計算機アーキテクチャ 137－1
ハイパフォーマンス
コンピューティング 80－1
（2000．3．2）

並列計算機 EM-X の細粒度通信機構を用いた共有メモリベンチマークの実行

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 EM-X において、その高い通信性能を利用した分散共有メモリの性能をSPLASH2 ベンチマークを用いて評価した結果を報告する。EM-X ではワード単位の細粒度なリモートメモリアクセス機構とグローバルポインタを用いて共有メモリプログラミングが可能である。まず、共有メモリアクセスに伴う細粒度通信のオーバヘッドを、逐次プログラムとの比較により示す。次にプロセッサ台数を増やした場合の並列処理の効果を評価する。プログラムの性質により台数効果は大きく異なり、ローカル演算が比較的多い BARNES では高い性能を示す一方、LU や FFT などの共有メモリアクセス頻度の高いプログラムでは細粒度通信オーバヘッドが性能向上を制限した。マルチスレッド実行では、通信レイテンシ隠蔽効果により、性能を最大 43% 向上させることができた。

Executing Shared Memory Benchmarks on the EM-X using Fine-grain Communication Mechanism

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

In this paper, we discuss performance of distributed shared memory on the EM-X with fine-grain communication by showing experimental results of benchmark programs. While the EM-X is a physically distributed memory machine, it can handle fine-grain remote memory access efficiently so that enhancement for shared memory programming is available by providing a global pointer facility. We chose three programs from the SPLASH2 benchmark suite to clarify the characteristics of shared memory execution of the EM-X. At first, we changed the programs to fit the EM-X manner by making it possible to investigate the basic performance with the least effort. We also made some optimized versions of the programs, including multithreaded versions for hiding communication latencies. Communication overheads of the parallel programs are discussed by comparing with sequential executions. BARNES shows very good speedups because of its many local executions compared with communication. Programs, such as LU and FFT, which require high shared memory access rate tend to limit their performance due to overheads caused by per word communication. Multithreaded execution, for the sake of latency hiding, brings gains of up to 43%.

1. はじめに

一般に、分散メモリ型並列計算機のプロセッサ間通信性能はローカルメモリアクセス性能に比べて低く、その上で共有変数を実現しようとすると通信オーバヘッドが問題となる。そこでコヒーレントキャッシュを用いてデータアクセスの局所性を高め、プロセッサ間通信を必要最小限に抑えることで性能を高める試みが数多くおこなわれている。

一方、EM-X[1] は分散メモリ型でありながら細粒度通信機構と高スループットネットワークを持つことにより、高い通信性能を持つ。EM-X の要素プロセッサ (PE) 内では演算ユニットと通信機構が密に融合されており、短い固定長パケットを用いることによって細粒度のプロセッサ間通信を低オーバヘッドで行うことを可能としている。EM-X はコヒーレントキャッシュを持たないため、リモートのデータは必ず通信を起動してアクセスされる。従って、共有メモリを実現しようとするとそのアクセスはワード単位の粒度で通信を引き起こし、細粒度通信のコストがプログラムの実行性能に大きな影響を与える。本稿の目的は、細粒度通信機構が共有メモリプログラムの実行にどの程度有効であるかを明らかにし、性能に影響を与えるオーバヘッドの削減の指針を得ることである。

本稿では、共有メモリベンチマーク SPLASH2 から性質の異なるプログラムを 3 つ選び、EM-X 上に実装したプログラムを用いて次のように評価を進める。まず、逐次プログラムと、1PE での並列プログラムの実行を比較することにより、局所的な並列化オーバヘッドを明らかにする。ローカルアクセス最適化とマルチスレッド化による性能向上を調べることにより、オーバヘッドの性質を論じる。次に、PE 台数を増加させて各プログラムの性能向上率を測定することにより、共有メモリプログラムが持つ固有のアクセスパタンが性能に与える影響を調べる。これらオーバヘッドの解析とプログラムごとの台数効果から、EM-X で共有メモリプログラムを実行する際の有効性と問題点について議論する。最後に EM-X の細粒度通信機構の得失についてまとめ、今後の課題について述べる。

