Vol.2016-ARC-221 No.17
2016/8/9

Pre-Promotionの動的切り替え手法の検討

甲地 弘幸[1]　入江 英嗣[1]　坂井 修一[1]

概要：メモリ・レイテンシを隠蔽することは，プロセッサの性能を向上させる上で重要である．我々は，メモリ・レイテンシを隠蔽する手法として置き換えアルゴリズムの判断に，プリフェッチャによる参照予測を用いる Pre-Promotion を提案した．Pre-Promotion はキャッシュのサイズを超える範囲を参照するワークロードで，再参照の間隔がある程度広いものに対して性能を向上させることができると期待できる．しかし，Pre-Promotion と相性が悪く，適用することでかえって性能が低下するベンチマークが存在しており，性能が最大で 10% 程も低下してしまう．本研究では，Pre-Promotion の適用，非適用を動的に判断させる手法を 2 つ提案し，それぞれについて検討する．

1. はじめに

プロセッサの性能向上を妨げる要因として，コアの処理速度に比べてメイン・メモリへのアクセス・レイテンシが大きいというメモリ・ウォールが挙げられる．そのため，メイン・メモリへのアクセスを減らすための様々な研究がなされている．

メイン・メモリへのアクセスを減らすためには，キャッシュでのヒット率を向上させるキャッシュ・マネージメントを考える必要がある．その 1 つのアプローチとして置換アルゴリズムがある．置換アルゴリズムには，Least Recently Used(LRU)[1] や，Re-Reference Interval Prediction(RRIP)[2] などが存在するが，どちらもセット内で最も再参照間隔が長くなるラインを予測し追い出すといったものである．また，置換アルゴリズムのみで対応しきれないキャッシュ・ミスとして初期参照ミスがあるが，これを削減する技術としてプリフェッチが存在する．[3], [4] プリフェッチでは，現在のアクセスから必要になりそうなキャッシュ・ラインを予測し，あらかじめキャッシュに挿入する．置換アルゴリズムとプリフェッチを組み合わせることで，多くのプログラムでは性能を向上させることができる．しかし，プログラムのメモリ・アクセスのパターンによっては，プリフェッチによってキャッシュを汚染し，かえって性能を低下させてしまうことがある．[5] そこで，プリフェッチャによってキャッシュが汚染されるのを防ぐ手法として，プリフェッチャ情報を置換アルゴリズムに適用させる Cache Replacement based on Access map Pattern matching(CRAP)[6] や，Prefetch-Aware Cache Management(PACMan)[5] などが研究されている．共に，プリフェッチされたラインは再参照される可能性が低いものとしてキャッシュから積極的に追い出そうとする．

これまでの研究では，プリフェッチによるアクセスを認識した上でプリフェッチされたラインを積極的に追い出すような研究がほとんどであった．しかし，それらの研究ではプリフェッチされたラインが再参照されないと仮定しており，プリフェッチが有効なプログラムでは性能が下がってしまう．一方，我々は，プリフェッチされたラインが再参照される可能性がデマンドによるラインと同程度であることを発見し，Pre-Promotion を提案した．[7]Pre-Promotion は，プリフェッチャを用いて近い未来の参照を予測し，その情報を置換アルゴリズムに用いることで，参照が迫っているが次の参照までにキャッシュから追い出されてしまう可能性のあるラインをキャッシュに保持することができる．Pre-Promotion は任意の置換アルゴリズムに適用することができ，ベンチマークとして SPEC CPU2006 を用いて，LRU，Dynamic RRIP(DRRIP)，PACMan Dynamic(PACMan-DYM) のいずれにおいても， Pre-Promotion を適用させることで性能を向上させることができた．ただし，Last Level Cache(LLC) で Pre-Promotion を適用したことでかえって性能が低下し，最大で性能が 10% ほども低下してしまうベンチマークが存在している．そこで本稿では，Pre-Promotion の適用，非適用を動的に切り替える手法の検討比較を行った．

