計算機アーキテクチャ 130－17
（1998. 8. 6）

細粒度並列計算機 EM-X におけるキャッシュメモリアーキテクチャ

坂根広史[†,††] 本多弘樹[††] 弓場敏嗣[††]
児玉祐悦[†] 山口喜教[†]

本稿では、並列計算機 EM-X をテストベッドとし、細粒度並列処理機構を持つ分散メモリ型アーキテクチャに適したローカルキャッシュ構成法ついて基本的な考察を行う。EM-X は主記憶に高速 SRAM を採用しているため、キャッシュメモリを持つことなく1クロックでローカルメモリにアクセスできる。しかしながら、今後クロックの高速化やメモリ容量の拡張を図るには、命令実行とメモリアクセスのスループットのギャップを緩衝するために、キャッシュの採用が不可欠となる。EM-X の細粒度並列処理サポート機構は数種類の異なったメモリアクセスを必要とし、それぞれのアクセスパターンには固有の特性がある。いくつかの並列プログラムを実行させた際のメモリアクセスパターンを調べ、それに基づいたキャッシュアーキテクチャの構想について述べる。

A Design of Cache Architecture for the Fine-grain Multithreaded Computer EM-X

HIROFUMI SAKANE,[†,††] HIROKI HONDA,[††] TOSHITSUGU YUBA,[††]
YUETSU KODAMA[†] and YOSHINORI YAMAGUCHI[†]

In this paper, we discuss a cache architecture suitable for distributed-memory multiprocessors which support fine-grain parallel processing by hardware. The cache design for the local memory would impact on the system performance since the support mechanisms for fine-grain operations require several kinds of memory access frequently. To quantify the individualities of the support units, we investigate behavior of the memory access pattern of a few benchmarks on the EM-X multiprocessor which allows that each memory access takes just one clock by adopting fast SRAMs as main storage. We also provide typical I and D caches behavior on the modified model of EM-X for the benchmarks measured by a trace-driven cache simulator assuming the processing element EMC-Y includes the caches. Finally, we discuss experimental results and present a preliminary cache model for the EMC-Y.

1. はじめに

効率的な細粒度並列処理をハードウェアでサポートする分散メモリ型並列計算機では、種々のローカルメモリ操作を必要とする。高速同期やスレッドの高速起動、通信バッファはその例である。要素プロセッサ内部でこれらのサポート機構が行うメモリ操作は、それぞれの固有のアクセスパターンだけでなく固有のメモリ空間を持つことが考えられるが、ここでは、ハードウェア資源量の上限のために各サポート機構は単一のローカルメモリを共有すると仮定する。そのようなシステムでは、ピン数制限等により各サポート機構を充足するだけのメモリバンド幅が用意できない場合、メモリアクセス競合の問題が発生する。また、主メモリアクセスのレイテンシ削減のためにキャッシュメモリを用いる場合、サポート機構毎にアクセス領域やパターンが異なるという性質がキャッシュミス率悪化を引き起こす可能性がある。本研究の目的は、分散メモリ型並列計算機 EM-X をテストベッドとして、メモリアクセス競合の削減と各サポート機構固有の特性を有効利用するためのキャッシュアーキテクチャを構築し、より効率的な細粒度並列処理を実現することである。

EM-X は RISC パイプラインを持つ高速な逐次スレッド実行機構と、データ駆動モデルに基づくスレッド起動機構を融合し、さらにリモートメモリアクセス機構を持つことにより、効率的な細粒度並列処理を可能としている。EM-X の主メモリは高速 SRAM で構成され、そのアクセス時間は命令実行クロックのサイクル時間より短いため、キャッシュメモリを持つことなく1クロックでローカルメモリにアクセスできる。このことは、パケットバッファリングやメモリ上でのマッチング処理の効率化、スレッド実行パイプラインの単純化等に大きく寄与している。これらは細粒度並列処理のサポートとスレッドの逐次実行の高速化を担っており、効率的な並列処理を可能とする。一方、近年のマイクロプロセッサをはじめ、商用の高速計算機の命令実行クロック周波数は 1GHz に達しようとしており、速度向上の緩やかなメモリとのスループットのギャップが拡大しつつあることから、高性能キャッシュによるギャップの緩衝が不可欠となっている[1]。EM-X アーキテクチャにおいても、クロック周波数の高速化・メモリ容量の拡大を図るには、キャッシュメモリの採用が必要となる。

