計算機アーキテクチャ　83−24
（１９９０．７．１８）

並列処理ワークステーションＴＯＰ－１の性能評価環境

大庭信之　　小原盛幹　　山崎秘砂　　清水茂則

日本アイ・ビー・エム東京基礎研究所

共有メモリ型マルチプロセッサワークステーションＴＯＰ－１上に実現された、性能評価のためのツールと、その使用環境について述べる。ＴＯＰ－１は共有バスのボトルネックを解消するためにスヌープキャッシュメカニズムを採用しているが、このスヌープキャッシュの性能がシステム全体の性能に大きく影響する。スヌープキャッシュとバスに関する統計情報をリアルタイムに収集・表示するためのツールについて述べる。また、マルチプロセッサのメモリ参照のトレースデータを得るためのトレーサについて述べる。ここで得られたトレースデータを用いて、マルチプロセッサの性能を評価する方法について言及する。

Performance Measurement Environment of a Multiprocessor Workstation TOP-1

Nobuyuki Ohba　　Moriyoshi Ohara
Hisa Yamasaki　　Shigenori Shimizu

IBM Research, Tokyo Research Laboratory,
IBM Japan Ltd.

This paper describes the tools and environment of the performance measurement, which has been implemented on a multiprocessor workstation TOP-1. TOP-1 has a snoop cache mechanism to reduce the traffic on the shared bus. The performance of the snoop cache much affects on the system performance. We developed tools which give us the statistical information about the cache and bus. We also implemented a program tracer to get memory references of the TOP-1. We report on the tracer environment and how to simulate various multiprocessor architecture using trace data.

１．はじめに

近年、マルチプロセッサ構成をとるワークステーションの研究、開発が盛んに行われており、いくつかはすでに市場に登場してきている。

日本アイ・ビー・エム株式会社・東京基礎研究所では、共有メモリ型マルチプロセッサのアーキテクチャ、ハードウェア、オペレーティングシステム、コンパイラ、アプリケーション等を研究するため、並列処理ワークステーションＴＯＰ－１(Tokyo Research Parallel Processor-1)を試作・開発した[1,2,3,8]。　現在のところＴＯＰ－１は、マルチプロセッサをサポートするためにＡＩＸに手を加えたオペレーティングシステム下で、実際に稼働している。さらに、いくつかのアプリケーションも動作しており、それらを使って様々な評価を行っている[4][5]。

ＴＯＰ－１は、インテル８０３８６・８０３８７及びＷｅｉｔｅｋ１１６７より成るプロセッサ１０台、１２８Ｍバイトメモリ、１．２ギガバイトハードディスクから構成される共有メモリ型マルチプロセッサである[1]。それぞれのプロセッサは１２８Ｋバイトのキャッシュを有する。ＴＯＰ－１は共有メモリ・共有バスのアクセス競合を減らすために、スヌープキャッシュを搭載しており、このキャッシュとバスの性能がシステム全体の性能を決めていると言っても過言ではない。そこで我々は、ＴＯＰ－１を使って、キャッシュとバスを中心にシステムの性能評価を行い、今後のマルチプロセッサのハードウェア、ソフトウェアの研究、開発に役立てようと考えている。

本稿では、ＴＯＰ－１上に開発したシステム性能評価のための環境について報告する。第２節で、ＴＯＰ－１のキャッシュ・バスコントローラに搭載されている統計情報収集ハードウェアについて簡単に触れ、それを用いた性能評価用のツールを紹介する。第３節で、マイクロプロセッサのメモリアクセストレースを採るためのツール（トレーサ）について、その実現方法について報告する。

２．統計情報収集機能

ＴＯＰ－１のキャッシュ・バスコントローラには、スヌープキャッシュ及びバス関係の統計情報を収集できるハードウェア統計情報収集回路が内蔵されている[6]。この回路はいわゆるイベントカウンタの集合であり、この回路によって、以下のイベントを実測することができる。

