Vol.2010-ARC-190 No.2
2010/8/3

メニーコアプロセッサ向き共有キャッシュ配分方式

小川周吾[†1]　平木敬[†1]

メニーコアプロセッサでは各コアが共有するラストレベルキャッシュ（LLC）を複数に分割することで，コア数の増加に伴うレイテンシ増加を防止できる．しかし LLC は複数のコアに共有されるため，分割された各 LLC の間で行われるラインの移動による競合ミスが発生する．本論文では我々が以前提案したマルチコア CPU 向けの共有キャッシュ配分方式である HFCA（History-Free Cache Allocation）に対して，アクセス頻度の低いラインを検出する RFF（Reference Frequency Filter）を追加した HFCA-RFF を提案する．HFCA-RFF はアクセス頻度の低いラインを選択して移動してきたラインの上書きを行うことで競合ミスを削減する．評価の結果，HFCA-RFF の適用によるキャッシュヒット率の低下幅は平均 0.2%であり，RFF によりアクセス頻度の低いラインが上書き対象とされた結果，多くのプログラムでは HFCA-RFF が与える影響が小さいことを確認した．

Shared Cache Allocation Method for Many-core Processors

Shugo Ogawa[†1] and Kei Hiraki[†1]

In many-core processors, shared last-level cache (LLC) is divided into several small cache units for avoiding the increase of access latency. However conflict misses are occurred by cache line migrations among cache units which organize shared LLC. In this paper, we propose a new method HFCA-RFF for avoiding conflict misses caused by cache line migrations. HFCA-RFF consists of HFCA (History-Free Cache Allocation) and RFF (Reference Frequency Filter). HFCA is our previous work which allocates ways in shared cache to reduce conflict misses. RFF is a new function for HFCA to detect rarely-accessed cache lines. HFCA-RFF inserts evicted data which is received from other cache units to rarely-accessed lines for reducing conflict misses. We evaluate the influence on cache hit rate which is caused by HFCA-RFF. Then we show that HFCA-RFF decrease cache hit rate only by 0.2% on the average by overwriting rarely-accessed cache lines.

1. はじめに

近年マルチコアプロセッサより多くのコアを持つメニーコアプロセッサが提案されている．メニーコアプロセッサではキャッシュミスの削減により各コアからのメモリアクセスによる性能低下を回避することが重要である．そのためマルチコアプロセッサと同様に，ラストレベルキャッシュをコア間で共有することによる容量効率の向上が効果的である．

一方でメニーコアプロセッサでは，多数のコアからの共有によるキャッシュの大容量化，複雑化により単一の共有キャッシュの実装が困難である．キャッシュの大容量化，複雑化はレイテンシの増大につながるため，キャッシュ内で一定のレイテンシの維持が困難な点が問題である．この問題に対して NUCA[1] に代表される，複数のバンクに分割されたキャッシュを用いてレイテンシの増加を防止する方式（以下，NUCA と総称）が提案されている．NUCA は各バンクへのアクセスレイテンシについて不均一性を許容することで，各バンクにおけるレイテンシを削減する．また NUCA 型のキャッシュを構成する各バンクは単独，または複数のバンク毎にそれぞれ独立したキャッシュとして動作する．各コアは複数の独立したキャッシュを跨って利用するため，NUCA を構成する各キャッシュの間でラインの移動が発生する．

しかし NUCA 型のキャッシュではラインの移動により，各バンク内で発生するコア間の競合ミスに加えて，バンク間で発生するラインの移動に伴った上書きによる競合ミスが発生する．NUCA 型のキャッシュを構成する各バンク間で発生する競合ミスの削減には，ラインの上書きによる新たなミスの発生を防ぐことが必要である．つまり他のバンクから移動してきたラインの上書き先として，アクセス頻度の低いラインを使用することが重要である．

本論文では我々が提案した共有キャッシュ配分方式である HFCA[2]（History-Free Cache Allocation）に対して，アクセス頻度の低いラインを検出する RFF（Reference Frequency Filter）を新規に追加した HFCA-RFF を提案する．HFCA-RFF は HFCA 単体による各バンク内の競合ミスの削減効果に加えて，HFCA-RFF によるバンク間のラインの移動で発生する競合ミスの削減効果を持つ．HFCA-RFF は RFF を利用して各コアの占有する領域と，特定のコアが占有していない共有領域に含まれるラインのアクセス頻度を調整することで，アクセス頻度の低いラインが上書き先に選ばれるように制御する．