キャッシュの利点は、プロセッサ・メモリ間のギャップによる性能低下を軽減するだけではない。典型的なマイクロプロセッサに見られるキャッシュにおけるハーバード・アーキテクチャ、すなわち命令キャッシュとデータキャッシュの分離方式では、メモリバンド幅の実質的な拡大により性能向上が見込まれる。一方 EM-X の各細粒度並列処理サポート機構はそれぞれ性質の異なるメモリアクセスを要求するが、要素プロセッサチップの外部メモリポートではそれらのメモリアクセスが重畳

† 電子技術総合研究所
Electrotechnical Laboratory
†† 電気通信大学
University of Electro Communications

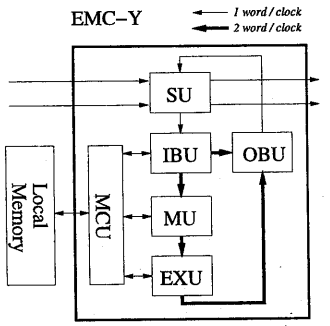


図1 EMC-Y の構成

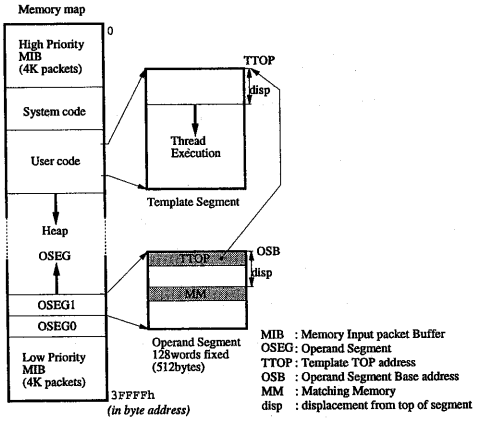


図2 メモリマップ

化されており、アクセス競合が発生する。そこで分離キャッシュを適用しサポート機構毎に個別のキャッシュを持つことにより競合がなくなり、バンド幅拡大の効果を期待できる。各サポート機構のアクセスパターンには固有の局所性があり、それぞれに適したキャッシュ構成をとることができる。ただし、キャッシュを分離すると各キャッシュの実質的なサイズが少なくなる問題があり、詳細な検討が必要である。なお分散共有サポートの議論は別の機会に譲り、本稿ではローカルメモリアクセスの効率化に焦点を当てる。

EM-X に適したキャッシュアーキテクチャを提案するために次の段階を踏む。まずスタートラインとして、1) 一般的なI,D キャッシュを EM-X のメモリインターフェースに挿入した場合の振る舞いを調べる。2) 次にメモリアクセスの特性を調べ、適切なメモリアクセス分離方式について検討する。3) 検討の結果導かれるアーキテクチャについて性能評価を行う。キャッシュの評価は、汎用キャッシュシミュレータおよび EM-X クロックレベルシミュレータによって行う。本稿では、1) および 2) の範囲を論じ、3) については継続課題とする。2 章で EM-X のアーキテクチャとソフトウェアの実行形態について述べる。3 章ではベンチマークプログラムのアドレストレースとキャッシュシミュレータを用いてキャッシュの振る舞いを調べる。4 章では新しいキャッシュシステムのアイデアについて議論する。最後に 5 章で総括と今後の課題を述べる。

2. EM-X アーキテクチャ

EM-X は、現在 80 台の要素プロセッサ (PE) で構成されるプロトタイプが稼働しており、その性能評価が進められている[3),4),6),7)]。この章では、EM-X のメモリアクセスに関わる機構と、関数およびスレッド起動方式について説明する。

2.1 EMC-Y

要素プロセッサ EMC-Y(図 1) は CMOS ゲートアレイのチップ上にネットワークインターフェース、マッチング機構、演算実行部を実装している。主メモリには高速 SRAM を用いており、1PE 当たり 1Mword(word = データ 32 bit+ タグ 6 bit) 実装されている。図 2にローカルメモリマップを示す。EM-X は実記憶システムであり、先頭から、高優先度パケットバッファ (High Priority MIB: Memory Input packet Buffer) に続いてシステム領域、ユーザコード領域が置かれる。最後部からは低優先度パケットバッファ (Low Priority MIB) に続いて、オペランドセグメント (OSEG) と呼ばれる固定長の関数インスタンスのワーク領域が並ぶ。

次に、EMC-Y の中でメモリアクセスに関わる細粒度並列処理サポート機構 (あるいは単にユニット) IBU, MU, EXU および MCU について詳細に説明する。

