Vol.2014-ARC-212 No.2
2014/10/6

プリフェッチ精度に基づくキャッシュライン保持手法

力 翠湖[1]　吉見 真聡[1]　吉永 努[1]　入江 英嗣[1]

概要：キャッシュメモリを管理するアルゴリズムは，メインメモリのデータ転送遅延を隠蔽するために重要である．近年では，大容量の LLC(Last Level Cache) が広く使われており，この LLC を管理するアルゴリズムとしてスキャンアクセスに耐性を持つ置き換えアルゴリズムやプリフェッチなどが提案されている．それぞれのアルゴリズムは LLC の性能を向上させるが，組み合わせた場合にお互いのアルゴリズムが悪影響を与え合い，かえってキャッシュの性能が低下することが知られている．我々は，このような衝突を防ぎ，かつプリフェッチと LLC 向けのキャッシュ置き換えアルゴリズムとが協調し合うことでよりキャッシュ性能を高めるプリ・プロモーションを提案している．この手法では，プリフェッチャのアドレス予測に基づいて，アクセスを予測されたキャッシュラインを投機的に MRU 側に移動させ，近い将来にアクセスされるラインを保持する．しかしながら，この手法はプリフェッチャの予測に依存するため，この予測精度によって性能が左右されてしまう．そこで，プリフェッチ精度によりラインを保持するかを決定する適応型プリ・プロモーションを提案し，プリフェッチ精度による性能の変化を調査，評価する．本提案手法を既存のキャッシュライン置き換えアルゴリズム RRIP に適用し，SPEC CPU2006 を用いてシミュレーションにより評価した．この結果，LLC を 1MB の L2 キャッシュとしたシングルコア構成において，RRIP に対して IPC が最大で 25.5%，幾何平均で 6.0%向上した．また，DRRIP に適応したプリ・プロモーションに対しては最大で 19.9%，幾何平均で 1.4%向上した．

1. はじめに

プロセッサとメインメモリの間のアクセスレイテンシは通常の命令処理速度に比べて 100 倍も大きく，プロセッサ性能の向上を妨げる主要なボトルネックである．このレイテンシを隠蔽するために，高速なアクセスが可能な小容量のキャッシュメモリをチップ内に置き，このキャッシュを介してデータや命令の転送が行われている．近年では，これまでに比べて大容量の Last Level Cache(LLC) が実装されるようになり，ほとんどのワーキングセットをキャッシュに保持することが可能となった．しかしながら，メモリインテンシブなアプリケーションは，一度しか参照されないアドレスへの参照が連続する**スキャン**アクセスパターンや，有用なキャッシュラインが再参照される前にキャッシュから追い出されてしまう**スラッシング**アクセスパターンを含む大きなワークロードを持っており，このようなワークロードに対して，一般的に広く使われている置き換えアルゴリズムの **LRU（Least Recently Used）**を用いると，有用なキャッシュラインが再参照前に追い出されてしまう．この問題を解決するために，**RRIP（Re-ReferenceInterval Prediction）**[1] をはじめとして数多くの置き換えアルゴリズムが提案されている [2], [3], [4].

一方で，プリフェッチを用い，キャッシュラインに予めデータを入れることで初期参照ミスを防ぐ手法が提案されている [5], [6]．しかし，置き換えアルゴリズムやプリフェッチはそれぞれがキャッシュの性能を向上させるものの，これらのアルゴリズムを同時にキャッシュに適用した場合に互いの予測アルゴリズムが衝突し，キャッシュの性能が低下してしまうことが知られている．この衝突を抑えるために，プリフェッチされたラインが再参照されないことに着目し，置き換えアルゴリズムとプリフェッチを組み合わせてキャッシュを改良する手法が提案されている [7], [8]．しかし，これらの手法では予測アルゴリズムの衝突を抑えるだけに留まってしまい，キャッシュの性能を向上させることは難しい．

我々は，このような衝突を防ぎ，更にプリフェッチと LLC 向けのキャッシュ置き換えアルゴリズムとが協調し合うことでよりキャッシュ性能を高める**プリ・プロモーション** [9] を提案している．プリ・プロモーションでは，プリフェッチャの予測によって直近にアクセスを予測されたキャッシュラインを **MRU（Most Recently Used）**側に寄せることで，アクセスされる可能性が高いラインを

[1] 電気通信大学システム学研究科
Graduate School of Information System, the University of Electro-Communications

キャッシュ内に保持することができる．しかし，この手法ではプリフェッチャの予測を用いており，プリフェッチャの予測精度によってはかえって有用なラインを追い出してしまう．そこで我々は，一定期間ごとにプリフェッチャの予測精度を算出し，その精度によってプリ・プロモーションの動作の可否を決定する**適応型プリ・プロモーション**を提案する．また，本提案手法を既存のキャッシュライン置き換えアルゴリズムの RRIP に適用し，最適な閾値とモニタリング期間の調査と評価を行った．SPEC CPU2006 を用いてシミュレーションにより評価した結果，LLC を 1MB の L2 キャッシュとしたシングルコア構成において，IPC が RRIP に対して最大で 25.5%，幾何平均で 6.0%向上した．また，DRRIP に適応したプリ・プロモーションに対して最大で 19.9%，幾何平均で 1.4%向上した．

以下，2 章では関連研究について述べ，3 章ではプリ・プロモーションについて述べる．また 4 章では，本論文で提案する適応型プリ・プロモーションについて述べ，5 章で提案手法の実装について述べる．6 章では，提案手法の予備評価及び調査と考察を行い，最後に 7 章で結論を述べる．

2. 関連研究

2.1 スキャン耐性を持つ置き換えアルゴリズム

キャッシュはメインメモリに比べて小容量であり，載せられるデータ量が限られている．そのため，キャッシュの利用効率を上げるには，有用なデータを保持するキャッシュ管理アルゴリズムが重要となる．この管理アルゴリズムとして，一般的には新しいデータをキャッシュに挿入するためにどのデータをら追い出すかを決定する置き換えアルゴリズムが用いられている．

最適な置き換えアルゴリズムとして，最も長く参照されないラインを置き換え対象とする OPT[10] が知られているが，OPT を実現するためには未来のメモリアクセスの情報が必要であり，その実装は現実的ではない．このため，一般的に置き換えアルゴリズムとして LRU が広く用いられている．LRU では，最も直近に参照されたラインを MRU 位置に挿入し，参照されていないラインを LRU 側に移動させることで，最も参照されていないラインを置き換え対象とする．しかし，ワーキングセットがキャッシュ容量に対して大きいと，スキャンやスラッシングが発生してしまう．特に大容量の LLC に載り切らないような大きなワーキングセットの場合，メモリアクセスが頻発する．また，再参照されるラインが追い出されているのにも関わらず，再参照されないライン (**デッドブロック**) が残ってしまう場合もあり，キャッシュの容量を有効に活用できない．

このようなワークロードに対応するために，デッドブロックを予測し，キャッシュから追い出す手法が提案されている．Lai らは各命令シーケンスが最後に参照されたのはいつかをモニタリングしてデッドブロックを予測 [2] し