本稿の構成は以下の通りである．まず，2 章にて Pre-Promotion の概要の説明をする．次に，Pre-Promotion を適応的にする方法として Set Dueling Monitor(SDM) による方法の検討を行う．4 章では，Pre-Promotion の異なる

[1] 東京大学大学院情報理工学系研究科

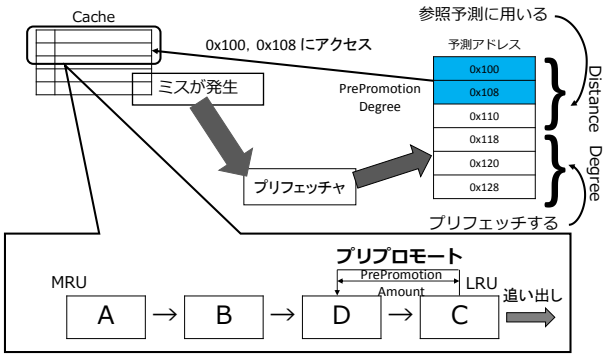

図 1　Pre-Promotion の動作例

適応手法としてプリフェッチャを用いる方法の検討を行う．5 章では，それぞれの手法の比較を行い，最後に 6 章で本論文をまとめる．

2. Pre-Promotion

2.1 概要

従来の置換アルゴリズムでは，キャッシュへのデマンド・アクセスに応じてキャッシュ内のラインに優先順位をつけることによって，追い出し発生時に優先順位の最も低いラインを追い出している．優先順位は，過去のアクセス情報によって最も使われなさそうなラインを推定しているため，ラインにアクセスが迫っていても追い出してしまう可能性がある．そこで，Pre-Promotion ではラインの参照予測情報を優先順位に反映させることで再参照が迫っているが追い出されそうなラインをキャッシュに保持させることを目的にしている．参照予測情報は，プリフェッチャによって供給される．

2.2 動作

Pre-Promotion の動作を説明する．Pre-Promotion の動作例を**図 1** に示す．キャッシュへのアクセスがミスしたときにプリフェッチャが起動する．このとき，プリフェッチャがいくつかアドレスを出し，その内の一部をプリフェッチとして用いる．残った部分のさらに一部（図 1 の青い部分）を参照が迫っているデータのアドレスとし，キャッシュにアクセスをする．このとき，目的のラインがキャッシュ内に存在していた場合，そのラインを追い出されにくくするためにプロモートする．この操作プリプロモートという．また，図 1 の青い部分を Pre-PromotionDegree と呼び，プロモートする量を Pre-Promotion Amount と呼ぶ．

2.3 問題点

Pre-Promotion は Least Recently Used(LRU)，DRRIP(Dynamic Re-Reference Interval Prediction)，PACMan-DYN(Prefetch-Aware Cache Management-DYN) などのどの既存手法に適用しても性能を向上させることができている．[8] しかし，**図 2** を見ると，ベンチマーク毎に性能変化を見た場合に静的に Pre-Promotion を用いることがよくないと言える．図 2 のデータは，ベンチマークとして，SPEC CPU2006 を用い，当研究室で開発しているサイクル・アキュレイトなシミュレータである鬼斬弐 [9] によってシミュレーションを行っており，アーキテクチャの構成は**表 1** とした．図 2 の横軸は，SPEC CPU2006 のベンチマークの一部を表し，縦軸は，Pre-Promotion を用いない場合をベースとした相対 IPC である．凡例は，例えば (2,1) の場合，L2 キャッシュの Pre-Promotion Amount が 2 で，L3 キャッシュの Pre-Promotion Amount が 1 であることを表し，Pre-Promotion を DRRIP に適用させた場合の性能が Pre-Promotion Amount によってどのように変わるのかがわかる．図 2 では，全ベンチマークの内，左から順に，

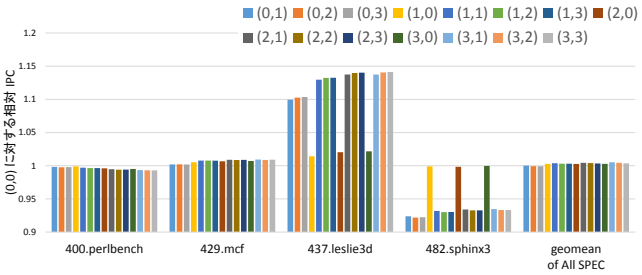

図 2　Pre-Promotion Amount を変更したときの性能変化 [8]

