情報処理学会論文誌　コンピューティングシステム　Vol.12　No.3　76–86 (July 2019)

適応型Stubbornキャッシュマネジメント手法の提案

野村 隼人[1,a)]　入江 英嗣[1,b)]　坂井 修一[1,c)]

受付日 2018年12月4日, 採録日 2019年4月4日

概要：プロセッサへのデータ供給能力は性能上最も重要な要素の 1 つであり，これを支えるため近年のプロセッサは大容量の Last Level Cache（LLC）を備えている．キャッシュ容量が大きくなるほど，再参照間隔のより長いキャッシュラインを的確に残すようなキャッシュマネジメントが求められるが，これは簡単なハードウェアで行うには難しい課題であり，現状，LLC には多くのデッドブロックが含まれている一方で，追い出しによるミスが発生していることが知られている．本研究では，ライン追い出しを一時的に凍結し，長期保持による統計的なヒット数向上を可能とする Stubborn 戦略をベースとして，その活用タイミングを適応的に決定する手法を提案し，性能向上を最大化させながら，性能低下の発生を抑えることを実現する．SPEC CPU 2006 からメモリセンシティブな 12 本のベンチマークをシミュレーションした評価では，LRU に対して最大 42.3%，幾何平均で 3.8%の性能向上を示した．

キーワード：キャッシュマネジメント，置き換えアルゴリズム，Stubborn 戦略，追い出し不可属性，適応型手法

Proposal of an Adaptive Stubborn Cache Management

Hayato Nomura[1,a)]　Hidetsugu Irie[1,b)]　Shuichi Sakai[1,c)]

Received: December 4, 2018, Accepted: April 4, 2019

Abstract: The ability to supply data to processor core is one of the most important factors in performance, and in recent years the processor has large capacity Last Level Cache (LLC). As cache capacity grows, cache management is required that can correspond to longer intervals re-reference accesses, however this is a difficult task with simple hardware. On the other hand, LLC has many dead blocks, but it also causes mistakes when replacement. In this research, we propose an adaptive cache management that based on Stubborn strategy. We evaluate our proposal method on a simulator with 12 memory-sensitive benchmarks of SPEC CPU 2006. The results show that our proposal method achieves outperforms LRU on IPC up to a maximum of 42.3%, geometric mean by 3.8%.

Keywords: cache management, replacement algorithm, stubborn strategy, “Don’t” evict attribute, adaptive

1. はじめに

プロセッサへのデータ供給能力は性能上最も重要な要素の 1 つであり，これを支えるため近年のプロセッサは大容量の Last Level Cache（LLC）を備えている．キャッシュ容量が大きくなることで，プログラマが意識せずとも得られるプログラムのメモリアクセス性能が良くなっている．しかしながら，大容量化した LLC には多くのデッドブロックが含まれており，その一方で追い出しによるミスが発生していることが知られている [1], [2]．これは，直近の履歴やその学習によって得られるキャッシュラインの保持の選択を行うような従来のアプローチが短期の再参照のヒット率を高めることを指向しているためで，大容量化した LLC をさらに活用するためには，従来の手法がターゲットとしていたものより再参照までの距離間隔（時間）がより長いキャッシュラインへのアクセスや，より大きいワーキングセットをターゲットとしたキャッシュマネジメントが必要

[1] 東京大学大学院情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo, Bunkyo, Tokyo 113–8656, Japan
[a)] nomura@mtl.t.u-tokyo.ac.jp
[b)] irie@mtl.t.u-tokyo.ac.jp
[c)] sakai@mtl.t.u-tokyo.ac.jp

となる．

先行する研究 [3] で，我々はLLCで生じる残されたキャッシュミスの多くを占めるものが再参照ミスであり，また10〜100 Mオーダの長期の命令数間隔にわたる再参照であることを明らかにした．また，そのようなミスに対処するためのキャッシュマネジメント手法として，Stubborn戦略を提案した．これは，キャッシュに挿入されたラインを一定の期間追い出さないよう指定する方式で，これにより，従来の手法では的確な追い出し選択が困難であった10〜100 M命令間隔の長期再参照が連続するワークロードや，激しいスラッシングによりキャッシュ容量がまったく活用されないワークロード等に有効であることを確認した．

Stubborn戦略は，従来の置き換えアルゴリズムとハイブリッドに実装され，キャッシュの一部の置き換えをアクセス履歴に関係なく凍結する．この性質から，もし不要なキャッシュラインをStubborn戦略で取り込んでしまった場合，キャッシュの容量効率を純減させることとなる．このため，プログラム傾向に合わせて，どのラインを，いつからいつまでStubborn戦略で運用するかを動的に判断する機構の実現がStubborn戦略実用上の課題となっていた．

そこで本論文では，Stubborn戦略が有効に機能する条件について調査し，さらなる性能向上と副作用の抑制を実現する方法を明らかにする．また，それに基づいて実行時ワークロード適応型のStubbornキャッシュマネジメント手法を提案する．シミュレータ上にこの提案手法Stubborn-HL，Stubborn-HL-Half，Stubborn-HL-Resetを実装した評価では，LRUに対して最大42.3%，幾何平均で3.8%の性能向上を確認した．また，先行研究における単純なStubborn戦略と比較した場合，今回提案した適応制御により2.8%の性能向上となり，特に，7.4%性能低下していたワークロードでは性能低下を抑え，性能を安定させる効果が確認された．

本論文の貢献は以下の3点である．

- 予備調査においてStubborn領域と性能の関係について傾向解析を行い，この結果からStubborn戦略が有効に機能する条件を明らかにした．
- 適応型Stubbornキャッシュマネジメント手法を提案し，ハードウェアで実現する手法を明らかにした．
- 評価において，性能向上と副作用の抑制の両立を確認した．

以下，本論文は次のように構成される．2章では置き換えアルゴリズムについて，従来手法とStubborn戦略を紹介する．3章では予備評価を行ってStubborn戦略の効果を解析し，さらなる性能向上のための道筋を立てる．4章では3章で示したStubborn戦略の持つポテンシャルを，実プロセッサで実行時の判断により得るための手法について提案する．5章では提案機構の実装について述べる．6章で提案手法を従来手法と比較して評価する．7章で関連研究について紹介し，8章でまとめを述べる．

2. キャッシュ置き換えアルゴリズム

2.1 履歴に基づく置き換えアルゴリズム

LRU

代表的なキャッシュ置き換えアルゴリズムにLRUがある．LRUは時間局所性のあるプログラムのキャッシュアクセスを支援する．しかしながら，過去に再参照のあったキャッシュラインを新たに挿入されたキャッシュラインと区別しないため再参照のあるキャッシュラインの保護が弱く，直近の履歴に惑わされてしまう．具体的には再参照頻度の高いアクセスがプリフェッチやスラッシング，ストリーミングアクセスにより追い出されてしまう．

