計算機アーキテクチャ 106－5
（１９９４．６．１３）

ビジネス向けマルチプロセッササーバに適した キャッシュー致保証方式の設計と評価

森岡道雄† 中三川哲明† 黒沢憲一† 石川佐孝‡

†日立製作所 日立研究所
〒319-12 茨城県日立市大みか町７丁目１番１号

‡日立製作所 オフィスシステム事業部
〒243-04 神奈川県海老名市下今泉810番地

ビジネス向けマルチプロセッサを対象に、ウイークリィオーダモデルを効率よく実装するキャッシュー致保証方式の提案と評価を行った。ビジネス向けのトランザクション処理では、ＯＳ共有データのピンポン現象やタスク競合に起因するキャッシュミスが性能低下の大きな要因となっており、ウイークリィオーダモデルを活用したメモリアクセスのパイプライン化やレイテンシ隠蔽が重要な課題である。本論文では、代表的なキャッシュ制御方式であるライト無効化方式とライト放送方式それぞれを対象に考察した。ライト無効化方式では、ストアミスのみを登録するストアミスバッファとノンブロッキングキャッシュを組み合わせることによって、複数のキャッシュミス処理をパイプライン処理する方式を提案した。また、ライト放送方式では、ブロードキャスト処理を一定期間遅延させることによって、複数のブロードキャストをまとめて処理する遅延ブロードキャスト方式を提案した。ＯＳ処理やタスクスイッチの影響も含めてOLTPプログラムを評価できるシミュレータにより、各方式の性能評価を行った。

Design and Evaluation of the Cache Coherent Mechanism for Business Multi-Processor Server

M. Morioka† T. Nakamikawa† K. Kurosawa† S. Ishikawa‡

†Hitachi Research Laboratory, Hitachi, Ltd.
7-1-1, Omika-cho, Hitachi-shi,
Ibaraki-ken, 319-12 Japan

‡Office Systems Division, Hitachi, Ltd.
810, Shimoimaizumi, ebina-shi,
kanagawa-ken, 243-04 Japan

This paper discusses the architecture and performance of a cache coherent mechanism under the weakly consistency model for a business multi-processor server. In an OLTP (On Line Transaction Processing) system, cache misses due to task conflicts and ping-ponging of the kernel shared data degrade the performance of the multi-processor. Since the weakly consistency model allows the pipelining or latency hiding of memory accesses, it becomes important feature for the multi-processor.

We propose a store miss buffer architecture for the write-invalid protocol. This architecture combines the store buffer, which only holds the store misses, and the non-blocking cache in order to execute the load/store miss recovery in parallel. As for the write-broadcast protocol, we propose the delayed broadcast architecture, which can delay and accumulate the broadcast transactions in order to reduce frequencies of the broadcast transactions. Proposed architecture are evaluated by trace-driven simulator which can evaluate the OLTP including kernel activities.

1．はじめに

銀行・証券や座席予約システム等のビジネス分野で、高性能・拡張性を兼ね備えたマルチプロセッササーバの重要性が高まっている[1)2)3)]。代表的なビジネスシステムであるOLTP(On-Line Transaction Processing)では、ディスクやネットワークへのアクセス頻度の高い多数個のタスクが同時に実行される。このため、マルチプロセッサ上で実行するとＯＳ共有データのピンポン現象やタスク競合に起因するキャッシュミスの頻度が高くなり、メモリ性能がシステム全体の性能を左右する重要な要因となる。

メモリ性能を大幅に改善できるメモリモデルとして、ウイークリィオーダモデルが注目されている[4)5)]。これは、共有データの排他制御をプログラム責任とし、プロセッサが一旦アクセス権を得たなら、共有データへのロード・ストアをアウトオブオーダで実行可能とするモデルである。本方式とノンブロッキングキャッシュ[6)]を組み合わせることによって、メモリアクセスのパイプライン化やレイテンシを隠蔽することが可能となる。ウイークリィオーダモデルを十分活かすためには、１つのプロセッサからの複数のメモリアクセスを並列に処理できる機構が不可欠である。従来これらの機構を効率良く実現する方式を提案した論文は少ない[6)7)]。また、ウイークリィオーダモデルの性能評価に関しては、従来数多くの論文が発表されている[9)10)]。しかし、主に科学技術計算プログラムによる評価が中心であり、OLTPプログラムにおける効果を定量的に評価した論文はない。特に、OLTPプログラムでは約３－４割がＯＳの処理と言われており[8)]、タスクスイッチによる影響やタスク同士の競合といった点で科学技術計算プログラムとは異なった特性を持つと考えられる。

