2009 年 4 月 2 日 情報システム基盤学専攻修士論文中間発表資料

CMP におけるインデックスごとの有効利用ウェイ数を考慮した共有キャッシュ分割に関する研究

高性能コンピューティング学講座　本多研究室
鈴木　翔平 (0853013)
主任指導教員：本多　弘樹

1　概要

近年、一つのチップ上に複数のコアを搭載したチップマルチプロセッサ (CMP) が主流になってきている。これは複数のコア上で複数のプロセスやスレッドを同時に実行することでスレッドレベル並列性を引き出すためである。

多くの CMP では L2、L3 といった最高次キャッシュをチップ上の全てのコアで共有している。これはキャッシュを共有することでキャッシュコヒーレンシ問題を解決できることや、他のコアが動作していないときに大きなキャッシュとして利用することができるといった利点があるためである。

しかし共有キャッシュにおいて追い出すキャッシュブロックを単純な LRU によって決定してしまうと、アクセス頻度によって各コアに割り当てられる容量が決定してしまうため、何らかの方法でキャッシュを分割する必要がある。

そこで本研究では、各コアのインデックスごとの有効利用ウェイ数をもとにキャッシュを分割する手法を提案する。

2　LRU の問題点

共有キャッシュに LRU を用いることの問題点は、LRU の「一番最近アクセスされたデータは MRU、つまり最も追い出されにくい位置に移動する」という性質によるものである。例えばアクセス頻度の高いコア A と低いコア B がキャッシュを共有した場合、A のデータは MRU 側に集まり B のデータは LRU 側に集ることになり、追い出しが発生すると B のデータが追い出される確率が高い状態となる。したがってキャッシュ上には A のブロックが多く残り B のブロックはあまり残らない。しかしアクセス頻度が高いこととキャッシュに残ったデータを再利用する確率は必ずしも相関があるわけではないので、A がキャッシュに残ったデータを再利用する確率が低ければキャッシュ容量を無駄にしてしまうことになる。

3　従来研究

共有キャッシュを分割する代表的な手法はウェイ粒度分割である [1][2]。この手法は総ウェイ数と同じ数のカウンタを用いて実現される。

情報取得方法:キャッシュアクセスの際にアクセスしたブロックの LRU 情報と同じ番号のカウンタをインクリメントする。キャッシュとカウンタの構成を図 1 に示す。

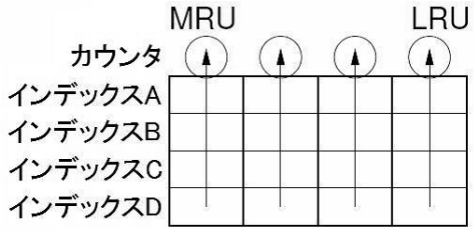

図 1: カウンタの構成

分割方法:カウンタの情報を基に使用できるウェイ数を制限したときにキャッシュミスが何回増えるかを計算し、全体として最もキャッシュミス回数が少なくなる割合でキャッシュを分割する。

例えばウェイ数 4 の共有キャッシュへのアクセスが 100 回行われ、その結果各カウンタ値が表 1 のようになったとする。

表 1: ヒットカウンタ

カウンタ番号	カウンタ値
1	30
2	20
3	15
4	10
キャッシュミス=25	

もし使えるウェイ数が 4 ウェイから 3 ウェイに減ったとすると、LRU 情報 4 でヒットしていたものが全てミスとなるため、増えるキャッシュミスはカウンタ番号 4 の値つまり 10 回となる。同様に 3 ウェイから 2 ウェイに減ると 15 回増え、2 ウェイから 1 ウェイに減ると 20 回増えることになる。

これによって各ウェイ数におけるキャッシュミス回数を見積もることができる。分割する際はプロセッサ

全体でキャッシュミスが少なくなる割合でキャッシュを分割する。

4 ウェイ粒度分割の問題点

ウェイ粒度分割では、キャッシュ内の全てのインデックスを一様に扱って情報を取得している。そのため、インデックス間に有効利用しているウェイ数の差があってもそれを吸収してしまう。

