Vol.2017-ARC-227 No.12
2017/7/26

プリフェッチラインの再参照間隔を予測するキャッシュマネジメント

甲地 弘幸[1]　入江 英嗣[1]　坂井 修一[1]

概要：CPU と DRAM の速度差を隠蔽するためのキャッシュに関する研究は，これまでに様々行われ，それによりキャッシュの性能が改善されてきた．本論文では，キャッシュに関する研究の内，プリフェッチや置換アルゴリズムに焦点を当てている．近年の研究ではキャッシュ・ラインの再参照間隔を学習，予測し，その予測を用いて置換するラインを決定するといったものが多くある．それらの手法では，プリフェッチ・ラインの再参照間隔は，ほとんどの場合で考慮されておらず，予測したとしても，事前に想定した傾向に当てはめる形で予測しており，学習によって動的に予測する手法は存在していない．しかし，プリフェッチ・ラインをどのように扱うべきかはプログラムごとに大きく異なることから，上手く扱うことによって性能が向上する可能性がある．そのため，我々は，プリフェッチ・ラインの再参照間隔を学習し，予測する手法を提案，評価する．

1. はじめに

半導体技術の進歩により，現在のプロセッサは大規模で高速なものとなっている．しかし，プロセッサの動作速度と比較して，メイン・メモリへのアクセス・レイテンシは数百倍も大きいため，レイテンシを隠蔽するためにプロセッサとメイン・メモリの間に複数のキャッシュ・メモリを挿入するのが普通である．キャッシュ・ミスが発生し，下層のメモリ，特にメイン・メモリへのアクセスが増加してしまうと，レイテンシが隠蔽できずプロセッサの性能が低下してしまうため，ミスをなるべく減らすキャッシュ・マネジメントが重要である．

キャッシュ・マネジメントには，ラインの追い出しを決定する置き換えアルゴリズムと，必要になりそうなデータをあらかじめキャッシュに挿入するプリフェッチが存在する．置き換えアルゴリズムでは，キャッシュ・セット内で最も再参照間隔が大きいものを追い出す OPT[1] が最も性能が良いとされているが，未来のアクセス情報を必要とするため実現が不可能である．そのため，実際の置き換えアルゴリズムでは未来のアクセスを予測し，その中から最も再参照間隔が大きいものを追い出すといった手段をとっている [2], [3], [4], [5], [6]．例えば，Least Recently Used (LRU)[7] では，アクセスがあった履歴が近いラインから再参照間隔が小さいと予測し，履歴が最も遠いラインを最も再参照間隔が大きいとしてキャッシュから追い出す．プリフェッチ [8], [9], [10] は，置き換えアルゴリズムだけでは対処できない初期参照ミスを減らすことが可能である．

近年のプロセッサは，最もメイン・メモリに近いキャッシュである Last Level Cache (LLC) が大容量になり，連想度も大きくなったため，LRU のようなセット内の履歴を全て持つような置き換えアルゴリズムでは，ハードウェア・コストが非常に大きくなってしまう．そのため，省面積を意識する必要がある [2], [11]．また，プリフェッチによるキャッシュへの悪影響を抑えたり，逆に，プリフェッチの情報を最大限用いるようなマネジメントも研究されるようになっている [12], [13], [14], [15], [16]．

本論文では，プリフェッチを認識するマネジメントとして，プリフェッチ・ラインの再参照間隔を予測するキャッシュ・マネジメントを提案する．また，その予測を PrePromotion[13], [17] に適用させることにより，PrePromotion の性能のばらつきを抑えることを目指す．

本論分の構成は以下のようになっている．第 2 章では，LRU に代わって現在の研究のベースとなることが多い Re-Reference Interval Prediction(RRIP)[2] と，それらをベースとした手法である Signature-based Hit Predictor(SHiP)[3]，Prefetch-Aware Cache Management(PACMan)[12]，また，アクセスが迫っている再参照間隔の大きいラインを保護する PrePromotion[13], [18] について述べる．第 3 章では，プリフェッチ・ラインの再参照間隔を予測する提案手法について述べ，第 4 章でシミュレーションによる性能評価と考察を行う．そして，第 5 章

[1] 東京大学大学院情報理工学系研究科

で本稿についてまとめる．

2. 関連研究

2.1 RRIP

まず，最も有名な置換アルゴリズムである Least Recently Used (LRU)[7] の問題点を解決し，現在の多くの研究のベースとなっている Re-Reference Interval Prediction (RRIP)[2] について述べる．

