Bulletin of Networking, Computing, Systems, and Software – www.bncss.org, ISSN 2186-5140
Volume 9, Number 1, pages 83–90, January 2020

Experimental Investigation of Lazy Evaluation Method in Replacement Algorithm for Long-term Re-reference Cache Management

Hayato Nomura
Department of Social Design Engineering, National Institute of Technology, Kochi College
200-1 Monobe-Otsu, Nankoku, Kochi 783-8508, Japan. E-mail: nom@gm.kochi-ct.jp

***Abstract*—Most replacement algorithm prioritizes temporal locality. In contrast, the Stubborn strategy, a cache management strategy, takes protection policy for cache lines that take long-term distance re-references. This behavior was considered by focusing on the characteristics of cache misses that occur in LLC. In this study, we proposed a method to delay the decision of freezing cache line evictions for more effective implementation of the Stubborn strategy. Furthermore, we implemented and evaluated a technique, "Reref". "Reref" has no achieved improving performance when based on an implement of the Stubborn area that fixed half area of cache sets and an implement of the Stubborn area that could change the number of ways adaptively. On the other hand, in the environment that majority cache ways work for Stubborn areas, "Reref" achieved a significant performance improvement of up to 32.3% and the geometric average of 6.9%. This revealed a condition suitable for the application of the Stubborn strategy.**

Keywords—cache management, replacement algorithm, long-term re-reference, lazy evaluation

I. はじめに

これまで，LRU[1]を筆頭に，RRIP[2]，最新の置き換えアルゴリズムに至るまで，キャッシュ置き換えアルゴリズムの研究者は時間的局所性を活かすために，最短の再参照間隔が得られるような置き換えの仕組みの構築に腐心してきた[3–8].

しかしながら，メモリサブシステムの全ての階層で同様の戦略を採ることは必ずしも最善の選択であるとはいえない．LLC で生じるキャッシュミスの多くは 10～100M オーダーの命令間隔の長期再参照であり[9]，一方で大容量の LLC では 86%を越えるデッドブロックが生じている[10].

先行する研究[9], [11], [12]で，我々は LLC の大容量を活かし，短期の再参照によるキャッシュアクセスを予知することよりも，長期の時間間隔で再参照されるアクセスを救うことにリソースを用い，トータルのキャッシュミス率を削減するようなキャッシュマネジメント手法を提案した．この Stubborn 戦略は，キャッシュに挿入されたラインを一定期間追い出さないよう指定する方式である．この戦略は，従来の手法では「追い出さずにキープし続ける」ことが困難であった 10～100M 命令オーダーの間隔で生じる長期の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全く活用されないワークロードなどで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を以下に構成するかという問題が依然としてあり，Stubborn 戦略のより効果的な適用のための課題となっている[13].

これまでの Stubborn 戦略の実装においては，置き換えを凍結するキャッシュラインの選定をキャッシュ挿入時に行っていた．さらに，採用の基準は先着順であり，アクセス履歴に関係なく判断されていた.

本研究では，このキャッシュラインの凍結の判断を遅延させる手法について提案し，評価した.

本論文のコントリビューションは以下の 4 点である.

- Stubborn 戦略において，ライン置き換え凍結の判断を遅延させて評価する手法を提案した.
- 「メモリサブシステムの全ての階層で同様の戦略を採ることは必ずしも最善の選択であるとはいえない」とする Stubborn 戦略の基本思想を補強する結果が得られた.

- Stubborn 戦略の適用において求められるのは，動的な Stubborn 領域の調整ではなく，Stubborn 戦略を適用するかしないかの判断であることを明らかにした．
- Stubborn 領域がセットの大半を占める構成であれば，再参照のないラインを採用しないという時間的局所性を最優先する実装であっても，より高い性能が得られることを明らかにした．

以降の本論文の構成を以下に示す．

2 章では，本研究の前提となるこれまでの Stubborn 戦略に関する先行研究を説明し，実装と課題を述べる．3 章では，キャッシュラインの凍結の判断を遅延させる手法について提案し，4 章で実装を説明する．5 章で評価環境，6 章で評価について述べ，7 章でまとめる．