1. Pre-Promotion Amount を大きくすると性能が下がるベンチマーク
2. Pre-Promotion Amount を大きくすると性能が上がるベンチマーク
3. L3 キャッシュでの Pre-Promotion Amount が少しでもあるほうが良いベンチマーク
4. L3 キャッシュでの Pre-Promotion Amount が 0 の方が良いベンチマーク

の代表的なものを表示しており，一番右は，全ベンチマークの相対 IPC の幾何平均である．3 と 4 のタイプのベンチマークでは，静的に Pre-Promotion を動作させた時の性能差が大きいため，動的に適用，非適用を切り替えることができれば更なる性能向上が見込める．

3. SDM による手法

3.1 SDM

2 章で，Pre-Promotion の動的制御を述べたが，その方法の 1 つとして Set Dueling Monitor(SDM)[10] が考えられる．SDM とは，複数の置換アルゴリズムを動的に切り替える手法である．その動作は，例えば，A，B の 2 つの置換アルゴリズムを切り替えたい場合は，キャッシュを A のみが作用する部分と，B のみが作用する部分とその他に分割し，A のみ，B のみが動く部分でのミス数を計上し，

表 1　アーキテクチャの構成

プロセッサ	
issue width	int:2, fp:2, mem:2
instruction window	int:32, fp:16, mem:16
branch pred	8KB, g-share
BTB	2KB entry, 4 way
LSQ	load:48 entry, store:48 entry
キャッシュ	
L1 I/D キャッシュ	32KB, 4 way, 64B line, 3cycle latency, LRU
L2 キャッシュ	256KB, 8 way, 64B line, 10cycle latency
L3 キャッシュ	2MB, 16 way, 24cycle latency
L2 と L3 の置き換えアルゴリズムは同じ	
メモリ・アクセス	200cycle latency
L2 プリフェッチャ	Stream Prefetcher, Distance:16, Degree:16, PrePromotionDegree:16

ミス数が小さい方の置換アルゴリズムをその他の部分に適用するといった手法である．

今回は，SDM にかける置換アルゴリズムを選択するために，あらかじめ複数の手法とパラメタの組み合わせでパフォーマンス測定した．その結果を**図 3** に示す．図 3 は，L2 キャッシュには Pre-Promotion_DRRIP を適用させ，L3 キャッシュでの置換アルゴリズムを変えた場合の性能変化を示している．図 3 の横軸は，図 2 で示したベンチマークと同じものを表示している．ノードの PPA0 や PPA1 は，それぞれ Pre-Promotion Amount が 0, 1 であることに対応する．すなわち，SRRIP_PPA0 は，Static RRIP(SRRIP) で Pre-Promotion Amount が 0，BRRIP_PPA1 は，Bimodal RRIP(BRRIP) で Pre-Promotion Amount が 1 であることを表す．縦軸は，SRRIP_PPA0 を 1 としたときの相対 IPC である．図 3 を見ると，SRRIP は，Pre-Promotion Amount が 0 でも 1 でもあまり性能が変化せず，全体的には微増であるのに対し，BRRIP は，Pre-Promotion Amount が 0 のとき性能が低下するベンチマークでは Pre-Promotion Amount を 1 にすることで性能が向上し，逆に，Pre-Promotion Amount が 0 のときに性能が向上するベンチマークでは Pre-Promotion Amount を 1 にすることで性能が低下していることがわかる．よって，