RRIP

Jaleelらが提案するRe-Reference Interval Prediction（RRIP）[4] では，再参照のあったキャッシュラインの保護を優先する．RRIPではLRUオーダから替ってその再参照頻度を学習するためのカウンタ，RRPVを持つ．RRPVは挿入時に取れる値の中間値で挿入され，再参照があれば0でリセットされる．RRPVは時間経過でインクリメントされ，RRPVが最大値のキャッシュラインが追い出されることで，セット中に保持している間に再参照のあったキャッシュラインを保護している．この手法をSRRIPという．また，この研究ではスラッシングに耐性を持たせるため，RRPVの初期値をランダムに変更する手法BRRIP，またSRRIPとBRRIPを動的に切り替えて使用するDRRIPを同時に提案している．SRRIPとBRRIPはそれぞれワークロードの実行フェーズごとに優劣がつくため，どちらを適用するかを動的に判断するための手法としてSet Dueling Monitor（SDM）[5] が用いられており，SDMではPSELカウンタでどちらの手法が優秀かを判断する．

このように，RRIPは再参照アクセスの保護を意識した置き換えアルゴリズムであるが，再参照のあったキャッシュラインの学習を保持できる期間はそのキャッシュラインが挿入されてから追い出されるまでの間に限られ，BRRIPで対処できない量のスラッシングが発生した際は再参照のあったキャッシュラインのRRPV値もインクリメントされていくことで結果的に追い出されてしまう．

2.2 Stubborn戦略

直近の履歴に基づきキャッシュラインが保持できる程度の期間での再参照を保護するような従来の置き換えアルゴリズムに対して，より長い間隔での再参照の保護を目的としたStubborn戦略 [3] がある．これは，キャッシュライン追い出しを一時的に凍結し，長期に渡って追い出さないことで長期保持による統計的なヒット数向上を狙い，またスラッシング耐性を持たせることを目的としたキャッシュライン追い出し対象選択の戦略である．

その実現方法として，それぞれのキャッシュラインにStubborn フラグを持ち，これを追い出し不可属性とする．挿入時の判断により Stubborn フラグが ON になったキャッシュラインは，一定の長期間が経過するまでそのセット内で追い出しの対象とならない．これにより，従来手法のような再参照予測では判断できなかった距離の再参照をヒットさせられることができ，長期の再参照アクセスが占める割合の高いワークロード，スラッシングが連続するワークロードでは有効に機能する．

先行する研究 [3] で，我々はこの戦略を LRU，DRRIP の既存手法と融合し，従来手法では対処できなかった長期に渡る再参照とスラッシングによるミスを削減できることを確認している．この研究 [3] では，どのラインを追い出し不可とするかの判断として，先着順で挿入されたものを採用している．このようなシンプルな戦略と機構，実装にもかかわらず性能向上が得られている．

しかしながら，Stubborn 戦略はワークロードの特性によって効果が大きく変化する．従来のどの手法でも対応できなかったワークロードでミスとなるようなケースでその戦略により長期再参照をヒットさせ性能向上が得られる一方で，無駄なラインを保持し続ければ，キャッシュの容量効率を低下させてしまう．このため，Stubborn 戦略の組み込みにはどのキャッシュラインをどれだけの期間 Stubborn 戦略によって固定するかの決定が欠かせない．

3. Stubborn 戦略のポテンシャル調査

先行研究における Stubborn 戦略の実装では，LLC 容量の半分を従来手法，残りの半分を Stubborn 戦略に基づく置き換えアルゴリズムで使用していた．LLC をすべて Stubborn 領域にしてしまわずに，半分を従来手法のために残すのは，従来どおり短期の再参照ミスを防ぎ，またプリフェッチャが活用する領域を残すためである．

しかし，半数に固定することが最適とは限らない．ワークロードやフェーズごとに最適な Stubborn 戦略の適用度合いの強さがあると予想される．そこで，本研究では，LLC に占める Stubborn 領域の割合を変化させ，ワークロードごとにそれぞれ Stubborn 領域がどの程度の割合を占めるのが最適であるのかを調査した．

Stubborn 戦略の適用度合いの強さは，セットあたりの Stubborn フラグを立てられる way 数の上限数で変化させることができる．調査方法として，Stubborn キャッシュに占める Stubborn 領域の割合を指定できる置き換えアルゴリズム Stubborn-Manually を実装し，評価した．ここで，Stubborn 領域以外の残りの領域を LRU で制御するものを LRU based Stubborn-Manually，DRRIP で制御するものを DRRIP based Stubborn-Manually とする．

調査の結果として，次の**図 1**，**図 2** のグラフに LRU based Stubborn-Manually，DRRIP based Stubborn-Manually でのそれぞれのワークロード，使用 way 数ごとに，割当て way 数 0 のときの IPC を標準としたときの IPC の増減をパーセントで示す．凡例は Stubborn 領域として使用する way 数を意味する．評価環境は 6 章で後述のものと同様である．これらのグラフを見るにあたって，way 数 4 のパターンが先行研究での評価結果に相当する．

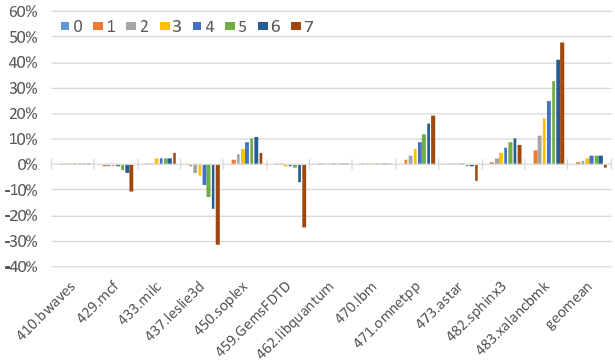

図 1　LRU based Stubborn-Manually での way 数別 IPC（way0 を標準とした増減差分）

Fig. 1　IPCs in LRU based Stubborn-Manually.

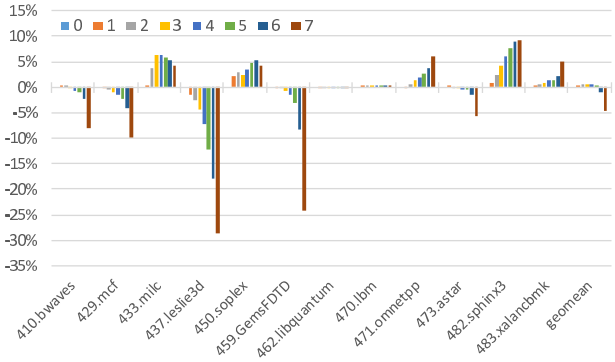

図 2　DRRIP based Stubborn-Manually での way 数別 IPC（way0 を標準とした増減差分）

Fig. 2　IPCs in LRU based Stubborn-Manually.