2.1.1 入力バッファ部 IBU

IBU は入力パケットのバッファリングのほかに、リモートメモリアクセス処理およびマッチング処理の一部を受け持つ。

オンチップパケットバッファとして、高優先度・低優先度それぞれに 8 パケット分の FIFO キューが実装されている。この FIFO が FULL となった場合には自動的に主メモリ上の MIB へ当該パケットがセーブされ、FIFO FULL が解消されるとリストアされる。MIB はシステム起動時に高低優先度ともそれぞれ 4K パケット分確保され、リングバッファとして管理されている。パケットは 2 ワードで構成されているため、セーブとリストアにはそれぞれ 2 回のメモリアクセスを伴う。

リモートメモリアクセスパケットが到着すると、IBU はバッファリングを行うことなく要求されたメモリアクセスを行う。書き込みの場合は指定されたメモリへデータを書き込んで終了し、読み出しの場合はメモリから読み出したデータをパケットとして出力バッファ OBU を経由して直接送り返す。

到着パケットが 2 入力マッチングの一方のデータであれば、アドレス部で指定されたマッチングメモリ (MM : 1 word) を読み相手データの確認を行う。相手が存在していなければパケットのデータを MM に書き込んで処理を終了する。

これらの各メモリアクセス要求はまず IBU 内で調停が行われ、単一ポートにまとめられて MCU と接続されている。その調停の優先順位は、リモートメモリリクエストサービス & マッチング > リストア > ストアの順である。

2.1.2 マッチング部 MU

MU はダイレクトマッチング機能を担う。2 入力パケットであれば、マッチング後半処理のために MM を 1 回ずつ読み書きする。次に TTOP を読み出し、パケットのアドレス部で示される変位 disp が加算されてスレッドの先頭アドレスとなる。MU の最後のステージは EXU の最初の命令フェッチステージと重なり、スレッドの先頭命令のフェッチ指令を出す。1 入力パケットの場合は MM のアクセスはなく、最初に TTOP を読み、次に命令フェッチを行う。システム予約スレッドが指定されていると TTOP 読み出しが省略される。

2.1.3 演算実行部 EXU

MU によりフェッチされた命令を EXU がメモリから直接受け取り、スレッドを起動する。一旦スレッドが起動すると、

stage	r/w	IBU	MU	EXU
1st	read		Matching-read I-fetch	I-fetch
2nd	read	Remote-read Matching-read Packet-restore		Load
	write	Remote-write Matching-write Packet-save	Matching-clear	Save

表1 EMC-Y のメモリアクセス種類

EXU は通常の RISC パイプラインとして命令フェッチと実行を繰り返す。メモリアクセスは命令フェッチおよびメモリアクセス命令 (load, store) 実行によるものの2種類である。

2.1.4 メモリ制御部 MCU

IBU, MU, EXU からのメモリリクエストは MCU で調停が行われる。それぞれのユニットのメモリリクエストを表1にまとめる。このように、一般の逐次処理向けマイクロプロセッサと異なり EMC-Y はメモリアクセスの種類が多い。チップのメモリインターフェースはピン数の制限から 38bit 幅1ポートのみであるが、競合を減らすために EMC-Y では1クロックに2回メモリアクセスすることによって実質2ポート (1st/2nd stage) となっている。これにより、例えば命令フェッチとメモリアクセス命令の実行を同一クロックで行うことができる。同一 stage のアクセス間では調停が必要であり、その優先順位は 1st stage では EXU > MU、2nd stage では原則として MU > EXU > IBU である。ただし 1st/2nd の MU オペレーションは不可分であり、2nd は 1st での要求受付が条件である。これらの調停では EXU の連続命令実行を優先していることになるが、命令実行時の IBU や MU のメモリリクエストは待たされ、その結果オーバヘッドとなることがある。

2.2 関数呼び出し方式

EM-X における関数呼び出し手順は、メモリ上で管理されているフリーリストから関数インスタンスとして OSEG を一つ取り出すことから始まる。OSEG は関数が呼ばれる都度取り出され、関数終了後フリーリストに戻される。フリーリスト要素の実体は各 OSEG の先頭アドレスに格納されており、関数コードとの対応を示す TTOP と入れ替えることによって OSEG が確保される。リモート PE の関数の場合は呼び出しに先行してパケットにより OSEG 確保要求を行う。要求された PE ではシステムが用意している短いスレッドが OSEG を取り出し、そのベースアドレス (OSB) を返送する。その後起動パケット投入によって関数のスレッドを起動する。OSEG は2入力スレッドのための MM が置かれるほか、各種ワークエリアとして使用できる。