1. SRRIP + Pre-Promotion
2. BRRIP + Pre-Promotion
3. BRRIP

の 3 つで SDM にかけることで性能が向上すると考えた．

3.2　評価

評価はアーキテクチャの構成を表 1 とし，シミュレータは鬼斬弐を用いた．ベンチマークは SPEC CPU2006 を用い，プログラムの先頭 10G 命令をスキップし，1G 命令分をシミュレートした．その結果を**図 4** に示す．図の横軸はベンチマークを示しており，図 2 と同じベンチマークを表示している．縦軸は，LRU にプリフェッチをかけたものを 1 としたときの相対 IPC である．凡例の DRRIP + Prefetch は，DRRIP を表し，Pre-Promotion_DRRIP は Pre-Promotion を DRRIP にかけたものを表し，Pre-PromotionDYN_DRRIP は，上述した SDM による手法を L3 キャッシュに適用させ，L2 キャッシュは Pre-Promotion_DRRIP としたものである．また，全てにおいて L2 キャッシュに対してプリフェッチをかけている．図 4 を見ると，Pre-Promotion_DRRIP と比較して leslie3d では性能が落ち，sphinx3 では性能が上がっていることがわかる．ただし，幾何平均では性能が落ちている．SDM はその実現方法から，leslie3d と sphinx3 性能変化は，良いものと悪いものの中間程度の性能になることから期待通りの性能である．しかし，幾何平均で性能が落ちてしまっているのは，ベンチマークが全体的に Pre-Promotion が少しでも効くものが多く，それらのベンチマークで性能を少しずつ下げてしまったからである．その代わり，sphinx3 で Pre-Promotion をかけたことによる 7% 程の性能低下を 4% に抑えつつ，SDM したことによる性能低下が leslie3d において 2% に留めることができたので，全体的に性能差を減らすことができたと言える．

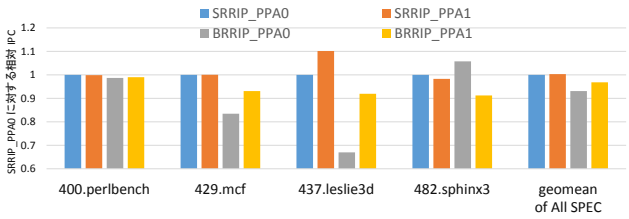

図 3　SRRIP，BRRIP と Pre-Promotion との相性

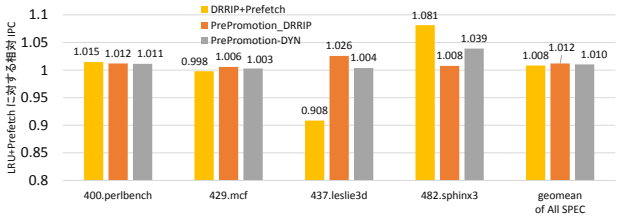

図 4　SDM を用いた Pre-Promotion の動的切り替え

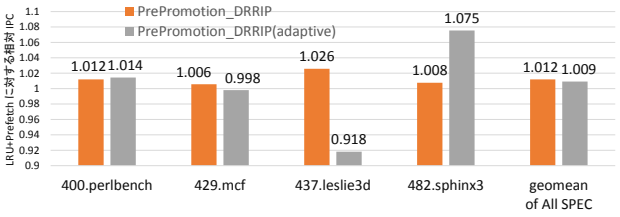

図 5　プリフェッチャの持つ統計データを用いた Pre-Promotion の動的切り替え

4. プリフェッチャを用いる手法

4.1 プリフェッチャによる制御

SDM とは異なる手法として，プリフェッチャの持つ統計データを用いてオンオフを判断することを検討する．プリフェッチャの持つ統計データとして，まず考えられるのが accuracy である．accuracy とは，プリフェッチャの精度を示す指標であり，式 (4.1) により定義される．[11]

$$accuracy = \frac{PAL}{PL} \tag{4.1}$$

PAL はプリフェッチされてかつアクセスされたライン数を表し，PL はプリフェッチされたライン数を表す．ただし，実際にはキャッシュを常に監視し数をカウントするのは，コストや速度面で難しいので，以下の式 (4.2) の accuracy' を使う．

