プログラミング 16-4
(1997. 11. 21)

高並列計算機EM-Xの性能モニタリングツール

君島 有紀[1] 坂根 広史[2] 児玉 祐悦[3]
中西正和[4]

[1,4] 慶應義塾大学大学院 理工学研究科 計算機科学専攻
[2,3] 電子技術総合研究所 情報アーキテクチャ部

並列プログラムの挙動は複雑なため、プログラムの実行に関する情報を収集し、多数のプロセッサに関する情報を分かりやすく視覚化することが必要となる。

EM-X は、ワード単位の細粒度通信およびマルチスレッドアーキテクチャをサポートする分散メモリ型の高並列計算機である。EM-X およびホスト計算機間のパケット通信機構、クロック制御機構、メンテナンスバス制御機構を用いることにより、各プロセッサの状態およびリモート関数呼び出しに関する情報を収集し、その視覚化を行なった。また、プログラム実行時に収集する情報量および情報収集が実行時間に与える影響について定量的な評価を行ない、許容範囲に収まることを確認した。

Performance Monitoring Tool on a Highly Parallel Computer EM-X

Yuki KIMISHIMA[1] Hirofumi SAKANE[2] Yuetsu KODAMA[3]
Masakazu NAKANISHI[4]

[1,4]Department of Computer Science,
Graduate School of Science and Technology,
Keio University
[2,3]Computer Science Division,
Electrotechnical Laboratory

It is necessary to show how a parallel program is executed and to visualize how multiple processors are running, because the behavior of parallel programs is complex.

EM-X is a highly parallel computer with a distributed memory. It supports fine-grain communication and multi-thread architecture. The state of each processor and information of remote function calls are gathered using packet communication, clock control and maintenance-bus control between EM-X and the host computer. And then those are visualized for the user. Some programs have been tested and the results show that the amount of information is small enough and performance impact of monitoring is permissible.

1 はじめに

並列プログラムの挙動は複雑なため、プログラマが容易にプログラムの挙動を理解出来るようなソフトウェア開発支援環境の存在が重要となる。その1つとして、プログラム実行時に収集した情報を解析し、多数のプロセッサに関する情報を見やすく表示する性能モニタリングツールが挙げられる。しかし、並列プログラムの挙動の観測および解析には、再現性やグローバルなクロックの欠如、情報収集時にかかるオーバヘッド、情報量の増大、情報の視覚化技術といった問題が存在する [1][2]。

そこで本研究では、分散メモリ型高並列計算機EM-Xにおいて、EM-Xおよびホスト計算機間のパケット通信機構、クロック制御機構、メンテナンスバス制御機構を用いることにより、性能モニタリングツールとして各プロセッサの状態およびリモート関数呼び出しに関する情報を収集し、その視覚化を行なった。

2 EM-Xアーキテクチャ

EM-Xは、通産省工業技術院電子技術総合研究所で開発された、現在80プロセッサからなる分散メモリ型高並列計算機である。現在の80PEシステムでは、1枚のPEボードに5プロセッサを搭載した16枚のPEボード、ホスト計算機(SparcStation2)と接続するためのインタフェース基板(以下IFSW)および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。各PE(要素プロセッサEMC-Y)は命令実行部内にRISCパイプラインとレジスタを持ち、ネットワークの各ポートはネットワークスループットを確保するために38ビットのデータ幅を持っている(図1)。ネットワークトポロジーはサーキュラオメガ網である [4]。

2.1 細粒度通信およびマルチスレッド

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(図2)、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、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。

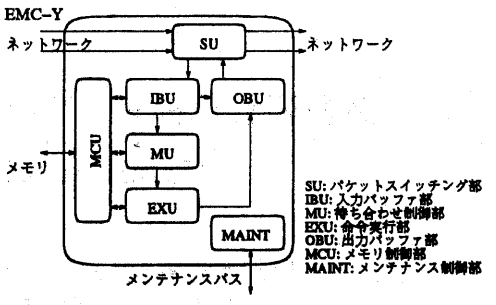

図 1: 要素プロセッサ EMC-Y

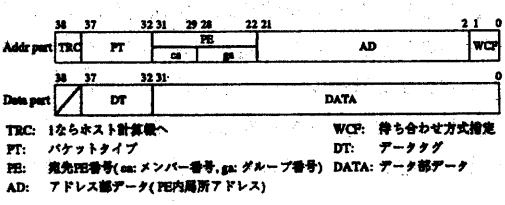

図 2: パケットフォーマット

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、パケット自身が命令実行列(スレッド)を直接起動するcontinuationとして働く。すなわち、パケットのアドレス部には実行を開始する命令アドレス、その実行に必要な引数や途中結果などを格納する作業領域のベースアドレスなどがパックされた形で収められており、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なわれる。