本論文では、著者等が試作したビジネス向けマルチプロセッササーバを対象に、ウイークリィオーダモデルを効率良く実現するキャッシュ一致保証方式を検討する。代表的なキャッシュ制御方式である、ライト無効化方式とライト放送方式それぞれに関して考察する。ライト無効化方式では、従来のストアバッファを拡張し、ストアミスのみを登録するストアミスバッファ方式を提案する。本方式により、ロードミス処理とストアミス処理を並列処理するとともに、複数のストアミス処理をパイプライン化することが可能となる。一方、ライト放送方式に関しては、ブロードキャスト処理を一定期間遅延させることによって、複数のブロードキャストをまとめて処理する遅延ブロードキャスト方式を提案する。これら方式の性能評価に関しては、OLTPの評価に適したトレースドリブンシミュレータを構築する。これは、OLTPの命令トレースを入力として、ＯＳ処理やタスクスイッチの影響も含めて評価できるシステムである。

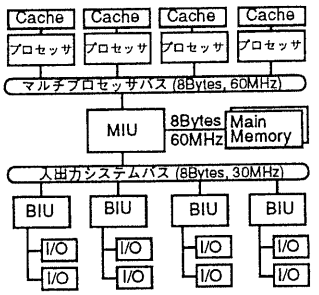

図1 マルチプロセッササーバのシステム構成

2．マルチプロセッサの概要と課題

2．1 システム構成

対象としたマルチプロセッサ試作機の構成を図１に示す。プロセッサは、120MHzのRISCプロセッサであり、キャッシュは命令・データ融合で、容量は４Ｍバイトまで拡張できる。ラインサイズは３２バイトでストアイン方式を採る。マルチプロセッサバスは８バイト幅のアドレス・データマルチプレックスであり、60MHzで動作する。スプリットバス方式により、実効性能で384Mバイト/秒を実現する。主メモリは、２バンク４ウエイのインターリーブ方式により最大480Mバイト/秒のデータ転送性能を持つ。マルチプロセッサバスでは、スヌープ方式によってキャッシュ一致保証を行う。ライト無効化方式を基本とし、キャッシュメモリ内の各ラインがModified、Exclusive、Shared、Invalidの４状態をとりうるキャッシュ一致保証プロトコル[11)]を採用する。

図２にキャッシュ一致保証機構の基本動作を示す。あるプロセッサがデータ読みだしのトランザクションを発行すると、全てのプロセッサにおいてキャッシュ一致保証制御が起動される。この時主メモリの読みだしも並列に起動される。キャッシュ一致保証の結果は、６バスサイクル後に報告される。競合を避けるために図２に示すように２つのキャッシュ一致保証制御をパイプライン処理できる。キャッシュ一致保証の結果、最新データが

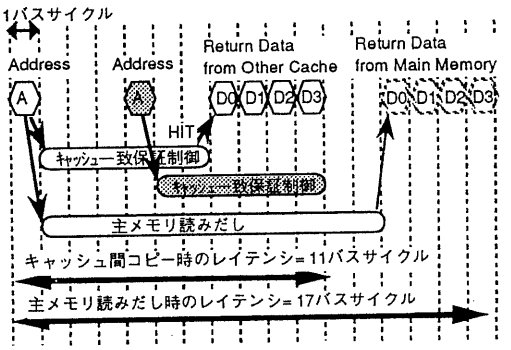

図2 マルチプロセッサバスの基本動作

他プロセッサのキャッシュにあれば、プロセッサ間で直接データが転送される（キャッシュ間コピー）。

2．2 OLTPシステムにおける課題

OLTPプログラムは、大規模データベースに対する読みだし・更新要求を受け付け処理するといった比較的単純なタスクが、多数個並列に処理されるプログラムである。マルチプロセッサの性能を低減させる要因として以下の２つが考えられる。第１は、頻繁なI/O処理のためにタスクスイッチの頻度が高くなる。このため、キャッシュ上に登録されたデータ構造が、他のタスクによって破壊されてしまい、キャッシュミス率が高くなる。第２は、ネットワークやディスクアクセスに関連してシステムコールの発生頻度が高くなる。１トランザクションで２４回のシステムコールが発行されるとの報告もある[8]。このため、ユーザ空間とカーネル空間での実行が頻繁に入れ替り、キャッシュ上で競合しミス率が高くなる。また、ＯＳが管理するI/O管理テーブル等のタスク間共有データが各プロセッサのキャッシュメモリ間で干渉を起こし、性能低下の原因となる。

特に、キャッシュ容量が数Mバイトまで大きくなると、タスク間共有データのキャッシュ間干渉が性能低下の主な原因になってくる。代表的なキャッシュ間干渉の制御方式には、ライト無効化方式とライト放送方式がある。ライト無効化方式は、あるプロセッサが共有データに書き込みを行うときに、他プロセッサのキャッシュに登録されている対応データを無効化する方式である。一方、ライト放送方式は、共有データに書き込む時には、他の全てのプロセッサに変更内容をブロードキャストする方式である。

OLTPプログラムをライト無効化方式のマルチプロセッサで実行すると、各プロセッサが共有データにアクセスするたびに、タスク間共有データが複数のキャッシュを行ったり来たりするピンポン現象が発生し、ミス率が増大する。一方、ライト放送方式では、タスク間共有データへの書き込みは、全てのプロセッサにブロードキャストされるため、ピンポン現象によるミス率増加を低減できる。しかし、ブロードキャストの頻度が多くなり、ブロードキャスト処理待ちのために性能が大幅に低下するといった問題がある。これらの課題を解決する手段として、ウイークリィオーダモデルを活用しキャッシュミス時のメモリアクセスをパイプライン化したり、ブロードキャスト等のキャッシュ一致保証処理を命令実行の裏に隠蔽することによって、性能向上をはかる技術が注目されている。

