタを用いて引数をオペランドセグメント内の引数領域に書き込む．最後に関数呼び出し後に再開するスレッドの continuation をデータとしてリモート関数を起動する．

3.2 特殊パケットの利用

上記の方法では毎回オペランドセグメントの割当を要求しなくてはならないため，細かなリモートメモリアクセスを繰り返すような場合には適さない．そのため，EM-X ではオペランドセグメントを割り当てなくても起動できる特殊パケットハンドラを64個までプログラミングすることができる．良く使われるハンドラはシステムハンドラとして提供されており，リモートメモリ参照用のハンドラもそのうちの1つである．このハンドラを起動するためには**図2**に示すような USRRD パケットを用いる．EM-X のパケットはアドレスワードとデータワードからなり，アドレス部はパケットタイプ(PT)，宛先プロセッサ番号(PE)，アドレスデータなどからなる．通常のパケットは，PT が0となっており，アドレスデータで continuation を示している．USRRD パケットは，PT でリモートメモリアクセス用のハンドラを起動することを，アドレスデータで参照すべき局所メモリアドレス(MA)を示している．EM-X では PE 番号と局所メモリアドレスの組み合わせでグローバルメモリアドレスを一意に指定することができる．データ部には読み出したデータを処理するスレッドへの continuation を指定する．

USRRD パケットによるリモートメモリアクセスのレイテンシは，他のパケットとの衝突・競合がない場合で平均23クロックである．平均というのは EM-X ではネットワークとして直接網を採用しており，宛先 PE によってその距離が異なるためである．レイテンシの内訳は，USRRD パケット出力(2 clock)，ネットワーク転送（往復各7 clock)，USRRD ハンドラ（4 clock)，結果パケットによるスレッド再開(3 clock)である．このレイテンシの23クロックのうち，図2に示すように実際に実行パイプラインを占有するのは7クロックに過ぎず，残りのクロックはマルチスレッディングにより有効利用が可能である．

USRRD パケットと同様に，USRWR パケットというリモートメモリ書き込みハンドラも用意されている．また，同期つきメモリ参照として知られるI-structure や読み出し/書き込み要求をそれぞれキューとして管理できる Q-structure もこの特殊パケットハンドラを用いてサポートされる．このほか，やってきたパケットの合計を計算するハンドラや，最大値を保持するハンドラなど自由に設定することが可能である．

3.3 優先処理パケットの利用

上記の USRRD パケットのレイテンシは，宛先 PE に他のパケットおよびスレッドがまったく無いとした場合の値である．しかし，実際にプログラムを実行し

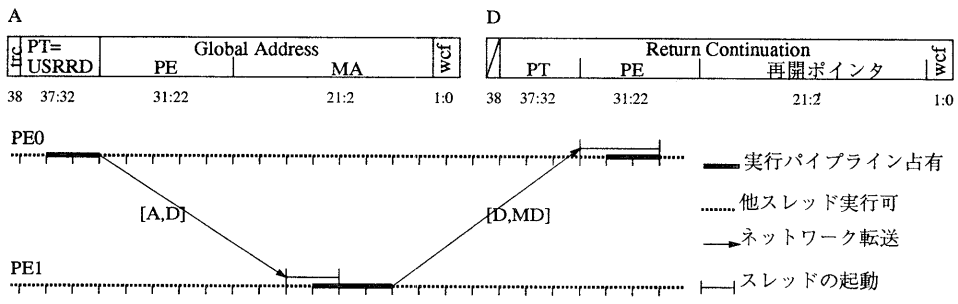

図2 リモートメモリ読み出し（USRRD）パケット
Fig. 2 Packet operation for remote memory read (USRRD).

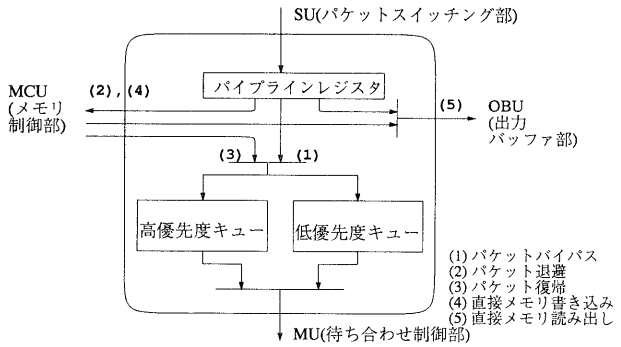

図3 入力パケットバッファ (IBU) 部
Fig. 3 Input packet buffer unit (IBU).