表 1　Stubborn-Manually 最大性能時 way 数

Table 1　Number of ways when max performance in Stubborn-Manually.

	way	
	LRU	DRRIP
410.bwaves	2	2
429.mcf	0	0
433.milc	7	4
437.leslie3d	0	0
450.soplex	6	6
459.GemsFDTD	0	0
462.libquantum	1	0
470.lbm	7	7
471.omnetpp	7	7
473.astar	3	1
482.sphinx3	6	7
483.xalancbmk	7	7

この結果から，多くのワークロードにおいてStubborn戦略との適応性が高いワークロードではStubborn戦略でセットを占める割合が高ければ高いほど性能への効果が高く，Stubborn戦略と相性の悪いワークロードもまたStubborn戦略でセットを占める割合が高ければ高いほど性能低下が大きいことが分かる．また，性能がStubborn領域の割合に影響されず，ほとんど変化のないワークロードもある．また，これらの結果の中で，最も高いIPCを記録したときのway数を**表 1** に示す．

4. Stubborn領域の動的な決定

前章より，Stubborn領域と従来手法を半分半分で動かすという先行研究に対して，Stubborn戦略との適応性が高いワークロードではStubborn領域をより多く取るべきであり，またStubborn戦略と相性の悪いワークロードではStubborn領域は与えないでいるべきだということが分かった．

これはつまり，先行研究における半分で固定されたStubborn領域の割合は中庸な選択であり中庸な結果をもたらしていたが，Stubborn領域として用いるway数を適応的に最小または最大のどちらかから選ぶ制御をすることでより高い性能を得ることができると予想される．

この考察をふまえ，本研究ではStubborn領域を動的に変動させる手法を提案する．つまり，Stubborn領域を大きくとるべきワークロードでは大きくとり，Stubborn領域を取るべきではないワークロードではとらない，という判断を実行時になすための機構を提案する．ここで，この判断機構を持つStubborn戦略の実装をStubborn-HL（High & Low）と名付ける．ここで，Stubborn領域を大きくするか小さくするかを判断するための機構としてSDMを使用する．SDMで用いるモニタリングセットの片方にStubborn領域を大きくとるポリシを，もう片側に小さくとるポリシを適用し，成績の良かった側がその他全体のセットに適用される，という動作をする．

LRUをベースとするStubborn-HL（以下LRU based Stubborn-HL）においては大きくするか小さくするかの2方向の判断をするだけで良いのでモニタリングセットは2セット，PSELカウンタも1つで実装可能となる．一方，DRRIPをベースとするStubborn-HL（以下DRRIP based Stubborn-HL）では，DRRIP内でSRRIPとBRRIPを切り替えるためにすでにSDMが用いられている．ここにStubborn-HLを適用するためには，さらに追加のモニタリングセットが必要となる．

ここで，原著[5]における本来のSDMは直近のアクセス傾向を重視して，PSELカウンタが更新される度に随時全体に反映する手法を選択・適用するが，本研究での予備実験において，同様にPSELに値の変動がある度に閾値を越えたかどうかの条件を判断しポリシの切り替えを随時行うという実装を試行した際，1M命令に満たない短い期間内でもポリシの切り替えが多発してしまうために，一度Stubbornフラグが有効になって保持されたラインが，ポリシの切り替わりによってStubbornフラグが無効となり追い出され，そしてまた短期間でStubbornが有効となり，といった動作の繰り返しにより，目的とするStubborn戦略による長期間の保持が行われなくなるという現象が見られた．そのため，本研究ではPSELカウンタの値によるSDMの切り替え判断を随時ではなく，一定以上の時間間隔（実行命令数間隔）を定め，その間隔ごとに行うことでこれを防ぐよう工夫した．

またSDMによるStubborn領域を適用するway数の判断について，3章の結果から，我々はway数を細かく変動させるよりも大きい値と小さい値のどちらかに切り替える手法を選択した．本研究ではStubborn領域をまったく適用しない0 wayとStubborn領域を可能な限り使用する7 wayの両極な2種類をモニタリングセットで動作させ，これらのうちモニタリング期間で優れた成績を出した方をその他のセットのポリシとして適用させる．つまり，0 wayとして指定されたセットではベースとなる置き換えアルゴリズムがそのまま動作し，7 wayとしたセットでは残りの1 wayにのみ追い出しが発生する．

このような設計をもってSDMを用いることで，本来短期的な決定を下すSDMをStubborn戦略のような長期的な判断が必要な機構と組み合わせることができるようになる．

5. Stubborn-HLの実装

5.1 Stubbornフラグビット

先行研究と同様に，キャッシュテーブル中の各キャッシュラインにStubbornフラグとして1 bitのフラグを追加する．挿入時のStubbornフラグを立てるか立てないかの判断は，先行研究と同様にセットあたりのStubbornフラグビットの許容数を満たすまでの間先着順でONにする方法とする．

5.2 SDMの実装

LRU based Stubborn-HLにおいて，10 bitのPSELカウンタを1つ，モニタリングセットを64セットあたり2セット設け，1つではStubbornフラグビットのセット数上限を7 wayとしたStubborn戦略動作，もう1つでは0 way，つまりStubborn戦略に基づく動作をしない通常のLRU動作を行う．それぞれのモニタリングセットでキャッシュミスが発生した際，0 way側ではPSELカウンタを+1，7 way側では−1する．構造を**図 3** に示す．

DRRIP based Stubborn-HLにおいて10 bitのPSELカウンタを4つ，モニタリングセットを64セットあたり4セット設け，順にSRRIP + Stubborn 0 way，SRRIP +

Stubborn 7 way，BRRIP + 0 way，BRRIP + 7way の 4 種の置き換えアルゴリズムで動作する．ここで，SRRIP + 7 way と BRRIP + 7 way は，残りの 1way での SRRIP・BRRIP 動作をすることになり，長期的に見ると結果としてほぼ同じ動作を行うことになるが，これは Stubborn 7 way のモニタリングセットを 1 組だけにした場合，0 way のモニタリングセット 2 組との PSEL カウンタの増減のやりとりになり不利になってしまうため，平等性のために 0 way になるモニタリングセットと 7 way になるモニタリングセットの数を合わせるためにこのような実装となっている．より具体的には SRRIP + 7 way と BRRIP + 7 way のセットについてはともに LRU + 7 way とする単純化が行える．

それぞれのモニタリングセットでキャッシュミスが発生した際，該当のモニタリングセットの PSEL カウンタを+3，他の PSEL カウンタを −1 する．

5.3　SDM 決定間隔

本研究では，SDM による支配的なポリシの選択を一定以上の間隔を空けて行う．この時間間隔を SDM 決定間隔と呼ぶ．実装においてはこの間隔を，実行命令数（経過命令数）を用いて指定する．このため，命令実行数を記録できるカウンタを持つ．いずれかのキャッシュラインでミスが生じた際にカウンタの値が決定間隔として指定された命令数を越えていれば，そのタイミングで最も成績の良いポリシ，LRU based Stubborn-HL では PSEL 値がマイナスまたは 0 であれば 7 way，プラスであれば 0 way，DRRIP based Stubborn-HL であれば PSEL 値の最も低いポリシをモニタリングセット以外のすべてのキャッシュセットのポリシとしてセットし，次の SDM 決定間隔が経過するまで保持する．また，実行開始から初回の SDM 決定間隔の経過までの間，Stubborn way 数は間をとって 4 とする．

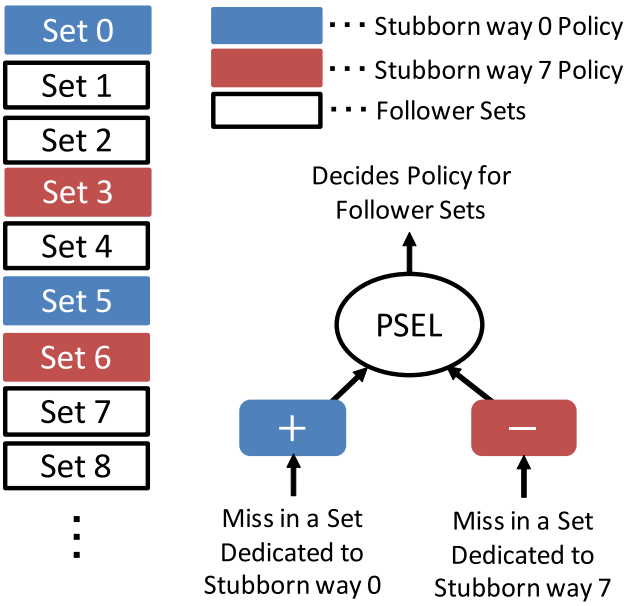

図 3　Stubborn-HL における SDM
Fig. 3　SDM for Stubborn-HL.