II. 先行研究

Stubborn 戦略の現実的な適用にあたって，LLC の各キャッシュセットは LRU，DRRIP とハイブリッドな構成をとっている．

- LRU based Stubborn
- DRRIP based Stubborn

L3 キャッシュの 8 way セットのうち半分までを Stubborn 領域として使用し，各ラインの Stubborn フラグが 1 であれば追い出し不可，0 であれば追い出し化と判断する．実行開始から十分に時間が経過すると，キャッシュセットの半分が追い出しをしない Stubborn 領域，もう半分が LRU または DRRIP を実行し，プリフェッチを受け付けるキャッシュ領域として機能している．

さらに，各セットに占める Stubborn 領域の割合を動的に判断する実装として Stubborn-HL がある．

- LRU based Stubborn-HL
- RRIP based Stubborn-HL

Stubborn 領域として使用する way 数を，SDM[3]を用いて動的に判断する．これにより，各セットの Stubborn 領域は最小で 0way，最大で 7way を占めることになる．

これまでの Stubborn 戦略の実装においては，置き換えを凍結するキャッシュラインの選定をキャッシュ挿入時に行っていた（図 1-a）．初期の実装(LRU based Stubborn, DRRIP based Stubborn)では先着順，アダプティブな構成の実装(Stubborn-HL)ではさらに Stubborn 領域として使用するセットあたりのウェイ数が増加した際にそれ以降のキャッシュ挿入時に先着順となっている．

一方で，LLC に生じるメモリアクセス，特にキャッシュミスを生じたアクセスについて，初期参照と再参照の比率に着目してみる．図 2 の棒グラフは LLC で残るミスに占める初期参照ミスと再参照ミスの内訳を示す．縦軸は MPKI (Misses per kilo instruction)で，初期参照ミスと再参照ミスを区別して積み上げている．各ベンチマークで左の棒グラフが LRU，右が DRRIP での MPKI を示す．

結果，SPEC CPU 2006 Benchmark Suite 中の memory-intensive なワークロード 12 本の実行にあたって，プリフェッチを適用した LRU で制御される LLC で生じるミスの 76.3%が再参照ミスであった．DRRIP では再参照時のミス自体が減ったことで 71.4%が再参照ミスである．このことから，対処すべき対象は再参照ミスであると見定めることができる．

これに対して，先着順採用では LLC に残るミスのうちボリュームゾーンではない「再参照のないアドレス」を Stubborn してしまっている可能性があると考えられる．

