計算機アーキテクチャ 134－2
（1999．8．2）

並列計算機用要素プロセッサの細粒度同期機構におけるキャッシュ方式の検討

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

分散メモリ型並列計算機における要素プロセッサの細粒度同期機構が行うメモリアクセスに注目し、その効率が全体の並列処理性能に与える影響と、キャッシュ機構を用いたオーバヘッド削減の方法について議論する。本稿でとりあげる並列計算機 EM-X は、ハードウェアによる細粒度同期機構を持つ。その同期機構で生じる複数のメモリアクセスに対し、要因別のオンチップ専用キャッシュを新たに設けることにより、メモリアクセス競合の低減と、キャッシュごとの最適化による性能向上を同時に図る。キャッシュ方式検討の予備評価として、アーキテクチャシミュレータによってマルチポート化による性能向上率を測定した。また、同期機構におけるキャッシュミス率と並列処理性能の関係を測定した。その結果として、性能改善の可能性を示すとともに、並列プログラムごとに異なる同期方式および粒度が、キャッシュの効果に大きく影響することを示す。

A Design of Cache Architecture for Fine-grain Synchronization

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

Improving the efficiency of fine-grain synchronization is one of the key issues in fine-grain parallel computing. Fine-grain synchronization, however, poses a technical challenge as several different types of simultaneous accesses to memory can cause serious contentions. This report presents a technique that helps eliminate the overhead associated with fine-grain synchronization for distributed-memory multiprocessors. In particular, we introduce a new cache scheme that uses several different caches for synchronization. The main idea is that each cache with a dedicated functionality can handle a particular type of memory accesses, as well as being able to reduce the memory contention at the fine-grain synchronization point.

To assess the potential impact of the proposed approach on the 80-processor EM-X multiprocessor, we performed various simulations using the EM-X architectural simulator. Simulation results indicate that the overhead associated with the fine-grain synchronization mechanism can be fairly large for problems that require fine-grain memory accesses for synchronization. To understand the proposed approach, we present the effects of wider memory bandwidth. The relationship between the parallel performance and various cache miss ratios are also discussed for possible performance improvement. We find that the separate cache effects depend much on the granularity of the parallel program and the synchronization mechanism used.

1. はじめに

分散メモリ型並列計算機において、柔軟かつ高性能な並列処理を実現するには、プロセッサ間の通信および同期処理をいかに効率よく行うかが問題となる。そのため、細粒度同期処理をハードウェアで支援する種々の方法がこれまでに考え出されてきた[1),2)]。分散メモリ型並列計算機における同期や通信の支援機構では、それらの処理にともなう種々のローカルメモリ操作が行われる。待ち合わせ処理、スレッドの高速起動、通信バッファ等はその例である。これらのメモリ操作に対してアクセスレイテンシが長くなると、同期処理にかかる時間が長くなり、並列処理の効率を低下させてしまう。レイテンシが長くなる要因はいくつかある。

- 年々拡大するプロセッサとメモリの動作速度差という一般的な問題。
- ピン数制限等により、各支援機構のアクセス要求を充足するだけのメモリバンド幅が用意できない場合に生じる、メモリアクセス競合の問題。
- キャッシュを用いてレイテンシ削減を図る場合の、キャッシュ構成とアクセスパタンが整合しないことによるミス率増大。
- 単一のローカルメモリあるいはキャッシュを各支援機構で共有する場合の、アクセスパタンの性質の違いに起因するキャッシュ汚染の問題。

これらのうち、アクセス競合を低減するためのバンド幅の拡大は、近年のマイクロプロセッサにも見られる、データキャッシュと命令キャッシュを分離するハーバード・アーキテクチャのアプローチと同種の問題である。細粒度同期・通信支援機構を持つ並列計算機用要素プロセッサでは、上記のようにさらに多くのメモリアクセスが要求され

† 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

るため、バンド幅をより広く確保する必要がある。また、アクセスパタン整合の問題やキャッシュ汚染の問題は、単一のキャッシュを複数のアクセスパタンが共有する場合に生じる。これらのことから、バンド幅確保という点と、アクセスの種類に応じたキャッシュ最適化をするという点で、複数のキャッシュを設けることが性能向上につながると考えられる。