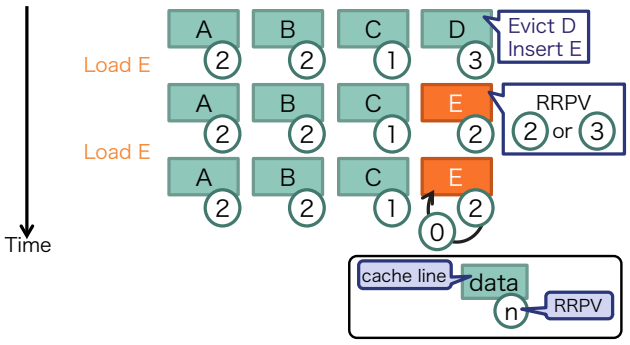

図 1 RRIP の動作例：データ E の参照による RRPV の変化

ており，Khan ら [3] はデッドブロックと予測したデータをキャッシュに配置しない事によって更に性能を向上させた．また，Hu らは，該当ラインが L1 キャッシュに残っている時間からデッドブロックを予測する手法 [11] を提案している．

その一方で，再参照される可能性が高いラインを予測し，キャッシュに保持しておく手法も提案されている．Lee らは，参照間隔から置き換えを行う LRU に，参照頻度を用いて置き換えを行う **LFU（Least Frequently Used）**を組み合わせる手法 [4] を提案している．これにより，スキャンアクセスパターンによるキャッシュの性能低下を回避することができる．また，Jaleel らは，一度でも再参照されたラインを優先的にキャッシュに残す RRIP[1] を提案している．

RRIP では, 置き換え順序を判断する指標として, 再参照されるまでにかかる時間を意味する **RRPV (Re-Reference Prediction Values)** を用いている．この RRPV はキャッシュライン毎に M ビットの値を持ち，この値が 0 であれば MRU 側，$2^M - 1$ であれば LRU 側の順序を持つことを示している．なお，この手法には，スキャンに耐性を持つ **SRRIP**，スラッシングに耐性を持つ **BRRIP**，**SDM (Set Dueling Monitor)** [12] を用い，SRRIP と BRRIP のうちキャッシュミスが少ない方のアルゴリズムを動的に切り替える **DRRIP** の 3 つの動作がある．

図 1 に RRIP の動作例を示す．この図では，各キャッシュラインが 2 ビットの RRPV を持っており，また縦に時間軸を取っている．まず，データ E へのアクセスが発生したとすると，RRPV が最大の 3 を持つラインが追い出され，E が挿入される．このとき，E が挿入されるラインの RRPV は 2 にセットされるため，新しく挿入されたデータは LRU 側に配置される．次に，E への再参照が発生した場合，このラインの RRPV を 0 とすることで E を MRU 側に移動させる．このようにして，RRIP では再参照されたラインを優先的に保持することができる．

2.2 Prefetch aware な置き換えアルゴリズム

スキャン耐性を持つ置き換えアルゴリズムのように，競

合・容量ミスを削減するアルゴリズムが提案されている一方で，初期参照ミスを削減する**ハードウェアプリフェッチ**も提案されている．プリフェッチでは，アドレス予測器を用いて，これまでのアクセスパターンから今後のメモリアクセスを予測し，実際のメモリアクセスが発生する前にアクセスされると予想したデータをキャッシュに挿入することでキャッシュの性能を向上させる．プリフェッチの典型的な手法として，連続メモリ領域へのアクセスを予測する**ストリーミングプリフェッチ** [5] やストライドアクセスを予測する**ストライドプリフェッチ** [6] が存在する．

スキャン耐性を持つ置き換えアルゴリズムやプリフェッチがそれぞれキャッシュ性能を向上させていることから，石井らは，これらのアルゴリズムを同時に適用することで更なる性能向上を図る CRAP（Cache Replacement based on Access map Pattern matching）[7] を提案した．CRAP では，プリフェッチが予測したメモリアクセスパターンが時間的局所性に乏しいことを利用し，プリフェッチによってキャッシュに載せられたラインを積極的に追い出している．また，Wu らは，スキャン耐性を持つ置き換えアルゴリズムとプリフェッチを同時に適用した場合に，互いの予測アルゴリズムが衝突してしまい，かえってキャッシュの性能が低下することに着目した PACMan（Prefetch-Aware Cache Management）[8] を提案している．

PACMan では，プリフェッチされたキャッシュラインが再参照されにくいことに着目し，プリフェッチされたラインを LRU 側に挿入してキャッシュから追い出されやすくしている．この手法は DRRIP に対して適用されており，キャッシュヒットから再参照を予測する **PACMan-H**，キャッシュミスから再参照を予測する **PACMan-HM**，ヒットとミスの両方から再参照を予測する **PACMan-HM**，PACMan-H と PACMan-HM を SDM を用いて切り替える **PACMan-DYN** の 4 つの動作がある．この手法では，RRIP に対して IPC を 1.38%向上させている．

3. プリ・プロモーション

3.1 プリフェッチされたラインの再参照可能性

PACMan などの既存手法では，プリフェッチされたラインは再参照されにくいと仮定され，プリフェッチされたラインはすぐにキャッシュから追い出されるべきだとしてキャッシュの制御が考えられている．しかし，文献 [7] で述べられているように，ワークロードによってプリフェッチされたラインの再参照性は異なる．このため，PACMan のようにプリフェッチされたラインを追い出しやすくした場合、不当に再参照間隔を広く見積もられてしまうラインも存在し，このようなラインは，有用にも関わらず再参照前に追い出されてしまう可能性がある．更に，RRIP のようなスキャン耐性を持つ置き換えアルゴリズムでは，参照される前のラインは LRU 側に配置されているために，プリフェッチされたラインがより追い出されやすくなる．

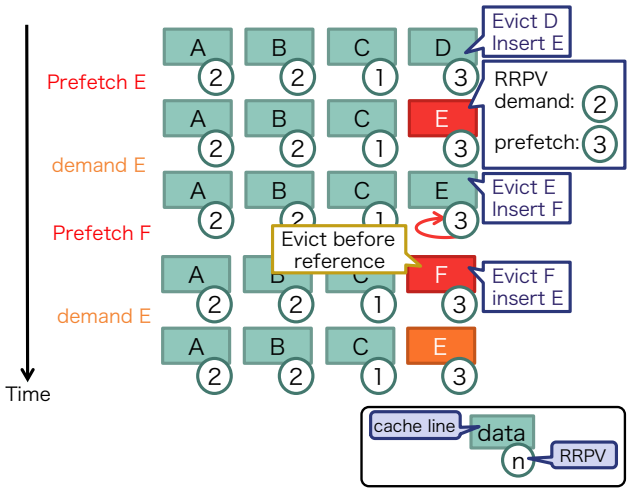

図 2 プリフェッチされたラインが不当に追い出されうるメモリアクセス

このような状況について，PACMan-HM を用いて説明する．図 2 に, PACMan-HM の動作例を示す．PACMan-HM では，この図は 2 章の図 1 と同様に各キャッシュラインが 2 ビットの RRPV を持っており，縦に時間軸を取っている．まず，E へのプリフェッチが発生すると，RRPV が最大値のラインが追い出される．PACMan-HM では，プリフェッチでキャッシュに挿入されたラインの RRPV が最大値にセットされるため，E の RRPV を 3 にする．次に，E に通常アクセスが発生する．この時，プリフェッチされたラインを追い出されやすい位置に維持するために，RRPV を操作しない．続いて，F へのプリフェッチが発生すると，RRPV が最大の E が追い出し対象となる．最後に，E への通常アクセスが発生するが，直前で F を挿入するために追い出してしまったので，E を改めて挿入する必要がある．このように，PACMan はプリフェッチされたラインがキャッシュから追い出されやすくなっているため，プリフェッチされたラインが再参照前に不当に追い出される．