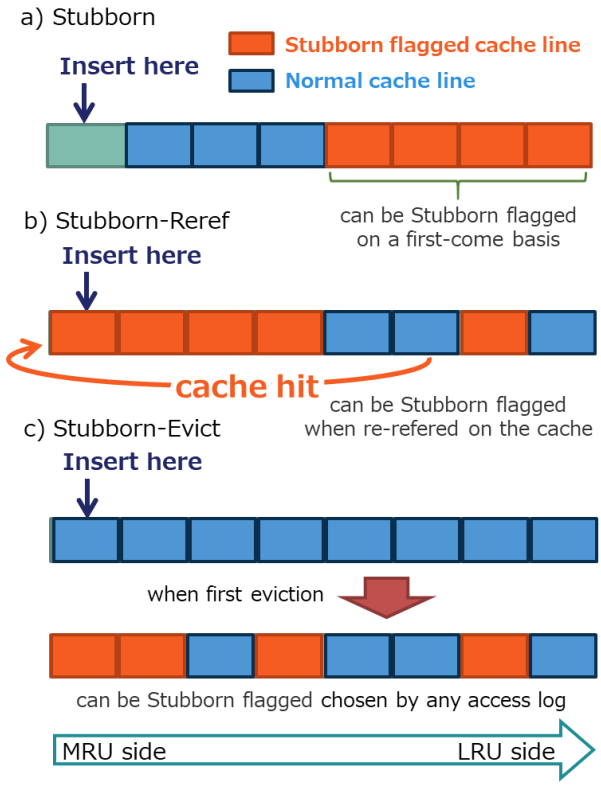

Fig. 1. Stubborn フラグ付与の選択タイミング

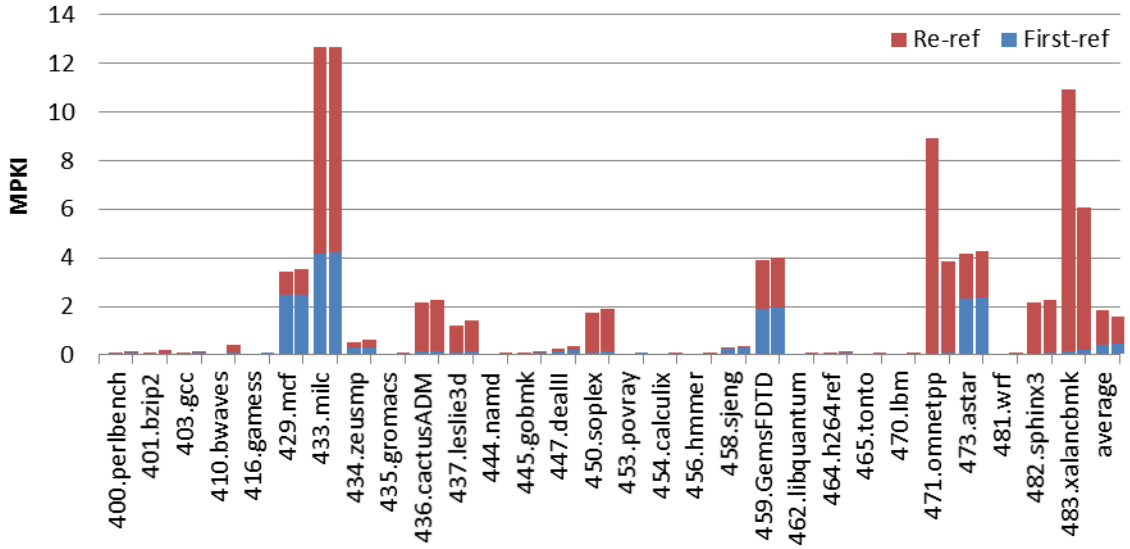

Fig. 2. LLC に占めるキャッシュミスの初期参照と再参照の内訳 (LRU, DRRIP)

III. 提案手法

これまでの Stubborn 戦略の実装においては，あるラインを凍結するかどうかの判断を挿入時に行っていた．しかしながら，その判断が挙動に反映されるのはそのラインが属するセットに対して追い出しの判断が行われるタイミングである．

つまり，あるキャッシュラインに Stubborn フラグを立てるかどうかは，あるラインがキャッシュセットに挿入されてから追い出されるまでの間に判断すればよい．したがって，遅延評価として次のタイミングでの判断が考えられる．

- あるキャッシュラインがキャッシュセット中で再参照されたとき（図 1-b）
- あるキャッシュセットに追い出しの判定が生じたとき（図 1-c）

本論文では，あるキャッシュラインがキャッシュセット中で再参照されたときに，初めて該当ラインについて Stubborn 戦略の適応に基づく追い出し凍結を行うという遅延評価の手法を提案する．これを Stubborn-Reref と名付ける．

具体的には，あるキャッシュラインがキャッシュセット中で再参照されたとき，キャッシュラインの生存期間中に再参照があったという実績をもってラインの追い出しを凍結する．これにより，先着順採用と比較して，再参照の可能性が低いラインの凍結を防ぐことができる．

これは，言わば時間的局所性を無視していた Stubborn 戦略に，時間的局所性を最優先する従来の置き換えアルゴリズムの思想を取り込むような手法である．

LRU，RRIP の機構には，それぞれ再参照のシグナルを発する仕組みが既に存在する．これを，Stubborn を許可するかしないかのトリガーとして利用する手法を提案する．これをそれぞれ LRU based Stubborn-Reref, DRRIP based Stubborn-Reref と名付ける．

さらに，アダプティブな Stubborn 戦略の実装である Stubborn-HL についても同様のライン凍結判断を実装する．これをそれぞれ LRU based Stubborn-HL-Reref, DRRIP based Stubborn-HL-Reref と名付ける．

IV. 実装

3 章で説明したアイデアに基づく実装について，Stubborn または Stubborn-HL からの差分を述べる．

- LRU based Stubborn-Reref

あるキャッシュラインの LRU オーダーを再参照により 0 にリセットするときに，そのキャッシュセットに占める Stubborn 領域が way 数の半数以下であれば Stubborn フラグを立てる．

- DRRIP based Stubborn-Reref

あるキャッシュラインの RRPV の値を再参照により 0 にリセットするときに，そのキャッシュセットに占める Stubborn 領域が way 数の半数以下であれば Stubborn フラグを立てる．

