計算機アーキテクチャ 142−29
ハイパフォーマンス
コンピューティング 85−29
（2001. 3. 9）

DSMシステムにおける投機的コヒーレンス制御機構の提案と評価

古川文人† 多田野陽介† 乗貞由華†
大津金光† 馬場敬信†

本稿では，ハードウェアによるCache Coherent DSMシステムにおいて，新たな投機的コヒーレンス制御機構を提案する．本機構の目的は，readとwriteの両方の種類のメモリアクセスにより発生するコヒーレンス制御のオーバーヘッドを削減することである．本機構を備えたDSMシステムを評価した結果，従来の投機的コヒーレンス制御機構を備えたものに対して最大1.97倍の性能向上を得られることがわかった．

A Speculative Coherence Control Mechanism for DSM Systems

FUMIHITO FURUKAWA,† YOUSUKE TADANO,† YUKA NORISADA,†
KANEMITSU OOTSU† and TAKANOBU BABA†

In this paper, we propose a new speculative coherence control mechanism for hardware DSM systems. Our purpose is to reduce the overhead of coherence control caused by the both memory accesses of read and write. We have evaluated the performance of the DSM system that equips our mechanism. As a result, we have found that the DSM system can achieve up to 1.97 times better performance than that of the DSM system that equips the conventional speculative coherence control mechanism.

1. はじめに

分散共有メモリ（以下，DSM）は，1)ハードウェア構成の優れたスケーラビリティ，2)プログラミングが容易な共有メモリモデルの2つの特徴を持つメモリアーキテクチャである．このため，メモリアーキテクチャにDSMを採用した並列システム（以下，DSMシステム）の研究が数多く行われている．[1)]

DSMシステムでは，リモートメモリへのアクセス（以下，リモートアクセス）が相互結合網を介した通信処理で実現される．このため，リモートアクセスが発行されてから完了するまでのレイテンシ（以下，リモートレイテンシ）はローカルメモリへのものと比較して格段に長い．そこで，現在では各ノードでリモートメモリ上のデータをキャッシュし，かつ各キャッシュ間でのデータの一貫性をディレクトリ法により維持するDSMシステム（以下，CC-DSMシステム）が提供されるようになった．しかし，真に共有しているデータへのアクセスでは相互結合網を介した通信は本質的に避けられず，リモートレイテンシはローカルメモリ上の非共有データへのものに対して3〜9倍程度[2),3)]と長い．これは，ノード内外の間に著しい動作速度の差が予測される将来ではより深刻な問題となる．この問題解決へのアプローチは，リモートレイテンシの隠蔽と削減の2つに大別できる．

リモートレイテンシの隠蔽手法として．緩和されたメモリコンシステンシモデル[4)]によるメモリアクセスのバッファリングおよびパイプライン処理がある．また，マルチスレッド処理[5),6)]による演算と通信処理のオーバーラップが研究されている．これらの手法ではユーザへのAPIの提供とそれを用いた特別なプログラミングを要求し，それをサポートするコンパイラまたはランタイムライブラリの開発が必要となる．

一方，リモートレイテンシの削減手法として，ソフトウェアまたはハードウェアによるキャッシュラインのプリフェッチがある．しかし，いずれの実現方法においてもプログラムにおいてアクセスが可能になる時点（以下，同期点）より以前にプリフェッチを行うことはできない．そこで，ハードウェアによるCC-DSMシステムにおいて，同期点以前にコヒーレンス制御を行う投機的コヒーレンス制御に関する研究[7)〜11)]が行われている．この手法の目的は，コヒーレンス制御のオーバーヘッド（以下，コヒーレンスオーバーヘッド）を軽減することで，リモートレイテンシを削減することである．投機的コヒーレンス制御は，データの共有状態を動的に予測し，その予測情報を基に行われる．予測のための処理と投機的コヒーレンス制御はユーザには完全に透過な専用ハードウェアが行う．したがって，ユーザに提供するプログラミング環境への影響は

† 宇都宮大学工学部
Faculty of Engineering, Utsunomiya University

全くない．

本稿では，ハードウェアによるCC-DSMシステムにおいて，新たな投機的コヒーレンス制御機構を提案する．この機構の目的は，readとwriteの両方の種類のメモリアクセスにより発生するコヒーレンスオーバーヘッドを完全に除去することである．なお，以降ではハードウェアによるCC-DSMシステムのことを単にDSMシステムと呼ぶ．