EM-X ではプログラミング環境として EM-C が用意されており[2]、上記の関数呼び出し手順に加え、引数、auto 変数、コンティニュエーション (関数リターン先) 退避、スレッド切り替え時のレジスタ退避等の領域として OSEG を用いるコンベンションが定められている。

2.3 スレッド起動方式

関数はスレッドの集合である。EM-X ではスレッドの切り替え点はプログラム内で明示的に指定されている。切り替え点ではスレッド起動の都度 MU が TTOP 読み出しとスレッド先頭の命令フェッチを行う。TTOP 読み出しは直前のスレッドの最終命令フェッチとオーバラップできず、競合が生じた場合は1クロック分のオーバヘッドとなる。システム予約のスレッドが起動する場合は TTOP 読み出しを伴わず、対応する OSEG なしでスレッド先頭アドレスが直接決定される。

なお現在の EM-C コンパイラでは、2入力マッチングのサポートはシステム予約の I-structure スレッドのみであり、ユーザスレッドの起動は1入力である。

3. 基本キャッシュ性能

EM-X のメモリシステムに一般的な命令・データキャッシュを与えた場合の振る舞いをシミュレーションで調べた。まず EM-X の RTL シミュレータによる EMC-Y のメモリアクセストレースを得た。トレースデータは、メモリアクセスの内訳を調べるためにも使用した。次にそのトレースデータを入力とし、トレース駆動キャッシュシミュレータを用いて3種類のベンチマークプログラムのキャッシュミス率を測定した。命令キャッシュには EXU と MU の命令フェッチパスを導き、残りのパスは全てデータキャッシュに割り当てた。ベンチマークプログラムの実行トレースに対して、キャッシュサイズ・キャッシュブロックサイズ・associativity を変化させた場合のミス率を得た。キャッシュシミュレータは、ウィスコンシン大学で開発された WARTS(Wisconsin Architectural Research Tool Set[8]) の中の DineroIII および Tycho を用いた。

3.1 ベンチマークプログラム

matmul (matrix multiply)

行列乗算の並列計算。行方向に分割配置し、ローカルな行ベクトルとリモートの列ベクトルの内積を最内ループで計算する基本的なアルゴリズムである。最内ループ1イテレーション毎に1ワードずつ規則的にリモートリードを発行する。レイテンシ隠蔽のために4スレッド/PE で実行。プログラムは EM-C で記述。64PE、行列サイズ 256 × 256。キャッシュのウォームスタートをするために 100 万クロック経過後の 500 万クロックの測定を行った。

radix (radix sort)

並列 radix ソート[6]。ワード単位のリモートライトを不規則アドレスで発行。EM-C によるプログラム。64PE、総データ数 4M ワード、radix:256 で実行。配列初期化部分を除外するため 300 万クロック経過させ、その後の 500 万クロックで測定した。

fib (fibonacci)

フィボナッチ数を再帰的に求めるプログラム。平均スレッド長が短くパケット出力頻度が高いという特徴を持つ。アセンブリ言語で記述。2入力マッチングによってデータ駆動的に結果を集積する。80PE で $fib(27)$ を計算。全実行にかかった 21 万 8522 クロック分を測定。

3.2 測定結果

代表の PE を1個選び (PE0) 各ベンチマークのメモリアクセス内容を調べた結果を表2に示す。1クロックに 2stage あるためメモリアクセス数はクロック数より多い。1) ミス率測定対象クロック数。2) 全メモリアクセス数。3) 命令フェッチ数。クロック数との比で演算部の稼働率がわかる。4,5) それぞれ OSEG, HEAP 領域に対するメモリアクセス命令実行数。6)IBU によるリモートアクセス回数。今回の測定では全てのリモートアクセスがヒープ領域に対するものであった。7) スレッド起動回数は MU の命令フェッチ回数である。8) ユーザスレッド起動回数は TTOP の読み出し回数である。9) 使用 OSEG 数とは、計測中にアクティブになった OSEG の数である。10) 空間使用率は OSEG 内 128word 中の何ワードが使用されたかを示す。アクセス回数の偏りは表に反映して

	matmul	radix	fib
1) クロック数	5,000K	5,000K	218.5K
2) 全メモリアクセス数	6,719K	6,300K	221.2K
3) 命令フェッチ数	4,847K	4,934K	78.7K
4) OSEG アクセス命令数	1,252K	1396	12.5K
5) HEAP アクセス命令数	314K	1,185K	4325
6) リモートアクセス数(HEAP)	153K	180K	0
7) スレッド起動回数	153K	208	19.0K
8) ユーザスレッド起動回数	153K	160	14.8K
9) 使用 OSEG 数	4	3	1816
10) OSEG 空間使用率(word)	11	4 ~ 22	5
11) 関数呼び出し回数	0	56	1815
12) MIB アクセス数	0	0	57.5K
13) ピーク MIB パケット数	0	0	2861
14) ピーク アクティブ MM 数	0	6	1815
15) 平均スレッド長	31.6	23.7K	4.4