例えば、4 ウェイの共有キャッシュにおいてインデックスが A、B、C、D の 4 種類あり、インデックス A には MRU ブロックから LRU ブロックまで 4 つのブロックにまんべんなくアクセスし、残りのインデックスには MRU ブロックに多くアクセスする状況を考える。この場合、インデックス A によって 4 つのカウンタ全てが増加させられるため、インデックス B、C、D は 1 ウェイで十分にも関わらずそれ以上のウェイが割り当てられてしまう可能性がある。

5 提案手法

提案する手法はインデックスごとの利用率の違いを考慮し、より適切な分割が行えるようにする共有キャッシュ分割手法である。ウェイ粒度分割同様、カウンタを用いてキャッシュアクセス情報を取得し、その情報をもとに分割を行う。全てのインデックスにカウンタを設けるとハードウェアオーバーヘッドが大きくなってしまうため、ウェイ粒度分割で用いたカウンタを流用して情報を取得する。

- 構成：
  基本的な構成はウェイ粒度分割と同様である。最初の分割はウェイ粒度分割を行い、次にインデックスごとのウェイ数を調整する。

- 分割方法：
  ウェイ粒度分割で用いたカウンタを流用するため、インデックス総数に対してカウンタの数はとても少ない。そこで、インデックスをいくつかのブロックに分けてその間で調整する方法と、調整が必要なインデックスが集中している箇所を見つけ、そこのみを調整する方法を考えている。

6 進捗状況

プロセッサシミュレータ M5 を用いて、インデックスごとの有効に利用しているウェイ数に違いがあるかどうかを調べた。まず利用可能ウェイ数を最大ウェイ数から 1 ウェイまで変化させ、それぞれの場合において各インデックスのヒット率を求め、ヒット率とヒット回数の積を各ウェイ数における「有効利用ポイント」とする。次にキャッシュを半分にした時の「全体のヒット率 × 全体のヒット回数/インデックス総数」をしきい値とし、各インデックスの有効利用ポイントがウェイ数をいくつ減らした時にそのしきい値を下回るかを調べた。そして下回るウェイ数+1 を「そのインデックスが有効に利用しているウェイ数」と定義し、SPEC ベンチマークにおいてインデックスごとの有効利用ウェイ数がばらつくかどうかを調べた。結果を図 2 に示す。

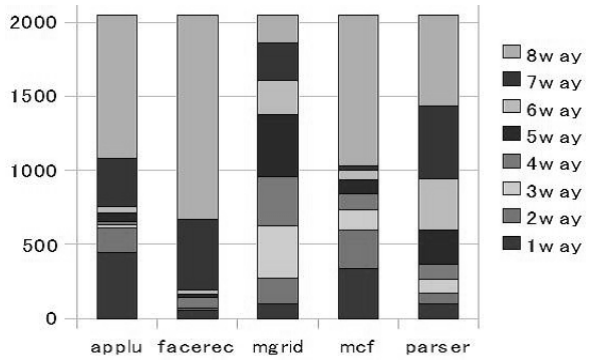

図 2: 有効利用ウェイ数のばらつき

図 2 において、縦軸はインデックスの数を表し、各棒グラフは有効利用ウェイ数ごとにインデックスを分類し積み上げたものである。この結果より、インデックス間には利用率に差があり共有キャッシュを分割する際はインデックスごとの有効利用ウェイ数を考慮する必要があることがわかった。

7 まとめと今後の方針

共有キャッシュの分割手法について、従来手法の問題点とそれに対する解決のための構想を述べた。

今後の方針としては、まずウェイ粒度分割において不適切な分割が行われてしまっているかを調べる必要がある。ウェイ粒度分割はキャッシュ全体のブロック数が「割り当てられたウェイ数 × インデックス数」のブロックになるよう設計されているため、インデックスごとに割り当てるウェイ数を違う数にすることは可能である。これを調べることで提案手法の有効性を確認する。提案手法の有効性を確認したのち、前述の M5 を用いて設計・実装・評価を進めていく予定である。

参考文献

[1] GE Suh et al:Dynamic Partitioning of Shared Cache Memory,*The Journal of Supercomputing*, pp7-26.2004

[2] MK Qureshi et al:Utility-Based Cache Partitioning: A Low-Overhead,High-Performance, runtime Mechanism to Partition Shared Caches, *International Symposium on Microarchitecture*, pp423-432.2006