- LRU based Stubborn-HL-Reref

あるキャッシュラインの LRU オーダーを再参照により 0 にリセットするときに，そのキャッシュセットに占める Stubborn 領域が SDM によって定めた way 数以下であればそのキャッシュラインに Stubborn フラグを立てる．

- DRRIP based Stubborn-HL-Reref

あるキャッシュラインの RRPV の値を再参照により 0 にリセットするときに，そのキャッシュセットに占める Stubborn 領域が SDM によって定めた way 数以下であればそのキャッシュラインに Stubborn フラグを立てる．

いずれも，挿入時には Stubborn フラグを立てない．

V. 評価環境

A. 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[14]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 1 に示した値を用いた．この構成は Jaleel らによる評価環境[2]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．

B. 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 Baseline (LRU,DRRIP) の 2 種，先行研究である LRU based Stubborn，DRRIP based Stubborn，LRU based Stubborn-HL，DRRIP based Stubborn-HL の 4 種，提案手法として先行研究の 4 種を Reref 化した LRU based Stubborn-Reref，DRRIP based Stubborn-Reref，LRU based Stubborn-HL-Reref，DRRIP based Stubborn-HL-Reref の 4 種を合わせた 10 種類の置き換えアルゴリズムを適用して実行する．

さらに，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 (LRU based, DRRIP based)において Reref を適用したときの静的な適用度合いの変化に対する Reref 化の影響を測定したものについても掲示し，比較する．

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

C. 実行ベンチマーク

SPEC CPU 2006[15]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 1 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[9]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 10G 命令スキップ後，続く 1G 命令の区間について cycle accurate に実行し，測定した．

TABLE I. ベースラインプロセッサパラメータ

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	8KB g-share
BTB	2K 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)	2MB, 8 way, 64 B line, 24 cycle latency Stream prefetcher (degr:16, dist:16)
Memory access	250 cycle latency