表2 メモリアクセス内容

いない。11) 関数呼び出し回数。12)MIB アクセスは2回でパケット1個分である。13)MIB ピークパケット数は MIB が保持する有効パケットの瞬間ピーク数を表す。14) ピーク アクティブ MM 数は、相手待ち MM 個数の瞬間ピーク数である。15) 平均スレッド長は命令実行クロック数(シミュレータで計測)をスレッド起動数で割ったものである。

同 PE のデータキャッシュのミス率のプロットを図3~5に示す。キャッシュサイズを 512B ~ 512KB まで変化させ、1 ~ 8 の associativity についてプロットしたものである。キャッシュブロックサイズは 32B のものを示した。ライトポリシーは全てライトバック、ライトアロケートとした。なお、使用したベンチマークは比較的小さなプログラムであるため命令キャッシュのミス率は 0 に近い。今後、命令キャッシュの議論をするために十分な大きさのプログラムを用いる、あるいは複数の関数でマルチスレッド実行を行う等、設定問題を増やす必要がある。

matmul

各 PE が受け持つ行列データは $256 \times (256/64) \times 4 = 4$KB が3セットである。このうち2セットが内積ループにおける自 PE アクセスと他の PE からのリモートアクセスによって繰り返し読み出されるため、主要なワーキングセットは 8KB である。これによりキャッシュサイズ 8KB 以上、2-way 以上でミス率は 0.2% 未満となっている。direct-mapped では衝突ミスの影響が残る。これらの現象はキャッシュを持つ計算機で科学技術計算を行う場合に一般的に見られるものに近い。

測定クロック期間中に使用された OSEG 数は4個であり、これはスレッド数に対応し、ほぼ均等に繰り返しアクティブになっている。最内ループの1イテレーション毎にスレッドが切り替わるためスレッド起動回数が多く、MU の TTOP 読み出しが待たされる機会が多い。実行スレッドが比較的単純なループで構成されているため変数の数が少なく OSEG 空間使用率は 11word と高くないが、連続した領域であり空間局所性は高い。一方 OSEG に対するメモリアクセス命令は命令実行全体に対して 25% を占めており、時間的局所性も高い。OSEG アクセスの効率は性能に大きく影響するが、ミス率の要因調査により、キャッシュサイズが小さくなるほどヒープ領域アクセスとの衝突によって OSEG のミス率の割合が大きくなっていることがわかった。

radix

各 PE が受け持つ被ソートデータは 64KB であり、転送先バッファとしてさらに同サイズが必要となる。加えて radix 分のカウンタの配列があるほか、全 PE の総和・部分和を求めるためのマッチング領域が少量使われている。以上のデータはヒープ領域に確保される。このプログラムでは、データ転送ループがスレッドを中断することなく実行されているため平均スレッド長が長い。スレッドが継続している間はレジスタ退避が不要なため OSEG アクセスは少ない。メモリアクセスのほとんどは命令フェッチとヒープデータアクセスであった。128KB のキャッシュサイズまでは容量ミス (Capacity miss) に加えランダムアクセスのため associativity によらずミス率が 10% 以上となっている。それを越えて 256KB 以上になると大きく減少するが、ランダムアクセスによる競合の影響が残っている。