以下，2章では，本研究の背景と新規性を述べる．3章では，提案する投機的コヒーレンス制御機構について述べる．4章では，評価を行い，5章でまとめる．

2. 研究背景

投機的コヒーレンス制御によってコヒーレンスオーバーヘッドの軽減を目指す研究は，それらが対象にするメモリアクセスの種類で大きく分類することができる．本章では，これらの研究を分類して概要を述べたあと，本稿で提案する新たな投機的コヒーレンス制御機構の新規性を示す．なお，以降ではメモリアクセスを発行したプロセッサを持つノードをマスタ，マスタの要求したアドレスと対応するメモリを持つノードをホーム，キャッシュにそのコピーを持つノードをスレーブと呼ぶ．また，マスタ内のプロセッサが発行したメモリアクセスにおいて，readアクセスがミスヒットしたときに起きるホームへの要求をread要求，writeアクセスがミスヒットしたときに起きるホームへの要求をread-ex要求，read要求により既に保持しているコピーへwriteアクセスしたときに起きるホームへの要求をupgrade要求と呼ぶ．

readアクセスのみを対象にした投機的コヒーレンス制御に関する研究として，Self-Cleanup Cache[7]（以下，SCC）とSpeculative Write-Invalidation[8]（以下，SWI）が挙げられる．SCCを持つノードで構成したDSMシステムでは，ホームへ書き戻すスレーブ上のダーティなキャッシュライン（以下，ライン）と書き戻すタイミングをスレーブ自身が予測し，予測したラインをホームへ投機的に書き戻す．これにより，マスタのread要求がホームに到着する以前に，この投機的な書き戻しが完了している場合，そのread要求のコヒーレンス処理内で書き戻しに要する時間が完全に除去される．また，SWIを行うDSMシステムでは，ホームへ書き戻すスレーブ上のラインとそのタイミングをホームが予測し，スレーブに対して投機的に書き戻しの要求をする．さらに，この書き戻しが完了した後，ホームはそのラインを参照するマスタを予測し，投機的に送信する．これにより，マスタがホームへread要求をする以前に，この投機的なホームへの書き戻しとマスタへの送信が完了している場合，そのread要求の全てのコヒーレンス処理に要する時間が完全に除去される．

readとwriteの両方の種類のメモリアクセスを対象にした投機的コヒーレンス制御に関する研究として，Self-Invalidation[9]（以下，SI）とメモリアクセス命令の履歴を基にした予測による機構[10]（以下，IBP）が挙げられる．SIは，スレーブ上の無効化するラインとそのタイミングをスレーブ自身が予測して無効化する．また，予測したラインがダーティな状態であるときは，ホームへ投機的に書き戻した後で無効化する．これにより，マスタのread要求，read-ex要求，upgrade要求がホームに到着する以前に，この投機的な無効化およびホームへの書き戻しが完了している場合，その要求のコヒーレンス処理内で無効化および書き戻しに要する時間が完全に除去される．また，IBPは，producer-consumer sharingとmigratory sharingの2つの共有状態（以下，それぞれPC共有状態，M共有状態）を予測することにより投機的コヒーレンス制御を行っている．PC共有状態にあるデータはproducerであるノード（以下，生産者）が検出し，生産者がconsumerであるノード（以下，消費者）を予測する．生産者は共有データを更新した後，予測した消費者へ投機的に対応するラインを送信する．これにより，マスタが消費者，スレーブが生産者のとき，SWIと同様にコヒーレンス処理に要する時間が完全に除去される場合がある．すなわち，readアクセスを対象にしている．また，M共有状態にあるデータへのアクセスはマスタが予測し，マスタ内のプロセッサがそのデータへのread アクセスを発行したとき，ホームに対してread要求ではなくread-ex要求を行う．これにより，readアクセスに続いてwriteアクセスを行った場合，writeアクセスによるコヒーレンス処理が不要になる．