本稿では、まず細粒度並列処理支援機構を持つ分散メモリ型並列計算機の要素プロセッサについて、同期処理にともなうローカルメモリアクセスの問題について述べる。同期処理における分離キャッシュ方式を提案し、その性質を調べるためのテストベッドとして用いる分散メモリ型並列計算機 EM-X のアーキテクチャを述べ、その同期機構の性質について分析する。次に、EM-X の要素プロセッサ上におけるメモリアクセス競合の様子を具体的に調べ、競合を削減するためにマルチポート化した場合の性能向上率を評価する。さらに、要因別の専用キャッシュの設置によりマルチポートを実現した場合の性能評価として、そのキャッシュのミス率とペナルティ時間を与えて並列プログラムの実行時間を測定し、本方式の予備的な評価とする。これらの測定・評価には EM-X のアーキテクチャシミュレータを用いる。

2. 分散メモリ型細粒度並列計算機

分散メモリ型並列計算機において効率の良い細粒度並列処理を実現するためには、通信部と演算部が互いの要求・状況に応じて機敏かつ潤滑に動作する必要がある。そのことを追求するための要素プロセッサがいくつか専用に設計されてきた[1),2)]。なお本研究では、要素プロセッサは可能な限りシンプルな部品となることが望ましいという観点から、各メモリ要素の実体は原則として単一のローカルメモリに置き、必要最小限のポート数でアクセスされることを前提とする。

2.1 通信・同期にともなうメモリアクセス

一般に、分散メモリ型並列計算機の要素プロセッサでは、通信インタフェースのためのバッファリングや同期処理、演算部の高速起動機構等のために種々のローカルメモリ構造を持つ。性能を低下させないためにはそれらのアクセス要求を速やかに処理する必要がある。一般に、メモリデバイスはプロセッサの動作速度に対して低速であり、多くの場合その性能ギャップの緩衝機構としてキャッシュメモリが用いられる。通信・同期処理のメモリアクセスも同様であり、高速処理を満足するためにはキャッシュの適用が有効である。しかしながらキャッシュミスが起きると、ミス処理の期間中は演算部の要求あるいはネットワークからの要求に応答できなくなり、全体の効率を低下させる原因となる。細粒度処理の場合は特にその影響が大きいため、メモリシステムの効率化は重要である。

要素プロセッサにおける種々のメモリ構造へのアクセスには、そのアクセスを行うユニットや要因に応じた特有のパタンがある。キャッシュが存在する場合にはそのパタンに合わせた最適化が可能であり、かつ重要である。

2.2 分離キャッシュ

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・アーキテクチャにみられるように、キャッシュの分離により、メモリバンド幅の実質的な拡大による性能向上が見込まれる。並列計算機の要素プロセッサにおいては、さらに通信・同期支援機構のメモリアクセスが加わる。これらのアクセス要求は、通信・同期が発生する時に同時に複数発生し、競合を起こしやすい。通信・同期・演算のすべてのメモリアクセスを一つの統合キャッシュで受けるとすると、次章に述べる EM-X の例では、そのミス率は 20 ～ 40 %に達することがある[3)]。

これらのことからも分離キャッシュがより有効であると考えられ、これら支援機構に専用のキャッシュを持たせることにより、主メモリのバンド幅を最小限に抑えながら、バンド幅拡大の効果を期待できる。

ただし、キャッシュを分離すると次のような問題点がある。

・統合キャッシュに比べ、各キャッシュの実質的なサイズが少なくなる。

・コヒーレンス維持の問題が発生し、その維持コストがかかる。

前者に対する補償としては、分離した各キャッシュを最適化することにより、分離前と同じ性能なら小さなキャッシュにできる可能性があり、同じサイズならより高性能が得られる可能性がある。後者については、一般に通信や同期処理の種類によってメモリの使用目的が異なることからアドレス空間を共有する場合は少ないと考えられる。設計時には、なるべくそのようなコヒーレンス維持の不要な組み合わせを選ぶことが重要である。