3．高性能キャッシュ制御方式

本章では、ウイークリィオーダモデルの概要を述べ、これを効率良く実装する高性能キャッシュ制御方式を提案する。

3．1 ウイークリィオーダモデルの概要

従来のマルチプロセッサでは、共有メモリへのアクセスに対してストロングオーダモデルを採用したものが多い。これは、マルチプロセッサ上の並列プログラムの実行があたかも１台の計算機の上で時分割に実行された結果と同じように見えるモデルである[5]。本モデルによれば、プログラム間の排他制御を変数の操作で容易に記述できる利点がある。しかし、このモデルを実現するには共有メモリへのロード・ストア命令はプログラムに現れた順序を厳密に守らねばならない。このためストアアクセスのバッファリングやメモリアクセスのパイプライン化が難しくマルチプロセッサの性能を十分に引き出すことができない。これに対してウイークリィオーダモデルは、共有メモリの排他制御をプログラム責任とし、プロセッサが一旦アクセス権を得たなら、共有メモリへのロード・ストアをアウトオブオーダで実行可能とするモデルである。メモリオーダの緩さによっていくつかのモデルが提案されている[4)5)]。本モデルによって、メモリアクセスのパイプライン化やレイテンシを隠蔽することが可能となる。

3．2 実装上の課題

メモリアクセスのパイプライン化やレイテンシ隠蔽を実装する場合、ロードアクセスよりもストアアクセスでの効率が高い。これは、ロードアクセスでは必要なデータがロードされるまで命令実行が継続できないのに対し、ストアアクセスは置いてきぼり制御が可能なためである。また、我々の評価ではOLTPではストアミスの頻度が全データミスの約50-60%を占め、ストアミス処理を隠蔽することにより高い効果が期待できることが明らかになっている。図３に従来型ストアバッファを活用したウイークリィオーダ方式の実装例を示す。プロセッサとキャッシュの間にストアバッファを設ける。ロードアクセスはアドレス競合が起きない限りストアバッファをバイパスしてキャッシュをアクセスできる。キャッシュはノンブロッキングであり、ストアミス処理中あるいはキャッシュ一致保証処理中（他プロセッサに対するデータパージやデータブロードキャスト）でもアクセスを許す。図３(a)の命令実行タイミングに示すようにストロングオーダでは、１つのミス処理が完了するまで次の処理に進むことができず、ストアバッファやノンブロッキングキャッシュを活用できない。一方、ウイークリィオーダ方式(b)では、ストアミス処理・キャッシュ一致保証処理の置いてきぼり制御による効果や、ロードミス処理とストアミス処理をパイプライン化する効果により、ストア処理に伴う待ち時間を隠蔽することが可能である。

しかし、本方式では以下の問題点がある。ストアミ

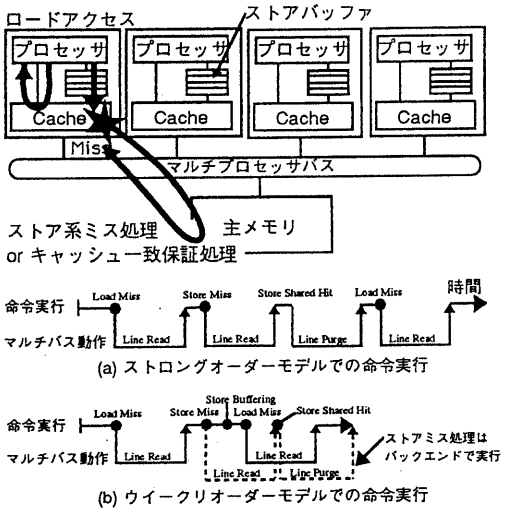

図3 ウイークリオーダーモデルの概要

ス処理やブロードキャスト処理等を置いてきぼりにできるが、データアクセスのローカリティから、ストア処理中のラインに続いてロードアクセスが発行される確率が高いと予想される。表１は、実際にOLTPプログラムの命令トレースを分析し、同一ラインへのストアアクセスと後続ロードアクセスの命令間隔の分布を示した。この結果、ロードアクセスの21.5%が同一ラインのストアアクセスに続いて発生し、この内の約半分である10.6%は０－１５以内の命令間隔で発生することが明らかとなった。このため、ライト無効化方式では、ストアミス発生に伴ってストアバッファに滞留が起こると、後続のロードアクセスが滞留エントリと競合を起こす確率が高い。また、ライト放送方式でもブロードキャスト待ちによってストアバッファの滞留が多くなると、後続のロードアクセスが競合を起こす確率が高くなる。このため、競合したストアアクセスが終了するまで待たされてしまい、ストア処理を十分隠蔽することができないといった問題が発生する。