ここで，SCC，SWI，SI，IBPが対象にしているメモリアクセスについて以下の4つに分類できる．なお，以降ではPC共有状態にあるデータをPC共有データ，M共有状態にあるデータをM共有データと呼ぶ．

(1) PC共有データへの消費者によるreadアクセス
(2) PC共有データへの生産者によるwriteアクセス
(3) M共有データへのreadアクセス
(4) M共有データへのwriteアクセス

(1)のSWIおよびIBPによるサポート，(3)のSWIによるサポート，(4)のIBPによるサポートは，それらのメモリアクセスのコヒーレンスオーバーヘッドを完全に除去できる可能性がある．しかし，(2)のコヒーレンスオーバーヘッドを完全に除去できるサポートは無い．以上をまとめると，readアクセスのコヒーレンスオーバーヘッドの完全な除去ができる投機的コヒーレンス制御機構としてSWIが存在する．しかし，readとwriteの両方の種類のメモリアクセスを対象とし，それらのコヒーレンスオーバーヘッドの完全な除去を目指した研究は存在しない．

本稿で提案する投機的コヒーレンス制御機構は，readとwriteの両方の種類のメモリアクセスにより発生するコヒーレンスオーバーヘッドの完全な除去を目指す

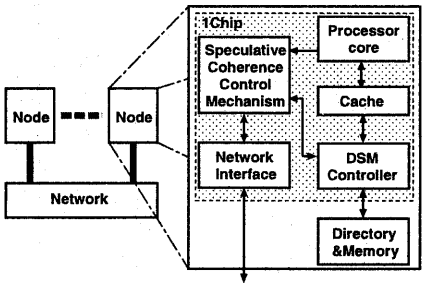

図1　DSMシステムとノード内の構成

ものであり，この点で上述した既存の投機的コヒーレンス制御機構とは異なる新規なものである．

3. 投機的コヒーレンス制御機構

本章では，3.1節で基盤とするDSMシステムとキャッシュコヒーレンスプロトコルについて述べ，3.2節以降で提案する投機的コヒーレンス制御機構(以下，SCCM(Speculative Coherence Contorol Mechanism))について述べる．

3.1 基盤とするCC-DSMシステム

本稿では，以下に示すDSMシステムとキャッシュコヒーレンスプロトコルを前提として議論する．

3.1.1 DSMシステムとノード内の構成

DSMシステムとノード内の構成を図1に示す．このDSMシステムは，複数のノードがネットワークで結合されたものであり，そのネットワークは，2ノード間のメッセージの到着順序を保証するものである．各ノードは，1台のプロセッサコア，キャッシュ，DSMコントローラ，ネットワークインターフェース（以下，NI），およびSCCMを集積するチップとディレクトリおよびメモリから構成される．また，キャッシュはライトバック型，ディレクトリはフルマップ形式である．

3.1.2 キャッシュコヒーレンスプロトコル

キャッシュコヒーレンスプロトコルは無効化型である．また，キャッシュラインとメモリブロックは1対1に対応しており，それらの状態遷移はそれぞれ図2と図3に示される．なお，図2と図3でのX/Yは，Xが要求，Yが遷移後の応答を意味する．

キャッシュラインの状態は，Modified，Exclusive，Shared，Invalid（以下，それぞれM，E，S^c，I）の4状態をとる．Mは，メモリブロックのコピーを専有し，その内容がメモリと一致していないことを示す．Eは，メモリブロックのコピーを専有し，その内容がメモリと一致していることを示す．S^cは，他のノードに同一内容のラインがあることを示す．Iは，そのラインの内容が無効であることを示す．

メモリブロックの状態は，Uncached，Shared，Private，Busy-Shared，Busy-Private（以下，それぞれU，S^m，P，BS，BP）の5状態をとる．Uは，システム内にそのメモリブロックのコピーが無いことを示す．S^mは，1個以上のノードのキャッシュがそのメモリブロックのコピーを保持し，その内容がメモリと一致していることを示す．Pは，1つのノードのキャッシュがそのメモリブロックのコピーを専有し，その内容がメモリと一致しない可能性があることを示す．BS，BPは，それぞれS^m，Pへ遷移するための処理を行っている最中であることを示す．BS，BPのメモリブロックへの要求があった場合，それを管理するDSMコントローラはその要求を全て保存し，その状態がそれぞれS^m，Pへ遷移した後で要求を受けた順に処理する．