3. EM-X アーキテクチャ

EM-X は、前章で述べたような要素プロセッサ内部に細粒度並列処理支援機構を持つ並列計算機であり、以下でそのアーキテクチャを説明する。

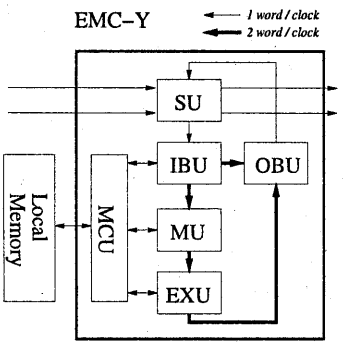
EM-X は RISC パイプラインを持つ高速な逐次スレッド実行機構と、データ駆動モデルに基づくスレッド起動機構を融合し、さらにリモートメモリアクセス機構を持つことにより、効率的な細粒度並列処理を可能としている。現在は 80 台の要素プロセッサ (PE) で構成されるプロトタイプが稼働しており、その性能評価が進められている[1),5)~7)]。

主メモリは高速 SRAM で構成され、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。これによって、RISC 命令実行のほか、高速な同期やスレッド起動を実現している。

一般に、計算機の性能を向上させる重要な要素として、プロセッサ動作クロックの高速化とメモリ容量の拡大が挙げられるが、EM-X アーキテクチャにおいても、これらの観点から今後性能向上を図るためにはキャッシュメモリの採用は不可欠である。細粒度処理の要である通信と演算のインタフェース部分において、このキャッシュがこれまでの SRAM アクセスと同程度の効率を提供できるかどうかは全体の性能に大きく影響する。

3.1 要素プロセッサ EMC-Y

EM-X の要素プロセッサ 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 =



SU : Switching Unit
IBU : Input Buffer Unit
MU : Matching Unit
EXU : EXecution Unit
OBU : Output Buffer Unit
MCU : Memory Control Unit

図1 EMC-Y の構成

データ 32 bit+ タグ 6 bit) 実装されている。

EM-X は実記憶システムであり、各 PE のローカルメモリには、パケットバッファ (MIB: Memory Input packet Buffer)、システム領域、ユーザコード領域、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作業領域が配置されている。

3.2 細粒度並列処理支援機構

以下に、EMC-Y の中のメモリアクセスに関係するユニットについて述べる。

IBU は入力パケットのバッファリングを担う。8 パケット分のオンチップ FIFO を 2 系統持つが、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。このほか、リモートメモリアクセスパケットの処理と 2 入力待ち合わせ処理の前半を受け持つ。2 入力待ち合わせパケットが到着すると、アドレス部で指定されたマッチングメモリ (MM : 1 word) がアクセスされる。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。

MU は待ち合わせ処理後半とスレッド起動処理を担う。必要に応じてマッチング後半処理を行い、関数コードのベースアドレスとなる TTOP(Template TOP) ワードを読み出した後、最後のステージで 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。

EXU はスレッドの逐次実行ユニットである。MU によりフェッチされた命令を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、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。

MCU は IBU, MU, EXU からのメモリリクエストの調停を行う。それぞれのユニットのメモリアクセスの種類を表 1に示す。本稿で議論する同期機構およびスレッド起動機構に関するアクセスは下線太字で表してある。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のデータ幅はピン数の制限から 38bit 幅

phase	r/w	IBU	MU	EXU
1st	read	—	**Matching-read** **TTOP-read** I-fetch	I-fetch
2nd	read	Remote-read **Matching-read** Packet-restore	—	Load
	write	Remote-write **Matching-write** Packet-save	**Matching-clear**	Store

表1 EMC-Y のメモリアクセスの種類

パケット	状態遷移	クロック占有時間
1 入力 :	IDLE →TT →EX	1
1 入力特殊 :	IDLE →EX	0
2 入力 :	IDLE →MM →TT →EX	2
2 入力特殊 :	IDLE →MM →EX	1

TT: TTOP read
MM: Matching Memory read/write
EX: First Instruction Fetch and Invocation of EXU

表2 パケットによるスレッド起動 (MU の状態遷移)

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、実質的に 2 ポート (1st/2nd phase) 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。ただし、同一 phase 内のアクセスは競合を起こすため、オーバヘッドとなることがある。原則として、命令フェッチや命令実行などの EXU に関するアクセスの優先度が最も高く、次に MU、最後に IBU という優先順位となっている。