3.2 プリフェッチのアドレス予測を用いたキャッシュライン保持手法

各キャッシュラインがプリフェッチされたかどうかで再参照可能性を区別するのではなく，再参照間隔をラインごとに評価してキャッシュの利用効率を上げるために，我々は**プリ・プロモーション** [9] を提案した．この手法では，プリフェッチャによるアドレス予測にキャッシュラインに対して参照が迫っていることを知らせる効果があると考える．つまり，プリフェッチャによってアクセスが予測されるアドレスを監視しすることで，キャッシュから追い出すべきではないラインを判別する．

例えば，図 3 に示すようなスラッシングアクセスが発

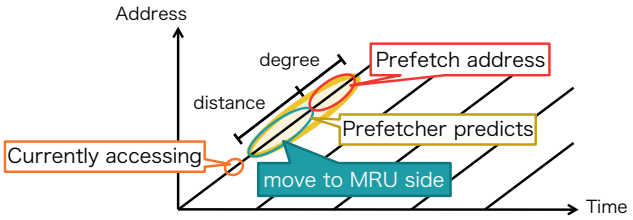

図 3 プリ・プロモーションの概要

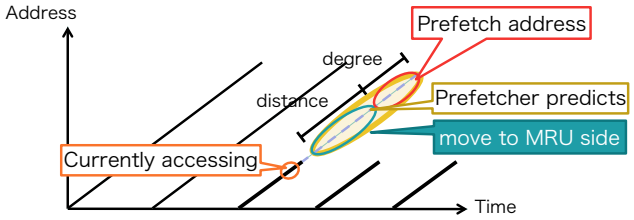

図 4 プリフェッチャの誤った予測に基づいたプリ・プロモーション

生した場合を考える．図の縦軸にメモリアドレス，横軸に時間を取る．また，図中の distance はアクセスを予測したもののプリフェッチが間に合わない領域であり，degree は実際にプリフェッチされる領域である．まず，プリフェッチャは，現在のメモリアクセスから今後のメモリアクセスを予測する．この予測に基づき degree の領域をプリフェッチするが，この時に distance の領域に対してプリ・プロモーションを行う．プリ・プロモーションされるラインは，プリフェッチされたラインかどうかに関わらずキャッシュに存在すれば MRU 側に移動させられる．このように，プリフェッチャのアドレス予測に基づいてラインの再参照間隔予測を行うため，プリフェッチされたかに関わらず有用とみなされたラインを保持することができる．

4. 適応型プリ・プロモーション

プリ・プロモーションではプリフェッチ情報を利用して投機的にキャッシュラインを MRU 側に移動させるため，その性能はプリフェッチ精度に依存する．プリフェッチ精度が高ければ，正しいアクセス予測に基づいて有用なラインを追い出しにくくすることができる．しかし，プリフェッチ精度が低い，すなわちプリフェッチの予測が外れてしまっている場合，誤った予測に基づいてアクセスが見込まれたラインが MRU 側に移動し，かえって有用なラインを追い出す可能性がある．

例えば図 4 に示すようなメモリアクセスが発生した場合を考える．この図では，3 章 の図 3 と同様に縦軸にメモリアドレス，横軸に時間を取る．まず，プリフェッチャは最初の 2 イテレーションから今後のメモリアクセスを予測し，プリフェッチを行う．この時，プリ・プロモーションではこのプリフェッチのアドレス予測からアクセスされると判断され，かつキャッシュ上に存在するラインを投機的に MRU 側に移動させる．しかし，実際のメモリアクセスはプリフェッチャの予測とは異るため，間違ったアドレス予測に基づいて保持されたラインにより，本当に有用なラインが追い出されてしまう可能性がある．

プリフェッチ情報に基づく誤った予測による性能低下を防ぎ，よりワークロードに適したプリ・プロモーションを実現するため，本稿では適応型プリ・プロモーションを提案する．本提案手法では，プリフェッチ精度に基づきアクセスが予測されたラインを保持するかを決定する．

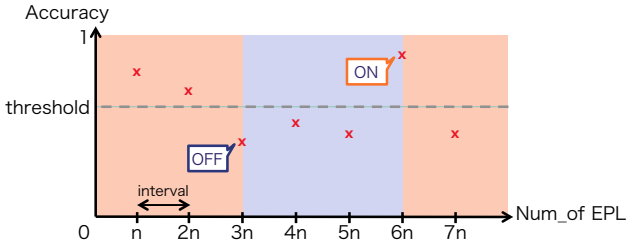

図 5 プリフェッチ精度の算出とプリ・プロモーションの切り替え

プリフェッチ精度（accuracy）は，プリフェッチされて追い出されたライン（Evicted Prefetched Lines / EPL）に対するキャッシュヒットしたライン（Evicted Prefetched Lines which Accessed / EPLA）の割合から求められる．すなわち，以下の式で算出することができる．

$$accuracy = \frac{Num\_of\_EPLA}{Num\_of\_EPL}$$

図 5 に示すように，プリフェッチされたラインが一定数キャッシュから追い出される期間で accuracy を算出し，この値が閾値よりも高ければプリ・プロモーションを動作させ，低ければ動作させない．このようにすることで，プリフェッチ精度に基づき，より有用なラインのみをキャッシュ内に保持することができる．

5. 適応型プリ・プロモーションの実装

5.1 プリ・プロモーションの実装

本提案手法のベースとなるプリ・プロモーションでは，プリフェッチャにアクセスが予測されたラインを MRU 側に移動させることで保持する．例えば，RRIP に適用したとすると，プリフェッチャによりアクセスが予測されたラインがキャッシュ上に存在した場合に RRPV をデクリメントすることで，有用なラインを保持することができる．なお，この動作を実現するために，ハードウェア資源を追加する必要はない．また，プリフェッチャが予測したアクセスのうち，プリ・プロモーションの対象となるのは，図 6 に示す distance の間の領域とする．

図 7 に，RRIP に適用したプリ・プロモーションの動作を示す．この図も，2 章の図 1 と同様に，各キャッシュラインが 2 ビットの RRPV を持ち，縦に時間軸を取る．まず，プリフェッチャにより E へのアクセスが予測される．