†1 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

© 2010 Information Processing Society of Japan

Vol.2010-ARC-190 No.2
2010/8/3

2. 研究背景

2.1 NUCA 型共有キャッシュにおけるライン置換方式

従来より NUCA 型の共有キャッシュを構成するバンク間において，追い出されたラインの移動先となるバンク，及び追い出されたラインによる上書き対象のラインを決定する方式が提案されている．Kim らは NUCA を構成するバンク間におけるラインの動的な移動ポリシーを 2 種類提案している[1]．一つは追い出されたラインを他のバンクに移動せずに破棄する方式，もう一つは 1 回に限り追い出されたラインを任意のバンクに移動する方式である．また Speight らは他のバンクから追い出されたラインについて，”Invalid” または”Shared” の状態のラインを優先的に上書きして格納する方式を提案している[3]．

また移動先の各ラインに格納されたデータに対するアクセス頻度を考慮して上書き対象のラインを決定する方式が提案されている．塩谷らは追い出されたラインの移動先となるバンクを，全バンクに関する大域 LRU 情報をもとに決定する方式を提案している[4]．Lira らは追い出されたラインの移動先となりうる各バンクから LRU のラインのアクセス頻度が最も低いバンクを選択して，さらに移動先のラインのアクセス頻度が追い出されたラインより低い場合にのみバンク間のラインの移動を行う方式を提案している[5]．

2.2 従来の NUCA 型共有キャッシュにおけるライン置換方式の課題

従来の NUCA 型共有キャッシュに対するライン置換方式は，ラインの状態によらず一定のプロトコルに基づいて上書きするラインを決定する方式と，アクセス頻度の情報から上書きするラインを決定する方式の 2 種類に分類される．一定のプロトコルに基づいて上書き対象のラインを決定する方式の利点は，アクセス頻度の管理が不要であるためハードウェア量が少ない点である．しかしアクセス頻度の高いラインを上書き対象に選択することによる競合ミスの増加が課題である．

アクセス頻度の情報から上書きするラインを決定する方式の利点は，常にアクセス頻度の低いラインを上書き先に選択することにより競合ミスの増加が少ない点である．しかし管理するアクセス頻度のビット数に応じてハードウェア量が増加する点が課題である．また各ラインの移動先となるバンクの決定に，アクセス元のコアの情報は反映されない．そのため各コアがレイテンシが最小となるバンクを優先的に利用できず，アクセス頻度の高いラインが他のバンクから移動したラインによって，レイテンシの大きいバンクに追い出される点が課題である．さらに LRU にアクセス頻度の高いラインが存在する場合，他にアクセス頻度が低いラインが存在する場合でもラインの移動が不可能な点が課題である．

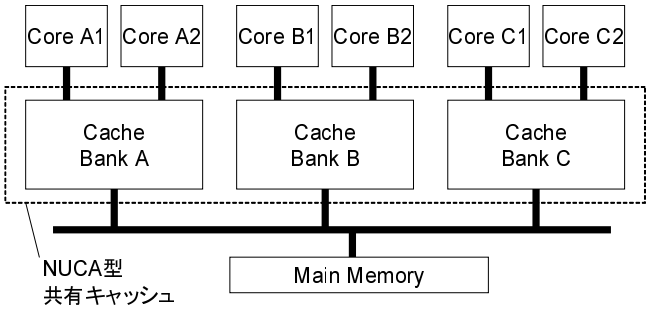

図 1 NUCA 型の共有キャッシュを持つプロセッサの基本構成例

本論文ではこれらの従来方式が持つ課題を解決するために，我々が以前提案した共有キャッシュ配分方式である HFCA[2] に，NUCA 型共有キャッシュの各バンクにおいて上書き対象のラインを選択するための RFF（Reference Frequency Filter）を追加した HFCA-RFF を提案する．HFCA-RFF は HFCA による共有キャッシュの配分機能を利用して各バンクにおいて各コアのキャッシュヒットに応じた占有領域を way 単位で作成する．これによりバンク内の競合ミス，及びアクセス頻度の高いラインがレイテンシの大きいバンクに追い出され，競合ミスが増加することを回避する．また HFCA-RFF は各コアの占有領域と共に追い出されたラインの上書き先となる共有領域を作成し，RFF によって共有領域がアクセス頻度の低いラインのみで構成されるように制御する．