(1)命令フェッチ、データ読み書きの回数
(2)命令フェッチ、データ読み書き、それぞれのキャッシュヒット率
(3)共有・占有データへの書き込み回数と比率
(4)プロセッサのローカルバス使用率
(5)共有バスの使用率
(6)スヌープ読み書きの回数とヒット率
(7)スヌープとプロセッサのキャッシュ参照衝突回数
(8)キャッシュ・バスシステムのレスポンス
(9)キャッシュのダーティラインのリプレイス回数

ＴＯＰ－１ではプロセッサ１０台を標準構成としているが、上に述べた情報が各プロセッサごとに得られる。さらに、物理アドレスで範囲を指定して、その中で起こったイベントのみをカウントすることもできる。この機能を使えば、スタックオペレーションのみに注目したり、ある共有データにだけ注目したりすることによって、さらに詳細な統計情報を得ることができる。

ＴＯＰ－１のＯＳには、２０ミリ秒に１回タイマ割り込みを発生し、その割り込み処理ルーチンのなかで、キャッシュ・バスコントローラ内の統計情報収集カウンタを読み出し、決められた共有メモリ上に格納する機能が組み込まれている。実際には、この読み出しルーチンが走ることによってもそれ自身カウントされることになってしまうが、一度には５個の２０ビットカウンタを読んで、メモリに足し込むだけの作業である。従って、タスクスイッチなどと比べればそのオーバヘッドは極めて少ない。

現在、この統計情報収集回路は共通のユーティリティとして開放されており、以下に述べるツールを通してユーザが利用できる。ＴＯＰ－１使用環境では、次節以降説明する、コマンドレベルのｒｓｔａｔ、プログラム埋込み型のｓｓｔａｔ、Ｘ－ウィンドウ上にリアルタイムで統計情報を表示するｘｓｔａｔの３種類の統計情報収集支援ツールが用意されている。

２．１　ｒｓｔａｔ

ｒｓｔａｔは、ＡＩＸのコマンドレベルで情報収集の対象となるプログラムを起動するツールである。得られた統計情報は標準出力に報告される。例えば、ｓａｍｐｌｅ１というプログラムを走らせてその統計情報を知りたい場合には、

```
>rstat sample1
```

と入力すればよい。すると、プログラムｓａｍｐｌｅ１が終了後、画面上に（標準出力に）以下に示すような測定結果が表示される。

```
Processor 0
Inst_ftch: Data_read: Data_write = 0.739: 0.180: 0.081
Instruction Fetch Hit Ratio = 0.9920
Data        Read  Hit Ratio = 0.9321
Data        Write Hit Ratio = 0.9352
Data Write Shared     Ratio = 0.3315
Memory Reference      Rate  = 0.5152 / clock
Access Block Rate by Snoop  = 0.049878
Access Block Rate to MPU    = 0.000348
Dirty Replace         Rate  = 0.293263
Snoop Read Hit Ratio        = 0.164672
Snoop Write Hit Ratio       = 0.307830
Memory Response             = 0.791886
Average Wait Cycle          = 0.262809
```

ここに示した例は、プロセッサ１台に関する情報だけだが、実際には、ＴＯＰ－１に搭載されているプロセッサの台数分

だけ出力が得られる。ｒｓｔａｔからプログラムを呼び出した場合でも、統計情報が集められるのはプログラムの最初と最後だけなので，対象となるプログラムへのオーバーヘッドはほとんどない。

２．２　ｓｓｔａｔ

ｓｓｔａｔは、ユーザプログラム中に統計情報収集の開始、停止、終了を明示することによって、プログラム内のさらに細かい単位で情報を収集することを目的として作られたツールである。例えば、ひとつのプログラム内の逐次実行セクションと並列実行セクションを別々に測定したい場合に有効である。プログラム例を以下に示す。initstat()は情報収集機能を初期化するために、必ず最初に置かれる。startstat()とstopstat()は、それぞれ情報収集の開始と停止を指示するもので、複数個のセクションが設定できる。closestat()は指定されたファイルに結果を出力する。

```
sample2(){
            . . .
        initstat(4);    /* 4 sections  */
            . . .
        startstat(0);   /* start sec_0 */
            . . .
        startstat(1);   /* start sec_1 */
            . . .
        stopstat(1);    /* stop  sec_1 */
            . . .
        stopstat(0);    /* stop  sec_0 */
            . . .
        startstat(2);   /* start sec_2 */
            . . .
        startstat(3);   /* start sec_3 */
            . . .
        stopstat(2);    /* stop  sec_2 */
            . . .
        stopstat(3);    /* stop  sec_3 */
            . . .
        closestat();    /* close all sections */
}
```