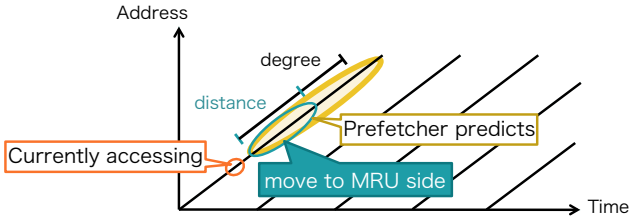

図 6 プリ・プロモーションの対象となる領域

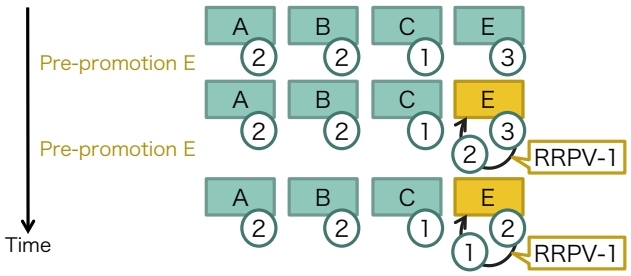

図 7 プリ・プロモーションの動作例

この時，E がキャッシュライン上に存在すれば，該当ラインの RRPV をデクリメントする．次に，再度プリフェッチャが E へのアクセスを予測した時も同様に RRPV をデクリメントする．通常のキャッシュヒット時は RRPV を 0 にするが，プリ・プロモーションでは RRPV をデクリメントするために，通常のヒット時よりも緩やかに追い出されにくい位置に移動する．また，プリ・プロモーションする毎に RRPV のデクリメントを繰り返すため，参照が迫るにつれて該当ラインがどんどん追い出されにくくなる効果が得られる．

5.2 適応型プリ・プロモーションのハードウェア概要

適応型プリ・プロモーションの概要を，図 8 に示す．この手法では，プリフェッチ精度をモニタリングするために，プリ・プロモーションに以下を追加している．

(1)EPL-counter EPL を計測するカウンタ
(2)EPLA-counter EPLA を計測するカウンタ
(3)enable-flag プリ・プロモーションを動作させるか判別する
(4)prefetch-flag プリフェッチされたラインか判別する
(5)accessed-flag キャッシュヒットしたラインか判別する

(1)-(3) は適応型プリ・プロモーションを適用するキャッシュ毎に用意し，(4), (5) はキャッシュライン毎に用意する．

まず，EPL-counter と EPLA-counter を 0 に初期化する．キャッシュラインの置き換えが発生した時，追い出されるラインがプリフェッチされたラインであれば，EPL-counter をカウントアップする．更に，この追い出されるラインがプリフェッチされたラインで，かつキャッシュヒットしていれば，EPLA-counter もカウントアップする．

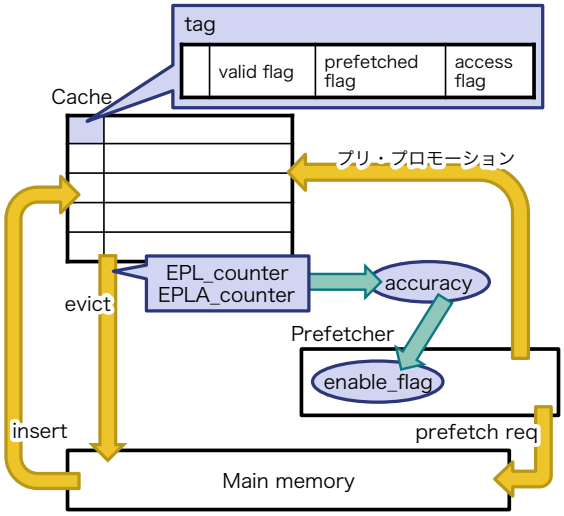

図 8 プリ・プロモーションの概要

表 1 アーキテクチャパラメタ

Processor	
Core, ISA	1Core, Alpha AXP
issue width	int:2, fp:2, mem:2
instruction window	int:32, fp:16, mem:16
branch pred	8KB g-share
BTB	2K entry, 4way
LSQ	64 entry

次に，プリフェッチがトリガされた際に EPL-counter が一定数以上だった場合に accuracy を算出する．算出した accuracy に基づき，プリ・プロモーションを動作させるか決定する．この時，accuracy が上の閾値よりも大きくなった場合に動作させ，下の閾値よりも小さくなった場合に動作させない．この操作が終わった後，EPL-counter と EPLA-counter を 0 に初期化し，enable-flag に基づいてプリ・プロモーションを実行する．

6. 評価

6.1 評価環境

プロセッサシミュレータ鬼斬弐 [13] に LRU，DRRIP, PACMan-DYN, LRU に適用したプリ・プロモーション，RRIP に適用したプリ・プロモーションおよび DRRIP に適用した適応型プリ・プロモーションを実装し，サイクルレベルの評価を行った．プロセッサ命令セットおよびマイクロアーキテクチャパラメタは**表 1** に示した値を用いた．また，キャッシュは**表 2** に示すように LLC を L2 キャッシュにした 2 構成と，L3 キャッシュにした構成を用いており，全ての評価で L2 キャッシュにプリフェッチをかけている．なお，提案手法における enable-flag の初期値は true とする．評価対象のプログラムとしては SPEC CPU2006 の全 29 プログラムを用い，先頭の 10G 命令をスキップ後，その後の 1G 命令を実行して評価した．

表 2 キャッシュ構成

L2 構成	
L1 I/D Cache	32KB, 4way, 64B line, 3cycle latency
L2 Cache (LLC)	256K/1M, 8way, 64B line, 10cycle latency
L2 prefetcher	Stream Prefetcher, degr 16, dist 16
Memory access	200cycle latency
L3 構成	
L1 I/D Cache	32KB, 4way, 64B line, 3cycle latency
L2 Cache	256K, 8way, 64B line, 10cycle latency
L2 prefetcher	Stream Prefetcher, degr 16, dist 16
L3 Cache (LLC)	2M, 16way 64B line, 24cycle latency
Memory access	200cycle latency

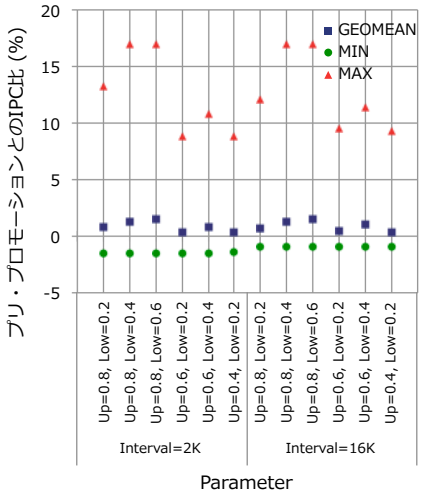

図 10 L3 構成：プリ・プロモーションと適応型プリプロモーションの IPC 比 (%)