† 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

PE台数	80
クロック周波数	20MHz☆
メモリ	1Mword/PE, 38bit/word
命令実行サイクル	1 CPI
浮動小数点演算精度	32bit(single)
ローカルメモリレイテンシ	1 clock
ネットワークスループット	2 clock/packet
共有ライト発行サイクル	1 clock
共有リード発行サイクル	2 clock
自PE内共有リードレイテンシ	9 clock
平均共有リードレイテンシ	20 clock

表1 EM-X の諸元と通信パラメータ

2. EM-X システム

分散メモリ型並列計算機 EM-X は専用に設計された PE (EMC-Y) を持つ。PE 内部で演算部と通信部を融合し、データ駆動モデルに基づいた命令スレッドの起動や効率的な細粒度通信を実現している。

EM-X はハードウェアの観点からは分散メモリであるが、コンパイラがグローバルポインタによる共有メモリを提供し、そのアクセスを通信命令に変換することにより、ソフトウェアからは ncc-NUMA として見ることができる。共有メモリアクセスに伴う通信には、パケットによる直接リモートメモリアクセス機構が用いられ、レイテンシが低く抑えられている。さらに、EM-X では通信レイテンシを隠蔽するための効率のよいマルチスレッド実行が可能であり、細粒度通信を発生する共有メモリモデルの実行を支援する。

本稿の評価に関係のある EM-X の主要な諸元と通信パラメータを表1に示す。

2.1 リモートメモリアクセス[2)]

EM-X の通信パケットはアドレス部とデータ部の2ワードの固定長であり、アドレス部はシステム全体の任意のメモリを一意に指定できるグローバルなアドレスを保持できるほか、パケットの機能を表すタグを持つ。データ部には通常のデータのほか戻り先アドレス (continuation) を納めることができ、リモートメモリアクセスの単純化と効率化に寄与している。リモートメモリアクセスパケットが到着した PE では、実行中の命令を中断することなく、通信処理ハードウェアが直接メモリを参照する。

リモート書き込み (SYSWR パケット) の発行に要する時間は1クロックで、リクエストを出した PE はすぐに次の処理を進めることができる。

リモート読み出し (SYSRD パケット) の発行には continuation 生成を含めて2クロックかかる。パケットが到着した目的 PE において、パケットのデータ部に埋め込まれている continuation を返送パケットのアドレス部に移してデータを返送する。リクエストを出した PE は、通信レイテンシの時間が経過するまでそのデータを使うことができない。その間ほかの処理をオーバラップさせなければ、通信レイテンシはオーバヘッドとなる。

2.2 EM-C[3)]

EM-C は、並列構文やグローバルポインタなどを備えた EM-X の並列プログラミング言語である。EM-C コンパイラは、グローバルポインタを用いたアクセスをリモートアクセス用パケット出力命令に変換して、オブジェクトコードに埋め込む。また、後述のマルチスレッドをサポートする。

☆ 現在は 16MHz で稼働中であるが、本稿では設計時の 20MHz に換算して性能を求める。

2.3 共有メモリサポート[3),4)]

EM-C ではグローバルポインタを用いることにより、共有メモリプログラミングが可能である。グローバルポインタは実際には PE-ID とローカルアドレスの組み合わせであり、目的アドレスとして直接パケット送出命令に渡される。グローバルポインタによるアクセスがあるとコンパイラはパケット出力命令を静的に生成し、それ以外は通常のロード・ストア命令を生成する。

しかしながら、MMU を持たない EM-X では、共有アクセスの対象がリモート PE か自 PE かの効率の良い動的チェックは困難である。従って共有メモリが自 PE にある場合 (この PE をオーナという) でも、グローバルポインタを用いる限りは自 PE 内で通信が発生する。このため、一般的な分散共有メモリ向けに用いられるようなオーナアクセスによる最適化だけでは性能向上に直接はつながらない。

2.4 マルチスレッド

EM-X ではリモート読み出しレイテンシ隠蔽のため、PE 内で複数のスレッドを切替えながら実行するマルチスレッド実行[4),5)] をサポートしている。その動作に関連するオーバヘッドは性能の解析に重要であるためやや詳しく説明する。