3. HFCA-RFF

3.1 基本構成

本論文で提案する HFCA-RFF の適用先となる，NUCA 型の共有キャッシュを持つプロセッサの基本構成例は図 1 に示した通りである．プロセッサ内部には複数のバンクに分割された共有キャッシュを持ち，図 1 は 3 つのバンクに分かれた共有キャッシュを 6 コアで共有するプロセッサの例を示している．共有キャッシュに含まれる各バンクは全て同じセット数，way 数，ラインサイズを持ち，互いに独立した LRU キャッシュとして動作する．各コアは必ず 1 つずつ隣接したバンクを持ち，他のバンクと比べて短いレイテンシでアクセス可能である．また各バンクに対して隣接するコアは 1 個以上存在する．

各コアは共有キャッシュに対するアクセス時に，隣接するバンクから優先的にアクセス対象のラインの探索を行い，データを格納する．アクセスを行うコアの隣接するバンクにおい

© 2010 Information Processing Society of Japan

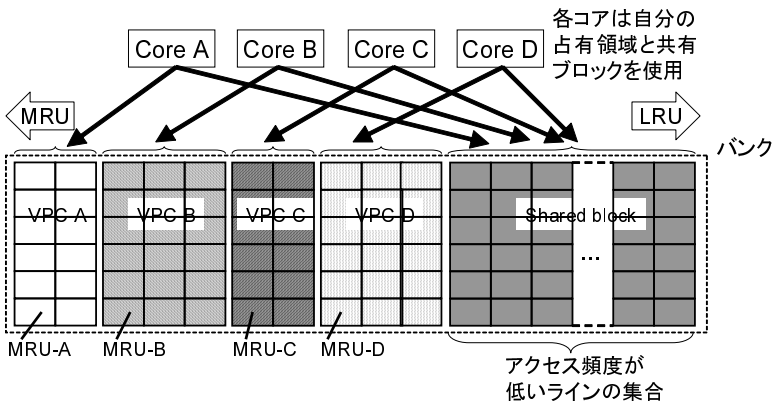

図 2 共有キャッシュを構成するバンクにおける各コアへの way の配分

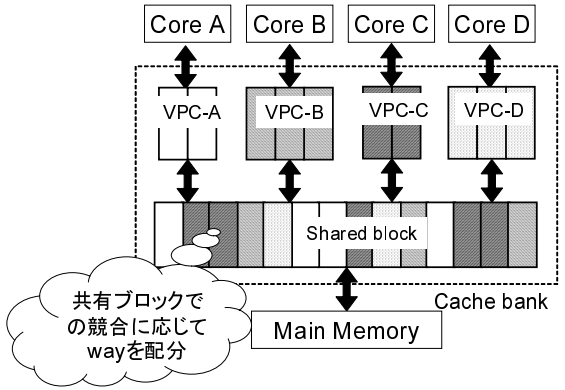

図 3 各バンク内で HFCA によって構成される仮想的なキャッシュ階層

てアクセス対象のラインが見つからなかった場合は，共有キャッシュを構成する他のバンクを探索してアクセスを行う．またアクセスを行うコアに対して隣接するバンクでラインの追い出しが発生した場合，追い出されたラインの移動先となるバンクを探す．追い出されたラインの移動は，追い出されたラインの上書き先となるラインが RFF によって検出されたバンクに対して行われる．

3.2 HFCA によるアクセス頻度に基づくキャッシュラインの分類

本論文で提案する HFCA-RFF では各バンクで他のバンクから追い出されたラインの上書き先となるラインを決定する．他のバンクから追い出されたライン上書き先には，競合ミスの増加を防ぐためにアクセス頻度の低いラインを選択する必要がある．HFCA-RFF は各バンクにおいて HFCA[2] を用いて各コアに対して競合ミスを削減するように共有キャッシュを way 単位で配分することで，バンク内で発生するコア間の競合ミスを防ぐと同時にバンク内のアクセス頻度の低いラインを特定する．