表 3 パラメタの概要

up	上の閾値
low	下の閾値
Interval	accuracy を計測するモニタリング期間 (EPL の数)
L2	L2 キャッシュサイズ

6.2 予備評価

まず，適応型プリ・プロモーションの最適な閾値とモニタリング期間を決定するため，パラメタサーチを行った．キャッシュ L2 構成と L3 構成におけるそれぞれの評価結果を図 9 と図 10 に示す．これらの図では，縦軸にプリ・プロモーションとの IPC（Instruction Per Cycle）の比の最大値，最小値，及び幾何平均（GEOMEAN）を取り，横軸は変化させたパラメタを示している．各パラメタを**表 3** に示す．また，図 11 に L2 の 256KB 構成における IPC の幾何平均を示す．縦軸に 29 ベンチマークの IPC の幾何平均，横軸に変化させたパラメタを閾値順を示している．

まず，L2 構成について考察する．図 11 から，Interval の変化による幾何平均への影響が少ないことがわかる．また，図 9 を見ると，他のキャッシュサイズでも同じ傾向が見られる．これらの図から，平均値，最大値共に最もプリ・プロモーションに対する性能向上が見られるのは，256KB 構成を除いてパラメタを Interval=16K, up=0.8, low=0.6 とした場合である．図 10 に示すように，L3 構成についても同じ傾向が見られる．Interval=16K とした場合，2K に比べて EPL-counter と EPLA-counter の保持にハードウェア資源を要するが，これらのカウンタはキャッシュ毎に 1 つずつしか実装されないため，コストが低い．また，モニタリング期間が長い方がより正確に accuracy を算出できると考えられる．これらの傾向をふまえ，本論文では Interval=16K，up=0.8, low=0.6 の構成を最適値として評価に用いる．

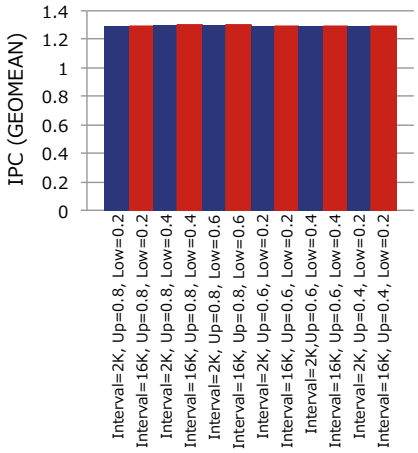

図 11 L2-256KB 構成：IPC(GEOMEAN)

6.3 評価結果

最初に、LLC を L2 とした構成について述べる．まず，L2-256KB 構成，L2-1MB 構成の評価結果を図 12 , 13 に示す．図の縦軸はプリフェッチを適用した LRU との IPC 比を示し，横軸はベンチマークを示している．

図 12 に示す L2-256KB 構成では，提案手法の IPC がプリフェッチを適用した DRRIP に対して最大で 36.5%，幾何平均で 3.9%，PACMan-DYN に対して最大で 13.2%，幾何平均で 2.0%向上している．また，DRRIP に適応したプリ・プロモーションに対して最大で 19.8%，幾何平均で 1.7%向上している．図において，401.bzip, 454.calclix などのベンチマークではプリ・プロモーションは DRRIP に対して IPC の向上が見られなかった．しかし，提案手法ではプリフェッチ精度に基づいてラインを保持するフェーズを捉えることができ，IPC を向上させることができた．また，PACMan-DYN が DRRIP より性能を落としている 410.bwaves, 456.hummer などのベンチマークでも，提案手法は DRRIP から IPC を向上させているため，PAMan-DYN において追い出されていた有用なラインをキャッシュ

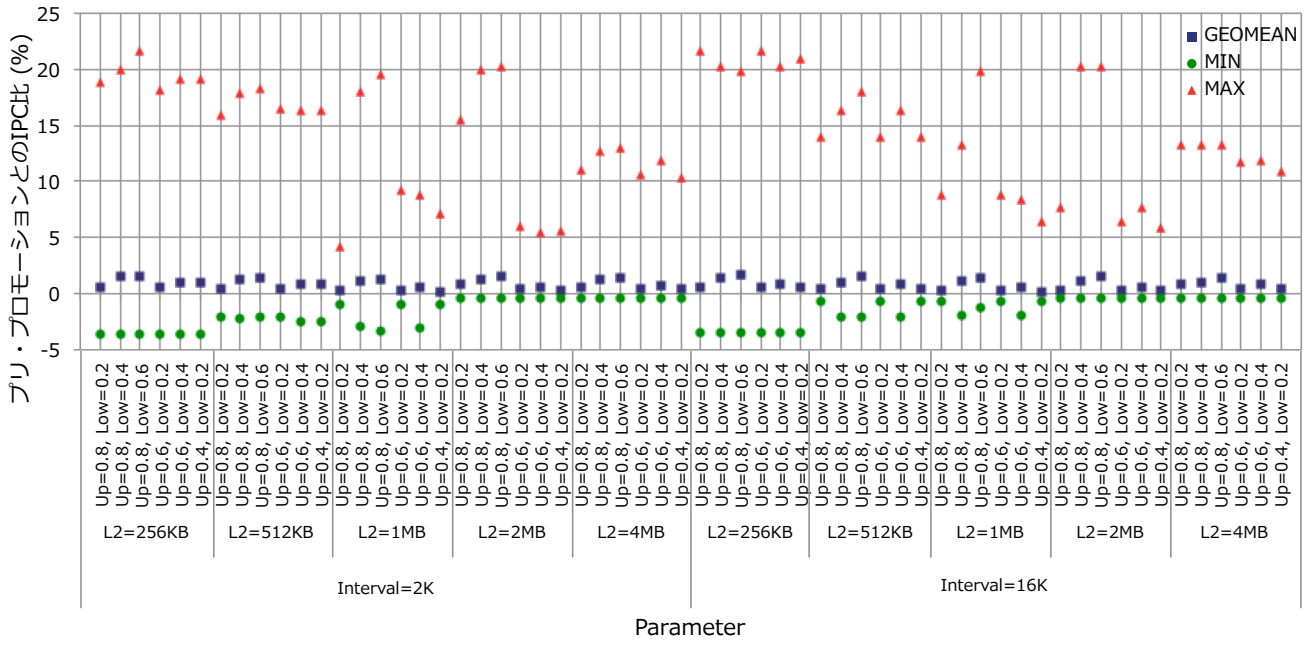

図 9 L2 構成：プリ・プロモーションと適応型プリプロモーションの IPC 比 (%)