EMC-Y のレジスタセットは1組であるため、スレッドが切替わるときにはコンテキストを保持するレジスタを待避する必要がある。このスレッド切替え点は命令列の上で静的に指定される。コンパイラがスレッド切替え点とセーブ・リストアが必要なレジスタを決定し、ストア命令とロード命令を埋め込む。このストアとロードの実行時間はレジスタ数が増えると大きなオーバヘッドとなる。スレッド切替えそのもののコストは1クロックだけであるため、一般にレジスタ待避の方が問題となる。スレッドの切替えは、一般に関数呼び出しとリモート読み出しのときに起きる。EM-C コンパイラはマルチスレッド実行を前提としているため、スレッドを1本しか用意しない場合でも、これらが発生するときには必ずスレッドを中断するコードを生成する。

スレッド切替え要因のうちでは、一般にリモート読み出し (SYSRD) の頻度が高く重要である。待避されるレジスタの数が n とすると、SYSRD パケットの生成も含めて $2n+3$ クロックだけパイプラインを占有する。通常のロード命令の実行は1クロックであり、それに対して $2n+2$ クロックのオーバヘッドが生じることになる。

このほか、スレッドの生成と同期もオーバヘッドの原因となるが、スレッド数を適正にすることと生成・同期の頻度を必要最小限にするよう注意すれば、このオーバヘッドは小さく抑えられる。レイテンシを隠蔽するのに必要なスレッド数は、スレッド長とリモートアクセスレイテンシに依存するが、通常は2～4個で十分である。

3. 性能評価

本章では EM-X 上での SPLASH2 プログラムの実装方法と、実行性能について述べる。

3.1 実装指針

共有メモリプログラムをなるべく手を加えずに分散メモリ型並列計算機で動かした場合の性能評価に重点をおくため、プログラムの実装に際して一定の基準を定めた。

3.1.1 逐次実装 (SEQ)

SPLASH2 では、逐次プログラムを得るために並列化記述を隠す Null Macro が用意されている。それをオリジナルプログラムに適用した後、浮動小数点演算精度を単精度とするなど EM-X 向けの変更を施したものを、逐次バージョン

(SEQ) とする。

3.1.2 並列プログラムの基本実装 (ST)

SPLASH2 のプログラムは、種々のアーキテクチャへ実装することを考慮して、同期や共有メモリ割り当てなどの並列プリミティブが PARMACS というマクロで記述されている。また、物理的な分散メモリを持つマシンに実装しやすいよう、共有空間のマッピングを実装者に任せている。各オリジナルソースプログラムには計算の性質に基づきマッピングの簡単なガイドラインが示してあり、EM-X での実装もそれに従った。

リーダとライタの最適割当てなど、プログラムの性質に応じた最大限の最適化については本稿の主題とは異なるため、今回の評価では必要以上の最適化は行わない。コンパイラにおける指示文の使用を念頭に置いて、実装に関して次のような指針を定めた。

- EM-C がもつ並列プログラミングサポートとして、マルチスレッド機能とグローバルポインタを用いる。同期プリミティブ等は PARMACS マクロに埋め込む。
- 通信主体 (リクエスト発行者) の変更や解法アルゴリズムの変更などの大きな変更はおこなわない。
- G_MALLOC マクロで確保される領域は原則として全て共有メモリとして扱う。すなわちその領域のアクセスはすべて通信を伴う。
- ただしローカル作業用配列を G_MALLOC で確保している部分はローカルメモリとして確保する。
- プログラムの初期化から終了時にわたって不変な少数のグローバル定数 (基本パラメータ等) を保持する変数は、初期化時に各 PE へ値を放送し、その後は各 PE でローカルアクセスを行う。
- 共有メモリ上の配列要素のグローバルポインタは、PE 間をまたぐポインタ変更があるごとに配列インデクスから求める。同一 PE 内であることが明らかな場合は通常のポインタ計算を適用する。この PE 境界の判断は手作業で行ったが、データフロー解析により自動的に検出する研究もある[6),7)]。
- 浮動小数点演算精度はハードウェアを利用するため単精度とする。

これらの指針に従うプログラムのバージョンを ST(straightforward) と表記する。

3.1.3 ローカルアクセス最適化 (LO)

EM-X における ST の実行では共有メモリのローカル / リモートの区別が行われないため、オーナによるアクセスも通信を発生してしまう。通信がブロック単位で規則的に行われるプログラムでは、ループの外側でチェックすることによりチェックのオーバヘッドを低く抑えることができ、ローカルアクセスに対する最適化が可能である。この最適化を行うバージョンを LO(local optimized) と表記する。

