CMP 向け分散キャッシュにおける ディレクトリの余剰エントリの活用手法

藤 枝 直 輝† 吉 瀬 謙 二†

1. はじめに

キャッシュを搭載する CMP（Chip Multiprocessor）においては，低レイテンシかつ柔軟性の高いキャッシュ構成が求められる．その一種として，占有キャッシュをベースとしながらも，あるコアのキャッシュから追い出されたラインを別のコアのキャッシュへと移動可能とする構成が提案されている[1),2)]．

我々は，そうした構成の 1 つである Distributed Cooperative Caching（DCC）[2)] を取り上げる．DCC はコヒーレンシ制御のために，Distributed Coherence Engine（DCE）とよばれるディレクトリキャッシュを持つ．DCE がキャッシュの利用効率を損なわないようにいくらかの余剰エントリを持っていることに注目する．これらの余剰エントリが持つタグの情報を捨てずに保持し続けることで，最近チップから追い出されたキャッシュラインを知ることができる．我々は，この情報を利用してキャッシュを効率化することを目標に，余剰エントリの活用手法について検討を行なっている．

本稿では，余剰エントリの活用手法のひとつとして I 参照フィルタを提案し，メニーコアシミュレータを用いて 8 コア・2 アプリケーションの環境で評価を行う．

2. ディレクトリの余剰エントリ

図 1 に DCC[2)] の構成を示す．各コアと DCE，メインメモリとは何らかのインターコネクトで接続される．各コアは Processing Element（PE）および L1，L2 のキャッシュを持つ．あるコアが自コアのキャッシュにミスすると，物理アドレスから一意に定められる DCE に対しリクエストを送信する．その際，もし DCE 内にタグが一致する有効なエントリが存在すれば，他コアのキャッシュ内に所望のキャッシュラインが存在する．そのため，DCE はリクエストをそのラインを持つコアに転送し，リクエストを受けたコアがリクエストを発したコアへ必要なラインを供給する．一方，もし DCE 内にタグが一致する有効なエントリがなければ，チップ内に所望のラインは存在しないので，メインメモリがそのラインを供給する．

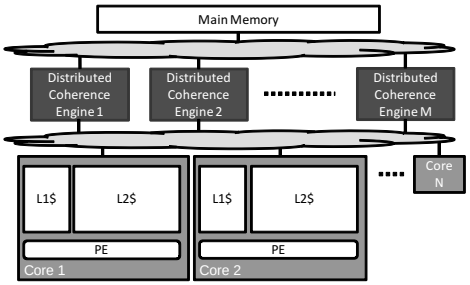

図 1 Distributed Cooperative Caching の構成.

DCC の重要なポイントは，あるコアのキャッシュから溢れて追い出されたラインを，別のコアのキャッシュへと移動することを許していることである．これにより，キャッシュを大量に必要とするコアが，他のコアの利用していないキャッシュ領域を借りることで，キャッシュ容量への要求のばらつきに対応することを可能としている．その際，どのラインをどのコアへと移動させるかを正しく制御することが，キャッシュの効率化に影響を与えることが知られている[3)]．

我々が注目するのは，DCE が自身の空きエントリが不足することに起因するキャッシュの無効化を防ぐために，ディレクトリの総エントリ数をキャッシュの総ライン数よりも大きく（例えば 1.5 倍に）設定していることである．このことは，少なくともある一定数のエントリは未使用のままとなることを意味する．我々は，これら未使用の領域が持つ情報を，追い出されたラインの移動を制御するためのヒントとして活用する，効率的なキャッシュ制御を目指している．

3. I 参照フィルタ

本節では，前節で述べた余剰エントリの活用法のひとつとして I 参照フィルタを提案し，その評価を行う．

本手法では，あるキャッシュラインがチップから追い出され，対応するディレクトリのエントリが DCE から取り除かれる時に，単にエントリの状態を無効と

† 東京工業大学 大学院情報理工学研究科
Graduate School of Information Science and Engineering, Tokyo Institute of Technology

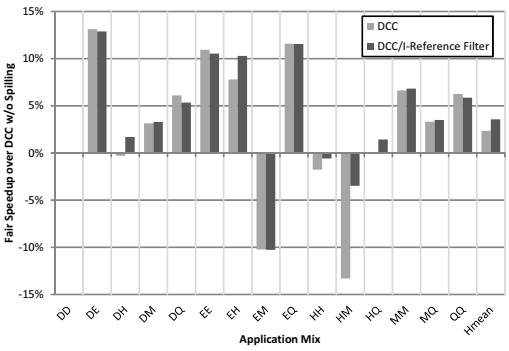

図 2 I 参照フィルタの性能評価結果.

