2008－ARC－179 (3)
2008／8／5

プログラム実行時のキャッシュ連想度の需要予測方式

小川周吾[†] 入江英嗣[†]
菅原豊[†] 平木敬[†]

近年プロセッサに対する省電力化が求められている．またマルチコア，マルチスレッドプロセッサにおける共有キャッシュの利用効率の向上が求められている．それらの実現方法の一つにキャッシュの使用可能な way を動的に調整する方式が提案されている．しかしキャッシュの連想度を性能及び電力効率で最適化するために，実行プログラムのキャッシュ連想度に対する需要を予測する必要がある．本稿ではキャッシュ上の置換直前のブロックでのヒット回数を用いた連想度の需要予測方式を提案する．ヒット回数をプロセッサ内のカウンタで測定し，その結果から連想度増減後の性能を予測する．本稿の提案方式の予測精度をシミュレータ上で評価した．その結果，way 数変更時の性能，及び実行時に一定のキャッシュヒット率を得るために必要な連想度を推定可能であることが判明した．

Dynamic Estimate Method of Requirement for Cache Associativity

Shugo Ogawa,[†] Hidetsugu Irie,[†] Yutaka Sugawara[†]
and Kei Hiraki[†]

Demand for power saving of processors is increasing recently. Moreover, demand for improvement of cache utility on multicore and multithread processors is also increasing. For satisfying those demands, the method which control the associativity of set-associative cache is proposed. However, the method which estimate way usage condition of cache is also needed. In this paper, we propose a method which calculate cache hit ratio at each number of way using counters in processor, and then estimate the performance after changing cache associativity. We evaluate estimation accuracy of our new method with simulation, then we show that we can estimate the performance after changing cache associativity and adequate cache associativity for achieving a certain performance with our method.

1. 序　　論

近年プロセッサの省電力化が求められている．その手法として不要な資源への電力供給の抑制が挙げられる．その対象となる資源の一つがキャッシュである．キャッシュはプログラム毎に使用頻度，実行速度の維持に必要な容量が異なる．そのためキャッシュの不使用箇所への電力供給の抑制により省電力化ができる[8)1)]．

また近年多くのマルチコア，マルチスレッドプロセッサは複数コア，スレッド間でキャッシュを共有する．複数のコア，スレッドがキャッシュを同時に使用することでキャッシュ競合の増加に伴い性能が低下する．複数コア，スレッド間のキャッシュ競合の回避には各コア，スレッドの使用キャッシュ量の制限が必要である．

キャッシュ容量の制限方法の一つに，今日広く用いられるセットアソシアティブ方式のキャッシュの way 数を動的に調整する方式が提案されている[2)4)]．この方式ではキャッシュの有効箇所を way 単位で変更することで使用するキャッシュ量を制限する．この方式で省電力化を行う場合，使用頻度の低い way への電力供給の停止により大幅な性能低下を起こさずに消費電力を抑制できる．しかしこの方式を用いてマルチコア，スレッドプロセッサの共有キャッシュでの競合を削減する場合[4)6)]，各コア，スレッドへのキャッシュの割当 way 数をプロセッサ全体のキャッシュミスが少なくなるように決定する必要がある．以下，本稿ではキャッシュヒット率や IPC に関する一定の性能条件の達成に必要な way 数を，キャッシュヒット率から見た連想度需要と定義する．

つまりキャッシュway 数の動的な調整による省電力化，共有キャッシュにおける競合削減の実現には，プログラムのキャッシュ使用量に応じた way の動的な割り当てが必要である．連想度需要はキャッシュの競合頻度から予測する．キャッシュの競合頻度は例えばキャッシュヒット率で表される．

† 東京大学大学院情報理工学系研究科
The University of Tokyo, Graduate School of Information Science and Technology

本稿ではヒット率から見た連想度需要を予測しway数を最適化するために，置換直前のキャッシュブロックのヒット回数を用いた連想度需要の予測方式を提案する．多くのプロセッサでは性能カウンタによりメモリアクセス数，キャッシュミス数を測定できる．また，置換直前のキャッシュブロックのヒット回数を記録するカウンタを新たに用意する．このカウンタ値からway数が現在より1少ない場合のキャッシュミス回数を計算する．更に各way数でのミス回数の差から連想度変更時のキャッシュミス回数を予測する．ミス回数の予測結果からキャッシュヒット率，IPCを予測し，予測値が一定の基準値を満たすようにway数を増減する．