$$accuracy' = \frac{PALE}{PLE} \tag{4.2}$$

PALE はキャッシュから追い出されたラインの内，プリフェッチされてかつアクセスされたラインの数，PLE はキャッシュから追い出されたラインの内，プリフェッチされたラインの数を表す．以下，特に断りがなければ accuracy' の意味で accuracy を使う．しかし，accuracy が高いからといって，プリフェッチによって性能が向上しやすいとは限らないことが知られている．[11] このことは Pre-Promotion にも当てはまる．実際，sphinx3 と leslie3d は accuracy が同程度にも関わらず，上述したとおり性能としては真逆の性質を示している．そのため，2 章で分類した 3 に属するベンチマークと 4 に属するベンチマークを明確に分類する統計データを考える必要がある．図 3 を見ると，BRRIP の効きやすさが 1 つの指標になると考えられるが，それを明確な数値にする方法はまだ検討中である．

4.2 評価

accuracy による制御がどの程度のものかを評価する．評価環境は SDM のときと同じで，L2 キャッシュと L3 キャッシュの置換アルゴリズムは共に Pre-Promotion_DRRIP である．**図 5** に結果を示す．図 5 の横軸はベンチマーク，縦軸は LRU にプリフェッチをかけたものに対する相対 IPC，ノードの Pre-Promotion_DRRIP(adaptive) が accuracy による制御をかけた場合を表す．accuracy の制御方法は，

1. Pre-Promotion がオンでかつ，accuracy がある閾値以下のときオフにする．
2. Pre-Promotion がオフでかつ，accuracy がある閾値以上のときオンにする．

であり，オフにするときは L2 キャッシュ，L3 キャッシュ共にオフになるようにしている．ここで，オンにするときもオフにするときも閾値は 0.8 とした．leslie3d も sphinx3 も accuracy の平均は 0.5〜0.6 であるため実行中はほとんど Pre-Promotion をオフにしていたと考えられ，leslie3d では性能が低下し，sphinx3 では性能が向上すると期待できる．図 5 を見ると，leslie3d と sphinx3 で期待通りの性能変化が見られ，幾何平均でも性能が落ちているため，accuracy が適切な指標でないことが実際に確かめられた．

5. 比較検討

以上 2 つの方針での Pre-Promotion の動的手法について検討したが，この章では 2 つの手法を 2 つの観点から比較する．

最初に，ハードウェア・コストの面で比較する．SDM による手法は DRRIP や PACMan-DYN でも実際に用いられており，PACMan-DYN に至っては今回の手法と同じようにアクセスが何であるかを区別しその上で SDM をしているので今回の提案は多くとも PACMan-DYN に Pre-Promotion によるアクセスを区別する比較器によるコストが増えるだけで済む．一方，プリフェッチャによる制御の場合，必要な統計データをカウントするための記憶領域をプリフェッチャに加える必要があり，制御を判断する回路と場合によっては，キャッシュにビットを追加する必要が生じる．また，今回は Pre-Promotion をオフにするときは全体をオフにしていたが，実際には L3 キャッシュのみで Pre-Promotion のオンオフを切り替えられることが望ましいので，L2 キャッシュから L3 キャッシュに降りてきたアクセスが Pre-Promotion であるかを判断するときに，Pre-Promotion のオンオフの情報を参照する回路をつける必要がある．

最後に，期待される性能面での比較をする．SDM ではその実現方法から，モニタリングしている置換アルゴリズムの中間程度の性能が出るので，性能向上としてはそこまで大きくならない．一方プリフェッチャによって制御した

場合，統計データをうまくとることができれば常に最高の性能を出すことができるので，期待値として大きくなる．

以上から，ハードウェア・コスト面では SDM が勝るが性能面ではプリフェッチャによる制御のほうが勝ることがわかる．ただし，ハードウェア・コストに関しては，既存手法で使われているものを使うことができれば，大きな差が出ない可能性がある．

6. おわりに

我々は，新しいキャッシュ・マネージメントとして未来の参照予測を置換アルゴリズムに適用させる手法である Pre-Promotion を提案している．しかし，Pre-Promotion は L3 キャッシュにおいて適用することでかえって性能が低下してしまうベンチマークが存在している．そのため，プログラムの実行中に L3 キャッシュにおけるオンオフを切り替えることができれば更なる性能向上が見込める．