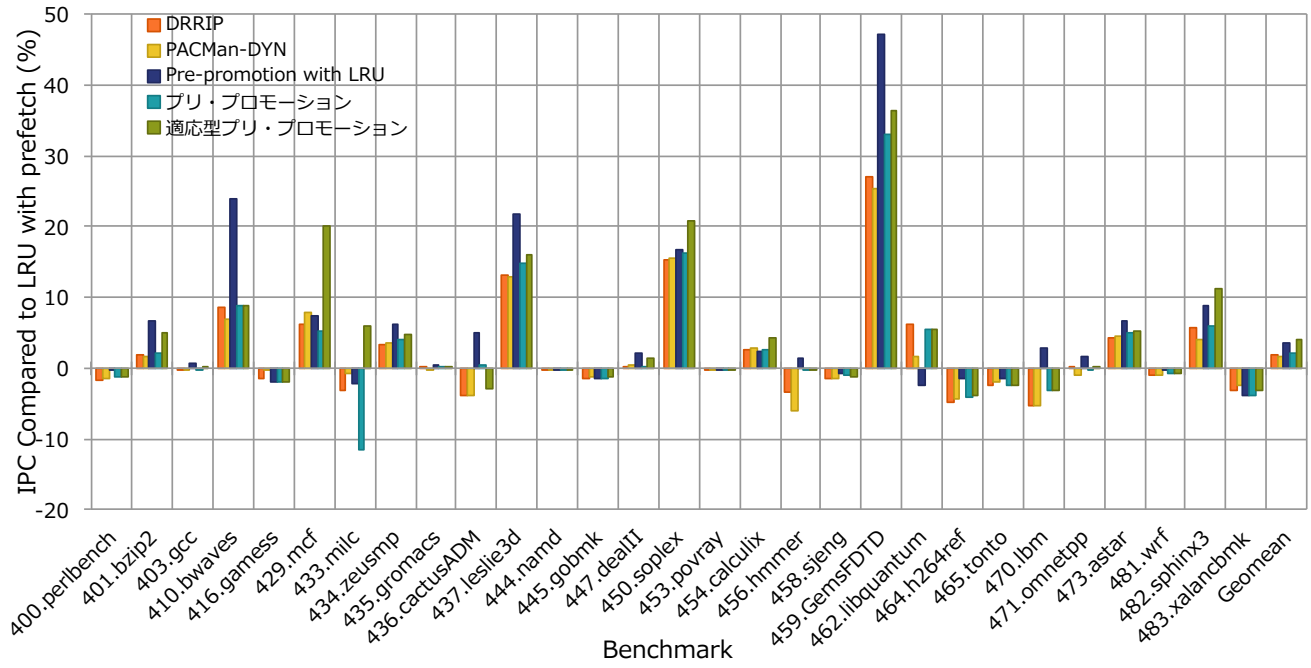

図 12 L2-256KB 構成：LRU with prefetch との IPC 比 (%)

に保持できたと考えられる．

また，図 13 に示す L2-1MB 構成では，提案手法の IPC がプリフェッチを適用した DRRIP に対して，最大で 25.5%，幾何平均で 6.0%向上している．また，PACMan-DYN に対して最大で 18.6%，幾何平均で 1.9%向上している．更に，DRRIP に適応したプリ・プロモーションに対して最大で 19.9%，幾何平均で 1.4%向上している．L2-256KB 構成で LRU よりも IPC の高い DRRIP が，この構成では，LRU よりも IPC が低くなっている．これは，DRRIP とプリフェッチの予測アルゴリズムが衝突したためだと考えられる．しかし提案手法は，459.GemsFDTD をはじめとする 17 本のベンチマークで LRU よりも高い IPC を示し，また 401.bzip2 や 436.cactusADM では DRRIP による性能低下を緩和させている．

ここで，459.GemsFDTD に着目する．まず，図 14 に，459.GemsFDTD における DRRIP，PACMan-DYN，および適応型プリ・プロモーションの IPC の変化と，提案手法で計測した accuracy およびプリ・プロモーションの切

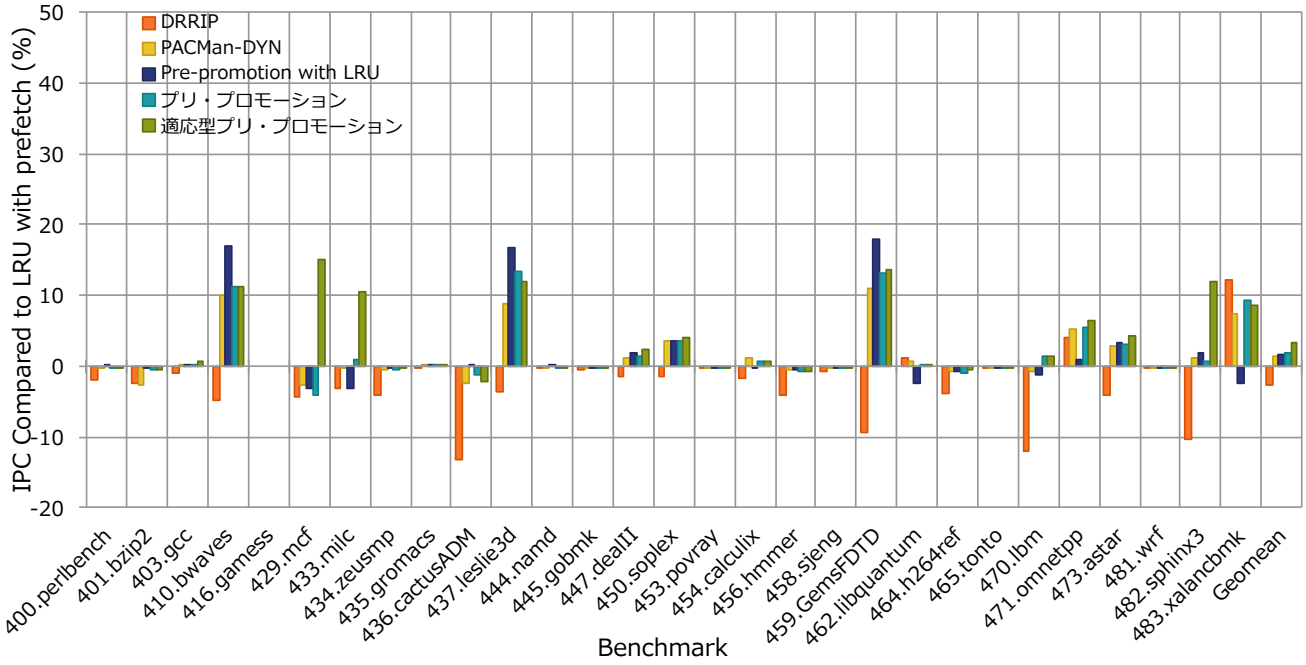

図 13 L2-1MB 構成：LRU with prefetch との IPC 比 (%)