2. 関連研究

過去に提案されたキャッシュの容量制限方式，パーティショニング方式及び使用量の予測方式を説明する．

2.1 キャッシュ使用量の制限による省電力化

省電力化を目的としたキャッシュ使用量の制限方法としてCache Decay[8]が提案された．Cache Decayでは各ブロックの最後のアクセス時間を調べ，長時間アクセスがないブロックを停止する．これにより性能低下幅を抑えつつ省電力化を実現する．

また，セットアソシアティブキャッシュにおいてセット毎に次のアクセス先として予測されるブロック以外を停止する適応型ウェイ予測キャッシュ[9][10]が提案された．またこの方式では前述のブロック停止方法をアクセス先予測の精度が高いセットのみに適用する．これによりキャッシュブロックの停止に伴うミス増加による性能低下幅の縮小と省電力化を実現している．

これらの省電力化方式はキャッシュの不使用箇所の発見，停止に有用である．しかしプログラムが頻繁にアクセスするデータを格納するために全体で必要とするキャッシュの量を調べることは困難である．

2.2 共有キャッシュのパーティショニング

近年キャッシュパーティショニング方式としてキャッシュをway単位で分割する方式が複数提案された[7]．以下ではway単位のパーティショニングに限定して述べる．Albonesi[1]らは，実行性能を落とさずにキャッシュの使用量を制限し，省電力化を図る方式としてSelective Cache Waysを提案した．この方法ではソフトウェアがキャッシュの使用量に基づき有効なwayを変更することで，性能を落とさずにキャッシュ使用量の削減を実現できることが示された．またway単位のキャッシュパーティショニングは文献[3][5]でも提案された．しかし具体的なキャッシュの連想度の決定方式，動的なway数の変更方法は議論されていない．

更に文献[2][4]では，パーティショニングによるwayの割り当てを動的に行う方式が提案された．Sueら[4]は，LRUで置換を行うキャッシュに対してパーティショニング変更の判断にキャッシュのアクセス分布を記録するカウンタを使用した．アクセス分布は，ヒットしたブロックの同一セット内での最終アクセス時刻の順位毎にヒット数を記録する．つまりMRUからLRUまでの各wayでのヒット数の分布を記録する．また同様の方式でキャッシュ使用状況を調べてパーティショニングする方式が文献[6]で提案された．しかしこれらの方式はway毎にヒット数を記録するカウンタが必要であり，連想度の大きい共有キャッシュ等では必要資源の増加につながる．加えて配線もカウンタ数に比例して増えるため実装が複雑という問題点がある．

3. キャッシュ連想度の需要予測方式

本稿では実行プログラムのヒット率から見た連想度需要の予測手段として，LRUカウンタを用いた連想度需要予測方式(LRUカウンタ方式)を提案する．本提案方式では，次の2種類の性能指標についての基準値を連想度需要の判断基準とする．

- キャッシュヒット率
- IPC

キャッシュヒット率を性能指標とした場合，キャッシュヒット率の基準値を満たすために必要なway数がキャッシュから見た連想度需要を表す．一方IPCを性能指標とする場合，プログラム毎にIPCの値は異なるためIPCを直接基準とすることは困難である．そこでIPCの最大値，つまりキャッシュのway数最大の場合のIPCをIPC_{max}とした場合現在のIPCの比率IPC/IPC_{max}を相対IPCと定義して性能指標とする．

3.1 連想度需要予測，way数増減処理の流れ

連想度需要予測，及びその結果によるway数の増減の一連の処理は以下の手順で行う．

(1) ヒット率，ヒット回数，IPCの算出
(2) way数増減時の性能予測
(3) 性能予測結果に基づくway数増減

これらの一連の処理を一定時間毎に行う(図1)．以下，本稿では性能カウンタによる計測時間の単位をセクションと定義する．各セクションの終了時に連想度需要の予測，way数の増減処理を行う．