5.4　PSEL リセット

決定間隔を長く持たせた SDM において，いずれかのポリシに強く振れるタイミングがあった後，その後の支配的なポリシ判断においてその局所的なアクセス履歴による影響が強く長く残ってしまう場合がある．これを防ぐために，SDM 決定間隔が経過してポリシの全体反映をする度に PSEL カウンタの値をすべて 0 リセットする手法と，前回までの影響も残しつつ緩和するために PSEL カウンタの値を半分にする手法を考案し実装した．これらは手法名としてそれぞれ Stubborn-HL-Reset，Stubborn-HL-Half といったようにサフィックスの形で表現する．

6.　評価環境

6.1　ベースラインプロセッサ

プロセッサシミュレータ鬼斬弐 [6] にベースライン置き換えアルゴリズム，提案置き換えアルゴリズム，プリフェッチャを実装し，サイクルレベルの評価を行った．プロセッサの構成は**表 2** に示した値を用いた．この構成は Jaleel らによる評価環境 [4] に合わせたものである．シングルコアシングルスレッド実行で，プリフェッチの適用は LLC のみとした．キャッシュ階層間の包含関係については non-inclusive，non-exclusive な構成である．同一階層でのキャッシュレイテンシは Jaleel らによる評価環境と同様に一定のレイテンシとしてモデル化している．

6.2　置き換えアルゴリズム

置き換えアルゴリズムについて，LLC にベースラインとして Stubborn 戦略を使用しない No Stubborn，先行研究

表 2　ベースラインプロセッサパラメータ
Table 2　Baseline-processor parameters.

Processor	
Core	Alpha ISA, single core, single thread
Issue width	int:2, fp:2, mem:2
Inst. window	int:32, fp:16, mem:16
Branch pred	8 KB g-share
BTB	2 K entry, 4way
LSQ	96 entry
Cache Memory	
I/D L1 Cache	LRU, 32 KB, 4 way, 64 B line, 3 cycle latency
L2 Cache	LRU, 512 KB, 8 way, 64 B line, 10 cycle latency
L3 Cache (LLC)	2 MB, 8 way, 64 B line, 24 cycle latency Stream prefetcher (degr:16, dist:16)
Memory access	250 cycle latency

である Normal Stubborn，提案手法として Stubborn-HL，Stubborn-HL-Half，Stubborn-HL-Reset の 3 種を合わせた 5 種類の置き換えアルゴリズムを適用して実行する．

評価の章では，適応的な制御手法である Stubborn-HL に対して，way 数を固定する非適応的な手法 Stubborn-Manually においてワークロードごとにベスト値が得られる way 数設定で揃えた結果（以降，Stubborn-Manually-Best）を静的な手法の中でも，事前実行で確認できる中で最善の性能を得た場合の参考値として比較する．これらの計 6 種類での結果を，LRU をベースとしたもの，DRRIP をベースとしたものそれぞれで比較評価する．

パラメータとして，SDM で用いる PSEL カウンタは 10 bit，DRRIP で用いる RRPV は 2 bit，SDM 決定間隔は 20 M 命令とする．これらの値は予備評価を行って決定した．

6.3　実行ベンチマーク

SPEC CPU 2006 [7] のベンチマーク 29 本中，L3 ミスが性能に影響を与えているベンチマーク 12 本を評価対象とした．具体的には，表 2 のプロセッサパラメータにおいて LLC の置き換えアルゴリズムを LRU としストリームプリフェッチャを除いた構成で予備評価を行い，デマンドで発生するミスの数が MPKI で 5 以上となるものを抽出した．この選出は先行研究 [3] と同じ基準によるものである．ベンチマークは GCC4.5.3 でコンパイルし，最適化レベルは O2 とした．

シミュレーションでは先頭 10 G 命令スキップ後，続く 1 G 命令の区間について cycle accurate に実行し，測定した．

7.　評価

7.1　IPC

性能評価として，ベースラインとする置き換えアルゴリズム（LRU，DRRIP）の IPC（Instruction per Cycle）を 1 としたときの相対 IPC の増減をパーセントで示す．LRU をベースラインとしたものを**図 4**，DRRIP をベースラインとしたものを**図 5** に示す．

LRU ベースラインの評価（図 4）において，提案手法 Stubborn-HL-Reset により LRU 比で最大 42.3%，幾何平均で 3.8%の性能向上の性能向上が得られている．Normal Stubborn 比では 2.8%の性能向上が得られた．

Stubborn-Manually-Best に対しても，ワークロード 450.soplex において Normal Stubborn で生じていたベースラインからの 7.4%の性能低下を抑えた上で得られた成績である．

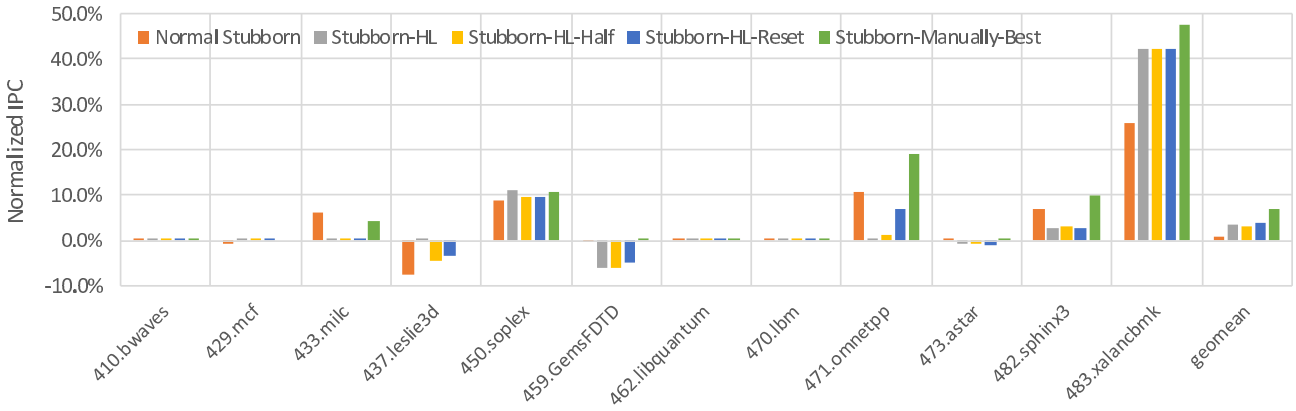

図 4　LRU ベースラインにおける相対 IPC 評価
Fig. 4　Normalized IPC (LRU baseline).