3.3 待ち合わせ処理とスレッド起動

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。スレッドの起動はパケットの到着により行われ、前のスレッドの実行が終了次第、スレッド起動条件の判定と命令フェッチが行われる。パケットによるスレッド起動には表 2の 4 種類がある。表は、パケットの種類に対する MU の状態遷移と、アクセス競合が起きた場合のオーバヘッド (クロック数) を示す。

2 入力の場合、スレッド起動条件の判定は IBU で行われ (Matching-read/write)、MU がそれを受け (Matching-read/clear)、TTOP 読み出し (TTOP-read) とスレッド先頭の命令フェッチ (I-fetch) を行う。1 入力の場合は Matching 処理は省略される。MU における MM および TTOP のアクセス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ないため、競合が生じた場合はスレッドの終了まで待たされる。特殊パケットの場合は TTOP 読み出しを伴わず、システム予約のスレッド先頭アドレスが直接決定される。

このように、入力パケットはパイプライン処理されており、その各段でメモリアクセスを必要としている。それらを EXU 等のアクセスとオーバラップさせることができれば性能を向上させることができる一方、それらのメモリアクセスのレイテンシが悪化すると、同期およびスレッド起動処理の性能低下を招き、並列処理性能を低下させる原因となる。

なお、現在 EM-X 用の 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入力だけが用いられる。

3.4 メモリアクセスの分類

表1に示したメモリアクセスの種類とその性質から、処理の同時進行(オーバラップ)の可否、論理的なメモリ空間の違い等を考慮して、以下のようなメモリアクセスの分類を行う。分類したグループ同士は原則としてオーバラップ可、グループ内では不可とする。本稿では、このうち同期処理とスレッド起動処理に直接関係のある MM と TTOP について焦点をあてて評価を行う。

- MIB : IBU Packet restore/save
- MM : IBU Matching-read/write, MU Matching-read/clear
- TTOP : MU TTOP-read
- DATA : EXU Load/Store, IBU Remote read/write
- I-fetch : EXU I-fetch, MU I-fetch

4. 性能評価

本章では、EM-X のアーキテクチャを対象として、細粒度同期およびスレッド起動処理にともなうメモリアクセスの性能と、全体の並列処理性能の関係を調べ、評価する。同時に、同期処理用のキャッシュを設けた場合のアクセス競合の低減による性能向上率も測定し、分離キャッシュの効果を評価する。

評価には、EM-X の動作をクロックレベルで再現できるアーキテクチャシミュレータを用い、4つのベンチマークプログラムの実行時間を測定した。

本稿執筆時点で、シミュレータの機能としてキャッシュ機構の組み込みは完了しておらず、分離キャッシュを想定したローカルメモリのマルチポート化と、ポートごとのミス率とミスペナルティをパラメータとして与えることによる各ユニットのパイプラインの間欠動作化が可能となっている。これにより、評価方法を1) 競合低減による性能向上率の測定、2) キャッシュミスによる実行時間の変化の測定の2点に絞った。

4.1 アクセス競合の影響

実際の EMC-Y の機能に加え、TTOP と MM のアクセスをそれぞれ他のアクセスと独立に行えるように設定し、EMC-Y オリジナルの性能と比較する。ローカルメモリのマルチポート化、あるいは次節に述べる任意のミス率を与えることのできる分離キャッシュとみなすことができる。

4.2 キャッシュミスの影響

ミスを起こす平均間隔とペナルティ時間に従って、ミス期間中の各支援機構のメモリアクセス要求に対する応答を止めることにより、キャッシュミスによる性能低下と同じ効果を得ることができる。なお、ミスを起こしたポートに関係のないユニットは実行を続けることができるノンブロッキング動作としている。