表１　同一ラインへのストアーロード間隔

同一ラインへのStore-Load命令間隔	Load命令当りの発生率(%)	全体に占める割合(%)
0 - 15	10.6	49.5
16 - 31	1.3	6.2
32 - 63	1.4	6.4
64 - 127	1.2	5.8
128 - 255	1.4	6.7
256 - 511	1.4	6.5
512 - 1023	0.8	3.6
1024 -	3.3	15.4
合計	21.5	100.0

３．３ ストアミスバッファ方式

ライト無効化方式において上記問題点を解決するには、ストアミスに伴って発生するストアバッファの滞留エントリ数を極力低減することが必要である。そこで、従来のストアバッファを拡張したストアミスバッファ方式を提案する。本方式の狙いは、ストアミスのみを登録する効果と、複数のストアミス処理をパイプライン処理する効果により、ストアバッファの滞留数を削減し、競合を低減することにある。

図４に、ストアミスバッファ方式の概念と、従来型ストアバッファとの比較を示す。ストアミスバッファでは、バッファに登録する前に一旦データキャッシュをアクセスし、ストアミスあるいは、キャッシュー致保証制御が必要なもののみストアミスバッファに登録する。ストアミスバッファにエントリがあれば、必要なトランザクションが発行される。エントリが複数個あれば、同時に複数のトランザクションが発行され得る。ストアミス処理中のラインへのストア命令は、アドレス比較によって検出されストアミスバッファ登録のみが行われる。従来型のストアバッファでは、図４(a)に示すようにストアミス処理が逐次的に処理されるため、前章で述べたように後続のロードアクセスがストアバッファとアドレス競合を起し、命令の実行が中断される。一方、ストアミスバッファ方式(b)では、ストアミス処理のパイプライン化が可能となり、滞留エントリを高速に処理することが可能である。更にストアミスしたもののみを登録するので、滞留エントリ数を削減でき、ロードアクセスとのアドレス競合を低減できる効果がある。

３．４ 遅延ブロードキャスト方式

ライト放送方式では、ブロードキャストの頻度が高いために、ストアバッファの滞留が多くなり、後続のロードアクセスが競合を起こし性能低下の大きな要因と

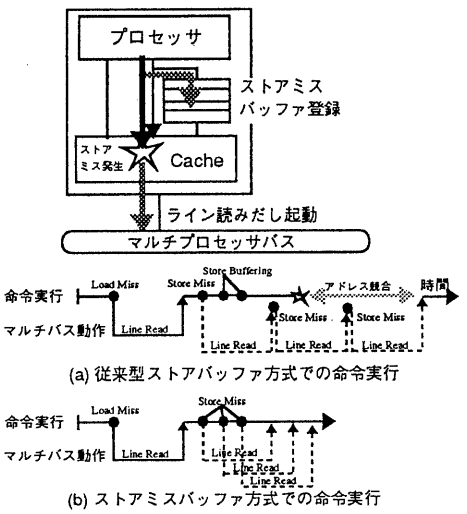

図4 ストアミスバッファ方式

なっている。これを解決するには、ブロードキャストの処理性能を高めストアバッファの滞留エントリ数を低減することが必要である。そこで、ウイークリィオーダモデルの特性を活かし、ブロードキャスト処理をバッファリングして一定期間遅延させ、複数のブロードキャストをまとめて処理する遅延ブロードキャスト方式を提案する。図5は、遅延ブロードキャスト方式の概念を示す。ストアバッファからのアクセスでブロードキャストが必要な場合、キャッシュに書き込むとともに、該当ラインを一旦ブロードキャストバッファに登録する。この時、ブロードキャストバッファに既に同一ラインが登録されていれば該エントリにデータを書き込む。オーバーフローを避けるために、ブロードキャストバッファに滞留できるエントリ数を限定する。限定数を超過した場合は古いエントリから追い出される。ブロードキャストバッファのエントリは以下の条件でマルチプロセッサバスにブロードキャストされる。即ち、同期命令が発行されたとき、許された滞留エントリ数を超過したとき、許された滞留時間を超過したときである。本方式によれば、ストアアクセスのローカリティから、複数のブロードキャストをまとめてブロードキャストすることが可能となり、ブロードキャストの処理性能を大幅に改善することが可能となる。

