計算機アーキテクチャ 139－1
（2000. 8. 3）

メモリバストレースを用いた 共有バス型並列計算機のキャッシュ評価

佐藤 充, 成瀬 彰, 久門 耕一
(株) 富士通研究所

〒211-8588 川崎市中原区上小田中 4-1-1
TEL: 044-754-2665, FAX: 044-754-2664
E-Mail: {msato, naruse, kumon}@flab.fujitsu.co.jp

あらまし
実機上でメモリバストレースを取得するバストレーサ GATES (General purpose memory Access TracE System) を開発した. 共有バス型並列計算機上で Commercial Workload (DBMS 2 種) を実行し, GATES を用いてメモリバストランザクションを取得した. 取得したトレースを元に, キャッシュサイズによるバストランザクションの変化を調査した. さらに, トレースを入力とするトレース・ドリブン・キャッシュシミュレーションを行ない, より大きなキャッシュサイズを持つプロセッサを用いた場合のメモリバストランザクションの挙動を予測した. その際, シミュレーションの妥当性を調べるため, 実トレースとの比較を行ない, シミュレーションの正当性を確認した.

キーワード　　共有メモリ型並列計算機, キャッシュ, トレース, シミュレーション

Evaluation of Cache Systems on Shared-Memory Multiprocessors, Using Memory-Bus Trace

Mitsuru Sato, Akira Naruse, and Kouichi Kumon
Fujitsu Laboratories, Ltd.

4-1-1, Kami-Kodanaka, Nakahara-Ku, Kawasaki 211-8588
TEL: +81-44-754-2665, FAX: +81-44-754-2664
E-Mail: {msato, naruse, kumon}@flab.fujitsu.co.jp

Abstract
We developed memory-bus trace system, called GATES (General purpose memory Access TracE System). GATES can capture memory transactions on the memory-bus of shared memory multiprocessors. We got traces on a real shared memory multiprocessor machine on which two types of DBMS are running as commercial workloads. We evaluated effects of cache with various sizes, using these memory-bus traces. Furthermore, we made trace-driven simulator using these traces and evaluated behavior of memory-bus with larger size of caches. We checked our evaluations comparing the result of simulation and real traces.

Key words　　Shared Memory Multiprocessor, Cache, Trace, Simulation

1 はじめに

現在, 並列計算機システムは実用システムとして広く用いられている. 特にビジネスアプリケーションの分野では, 小〜中規模のサーバ計算機がシステムの中核を担っている. このような, ビジネスアプリケーション分野における並列計算機システムは, 定型的な演算処理を行なう計算機システムと比べて, その挙動を予測しにくいという特徴がある.

このようなシステム上でアプリケーションあるいは OS のチューニングを行なったり, 新しいビジネスアプリケーション向け計算機システムを設計・開発する場合には, 現状の正確な把握と将来の予測が必要となる.

システムの挙動を把握するための手段として, 現状ではシミュレーションが多く用いられている. しかしシミュレーションでは, 実機での動作に比べて速度が遅いため, 大規模なデータを扱い実行時間も長いアプリケーションを測定するには適していない. さらに, ビジネスアプリケーションのように, I/O アクセスを頻繁に起こすシステムをシミュレートするためには, I/O システムを含んだシミュレータを作成せねばならず, 現実的ではない.

そこで近年多く用いられている方法が, 実機上でトレースを取り, 取得されたトレースを元に性能評価を行なう方式である. トレースを解析することによって, アプリケーションのメモリや I/O アクセスの特性を調べ, ソフトウェアあるいはハードウェアのチューニングに役立てる. シミュレータと違い, 実機でのトレースは, 実時間に近い実行が可能であり, 大規模な商用ワークロードの解析手法として欠かせないものとなっている. このような商用ワークロードの解析としては, たとえば文献 [1] などが挙げられる.