まず(1)ではプロセッサの性能カウンタを用いて実行命令数，メモリアクセス回数，キャッシュミス回数を計測する．多くのプロセッサでは内部のイベント発生回数を記録する性能カウンタを持つ．この計測結果からキャッシュヒット回数，ヒット率，IPCが得られる．

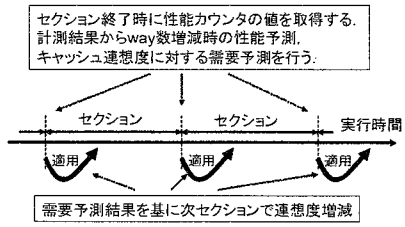

図 1　セクションと性能カウンタ計測，連想度増減時期の関係

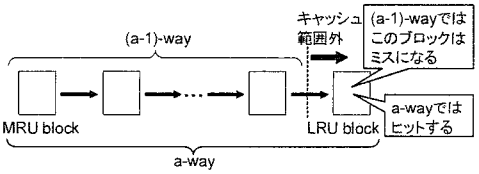

図 2　キャッシュにおける連想度減少と LRU の関係

次に (2) ではプロセッサ上に新たにカウンタを追加して，キャッシュの各セットで次の置換対象ブロックのヒット回数の合計を記録する．多くのプロセッサのキャッシュでは LRU，疑似 LRU で置換対象ブロックを決定する．これらのキャッシュでは各セット上の最もアクセス時間が古いブロックが置換される．よってヒットしたブロックが次の置換対象であるか判定することでこれらのブロックのヒット回数を計測できる．以下，本稿ではキャッシュの各セットで次の置換対象のブロックを LRU ブロック，LRU ブロックのヒット回数を計測するカウンタを LRU カウンタと定義する．このカウンタ値から way 数を 1 減らした場合のヒット率，IPC を求める．さらにキャッシュway 数増加時のキャッシュヒット率または IPC を予測する．

最後に (3) では (2) までの結果を用いてヒット率から見た連想度需要を算出する．需要の予測結果と現在の性能を比較して需要が満たされていない場合はキャッシュの way 数を増やす．一方現在需要が満たされており，way 数を 1 減らした場合でも連想度需要を満たすと予測される場合はキャッシュの way 数を 1 減らす．

3.2　キャッシュ連想度増減時の性能予測

3.2.1　連想度が 1 減少した場合の性能予測

まずキャッシュの連想度が 1 少ない場合のキャッシュミス数を予測する．LRU ブロックのヒット回数を n_{LRU} とする．LRU ブロックは次の置換対象ブロックであるため，LRU ブロックにヒットしたアクセスは way 数が 1 少ない場合にはミスになると仮定できる (図 2)．よってキャッシュの way 数が 1 少ない場合のキャッシュミス回数は，LRU ブロックのヒット回数 n_{LRU} とキャッシュミス回数 m の合計 $n_{LRU}+m$ と予測できる．このときキャッシュヒット率は，総メモリアクセス回数を N とすると $n_{LRU}+m/N$ と予測できる．

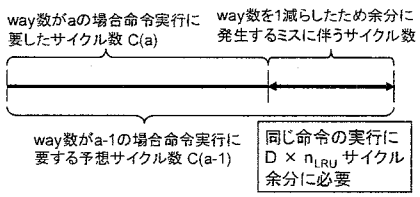

図 3　way 数減少と命令実行に要するサイクル数の増加量の関係

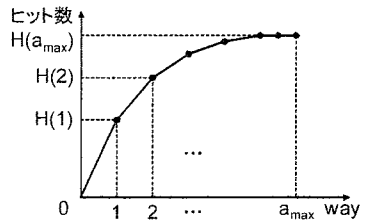

図 4　キャッシュ連想度とヒット数の一般的な関係

一方 way 数が 1 少ない場合の IPC の予測値は，同じセクションの命令の実行に必要なサイクル数の予測値から求められる．図 3 を用いて way 数が 1 減少した場合の命令実行に要するサイクル数の予測手順を述べる．メモリアクセスの増加 1 回あたりの実行サイクル数の増加分を D とする．また way 数が x の場合に直前のセクションと同一命令列の実行に要するサイクル数を $C(x)$ とする．現在の way 数が a の場合，$C(a)$ は直前のセクションのサイクル数である．