LRU は，メモリ・アクセスの時間的局所性に着目した手法であり，現在でもよく使われているが，最もメイン・メモリに近いキャッシュである Last Level Cache(LLC) は，近年では容量が増え，連想度が大きくなっているため，LRU に必要なハードウェア・コストが増大している．そこで，巨大な LLC に適用させる手法として RRIP が提案された．RRIP は，ラインごとに追加するメタデータ用のビットがキャッシュの連想度によらず一定であり，大容量のキャッシュにも適用できる上，LRU を超える性能を出すことが可能である．以下，具体的な動作を述べる．

RRIP では，Re-Reference Prediction Value(RRPV) と呼ばれる，再参照間隔を予測するためのビットを各キャッシュ・ラインに付与する（以下簡単のために 2 ビットの情報とする）．付与したビットの値が大きいときには再参照が遠い，小さいときには再参照が近いと予測する．RRIP では，主に以下の 3 つの動作が存在する．

(1) Static RRIP(SRRIP)
(2) Bimodal RRIP(BRRIP)
(3) Dynamic RRIP(DRRIP)

SRRIP と BRRIP はキャッシュ・ミスが発生したときの動作のみが異なり，DRRIP は SRRIP と BRRIP を後述の Set Dueling Monitor(SDM)[19] によって動的に切り替えた手法である．RRIP 全体に共通した動作としては，ヒットした場合には，ヒットしたラインの RRPV を 0 にセットし，ミスした場合には RRPV が 3 のラインを追い出し，新しいラインを挿入する．このとき，RRPV が 3 であるラインが存在しない場合は，全てのラインの RRPV をインクリメントする．挿入するラインの RRPV は，SRRIP の場合は 2，BRRIP の場合は高確率で 3，低確率で 2 にセットする．SRRIP は，挿入時の RRPV を中程度にすることで，追い出しまでの猶予を持たせることが出来る．追い出されるまでの間に参照された場合，再参照が近い未来にくると予測する．このようにして，再参照間隔を学習し予測できる．BRRIP はある程度の範囲のメモリ・アクセスが繰り返すスラッシング・アクセスに対して耐性を持つことができる．

DRRIP は，アクセスパターンが切り替わっても対応できるように SRRIP と BRRIP をプログラムの動作中に動的に切り替える手法である．ここで，SDM とは，キャッシュ内のセットからいくつかセットを選び，互いに異なる置換アルゴリズムで動作させて性能を比較した上で，性能が良いものを残りのセットに適用するといった手法である．例えば，DRRIP の場合，キャッシュの 1/32 を SRRIP，別の 1/32 を BRRIP とし，それぞれのセット内でのミス数をカウントし，ミスが少ない方の置換アルゴリズムをキャッシュの残りの部分に適用する．

2.2 SHiP

RRIP では，SRRIP と BRRIP を SDM によって切り替えることによって，異なるアクセスパターンに耐性を持たせることに成功しているが，Signature-based Hit Predictor (SHiP)[3] は挿入時に，ラインの再参照間隔をあらかじめ予測して，挿入位置を切り替える手法である．そのため，キャッシュ・ヒット時の動作は変化させない．また，SHiP は，必ずしも RRIP をベースとする必要がなく，LRU にも適用することが可能である．なお，SHiP は，Cache Replacement Championship の規定した環境でシミュレーションされているため，プリフェッチが存在しない前提で考えられている．

SHiP では，ある特徴量に着目し，その特徴量が同じときのラインの振る舞いが同じであると考えている．そこで，各特徴量をインデックスにもつテーブル Signature History Counter Table(SHCT) を用いて，特徴量ごとの再参照間隔を学習する．また，学習するために，各キャッシュ・ラインに特徴量を保持する signature と，再参照があったかどうかを表す outcome 用のビットを用意する．SHCT の各エントリが持つ値が 0 だった場合は，挿入するラインの再参照が遠いと推測し，そうでない場合は再参照がやや近いと推測する．例えば，SRRIP に SHiP を適用させた場合，SHCT のエントリの値が 0 の場合には，RRPV を 3 にセットして挿入し，そうでない場合には RRPV を 2 にセットして挿入する．

再参照間隔の学習は，キャッシュ・ヒットした場合，ヒットしたラインの outcome を 1 にし，ラインの持つ signature をインデックスとして SHCT にアクセスし，対応するエントリの値をインクリメントする．キャッシュ・ミスした場合，追い出すラインの outcome を参照する．0 の場合は，そのラインの持つ signature をインデックスとして SHCT にアクセスし，値をデクリメントする．outcome が 1 の場合は SHCT の更新を行わず，挿入するラインの outcome を 0，signature をそのときの特徴量とし，上述の方法で挿入位置を決定し，挿入する．SHiP 全体の構成は**図 1** のようになっている．