ている場合には，宛先 PE の入力バッファにパケットが溜っていることが普通である．EM-X ではバッファを FIFO 制御しているため，後からやってきた USRRD パケットは，先に到着した他のパケットの処理がすべて済んだ後に処理が始まるため，実際のレイテンシはより大きな値となる．

バッファの FIFO 制御は，同一処理を行うパケットに対しては制御が簡単で公平なスケジューリングが行えるが，異なる処理を行うパケットに対しては優先度処理が行えないため，十分とはいえない．このため，EM-X では**図3**のように入力バッファを2種類用意し，パケット内の1 bit の優先度情報により使用するバッファを切替えることにより優先度処理を行っている．低優先度バッファ中のパケットは，高優先度処理バッファにパケットが存在しない時のみ，スレッドの起動が行える．各入力バッファ内では FIFO 制御が行われている．パケット中の優先度はパケット出力命令で静的に指定される．2種類の優先度しかないため，高優先度のパケットは主にシステム処理的なもののうち特に動的レイテンシを下げたいパケットに使用する．例えば，動的負荷分散のための他のプロセッサの負荷の取得であるとか，以下の直接メモリ参照機構では対応できない同期つきのリモートメモリ操作やグローバル変数の処理などである．

3.4 直接リモートメモリ参照の利用

パケットの優先度処理により，入力バッファ中のパケットに起因するレイテンシを低減することができる．しかし，EM-X ではスレッドの実行は，他のスレッドによっては割り込まれない排他性を保証しているため，実行中のスレッドの終了待ちに起因するレイテンシには対応できない．これでは，例えば，全対全通信のように各プロセッサがデータの送り手であると同時に受け手でもあるような場合，到達したパケットが送信中のスレッドにブロックされて処理されず，レイテンシが増大する．これを解決するために，EM-X ではリモートメモリ読み出しパケット (SYSRD) およびリモートメモリ書き込みパケット (SYSWR) に限り，入力パケットバッファ上のパケットおよび現在実行中のスレッドにブロックされずにパケット処理を行う直接リモートメモリ参照を可能にしている．

EM-X では入力バッファが溢れた時には，新たにやってきたパケットはメモリ上の入力バッファへ自動的に退避され，入力バッファに空きが生じると退避した順序を保って自動的に復帰される．このメモリ入力バッファを参照するためのメモリバンド幅は，データフェッチのためのメモリバンド幅と共有されているが，命令フェッチのためのメモリバンド幅とは独立して確保されている．命令実行パイプラインによるメモリ参照が優先されるが，それ以外の命令を実行している場合には，パケット転送を待たせることなくメモリバッファへ格納できる．直接リモートメモリ参照にはこのメモリバンド幅を利用する．すなわち，SYSWR パケットが PE に到達すると，パケットを入力パケットバッファに格納するのではなく，パケットのアドレス部で示されるメモリにデータ部を直接書き込む．SYSWR パケットは入力バッファを経由せずに処理されるため，バッファ内で待っているパケットの影響を受けず，また，このためのメモリバンド幅は命令フェッチとは独立なため，現在実行中のスレッドの影響も非常に小さく抑えることができる．さらに，SYSWR パケットはバッファ領域を使用せず直接相手先アドレスに格納するため，大規模データ転送時においてもバッファ溢れを抑制する効果が期待できる．同様に，SYSRD パケットが到着すると，パケットのアドレス部で示されるメモリからデータを読み出し，そのデータをパケットデータ部に格納されている continua-

```
matmul-rd()
{
    int i,j,s;

    for (i=0;i<64;i++) {
        s = 0;
        for (j=0;j<64;j++)
            s += A[j] *
                 mem_read(j,B[i]);
        C[i] = s;
    }
}
```

```
matmul-wr(id)
int id;
{
    int i,j,s;

    for (i=0;i<64;i++) {
        for (j=0;j<64;j++)
            mem_write(j,T[id],B[i]);
        barrier();
        s = 0;
        for (j=0;j<64;j++)
            s += A[j] * T[j];
        C[i] = s;
    }
}
```

図4 行列乗算プログラム
Fig. 4 Programs of matrix multiply.