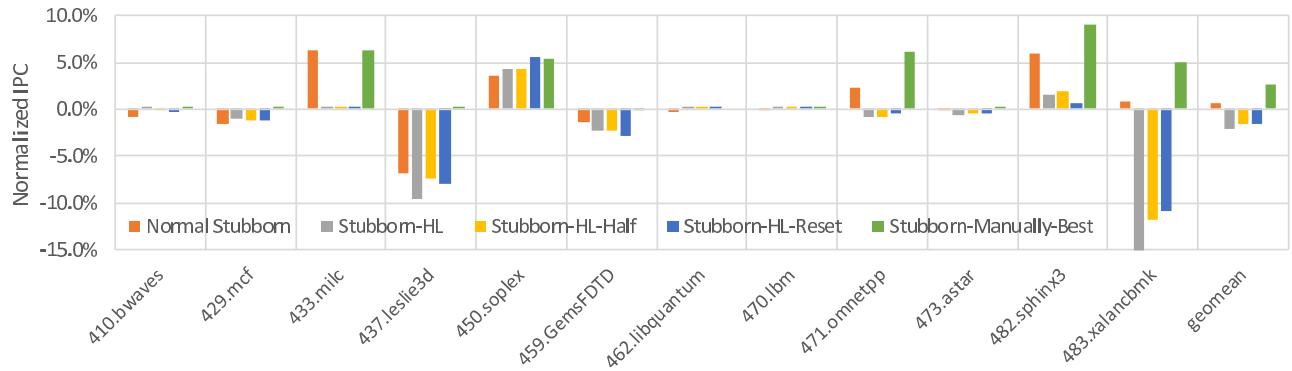

図 5　DRRIP ベースラインにおける相対 IPC 評価
Fig. 5　Normalized IPC (DRRIP baseline).

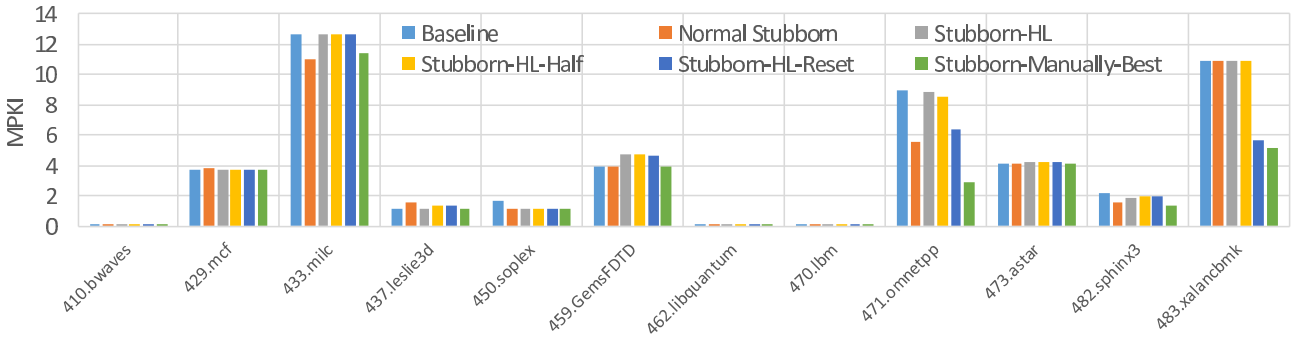

図 6　LRU ベースラインにおける MPKI 評価
Fig. 6　MPKI (LRU baseline).

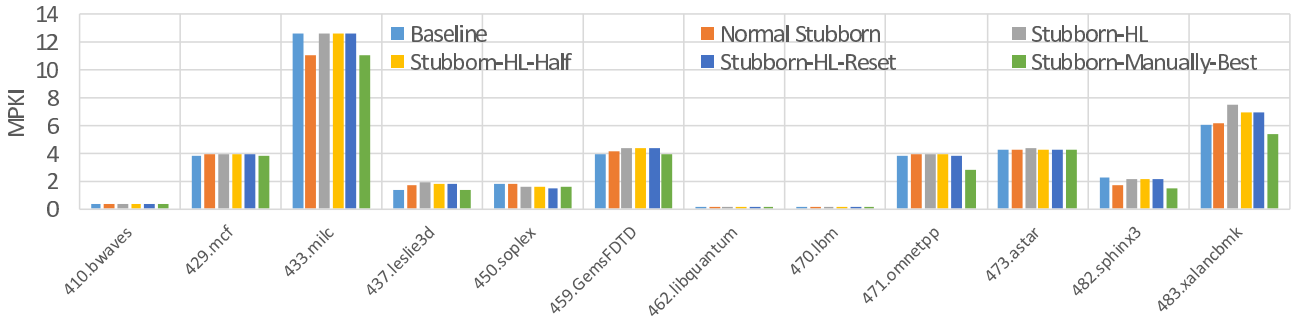

図 7　DRRIP ベースラインにおける MPKI
Fig. 7　MPKI (DRRIP baseline).

一方, DRRIP ベースラインの評価 (図 5) にて, Stubborn-HL では DRRIP 比で最大 5.6%の性能向上が得られたものの，幾何平均では 1.6%の性能低下が生じた．

LRU ベースラインの提案手法と DRRIP ベースラインの提案手法の比較では，LRU ベースラインで最善の LRU based Stubborn-HL-Reset が DRRIP ベースラインで最善の DRRIP based Stubborn-HL-Reset に対して 2.5%高く，DRRIP に対して 1.7%，Normal Stubborn DRRIP に対しても 0.9%の差で高い性能を示した．

7.2　MPKI

性能評価に付随して，キャッシュミス数を MPKI（Miss per Kilo Instruction）で示す．LRU をベースラインとしたものを**図 6**，DRRIP をベースラインとしたものを**図 7** に示す．ミスの削減率が IPC の向上率とおおむね対応しており，結果を裏付ける数値が出ている．

410.bwaves，462.libquantum，470.lbm において MPKI の値が極端に低いのは，これらのワークロードではプリフェッチがきわめて有効なためである．そして，この結果は，提案手法による Stubborn 戦略適用時にもプリフェッチを阻害しなかったことも意味している．

7.3　SDM の判断

SDM による Stubborn 領域の割当ての判断の評価として，それぞれの提案手法による way 数の判断がどの程度

表 3　Stubborn way 数（LRU ベースライン）
Table 3　Number of Stubborn way (LRU based).