HFCA はアクセス履歴の情報を持たずに少量のハードウェアで共有キャッシュの配分を行う方式であり，次に示す 2 点の特徴を持つ．第 1 の特徴は，共有キャッシュを各コアの占有ブロック（以下，VPC: Virtual Private Cache）と共有ブロックに分割する点である（図 2）．各コアの VPC は way 数によって定義され，各セットの MRU から VPC の way 数分のアクセス頻度の高いラインが VPC として機能する．一方共有ブロックは各セットの LRU 側の VPC に含まれないラインであり，VPC から追い出されたデータが格納される．つまり共有ブロックは各バンクにおいて，仮想的に各コアの VPC とは異なるキャッシュ階層として機能する（図 3）．各コアのアクセス頻度の高いデータが VPC に格納されるように VPC の way 数を制御することで，コア間の競合ミスを防ぐ．

第 2 の特徴は，共有キャッシュ内に残された共有ブロックで発生する競合を検出して各コアの VPC の way 数を増やす点である．共有ブロックではキャッシュヒットするラインを他のコアが上書きすることで競合ミスが発生する．これに対して HFCA では共有ブロックにおけるキャッシュヒットを競合として扱い，VPC の way 数を増やしてヒットしたラインが VPC に含まれるようにすることで競合ミスを回避する．HFCA は上記の動作を繰り返すことで，コア間の競合ミスの発生を回避するように各コアに way を配分する．

つまり HFCA によって共有キャッシュを構成するバンク内のラインは，各コアのアクセス頻度の高いラインで構成された VPC と，アクセス頻度の低いラインの集合である共有ブロックに分割される．RFF は HFCA によって定義された共有ブロックに含まれるラインを他のバンクから追い出されたラインの上書き先に選択することで，バンク間で発生する競合ミスを最小化する．加えて HFCA を利用してバンクに隣接する各コアの占有ブロックを定義することで，アクセス頻度の高いラインがレイテンシの大きいバンクに追い出されることを防止する．

3.3 RFF によるアクセス頻度の判定基準の制御

RFF は HFCA によるキャッシュの占有ブロックと共有ブロックへの分割を利用してアクセス頻度の低いラインを判定する．しかしアクセス頻度の低いラインの判定基準となる共有

© 2010 Information Processing Society of Japan

ブロックの way 数は共有ブロックにおけるキャッシュヒットの発生に応じた VPC の way 数増加により変化する．よって HFCA が各コアの VPC に way の割り当てを開始した直後は，共有ブロックにアクセス頻度の高いラインが含まれることが問題である．

この問題に対してアクセス頻度の低いラインの判定基準である共有ブロックについて，常に各ラインのアクセス頻度が一定以下になるように制御する必要がある．RFF は VPC に含まれるラインについてアクセス頻度を制御することで，少量のハードウェアで共有ブロックに含まれるラインのアクセス頻度を制御する．具体的には，一定以上のアクセス頻度のラインが全て VPC に含まれるように各コアの VPC の way 数を制御する．そのため一定以上のアクセス頻度のラインを格納するために必要な VPC の way 数を調べる必要がある．

HFCA-RFF では VPC に含まれるラインのアクセス頻度と VPC の way 数の関係を調べるために，バンクを共有する各コアに対して VPC 内部で実際にアクセスされたラインについての最大 way 数を常に記録するレジスタを新たに追加する．例えばこのレジスタの値が a である場合，記録を開始してからレジスタを参照するまでに VPC において a-way より MRU 側のラインのみがヒットしたことを表す．このとき記録を開始してからレジスタを参照するまでのキャッシュヒットの回数を n，各キャッシュヒットで a-way より LRU 側のラインがヒットする確率を p とすると，n 回のキャッシュヒットにおいて a-way より LRU 側のラインが 1 回もヒットしない確率 P は以下の式（1）で表される．

$$P = (1-p)^n \tag{1}$$

式（1）において確率 P は p，n が大きいほど減少する．ここでキャッシュヒットしたラインの最大 way 数が a であるという前提により，レジスタを参照するまでのキャッシュヒット回数 n を増やすほど，確率 p の期待値が小さくなることが分かる．

つまり RFF は新たに追加したレジスタから得られるヒットしたラインの最大 way 数の結果を VPC の way 数に設定し，レジスタを参照するまでのキャッシュヒット回数 n を増減することで，VPC の way 数を a-way とした場合に VPC の外部で発生するキャッシュヒットの割合 p を制御できる．具体的にはレジスタを参照するまでのキャッシュヒット回数を増やすことで，VPC にはアクセス頻度が低いラインのみが含まれるように制御する．よって共有ブロックに含まれるラインのアクセス頻度が低くなるため，RFF が上書き対象に選択するラインのアクセス頻度の期待値が低下することで，他のバンクから追い出されたラインによる上書きが原因となる競合ミスを低減できる．