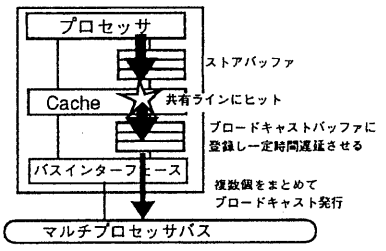

図5 遅延ブロードキャスト方式

4．性能評価環境

4．1 評価システム

OLTPプログラムでは、タスクスイッチやI/O制御も含めた評価が不可欠である。しかし、これらをソフトウエアだけで評価することは困難であり、一般にはマルチプロセッサ実機のハードウエアモニタで評価が行われている[12)13)]。これに対し本論文では、OLTPの特徴を活かし、ＯＳ動作を含めた評価が可能なトレースドリブンシミュレータを構築した。本システムは、命令トレーサとマルチプロセッサシミュレータから構成される。命令トレーサ[15)]により、単一プロセッサ実機上でOLTPプログラムを実行し１トランザクション相当の命令列を複数採取する。この命令列は、タスクスイッチやシステムコールといったＯＳ処理を含み、ロードストア命令ではそのメモリアドレスも一緒に採取する。マルチプロセッサシミュレータでは、採取した複数の命令トレースをそれぞれプログラムカウンタを持ったサーバタスクとみなしサーバプールを定義する。そして、複数のサーバタスクを並列に実行し、マルチプロセッサバスやメモリシステムの動作を１サイクル単位で模擬する。タスクスイッチポイントに来ると指定されたタスクスケジュールに従い、サーバプールから次に実行すべきサーバタスクが選択される。

4．2 マルチプロセッサモデル

試作したマルチプロセッサは、命令・データキャッシュ融合を採用しているが、本評価ではマルチプロセッサバスの動作を分析することに重点を置き、単純な命令・データ分離型のモデルで評価した。各キャッシュは、ダイレクトマップ方式、コピーバック型で、ラインサイズは32バイトとした。競合をさけるために、キャッシュ検索アドレスはタスク番号によりハッシュされる。キャッシュはウオームスタートとする。待機するサーバタスク数は1プロセッサ当たり16タスクとし、ラウンドロビンポリシーで割り当てられる。

メモリアクセスのレイテンシは、競合が無い場合キャッシュヒットで１マシンサイクル、キャッシュミスで３９マシンサイクルとした。キャッシュ一致保証のタイミングは図２に示したモデルを仮定する。ライト無効化方式では、全アドレス領域をライト無効化プロトコルのモデルで評価するが、ライト放送方式では、カーネル領域のみライト放送プロトコルとし、ユーザ領域はライト無効化プロトコルとするモデルで評価した。これは、OLTPでは、タスク間共有データはカーネル領域だけであり、カーネル領域だけをブロードキャストする方式が最適と判断したためである。

本評価システムの限界は以下である。ウイークリィオーダモデルにおける同期命令のオーバヘッドを評価できない。これは、基本となる命令トレースが単一プロセッサ実機上で採取されたため同期命令が明示されていないことに起因する。同期命令の影響は、数%程度[9)]と予想される。

4．3 ベンチマークプログラム

評価には、OLTPの代表的なベンチマークであるTPC-B[14)]を用いた。表２は、TPC-BとSPEC89の命令出現頻度を比較したものである。TPC-BはSPEC89の整数系プログラムとほぼ同等の命令出現頻度となっているが、ＯＳ動作などかなり異なる特徴を持つと予想される。表３は、アドレス領域ごとのアクセス比率を示している。I/O関連のシステムコールや割り込み処理のために、カーネル領域へのアクセスが全体の45%を占める。データ領域だけでみるとカーネルアクセスが29%を占める。タスク間共有データの比率はカーネルデータ領域の比率と同じで

あり11%となる。ユーザ・カーネル領域におけるスタック構造のライト比率は約50%となっている。

表2 命令出現頻度の比較

Instruction		TPC-B	SPEC89	
			Integer Gr.	Floating Gr.
Load		23.2%	26.8%	35.6%
Store		12.1%	10.5%	10.0%
Branch	Conditional	17.3%	17.9%	4.9%
	Un-conditional	5.8%	4.1%	1.4%
Others		41.6%	40.7%	48.2%

表3 アドレス領域毎のアクセス比率

Space	Area	Ratio(%)		Write Ratio(%)
User	Text	27.7	27.7	0.0
	Data	10.2	26.8	26.3
	User stack	10.6		35.1
	U area	2.1		54.6
	Kernel stack	3.9		51.6
Kernel	Text	34.5	34.5	0.0
	Data	3.0	11.0	22.3
	Interrupt stack	2.8		47.5
	Shared-Memory	2.7		33.7
	Other data	2.5		18.7
sum		100.0	100.0	

5. 評価結果

5.1 TPC-B基本特性

図6、図7はストロングオーダモデルにおける命令・データキャッシュのミス率分析を示す。ウイークリィオーダモデルでも結果は同様になる。図中、4台マルチと単一プロセッサを比較して示しており、またデータキャッシュでは、ライト無効化方式とライト放送方式を比較している。キャッシュ容量が64KBと小さい場合、データミス率よりも命令ミス率のほうが大きくなる。これは、OLTPでは科学技術計算に比べてループ処理が少なく、また、頻繁なシステムコールによってユーザとカーネルが競合を起こすためと考えられる。命令ミス率はキャッシュ容量の増加に伴って大幅に減少する。これは、OLTPでは共有ライブラリを使用する頻度が高く、命令の再利用率が高くなるためである。