3.1.4 マルチスレッドによるレイテンシ隠蔽 (MT)

ST にマルチスレッド化を施したバージョンを MT(multithread) と表記する。マルチスレッド化手法として、ループイテレーションをインターリーブする方法を用いる。すなわち、隣り合うイテレーションを別のスレッドに巡回的に割り当てる。その際、スレッド生成・同期オーバヘッドを抑えるために、スレッドの片寄りが起きない程度になるべく外側のループに適用する。マルチスレッド化可能な部分は、PE 内スレッド同士で冗長な処理をしないよう大局的に手作業で判断する。また、必要に応じて作業用変数をスレッドごとに局所化する。

このほか、LO と MT の相乗効果を見るために、これらを組み合わせたバージョンも作成する。

3.2 ベンチマークプログラム

SPLASH2 の中から次の 3 つのプログラムを選んだ。

- LU(contig) : ブロック化した密行列 LU 分解の計算を行う。ブロック内データはメモリ上で連続配置される。データサイズは 512×512 とする。
- FFT : 1 次元 FFT を、局所計算と行列転置を交互に計 6 ステップ行うことで計算する。データサイズは 65536 点とする。
- BARNES : 粒子の相互作用の計算を Barnes-Hut の Tree アルゴリズムで行う。データサイズは 2048 粒子とする。

3.2.1 LU

LU ではブロックサイクリック状に PE を使用するため、データ分散もそれに従い ST を実装した。プログラムを注意して見ると、ブロックの更新はそのブロックのオーナだけが行うことがわかる。さらに参照アクセスも含め全アクセスをオーナが行う計算ステップもある。これらにより静的なローカルアクセス最適化が可能である。静的最適化に当てはまらないブロックでも、ブロックの外でオーナのチェックを行う動的な最適化が可能である。これらの性質を利用して LO では静的最適化と動的最適化を併用した。その結果ローカルメモリ上の共有アクセスは全てロード・ストア命令で行われる。MT については、互いに依存性のない最内周の内積ループ単位でスレッドに割り当てた。スレッドの生成はブロック処理単位で行い、同期は内積ループ単位で行った。

3.2.2 FFT

FFT では 1 次元データを 2 次元配列として扱う。行ブロック単位で PE 内計算を行うため、データ分散も行ブロックとして ST を実装した。ブロック割り当ては非巡回である。3 つの計算ステップにおいて、部分 FFT 計算とそこで発生するデータアクセスは全てブロック単位でオーナにより行われる。通信は残りの 3 ステップの行列転置で発生する。その通信は規則的で、更新はオーナが行い参照はオーナ以外が行う。LO ではこれらの規則性を利用し、全てのオーナアクセスを静的に最適化した。MT については、計算ステップではブロック内の行単位でスレッドを割り当て、転置ステップでは転送単位である正方小ブロック内の列単位で割り当てた。

3.2.3 BARNES

BARNES は、複数パラメータを持つデータ要素を主に 8 進木構造で管理するため、複雑なデータ構造を持つ。主要データ構造のうち、leaf と cell データは各 PE に割り付けられる。body(=particle: 粒子データ本体) は計算主体の定まらない共有データであるが、ST の実装をする際は leaf や cell と同様に初期化時に各 PE に均等に割り付けた。計算は主に木の生成、相互作用の計算、座標計算等のステップで進み、その中で相互作用の計算が最も計算量が多い。各データ構造の計算主体はオーナ以外に広く分散するため、通信パタンは不規則である。LO については、静的最適化が困難で動的チェックのオーバヘッドも大きいことから適用を見送った。MT では相互作用計算のステップにマルチスレッドを適用した。相互作用計算を開始する時に複数スレッドを生成し、粒子インデックスでスレッドを振り分けた。