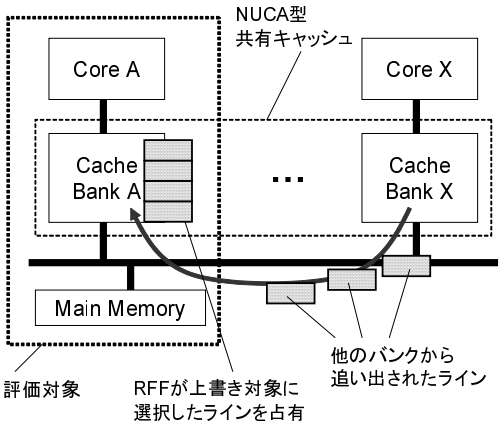

図 4 評価対象とシミュレーション環境

4. 性能評価

4.1 評価環境

本論文で提案した HFCA-RFF を適用して他のバンクから追い出されたラインによる上書きが継続的に行われる状態での性能への影響について，シミュレータ上でキャッシュヒット率，使用ライン数を比較する．評価は図 4 に示した，単体のコアとコアに隣接する NUCA 型共有キャッシュのバンクに対して行い，図 4 の評価対象に含まれるバンクに対してバンク間のラインの追い出しが発生しない場合（base），及び RFF により上書き対象に選択されたラインが，他のバンクから追い出されたラインによって全て占有された状態を比較する．HFCA-RFF は SESC[6] ベースのシミュレータ上に実装を行い，HFCA-RFF の適用時はキャッシュヒットしたラインの最大 way 数を取得するキャッシュヒット回数の間隔が 500 回（RFF-500），1000 回（RFF-1000），2000 回（RFF-2000）の場合について評価を行う．

コア及びキャッシュに関するシミュレーションパラメータはは特に断りのない限り表 1 に示した値を使用する．各コアの L1 キャッシュは各コアに含まれている一方，L2 キャッシュは各コアが隣接する NUCA 型共有キャッシュのバンクである．各コアにおいて L2 キャッシュから追い出されたラインは，他のバンクにおいて RFF によって上書き先となるラインが検出された場合にのみバンク間でラインの移動が行われる．一方上書き先となるラインが

© 2010 Information Processing Society of Japan

表 1 コアとキャッシュの設定

コアの構成	
Instruction set	MIPS
L1 cache	Instruction/Data 独立 32KB/2-way/LRU 置換 line size 32B/1-cycle latency
キャッシュのバンク・メモリの構成	
Cache bank	Instruction/Data 共有 2MB/16-way/line size 32B/11-cycle latency
Memory	300-cycle latency

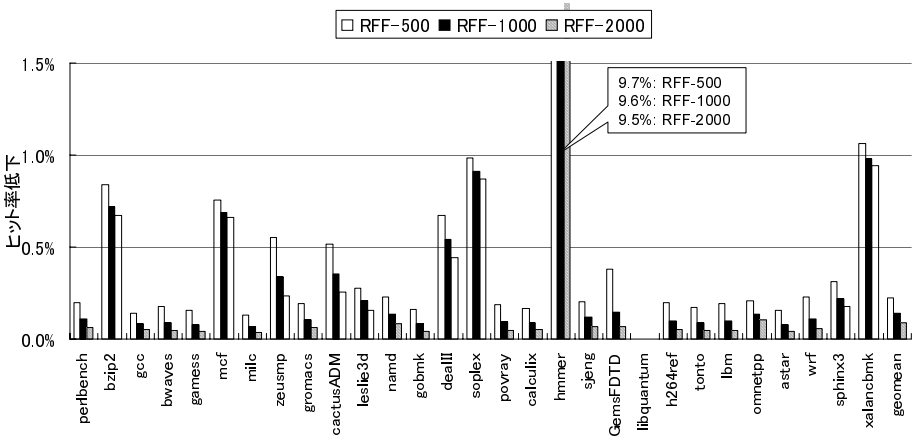

図 5 HFCA-RFF 適用時のキャッシュヒット率の減少幅

存在しない場合は，L2 キャッシュから追い出されたラインの，他のバンクへの移動は発生しない．

ベンチマークは SPEC CPU2006 のうち実行時間の短い specrand を除いたプログラム，入力データは ref を用いる．各コア上のプログラムは，それぞれ先頭 5G 命令スキップ後に 500M 命令を実行して評価を行う．