我々は, PC サーバを対象としたトレースシステムとして, ソフトウェアによるトレーサやハードウェアトレーサを開発してきた. さらにこれらを用いて実システムの測定を行ない, 現状の解析および将来の設計に役立つデータを収集中である. 本研究では, これらのトレースシステムのうち, ハードウェアバストレーサである GATES[2, 3] を用いた結果について報告する.

2 ハードウェアバストレーサ GATES

GATES (General-purpose memory Access TracE System) は, PC サーバ用の汎用メモリアクセストレーサである (図 1). GATES は PC サーバのメモリバスを直接観測することによって,

- 各プロセッサからのメモリアクセス.
- 各プロセッサからの I/O アクセス.
- 2 次キャッシュの応答 (スヌープ結果).

などを、被測定環境に影響を与えず測定することができる. 本節では GATES の構造と特徴, また GATES によって得られるデータについて説明する.

2.1 GATES の構造と動作

GATES の全体像を図 2 に示す. GATES は, Pentium Pro processor bus[4] から GTL+[4] レベルの信号を受け取ると, それを標準 LVTTL へ電圧変換しトランザクション解析部に送る. 同時に Pentium Pro processor

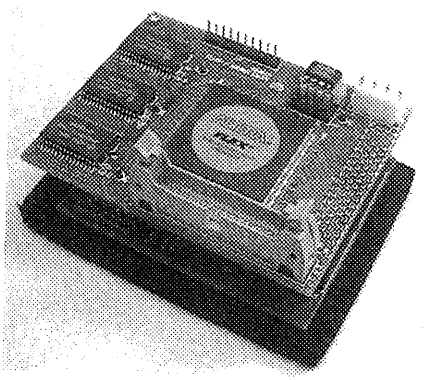

図 1: GATES

bus からクロック信号も取り出し, トランザクション解析部に送る.

トランザクション解析部では, LVTTL に変換されたバス信号を受け取り, それを解析してメモリバス・トランザクションを抽出する. このトランザクション解析部分は書き換え可能な PLD (Programmable Logic Device) で構成されているので, 処理の目的に応じた操作, たとえば信号レベルのトレースからメモリバストランザクションを再構成したり, 取得された情報を元にフィルタリングを行なったりすることが可能である.

ここで処理されたメモリバスのトランザクション情報は, メモリインタフェースを通じて GATES に搭載されているメモリ (128MB) に記憶される. このメモリに蓄えられたデータは, ホストインタフェースを通じてホスト計算機から読み出すことができる.

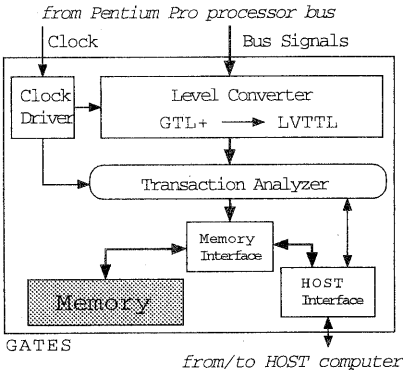

図 2: Structure of GATES

2.2 GATES によるバストレース測定

測定対象アプリケーションとしては, DBMS (DataBase Management System) を選択した. DBMS は現在 PC サーバで最も良く使われている実アプリケーションであり, 商用ワークロードの代表でもある. 今回は, DBMS システムとして 2 種類の異なる DBMS (DBMS A, DBMS B) を用いて測定を行なった.

測定の条件を表 1 に示す.

