2ZB−04

DSMシステムにおける投機的コヒーレンス制御機構のSPLASH-2による性能評価*

多田野 陽介　古川 文人　乗貞 由華　門脇 健司
鈴木 圭介　大津 金光　横田 隆史　馬場 敬信 †
宇都宮大学工学部情報工学科‡

1 はじめに

分散共有メモリ(以下, DSM)システムにおいて, 相互結合網を介した通信により実現するリモートメモリへのアクセスレイテンシ(以下, リモートレイテンシ)はローカルのものに対して格段に長く, 問題となっている.

この問題解決のアプローチの一つとして, ハードウェアによるCache Coherent DSMシステム(以下, CC-DSMシステム)において, 投機的コヒーレンス制御により, 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)を軽減し, リモートレイテンシを短縮する方法がある.

我々は, CC-DSMシステムにおいてwriteとreadの両方のリモートレイテンシを短縮することを目的とした, 投機的コヒーレンス制御機構[1]を提案してきた. 本稿では, SPLASH-2を用いて本機構のシミュレーションを行うことにより性能評価を行う.

2 投機的コヒーレンス制御機構

2.1 DSMシステムのモデル

図1に我々が提案しているDSMシステムの構成を示す. 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, 各ノードは, プロセッサコア, キャッシュ, DSMコントローラ, ディレクトリおよびメモリ, ネットワークインターフェース, および投機的コヒーレンス機構(以下, **SCCM**(Speculative Coherence Contorol Mechanism))から構成される.

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, キャッシュはModified, Exclusive, Shared, Invalid(以下, M, E, Sc, I)の4状態をとり, メモリブロックは, Uncached, Shared, Private, Busy-Shared, Busy-Private(以下, U, Sm, P, BS, BP)の5状態をとる.

2.2 SCCMによる投機的コヒーレンス処理

SCCMは, 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-invalidate
(3)speculative send in shared state

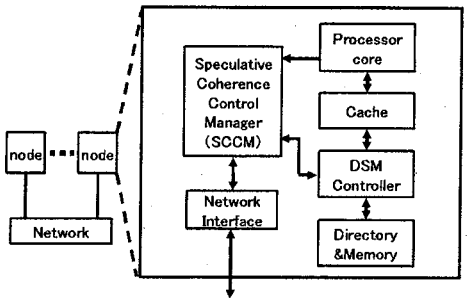
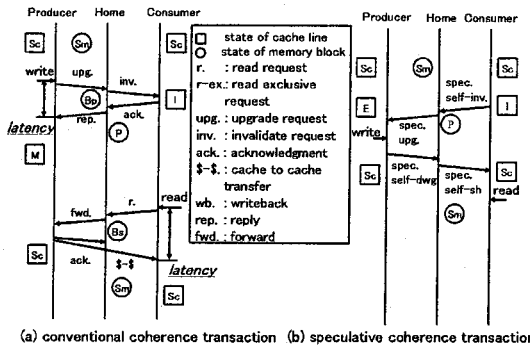


図 1: DSMシステムとノード内の構成



(a) conventional coherence transaction (b) speculative coherence transaction

図 2: PC共有状態における投機的コヒーレンス処理の例

(4)speculative send in exclusive state
(5)speculative upgrade

これらの処理は, producer-consumer sharing(以下, PC共有状態)とmigratory sharingの2つの共有状態にあるデータへのメモリアクセスにおいて, readおよびwrite両方のメモリアクセスに要するコヒーレンス処理を投機的に行うものである.

SCCM内では, **MSP**[2](Memory Sharing Predictor)と**LTP**[3](Last-Touch Predictor)をそれぞれ独自に拡張することにより, 投機的コヒーレンス処理の種類, 開始のタイミングを予測する.