Workload	Manually	HL	HL-Half	HL-Reset
410.bwaves	2.0	0.0	0.1	0.7
429.mcf	0.0	0.0	0.0	0.0
433.milc	7.0	0.0	0.0	0.6
437.leslie3d	0.0	0.1	2.7	2.4
450.soplex	6.0	7.0	5.7	6.0
459.GemsFDTD	0.0	7.0	5.4	4.4
462.libquantum	1.0	7.0	5.4	5.3
470.lbm	7.0	6.7	5.4	4.3
471.omnetpp	7.0	0.0	0.6	3.4
473.astar	3.0	7.0	6.7	5.3
482.sphinx3	6.0	1.9	2.0	1.9
483.xalancbmk	7.0	7.0	7.0	7.0

Stubborn-Manually-Best の判断に近似しているかを確認した．ここで，評価のため，一定時間間隔で way 0，7 で切り替えられた way 数の判断の Stubborn-Manually-Best に対する一致率を以下の手順により集計した．

(1) シミュレータ実行時，SDM 決定間隔で定めた命令数経過ごとに採択 way 数（0 or 7）を記録する．
(2) この記録を集計し，提案手法・実行ワークロードごとにその平均値を求める（**表 3**，**表 4**）．
(3) 2 で求めた平均値から Stubborn-Manually-Best での way 数を引く．
(4) 3 で求めた値の絶対値をとり，これを Stubborn-

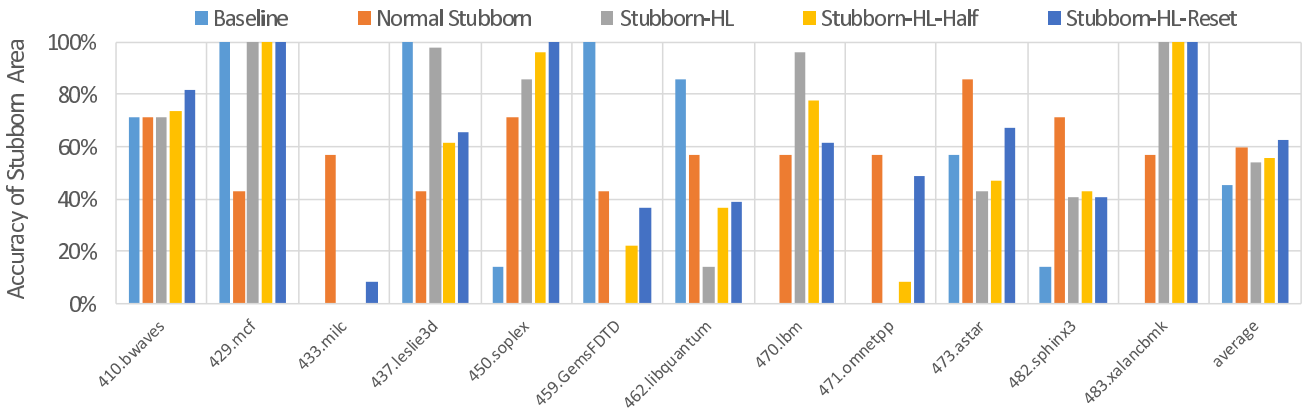

図 8　LRU ベースラインでの Stubborn 領域割合判断の一致率
Fig. 8　Concordance rate of Stubborn Area (LRU baseline).

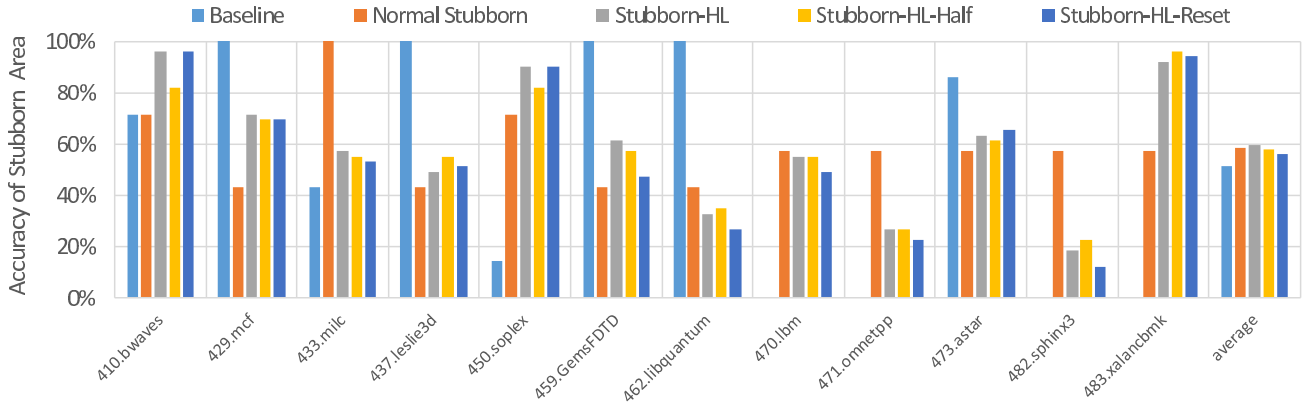

図 9　DRRIP ベースラインでの Stubborn 領域割合判断の一致率
Fig. 9　Concordance rate of Stubborn Area (DRRIP baseline).

表 4　Stubborn way 数（DRRIP ベースライン）
Table 4　Number of Stubborn way (DRRIP based).

Workload	Manually	HL	HL-Half	HL-Reset
410.bwaves	2.0	2.3	3.3	2.3
429.mcf	0.0	2.0	2.1	2.1
433.milc	4.0	1.0	0.9	0.7
437.leslie3d	0.0	3.6	3.1	3.4
450.soplex	6.0	5.3	4.7	5.3
459.GemsFDTD	0.0	2.7	3.0	3.7
462.libquantum	0.0	4.7	4.6	5.1
470.lbm	7.0	3.9	3.9	3.4
471.omnetpp	7.0	1.9	1.9	1.6
473.astar	1.0	3.6	3.7	3.4
482.sphinx3	7.0	1.3	1.6	0.9
483.xalancbmk	7.0	6.4	6.7	6.6

Manually-Best による判断 way 数からの差とする．

(5) 4 で求めた差を最大 Stubborn way 数の 7 で割り，これを Stubborn-Manually-Best からの乖離率とする．

(6) 1 から乖離率を引いて一致率とする（**図 8**，**図 9**）．

手順 2 の時点での各ワークロード・手法ごとの Stubborn-Manually-Best での各 way 数と提案手法での Stubborn 領域使用 way 数の平均値を LRU ベースラインのときのものを表 3，DRRIP ベースラインのときのものを表 4 に示す．この表から，求められている way 数に対する，提案手法によって判断した way 数の平均値の過不足が読み取れる．

また，手順 6 で求めた一致率を LRU ベースラインのときのものを図 8，DRRIP ベースラインのときのものを図 9 に示す．IPC，MPKI と合わせて見ると，Stubborn-Manually-Best での way 数の一致率が高いほどミスが少なく IPC が高い傾向にある，個々の例を見ていくと，LRU ベースラインでは，提案手法が Stubborn-Manually-Best に並んで高い成績を出した 450.soplex では，LRU ベースラインでは way 数を多くとるという傾向が一致しているとともに，高い IPC の再現も得られている．482.sphinx3 についても同様の傾向が見られる．

483.xalancbmk においては，すべての提案手法がすべての判断タイミングで 7 way の判断を下せており，Stubborn-Manually-Best の実行と高く一致する．Stubborn-Manually-Best と提案手法の間に残る性能差は初回の SDM 決定間隔までの間の挙動の差と，常時 0 way で稼働するモニタリングセットの片側で生じるロスによるものだ．一方で，DRRIP ベースラインでは 6.4〜6.6 と，傾向が一致していたにもかかわらず，IPC では 100%を下回る結果となった．これは，実行時に 0 way にする判断が

7 にする判断に混じったことで，実行の当初から保持されているべきキャッシュセットで 0 way の判断が降りるつどに追い出されてしまい，483.xalancbmk で生じる激しいスラッシングへの耐性を失ってしまったためである．このことから，483.xalancbmk においては LRU ベースラインでの提案手法の判断のように，1 G 命令期間中，Stubborn 領域に留めるキャッシュラインを一切更新しないことが有効であると分かった．この結果は，483.xalancbmk のような過密で長いスラッシングの連続するワークロードにおいては，キャッシュラインを入れ替えないことだけが置き換えアルゴリズムにできる対処であることも示している．