り替えのタイミングを示す．この図では，縦軸に IPC と提案手法で計測した accuracy，横軸は実行命令数を示す．また，プリ・プロモーションの切り替えのタイミングは，図中の三角形によって示されている．更に，図 15 に，(a) DRRIP, (b) PACMan-DYN, (c) 提案手法におけるメインメモリへのミスアクセスとプリフェッチリクエストを示す．この図では，10G 命令実行後 300M 命令実行時のメモリアクセスを示している．また，縦軸にメモリアドレス，横軸に実行命令数が示されている．まず，図 14 を見ると，実行命令数 100M から 200M の間でプリ・プロモーションがオフになり，かつ提案手法の IPC が他のベンチマークに比べて向上している．この時のメモリアクセスを図 15 で見ると，実行命令数 100M から 150M の間でひとつのフェーズが終わって，連続したアドレスにメモリアクセスが発生するフェーズとなる．2 つの図から，フェーズが変わった後に提案手法のミスアクセスが大きく減っていることがわかり，このフェーズにおいてプリ・プロモーションが有用なラインをキャッシュ内に保持できていることがわかる．ここで図 14 を見ると，プリ・プロモーションがオフになった後に IPC が向上している．これはプリ・プロモーションの切り替え前に保持したラインが，このタイミングでキャッシュヒットしたためであると考えられる．

次に，LLC を L3 とした構成について述べる．L3 構成の評価結果を図 16 に載せる．L2 構成の図 12 , 13 と同様に，縦軸はプリフェッチを適用した LRU との IPC 比を示し，横軸はベンチマークを示している．L3 構成では，提案手法の IPC はプリフェッチを適用した DRRIP に対して最大で 22.2%，幾何平均で 3.9%，PACMan-DYN に対して最大

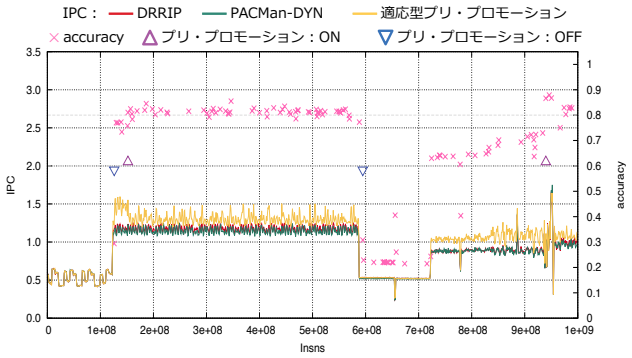

図 14 459.GamsFDTD：フェーズによる IPC の変化

で 13.7%，幾何平均で 1.0%向上している．また，DRRIP に適応したプリ・プロモーションに対して最大で 17.0%，幾何平均で 1.5%向上した．L2 構成と同様に，L3 構成でも DRRIP が LRU に対して性能を低下させているが，ほとんどのベンチマークでは提案手法によって LRU よりも IPC を向上，または DRRIP による性能低下を緩和することができた．

しかし，462.libquantum や 483.xalancbmk において，提案手法は DRRIP や PACMan-DYN よりも低い IPC となっている．ここで，462.libquantum に着目し，性能低下の原因を考察する．図 14 と同様に，図 17 に DRRIP, PACMan-DYN および提案手法おける IPC の変化，提案手法で計測した accuracy とプリ・プロモーションの切り替えのタイミングを示す．図 17 から，PACMan-DYN の性能が一番高いフェーズと，全ての手法がほぼ同じ性能を示すフェーズが順番に現れているのがわかる．このベンチマー

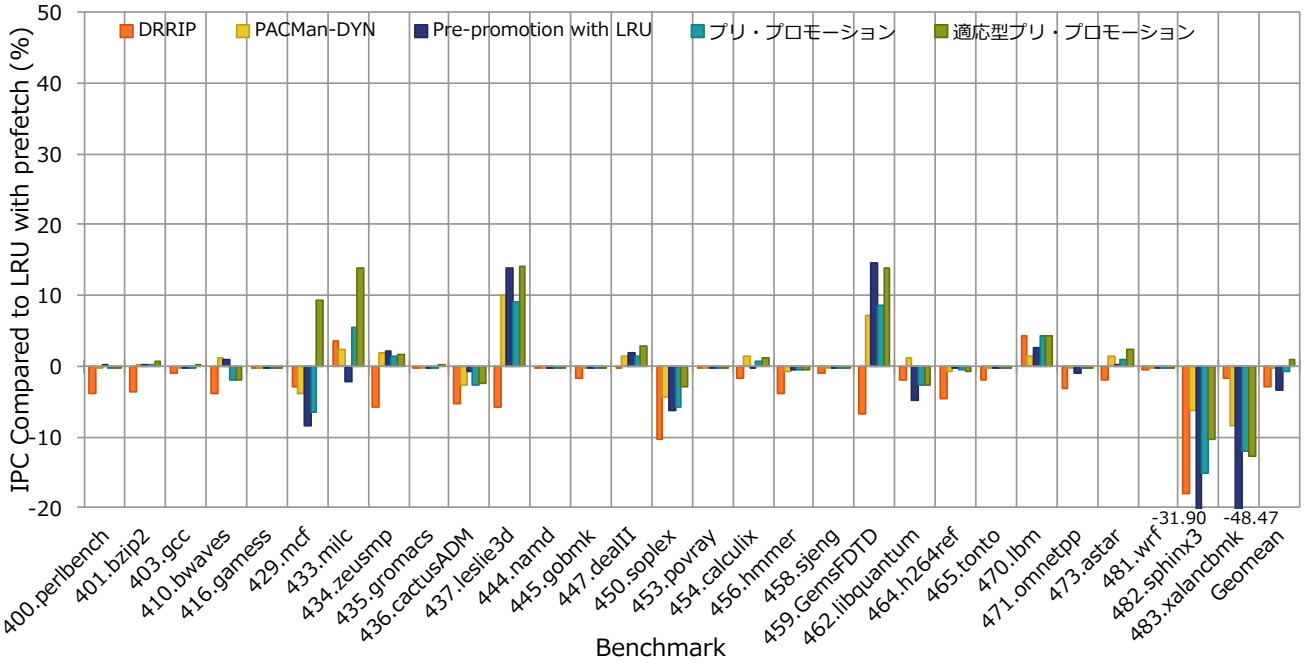

図 16 L3 構成：LRU with prefetch との IPC 比 (%)

クの accuracy は低くないものの，ワークロード自体が非常に小さい．PACMan-DYN ではプリフェッチされたラインを追い出されやすい位置に配置するため，フェーズが切り替わってプリフェッチされたラインが不要になった時にも有用なデータを保持できる．また，accuracy が下がった直後に IPC が上がっていることからも，プリフェッチャの悪影響を受けずに済んでいることがわかる．逆に，提案手法では accuracy が下がった直後の IPC 向上が見られない．これは，プリフェッチャのアドレス予測に基づいて投機的にラインを MRU 側に移動させたものの，利用されることなく次のフェーズに突入してしまい，不要なラインがキャッシュに保持されてしまったのだと考えられる．しかし，accuracy が閾値以下になるほど下がらないために，このようなフェーズでも更にプリ・プロモーションを実行してしまうことで性能が低下したと考えられる．

7. 結論

大容量の LLC の管理アルゴリズムとして，スキャン耐性を持つ置き換えアルゴリズムやプリフェッチが提案されてきた．それぞれ LLC の性能を向上させているが，これらを同時に適用すると 2 つの予測アルゴリズムが衝突し，かえって性能を低下させてしまう．