つまり way 数が 1 減少した場合に命令実行に要するサイクル数はキャッシュミスの増加数 n_{LRU} から $C(a-1)=C(a)+D\times n_{LRU}$ と予測できる．更にセクション内の実行命令数を $Inst$ とすると，way 数が $a-1$ の場合の IPC は $IPC(a-1)=Inst/C(a-1)$ と予測できる．本稿では単純に，実行サイクル数の増加分 D をメモリアクセスのレイテンシに等しいと仮定する．この仮定は各メモリアクセスのレイテンシが命令実行順序の最適化で隠蔽されない場合に成立する．また例えばメモリが SDRAM である場合，連続領域へのアクセスは高速に処理される．そのため仮定は，キャッシュミスを起こすアクセスが一度に複数ブロックに渡る連続領域で発生しない場合に成立する．

3.2.2　way 数増加時のキャッシュヒット数予測

LRU ブロックのヒット数と，way 数とヒット数の関係から way 数増加時のキャッシュミス数を予測する．図 4 は way あたりのサイズが一定の場合のキャッシュの way 数とヒット数の関係を表す．アクセスに時間

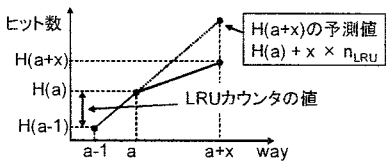

図 5 LRU カウンタを用いた way 数増加時のヒット数予測

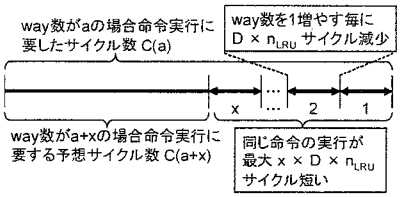

図 6 連想度増加と命令実行に要するサイクル数の関係

的局所性を持つプログラムでは，way 数の増加に伴いヒット数の増加幅が減少する (図 4)．つまり way 数 a でのヒット数を $H(a)$ とすると式 (1) が成立する．

$$H(a) - H(a-1) \geq H(a+1) - H(a) \quad (1)$$

図 5 の例で way 数 $a+x(x>0)$ でのヒット数の予測手順を述べる．式 (1),(2) から way 数 $a+x$ でのヒット数 $H(a+x)$ について式 (3) が成り立つ．

$$H(a)+\sum_{k=1}^{x}(H(a+k)-H(a+k-1)) = H(a+x) \quad (2)$$

$$H(a)+x\times(H(a)-H(a-1)) \geq H(a+x) \quad (3)$$

way 数 a と $a-1$ のヒット数の差は LRU カウンタの値 n_{LRU} である．よって式 (3) から式 (4) が成り立つ．

$$H(a) + x \times n_{LRU} \geq H(a+x) \quad (4)$$

ヒット数 $H(a+x)$ はメモリアクセス数 N 以下なのでヒット率 $R(a+x)$ について式 (5) が成り立つ．

$$\min((H(a)+x\times n_{LRU})/N, 1) \geq R(a+x) \quad (5)$$

一方 IPC は，連想度が 1 少ない場合の IPC の予測値と同様の方法で予測できる．メモリアクセス 1 回あたりの実行サイクル数の増加量を D，実行命令数を $Inst$ とした場合，連想度 $a+x$ での予想実行サイクル数 $C(a+x)$ について式 (6) が成り立つ (図 6)．

$$C(a+x) \geq C(a) - x \times D \times n_{LRU} \quad (6)$$

IPC は $Inst\,/\,C(a+x)$ であり式 (6) から予測できる．

3.3 キャッシュ連想度に対する需要予測

3.3.1 性能指標がキャッシュヒット率の場合

性能が基準値を満たしているかの確認は実際のキャッシュヒット率と基準値の大小比較で確認できる．つまり基準値のヒット率を満たすためには，それに必要となる way 数の予測は不要である．