表 1: Measurement Environment

Hardware System	Fujitsu GP5000* MODEL 570 (Pentium Pro × 1 ~ 4)
CPU	Pentium Pro L2C size = 256KB ~ 1024KB
Memory	4GB EDO RAM
HDD	RAID Disk Array (18 disk nodes)
Operating System	Microsoft Windows NT Enterprise Edition 4.0
Load	OLTP transactions

*: GRANPOWER 5000

GRANPOWER 5000 MODEL 570:

GRANPOWER5000 (GP5000) は, CPU として Pentium Pro を用いた共有メモリ型並列計算機である. GP5000 は, 図 3 に示すように, ひとつのメモリバス (Pentium Pro Bus: P6 Bus) 上に 4 つの CPU ソケットが接続している構成をとる. メモリ・コントローラ (ChipSet) は, 同じく P6 Bus に 2 つ接続しており, それぞれ 2GB のメモリと PCI バスを持っている. それぞれのメモリ・コントローラは, プロセッサからのメモリ・リクエストのアドレスを判別し, 自分の持つメモリおよび PCI デバイスのアクセスを受け取る. したがって, P6 Bus を観測することによって,

- CPU とメモリとのデータのやり取り.
- CPU 同士 (キャッシュ間での) データのやり取り.
- CPU と I/O とのデータのやり取り.

をすべて知ることができる.

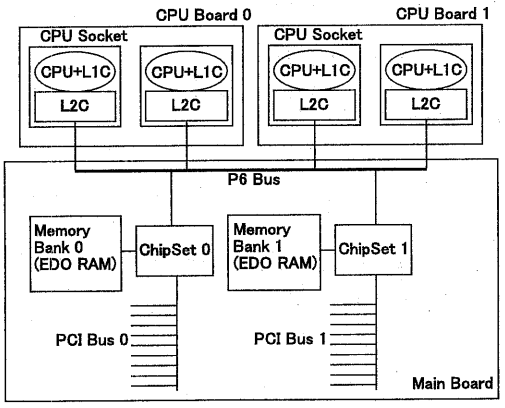

図 3: Structure of GRANPOWER 5000 MODEL 570

GATES 専用ソケット:

前節で述べたように, GATES はプロセッサ・ソケットを利用してシステムに取りつけられる. したがって, 図 3 に示すシステムでは, GATES を取りつけることによって最大構成可能プロセッサ数が 3 に減少してしまう. そのため, 今回は特別に, GATES 用のソケットを追加した CPU ボードを作成し, 最大 4 プロセッサシステムのメモリ・トランザクションまで観測できるようにした. 改造した CPU ボードを図 4 に示す. GP5000 では, 1 つの CPU ボードに 2 つの CPU が搭載され, その CPU ボードを 2 枚システムボードに指すことによって 4CPU を実現している. 図 4 に示した CPU ボードは, そのうちの 1 枚で, CPU と反対側にあるソケットが GATES 専用のソケットとなっている.

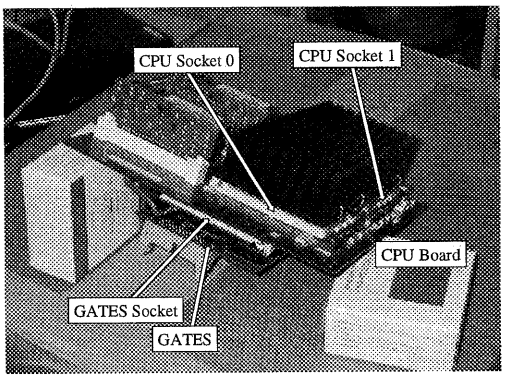

図 4: Modified CPU Board

GATES の PLD コンフィグレーション:

GATES の PLD には, トランザクション単位の記録ロジックを実装した. これは, P6 Bus 上の信号を追いかけ, パイプライン化されたバストランザクションを再構成し, GATES 内蔵のメモリに記録するものである. ひとつのバストランザクション情報は 64 ビットで構成され, GATES には最大 16M バストランザクションの情報を格納することができる. これは時間に直すと最短で約 1 秒となる, 実際にはバストランザクションの頻度はそれほど高くなく, 測定時間は 2~11 秒程度であった.

測定対象となった主な信号線は以下のものである.

- イニシエータ
- コマンド・タイプ
- メモリアドレス
- スヌープ結果 (HIT / HITM)

トレース取得方式:

トレース取得は, 条件を一定にするため以下のような手法で行なった.