上記の例で示すように、各セクションは相互にネストしたりクロスしたりしていてもかまわないが、開始と停止を示すstartstat()とstopstat()は１対１に対応していなければならない。統計情報はセクション毎に別々のファイルに記録される。記録される内容はｒｓｔａｔの出力と同じであり、プロセッサごとに結果が得られる。

２．３　ｘｓｔａｔ

ｘｓｔａｔは、ｒｓｔａｔ／ｓｓｔａｔで述べた統計情報を、Ｘウィンドウ端末ディスプレイ上にリアルタイムで表示するツールである。２．１節に示したような統計情報を図１のようにグラフで表示する。ｘｓｔａｔ自身はユーザプログラムとして動作するが、他のプロセスとは独立に動作し、結果をＸサーバにリアルタイムで表示する。統計情報をプロセッサごとに表示することも可能であるし、ＴＯＰ－１全体の平均を表示することも可能である。

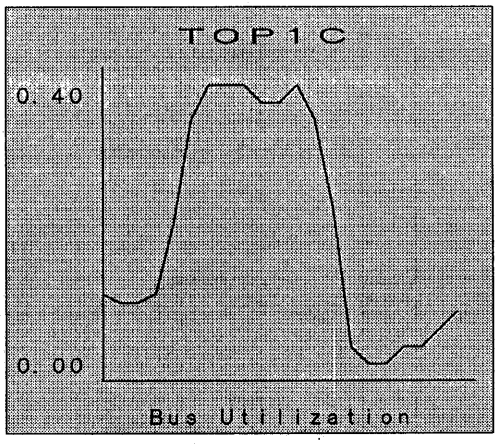

図1．ｘｓｔａｔの出力例

ｘｓｔａｔ自身は全体で１００ＫＢ程度のプログラムであり（コア部分は１０ＫＢ程度）、局所性の高いものなので、これが動作することによるオーバヘッドはごく少ない。ラプラス方程式を解くアプリケーションと同時に走らせても影響は１％以下であった。

３．トレーサ

３．１　ＴＯＰ－１トレーサ

従来から、大型・中型・小型計算機を問わず、プロセッサやキャッシュの性能を評価するために、プロセッサのメモリアクセストレースを使う方法、即ちトレースドリブン型シミュレーションが多く使われてきた。このトレースデータを採集する方法もハードウェアによるもの、ソフトウェアによるものなどいくつかあるが、それぞれ一長一短がある。

我々は、マルチプロセッサ環境下でマイクロプロセッサのメモリアクセストレースを採集できる、ＴＯＰ－１トレーサを実現した。ユーザは、トレースをとりたいプログラムのソースコードに若干手を加えるだけで、マルチプロセッサ環境でトレースを得ることができる。例えば、ユーザはアプリケーションプログラム中で、フォークもしくはジョインによってプロセスを生成したり、消滅させても一向に構わない。トレーサは実際に実行されたプロセスをそのままトレースする。このトレーサは８０３８６のシングルステップトラップ機能を利用したソフトウェア手法によるものである。現在のとこ