2.2 インターフェイススイッチ基板

ホスト計算機はS-BUSによりIFSWと接続されており、1個のEMC-Y(EXT-PE)をパケットのルータとし、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なうことが可能となっている。パケット経路には、ホスト計算機からEXT-PEへは1パケット分のレジスタ、EXT-PEからホスト計算

機へは1024パケット分のFIFOが設けてある。

クロック制御機構は、カウンタに設定した個数のクロックを供給した時点でクロックの供給を停止することが可能である。クロック供給が停止したかどうかは、ホスト計算機からポーリングによって知ることが出来る。

メンテナンスバス制御機構は、各PEボード上にあるメンテナスバススレーブ機構との通信により、全てのEMC-Yとのメンテナンス入出力を行なう。EMC-Yのメンテナンス制御部では、EMC-Y内のフリップフロップが32ビット単位で7ビットのアドレス付け(メンテナンスアドレス)され、7ビットのうち上位3ビットによりEMC-Yの機能ユニット毎に分けられている。

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、1～2MB/sec程度のスループットを確保している。

2.3 ソフトウェア環境

EM-Xでは、標準C言語の拡張仕様であるEM-CおよびアセンブラがPE使用可能であり、標準ライブラリ、スレッドライブラリ等が提供されている。

2.3.1 並列化

EM-Cにおける並列化には、並列構文やPE番号とPE内局所アドレスを組み合わせたグローバルポインタによる仮想共有メモリ的プログラミングといった言語仕様によるものと、スレッド制御やバリア同期、局所同期、メッセージ通信といったスレッドライブラリによるものがある。プログラムは全てのPEにロードされるが、最初はPE0のみで実行が始まるため、並列処理を行なうためにはSIMD的にプログラミングを行ない、PE0でプログラムが起動した後にスレッド(リモート関数)を各プロセッサで陽に起動する必要がある。

指定PEでのスレッドの起動

```
void fork(pe_addr, func,
                        n, arg1, .. , argn)
  pe_addr_t pe_addr;    /* プロセッサID */
  void (*func)();       /* 関数名 */
  int n;                /* 引数の個数 */
int remote_call(pe_addr, func,
                        n, arg1, .. , argn)
  pe_addr_t pe_addr;
  void (*func)();
  int n;
```

2.3.2 開発支援システム

現在、各PEの状態を表示する機能や、デバッグ支援として実行トレース機能(プロファイル表示)などが用意されている。プログラム中に以下の関数を挿入することにより、プログラム実行中にトレースファイルに情報を記録し、その情報を解析して表示を行なっている。

```
int em_mtrace(path, ctrl, mode, pe, clks)
  char *path;   /* トレースファイル名 */
  int ctrl;     /* トレースコントロール */
  int mode;     /* トレースの種類 */
  int pe;       /* メンテナンスPE番号 */
  int clks;     /* クロック刻み */
```

しかしこのシステムでは、各PEの状態を32ビットでトレースファイルに記録しているため、トレースファイルのサイズが非常に大きなものとなっている。また、記録した各PEの状態を時間経過に伴って色別に表示しているため、各PEの関数実行など詳細なプログラムの挙動を把握することが困難となっている。

3 本研究の方針

そこで本研究では、EM-Xおよびホスト計算機間のパケット通信機構を用いることにより、各PEのリモート関数実行に関する情報を記録し、その視覚化を行なう。

3.1 リモート関数呼びだし

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128ワード固定長の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。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 (図 3)。

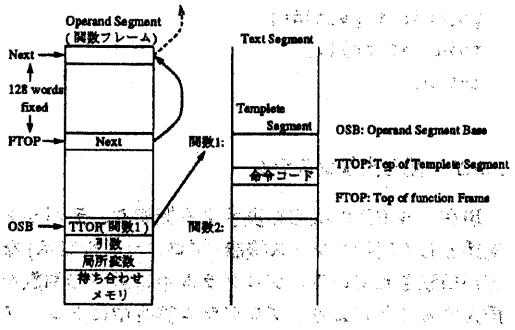

図 3: オペランドセグメント

リモート関数呼び出しを行なう際には、まずリモート PE に対してオペランドセグメントの割り当てを要求し、それへのポインタ (OSB) 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。最後に関数呼出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 特殊パケットハンドラ