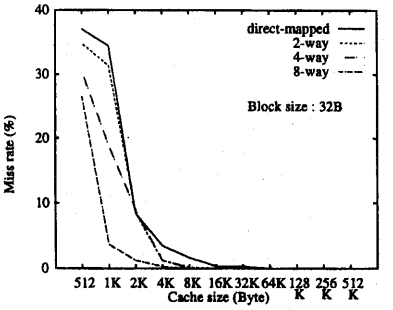


図3 matmul

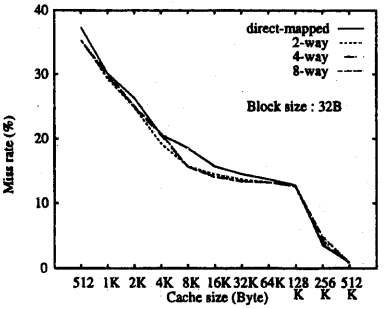


図4 radix

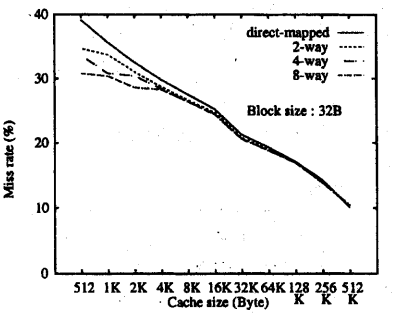


図5 fib

fib

データ駆動的に細粒度実行が進むため通信とマッチングの頻度が非常に高い。データメモリアクセスは MIB, MM, TTOP だけである。fibonacchi の計算ノード1個につき OSEG 1個がアロケートされ、ノードの計算が終わるまで保持されるので OSEG を多数消費するが、OSEG 内の空間利用率と空間局所性は低い (5 の内訳 : TTOP と 4 個所の MM)。計算の進行に対して通信量が多く、かつ1つの PE から見た通信量の

増減が急であるためパケットの MIB への退避の頻度が高い。MIB アクセスを観測すると 4K パケットのリングバッファが3周半しており、パケットの退避量はピーク時に容量の7割であった。*fib* 関数に与える引数を 27 →28 とすると 4K パケットのバッファが溢れた。演算効率が良くない理由はネットワークの混雑による。この混雑は各 PE の IBU における MIB へのパケット退避リクエストが EXU や MU との競合に負けることが多いことが原因である。キャッシュサイズを増やしてもミス率の減少が緩やかな理由は、OSEG および MIB の初期ミス (Compulsory miss) が多いためと、OSEG と MIB が conflict を起こすためである。

4. キャッシュ構成

2章で説明したように EM-X ではユニット別に多様なメモリアクセスパターンが存在し、それらが重畳されて単一メモリ空間に投入されている。そこで、ユニット相互の干渉と分離キャッシュの効果について検討する。干渉とは、各ユニットのアクセスパターンが違うことから、それらをひとつのキャッシュにまとめると cache pollution によって性能が低下することである。キャッシュの分離の得失を以下に列挙する。

- 利点
  - ユニット同士のアクセス競合削減 (バンド幅拡大)
  - cache pollution 削減
  - キャッシュ毎の最適化
- 損失
  - 領域が重なる場合のコヒーレンス維持コスト
  - 各ユニットから見たキャッシュサイズの減少

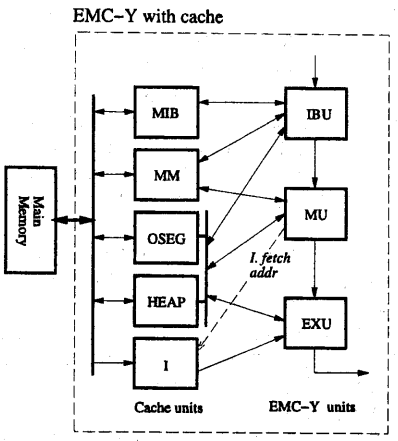
各ユニットが行うメモリアクセスの種類別にキャッシュを想定しながら、上に挙げた得失を定性的に検討することにより、EMC-Y 内に 5 個のキャッシュユニットをモデル設定した (図6)。EMC-Y ユニット (IBU,MU,EXU) のメモリアクセスを各キャッシュユニットが受け、キャッシュの先に主メモリが接続される。キャッシュユニット毎の最適化は、それぞれを必要最小限のサイズに抑えることを含む。それにより一定のハードウェア資源のもとで必要なところへ多くの資源を割り当てることができ、実効キャッシュサイズの減少の問題も軽減されると考えられる。それぞれの得失がどのように全体性能に影響するかを知るには、キャッシュの振る舞いをより詳細にシミュレーションすることが必要である。本稿では各キャッシュユニットの説明にとどめ、それらの評価・再検討は継続課題とする。

4.1 命令 (I) キャッシュ

一般の命令キャッシュと同じように読み出し専用である。命令フェッチアドレスは MU と EXU のいずれかから発行されるが、フェッチデータは EXU のみに返る。命令フェッチパターンはマルチスレッド実行形式に依存する。同じ関数コードをマルチスレッド実行する場合は、同じ関数の命令コードがスレッド単位でインターリーブされてフェッチされる。1スレッドの命令数が特に多くない限り、ミス率は小さいと考えられる。複数の関数をマルチスレッド実行する場合は、異なる命令列がインターリーブされるため、ミス率は高くなる。後者に対応するには associativity を大きくするか、スレッド単位で専用キャッシュブロックを設けることが考えられる。