データキャッシュでは、キャッシュ干渉に伴うミス要因として共有ミス、マイグレートミスが重要になってくる。共有ミスは、主にカーネル領域において共有データのピンポン現象に起因するミスである。マイグレートミスは、タスクがマイグレートすることによって、他プロセッサ上に登録していた自分のデータ構造を無効化することに起因するミスである。ライト無効化方式では、キャッシュ容量を増すにつれて共有ミス、マイグレートミスの占める割合が高くなる。このため、データミス率は大容量キャッシュにも係わらず高い比率になる。一方、ライト放送方式では、共有データを毎回ブロードキャストすることによりピンポン現象を避けることができ、共有ミスを低減できる。

表4は、命令当たりのロード・ストアミス発生率とその比率を示している。ライト無効化方式、ライト放送

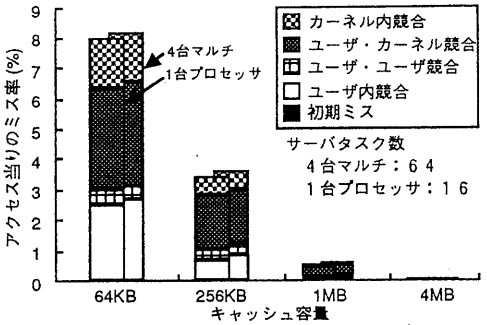

図6 命令キャッシュのミス率

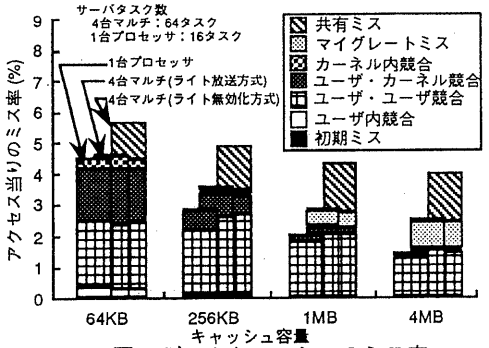

図7 データキャッシュのミス率

方式共に、ロード・ストア命令の出現頻度が2対1であるのに対し、ミス率の比率はほぼ同等となっている。このことから、OLTPではストアミス処理を隠蔽することによって高性能化できることが明らかになった。

表4 ロードミス／ストアミスの比率

ライト処理方式	キャッシュ容量	Load Miss(%)(命令当り)	Store Miss(%)(命令当り)	L/S比率
ライト無効化方式	64KB	1.00	0.87	1.15
	256KB	0.83	0.82	1.01
	1MB	0.70	0.72	0.97
	4MB	0.63	0.72	0.88
ライト放送方式	64KB	0.79	0.77	1.03
	256KB	0.57	0.62	0.92
	1MB	0.42	0.52	0.81
	4MB	0.35	0.50	0.70

5.2 ストアミスバッファ方式の効果

図8は、ライト無効化方式に関して、ストロングオーダ方式(SO)、従来型ストアバッファを用いたウイークリィオーダ方式(WO)、及び提案したストアミスバッファ方式(SMB)のCPI(Cycle Per Instruction)を比較したものである。キャッシュ容量は64KBと1MBで比較した。ここで、キャッシュ容量64KBとは、命令キャッシュ64KB/データキャッシュ64KBを意味する。CPIは各容量において、ストロングオーダ方式を100%とした相対比率で示している。各CPIは、要因ごとに分類して示している。要因のなかで、ストアバッファ競合は、ストアバッファ溢れによる待ち時間や、後続のロードアクセスがス

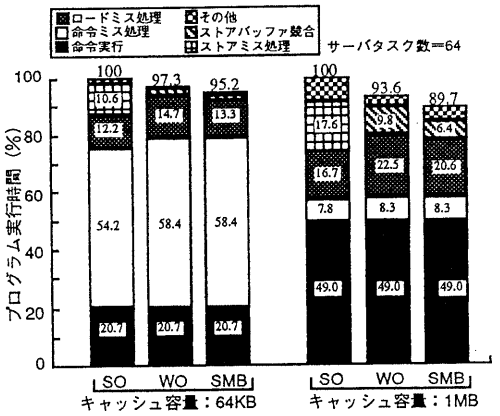

図8　ストアミスバッファ方式の効果

トアバッファとアドレス競合して待たされる時間を含む。"その他"には、データパージ処理の待ち時間や、キャッシュビジーに伴う待ち時間などが含まれる。

ストロングオーダ方式のなかで隠蔽可能な部分はストアミス処理とデータパージ処理待ちであり、キャッシュ容量64KBで全体の12.0%、1MBで22.5%を占める。この値は、ウイークリィオーダ方式による理想的な性能向上値となる。従来型ストアバッファを用いたウイークリィオーダ方式では、ストアミス処理時間を隠蔽できるが、命令ミス処理、ロードミス処理時間増加や、ストアバッファ競合増加のために全体としてはキャッシュ容量64KBで2.7%、1MBで6.4%程度の性能向上にとどまる。命令ミス・ロードミス処理時間の増加は、バックエンドで実行されるストアミス処理との競合に起因するものであり、ストアバッファ競合の増加は3．2章で述べたように、データアクセスのローカリティからストアミス処理の終了していないラインに後続ロードアクセスが発行され待たされるためである。