議論を単純にするため、TTOP と MM 以外のメモリアクセスは WAIT なしで実行できるものとする。

4.3 ベンチマークプログラム

細粒度処理のメモリアクセスについて調べるため、次の3種類、4個のベンチマークプログラムを使用した。

fib フィボナッチ数を求めるプログラム。アセンブリ言語で書かれており、2入力待ち合わせによるスレッドの直接起動機構を利用している。
パラメータ: 80PE, $fib(27)$
平均粒度: 4.3 clock

subst (thread invoked) LU 分解法の後半に現れる三角方程式の求解問題。1要素ごとの細粒度並列計算を行う[6]。1入力パケットで1要素の計算ノードのスレッドを直接起動させており、シストリックアレイ的な動作をする。すなわち、プロセッサ間で規則的な演算と通信が同じ頻度で繰り返され、よどみなく並列計算が進む。EM-C を用いて記述し、要素計算部分はアセンブリ言語レベルで最適化した。
パラメータ: 32PE, n=2000, single thread
平均粒度: 11.0 clock

subst (I-structure, multithread) このアルゴリズムは thread invoked 版と同じであるが、1要素ごとの細粒度同期を、2入力待ち合わせを用いた I-structure によって取っている。I-structure のレイテンシを埋めるためにマルチスレッド動作させている。EM-C のみで記述した。
パラメータ: 32PE, n=2000, 3 threads
平均粒度: 13.0 clock

mp3d SPLASH ベンチマークに含まれる3次元粒子シミュレータ。共有メモリプログラミングモデルにより記述され、EM-X では各 PE に分散されたグローバルメモリ空間を細粒度パケットでアクセスする[4]。マルチスレッドによりリモートアクセスレイテンシ隠蔽を図っている。リモートリードパケットは、リモート PE の IBU により直接サービスされ、元のスレッドを再起動するために1入力の通常パケットとしてデータが返る。EM-C のみによる記述である。
パラメータ: 16PE, 3000 particles, 2 threads
平均粒度: 17.8 clock

4.4 測定結果

各ベンチマークプログラムの実行結果を図2～5に示す。各グラフは、0～20% までのミス率に対する実行時間の関係を示している。また、それぞれのミス率について、ポートを統合 (Unified) した場合と分離 (Separated) の場合を並べて示してある。実行時間は、オリジナルの EMC-Y(Unified, 0%) を用いた場合の時間を1として正規化してある。なお図5だけ時間軸のスケールが異なることに注意されたい。

分離条件は、TTOP と MM をそれぞれ独立させ、他のアクセスとの干渉をなくしたものである。比較を容易にするために、統合および分離ともに、キャッシュミスは TTOP および MM だけで起き、それぞれに同じ条件が与えられるが、ミスはそれぞれ独立して起こる。キャッシュミスはミス率に応じて定期的に発生し、ミスペナルティは16クロックとした。

fib : 結果を図2に示す。本プログラムの特徴は、粒度が非常に細かいことと、2入力の待ち合わせでスレッドを直接起動することである。パケットの内訳は、2入力パケットが約73%、1入力パケットは約27% である。このため、待ち合わせおよびスレッド起動の際のメモリアクセス競合の頻度が多い。TTOP, MM を分離することにより、実行時間が約7% 短縮された。また、ミス率が性能に

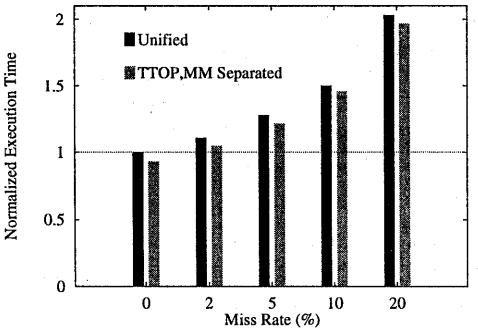


図2　ミス率に対する性能：fib(27)

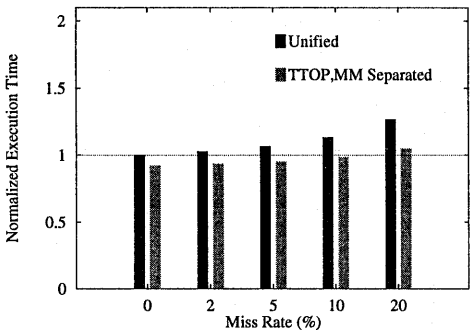


図3　ミス率に対する性能：subst(thread invoked)