3.3 測定結果

各プログラムの実行時間を図 1～3 に、台数効果を図 4～6に示す。台数効果は SEQ の実行時間を 1 としている。明記

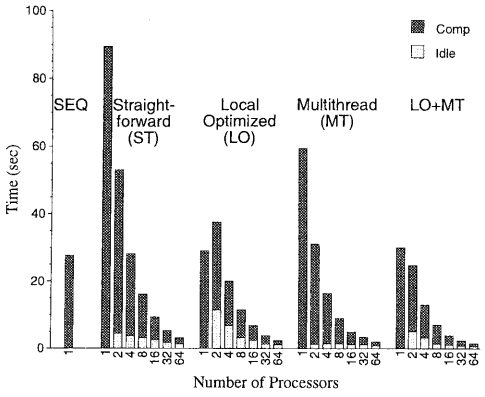

図1 実行時間 (LU : 512 × 512 matrix)

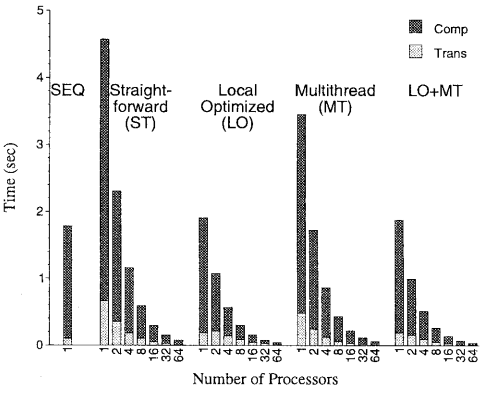

図2 実行時間 (FFT : 65536 points)

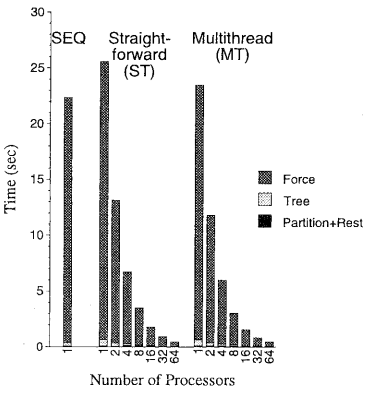

図3 実行時間 (BARNES : 2048 particles)

していない実行パラメータは SPLASH2 のデフォルト値を使用した。台数効果のグラフで PRAM と表記されているのは文献 8) から引用した PRAM モデルにおける理論性能であり、その問題固有の性能上限を表す。

LU の実行時間において、Comp は計算時間である。Idle はバリア同期による待ち時間であり、各 PE がバリア待ちに入ってから残りの全 PE がバリアに到達するまでの時間の PE ごとの累計の平均値である。PE 台数が 2 台以上の場合はバリアにおける待ち時間が性能向上を制限しているが、これは行列のブロック化にともなう負荷のアンバランスに起因し、オリジナルのアルゴリズムが本質的に持つ性質である。SEQ に対し 1PE の ST は実行時間が 3 倍以上になっているのは共有アクセスに伴うオーバヘッドが原因である。オーバヘッドに関しては次節でまとめて議論する。ST に対する最適化の効果は、1PE の場合 MT の性能向上率が 34% であるのに対し LO は 68% と高い。2PE 以上ではこれが逆転し、MT の効果が高い。これは実際にリモートアクセスが必要な割合が増加するためである。2PE 以上の MT の性能向上率は 38 ～ 43% であり、台数の影響は少ない。PRAM に対する性能は、いずれの最適化バージョンも台数が増えるに従い相対的に低下し、LO+MT の 64 台の場合で PRAM の約半分 (SEQ の約 18 倍) になっている。

FFT の実行時間において、Comp は計算時間、Trans は行列転置の時間である。台数が少ない場合の傾向は LU に似ている。特徴は台数が多い場合に現れており、LU とは異なり MT よりも LO の効果が大きい。これは部分 FFT の計算ステップが全てローカルアクセスであるためである。MT の寄与は行列転置ステップにしか現れないため、性能向上効果は 24 ～ 27% にとどまった。負荷の片寄りは小さいため、PRAM の上限は高い。LO および LO+MT が PRAM に接近している理由は、FFT はオーナによるローカル計算の割合が多く、EM-X 固有の通信オーバヘッドの影響が小さいためである。