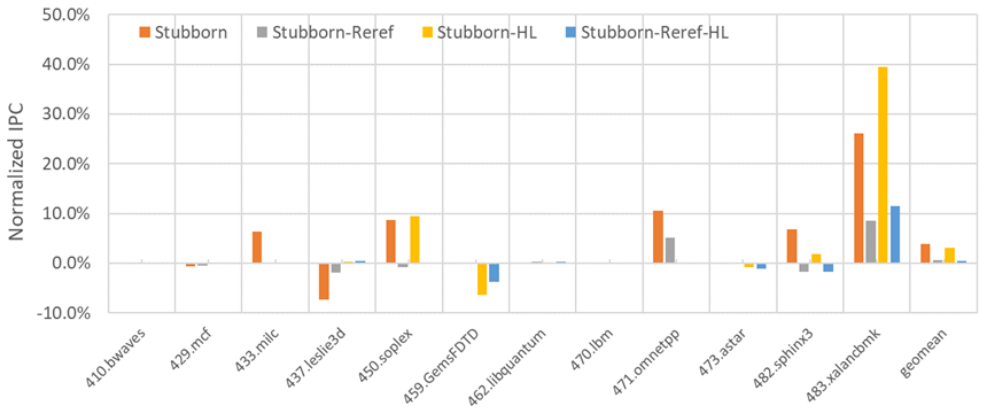

Fig. 3. LRU をベースラインとしたときの相対 IPC 評価

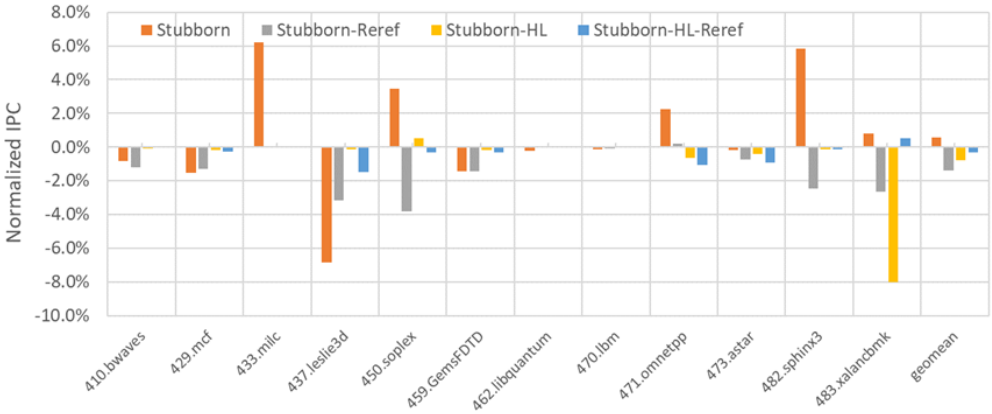

Fig. 4. DRRIP をベースラインとしたときの相対 IPC 評価

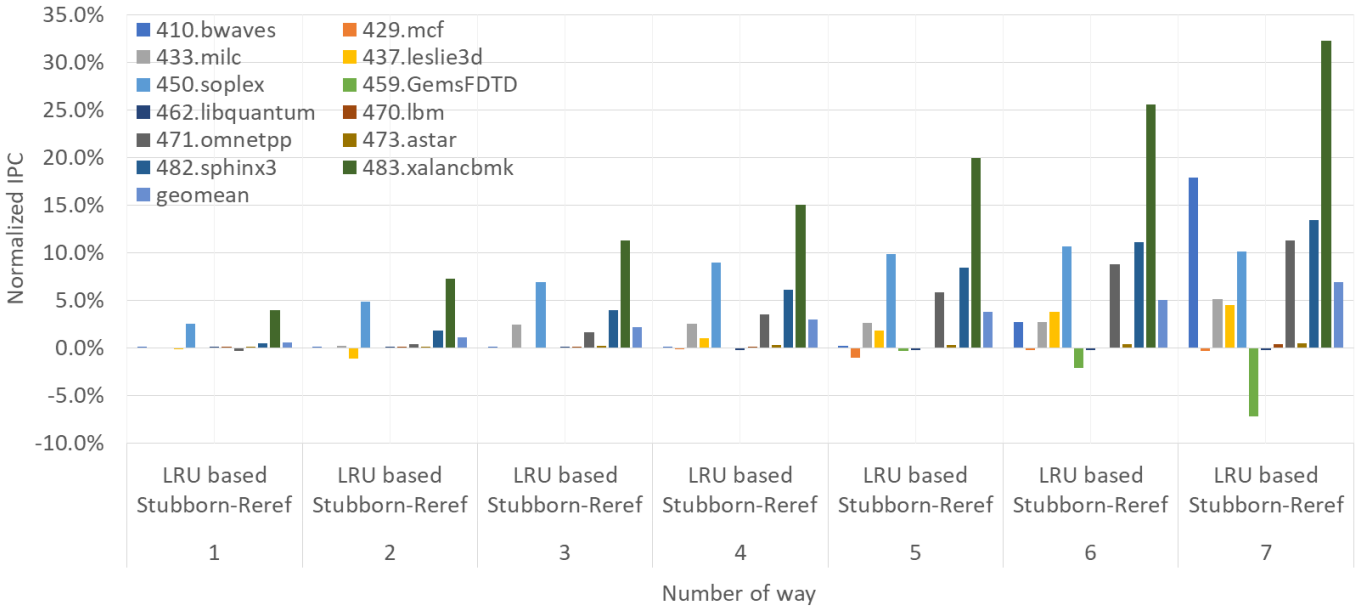

Fig. 5. LRU based Stubborn-Manually-Reref での way 数別 IPC 評価 (同 way 数 LRU based Stubborn-Manually を標準とした相対 IPC)