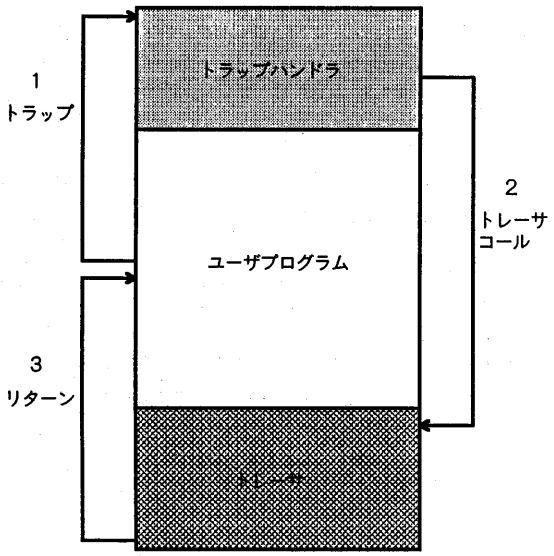

図2.　トレーサの動作手順

ろユーザプログラムのトレースのみが対象であり、ＯＳ自身のトレースをとることはできない。

３．２　トレーサの実現方法

次にトレースの実現方法を簡単に説明する。８０３８６に１命令ごとにトラップを発生させるため、トレース開始時点でトレースビットをたてておく。８０３８６は１インストラクションごとにトラップを発生し、トレース用のトラップハンドラに飛び込む。トラップハンドラは、８０３８６のレジスタの内容を共有メモリの予め決められた場所に格納し、トレーサに制御を移す。トレーサは、インストラクションポインタを使って命令を読み出してデコードし，さらにレジスタの内容を使ってメモリアクセスのアドレスとバイト数を計算する（図２）。トレーサはその結果をディスク上のファイルに記録する。ファイルに書き出す作業はシリアルに実行されるが、トラップハンドラと、アドレス計算をつかさどるトレーサはプロセス独立にかつ複数のプロセッサで並列に実行される。言い換えれば、複数のプロセスを同時にトレースした場合には、負荷の重いデコードやアドレス計算が並列処理されることになる。

トラップハンドラはカーネルに組み込まれており、Ｃ言語で２００ステップ程度の量である。一方、トレーサはほとんどの８０３８６及び８０３８７の命令をデコードし、アクセスのアドレスを計算しなければならないため、Ｃ言語で７，０００ステップ程度のプログラムとなっている。

トレーサのおおまかな性能を把握するため、Ｃで記述されたＤｈｒｙｓｔｏｎｅベンチマークをＴＯＰ－１上で実行し、トレースしてみた。並列化はしていない。トレース結果をＴＯＰ－１のハードディスクに記録する条件で、１秒間に約１，５００命令、メモリアクセス数にして約２，０００回をトレースできた。この速度はリアルタイムで実行した場合のおよそ３桁遅い速度に相当する。

３．３　トレースの実現例

ユーザはプログラムに若干のステートメントを書き加えることによって、プログラムのトレースをとることができる。例を以下に示す。

```
sample3()
{
        . . .
        inittrace("outfilename");
        starttrace();
        . . .
        /* Program Body To Be Traced. */
        . . .
        stoptrace();
        closetrace();
        . . .
}
```

inittrace()でトレーサの初期化を行う。inittrace()の引数でトレース結果の出力ファイル名を指定する。トレースの開始と停止を指定するstarttrace()と stoptrace()は、動的に対応がとれている限りプログラム中に何回現れてもよい。closetrace()によってトレースを終了する。

トレースの結果として得られる情報は次の通りである。

（１）物理プロセッサ番号
（２）プロセス番号
（３）命令フェッチ、データ読み書きの種類
（４）アクセスの実効アドレス
（５）アクセスの物理アドレス
（６）バイト数
（７）ロックの有無
（８）読み書きしたデータ
（９）ニーモニック

以下に、実際の出力結果例を示す。

```
DR 1: 1100115: 1FFFFD48   66AD48  4 0   800000
IF 1: 1100115:      2DC   6992DC  3 0        0    SUB
DR 1: 1100115:    80001C  64E01C  4 0        0
DW 1: 1100115:    80001C  64E01C  4 0 FFFFFFF9
IF 1: 1100115:      2DF   6992DF  1 0        0    INC
IF 1: 1100115:      2E0   6992E0  2 0        0    MOV
IF 2: 1100116:     8522   683522  3 0        0    CMP
IF 3: 1100116:     8525   683525  2 0        0     JZ
IF 3: 1100116:     8527   683527  5 0        0    MOV
IF 3: 1100116:     852C   68352C  1 0        0    RET
```

```
DR 3: 1100116: 1FFFFD4C   65BD4C  4 0      245
IF 3: 1100116:      245   699245  2 0        0   MOV
IF 3: 1100116:      247   699247  2 0        0   OR
IF 3: 1100116:      249   699249  2 0        0   JNZ
IF 3: 1100116:      24B   69924B  5 0        0   PUSH
DW 3: 1100116: 1FFFFD4C   65BD4C  4 0        A
IF 3: 1100116:      250   699250  6 0        0   PUSH
DR 3: 1100116:  4028B8    6588B8  4 0   800000
DW 3: 1100116: 1FFFFD48   65BD48  4 0   800000
```