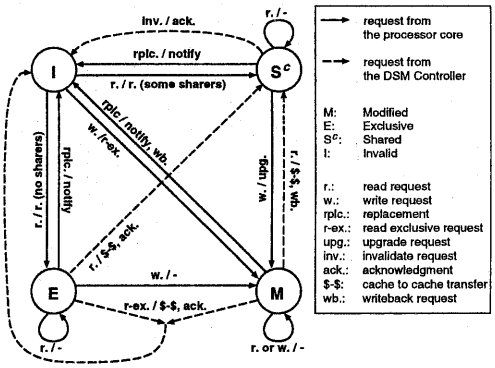

図2　キャッシュラインの状態遷移

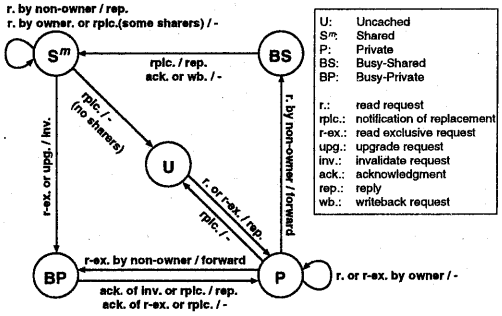

図3　メモリブロックの状態遷移

3.2 投機的コヒーレンス処理

SCCMは，仮定したキャッシュコヒーレンスプロトコルを用いて以下の5つの投機的コヒーレンス処理を行う．

(1) speculative self-downgrade
(2) speculative self-invalidation
(3) speculative send block in shared state
(4) speculative send block in exclusive state
(5) speculative upgrade

speculative self-downgrade（以下，spec.self-dwg.）は，PC共有データへのreadアクセスによって発生するコヒーレンスオーバーヘッドを軽減するための処理である．spec.self-dwg.は，EまたはMの状態のコ

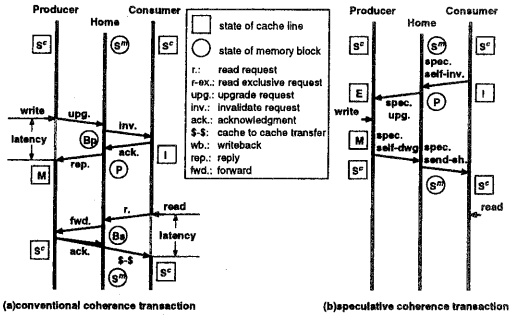

図4　PC共有状態における投機的コヒーレンス処理の例

ピーを既に保持するノードが自身のキャッシュに対して発行し，そのコピーの状態をS^cに遷移させる．その後，コピーの状態がS^cに遷移したことをホームに対して通知する．また，保持しているコピーの状態がMのときは，その内容をホームに対して書き戻す．

speculative self-invalidation（以下，spec.self-inv.）は，PC共有データへのwriteアクセス，およびM共有データへのreadとwriteの両方のメモリアクセスによって発生するコヒーレンスオーバーヘッドを軽減するための処理である．spec.self-inv.は，コピーを既に保持するノードが自身のキャッシュに対して発行し，そのコピーを無効化する．その後，コピーを無効化したことをホームに対して通知する．また，保持していたコピーの状態がMのときは，その内容をホームに対して書き戻す．

speculative send block in shared state（以下，spec.send-sh.）は，PC共有データへのreadアクセスによって発生するコヒーレンスオーバーヘッドを軽減するための処理である．spec.send-sh.は，ホームにおいてS^mの状態のメモリブロックに対してread要求を発行するノードを予測し，そのノードに対してS^cの状態のコピーを送信する．それと同時に，予測されたノードがS^cの状態のコピーを保持していることをDSMコントローラに対して通知する必要がある．

speculative send block in exclusive state（以下，spec.send-ex.）は，M共有データへのreadおよびwriteアクセスによって発生するコヒーレンスオーバーヘッドを軽減するための処理である．spec.send-ex.は，ホームにおいてUの状態のメモリブロックに対してread要求に引き続くread-ex要求を発行するノードを予測し，そのノードに対してEの状態のコピーを送信する．それと同時に，予測されたノードがコピーを専有し，その内容がメモリブロックと一致しない可能性があることをDSMコントローラに対して通知する．

speculative upgrade（以下，spec.upg.）は，PC共有データおよびM共有データへのwriteアクセスによって発生するコヒーレンスオーバーヘッドを軽減するための処理である．spec.upg.は，ホームにおいて