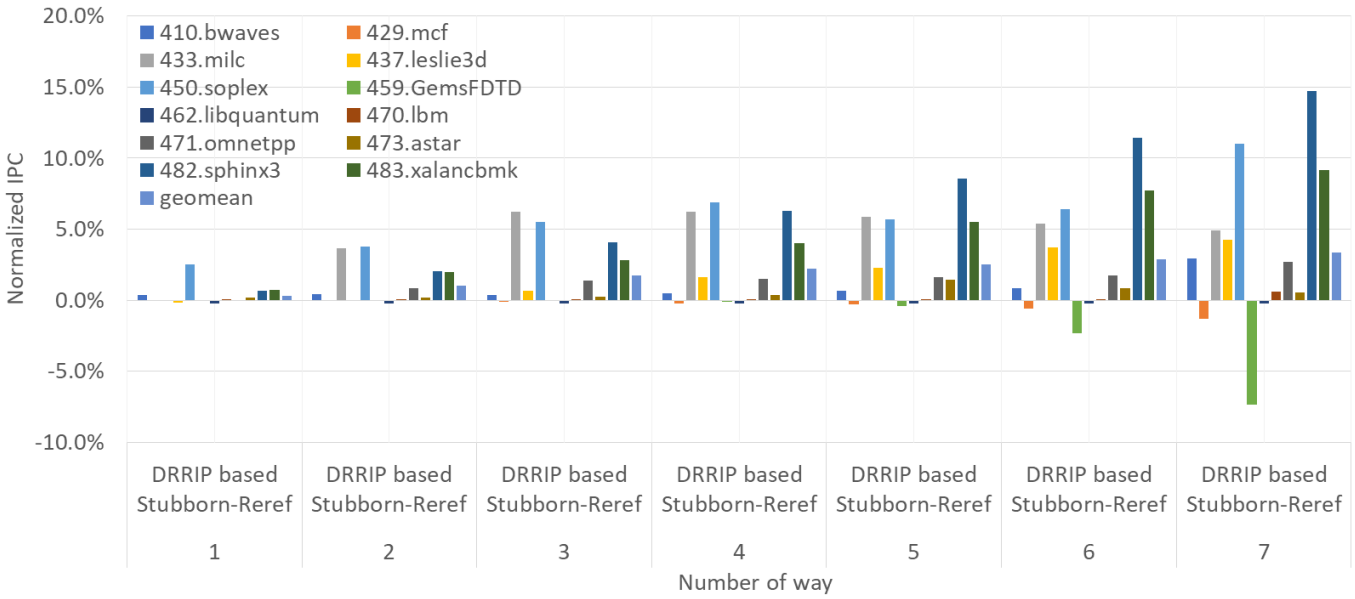

Fig. 6. DRRIP based Stubborn-Manually-Reref での way 数別 IPC 評価 (同 way 数 DRRIP based Stubborn-Manually を標準とした相対 IPC)

I. 評価

A. 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 3，DRRIP をベースラインとしたものを図 4 に示す．

LRU ベースラインの評価（図 3）において，提案手法 Stubborn-Reref により LRU 比で最大 8.6%，幾何平均で 0.7%の性能向上が得られている．提案手法 Stubborn-HL-Reref により LRU 比で最大 11.5%，幾何平均で 0.4%の性能向上が得られている．ただし，Stubborn および Stubborn-HL により，それぞれの LRU 比で最大 26.1%，39.4%，幾何平均で 3.9%，3.1%の性能向上が得られているため，Stubborn-Reref および Stubborn-HL-Reref はこれに劣る結果を示した．

このように，直感に反して，Stubborn-Reref は Stubborn に対して性能を落とした．Reref が Stubborn に勝っているのは DRRIP をベースラインとしたときの 483.xalancbmk の HL と HL-Reref の比較だが，これは HL-Reref が大きく性能を伸ばしたわけではなく，むしろ，HL が Baseline (DRRIP 比)で-8%と性能を落としすぎているためで，HL-Reref は Baseline (DRRIP 比)で 0.5%しか性能を伸ばしていない．

では，HL-Reref には HL にはない性能低下回避の機能があるかというと，HL-Half と HL-Reset がさらに高い性能を上げているので，それらとの差分を考えると，HL-Reref が持っているのは「初期参照がフェーズの山場ではない場合に Stubborn 化を避けることができる」という機能になる．

逆に言えば Reref-aware 無しの一群が Reref-aware なものより大きく性能を伸ばしているのは，「短期で再参照の実績のあるものより，先の見込みナシで先着順ででも保持した方が性能に影響する再参照によるレイテンシ隠ぺいができている」現象を説明していることなる．つまり，時間的局所性を無視していた Stubborn 戦略に，時間的局所性を最優先する従来の置き換えアルゴリズムの思想を取り込んだ結果，かえって弱体化したということだ．

これは直感に反する一方で，「メモリサブシステムの全ての階層で同様の戦略を採ることは必ずしも最善の選択であるとはいえない．」とする Stubborn 戦略の基本思想に回帰する結果を示している．