プロセス番号　実効アドレス　物理アドレス　ニーモニック

プロセッサ番号　バイト数　ロック　データ

命令フェッチ(IF)
データリード(DR)
データライト(DW)

3．4　汎用ソフトウェアトレーサ

前節まではＴＯＰ－１の上に実現したトレーサについて述べた。それで得られたメモリアクセストレースは、８０３８６系の命令セットを持つマイクロプロセッサを用いたシステムの性能評価には非常に有効である。しかし、８０３８６系以外のマイクロプロセッサを用いたシステムの性能評価に関しては、メモリアクセスパターンが異なるため，考慮が必要である。そこで我々は現在、ＡＩＸのデバッガを利用して、ＲＩＳＣを対象としたソフトウェアによるシミュレータを開発中である。

開発中のソフトウェアシミュレータは、図３に示すように、トレーサプロセス、デバッガプロセス、及びターゲットプロセスから成る。ターゲットプロセスはトレースを採りたいユーザアプリケーションプログラムである。デバッガはターゲットプロセスをシングルステップで実行し、結果をトレーサプロセスに渡す。トレーサプロセスは各実行命令のオペランドのアドレス計算などを行い、トレースファイルを出力する。トレーサプロセスとデバッガプロセスはＡＩＸのパイプを利用して交信する。

3．5　マルチプロセッサシミュレータ

並列プログラムは，並列に動作する単位間で、そもそも非決定的な性質を持ち合わせている。即ち、同期をとる部分以外では、まったく独立に走って構わない。言葉を換えれば、並列プログラムの動作は、それが走るシステムによって異なる。従って、ある状況下で得られたトレースは、違ったアーキテクチャを持つシステムの性能評価にいつも有効であるとは限らない。特に、同期の方法はジョブスケジューリングも含めて、アーキテクチャやＯＳに依存すると思われる。よって、同期部分のトレースはそのままシミュレータにかけてよいとは限らない。