tion を用いてネットワークに転送する．ただし，デッドロックを防ぐため，出力バッファがいっぱいの時は通常の優先処理パケットとして扱われる．

この直接リモートメモリ参照機構と同様の方法により，待ち合わせ処理をスレッド実行とは独立に先行処理することが可能である．EM-X では直接待ち合わせという高速な同期機構により2入力の待ち合わせをサポートしているが，それでも2つの入力のうち最初に到着するパケットは待ち合わせミスとなり，実行パイプラインの効率を低下させてしまう．そこで，パケットを入力バッファへ格納する前に，待ち合わせ相手が存在するかどうかを確認し，待ち合わせ相手が存在しなければ，当該パケットを待ち合わせアドレスに直接格納し，待ち合わせ相手が存在する時のみ，当該パケットを入力バッファへ格納するという処理を行う．この待ち合わせ処理に必要となるメモリバンド幅として，メモリパケットバッファへのアクセスバンド幅を用いることにより，命令実行とは並行して行うことができるため，待ち合わせミスのオーバヘッドを隠蔽することができる．

4. リモートメモリ参照機構の評価

前章で述べた EM-X のリモートメモリ参照機構について，レジスタ転送レベルシミュレータ[10]の上で評価を行った．このシミュレータは EM-X のプロセッサである EMC-Y を設計する際のテストベクタ生成にも使われたものであり，細部に至るまで実際の EM-X を忠実にシミュレートしている．また，出力バッファのサイズやパケット入出力優先順位などハードウェアパラメータを変えることができ，ハードウェアチューニングにも役立った．

このシミュレータ上で，2種類の行列乗算プログラムおよびナップザック問題を実行し，その実行時間ならびに各要素プロセッサの実行状況について評価を行った．

4.1 リモートメモリ読み出しを用いた行列乗算

まず最初に，必要な行列要素をリモートメモリ読み出しを用いて参照するプログラムを考える．本プログラムでは，要素プロセッサ(PE)数は64台，行列のサイズは64×64で，i番目の PE がそれぞれの行列の i 行目を持つこととしている．カーネル部分を図4 matmul-rd に示す．ここではリモートメモリ参照の部分を明らかにするため，mem_read というライブラリ関数を明示的に用いているが，EM-C では変数宣言時に分散メモリ上への割当を指定しておけば，コンパイラが自動的にリモートメモリ参照に変換してくれる．

このリモートメモリ参照に USRRD パケットを用いた場合と SYSRD パケットを用いた場合を比較する．上記のプログラムでは，一度に1行列要素のみを参照しているが，これをループアンローリングを用いて複数の行列要素をプリフェッチした場合も合わせて図5(a)に示す．

図5(a)によると，プリフェッチ数が1,2,4それぞれの場合で，SYRRD は USRRD に比べて速度が平均20%向上している．SYSRD 処理が他の命令実行と完全にオーバラップしたと仮定した場合，それによる実行命令数の削減効果はプログラムの静的解析により12%程度であると見積もられる．それ以上の速度向上はレイテンシ低減によるクリティカルパスの減少の効果と考えられる．この効果は，本プログラムではそれほど大きくは現われていない．これは，行列乗算プログラムのカーネル部分ではブロック長の短い単一の命令ブロックのみが動作しているため，もともとのレイテンシがそれほど大きくなかったためであると思われる．SYSRD のより有効な例は，より大きなプログラムで現われることが期待されるが，シミュレータでは限界があり，実機の完成を待って再度検証を行う予定である．

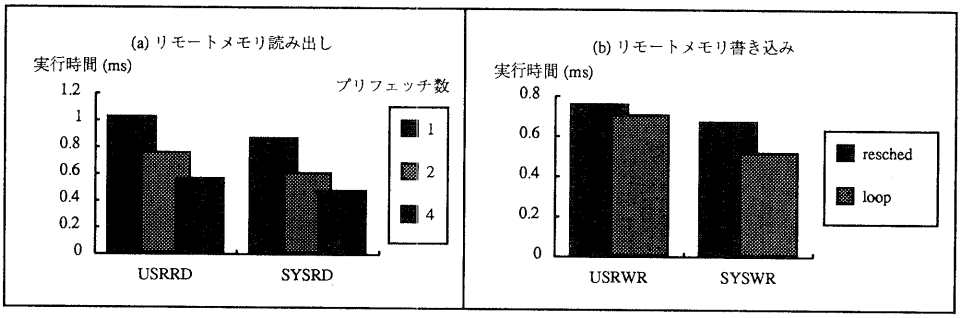

図5 行列乗算実行結果
Fig. 5 Results of matrix multiply.