LRU ベースラインでの提案手法による 437.leslie3d の実行では，先行研究である Normal Stubborn で生じている性能低下を，提案手法では Stubborn 適用割合を動的に抑えて，性能低下も小さく抑え，克服することができている．このケースでは，プリフェッチとの親和性が高い 437.leslie3d のようなワークロードで，Stubborn 適用 way 数を 0 としたことでプリフェッチを阻害せず，ベースラインで得られていた性能を維持することができている．

このように，Stubborn-Manually-Best で低い Stubborn 適用割合を示すワークロードにおいて，提案手法により Stubborn 適用割合を低く抑えられたことでスラッシング耐性が得られ，プリフェッチとの親和性が得られた．

LRU ベースラインと DRRIP ベースラインでの比較では，LRU ベースライン（図 8）では Stubborn-Manually-Best の一致率を 100%としたとき，一致率は LRU に対して Normal Stubborn が 14.3%，Stubborn-HL-Reset が 17.2%の向上を果たした．DRRIP ベースライン（図 9）では DRRIP に対して Normal Stubborn が 7.1%，Stubborn-HL-Reset が 5.1%の向上となった．結果として，DRRIP ベースラインでの Stubborn 戦略適用より，LRU ベースラインでの Stubborn 戦略適用の方が SDM による判断がより良くなっている．これは，IPC での順列とも合致する．

7.4　ハードウェアコスト

ベースライン，先行研究，提案手法におけるハードウェアコストを評価する．**表 5** に予測機構とキャッシュテーブルで用いるハードウェア量，その容量に対するオーバヘッドを示す．ここで，予測機構とは SDM のための PSEL カウンタを意味し，キャッシュテーブルにおけるハードウェア量は LRU では LRU オーダ，DRRIP では RRPV，Stubborn 戦略ハイブリッドのポリシではそれらに Stubborn フラグを加えた置き換えアルゴリズム状態管理のための記憶領域を指す．これより，LRU ベースライン，DRRIP ベースラインともに，提案手法は 4 KB のテーブルサイズ増加，2 MB のキャッシュ容量に対して 0.2%のハードウェア増加コストで実現できることが分かる．このコストは得られた性能向上に対して十分に小さい．

表 5　Stubborn way 数（DRRIP ベースライン）
Table 5　Number of Stubborn way (DRRIP based).

Policy		Predictor Structures	Cache Tag Meta-data	Overhead for Capacity
LRU based	LRU	None	12 KB	0.6%
	Normal Stubborn	None	16 KB	0.8%
	Stubborn-HL	10 bit	16 KB	0.8%
DRRIP based	DRRIP	10 bit	8 KB	0.4%
	Normal Stubborn	10 bit	12 KB	0.6%
	Stubborn-HL	40 bit	12 KB	0.6%

IPC，MPKI，SDM による判断，ハードウェアコストを総合した結論としては，適用型の Stubborn 戦略を組み合わせた置き換えアルゴリズムとして LRU ベースライン，更新時 PSEL リセットの構成の提案手法，LRU based Stubborn-HL-Reset が本研究による最大の成果物である．

Stubborn-HL の 3 種の手法の中では，Stubborn-HL-Reset が Stubborn-HL，Stubborn-HL-Half に対して優位な結果を示した．これは，前回の区間までの PSEL の結果を残し，ポリシ判断で前期間までの履歴による影響が強く長く残る Stubborn-HL においては，5.4 節での見立てのとおり不利になったためである．この差は，IPC では特に 471.omnetpp の結果で顕著であることが分かる．これは，PSEL の値を半分にして継承する Stubborn-HL-Half でも同様の傾向が見られる．このことから，ポリシの更新期間として 20 M の区間を設けたことで，結果として区間をまたいでの PSEL の保持は不要であり，Stubborn-HL-Reset のようにそのときそのときの区間の成績に基づく判断が Stubborn 戦略適用の是非に重要な情報であったことが分かった．

8.　関連研究

関連研究として，最新の置き換えアルゴリズムおよびプリフェッチの一部，そして Cache Locking を紹介し，それぞれの立ち位置と本研究との関係を述べる．以下の PACMan [8]，SHiP [9]，Hawkeye [10] はいずれも RRIP の影響を受けた後続的な手法の置き換えアルゴリズムである．

PACMan

Wu らによる Prefetch-Aware Cache Management（PACMan）は，挿入時にデマンドによるものかプリフェッチによるものかを区別し，プリフェッチにペナルティを与えることでデマンドでしか挿入されないキャッシュラインの保護の強化を目的とした置き換えアルゴリズムである．

プリフェッチで挿入されたキャッシュラインについては，必要となるタイミングで再度プリフェッチにより挿入されることが予測されるためキャッシュに保護する必要性が低く，本研究においてもこれに習いプリフェッチによる挿入では Stubborn フラグを立てない実装となっている．

SHiP

DRRIP では SRRIP，BRRIP の切り替えによる RRPV の初期値変更を行い，これによりキャッシュラインごとの再参照予測としていた．これに対し，Wu らによる Signature-Based Hit Predictor（SHiP）では，SHiP と呼ばれる PC ベースの学習による再参照予測器により，挿入時に再参照の可能性を予測し，RRPV の初期値を判断している．学習におけるプリフェッチ対応を施した SHiP++ [11] がある．

Hawkeye

Jain らによる Hawkeye は再参照予測において OPT [12] を模した予測器を用いた手法である．真の OPT の再現にはすべての未来のアクセスを事前に知っている必要があるため実プロセッサでは再現できないが，Hawkeye における学習機 OPTgen では過去の時点から見て現在までのアクセス履歴を用いることで OPT の動作を模し，その判断を用いて現在から未来でも同様の結果が得られるものとして利用している．OPT の振舞いについてプリフェッチを考慮した Hawkeye+ [13] がある．

AMPM プリフェッチ

ストリーミングプリフェッチやストライドプリフェッチのような従来のプリフェッチではアクセス予測をメモリアクセスや命令アドレスの連続性を用いているが，こうしたプリフェッチャではアウトオブオーダ実行やループアンローディングのための最適化がなされたコードに対応できないことがある．そのようなワークロードの実行時でのプリフェッチに対処する手法に Access Map Pattern Matching Prefetch（AMPM）[14], [15] がある．Map 構造のアクセス履歴記録を用いてパターンマッチングを行い，アクセスの連続性に依存しない Order-Free なプリフェッチを実現している．

本研究の提案手法はこれらの関連研究と競合するものではなく直交する手法であり，置き換えアルゴリズムについては LRU や DRRIP のようにベースラインとして組み込んで共存することができる．

プリフェッチについても，本研究でのストリーミングプリフェッチとの共存の結果が示したように，提案手法により Stubborn 領域の判断が正しくなされることで，プリフェッチが優先して積極的に動くべきタイミングで Stubborn は無効（Stubborn 領域 0 way）になり，プリフェッチを阻害することなく共存することができる．

