オーバヘッドの少ないキャッシュコヒーレンスプロトコルの提案

4K－3

細見 岳生　加納 健　丸山 勉
日本電気（株）C&C 研究所

1 はじめに

分散共有メモリ型並列計算機において，メモリアクセスレイテンシの削減/隠蔽は必須である．とりわけ，隠蔽が困難なロードミス時のレイテンシを削減することが重要となっている．

本稿では，ロードミス時のレイテンシの削減を目的としたキャッシュコヒーレンスプロトコル *"Write-Nth"* の提案を行う．

2 従来のコヒーレンスプロトコルの問題点

図1を参照しながら，従来のインバリデート型 [1] のコヒーレンスプロトコルの動作の概要について述べ，その問題点を明らかにする．

今，複数のプロセッサで共有されているブロックに対してストアが行われると，まずメモリに対して書き込み要求が発行される（Write Req. メッセージ）．書き込み要求を受けたメモリは，ブロックを保持する他の全てのキャッシュに対して無効化要求を発行する（Invalidate メッセージ）．無効化要求を受けたキャッシュは，当該ブロックを無効化し，無効化が完了した旨をメモリに通知する（Ack メッセージ）．その後，メモリはキャッシュに対して処理完了を通知する（Write Ack メッセージ）．この処理の結果，最新のブロックは Store が行われたキャッシュにのみ存在する状態となる．

次に，別のキャッシュでロードが行われると，最新のブロックはそのキャッシュに存在しないので，メモリに対して読み出し要求が発行される（Read Req. メッセージ）．読み出し要求を受けたメモリは最新のブロックがメモリに存在するかどうかをチェックする．この場合は，Store が行われたキャッシュにのみ最新のブロックが存在するので，最新のブロックを保持するキャッシュに対して書き戻し要求を発行し（WB Req. メッセージ），キャッシュに最新のブロックの書き戻しを行わせる（Write Back メッセージ）．メモリは Write Back メッセージを受けて，そのメッセージに付加されていた最新のブロックを取り出し，読み出し要求を行ったキャッシュに送信する（Data メッセージ）．

A Proposal of a Low Overhead Cache Coherence Protocol
Takeo Hosomi, Yasushi Kanoh and Tsutomu Maruyama
NEC Corporation. C&C Research Laboratories.
4-1-1 Miyazaki,Miyamae,Kawasaki 216,Japan

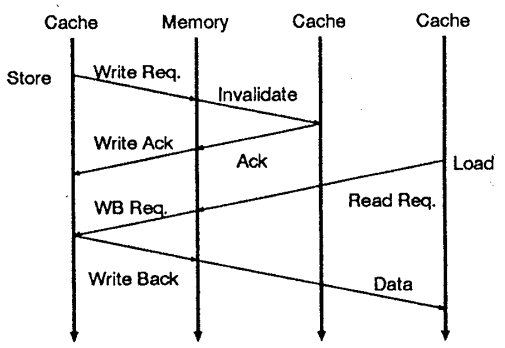


図 1: 従来のインバリデートプロトコルの動作

以上から明らかなように，従来のインバリデート型のプロトコルでは，あるキャッシュにおいてストアが行われた後に別のキャッシュで発生したロードミスのレイテンシは，メモリには最新のブロックが存在しないので，非常に大きなものとなってしまう．

3 Write-Nth プロトコル

Write-Nth プロトコルは，ロードミス時のレイテンシを削減し，ネットワークトラフィックを減少させることを目的としたインバリデート型をベースとしたプロトコルである．

キャッシュは，共有しているブロックに関してはライトスルー方式でメモリへの書き戻しを制御し，共有していない，即ち私有しているブロックに関してはライトバック方式でメモリへの書き戻しを制御する．これにより，2章に示したプロトコルと異なり，共有しているブロックに対するストアはメモリにも反映され，ストアが行われたキャッシュおよびメモリに最新のブロックが存在する状態となる．これにより，ロードミス時のレイテンシおよびメッセージを削減することが期待できる．

また，メモリにおいては，各ブロック毎に数ビットのカウンタが設けられており，以下のように動作する．

- 読み出し要求（Read Req. メッセージ）を受けたとき，カウンタの値を 0 に初期化する．
- 書き込み要求（Write Req. メッセージ）を受けたとき，カウンタの値をインクリメントする．

メモリは，上記カウンタの値を用いて，Write Ack メッセージを送信する際に，Write Req. メッセージを発行し

表 1: シミュレーションのパラメータ

プロセッサ	1 命令の実行	1 (cycle)
キャッシュ	ブロックアクセス	5 (cycle)
	ワードアクセス	1 (cycle)
	タグアクセス	1 (cycle)
ネットワーク	1 段	6 (cycle)
メモリ	ブロックアクセス	14 (cycle)
	ワードアクセス	10 (cycle)
	ディレクトリアクセス	10 (cycle)