パケットアドレス部のパケットタイプをセットすることにより、オペランドセグメントを必要としない関数起動を実現することが出来る。オペランドセグメントの割り当て要求など、パケットタイプとして 6 ビット (64 通り) のハンドラを指定することが可能である。しかし、直接待ち合わせによるスレッドの起動を行なう通常パケットや、オペランドセグメント内への引数の書き込みなどの直接リモートメモリ参照パケットなどハードウェアで規定されている特殊パケットに関しては、特殊パケットハンドラは用いられない。

3.3 プログラムの実行動作の記録

リモート関数呼び出しを行なうために他の PE から送られてきたパケットは、入力バッファ部の SU からのパケット受信バッファに格納され、パケットタイプによってそれぞれ処理される (図 4)。

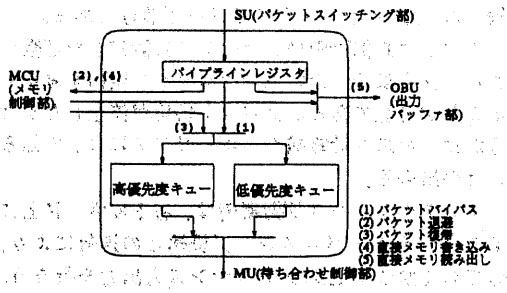

図 4: 入力バッファ部

そこで本研究では、callee 側 PE にオペランドセグメントの割り当て要求のパケットが到着し特殊パケットハンドラが起動される際に、ホスト計算機宛に OSB、TTOP および continuation をパックしたパケットを送信する。ホスト計算機はそれらのパケットからリモート関数呼びだしに関する情報を取り出してトレースファイルに記録する。そしてクロックを止め、パケットタイプを特殊パケットにセットした ack パケットを callee 側 PE に対して送信する。ack パケットを受信した callee 側 PE は特殊パケットハンドラを起動し、caller 側 PE に対して OSB をデータとする continuation パケットを送信する。

ホスト計算機側は ack パケットを送った後クロックを 1 つずつ進め、callee 側 PE の IBU の受信バッファの監視を行なう。callee 側 PE のオペランドセグメント内への引数データの書き込みが終了し、リモート関数起動パケットが到着した時点で関数の引数データをトレースファイルに記録し、クロックを再び進める。

また、リモート関数呼び出しに関するパケットがホスト計算機に届いていない場合は、指定クロック刻みでクロックを止め、各 PE の状態を 2 ビットでトレースファイルに記録することにより、トレースファイルのサイズを抑えている。

4 情報の解析および表示

トレースファイルに記録されたプログラム実行に関する情報を解析し、その視覚化を行なう。表示する情報は、各 PE の状態、引数データや呼応関係などのリモート関数呼び出しに関する情報である。多数のプロセッサに関する情報を一度に全て表示する

と表示が繁雑になり理解しにくいものとなってしまう。そこで一度に表示する情報は各 PE の状態のみとし、その後はプログラマの要求に応じて表示を行なう。

5 考察

以上で述べた手法により、フィボナッチ数およびラディックスソートを EM-X 上で実行し、生成されるトレースファイルの大きさや情報収集が実行時間に与える影響について評価を行なった。

5.1 フィボナッチ数

80PE でフィボナッチ数の 14 を求めるプログラムを実行した。従来システムおよび本システムに関しては指定クロック刻みを変化させ、実行に関する情報のトレースファイルへの記録を行なった。実行記録を収集しない通常の実行時間、従来システムおよび本システムによる実行時間を図 5 に、従来システムおよび本システムによるトレースファイルの大きさを図 6 に示す。

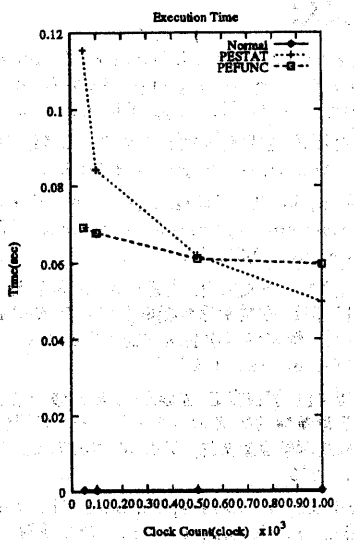

図 5: フィボナッチ数の実行時間

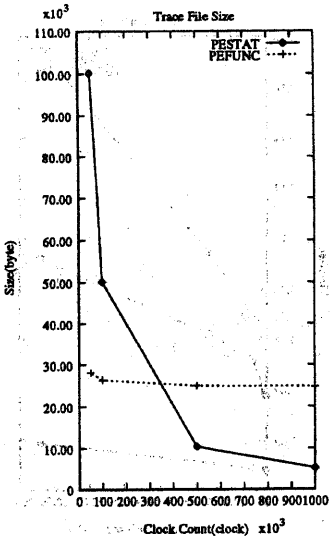

図 6: フィボナッチ数のトレースファイルの大きさ

5.2 ラディックスソート

64PE で基数を 64 としてラディックスソートを実行した。従来システムおよび本システムに関しては指定クロック刻みを 500 クロックとし、ソートするデータのサイズを変化させ、実行に関する情報のトレースファイルへの記録を行なった。実行記録を収集しない通常の実行時間、従来システムおよび本システムによる実行時間を図 7 に、従来システムおよび本システムによるトレースファイルの大きさを図 8 に示す。