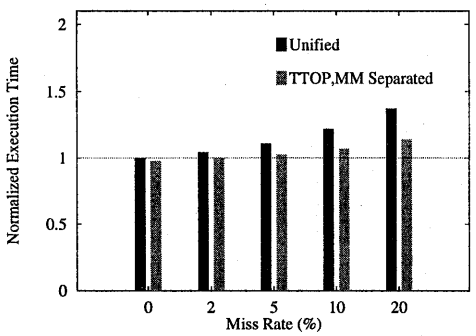


図4　ミス率に対する性能：subst(I-structure, multithread)

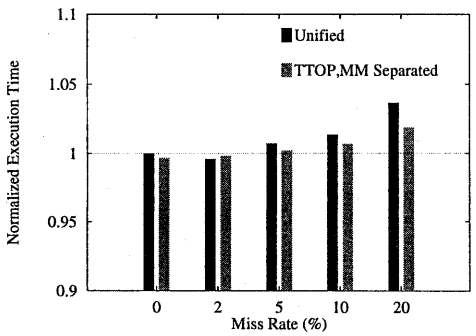


図5　ミス率に対する性能：mp3d

及ぼす影響は大きく、20% のミス率で実行時間が約 2 倍に増加した。

subst(thread invoked)：　結果を図 3 に示す。本プログラムでは、1 要素の計算の終了に対して、十分先行して次スレッドを起動するパケットが到着しているが、Unified ではスレッド起動処理のオーバラップができないため、1 clock のオーバヘッドがある。スレッド長は平均 11 クロックであるため、オーバヘッドは約 9.1% と見積もることができ、測定結果と一致する。ミス率が大きくなると分離の効果が大きくなり、ミス率 20% 時には約 17% の時間短縮となった。これは、平均のスレッド長が比較的長いため、TTOP アクセスが命令実行とオーバラップする結果、ミスが起きてもペナルティ時間は命令実行に隠されるためである。

subst(I-structure, multi thread)：　結果を図 4 に示す。I-structure そのものには 2 入力特殊パケットが用いられる。また、I-structure の呼び出し部分で 1 入力パケットが用いられる。シミュレータによる詳細な観察によると、オーバヘッドの発生状況はやや複雑で、アクセス競合にともなうオーバヘッドのほか、微妙なタイミングのずれによるパケット到着の遅れ等がスレッド起動を遅らせる原因となることがある。測定の結果、ポート分離により約 2.3% の時間短縮が得られたが、前述のように、アクセス競合低減では削減できないオーバヘッド要因が多くを占めている。ミス率が大きくなると分離の効果が大きくなるのは thread invoked と同じ理由による。

mp3d：　図 5は、mp3d ではミス率が小さい場合には、ポート分離の影響は非常に小さいことを示している。ミス率が 2% の場合には、わずかであるが Unified のミス率が性能向上を導いているように見える。これは、ミス率の増加が命令実行を抑制し、ネットワークの混雑を減らす効果である可能性があるが、詳しい理由は現在解析中である。また、別途得たシミュレータの情報では、3% 程度のポート分離利得 (すなわちアクセス競合) の可能性が示されていたが、ミス率 0% 時には 0.3%、ミス率 2% 時には負の値となっている。これも、前述のネットワーク効果である可能性がある。

4.5 考　察

分離キャッシュの狙いとして、より進んだ最適化によるミス率低減があるが、冒頭で述べたようにミス率悪化要因の側面もある。分離後もミス率が同じとすれば、アクセス競合低減効果により性能向上が期待できる。分離後、ミス率が低下すれば、相乗効果によりさらに性能向上が期待できる。分離後、ミス率が悪化した場合、総合的に性能低下となるかどうかの判断には、実際のキャッシュ機構を組み込んだシミュレーションを行う必要がある。

fib では、ポート分離の効果およびミス率の影響ともに大きいが、現実的なプログラムではない。実用プログラムにおいて、分離キャッシュ方式が fib と同程度に有効となるには、fib と同程度の粒度を得るコンパイラの研究が重要となってくる。

subst では、規則的な通信・演算パタンを繰り返す thread invoked 版が、設定条件に対し比較的大きな影響を受けることを示したが、細粒度の同期機構を活かす点では、I-structure 版の方が重要である。前者は、通信の順序が乱れれば正しい計算結果が得られなくなってしまう