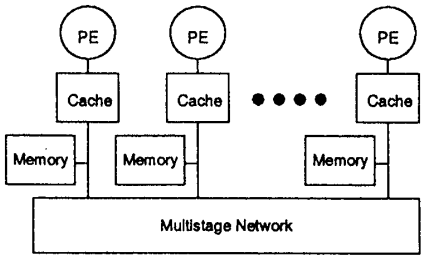
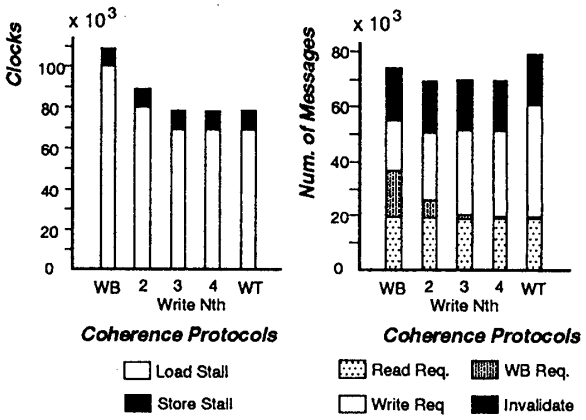


図 2: シミュレーション対象の計算機モデル



(a) メモリアクセスレイテンシ (b) ネットワークトラフィック

図 3: ネットワークトラフィックおよびレイテンシ

たキャッシュのブロックの状態を共有状態のままとするか私有状態に遷移させるかを決定する．カウンタの値が一定値（以降更新回数の上限値とする）未満であった場合は共有状態のままとし，以上であった場合は私有状態に遷移させる．この情報は Write Ack メッセージに付加されてキャッシュに通知され，ブロックの状態に反映される．これにより，不必要なメモリへの書き込み（Write Req. メッセージ）を抑止し，ネットワークトラフィックの増加を押さえることができる．

4 評価

実行駆動型のシミュレータ上で，モンテカルロ法を用いた流体力学シミュレーションプログラム（SPLASH[2] に属する mp3d）を 32 プロセッサで動作させて，ネットワークトラフィック（システム全体の合計値）および，ロードおよびストアによるプロセッサストール時間（プロセッサの平均値）を計測した．

図 2 にシミュレーション対象の計算機のモデルを示す．また，各部でのレイテンシを表 1 にまとめる．メモリアクセスオーダリングモデルは Weak[3] を用い，プロセッサでは実際に命令レベルのシミュレーションを行い全命令を 1 クロックで処理するものとした．また，キャッシュは 1 階層で，構成はダイレクトマップ，サイズは 1M バイト，ブロックサイズは 32 バイトとし，*Write-Nth* プロトコル用に同一ラインに対する書き込みのマージを行うライトバッファを 2 エントリ用意した．メモリはフルマップのディレクトリをブロック単位に保持し，ネットワークは 3 段からなる多段網でフローコントロール方式はワームホールとした．

図 3 にシミュレーションの結果を示す．図において，WB は従来のインバリデート型プロトコルを示しており，*Write-Nth* の 2 から 4 は更新回数の上限値をそれぞれ 2 から 4 に設定した場合である．また WT は，*Write-Nth* プロトコルで，メモリの指示によりキャッシュを共有状態から私有状態に遷移させなかった場合である．

図より，*Write-Nth* プロトコルを用いることで，ロードミス時のレイテンシを削減する効果が得られることが確認できる．また，ネットワークトラフィックを見ると，Write Req. メッセージは増加することにより，私有状態への遷移を行わない WT ではトラフィックは増加するものの，*Write-Nth* プロトコルでは，WB Req. メッセージを削減することでトラフィックを削減していることが確認できる．

5 おわりに

本稿では，ロードミス時のレイテンシを削減することを目的とした *Write-Nth* プロトコルの提案を行い，簡単な評価を行った．その結果，ロードミス時のレイテンシを削減し，またネットワークトラフィックを削減する効果が確認できた．

参考文献

[1] Papamarcos, M. S. and Patel, J. H.: A low overhead coherence solution for multiprocessors with private cache memories, *Proc. 11th Int'l Symp. Computer Architecture*, Vol. 12, No. 3, pp. 348–354 (1984).

[2] Singh, J. P., Weber, W.-D. and Gupta, A.: SPLASH: Stanford Parallel Applications for Shared-Memory, Technical Report CSL-TR-91-469, Stanford Univ. (1991).

[3] Adve, S. V. and Hill, M. D.: Weak Ordering - A New Definition, *Proc. 17th Int'l. Symp. Computer Architecture*, Vol. 18, No. 2, pp. 2–14 (1990).