BARNES の実行時間において、Force は相互作用の計算時間、Tree は 8 進木の生成時間、Partition+Rest は粒子ポインタの分散と座標計算である。実行時間のほとんどを Force が占めている。文献 8) の PRAM 性能は 65536 のサイズのものであるため図示しなかったが、Linear に近いものとなっている。デフォルトサイズ 65536 は 1PE のメモリに収まらなかったため、2048 とした。データサイズが小さくなると一般には負荷バランスが悪化するにもかかわらず、今回の結果では比較的高い性能が得られ、MT の 64 台時に 46 倍に達した。BARNES は共有アクセスの頻度が少なく通信のオーバヘッドが小さいうえ、比較的細粒度のデータを扱うため負荷の分散が良好であり、高い性能を得やすいプログラムであるといえる。これは PRAM 性能が Linear に近いことからも裏付けられる。MT の効果が小さいのは通信割合が小さいことが原因である。LO は前述のとおり実装しなかったが、SEQ に対する ST の性能低下が小さいことから、実装できてもその効果は小さかったと考えられる。

3.3.1 オーバヘッドに関する考察

逐次プログラム (SEQ) の実行結果と、並列プログラムを 1 PE で実行した結果を比較することにより、並列化による種々のオーバヘッドを調べることができる。これらはシステムの設計に大きく影響を受けるため、要因と性能の関係を詳しく調べることにより新しい設計指針を導くことができる。

EM-X では、逐次プログラムに対する並列化オーバヘッドとして次のようなものが挙げられる。

グローバルポインタの生成と操作: 共有メモリをアクセスするためのグローバルポインタは、オーナ PE の ID とローカルアドレスから生成される。それがスカラー変数の場合はオーナ PE はあらかじめ定めてあるが、配列の場合は、インデックスからオーナを決定する必要があり、手間がかかる。また、ポインタの更新が PE 内で収まる場合は通常のポインタ操作でよいが、PE 間をまた

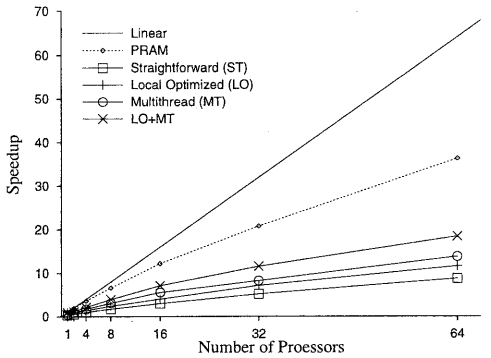

図4 台数効果(LU : 512 × 512 matrix)

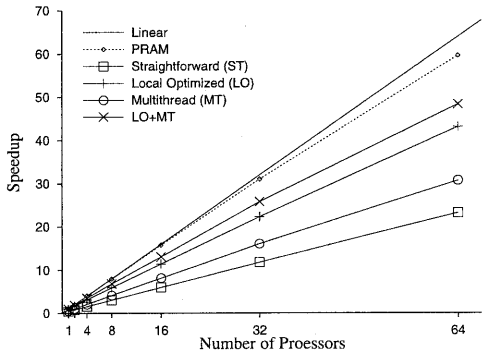

図5 台数効果(FFT : 65536 points)

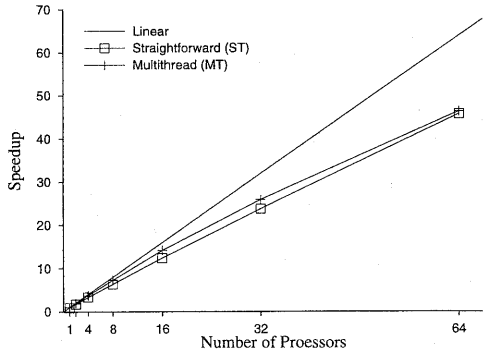

図6 台数効果(BARNES : 2048 particles)

がるような更新では生成しなおす必要がある。

通信レイテンシ: 一般に共有メモリのオーナは別のPEであり、そのアクセスは通信レイテンシをともなう。

通信起動: EM-Xでは、共有アクセスの通信起動時間は書き込み時1クロック、読み出し時2クロックである。

ローカルアクセスのチェック: 共有メモリデータを使用するPEとオーナPEが一致する場合は、通信をおこなわずロード・ストア命令を用いれば共有アクセスのオーバヘッドを削減できる。MMUを備えるシステムでは自動的にこの処理を行うことができるが、EM-Xではグローバルポインタをソフトウェアでチェックする必要があり、チェックコードがオーバヘッドとなる。