動的に Pre-Promotion を切り替える手法として，SDM を用いる手法とプリフェッチャによって制御する手法をそれぞれ検討した．その結果，SDM による手法は静的に動作させる場合よりも性能が落ちてしまうものの，性能が大きく落ち込むベンチマークでの性能低下を抑えることができた．一方，プリフェッチャによる制御は現時点では性能を向上させることができていない．しかし，プリフェッチャによって制御する場合に用いる統計データとして性能と正の相関があるものを見つけ出し，実際にそれを用いて制御しることで，最も性能が向上することが見込める．

謝辞 本論文の研究は一部，文部科学省科学研究費補助金 No. 25730028 による．

参考文献

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

[2] Jaleel, A., Theobald, K. B., Steely, Jr., S. C. and Emer, J.: High Performance Cache Replacement Using Re-reference Interval Prediction (RRIP), *SIGARCH Comput. Archit. News*, Vol. 38, No. 3, pp. 60–71 (online), DOI: 10.1145/1816038.1815971 (2010).

[3] Smith, A. J.: Sequential Program Prefetching in Memory Hierarchies, *Computer*, Vol. 11, No. 12, pp. 7–21 (online), DOI: 10.1109/C-M.1978.218016 (1978).

[4] Fu, J. W. C., Patel, J. H. and Janssens, B. L.: Stride Directed Prefetching in Scalar Processors, *SIGMICRO Newsl.*, Vol. 23, No. 1-2, pp. 102–110 (online), DOI: 10.1145/144965.145006 (1992).

[5] Wu, C.-J., Jaleel, A., Martonosi, M., Steely, Jr., S. C. and Emer, J.: PACMan: Prefetch-aware Cache Management for High Performance Caching, *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-44, New York, NY, USA, ACM, pp. 442–453 (online), DOI: 10.1145/2155620.2155672 (2011).

[6] 石井康雄，稲葉真理， 平木敬：マップ型履歴を用いたプリフェッチ方式とキャッシュ置換方式の協調動作，研究報告計算機アーキテクチャ（ARC），Vol. 2010, No. 13, pp. 1–8（オンライン），入手先 ⟨http://ci.nii.ac.jp/naid/110007997663/⟩ (2010).

[7] 力翠湖，眞島一貴，藤原大輔，吉見真聡， 吉永努，入江英嗣：プリフェッチ情報から再参照予測を行うキャッシュライン置き換えアルゴリズム，情報処理学会研究報告. 計算機アーキテクチャ研究会報告， Vol. 2013, No. 20, pp. 1–7（オンライン），入手先 ⟨http://ci.nii.ac.jp/naid/110009587862/⟩ (2013).

[8] 甲地弘幸，入江英嗣，坂井修一：D-6-14 置き換えアルゴリズムとプリフェッチが協調動作するキャッシュマネージメント・プリプロモーションの評価 (D-6. コンピュータシステム, 一般セッション)，電子情報通信学会総合大会講演論文集，Vol. 2016, No. 1, p. 68（オンライン），入手先 ⟨http://ci.nii.ac.jp/naid/110010036702/⟩ (2016).

[9] 塩谷亮太，五島正裕，坂井修一：プロセッサ・シミュレータ「鬼斬弐」の設計と実装，先進的計算基盤システムシンポジウム SACSIS2009， Vol. 2009, No. 4, pp. 120–121 (2009).

[10] Jaleel, A., Hasenplaugh, W., Qureshi, M., Sebot, J., Steely, Jr., S. and Emer, J.: Adaptive Insertion Policies for Managing Shared Caches, *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, PACT '08, New York, NY, USA, ACM, pp. 208–219 (online), DOI: 10.1145/1454115.1454145 (2008).

[11] Ishii, Y., Inaba, M. and Hiraki, K.: Access Map Pattern Matching for Data Cache Prefetch, *Proceedings of the 23rd International Conference on Supercomputing*, ICS '09, New York, NY, USA, ACM, pp. 499–500 (online), DOI: 10.1145/1542275.1542349 (2009).