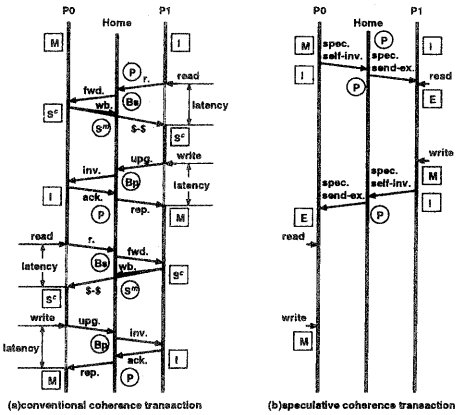

図5　M共有状態における投機的コヒーレンス処理の例

S^mの状態のメモリブロックに対してupgrade要求を発行するノードをS^cの状態のコピーを持つノードの中から予測し，そのノードに対してEの状態への遷移を許可したことを通知する．それと同時に，予測されたノードがコピーを専有し，その内容がメモリブロックと一致しない可能性があることをDSMコントローラに対して通知する．

図4，図5は，コヒーレンス処理の流れを示したものである．図4(a)，図5(a)は，それぞれPC共有データおよびM共有データへのメモリアクセスを従来のコヒーレンス処理で解決する実行の例である．また，図4(b)，図5(b)は，いずれも上記の投機的コヒーレンス処理により，コヒーレンスオーバーヘッドを完全に除去できた実行の例である．

3.3 SCCMの概要

SCCMの構成を図6に示す．SCCMは，SDIG (Speculative self-Downgrade and self-Invalidation Generator)，SSUG(Speculative Send block and Upgrade Generator)，SRC(Speculative Remote Cache)およびSCoC(Speculative Coherence Controller)から構成される．

3.3.1 SDIG

SDIGは，spec.self-dwg.およびspec.self-inv.の発行のタイミングと対象となるラインのアドレスを予測し，SCoCに対してそれを伝える．この予測器として SCM[7]およびLTP[9]を応用することができる．本稿では，LTPの応用を想定している．LTPは，spec.self-dwg.もしくはspec.self-inv.のどちらか一方に関する予測器である．例えば，spec.self-inv.に関する予測は，無効化要求とそのラインへのメモリアクセス命令のトレース（以下，アクセストレース）を観測することで行う．また，予測の信頼性を確保するために，予測の成功/失敗を知らせる情報（以下，予測検証情報）から予測を行うか否かの判断を行う．LTPは，無効化要求のあったラインのアドレスをインデクスとして，そのラインに対する前回の無効化の直後から現在の無効化

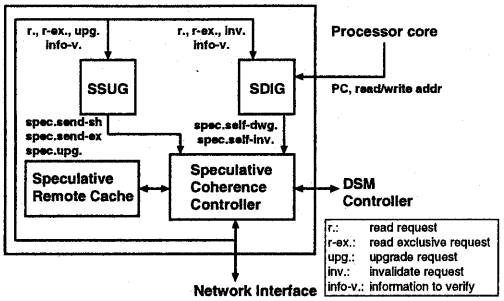

図6　SCCMの構成

の直前までのアクセストレースを履歴として蓄積する．この履歴と現在観測しているアクセストレースが一致したときをそのラインに対してspec.self-inv.を発行するタイミングと予測する．SDIGは，spec.self-dwg.とspec.self-inv.の両方に関して予測するため，履歴の各エントリにそのどちらかを識別するビットを持つ．

