エントリの再利用性を考慮した共有キャッシュの動的分割手法

進藤 智司*（名古屋工業大学），濵西 夏生（電気通信大学），津邑 公暁（名古屋工業大学）

Dynamic Cache Partitioning based on Entry Reusability

Satoshi Shindo (Nagoya Inst. of Tech.), Natsuki Hamanishi (UEC), Tomoaki Tsumura (Nagoya Inst. of Tech.)

1. はじめに

これまで，ゲート遅延に対する配線遅延の相対的な増大により，配線遅延を隠蔽するキャッシュメモリの重要性が高まってきた．また，消費電力削減と高性能化を実現するためにマルチコアプロセッサが広く普及している．このようなマルチコアプロセッサでは，複数のコアがキャッシュの一部を共有する場合が多く，あるコアで動作しているプロセスによって確保されたキャッシュエントリが，他のコアで動作しているプロセスに追い出される干渉という問題が発生する．本稿では，共有キャッシュ領域を分割し，各コアに占有領域を割り当てることで干渉による性能低下を抑制する手法を提案する．

2. RWP

キャッシュ領域を分割することでキャッシュメモリの性能向上を図る手法の 1 つである RWP（Read Write Partitioning）[(1)]では，キャッシュエントリの再利用性を考慮してキャッシュ領域を分割する．具体的には，セットアソシアティブキャッシュにおいて各セットを，読み出しアクセスによってのみ再参照されるデータが配置される Clean 領域と，読み出しと書き込みアクセスの両方よって再参照されるデータが配置される Dirty 領域の 2 つの領域に分割し，これら 2 つの領域のサイズを動的に変化させることで，キャッシュメモリに対する読み出しアクセスのミスが発生する回数を抑制している．これを実現するために，RWP ではキャッシュヒットした際に，各セット内のエントリに対する読み出しアクセスの成功回数を追加ハードウェアであるカウンタ（Age Hit Counter）に記憶する．そして，キャッシュミスした際にこのカウンタの値に基づいて，最も読み出しアクセスの成功回数が多くなるような Clean/Dirty 領域の内訳を予測し，それに近づくようにキャッシュエントリを管理する．

3. RWP の問題点とその解決手法

RWP では，マルチコア環境において重大な問題である干渉によるキャッシュの性能低下を回避できない．また，RWP では複数の異なるプロセスが動作している場合も，1 つの Age Hit Counter の値に基づいて一意に決定した Clean/Dirty 領域の内訳に近づくようにキャッシュエントリを管理する．このためプロセス毎の特徴を考慮した Clean/Dirty 領域の分割を行うことができない．1 つ目の干渉による性能低下は，各コアに占有領域を与えることで解決でき，それに伴い占有領域毎に Clean/Dirty 領域の内訳サイズを設定することも可能となるため，2 つ目の問題も同時に解決できると考えられる．そこで本稿では，キャッシュをセット方向へ分割し，各コアに占有領域を与えることで，キャッシュ性能の向上を図る．さらに，各プロセスの必要量に応じ，各占有領域のサイズを動的に調整することで，更なるキャッシュ性能の向上を図る．

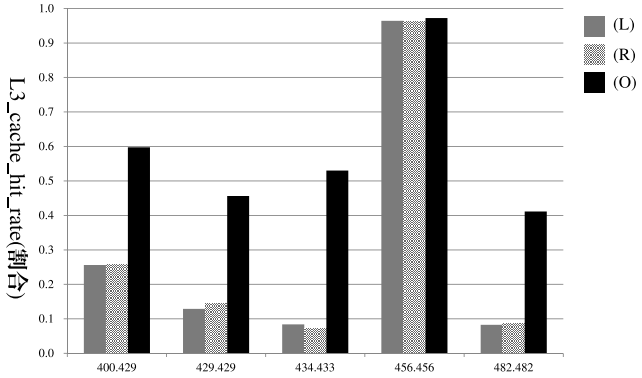

Fig.1　Cache Hit Rate

4. 提案手法の実装

提案手法を実現するために，コア毎の読み出しアクセスのミス回数を記憶するハードウェアと，各セットが含まれている占有領域がどのコアに保持されているかを記憶するハードウェアを新たに追加する．さらに実装する環境のコア数と同数の Age Hit Counter を用意する．これらに必要なハードウェアコストは約 6.7KBytes であり，既存の共有キャッシュサイズ 4MBytes と比較しても，非常に小さい増加量で実現可能である．

5. 評価

提案手法をシミュレータ上に実装し，汎用ベンチマークプログラムである SPEC CPU2006 を用いて評価した．L3 キャッシュを共有した構成で，250M 命令のスキップ後 250M 命令を実行した場合の共有キャッシュに対する読み出しアクセスのヒット率を，(L) 追い出しアルゴリズムとして LRU のみを用いた場合，(R) RWP を用いた場合，(O) 提案手法を用いた場合で比較した結果を Fig.1 に示す．評価の結果，既存の RWP と比較して，共有キャッシュに対するキャッシュヒット率を最大 46 %，平均 29 %向上できることを確認した．

文　献

(1) Khan, S., et.al.: Improving Cache Performance by Exploiting Read-Write Disparity, *Proc. 20th Int'l Symp. on High-Performance Computer Architecture (HPCA)*, pp.452–463 (2014).