4.2 データ (HEAP) キャッシュ

EXU が行うデータメモリアクセスには、OSEG を対象としたものとヒープ領域を対象としたものがある。現状のメモリアクセスでは両者を区別していないが、アクセスパターンを調べると、一般に OSEG のアクセスは変数操作やレジスタ退避等のために時間的・空間的局所性が高い。また、関数呼び出し



MIB : Memory Input packet Buffer
MM : Matching Memory
OSEG : Operand SEGment
HEAP : HEAP area
I : Instruction

図6 キャッシュ構成

方式に起因する時間的・空間的な規則的構造も存在する。一方ヒープ領域は OSEG に比べアクセスパターンが分散する傾向にある。このような特性の違いにより OSEG 用キャッシュは分離した方がよいと考えられる (→4.3節)。ただし EXU からのアクセスポートは OSEG・ヒープ合わせて1本でよいため、完全な分離ではなくポート等を共有することも考えられる。OSEG 処理と区別するため、ヒープ領域のデータキャッシュは HEAP キャッシュと呼ぶ。

データアクセスの種類としてはこのほか、IBU 経由のリモートメモリアクセスがある。EXU のデータアクセスと IBU のメモリアクセスは競合するため、その点では分離が望ましい。しかしながらリモートアクセスとローカルアクセスは共通のロケーションとなることが多いと考えられ、分離するとコヒーレンスの維持にコストがかかる。一方、リモートアクセスのレイテンシ増加のペナルティはローカルアクセスに比べると相対的に安価である。これらのことから、リモートアクセスについては分離しない、あるいは IBU からのアクセスはキャッシュしない、という方針が導かれる。このコヒーレンス維持コストの検討は今後の課題である。

4.3 OSEG キャッシュ

OSEG 専用のキャッシュユニットを仮定した場合、そのアクセスの空間的・時間的構造を利用することにより、高いキャッシュ動作効率を実現できる可能性がある。サイズが固定であることや、先頭に置かれるフリーリスト要素と TTOP、EM-C 関数呼び出しコンベンションの利用、新しい OSEG 確保の予測性等である。`fib` 以外の実験結果では、アクティブな OSEG は少なくてよいことがわかる。関数ネストが深くなってもネストレベルの浅い関数フレームはしばらく使用されないため、他のキャッシュブロックによって追い出されても問題は少ない。マルチスレッド実行ではスレッド数分の OSEG が確保されるが、効率の良いスレッド数は高々4程度である[5]。これらの場合はキャッシュに納められる OSEG 数は少なくてよい (数個～10数個)。`fib` のように OSEG 消費が動的に大きく変化するプログラムについては、OSEG 確保を予測するプリフェッチ方式をとることが有望であると考えられる。

OSEG キャッシュにおいて、キャッシュブロックのサイズ

の決定は重要な問題である。OSEG 内の空間がほとんどが使われるとすると、OSEG 1個分を単位としてキャッシュすることが望ましい。ミスペナルティは大きいが、ミスのほとんどは関数呼び出し時に起きるとすると、matmul や radix のようにOSEG の再利用が多い場合には全体の性能に与える影響は小さい。一方、空間使用率が低い場合はブロックサイズが小さいことが望ましい。EM-C では OSEG の先頭付近および後端付近が集中的にアクセスされる傾向があり、その利用も考えられる。

OSEG 内へのアクセスは、このほかに IBU によるマッチング処理前半、MU によるマッチング処理後半と TTOP 読み出し、および IBU によるリモートアクセスがある。マッチングについては新たに MM キャッシュを仮定して、節を改めて議論する (4.5節)。リモートアクセスについては、4.2節で述べた理由から分離は行わないこととする。

4.4 MIB キャッシュ

パケットバッファは通信量の変化に適応できるよう可変サイズが望ましいため、MIB の主メモリへの割り付けが有用である。実際、プログラムによって MIB へのパケット待避量が大きく異なることは前章で述べた。一方、パケットバッファリングの処理に時間がかかるとネットワークを停滞させるとともに、スレッド起動が遅れることによるアクティビティの低下を引き起こす。従って主メモリのアクセスレイテンシを低く抑えることが重要になり、適切なキャッシュが必要となる。MIB は独立したメモリ領域を持つためアクセスの分離が容易であり、それにより他との競合がなくなる利点がある。リングバッファとしての MIB アクセスはシーケンシャルなメモリアドレスの変化を生じ、その方向は一方向のみである。この性質はプリフェッチ方式ならびに大きなキャッシュブロックの有効性を示唆する。さらに、パケットのストアとリストアは必ずペアの関係であり、リストア後 MIB 上のそのデータは不要という性質がある。これを利用することにより競合ミス回数や書き戻し回数を減らせるとともに、キャッシュサイズを縮小できると考えられる。