するだけで，タグの情報はそのまま保持しておく．もしそのエントリが別のエントリにより再利用されるよりも前に，そのエントリが持つタグと一致するラインへの要求が再度発生した場合，このような無効化エントリへの参照を I 参照と呼び，他と区別する．

I 参照フィルタでは，ディレクトリの各エントリに対して，そのエントリが I 参照されたかどうかを示す 1 ビットの状態を追加する．この状態は最初に有効になった時に 0 にクリアされ，エントリが I 参照された時に 1 にセットされる．そして，あるラインがキャッシュから追い出されたときは，対応するディレクトリのエントリを参照する．もし状態のビットが 1 であればラインを他のコアへと移動することを許可し，0 であればラインの移動を許可せず，ラインを単に無効化（またはライトバック）する．これにより，再利用される可能性が低いラインが他のコアへと移動することによるキャッシュの汚染を軽減することを狙う．

I 参照フィルタの性能を評価する．評価には，メニーコアプロセッサシミュレータ SimMc[4] に対し，キャッシュやディレクトリなどの拡張を施したものを用いる．計算コアと DCE をそれぞれ 8 個とする．4 並列のアプリケーションを 2 つ同時に実行し，各アプリケーションの実行時間比の調和平均を計測する．アプリケーションは，キャッシュをほとんど使用しない D(Dijkstra 法による経路探索)，大量のキャッシュを必要とする E(Equation Solver Kernel)，キャッシュサイズに関わらずミスを多く発生させる H(姫野ベンチマーク)，キャッシュの汚染による悪影響を受けやすい M(行列積)，コアによって要求されるキャッシュの容量が異なる Q(クイックソート) の 5 種類である．

評価結果を**図 2** に示す．横軸はアプリケーションの組み合わせ，縦軸は全てのラインの移動を許可しなかった場合に対する，DCC および I 参照フィルタを適用した DCC(DCC/I-Reference Filter) の相対性能である．横軸の Hmean は調和平均である．

I 参照フィルタを適用した結果，I 参照フィルタを適用しない DCC と比べて，平均で 1.2%性能が改善された．いくつかのアプリケーションでわずかに性能オーバーヘッドが発生する一方で，キャッシュの汚染を引き起こしやすい H を含む組み合わせで性能向上が見られた．これは，再利用の可能性が低いラインを移動の対象から除外した効果とみられる．

一方で，全てのラインを移動しない場合と比べて，I 参照フィルタを利用してもなお性能が低下する組み合わせも存在する．特に低下幅が大きいのは，キャッシュの汚染に影響されやすい M を含む組み合わせ 2 種類であり，これを改善するにはラインの移動先コアの考慮が必要である．しかしながら，I 参照フィルタはどのラインを移動するかのみを決定しており，ラインの移動先については考慮していない．適切なラインの移動先の選択については，他の手法とも組み合わせることで効率化が達成できると考えられる．

4. ま と め

我々は，DCC においてディレクトリの余剰エントリが持つ情報を活用する手法について検討している．本稿では，そのひとつとして I 参照フィルタを提案し，評価を行った．I 参照フィルタを適用しない DCC と比較した性能向上は平均 1.2%であった．

今後は，余剰エントリの活用手法を引き続き検討していくとともに，これらを既存の手法と組み合わせ，更なるキャッシュの効率化を図ることを考えている．

謝辞 本研究の一部は，科学技術振興機構・戦略的創造研究推進事業（JST CREST）の「アーキテクチャと形式的検証の協調による超ディペンダブル VLSI」の支援による．

参 考 文 献

1) Chang, J. and Sohi, G. S.: Cooperative Caching for Chip Multiprocessors, *Proceedings of the 33rd annual international symposium on Computer Architecture*, pp. 264–276 (2006).
2) Herrero, E., González, J. and Canal, R.: Distributed cooperative caching, *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, pp. 134–143 (2008).
3) Qureshi, M.: Adaptive Spill-Receive for robust high-performance caching in CMPs, *Proceedings of the 15th IEEE International Symposium on High Performance Computer Architecture*, pp. 45–54 (2009).
4) 植原昂, 佐藤真平, 吉瀬謙二: メニーコアプロセッサの研究・教育を支援する実用的な基盤環境, 電子情報通信学会システム開発論文特集号, pp. 2042–2057 (2010).