一方、ストアミスバッファ方式では、ストアミスのみをバッファに登録する効果や複数のストアミス処理を並列に処理できる効果により、ストアバッファ競合の低減が可能となる。キャッシュ容量1MBでは、従来型ストアバッファ方式に対して3.9%、ストロングオーダ方式に対して10.3%の性能向上が可能となる。しかし、キャッシュ容量64KBではほとんど性能向上率がみられない。これは、命令ミス処理の占める割合が多く、ストアミス処理を隠蔽した効果が見えにくいこと、またバスビジー率が約60%を越えるため複数のトランザクションを発行しても、効率良く処理されないことに起因する。

表5は、ロードアクセスがストアバッファ競合を起こす比率及びその平均待ち時間を、ウイークリィオーダ方式とストアミスバッファ方式で比較したものである。ストアミスバッファ方式では、ストアバッファの滞留エントリ数を低減できる効果から、ストアバッファ競合比率をキャッシュ容量64KBで33.3%、1MBで50%低減することが可能である。一方、平均待ちサイクルは約2割程度増加している。これはストアミスバッファでは、複数のトランザクションが発行されるため、バスビジー率が増加するためである。総合すると、競合比率低減の効果が大きく性能向上が実現できる。

表5　ライト無効化方式のストアバッファ競合

ライト処理方式	キャッシュ容量	ストアバッファ競合比率(Load当り)	平均待ちサイクル
Weakly Order方式	64KB	1.2%	37.8mc
	1MB	2.4%	31.9mc
ストアミスバッファ方式	64KB	0.8%	47.3mc
	1MB	1.2%	41.9mc

5．3 遅延ブロードキャスト方式の効果

図9は、ライト放送方式に関して、ストロングオーダ方式(SO)、従来型ストアバッファを用いたウイークリィオーダ方式(WO)、及び提案した遅延ブロードキャスト方式(DBC)のCPIを比較したものである。遅延ブロードキャスト方式では、ブロードキャストバッファ段数を4段（128バイト）で滞留許可エントリ数を2としたもの、及び段数16段（512バイト）で滞留許可エントリ数を8段としたものを示している。両者とも各エントリの滞留許可時間は500マシンサイクルとした。CPIの詳細分析でブロードキャスト処理が追加されており、これはSharedラインへのストアに伴うブロードキャスト処理が終了するまでの待ち時間を示す。

ストロングオーダ方式のなかで隠蔽可能な部分はストアミス処理、ブロードキャスト処理及びデータパージ処理待ちであり、キャッシュ容量64KBで全体の18.1%、1MBで37.7%を占める。従来型ストアバッファを用いたウイークリィオーダ方式では、ストアミス処理時間、ブロードキャスト処理時間を隠蔽できるが、命令ミス処理、ロードミス処理時間増加や、ストアバッファ競合増加のために全体としてはキャッシュ容量64KBで4.6%、1MBで

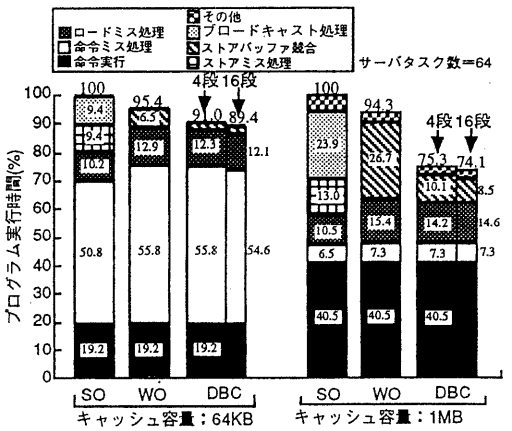

図9　遅延ブロードキャスト方式の効果

5.7%程度の性能向上にとどまる。一方、遅延ブロードキャスト方式では、４段のブロードキャストバッファにより、ストアバッファ競合を大幅に低減することが可能となり、ウイークリィオーダ方式に対して、キャッシュ容量64KBで4.4%、1MBで19.0%の性能向上が実現できる。これは、ブロードキャストバッファにより、複数のブロードキャストをまとめて処理できることから、ストアバッファに滞留するエントリ数を削減でき、後続ロードアクセスとの競合を避けることができるからである。ブロードキャストバッファが４段と１６段を比較しても性能向上はほとんど見られない。これは、ストアアクセスのローカリティから、比較的少ない段数で効果が十分得られるためと予想される。

表６には、ブロードキャストバッファのヒット率及びブロードキャストトランザクションの発生率を示した。ヒット率は４段で48.4-53.2%となり、小容量のバッファでブロードキャスト頻度を半減できることが明らかになった。これに伴って、ブロードキャストトランザクションの命令当たり発生率も約半減されている。表７には、ロードアクセスがストアバッファ競合を起こす比率及びその平均待ち時間を、ウイークリィオーダ方式と遅延ブロードキャスト方式で比較したものである。遅延ブロードキャスト方式では、ストアバッファの滞留エントリ数を低減できる効果から、ストアバッファ競合比率をキャッシュ容量64KBで29.5%、1MBで27.3%低減することが可能である。また、平均待ちサイクルも低減でき、大幅な性能向上が実現できている。