4.2 HFCA-RFF によるキャッシュヒット率への影響

まず HFCA-RFF 適用時の SPEC CPU2006 の各テストにおける，base と比べたキャッシュヒット率の低下幅を図 5 に示す．横軸は SPEC CPU2006 の各テスト，縦軸は HFCA-

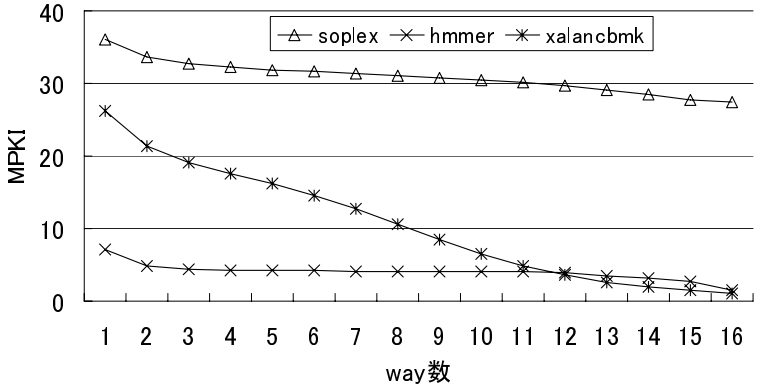

図 6 ヒット率の低下幅が大きいテストにおける way 数とミス発生頻度の関係

RFF 非適用時のキャッシュヒット率からの低下幅を表す．まず全テストにおけるキャッシュヒット率の平均低下幅は RFF-500 で 0.2%，RFF-2000 では 0.1%である．また多くのテストではキャッシュヒット率の低下幅が最も大きい RFF-500 の場合でも 0.5%未満である．つまり多くの場合は HFCA-RFF によるキャッシュヒット率への影響は小さく，RFF は他のバンクから追い出されたラインの上書き先として，アクセス頻度の低いラインを選択することで競合ミスを低減していると推測できる．また全テストにおいて，最大 way 数を取得する間隔を長くするほどキャッシュヒット率の低下幅が縮小している．よって最大 way 数を取得する間隔を調整することで，RFF が他のバンクから追い出されたラインの上書き先として選択するラインのアクセス頻度の制御が可能であることが分かる．

一方で図 5 に示した通り，soplex，hmmer, xalancbmk では他のテストに比べてキャッシュヒット率の低下幅が大きい．特に hmmer では RFF-500 の場合に約 9.7%，RFF-2000 の場合に約 9.5%の低下幅であり，RFF におけるヒットしたラインの最大 way 数の取得間隔の設定によらず大きくキャッシュヒット率が低下する．これらのテストにおいてキャッシュヒット率が大きく低下する原因について，図 6 を用いて説明する．図 6 はキャッシュのセット数を変更せずに，各テストの使用するキャッシュの way 数（横軸）のみを変更した場合の，キャッシュミスの発生頻度を MPKI（縦軸，Misses Per Kilo Instructions）で表している．これらのテストは特に使用する way 数が多い状態では，キャッシュの way 数の増加