4.2 リモートメモリ書き込みを用いた行列乗算

次に，計算に必要なデータを，データを保持しているプロセッサが必要とするプロセッサに書き込むプログラムを考える．カーネル部分を図4 matmul-wr に示す．この場合には，リモートメモリ書き込みが完了したことを保証する barrier を用いて同期をとる必要がある．

このリモートメモリ書き込みに USRWR パケットを用いた場合と SYSWR パケットを用いた場合とを比較する．また，メモリ書き込みのループごとに命令ブロックを中断した場合 (resched) と中断しない場合 (loop) とを比較する．これは実行スレッド長の長さにより SYSWR パケットの効果がどう変化するかを調べるためであり，当然 resched の方がスレッド長は短くなる．

図5(b)によると，SYSWR を用いると USRWR に比べて resched で13%，loop で37%の速度向上がある．loop の場合に効果が大きいのは，スレッド長が長いとそれだけ SYSWR の処理とスレッドの処理とがオーバラップする可能性が高いためである．resched のようにスレッド長が短いと直接メモリ参照の効果が薄れる．また loop の方が繰り返し処理などの逐次処理の最適化が行われるためスレッド自体の実行時間も短くなっている．一方，USRWR を用いた loop の場合には実際のメモリ書き込みはパケット出力処理にブロックされており，そのパケット出力処理が終ってから連続して処理される．USRWR の処理ルーチンはローカルメモリへの書き込みを行う1命令のみからなるため，USRWR の処理を連続して行う場合には，命令実行とパケットバッファリングおよびスレッドの起動をオーバラップさせることができず，そのオーバヘッドが実行時間に現われてしまう．このオーバヘッドと逐次処理の最適化の効果が相殺し，USRWR では resched と loop の差は非常に小さい．

表1 ナップザック問題の実行結果
Table 1 Results of knapsack problem.

	高優先度	通常優先度
実行時間 (ms)	25.0	114.6
ratio	1.0	4.58
探索木	28,042	129,846
ratio	1.0	4.63

4.3 ナップザック問題

ナップザック問題とは，与えられたものの中からその合計の重さがある値以下にするという制限の下に，価値を最大にする組み合わせを求める探索問題である．ものは単位重さ当たりの価値によりソートされているものとする．branch-and-bound 法では，すでに発見された最大の値を大域変数(G)として保存しておき，探索木における最大見積りと比較することにより，探索木を枝刈りすることを可能にしている．このアルゴリズムを並列化するためには，この大域変数 G の管理が重要となる．本評価では，各プロセッサ (PE) がこの大域変数のコピーを持ち，ある PE が G を更新する際に，他の PE に各 G のコピーを更新するスレッドをフォークすることとする．EM-X ではスレッドの実行は排他的に行われるため，この G を更新するスレッドでは，各 PE でローカルに G を最大に保つようにチェックさえすれば，全 PE で同期をとる必要はない．ここでは，この大域変数の更新スレッドの起動に，他と同じ優先度を持たせた場合と，高い優先度を持たせた場合で比較を行う．

表1 に物体の総数を 60 個とした場合の結果を示す．表には各方式での実行時間と探索に要した木の数を示している．表によると，大域変数 G を高優先度を用いて更新した方が，約 4.6 倍の処理速度を示している．探索木の総数も処理速度に比例した分だけ減っているため，高速化のほとんどが枝刈り効率の向上によりも

たらされていることが分かるが，これは高優先度パケットを用いることにより大域変数の更新を高速に行えたことによるものと考えられる．探索問題の処理速度については，探索木の条件により枝刈り効率が大きく異なるため，より多くの条件について調べることが必要であるが，現在シミュレータによる評価のため困難である．実機の完成を待ってより平均的な挙動を調べる予定である．

5. まとめ

レジスタ転送レベルシミュレータを用いて，行列乗算プログラムおよびナップザックプログラムを実行し，EM-X のリモートメモリ参照機構の評価を行った．この結果，直接リモートメモリ参照パケットを用いることにより，従来のスレッド起動型のリモートメモリ参照に比べて 2 割から 4 割の性能向上が確認された．また，優先度制御が並列探索問題で行うような大域変数の管理に有効であることが確認された．