しかし way 数増加時の性能予測結果から基準値を満たすために必要な way 数を求めることで，基準値を満たすまでの時間を短縮できる．例えば way 数 $a+x$ でヒット数が $H(a+x) = H(a)+x\times n_{LRU}$ の場合に $x \geq 2$ でヒット率が基準値を満たすとする．この場合，way 数の不足分は 2 以上である．これにより way 数を 1 増加させた場合にヒット率が基準値を満たすか調べる必要がなくなり，早く基準値が満たされる．

3.3.2 性能指標が相対 IPC の場合

性能指標が IPC であれば，性能が基準値を満たしているか判定するには前述のキャッシュヒット率と同様に基準値と IPC を比較すればよい．しかし IPC は実行条件が同一でもプログラム毎に値が異なり，基準値への利用は困難である．そこでキャッシュの way 数による性能差を明確にするために，相対 IPC を用いる．

相対 IPC を評価するためには，way 数最大時の IPC を予測する必要がある．前述の way 数増加時の性能予測方法を用いて連想度最大時の IPC を予測できる．way 数最大時の IPC の予測値は，前述の式 (6) から求められる，命令実行に必要なサイクル数から算出する．way 数最大時は way 数の増分 x は，最大 way 数の値から現在の way 数 a を引いた値に等しい．この式から way 数最大時の実行サイクル数の最小値，及び IPC の最大値が求められる．本稿では IPC の予測値をこの最大値と予測することで，現在の way 数での IPC の値から相対 IPC の値を予測できる．

4. LRU カウンタ方式の評価

提案した LRU カウンタ方式の連想度の性能予測，需要予測に対する効果を評価する．性能指標はキャッシュヒット率，または相対 IPC とする．

4.1 評価環境

評価はシミュレータ上に LRU カウンタ方式としてセクション毎の性能予測，需要予測，way 数の増減処理を実装して行う．評価では LRU カウンタ方式による性能予測結果と実際の値を比較して予測精度を評価する．また，需要予測結果の精度を基準値を満たす実際の最適 way 数と比較して評価する．

シミュレーションパラメータについて述べる．シミュレータは SipleScalar 3.0d[11] に含まれる sim-outorder を用いる．ベンチマークは SPEC CINT2000，入力データは ref を用いて各プログラムの先頭 10 億命令から更に 10 億命令をシミュレータで実行する．性能，需要の予測，way 数の増減を行うセクションの長さは 1000 万命令分とする．シミュレータの CPU 及びキャッシュは図 7 に示した通りである．また L1/L2/メモリのレイテンシはそれぞれ 1/10/100 サイクルと

CPU 命令セット	Alpha
L1 Instruction/Data (cache)	独立
L2 Instruction/Data (cache)	共用
キャッシュブロック長	512-bit
L1 キャッシュセット数	128
L2 キャッシュセット数	2048
L1 キャッシュ連想度	4-way
L2 キャッシュ連想度	最大 16-way
ブロック置換ポリシー	LRU