© 2010 Information Processing Society of Japan

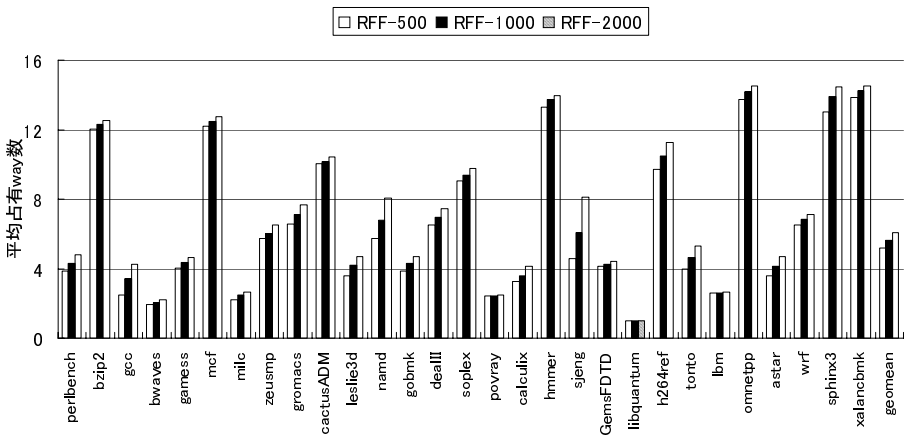

図 7 HFCA-RFF 適用時の各テスト実行時の隣接コアの占有 way 数

に従い常にキャッシュミスが減少する点が共通している．つまり HFCA-RFF によりキャッシュヒット率が大きく低下したテストは使用可能な way 数の制限が少量であっても影響を受けやすく，他のテストより大きくキャッシュヒット率が低下したと推測できる．

4.3 HFCA-RFF によるキャッシュラインの占有 way 数変化

続けて HFCA-RFF の適用時にバンク内で，各テストを実行するコアが占有するキャッシュラインの平均 way 数を図 7 に示す．横軸は SPEC CPU2006 の各テスト，縦軸は HFCA-RFF により配分された占有するキャッシュラインの平均 way 数を表す．図 7 の結果は，RFF が未適用の場合はバンク内の全ての way（16-way）を使用する一方でいずれのテストでも占有する way 数が減少して RFF による他のバンクから追い出されたラインの上書き先のラインが確保されていることを表している．また RFF におけるヒットしたラインの最大 way 数の取得間隔が大きくなるほど，各テストを実行するコアが占有する way 数が増加して，図 5 で示した通りキャッシュヒット率の低下幅も削減されることが分かる．

一方図 5 の結果でキャッシュヒット率の低下幅が大きいテストと図 7 における占有 way 数を比較すると，他のテストよりキャッシュヒット率の低下幅が大きい bzip2，mcf，hmmer，xalancbmk では占有 way 数も大きく，LRU に近いラインに対して頻繁にアクセスを行うテストでは HFCA-RFF による性能低下の影響を受けやすいと推測できる．しかし占有 way 数の大きい omnetpp，sphinx3 についてのキャッシュヒット率の低下幅は他のテストと同程度である．つまり占有 way 数が大きく，LRU に近いラインに対して頻繁にアクセスを行うテストの全てが HFCA-RFF による性能低下の影響を受けるわけではないことが分かる．

5. 関 連 研 究

Kim らは大規模なキャッシュにおけるレイテンシの問題を解決するために，複数のバンクによって構成されるキャッシュである NUCA を提案している[1)]．Kim らはバンク間のラインの動的な移動を行わない S-NUCA と，ラインの動的な移動を行う D-NUCA の 2 種類のアーキテクチャを定義し，D-NUCA におけるバンク間のラインの動的な移動ポリシーを 2 種類提案している．また Kandemir らによって追い出されたラインの移動先を各バンクに付加された優先度によって決定し，複数コアが共有するラインを各コアからのアクセス頻度で最適なバンクに移動する方式が提案されている[7)]．しかしこれらの NUCA におけるキャッシュラインの移動ポリシーでは，HFCA-RFF に比べて移動先となるキャッシュラインのアクセス頻度を考慮せずに移動を行うため，キャッシュラインの移動による競合ミスの増加が課題である．

これに対して Lira らは各ラインについてアクセス頻度を管理し，バンクから追い出されたラインをアクセス頻度をもとに，他のバンクの LRU のラインから選択する方式を提案している[5)]．この方式ではアクセス頻度の低いラインに上書きを行うことで，バンク間のラインの移動に伴うミスの増加を回避する．しかしこの方式は HFCA-RFF に比べて大規模なハードウェアが必要である点，各コアからのアクセスのレイテンシを考慮したラインの移動が行われない点が課題である．

また NUCA ではなく，マルチコア CPU 内の各コアが独立したキャッシュを持つアーキテクチャにおいて，キャッシュ間のラインを配置，移動するための方式が提案されている．Speight ら[3)] はバンクから追い出されたラインの格納先として，コヒーレンスの状態が”Invalid”，見つからない場合は”Shared” なラインを他のバンクから探す方式を提案している．しかしこの方式は NUCA における複数のライン移動ポリシー[1) 7)] と同様に，HFCA-RFF に比べて移動先となるキャッシュラインのアクセス頻度を考慮せずに移動を行うことによる競合ミスの増加が課題である．