MIB キャッシュではセーブ優先かリストア優先かの選択が問題となる。すなわち、溢れたパケットのセーブ処理に時間がかかるとネットワークの混雑の原因となり、リストア時間が長くなるとスレッドアクティビティの低下を招く。パケット到着頻度により最適なパラメータが異なるため[3]、多くのベンチマークやアプリケーションによるテストが必要である。このほか、キャッシュの代りにオンチップ FIFO キューを大きくする方法がある。MIB ライトキューとの併用で、キャッシュと同等の効果が得られないか検討中である。

4.5 マッチングメモリ (MM) キャッシュ

IBU と MU で行われる 2 入力マッチング処理は、1 write / 1 read、マッチング成立後のデータは無効という点で MIB と類似点がある。アクティブな MM のピーク量はプログラムにより大きく変化するため、キャッシュサイズは慎重に決定する必要がある。また、2 入力マッチングによるスレッド起動や I-structure の種々の使用形態を考えると、空間的局所性が乏しい場合と十分存在する場合の両方が想定できる。このためブロックサイズの決定も慎重さを要する。*fib* のような例では各 OSEG においてマッチング位置が同一オフセットとなるためキャッシュ競合が起きやすい。この回避には associativity を大きくとる。簡単なハッシュ機構の併用も衝突の低減に有効である。

5. ま と め

細粒度並列処理機構を持つ分散メモリ型並列計算機は、要素プロセッサ内で種々のメモリ構造を扱う。本稿ではそれらの性能向上を図るためキャッシュアーキテクチャの最適化を検討した。EM-X をターゲットとして RTL シミュレータによってベンチマーク実行時のメモリアクセストレースをとり、トレース駆動キャッシュシミュレータによって I, D キャッシュを追加した場合のミス率を測定した。メモリアクセストレースの解析を行うとともに、ベンチマークの性質とミス率の振る舞いの関係を考察した。その上で、EMC-Y 内の各サポート機構(ユニット)によるメモリアクセスを分離することによる得失を定性的に評価し、効果的と考えられるキャッシュモデルを構築した。

本稿で示したキャッシュミス率は EM-X のメモリアクセストレースをもとに測定したものであり、ミスペナルティがシステムの動作に与える影響は考慮していない。現実にはペナルティ時間の間に要素プロセッサの各ユニットの状態が変化するため、複雑な挙動を示すと考えられる。各キャッシュユニットと主メモリ間のトラフィックの競合がペナルティ時間に与える影響も考慮しなければならない。本稿で挙げた種々の課題への取り組みに加え、現実に即した現象の観察と理解のため提案するキャッシュモデルを EM-X シミュレータに組み込み、それが性能に与える影響を評価していく予定である。

謝　辞

本研究を遂行するにあたり御指導、御討論いただいた電子技術総合研究所の大蒔情報アーキテクチャ部長ならびに並列アーキテクチャラボの同僚諸氏、特に、有益な議論をしていただいた小池汎平氏と山名早人氏に感謝いたします。

参 考 文 献

1) Hennessy,J.L., Patterson,D.A.: Computer Architecture a Quantitative Approach – Second Edition, Morgan Kaufman, Chap.5, pp.373-483, 1996.
2) 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.
3) 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ー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
4) 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.
5) Sakane,H., Sato,M., Kodama,Y., Yamana,H., Sakai,S., Yamaguchi,Y.: Dynamic Characteristics of Multithreaded Execution in the EM-X Multiprocessor, Proc. of PERMEAN '95, pp.14-22, 1995.
6) 児玉, 坂根, 佐藤, 山名, 坂井, 山口: 細粒度通信機構を用いた radix ソートの実行, 情報処理学会論文誌, Vol.38, No.9, pp.1726-1735, 1997.
7) 佐藤, 児玉, 坂根, 山名, 坂井, 山口: 細粒度通信機構をもつ並列計算機 EM-X による疎行列計算の性能評価, 情報処理学会論文誌, Vol.38, No.9, pp.1761-1770, 1997.
8) http://www.cs.wisc.edu/~larus/warts.html