まず, システム起動後, アプリケーション (DBMS) を起動させ, 一定時間稼働しシステムが定常状態に達するまで待つ. この時間は, DMBS システムにおけるトランザクション処理能力が一定値になるまでの時間をあらかじめ測定しておき, その値から一定値を足すことによって決定した.

次に, GATES をスタートさせトレースを取得する. GATES のデータ取得は, P6 Bus のバストランザクション量にもよるが, およそ 2~11 秒程度であった. その

後, GATES のホストインタフェースを用いて, GATES に蓄えられたバストランザクション情報をホスト計算機に吸い上げた.

3 トレースデータ解析

GATES によって得られたトレースデータを解析し, そこから DBMS 実行時のメモリトラフィックの特徴を調べた.

図 5 に, DBMS A および DBMS B のそれぞれについて, メモリバストランザクションをカウントした結果を示す. 図 5 で用いたトランザクションの分類を表 2 に示す. グラフは DBMS におけるトランザクション数の値で正規化している. これは同じ処理を行なうのにどれくらいのメモリバストランザクションが必要であったか, という観点から比較を行なうためである.

表 2: Classification of Memory-Bus Transactions

IOR : I/O Read	IOW : I/O Write
Invalidate: Invalidate	RdInv: Read & Invalidate
RdCd : Read Code	RdDt : Read Data
WrWb : Writeback	Wr : Write

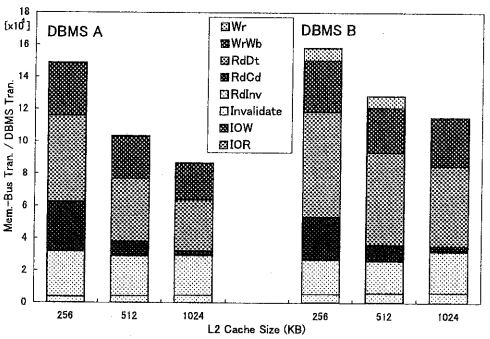

図 5: Memory-Bus Transactions (DBMS A & DBMS B)

図 5 から以下のことがわかる.

- L2 キャッシュサイズが増えると, バストラフィックが全体的に減少する. 特にコード読み出しに関してその傾向が顕著である. これはデータ領域に比べて, コード領域の方が範囲が狭い (空間的局所性が高い = キャッシュに当たりやすい) からであると考えられる.
- キャッシュサイズが増えると, invalidate (無効化要求) が増える (256KB→1024KB で約 20%の増加). これは L2 キャッシュサイズが大きくなるにつれ, L2 キャッシュ間で共有している情報が増え, データ更新時に invalidate を発行しなくてはならない場面が増えるためであると考えられる.

4 キャッシュシミュレーションによるバストラフィック解析

前章では, GATES によって得られたトレースを解析した結果について述べた. しかし GATES のトレースデータは, それ自身を解析するだけでなく, トレース・ドリブン・シミュレータの入力として用いることもできる. 本章では GATES のデータをシミュレータの入力とし, キャッシュシミュレーションを行なった結果について述べる.

4.1 シミュレーションの目的

今回のシミュレーションの目的は, 被測定対象のシステムよりも大きなキャッシュを搭載した場合, メモリトランザクションがどのように変化するかを調査することである. 前章で述べたように, 被測定対象として 256KB, 512KB, 1024KB の L2 を持ったシステムを用いたので, この範囲に関しての情報は得られている. しかし, 将来的にさらに大きなキャッシュを持ったプロセッサが登場した場合, メモリシステムにはどのような負荷がかかるかについては, 現状ではデータを外挿するしかない. しかし, キャッシュサイズの大きさとバストランザクションの特性は必ずしも線形の関係を保っているわけではなく, 複雑な要素が絡み合っている. そこで, GATES のデータを入力とし, さらに大きなキャッシュを外部に設け, シミュレーションを行なうことによって, より大きなキャッシュを用いた場合のバストラフィックを予想する.