そこで、図４に示すように、並列プログラムを逐次実行部

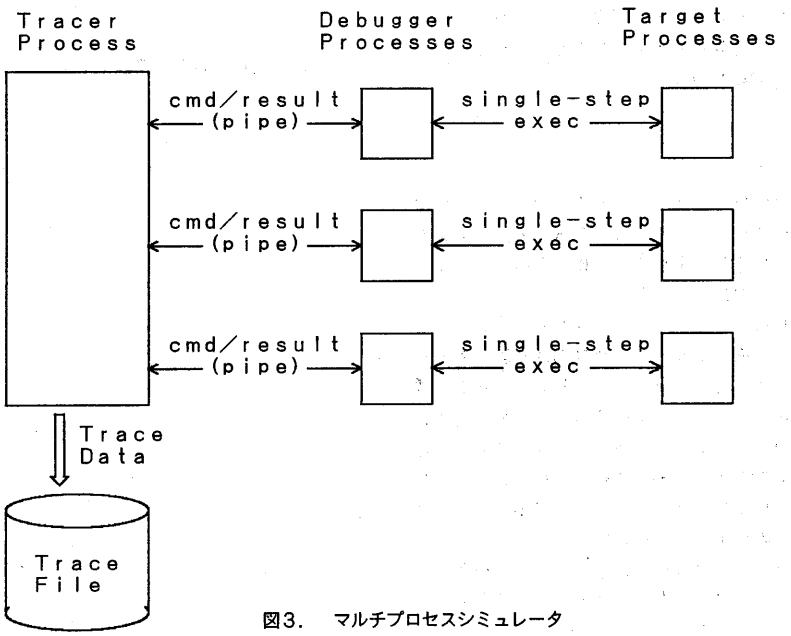

図3．　マルチプロセスシミュレータ

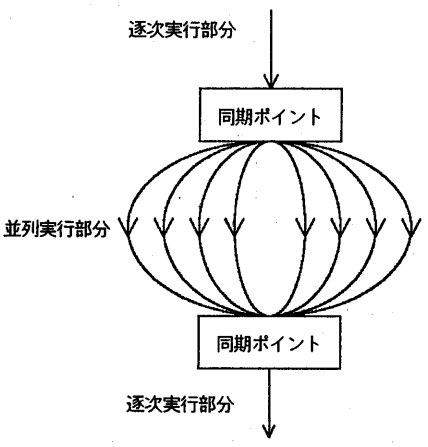

図4.　逐次実行部分と並列実行部分

分、同期ポイント、並列実行部分の３つに分けて評価することにした。逐次実行部分並びに並列実行部分の１本１本の流れはトレーサの出力をそのまま用い、同期ポイントではトレースドリブンシミュレータ自身が目的計算機の動作をシミュレートする。即ち前者に関しては、トレーサによって得られたトレースデータに基づいて、メモリアクセスを発行する。後者に関してはシミュレータ自身が、対象とするマシンの同期の手段及び構造を考慮して、シミュレートする。

このようなシミュレーションを行うためには、同期部分を明確に抜き出せるアプリケーションを使う必要がある。そこで現在、すでにＴＯＰ－１上にインプリメントされた、ＦＯＲＴＲＡＮのループを並列に実行するアプリケーションプログラムを対象として実験を行っている。

４．むすび

本稿では、並列処理ワークステーションＴＯＰ－１上に実現した性能評価ツールについて述べた。今後とも、共有メモリ型マルチプロセッサのアーキテクチャは、そのソフトウェア開発の容易さから、汎用ワークステーションに広く使われていくものと思われる。そして、スヌープキャッシュメカニズムも、共有バス、共有メモリのアクセス頻度を軽減するために、標準的に利用されていくであろう。新しく計算機を構築する場合には、キャッシュとバスの設計には十分検討が必要である。そのために、キャッシュ及びバスに関するさまざまな統計情報は必要不可欠である。また、トレースドリブン型シミュレーションは新しい計算機の性能を予測し、アーキテクチャを決定するのに重要な手段となろう。

我々は、ＴＯＰ－１から得られる、キャッシュとバスに関する統計的情報と、プログラムトレースを今後の研究開発にいかしていこうと考えている。

参考文献

[1]鈴木、黒川他　「高速並列処理ワークステーション　ＴＯＰ－１」、第３７回情報処理学会全国大会論文集，１９８８年９月。

[2]清水、大庭、森脇、中田、小原　「高性能マルチプロセッサワークステーションＴＯＰ－１」、並列処理シンポジウムＪＳＰＰ’８９論文集、１９８９年２月。

[3]穂積、白鳥他　「ＴＯＰ－１オペレーティングシステム、基本方針」、第３９回情報処理学会全国大会論文集、１９８９年１０月。

[4]Oba, Moriwaki, Shimizu :"A Snoop-Cache-Based Multiprocessor TOP-1," Proceedings of IEEE International Pheonix Conf. on Computer and Communications, March 1990.

[5]Horiguchi and Nakada : "Experimental Performance Evaluation of Parallel Fast Fourier Transform on a Multiprocessor Workstation," Proceedings of International Conference on Parallel Processing 1990.

[6]大庭、清水　「高速並列処理ワークステーション　ＴＯＰ－１　統計情報収集回路を用いた性能評価」、第３８回情報処理学会全国大会論文集，１９８９年３月。

[6]森脇、清水　「高速並列処理ワークステーション　ＴＯＰ－１　無効化型と更新型スヌープキャッシュプロトコルの共存を許すための機構」、第３８回情報処理学会全国大会論文集，１９８９年３月。

[8]清水、山内　「マルチプロセッサワークステーションＴＯＰ－１」　ｂｉｔ誌、１９９０年７月号。

*ＡＩＸはＩＢＭの登録商標です。