リモート読み出しに伴うスレッド中断: スレッドを中断する場合レジスタ待避のための命令実行を必要とし、これが主要なオーバヘッドとなる。使用変数の個数のほかループイテレーションあたりのリモート読み出し回数等にも依存し、およそ1～10クロック程度の範囲である☆。

マルチスレッドに伴うスレッド生成およびスレッド間同期: マルチスレッドを行う場合、ローカルPE上でスレッドの生成を行う必要がある。同期は必要に応じて行う。スレッド生成時間はおよそ20～30クロックであり、局所同期時間は10～15クロックである。

このほか、一般に並列処理では負荷のアンバランスが問題になることが多く、オーバヘッドとして扱われることがある。これは並列プログラムが持つ性質であり、システムの性質は直接影響しない。従って本稿では議論の対象としない。

上で列挙したオーバヘッドについて、LUの1PEの結果に注目して考察する。

- SEQに対するSTの性能低下の原因は、グローバルポインタ生成、スレッド中断、PE内通信レイテンシである。この中でグローバルポインタ生成はループの外で行われ、共有アクセスごとに発生する後二者が支配的である。
- LOでは、STに比べてスレッド中断とレイテンシのほとんどが削減され、ポインタチェックのオーバヘッドが追加される。SEQとの差はグローバルポインタ生成とそのチェックにかかる時間となるが、いずれもループ外であるため小さい。ただし、PE数が2以上になると通信が発生するため、台数増加に相当する割合でLOのSTに対する性能向上率は低下するが、オーナによるアクセス頻度が高ければ低下率は少ない。従って、共有メモリのローカルアクセスオーバヘッドを削減することができれば全体の性能向上に大きく寄与すると考えられる。
- STに対し、MTはPE内レイテンシ削減分の性能向上を得る。削減分のPE内レイテンシは共有リード回数から簡単に算出可能である。LUの主要な共有リード回数は89.2M回であり、共有リードオーバヘッドが7クロックであることから、その累積時間は31.2秒となる。実測ではSTとMTの差は30.3秒であり、レイテンシ削減の効果が支配的であることが確認された。計算値との差はスレッドの生成と同期のオーバヘッドである。
- SEQとMTの差はグローバルポインタ生成とスレッド中断のオーバヘッドであるが、前述の通り後者が支配的となる。実測では31.7秒であり全実行時間の約半分を占める。この割合は台数が増加しても変らないことから、マルチスレッド実行においてスレッド中断オーバヘッドを削減することができれば全体の性能を大きく向上させることができると考えられる。
- 1PEでのLO+MTは通信の起動がほとんどないことから、LOに対するマルチスレッドの利点はない。従ってスレッド生成のオーバヘッド分、LOよりわずかに性能が低下する。ただしPEが複数になるとマルチスレッドによるレイテンシ削減効果が現れる。

4. 関連研究

分散共有メモリの研究の多くはSPLASH2のような共有メモリプログラムを使用している。そのほとんどはページ

☆ 例えばLUの内積ループでは読み出し1回あたり4.5クロック。FFTの転置最内ループでは同8.5クロック。

ベースのコヒーレントキャッシュを用いたDSMシステムについてのものである[7),9),10)]。それらによれば、FFTやBARNESは、ページフォルトの多発や不規則な細粒度共有アクセスによるfalse sharingによって性能向上が得られなかったり、あるいは低い向上率しか得られていない。LUではローカルコピーの再利用性が高いため、共通して比較的高い性能向上が得られており、特にSCASH[9)]では良好な台数効果を得ている。EM-Xではこれらのようなキャッシュに関連する特性はなく、FFTやBARNESで高性能を示すが、LUのように通信量が多くなるとワードごとのオーバヘッドが顕在化する。

SGI Origin2000[11)]はハードウェアによるキャッシュブロック単位のコヒーレントキャッシュにより、LUとBARNESを含む多くのプログラムで高い性能を示している。詳しい解析はおこなわれていないが、高い通信性能とキャッシュの細粒度管理が有効に働いていると考えられる。

Shasta[12)]は、ソフトウェア制御の細粒度コンシステンシ維持機構を組み込んだシステムの研究である。細粒度管理の特性によりBARNESでも比較的高い台数効果が得られているが、OriginやEM-Xには及ばず、全体としてはページベースシステムと同様な傾向を示している。WSクラスタにおける実装としては高く評価できる。