が、後者は順序に影響されず、より不規則な演算パタンでも対応でき、同様の手法を適用できる用途が広いためである。

mp3d では、ネットワークの効果の可能性が残っているとはいえ、予測された性能向上も高くなかった。性能が同期処理部に左右されにくいのは、このプログラムではネットワークが性能におよぼす割合が支配的であるためと思われる。

このように予備評価結果は、提案した方法の有効性が、プログラムの種類、特にその中で使われている演算・通信・同期のパタン、頻度に大きく左右されるものであることを示している。今後、細粒度向けキラーアプリケーションの研究とともに、より普遍的な性能改善の手法を探る必要がある。

4.6 同期機構向きキャッシュ方式

検討中の分離キャッシュ方式は、分離して実質的サイズが小さくなることによる性能低下要因と、アクセス要因のパタンに対して最適化することによる性能向上要因がある。チップ製造条件およびコスト面から、ハードウェア増加要因は小さい方が望ましい。小さいサイズで十分な性能を得られれば、メリットは大きい。

現在検討中のキャッシュ性能向上技法につながる性質として、次のようなものが挙げられる。シミュレータに対しこれらを実装中である。

- TTOP, MM ともに、マルチスレッド実行時のワーキングセットは比較的小さいと考えられる。MM は subst(I-structure) で示したような使用方法もあるため常に小さいとは限らないが、この例ではペアとなるデータの発生時間は比較的短いため、サイズが小さくてもその局所性を有効利用できると考えられる。
- MM では、2 入力待ち合わせ成功後はそのエントリは解放できる。
- TTOP は各 OSEG に存在する必要はなく、主メモリ上に連続領域として確保してあってもよい。バースト転送機能のある下位階層メモリを前提とすればそのほうが良い。
- 小さなキャッシュであれば、連想度の増加、ポート数の増加等が実現できる可能性がある。

5. ま と め

分散メモリ型並列計算機の要素プロセッサが行う同期処理のためのメモリアクセスに注目し、そのオーバヘッド削減の手法の効果と、キャッシュ適用時のミス率増加の性能に対する影響を調べた。非常に細粒度な並列プログラムでは提案する分離キャッシュ方式の効果が認められたが、別のプログラムではほとんど効果がみられないものもあった。これは、通信・同期の種類や粒度によって同期処理部の重要性が異なってくることと、ネットワークの混雑度の変化の影響が原因と思われる。今後、より普遍的な性能改善の手法を探る必要がある。

また、本稿で述べた TTOP, MM のほか、IBU の MIB アクセス、Remote サービス等のバッファ機構によるメモリレイテンシ隠蔽方式を検討中であり、多方面から性能向上の可能性を探っていく予定である。報告は別の機会に譲るが、通信・同期処理部のパイプライン方式、命令実行部の改良も課題として重要である。本文中に述べたキャッシュ汚染削減効果や最適化の検討については、その検証と実装が進行中である。

現在、種々のアーキテクチャ実証実験ができるよう、FPGA による EMC-Y の実装も進めている[8]。FPGA ベースでも、本稿で示したような小規模のキャッシュ / テーブル機構が実装可能となると考えられ、今後 FPGA を用いた検証も行う予定である。

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諸氏に感謝いたします。

参 考 文 献

1) 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.

2) 松岡, 岡本, 廣野, 佐藤, 横田, 坂井: 超並列要素プロセッサ RICA-1 の高速メッセージハンドリング機構, JSPP'98, pp.79-86, 1998.

3) 坂根, 本多, 弓場, 児玉, 山口: 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, 情報処理学会研究報告, ARC-130-17, pp.97-102, 1998.

4) 佐藤, 児玉, 坂井, 山口: 並列計算機 EM-4 の細粒度通信による共有メモリの実現とマルチスレッドによるレーテンシ隠蔽, 情報処理学会論文誌, Vol.36, No.7, pp.1669-1679, 1995

5) 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.

6) 坂根, 児玉, 建部, 小池, 山名, 山口, 弓場: 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, 情報処理学会論文誌, Vol.30, No.5, pp.2281-2292, 1999.

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) 佐谷野, 児玉, 坂根, 山口: 並列計算機ノードプロセッサの FPGA を用いた実装と評価, 1999 年 8 月 SWoPP'99 (情報処理学会研究報告:ARC) で発表予定