これに対して塩谷らはマルチコア CPU の各コアが持つキャッシュへのアクセスに関する大域 LRU 情報を追加して，キャッシュからラインが追い出された場合の移動先を大域 LRU 情報をもとに決定する方式を提案している．この方式は Speight らの方式[3)] に比べてアクセス頻度情報を持つことで，アクセス頻度の高いラインの上書きによるミスの増加を回避で

© 2010 Information Processing Society of Japan

きる利点を持つ．しかしこの方式は HFCA-RFF に比べて，プロセッサの全域に広がる大規模なハードウェアが必要であるため，コア数の増加によりハードウェアの実装が困難になる点が課題である．

Dybdahl らは各コアのキャッシュを private な領域と shared な領域に分類し，private な領域を各コアのキャッシュ使用状況に応じて変更し，各コアの shared な領域を共有キャッシュとして使用するパーティショニング方式を提案している[8]．この方式は各コアのキャッシュを private な領域と shared な領域に分ける点が HFCA-DFF と共通している．しかし shared な領域におけるキャッシュラインの配置，移動に関する提案が行われていない点が課題である．

6. ま と め

本論文では我々が以前提案したマルチコア CPU 向けの共有キャッシュ配分方式である HFCA[2] を拡張して，NUCA 型の共有キャッシュにおける追い出されたラインの格納に適したラインを選択する方式である RFF（Reference Frequency Filter）を追加した HFCA-RFF を提案した．HFCA-RFF は HFCA による共有キャッシュの配分機能を利用して NUCA 型共有キャッシュに含まれる各キャッシュについて，レイテンシが短い隣接したコアに対してキャッシュ使用量に応じた占有領域を割り当てる．また HFCA-RFF は各コアの占有領域と共に追い出されたラインの上書き先となる共有領域を作成し，RFF によって共有領域にアクセス頻度の低いラインのみが含まれるように制御することで，ラインの上書きに伴う競合ミスの増加を回避する．評価の結果，HFCA-RFF の適用によるキャッシュヒット率の低下幅は平均 0.2%であり，RFF によりアクセス頻度の低いラインが上書き対象とされた場合に多くのプログラムでは HFCA-RFF が与える影響が小さいことを確認した．以上の評価から本論文では NUCA 型共有キャッシュを構成する各キャッシュにおいて，キャッシュ間のライン移動に伴う競合ミスの削減に対する HFCA-RFF の有用性を示した．

参 考 文 献

1) Kim, C., Burger, D. and Keckler, S.W.: An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches, *Proc. 10th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 211–222 (2002).
2) 小川周吾，入江英嗣，平木　敬：アクセス履歴の不要なマルチコア CPU 向け共有キャッシュ配分方式，先進的計算基盤システムシンポジウム SACSIS2010，pp.267–276 (2010).
3) Speight, E., Shafi, H., Zhang, L. and Rajamony, R.: Adaptive Mechanisms and Policies for Managing Cache Hierarchies in Chip Multiprocessors, *Proc. 32nd Annual International Symposium on Computer Architecture*, pp.346–356 (2005).
4) 塩谷亮太，ルォンディンフォン，入江英嗣，五島正裕，坂井修一：マルチコア・プロセッサの不均質共有キャッシュにおける LRU 大域置き換えアルゴリズム，先進的計算基盤システムシンポジウム SACSIS2006，pp.23–31 (2006).
5) Lira, J., Molina, C. and González, A.: The Auction: Optimizing Banks Usage in Non-Uniform Cache Architectures, *Proc. 24th ACM International Conference on Supercomputing*, pp.37–47 (2010).
6) Renau, J., Fraguela, B., Tuck, J., Liu, W., Prvulovic, M., Ceze, L., Sarangi, S., Sack, P., Strauss, K. and Montesinos, P.: SESC simulator (2005). http://sesc.sourceforge.net.
7) Kandemir, M., Li, F., Irwin, M.J. and Son, S.W.: A Novel Migration-Based NUCA Design for Chip Multiprocessors, *SC '08: Proc. of the 2008 ACM/IEEE conference on Supercomputing*, pp.1–12 (2008).
8) Dybdahl, H. and Stenström, P.: An Adaptive Shared/Private NUCA Cache Partitioning Scheme for Chip Multiprocessors, *Proc. 13th International Symposium on High-Performance Computer Architecture*, pp.2–12 (2007).

© 2010 Information Processing Society of Japan