3.3.2 SSUG

SSUGは，spec.send-sh.，spec.send-ex.およびspec.-upg.の発行のタイミングと対象となるメモリブロックのアドレスを予測し，SCoCに対してそれを伝える．この予測器にはCosmos[11]およびMSP[8]を利用できる．本稿では，MSPの利用を想定している．MSPは，自ノードのメモリブロックに対する要求を監視し，そのアドレスをインデクスとして，要求の種類と発行元のノードの識別子を履歴として蓄積する．現在観測した要求とその対象であるアドレスからこの履歴を検索し，次に観測している要求の種類と発行元のノードの識別子を予測結果として出力する．SSUGは，この予測結果の要求の種類がread要求，read-ex要求，upgrade要求のときを，それぞれ対応するノードに対してspec.send-sh.，spec.send-ex.，spec.upg.を発行するタイミングと予測する．また，SDIGと同様に，予測検証情報を予測結果に反映する．

3.3.3 SRC

SRCは，spec.send-sh.，spec.send-ex.によりホームから送信されたコピーを適切な状態で保持するキャッシュである．また，SRCは各コピーの参照の有無を管理し，そのコピーが無効化されたとき，予測検証情報としてホームに対して送信する．

3.3.4 SCoC

SCoCは，DSMコントローラに対して仮想的なNIとして動作し，SDIG，SSUGが予測した投機的コヒーレンス処理をDSMコントローラのサポートするプロトコルを用いて発行する．以下に，各投機的コヒーレンス処理におけるSCoCの動作を示す．

- spec.self-dwg.，spec.self-inv.
  自ノードのラインに対してそれぞれread要求，無効化要求を発行し，その応答をそれぞれspec.self-dwg.，spec.self-inv.である情報を付加してホームへ転送する．これを受信したホームのSCoCは，そのラインの置換の通知としてDSMコントローラに伝える．また，このホームのSCoCは，spec.self-dwgおよびspec.self-inv.が発行されたメモリブロックに関して，そのタイミングが早すぎたか否かの情報を管理し，予測検証情報としての発行元に送信する．
- spec.send-sh.，spec.send-ex.
  自ノードのメモリブロックに対して仮のノードからの要求としてそれぞれread要求，read-ex要求を発行し，その応答をそれぞれspec.send-sh.，spec.send-ex.である情報を付加して予測されたノードへ転送する．これを受信したノードのS-CoCは，そのコピーをSRCに保存する．
- spec.upg.
  自ノードのメモリブロックに対して予測されたノードからの要求としてupgrade要求を発行する．

また，SCoCはDSMコントローラからのリモートノードへのread要求およびread-ex 要求を観測した場合，SRCが要求しているデータを既に保持しているかをチェックする．もし，保持していれば，そのデータを応答としてDSMコントローラに返し，保持していなければ，その要求をNIへ転送する．

表1　シミュレーションパラメータ

ノード数	16
プロセッサコア	4-way スーパースカラ
キャッシュ	
ラインサイズ	64B
命令キャッシュ	完全ヒット
データキャッシュ	
サイズ	無制限
アクセスレイテンシ	1MC
最大未解決ミス数	8
メモリアクセスレイテンシ	40MC
ネットワーク	
メッセージヘッダサイズ	16B
トポロジ	2D-メッシュ
リンク幅	8B
1ホップ	4MC

4. 評　価

本章では，readアクセスのみをサポートするSWIと比べて，readとwriteの両方をサポートするSCCMの性能向上がどの程度かを実験により明らかにする．

4.1 評価方法

評価は，表1に示すDSMシステムのモデル上でSPLASH-2[12]のFFTの一部をCC-NUMA型並列計算機シミュレータRSIM[13]でシミュレーションすることにより行った．

シミュレーションを行った部分は，FFT内の転置行列を作成するフェーズ（以下，転置行列作成フェーズ）である．このフェーズ内の大部分の処理は，PC共有データに対するreadおよびwriteアクセスである．こ

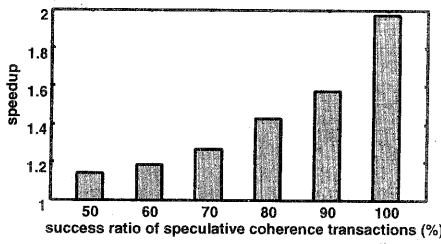

図7　FFT内の転置行列作成フェーズの性能向上比

のフェーズをSWIおよびSCCMでサポートしたときのそれぞれの実行時間を投機的コヒーレンス処理の成功率を50%から100%へ変化させて計測した．なお，投機的コヒーレンス処理の成功率pは，以下の式で与えられる値である．

$$p = \frac{a-b}{a} \times 100 \quad (\%)$$

ここで，aは投機的コヒーレンス制御をしない場合にコヒーレンス処理を必要としたメモリアクセスの総数，bは投機的コヒーレンス制御をした場合にコヒーレンス処理を必要としたメモリアクセスの総数である．