表６　ブロードキャストバッファの効果

ライト処理方式	BCバッファ段数	キャッシュ容量	BCバッファヒット率	BC発生率(命令当り)
Weakly Order方式		64KB		1.51%
		1MB		1.68%
遅延ブロードキャスト方式	4段	64KB	48.4%	0.76%
	(滞留許可2段)	1MB	53.2%	0.81%
	16段	64KB	50.7%	0.72%
	(滞留許可8段)	1MB	58.4%	0.70%

滞留許可時間:500mc　　BC: Broadcast

表７　ライト放送方式のストアバッファ競合

ライト処理方式	キャッシュ容量	ストアバッファ競合比率(Load当り)	平均待ちサイクル
Weakly Order方式	64KB	1.7%	55.4mc
	1MB	3.3%	61.2mc
遅延ブロードキャスト方式	64KB	1.2%	42.3mc
	1MB	2.4%	38.6mc

６．おわりに

ビジネス向けマルチプロセッササーバを対象に、ウイークリィオーダモデルを効率よく実装するキャッシュ一致保証方式の提案と性能評価を行った。代表的なキャッシュ制御方式であるライト無効化方式とライト放送方式それぞれについて考察した。ライト無効化方式では、ストアミスのみを登録するストアミスバッファにより、ロードミス処理とストアミス処理を並列処理し、且つ複数のストアミス処理をパイプライン処理する方式を提案した。また、ライト放送方式に関しては、ブロードキャスト処理を一定期間遅延させることによって、複数のブロードキャストをまとめて処理する遅延ブロードキャスト方式を提案した。

ＯＳ処理やタスクスイッチの影響も含めて評価が可能なトレースドリブンシミュレータを構築し、OLTPのベンチマークであるTPC-Bプログラムを用いて性能評価を行った。その結果、ストアミスバッファ方式により、ストロングオーダ方式に比べて4.8-10.3%の性能向上が可能であることを明らかにした。性能向上の主な要因は、ストアミスのみを登録する効果と、複数のストアミス処理をパイプライン処理する効果により、ストアバッファでのアドレス競合を低減できるためである。また、遅延ブロードキャスト方式では、ブロードキャスト頻度を半減でき、9.0-24.7%の性能向上が可能であることを明らかにした。

参考文献

1) Chan, K. at al.: *Multiprocessor features of the HP Corporate Business Servers, Proc. of COMPCON93*, pp. 330-337 (1993).
2) Ceklcov, M. at al.: *SPARCcenter2000: Multiprocessing for the 90's!, Proc. of COMPCON93*, pp. 345-353 (1993).
3) Allison, B.: *DEC 7000/10000 Model 600 AXP Multiprocessor Server, Proc. of COMPCON93*, pp. 456-464 (1993).
4) Dubois, M. at al.: *Memory Access Buffering in Multiprocessors, Proc. of 13th ISCA*, pp. 434-442 (1986).
5) Adve, S. V. and Hill, M. D.: *Weak Ordering - A New Definition, Proc. of 17th ISCA*, pp. 2-14 (1990).
6) Kroft, D.: *Lockup-Free Instruction Fetch/Prefetch Cache Organization, Proc. of 8th ISCA*, pp. 81-87 (1981).
7) Sohi, G. S. and Franklin, M.: *High-Bandwidth Data Memory Systems for Superscalar Processors, Proc. of ASPLOS-IV*, pp. 53-62 (1991).
8) McGrory, J. J. et al.: *Transaction Processing Performance on PA-RISC Commercial Unix Systems, Proc. of COMPCON92*, pp. 199-206 (1992).
9) Gharachorloo, K. at al.: *Performance Evaluation of Memory Consistency Models for Shared-Memory Multiprocessors, Proc. of ASPLOS-IV*, pp. 245-257 (1991).
10) 永尚哉, 村上和彰: メモリ・コンシステンシ・モデル―新しいモデルの提案、および、その能力比較―, 並列処理シンポジウム'93 論文集, pp. 253-260 (1993).
11) Sweazey, P. and Smith, A. J.: *A Class of Compatible Cache Consistency Protocols and their Support by the IEEE Futurebus, Proc. of 13th ISCA*, pp. 414-423 (1986).
12) Lenoski, D. et al.: *The DASH Prototype: Implementation and Performance, Proc. of 19th ISCA*, pp. 92-103 (1992).
13) Torrellas, J., Gupta, A. at al.: *Characterizing the Caching and Synchronization Performance of a Multiprocessor Operating System, Proc. of ASPLOS-V*, pp. 162-174 (1992).
14) TPC Benchmark A, Transaction Processing Performance Council (TPC), (1991).
15) 高崎繁夫ほか: PA-RISC用ステップ解析システムの開発, 第４７回情報処理学会全国大会論文集, pp. 6-33 (1993).