B. Stubborn Manually-Reref の結果

次に，Stubborn 戦略適用 way 数を固定する非適応的な手法 Stubborn-Manually と，その Reref 化実装である Stubborn-Manually-Reref についての評価結果を示す．

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 based Stubborn-Manually, DRRIP based Stubborn-Manually) 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 based Stubborn-Manually をベースラインとしたものを図 5，DRRIP based Stubborn-Manually をベースラインとしたものを図 6 に示す．

LRU based Stubborn-Manually ベースラインの評価（図 5）の Stubborn 領域 way 数 7 において，LRU based Stubborn-Manually-Reref により LRU based Stubborn-Manually 比で最大 32.3%，幾何平均で 6.9%と，大きな性能向上が得られている．

DRRIP based Stubborn-Manually ベースラインの評価（図 6）の Stubborn 領域 way 数 7 において，DRRIP based Stubborn-Manually-Reref により DRRIP based Stubborn-Manually 比で最大 14.7%，幾何平均で 3.3%の，大きな性能向上が得られている．

いずれも，Reref の採用で最も性能を伸ばしたのは way 数 7 のときであり，特に soplex, omnetpp, sphinx3, xalancbmk と，いずれも Stubborn 適用により性能を伸ばしてきたワークロードである．これらのワークロードでは，Reref 採用をすることによりさらに伸びしろが得られることがわかった．

性能低下のあるワークワードも確認された．GemsFDTD は Reref の導入により最大 7.4%と性能を落としている．GemsFDTD は先行する研究においても Stubborn-Manually の適用による性能低下が認められていた．一方，同じく Stubborn-Manually の適用による性能低下が認められていた leslie3d については，Reref 採用による悪化はほとんど見られなかった．

C. Stubborn-HL-Reref と Stubborn-Manually-Reref の比較

L3 キャッシュへのアクセスのうち，ミスを起こすアクセスの多数は長期の再参照距離を持つことは既知であった．これは L1, L2 キャッシュと L3 キャッシュの一部によって短期の再参照距離でのアクセスの多数を救いきっていること，短期の再参照をターゲットとした従来の置き換えアルゴリズムでは L3 キャッシュの容量を持ってしても対応できる再参照距離の長さに限界があり，次回の再参照までにスラッシングに負けて追い出されてしまっていることを示している．

それに対し，HL-Reref は，時間的局所性を無視していた Stubborn 戦略に，時間的局所性を最優先する従来の置き換えアルゴリズムの思想を取り込むような手法であり Stubborn 戦略導入以前に振り戻しをしている．

そもそも，短期で再参照のあるラインは Stubborn 戦略とハイブリッドに実装されている LRU，RRIP のベースラインアルゴリズムの動作の範疇で対応可能であったため，Stubborn 戦略による凍結を行う必要がない．にも関わらず Stubborn 領域に固定するキャッシュラインをそれらの時間的局所性の高いラインに限定したことで，かえって Stubborn 戦略が持っていた長期再参照への対応能力を失ってしまったためと考えられる．

つまり，Stubborn 戦略により保護されるべき長期再参照を起こすラインが，保護する必要のない短期での再参照のあるラインの凍結によって保護されなくなってしまった．

一方，Stubborn-Manually-Reref では Stubborn-Manually に対してさらなる性能向上の余地を見せた．ではなぜ HL-Reref でここまで性能を伸ばせなかったか？Stubborn-HL-Reref との違いは，way 数の変動にともなう Stubborn 凍結ラインの更新の有無である．Stubborn-Manually-Reref において性能向上の幅が大きいのが way 数 7 であることからも，「Stubborn 戦略を適用するかしないか」の判断のみが重要であり，一度採用したラインは手放さないことが重要であることを示している．反対に，「Stubborn 戦略の適用具合の動的な変動」は，該当期間での同一ラインの保持をする連続期間を短くするため，結局時間的局所性に囚われてスラッシングの影響を受けることになる．

II. まとめ

本研究では，長期の再参照アクセスを支援するキャッシュマネジメント手法である Stubborn 戦略の，凍結ライン選択手法への遅延評価の導入について検討し，評価した．

これは，言わば時間的局所性を無視していた Stubborn 戦略に，時間的局所性を最優先する従来の置き換えアルゴリズムの思想を取り込んだものであるが，Stubborn-Reref においても Stubborn-HL-Reref においても，そこから直感的に期待されるような結果は得られなかった．