4.2 評価結果

図7に，転置行列作成フェーズのSCCMによる性能向上比を示す．この性能向上比は、SWIによる投機的コヒーレンス制御をした場合の実行速度を基準とした．この結果から，投機的コヒーレンス制御の成功率が50%以上の場合，SCCMを備えたDSMシステムがSWIを備えたものに対して1.14～1.97倍の性能向上を得られることがわかる．

5. まとめ

本稿では，ハードウェアによるCC-DSMシステムにおいて，新たな投機的コヒーレンス制御機構SCCMを提案した．また，本機構を備えたDSMシステムを評価した結果，SCCMを備えたDSMシステムがSWIを備えたものに対して1.14～1.97倍の性能向上を得られることがわかった．

今後は，提案した機構の詳細なシミュレーション環境を構築し，プログラム全体の実行を通して性能評価を行う予定である．

謝辞　本研究は，一部文部省科学研究費基盤研究(B)課題番号10558039，基盤研究(C)課題番号12680328の援助によるものである．

参考文献

1) Keleher, P. J.: Distributed Shared Memory Home Pages. `http://www.cs.umd.edu/~keleher/dsm.html`.
2) 細見岳生, 加納健, 中村真章, 広瀬哲也, 中田登志之: 並列計算機Cenju-4の分散共有メモリ機構, 並列処理シンポジウムJSPP'99論文集, pp. 15–22 (1999).
3) Laudon, J. and Lenoski, D.: The SGI Origin: A ccNUMA Highly Scalable Server, *Proceedings of the 24th Annual International Symposium on Computer Architecture* (1997).
4) Adve, S. V. and Gharachorloo, K.: Shared Memory Consistency Models: A Tutorial, Technical Report 9512, Rice University ECE (1995).
5) Fillo, M., Keckler, S. W., Dally, W. J., Carter, N. P., Chang, A., Gurevich, Y. and Lee, W. S.: The M-Machine multicomputer, *Proceedings of the 28th Annual International Symposium on Microarchitecture*, pp. 146–156 (1995).
6) 佐藤三久, 児玉祐悦, 坂井修一, 山口喜教: 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).
7) 森眞一郎, 福島直人, 五島正裕, 中島浩, 富田眞治: Self-Cleanup Cacheの提案, 情報処理学会論文誌, Vol. 38, No. 2, pp. 321–331 (1997).
8) Lai, A. and Falsafi, B.: Memory Sharing Predictor: The Key to a Speculative Coherent DSM, *Proceedings of the 26th Annual International Symposium on Computer Architecture* (1999).
9) Lai, A. and Falsafi, B.: Selective, Accurate and Timely Self-Invalidation Using Last-Touch Prediction, *Proceedings of the 27th Annual International Symposium on Computer Architecture* (2000).
10) Kaxiras, S. and Goodman, J. R.: Improving CC-NUMA Performance Using Instruction-Based Prediction, *Proceedings of the Fifth International Symposium on High-Performance Computer Architecture*, pp. 161–170 (1999).
11) Mukherjee, S.S. and Hill, M.D.: Using Prediction to Accelerate Coherence Protocols, *Proceedings of the 25th Annual International Symposium on Computer Architecture*, pp. 179–190 (1998).
12) Woo, S. C., Ohara, M., Torrie, E., Singh, J. P. and Gupta, A.: The SPLASH-2 Programs: Characterization and Methodological Considerations, *Proceedings of the 22th Annual International Symposium on Computer Architecture*, pp. 24–36 (1995).
13) Pai, V. S., Ranganathan, P. and Adve, S. V.: RSIM: An Execution-Driven Simulator for ILP-Based Shared-Memory Multiprocessors and Uniprocessors, *Proceedings of the Third Workshop on Computer Architecture Education* (1997).