まず第 1 段階として, 256KB キャッシュの結果を用いて, 512KB キャッシュ, 1024KB キャッシュの場合のバストランザクションを予想する. その結果と, 実測で得られた 256KB, 1024KB キャッシュのバストランザクションと比較し, 現実とシミュレーションの差を明らかにする.

次に, 1024KB キャッシュのデータをベースとして, 2048KB, 4096KB などの, 現実にはまだ存在しない大きさの L2 を持ったプロセッサのバストランザクションを予想する. この予想と, 先に調べた現実との差を加味して, 将来のシステムにおけるメモリトランザクション, 及びそれを高速化するための手法について考察する.

4.2 シミュレータの構成

大きなキャッシュを付加したシミュレータは, 図 6 のような構造をしている.

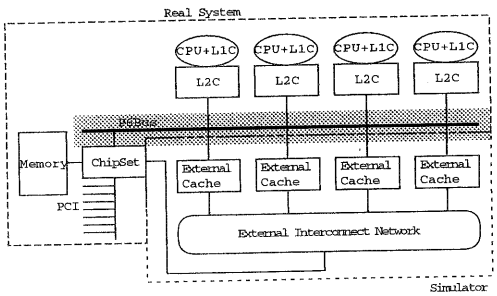

図 6: Structure of Simulator

本来の P6 Bus の外側に, 適当な大きさの外部キャッシュ (External Cache) と仮想的な相互結合網 (External Interconnect Network) を設ける. トレースに記録されている P6 Bus 上のメモリトランザクションは, すべて外部キャッシュに対する入力として扱う.

外部相互結合網としては共有バスを用いる. これはP6 Bus の代わりの, 仮想的な共有バスとなるもので, 外部キャッシュのさらに外側に位置する. メモリコントローラや I/O ブリッジである Chipset は, P6 Bus の代わりにこの外部共有バスに接続することになる.

外部キャッシュに大きな容量を持たせ, 小さな L2 キャッシュサイズ (たとえば 256KB など) のトレースを入力とすることによって, 大きな L2 キャッシュを搭載した場合のメモリバストランザクションをシミュレートすることができる. この場合のメモリバストランザクションとは, 外部共有バスのメモリトランザクションを指す.

4.3 外部キャッシュのプロトコル

外部キャッシュは, 大容量 L2 キャッシュの代わりとなるものであるため, 基本的に L2 キャッシュと同様の無効化ベースの MESI プロトコルを用いる. しかし, 外部キャッシュは L2 キャッシュからあふれてきたメモリトランザクションを入力とするため, いくつかの点で注意が必要である.

- 外部キャッシュのデータは inclusive (L2 キャッシュの内容を完全に含む) である. したがって, たとえば L2=256KB, 外部キャッシュ=512KB という構成で 512KB のキャッシュをシミュレートする.
- I/O トランザクションや uncachable トランザクションはそのまま外部共有バスに流す.
- 明示的な writeback トランザクションに関しては, キャッシュミス時に新たなキャッシュブロックをアロケートしない. これは明示的な writeback の場合には L2 キャッシュからデータが追い出されているので, 外部キャッシュでも同様のプロトコルを採用すべきであるとの判断からである. しかし暗黙の writeback (他プロセッサのデータリード時に dirty データを書き戻すための writeback) 時には, ミスした場合はキャッシュブロックをアロケートする. これは, L2 は暗黙の writeback 時には, データを shared 状態のまま保持するというプロトコルに基づくものである.

これらの処置を行なっても, 外部キャッシュは大容量 L2 キャッシュを完全にはシミュレートできない. それは, L2 キャッシュに対する read/write 要求が外部バス上からでは見えないからである. たとえば, read によって L2 キャッシュ内の LRU 制御の順序が変わるため, キャッシュラインがコンフリクトした場合どのデータが追い出されるかは外部から不明である. 追い出されるラインが dirty である場合は writeback によって検出できるが, clean の場合は外部から観測することができない. また, exclusive なデータに対する書き込みも外部からは観測できないので, 大容量キャッシュであれば本来 invalidate, 再読み出しが必要であった場合でも, シミュレーション上は clean な状態のままキャッシュからの読み出しとなってしまう可能性もある.