EM-X はプロセッサ EMC-Y の設計が終了し，現在エンジニアリングサンプルを用いた実機による検証を行っており，80 台版のシステムの完成に向けて開発が進んでいる．レイテンシが実行時間に支配的な場合には，優先処理パケットや直接リモートメモリ参照の効果がより大きく期待できる．それらの検証も含め，今後は，実機を用いた実用規模のプログラムによる検証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．

参考文献

1) Lenoski, D., Laudon, J., Gharachoroloo, K., Gupta, A. and Hennessy, J.: The Directory-Based Cache Coherence Protocol for DASH Multiprocessor, *Proc. ISCA 90*, pp. 148-159 (1990).
2) Agarwal, A., Lim, B. H., Kranz, D. and Kubiatowicz, J.: APRIL: A Processor Architecture for Multiprocessing, *Proc. ISCA 90*, pp. 104-114 (1990).
3) Hiraki, K., Nishida, K., Sekiguchi, S., Shimada, T. and Yuba, T.: The SIGMA-1 Dataflow Supercomputer: A Challenge for New Generation Supercomputing Systems, *J. Info. Process.*, IPS Japan, Vol. 10, No. 4, pp. 219-226 (1987).
4) Papadopoulos, G. M.: *Implementation of a General Purpose Dataflow Multiprocessor*, MIT Press (1991).
5) Nikhil, R. S., Papadopoulos, G. M. and Arvind: *T: A Multithreaded Massively Parallel Architecture, *Proc. ISCA 92*, pp. 156-167 (1992).
6) Yamaguchi, Y., Sakai, S., Hiraki, K., Kodama, Y. and Yuba, Y.: An Architectural Design of a Highly Parallel Dataflow Machine, *Proc. IFIP 89*, pp. 1155-1160 (1989).
7) Sakai, S., Yamaguchi, Y., Hiraki, K., Kodama, Y. and Yuba, Y.: An Architecture of a Dataflow Single Chip Processor, *Proc. ISCA 89*, pp. 46-53 (1989).
8) Sakai, S., Kodama, Y. and Yamaguchi, Y.: Design and Implementation of a Circular Omega Network in the EM4, *Parallel Computing*, Vol. 19, No. 2, pp. 125-142 (1993).
9) Kodama, Y., Koumura, Y., Sato, M., Sakane, H., Sakai, S. and Yamaguchi, Y.: EMC-Y: Parallel Processing Element Optimizing Communication and Computation, *Proc. ICS 93*, pp. 167-174 (1993).
10) 坂根，児玉，佐藤，山名，坂井，山口：並列計算機 EM-X のプロセッサ・ネットワークインタフェースの最適化の検討，情報処理学会研究報告 ARC-104-14, pp. 105-112 (1994).

（平成 6 年 9 月 20 日受付）
（平成 7 年 1 月 12 日採録）

児玉 祐悦（正会員）

昭和 37 年生．昭和 61 年東京大学工学部計数工学科卒業．昭和 63 年同大学院工学系研究科情報工学専門課程修了．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，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．現在，情報アーキテクチャ部計算機方式研究室に所属．

坂根 広史

昭和 41 年生．平成 2 年山口大学工学部電子工学科卒業．平成 4 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，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，神経回路学会各会員．

佐藤 三久（正会員）

昭和 34 年生．昭和 57 年東京大学理学部情報科学科卒業．昭和 61 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．平成 3 年より，通産省電子技術総合研究所勤務．現在，同所情報アーキテクチャ部計算機方式研究室主任研究官．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等の研究に従事．日本応用数理学会会員．

坂井 修一（正会員）

昭和 33 年生．昭和 56 年東京大学理学部情報科学科卒業．昭和 61 年同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．平成 3 年 4 月より 1 年間米国 MIT 招聘研究員．平成 5 年 3 月より RWC 超並列アーキテクチャ研究室室長，現在に至る．計算機システム一般，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．情報処理学会研究賞（平成元年），同論文賞（平成 2 年度），元岡記念賞（平成 3 年），日本 IBM 科学賞（平成 3 年）各受賞．

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，データフロー計算機などの研究に従事．現在，情報アーキテクチャ部計算機方式研究室長．工学博士．1991 年情報処理学会論文賞．著書「データ駆動型並列計算機」（共著）．電子情報通信学会会員．