5.3 実験結果の考察

以上の結果から本システムでは、フィボナッチ数の様に実行時間に対してリモート関数呼び出しが多い (377 回) アプリケーションでは、指定クロック刻みが大きくなるに従い、従来システムに比べトレースファイルが大きくなり、リモート関数呼び出しに関する情報の記録およびファイルアクセスが実行時間に影響を与えている。一方、ラディックスソートの様に実行時間に対してリモート関数呼び出しが少ない (64 回一定) アプリケーションでは、データサイズが大きくなるに従い、従来システムに比べトレースファイルの増加率が小さくなり、実行時間に与え

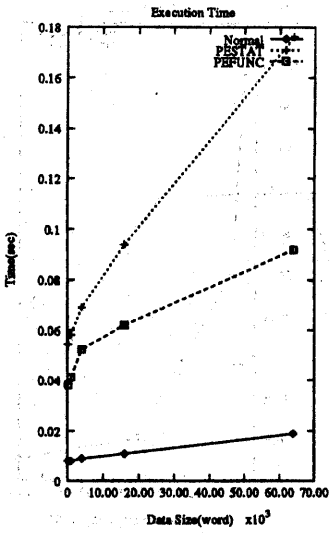

図 7: ラディックスソートの実行時間

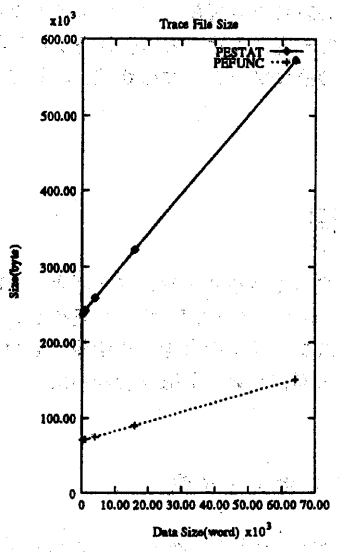

図 8: ラディックスソートのトレースファイルの大きさ

る影響も少なくなっている。

6 結論および今後の展望

本研究では、高並列計算機 EM-X 上に性能モニタリングツールを実装し、プログラム実行時に収集する情報量および情報収集が実行時間に与える影響について定量的な評価を行ない、許容範囲に収まることを確認した。本システムにより、EM-X 上におけるソフトウェア開発が促進されると思われる。

また今後は、行列演算や探索問題などの実用的なアプリケーションを実行し、同様に解析および評価を行なう予定である。

また EM-X では、各 PE で 2 ～ 4 程度の実行待ちスレッドを用意することにより、より効率的な実行が可能となると言われている。そこで、本システムを用いてスレッド切替の状況を視覚化することにより、EM-X のマルチスレッド機構についての評価も行なう予定である。

謝辞

本研究を行なうにあたり、通産省工業技術院電子技術総合研究所において実習させて頂きました。快く実習を引き受けて下さり、御指導、御検討を頂きました坂根広史氏、児玉祐悦氏ならびに並列アーキテクチャラボの方々に深く感謝致します。

参考文献

[1] Charles E. Mcdowell, David P. Helmbold. *Debugging Concurrent Programs, ACM Computing Surveys*, Vol. 21, No. 4, pp. 593-622, Dec. 1989.

[2] 菊池純男. 並列化支援システム, 情報処理学会論文誌, Vol. 34, No. 9, pp. 1158-1169, Sep. 1993.

[3] Heath, M. T., Etherridge, J. A. *Visualizing the Performance of Parallel Programs, IEEE Software*, pp. 29-39, Sep. 1991.

[4] 児玉祐悦, 坂根広史, 佐藤三久, 山名早人, 坂井修一, 山口喜教. 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, Sep. 1997.

[5] 児玉祐悦, 坂根広史, 佐藤三久, 坂井修一, 山口喜教. 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, Jul. 1995.

[6] Kodama, Y., Sakane, H., Sato, M., Yamana, H., Sakai, S., Yamaguchi, Y.. *The EM-X Parallel Computer: Architecture and Basic Performance, Proc. ISCA '95*, pp. 14-23, 1995.