Cache Locking

キャッシュラインをロックするというアイデア自体はすでにあり [16]，商用ハードウェア上にも実現されている [17], [18]．Cache Locking は，保持し続けるデータと期間の指定を，プログラムソースコード上で人の手でアノテーションし，あらかじめ static に選ぶことを前提として，アクセス頻度が高いデータをキャッシュに止めておくことで性能向上を図り，あるいはハードリアルタイムシステムにおいてキャッシュを用いた際のリアルタイム性を保証するための手法である．実装としては，特権命令によりキャッシュラインを way 単位でロックする．

この手法と Stubborn 戦略の相違点は，Cache Lock は既知のワークロード，ワークセットを前提にプログラムに静的なアノテーションを必要とするもので，プログラマによるチューニングを前提としている．一方，Stubborn 戦略は実行時に動的に保持するラインを決定するものであるため，事前のアノテーションを必要とせず，実行するバイナリへの工夫は必要なく動的に判断を行えるという点にある．

Cache Locking のための機構と Stubborn 戦略のための機構は共有して利用するような構成をなすことが可能であり，Cache Locking によるアノテーションなしでの実行時にも Stubborn 領域に止めておくべきラインの選択においてこれらの研究の知見が活用できて，協調が見込める．

9. おわりに

Last Level Cache（LLC）の大容量化にともない，再参照間隔のより長いキャッシュラインを的確に残すようなキャッシュマネジメントが求められる．本研究では，直近の履歴やその学習によって得られるキャッシュラインの保持の選択とは異なるアプローチをとる Stubborn 戦略をベースとして，調査により Stubborn 戦略の適用領域の増減によりさらなる性能向上が得られることを明らかにした．さらに，その活用タイミングを適応的に決定する手法を提案し，これにより性能向上を維持したまま，性能低下の発生を抑えることを両立する機構を実現した．シミュレータによる評価では，LRU に対して最大 42.3%，幾何平均で最大 3.8%，性能向上の性能向上を示した．先行研究における単純な Stubborn 戦略と比較した場合 2.8%の性能向上となり，特に，7.4%性能低下していたワークロードでは性能低下を抑え，提案の適応制御により性能を安定させる効果が確認された．

参考文献

[1] Khan, S.M., Tian, Y. and Jimenez, D.A.: Sampling Dead Block Prediction for Last-Level Caches, *Proc. 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, pp.175–186 (2010).

[2] Faldu, P. and Grot, B.: Leeway: Addressing Variability in Dead-Block Prediction for Last-Level Caches, *Proc. 26th International Conference on Parallel Architectures and Compilation Techniques* (*PACT*), pp.180–193 (2017).

[3] Nomura, H., Katchi, H., Irie, H. and Sakai, S.: “Stub-

born" strategy to mitigate remaining cache misses, *Proc. IEEE 34th International Conference on Computer Design* (*ICCD*), pp.388–391 (2016).
[4] Jaleel, A., Theobald, K.B., Steely, Jr., S.C. and Emer, J.: High Performance Cache Replacement Using Re-reference Interval Prediction (RRIP), *Proc. 37th Annual International Symposium on Computer Architecture*, pp.60–71 (2010).
[5] Qureshi, M.K., Jaleel, A., Patt, Y.N., Steely, S.C. and Emer, J.: Adaptive Insertion Policies for High Performance Caching, *Proc. 34th Annual International Symposium on Computer Architecture*, pp.381–391 (2007).
[6] Watanabe, K., Ichibayashi, H., Goshima, M. and Sakai, S.: Design of Processor Simulator 'Onikiri', *Advanced Computing Systems and Infrastructures* (*SACSIS*), pp.194–195 (2007).
[7] Standard Performance Evaluation Corporation: SPEC CPU 2006 Benchmark Suite (2006) (online), available from ⟨https://www.spec.org/cpu2006/⟩.
[8] Wu, C.-J., Jaleel, A., Martonosi, M., Steely Jr., S.C. and Emer, J.: PACMan: Prefetch-aware Cache Management for High Performance Caching, *Proc. 44th Annual IEEE/ACM International Symposium on Microarchitecture*, pp.442–453 (2011).
[9] Wu, C.-J., Jaleel, A., Hasenplaugh, W., Martonosi, M., Steely Jr., S.C. and Emer, J.: SHiP: Signature-based Hit Predictor for High Performance Caching, *Proc. 44th Annual IEEE/ACM International Symposium on Microarchitecture*, pp.430–441 (2011).
[10] Jain, A. and Lin, C.: Back to the Future: Leveraging Belady's Algorithm for Improved Cache Replacement, *Proc. ACM/IEEE 43rd Annual International Symposium on Computer Architecture* (*ISCA*), pp.78–89 (2016).
[11] Young, V., Chou, C.-C., Jaleel, A. and Qureshi, M.K.: SHiP++: Enhancing Signature-Based Hit Predictor for Improved Cache Performance (2017).
[12] Belady, L.A.: A study of replacement algorithms for a virtual-storage computer, *IBM Syst. J.*, Vol.5, No.2, pp.78–101 (1966).
[13] Jain, A. and Lin, C.: Hawkeye: Leveraging Belady's Algorithm for Improved Cache Replacement (2017).
[14] Ishii, Y., Inaba, M. and Hiraki, K.: Access Map Pattern Matching for Data Cache Prefetch, *Proc. 23rd International Conference on Supercomputing*, pp.499–500 (2009).
[15] Ishii, Y., Inaba, M. and Hiraki, K.: Access Map Pattern Matching Prefetch: Optimization Friendly Method, *Workshop on JILP Data Prefetching Championship*, p.5 (2009).
[16] Mittal, S.: A Survey of Techniques for Cache Locking, *ACM Trans. Autom. Electron. Syst.*, Vol.21, No.3, pp.49:1–49:24 (2016).
[17] ARM: Cortex-A8 Technical Reference Manual (2010).
[18] Fujitsu: SPARC64TM VIIIfx Extensions (2011).

野村 隼人

2012 年和歌山工業高等専門学校電気情報工学科卒業．2014 年豊橋技術科学大学工学部情報・知能工学科卒業．2016 年電気通信大学大学院情報システム学研究科情報ネットワークシステム学専攻修士課程修了．同年東京大学大学院情報理工学系研究科博士課程進学．プロセッサアーキテクチャの研究に従事．IEEE 学生会員．

入江 英嗣　（正会員）

1999 年東京大学工学部電子情報工学科卒業．2004 年東京大学大学院情報理工学系研究科博士課程修了．博士（情報理工学）．JST 研究員，東京大学理学部助教，電気通信大学准教授を経て 2015 年より東京大学情報理工学系研究科准教授．コンピュータシステム，特に計算機アーキテクチャ，ヒューマン・コンピュータ・インタラクション，セキュアプロセッサ等に興味を持つ．電子情報通信学会各シニア会員，IEEE，ACM 各会員．

坂井 修一　（正会員）

1981 年東京大学理学部卒業．1986 年同大学大学院工学系研究科博士課程修了，工学博士．電子技術総合研究所（現，産業技術総合研究所），MIT，筑波大学等を経て，現在，東京大学大学院情報理工学系研究科教授．専門は情報システムとその応用，特に計算機アーキテクチャ，並列処理，スケジューリング，省電力情報処理，ディペンダブル情報処理．電子情報通信学会フェロー．IEEE シニア会員．人工知能学会，ACM 各会員．日本学術会議連携会員．本会フェロー．