そのため, シミュレーション結果は現実の結果と合わせて, どのくらい有効であるかを判断しなくてはならない.

4.4 シミュレーション結果

まず, シミュレーションと現実との差を見るため, 外部キャッシュのサイズを変えた結果と, 現実のシステムを測定した結果を比較したものを図 7 に示す. なお, 図 7 における値も, 図 5 と同様に各 DBMS のトランザクション数で正規化している.

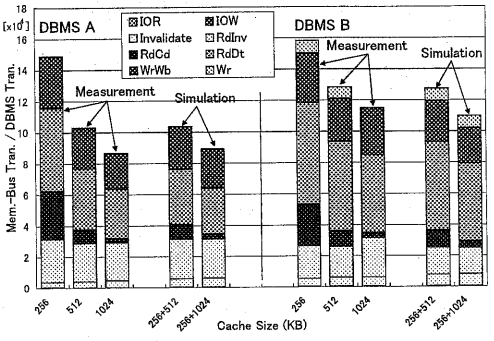

図 7: Result of Simulation (DBMS A & DBMS B)

図 7 から, L2=256KB のトレースデータを用いたシミュレーションによって, L2=512KB の場合, L2=1024KB の場合をうまくシミュレートできていることがわかる.

4.5 より大きなキャッシュを持つシステムの予測

前節の結果により, 本シミュレータで大容量 L2 キャッシュの挙動を予測することが妥当なものであると判断できたので, さらに大きなキャッシュを取りつけた場合のメモリトランザクションの挙動を調査した.

図 8 (DBMS A) および図 9 (DBMS B) は, 1024KB をベースとして, より大きなキャッシュを持つシステムに関してシミュレートした結果である. パラメータは以下の通りに設定した.

- External Cache size = 2048KB ~ 8192KB
- Associatibity = 4-way
- Protocol of External Cache = No writeback-allocate

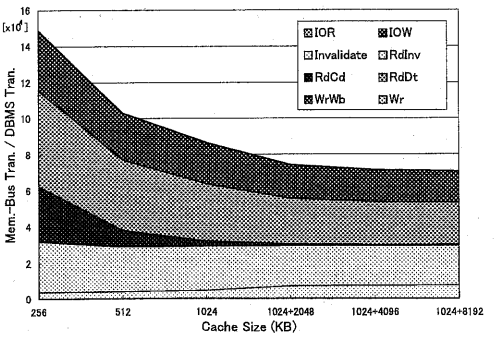

図 8: Result of Simulation (DBMS A)

図 8, 9, および前節での結果より, 次のことがわかる.

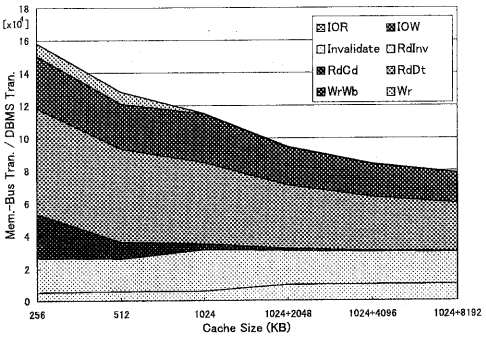

図 9: Result of Simulation (DBMS B)

- キャッシュサイズが大きくなるにつれ, code read が激減する.

  特に 4096KB 以上の場合には, コード読み出しに関するメモリトランザクションは, 全体に比べてほとんど無視できるくらいまで減少する.