図 7 CPU とキャッシュの設定

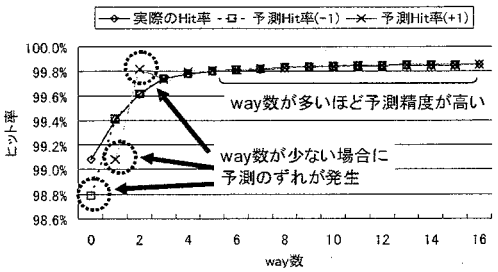

図 8 LRU カウンタ方式によるヒット率予測結果と実際の値

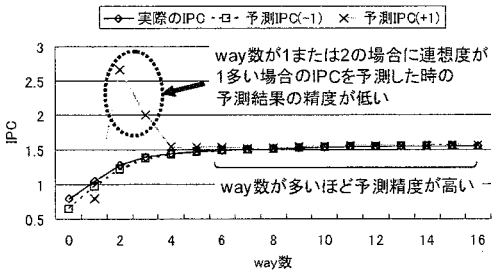

図 9 LRU カウンタ方式による IPC 予測結果と実際の値

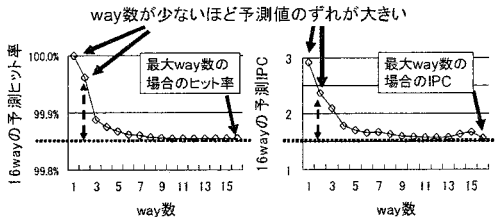

図 10 最大 way 数での性能の予測結果と予測時点の way 数の関係

する．以降は L2 キャッシュを用いて評価を行い，単にキャッシュと表記した場合は L2 キャッシュを示す．

4.2 評価結果・考察

4.2.1 LRU カウンタによる性能予測の評価・考察

LRU カウンタ方式の性能予測精度を評価する．評価はキャッシュの way 数を 1 増減した場合，way 数最大の場合の性能の予測結果について精度を評価する．

まず SPEC CINT2000 実行時に LRU カウンタを用いて，キャッシュの way 数を 1 増減した場合のヒット率の予測結果を図 8 示す．評価時はセクション毎にヒット率の予測を行い，評価結果は 100 セクション分の結果の平均値を示している．更に図 8 では SPEC CINT2000 の全プログラムの結果の平均を示している．

図中の予測 Hit 率 (-1) について way 数 x での値は，way 数 $x+1$ の時に way 数が 1 少ない状態つまり way 数 x の場合のキャッシュヒット率を予測した結果である．同様に予測 Hit 率 (+1) についても way 数が 1 多い状態のヒット率の予測結果である．各テストの結果は，キャッシュの効果が低い mcf を除く全プログラムで図 8 と同様の傾向を示している．

図 8 の結果に着目すると，ヒット率の予測のずれ幅は多い場合で 0.3%程度である．特にキャッシュの way 数が大きい状態で way 数増減時のヒット率予測を行うほど，予測結果が実際のヒット率に近い．これは way 数が増加するほどヒット率の変化幅が小さくなるためであると考えられる．特に way 数を 1 増やした場合のキャッシュヒット率は，前述の通り way 数を 1 減らした場合のキャッシュヒット数の減少幅と同数のキャッシュヒット数増加があると仮定して予測する．そのため LRU ブロックでのヒット数が減少する．way 数が多い状態では実際の値と予測値のずれが少なくなる．

また図 8 の結果から，way 数を 1 減らした場合のヒット率を予測する方がより精度が高いことが分かる．これは LRU カウンタの原理から，way 数が 1 少なかった場合に LRU キャッシュで余分にミスになるアクセスの回数を取得可能なことによる．

続いてヒット率と同条件で，連想度を 1 増減した場合の IPC の予測値と，実際にキャッシュが各 way 数の場合の IPC を比較した結果を図 9 に示す．IPC もヒット率同様に way 数が多いほど予測値と実際の IPC の差が小さくなる．また，way 数が多くなるにつれて 1-way あたりの IPC 増加への影響が小さくなる点もヒット率の場合と同様である．way 数が少ない場合は，キャッシュの way 数を 1 増減する毎のメモリアクセス回数の変化幅が大きい．つまり way 数を 1 減らした場合のメモリアクセス数の増加幅から，way 数を 1 増やした場合のアクセス数の減少幅を求める場合，より誤差が発生しやすい．

更に同条件でキャッシュが各 way 数であるときに，way 数が最大の 16 である場合のキャッシュヒット率，

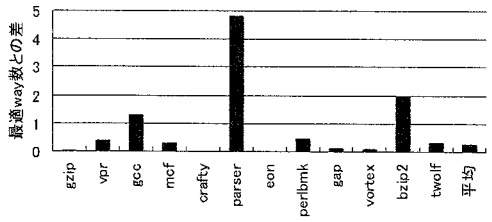

図 11 基準値のヒット率を満たすための最適 way 数との差

IPC を予測した結果を図 10 に示す．この結果では way 数が 16 の場合のヒット率，IPC が実際に way 数が最高の場合のヒット率，IPC である．この評価も前述の評価と同様に，way 数が多い場合に予測精度が高い．また，way 数が少ない場合に実際のヒット率，IPC との誤差が大きくなる．これらの誤差は，LRU カウンタによる性能予測を利用した，ヒット率から見た連想度需要の推定を高精度に行う妨げとなるため，精度の向上が今後の課題である．