山本ら[13)]は、明示的な通信コードの挿入によりSPLASH2の各プログラムをメッセージ通信用に変更している。本論文では行わなかった通信主体の変更によって、多くのプログラムにおいてキャッシュを用いたDSMよりも高い性能を得ているが、不規則アクセスを生じるBARNESでは成功していない。

Mowry[14)]らはSGI Originをモデルとした共有メモリモデルでのソフトウェア制御マルチスレッドの性能を報告している。キャッシュミスの際にソフトウェアでスレッドを切替えることによりレイテンシを隠蔽している。EM-Xと同様にコンテキスト待避のための明示的なセーブ・リストアによるオーバヘッドが問題であるとしているが、低コストな削減テクニックとしてレジスタ分割(register partitioning)を試みており興味深い。ただしスレッドあたりのレジスタ数が減るため、性能が向上する例は一部である。また、4プロセッサまでの評価となっており、より多数のプロセッサでの性能は不明である。

5. まとめ

EM-Xでは、分散共有メモリに対するアクセスは高い通信性能のもとでワード単位の粒度でおこなわれ、コヒーレントキャッシュを用いなくとも効率的な共有メモリを実現可能であることがわかった。通信レイテンシはマルチスレッドにより効果的に隠蔽できた。しかしながらマルチスレッド実行の際のレジスタ待避オーバヘッドは、EM-Xの高速スレッド切替え能力や効率的なリモートアクセスの効果を半減させていた。その対策としてはレジスタ分割や複数のレジスタセットの採用などが考えられる。ただし両者は性能と実装コストに関してトレードオフの関係にあり、今後研究が必要である。また、最適化を行わない場合の共有アクセスは非効率的なローカルアクセスを生じることがわかった。これはプログラムを修正して粗粒度なチェックコードを挿入し、ローカルアクセスを最適化することにより改善できた。今後、細粒度問題での改善を検証し、MMU機構などの検討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. 22nd Int. Symp. on Computer Architecture, pp.14-23, 1995.
2) 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.
3) 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, Proc. JSPP'93, pp.183-190, 1993.
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) 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) Wilson,R.P., Lam,M.S.:Efficient Context-Sensitive Pointer Analysis for C Programs, Proc. ACM SIGPLAN '95 Conf. on PLDI, pp.1-12, 1995.
7) 丹羽, 稲垣, 松本, 平木: 非対称分散共有メモリ上における最適化コンパイル技法の評価, 情報処理学会論文誌, Vol.39, No.6, pp.1729-1737, 1998.
8) Woo,S.C., Ohara,M., Torrie,E., Singh,J.P., Gupta,A.: The SPLASH-2 Programs: Characterization and Methodological Considerations, Proc. 22nd Int. Symp. on Computer Architecture, pp.24-36, 1995.
9) 原田, 手塚, 堀, 住元, 高橋, 石川:Myrinetを用いた分散共有メモリにおけるメモリバリアの実装と評価, Proc. JSPP'99, pp.237-243, 1999.
10) バルリ, 渡辺, 坂井, 田中: 高速通信機構を用いたソフトウェアDSMのパフォーマンス解析電子情報通信学会技術研究報告, CPSY99-66, pp.33-41, 1998.
11) Laudon,J., Lenoski,D.:The SGI Origin: A cc-NUMA Highly Scalable Server, Proc. 24th Int. Symp. on Computer Architecture, pp.241-251, 1997.
12) Scales,D.J., Gharachorloo,K., Thekkath,C.A.: Shasta: A Low Overhead, Software-Only Approach for Supporting Fine-Grain Shared Memory, Proc. 7th Int. Conf. on ASPLOS, pp.174-185, 1996.
13) 山本, 宮脇, 坂, 工藤: 共有メモリ向プログラムの通信の解析による最適化, 電子情報通信学会技術研究報告, CPSY98-61, pp.9-14, 1998.
14) Mowry,T.C., Ramkissoon,S.R.: Software-Controlled Multithreading Using Informing Memory Operations, Proc. 6th Int. Symp. on HPCA, pp.121-132a, 2000.