- Read data, writeback は減少するが invalidate は減少しない

  キャッシュサイズが大きくなるにしたがって, 全体のトランザクションに占める invalidate の割合はかなり大きくなる. 現在シミュレートしているシステムは共有バス型のシステムを仮定しているが, これが point-to-point のネットワークをもったシステムであれば, invalidate は broadcast できず, 各コピーノードにひとつづつ配ることになる. そうなれば, さらに多くの invalidate トラフィックが発生することになり, ネットワーク全体の大きな部分を占める.

  invalidate はコントロールメッセージであるので, そのメッセージ長は短い. 一般に, 短いメッセージ長のトランザクションが大量に発生すると, ネットワークの利用率は低減する. そのため, 将来, 大きなキャッシュを持つプロセッサを PE (Processing Element) とするシステムは, ネットワークのバンド幅に過大な負荷をかける可能性がある. 逆に言うと, それだけの負荷に耐えるだけのネットワークを用意するか, invalidate を効率的に行なう方法, たとえば multicast や broadcast の利用を検討する必要がある.

- Read 系 (read code, read data) に比べて writeback の減少度合いが小さい.

  従来, メモリに対する書き込みは読み出しに比べて少なく, かつプロセッサの構造から書き込みは遅延しても問題ない場合が多いため, メモリシステム全体を読み出しに対して最適化し, 書き込みを遅延する場合が多かった. しかし, キャッシュの大きな CPU を用いたシステムでは, 書き込みトランザクションが従来より多くの割合を占めるため, 書き込みトランザクションがメモリ系のスループットを圧迫し, 従来の最適化が通用しない可能性が考えられる.

5 まとめ

本研究では, ハードウェアバストレーサを用いて, PC サーバのメモリバストラフィックを解析した結果について報告した.

まず PC サーバ上で実アプリケーション (DBMS 2 種) を実行したときのメモリバストレースを取得し, そのデータを解析し, プロセッサの L2 キャッシュサイズとバストラフィックの関係について調査した. その結果, キャッシュサイズが大きくなるにつれ, DBMS の 1 トランザクションを処理するのに必要なメモリバストラフィック量は減少すること, 特にコード読み出しが減少することがわかった.

さらに, このトレースデータを元にトレース・ドリブン・シミュレーションを行ない, シミュレーション結果による予測と実際の測定との差異を調べた. その結果, シミュレーション結果は現実をうまく反映しており, シミュレーションによってより大きなキャッシュを持つプロセッサを用いた場合のシステムの挙動を予測することができることが明らかになった.

このシミュレーションを用いて, 現実には存在しない大きな L2 キャッシュを持ったシステムに関してシミュレーションを行ない, 将来のシステムのメモリにかかる負荷について予想した. その結果, invalidate や書き込みのトランザクションが従来よりも大きな割合を占め, これらについて充分に検討したシステムを構築する必要があることがわかった.

現在トレースを取得している GATES は, Pentium Pro という一世代前のプロセッサを対象としている. しかし現在では, Pentium II(III) Xeon や Itanium など, さらに高速かつキャッシュ容量の大きなプロセッサが市場に出てきている. 我々の GATES もそれに対応すべく, 現在 GATES-II という, Pentium II(III) Xeon 対応のものを作成し, 新たなサーバ環境での測定を行なっている.

今後は, 現在のデータと新たに測定したデータとを合わせて, さらに正確な挙動予測を行なっていく予定である.

参考文献

[1] Luiz Andrè Barroso, Kourosh Gharachorloo, and Edouard Bugnion. Memory System Characterization of Commercial Workloads. In *Proceedings of the 25th Internaltional Symposium on Computer Architecture*, June 1998.

[2] 佐藤充, 成瀬彰, 久門耕一. GATES (PC サーバ用 汎用メモリアクセストレースシステム) の開発. 情報処理学会 第 59 回全国大会 講演論文集, September 1999.

[3] 成瀬彰, 佐藤充, 久門耕一. GATES による Commercial Workload アクセスパターンの分析. 情報処理学会 第 59 回全国大会 講演論文集, September 1999.

[4] Intel Corporation. *Pentium Pro Family Developer's Manual Volume1: Specifications*, 1996.