4.3 LRU カウンタによる連想度需要予測の評価

LRU カウンタを用いた性能予測から，キャッシュヒット率を性能指標として一定のヒット率を満たすために必要な way 数の算出し，精度を評価する．基準値のヒット率を 99.9%，初期 way 数を 1，way 数の最低値を 1 として，セクションの区切り毎に基準値を満たす最適な way 数に調節する．図 11 に各セクションでの way 数と最適 way 数との差の平均を示す．

parser，bzip2，gcc が最適 way 数との差が 1 を超えているものの，それ以外では LRU カウンタ方式の連想度需要予測によって最適 way 数との差が 0.5-way 以下の範囲で way 数を調節している．つまり多くのプログラムでヒット率の予測，一定のヒット率を満たすために必要な連想度需要予測が機能している．一方で parser では最適 way 数との差が他のプログラムより大きい約 5 である．その原因は，parser ではセクション毎にヒット率から見た連想度需要が大きく変化することである．つまり LRU カウンタ方式による way 数の増減が最適 way 数の変化速度に追いつかず，最適 way 数との差が発生したと考えられる．

5. ま と め

本稿では way 単位で使用不使用を切り替え可能なキャッシュに対して way 数変化時の性能予測，一定の性能を満たすために必要な way 数を予測するためにキャッシュの LRU ブロックのヒット回数を計測する LRU カウンタ方式を提案した．また，LRU カウンタ方式により一定の性能を満たすために必要な way 数を予測して動的に way 数を変化させる方法を述べた．評価により，まず LRU カウンタ方式による way 数変更後の性能，特にキャッシュヒット率の予測に対して有用な精度を持つことを確認した．そしてヒット率から見た連想度需要を予測して，適量な way 数の動的な維持が可能であることを確認した．その結果 LRU カウンタ方式の性能低下を抑えたキャッシュ使用量の抑制，及び共有キャッシュの配分への有用性を確認した．

参 考 文 献

1) D. H. Albonesi: Selective cache ways: On-demand cache resource allocation, In Proceedings of the 32nd Annual International Symposium on Microarchitecture (MICRO-32), pp.248-259, 1999
2) D. T. Chiou: Extending the Reach of Microprocessors: Column and Curious Caching. PhD thesis, MIT, 1999
3) P. Ranganathan, S. Adve, N. Jouppi: Reconfigurable Caches and Their Application to Media Processing, In Proceedings of the 27th Annual International Symposium on Computer Architecture (ISCA-27), pp.214-224, 2000
4) G. E. Suh, L. Rudolph, S. Devadas: Dynamic Partitioning of Shared Cache Memory, Journal of Supercomputing, 28, pp.7-26, 2004
5) M. K. Qureshi, D. Thompson, Y. N. Patt: The V-way Cache: Demand Based Associativity via Global Replacement, In Proceedings of the 32nd Annual International Symposium on Computer Architecture (ISCA-32), pp.544-555, 2005
6) M. K. Qureshi, Y. N. Patt: Utility-Based Cache Partitioning: A Low-Overhead, High-Performance, Runtime Mechanism to Partition Shared Caches, In Proceedings of the 39th Annual International Symposium on Microarchitecture (MICRO-39), pp.423-432, 2006
7) J. Chang, G. S. Sohi: Cooperative Cache Partitioning for Chip Multiprocessors, In Proceedings of the 21st ACM International Conference on Supercomputing (ICS-21), pp.242-252, 2007
8) S. Kaxiras, Z. Hu, M. Martonosi: Cache Decay: Exploiting Generational Behavior to Reduce Cache Leakage Power, In Proceedings of the 28th Annual International Symposium on Computer Architecture (ISCA-28), pp.240-251, 2001
9) 田中 秀和, 井上 弘士, モシニャガ ワシリー: 低消費電力化を目的とした適応型ウェイ予測キャッシュとその評価, 電子情報通信学会技術報告, VLD2004-139, ICD2004-235(2003-3), pp.13-18, Mar. 2005
10) Hidekazu Tanaka and Koji Inoue: Adaptive Mode Control for Low-Power Caches based on Way-Prediction Accuracy, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol.E88-A, No.12, pp.3274-3281, Dec. 2005
11) SimpleScalar tool set, http://www.simplescalar.com/