我々はこの問題に対して，2 つの予測アルゴリズムの衝突を防ぎ，かつプリフェッチと置き換えアルゴリズムとが協調しあうことでキャッシュ性能を高めるプリ・プロモーションを提案した．この手法ではプリフェッチャのアドレス予測を利用し，近い将来にアクセスが見込まれるキャッシュラインを検出し，該当ラインをキャッシュから追い出されにくくする．しかしながら，この手法はプリフェッチャ予測に大きく依存するため，予測精度によって性能が左右されてしまう問題があった．そこで本稿では，プリフェッチャの予測精度を用いてキャッシュラインを保持するか決定する適応型プリ・プロモーションを提案し，RRIP とプリフェッチを組み合わせたキャッシュに実装した．

SPEC CPU2006 を利用し，提案手法で用いる閾値のパラメタサーチと，有効性を確認するための評価を行った．この結果，LLC を 1MB の L2 キャッシュとしたシングルコア構成において，RRIP に対して IPC が最大で 25.5%，GEOMEAN で 6.0%向上し，PACMan-DYN に対して最大で 18.6%，幾何平均で 1.9%向上した．また，DRRIP に適応したプリ・プロモーションに対して最大で 19.9%，幾何平均で 1.4%向上した．今後の課題として，最新のプリフェッチアルゴリズムに対する性能評価や，マルチコア環境での評価などが挙げられる．

謝辞 本研究の一部は，平成 24 年度服部報公会工学研究奨励援助金による．

参考文献

[1] Jaleel, A., Theobald, K. B., Steely, Jr., S. C. and Emer, J.: High performance cache replacement using re-reference interval predivtion (RRIP), *Proc. 37th Annual Int. Symp. on Computer Architecture*, ISCA '10, ACM, pp. 60–71 (2010).
[2] Lai, A.-C. and Falsafi, B.: Selective, Accurate and Timely Self-Invalidation Using Last-Touch Prediction, *Proc. 27th Annual Int. Symp. on Computer Architecture*, ACM, pp. 139–148 (2000).
[3] Khan, S. M., Tian, Y. and Jimenez, D. A.: Sampling

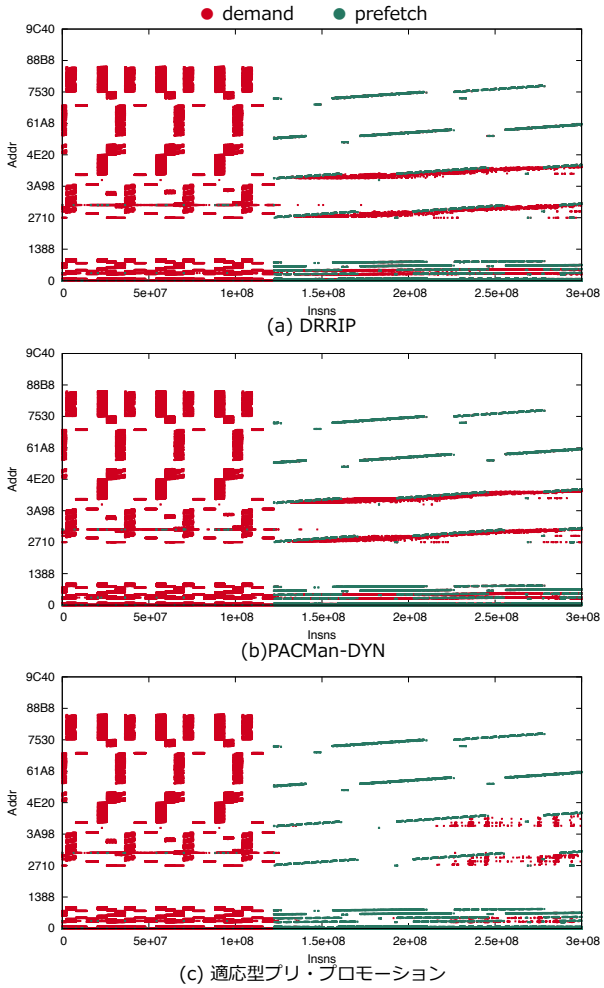

図 15 459.GamsFDTD：メインメモリへのミスアクセスとプリフェッチリクエスト

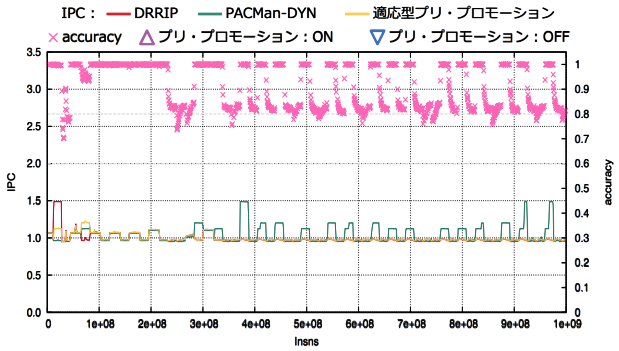

図 17 462.libquantum：フェーズによる IPC の変化

Dead Block Prediction for Last-Level Caches, *Proc. 43rd Annual IEEE/ACM Int. Symp. on Microarchitecture*, IEEE Computer Society, pp. 175–186 (2010).

[4] Lee, D., Choi, J., Kim, J. H., Noh, S. H., Min, S. L., Cho, Y. and Kim, C. S.: LRFU:A Spectrum of Policies that Subsumes the Least Recently Used and Least Frequently Used Policies, *IEEE Trans. on Computers*, Vol. 50, No. 12, pp. 1352–1361 (2001).

[5] A., S.: Sequential Program Prefetching in Memory Hierarchies, *IEEE Computer*, Vol. 11, No. 12, pp. 7–21 (1978).

[6] Fu, W. C. J. and Patel, J. H.: Stride directed prefetching in scalar processors, *Proc. 25th annual IEEE/ACM Int. Symp. on Microarchitecture*, Vol. 23, ACM, pp. 102–110 (1992).

[7] 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，情報処理学会研究会報告，2010-ARC-190, No. 13, pp. 1–8 (2010).

[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 Int. Symp. on Microarchitecture*, ACM, pp. 442–453 (2011).

[9] 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，情報処理学会研究会報告，2013-ARC-206, No. 20 (2013).

[10] Belady, L. A.: A study of replacement algorithms for a virtual-storage computer, *IBM Systems Journal*, Vol. 5, No. 2, pp. 78–101 (1966).

[11] Hu, Z., Kaxiras, S. and Martonosi, M.: Timekeeping in the memory system: predicting and optimizing memory behavior, *Proc. 29th Annual Int. Symp. on Computer Architecture*, IEEE, pp. 209–220 (2002).

[12] Qureshi, M. K., Jaleel, A., Patt, Y. N., Steely, S. C. and Emer, J.: Adaptive insertion policies for hight performance caching, *Proc. 34th Annual Int. Symp. on Computer Architecture*, ISCA '07, ACM, pp. 381–391 (2007).

[13] 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，pp. 120–121 (2009).