一方で，Stubborn-Manually への Reref 適用である Stubborn-Manually-Reref では，さらなる性能向上が得られた．この結果は，再参照がないラインを保持しないこと自体はやはり Stubborn 戦略の効果的な実装につながる，ということを示した．同時に，Stubborn 領域が十分に大きければ，短期の再参照を生じるラインと長期の再参照を生じるラインを両立して保持できる，ということを示している．

今後，本論文の評価に基づく考察についてより詳細な評価を行い，定量的な判断を行う．

また，Hawkeye[6], [7]などの，あるラインについて「挿入から追い出しまで」以上の長い期間についての履歴をベースとした追い出しの判断が行える置き換えアルゴリズムの追い出し判断を Stubborn 戦略によるライン凍結に用いる手法を検討している．

REFERENCES

[1] E. G. Coffman Jr. and P. J. Denning, Operating Systems Theory. Prentice Hall Professional Technical Reference, 1973.

[2] A. Jaleel, K. B. Theobald, S. C. Steely,Jr., and J. Emer, “High Performance Cache Replacement Using Re-reference Interval Prediction (RRIP),” Proceedings of the 37th Annual International Symposium on Computer Architecture, New York, NY, USA, pp.60–71, 2010.

[3] A. Jaleel, W. Hasenplaugh, M. Qureshi, J. Sebot, S. Steely Jr., and J. Emer, “Adaptive Insertion Policies for Managing Shared Caches,” Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques, New York, NY, USA, pp.208–219, 2008.

[4] C.-J. Wu, A. Jaleel, W. Hasenplaugh, M. Martonosi, S. C. Steely Jr., and J. Emer, “SHiP: Signature-based Hit Predictor for High Performance Caching,” Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture, New York, NY, USA, pp.430–441, 2011.

[5] V. Young, C.-C. Chou, A. Jaleel, and M. K. Qureshi, “SHiP + + : Enhancing Signature-Based Hit Predictor for Improved Cache Performance,” 2017.

[6] A. Jain and C. Lin, “Back to the Future: Leveraging Belady’s Algorithm for Improved Cache Replacement,” 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), pp.78–89, Jun. 2016.

[7] A. Jain and C. Lin, “Hawkeye : Leveraging Belady ’ s Algorithm for Improved Cache Replacement,” 2017.

[8] C.-J. Wu, A. Jaleel, M. Martonosi, S. C. Steely Jr., and J. Emer, “PACMan: Prefetch-aware Cache Management for High Performance Caching,” Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture, New York, NY, USA, pp.442–453, 2011.

[9] H. Nomura, H. Katchi, H. Irie, and S. Sakai, ““Stubborn” strategy to mitigate remaining cache misses,” 2016 IEEE 34th International Conference on Computer Design (ICCD), pp.388–391, Oct. 2016.

[10] S. M. Khan, Y. Tian, and D. A. Jimenez, “Sampling Dead Block Prediction for Last-Level Caches,” Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture, Washington, DC, USA, pp.175–186, 2010.

[11] H. Nomura, T. Nakajima, M. Yoshimi, T. Yoshinaga, and H. Irie, “Stubborn Cache: A Novel Strategy for Repeating Thrashing Access Patterns,” The 18th International Symposium on Low-Power and High-Speed Chips (COOL Chips XVIII), Yokohama, p.19, Apr. 2015.

[12] H. Nomura, H. Irie, and S. Sakai, “Proposal of an Adaptive Stubborn Cache Management,” IPSJ Transactions on Advanced Computing System (ACS), vol.12, no. 3, pp.76–86, Jul. 2019 (in Japanese).

[13] H. Nomura, T. Nakamura, Toru Koizumi, H. Irie, and S. Sakai, “Preliminary Discussion of a Time Stride Prefetching,” IEEE Symposium on Low-Power and High-Speed Chips and Systems (COOL Chips 22), Yokohama, pp.40–41, Apr. 2019.

[14] K. Watanabe, H. Ichibayashi, M. Goshima, and S. Sakai, “Design of Processor Simulator ‘Onikiri’,” Advanced Computing Systems and Infrastructures (SACSIS), pp.194–195, 2007.

[15] “Standard Performance Evaluation Corporation: SPEC CPU 2006 Benchmark Suite,” https://www.spec.org/cpu2006/.