図2に, PC共有状態における投機的コヒーレンス処理の例を示す. PC共有状態とはあるメモリブロックをProducerノードがwriteを行い, Consumerノードがreadを

* Performance Evaluation of the Speculative Coherence Contoroller for a DSM system using SPLASH-2
† Yousuke Tadano, Fumihito Furukawa, Yuka Norisada, Kenji Kadowaki, Keisuke Suzuki, Kanemitsu Ootsu, Takasi Yokota, and Takanobu Baba
‡ Department of Information Science, Faculty of Engineering, Utsunomiya University

表 1: 評価パラメータ

ノード数			16台
プロセッサコア			4命令同時発行 スーパースカラ
データキャッシュ(ノンブロッキング)			
サイズ (Kバイト)	L1		64
	L2		512
アクセス レイテンシ (サイクル)	L1		1
	L2	タグ	2
		データ	3
メモリアクセスレイテンシ(サイクル)			40
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 (サイクル)	データなし		20
	データ付き		42

行う共有パターンである.

図2(a)は, PC共有状態にあるデータ(以下, PC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例である. この例では, Producerによるwriteアクセスが開始された後, Consumerの所有するラインの無効化が始めて開始される. また, Consumerによるreadアクセスが開始された後, Producerへのread要求が初めて開始される.

一方で, 図2(b)は, 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. この例では, Producerによるwriteアクセスが開始される以前に, Consumerが該当するラインを無効化し, ホームがProducerの保持するラインをEに遷移させている. また, Consumerによるreadアクセスが開始される以前に, Producerがラインの状態をScに遷移させ, ホームがConsumerにコピーを送信し, Consumerがその受信を完了している. このため, メモリアクセスの開始から完了までの間に, コヒーレンス処理のためのレイテンシは存在しない. すなわち, この投機的コヒーレンス処理の例では, コヒーレンスオーバーヘッドが完全に除去される.

3 評価

3.1 評価方法

評価の目的は, 投機を全く行わない場合と比べて, readとwriteを投機の対象とするSCCMによる性能向上を実験により明らかにすることである.

本稿では, SPLASH-2のbarnesをCC-NUMA型並列計算機シミュレータRSIMと, SCCMを実装したRSIM上においてシミュレーションすることにより評価を行った. 評価パラメータを表1に示す.

3.2 評価結果

図3にSCCMでの, barnesの実行時における性能向上比及び, readアクセス, writeアクセスにおける平均メモリアクセスレイテンシを示す. 性能向上比は, 投機的コヒーレンス制御を全く行わない場合の実行速度を基準とした. また, 平均メモリアクセスレイテンシは, リモートとローカル全てのメモリアクセスレイテンシの平均としている. この結果から, SCCMを備えたDSMシステムが投機的コヒーレンス制御を全く行わないものに対し, 性能向上比は0.91倍となった. また, この理由として, 投機的コヒーレンス制御を全く行わないものに比べ, SCCMを備えたDSMシステムではread, writeの平均メモリアクセスレイテンシがそれぞれ111%, 119%要することがわかった.

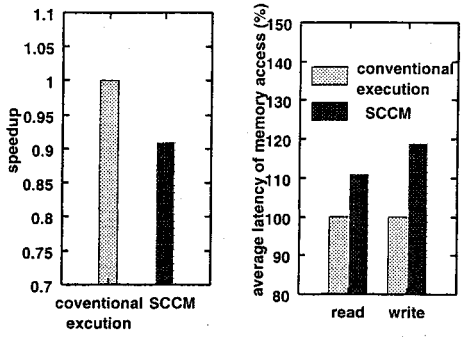


図 3: SCCMによる性能向上比と平均メモリアクセスレイテンシ

4 おわりに

本論文では, readとwriteアクセスを投機の対象とする投機的コヒーレンス制御機構SCCMを備えたCC-DSMシステムを評価した.

その結果, 従来のコヒーレンス制御に対して, 性能向上比は0.91倍になることがわかった.

今後の課題は, 問題サイズを大きくしてシミュレーションすることと, プロセッサの台数を増やして評価を行うことである.

謝辞 本研究は, 一部日本学術振興会科学研究費補助金(基盤研究(C)課題番号12680328)の援助による.

参考文献

[1] 古川文人多田野陽介, 乗貞由華, 大津金光, 馬場敬信: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, 情報処理学会研究報告 2001-ARC-142 2001-HPC-85, pp. 169-174(2001).

[2] 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)

[3] 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)