SHiP には以下の 3 つの動作が存在する．

(1) SHiP-Mem
(2) SHiP-PC
(3) SHiP-ISeq

それぞれ，SHiP-Mem は，アクセスするメモリ・アドレ

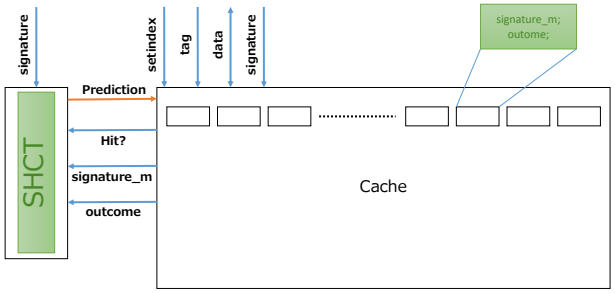

図 1　SHiP Structure[3]

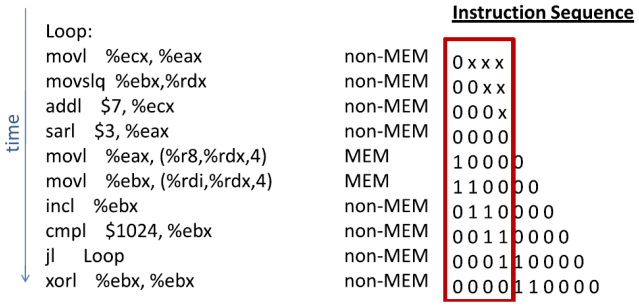

図 2　Instruction Sequence History[3]

スの下位ビット，SHiP-PC は，メモリ・アクセスの PC の下位ビット，SHiP-ISeq は命令列の一部を特徴量として用いる．ここで，命令列の一部とは，**図 2** に示すように，メモリ命令を 1，それ以外の命令を 0 として並べた文字列を特徴量として用いる．

SHiP は，特徴量用に使うビットはデフォルトで 14bit，outcome に 1bit，SHCT の各エントリに 3bit 使用している．そのため，1MB のセット数 1024，連想度 16 の LLC の場合 30KB もの資源が必要となる．そこで，性能をあまり落とさずに，ハードウェア資源を減らすために，1024 セットの内，64 セットのみで SHCT の学習を行う SHiP-S や，SHCT の各エントリに用いるビットを減らす SHiP-R などが考案されている．SHiP-S はデフォルトに比べて性能を落とすが，SHiP-R は，学習速度が速くなるため，性能が向上する場合がある．

2.3　PACMan

Prefetch-Aware Cache Management (PACMan)[12] は，RRIP をベースとした手法であり，プリフェッチャが存在する前提で考えられている．PACMan では，プリフェッチされたラインは再参照間隔が非常に大きいと考えており，キャッシュが参照のされないプリフェッチ・ラインによって満たされ，汚染されてしまうのを防ぐ手法として考案された．PACMan には，以下の 4 つの動作が存在する．

(1) PACMan-H
(2) PACMan-M
(3) PACMan-HM
(4) PACMan-DYN

PACMan-H や，PACMan-M の H や M はそれぞれ Hit，Miss を意味し，ヒット，ミスした時の動作が RRIP と異なっている．

PACMan-H では，デマンド・ラインへの扱いは，通常の RRIP と同じであるが，プリフェッチ・ラインにヒットした場合には，RRPV を変化させない．また，PACMan-M では，同様にデマンド・ラインへの扱いは変化させずに，キャッシュ・ミスが発生した時に，プリフェッチ由来のラインのみ，RRPV を 3 にセットして挿入する．PACMan-HM は上述した，2 つの動作を単純に組み合わせたものである．最後，PACMan-DYN は，SDM によって，動作を動的に切り替える手法である．ここで，切り替える候補は，

(1) SRRIP + PACMan-H
(2) BRRIP + PACMan-H
(3) SRRIP + PACMan-HM

の 3 つである．Wu らは，BRRIP と PACMan-M は動作が近いと考え，その組み合わせを除き，また，PACMan-H は総じて優秀であったため，上記の 3 つの組み合わせとなっている．

2.4　PrePromotion

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を元に推定しているため，アクセスが迫っているラインが追い出されてしまう可能性がある．そこで，PrePromotion[13], [18] 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．また，SHiP と同様に，元となる手法は特に制限されず，LRU，RRIP，PACMan など様々な置換アルゴリズムに適用することが可能である．

実際の動作を**図 3** 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 3 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（例えば，RRPV をデクリメントする）する．この操作をプリプロモートという．

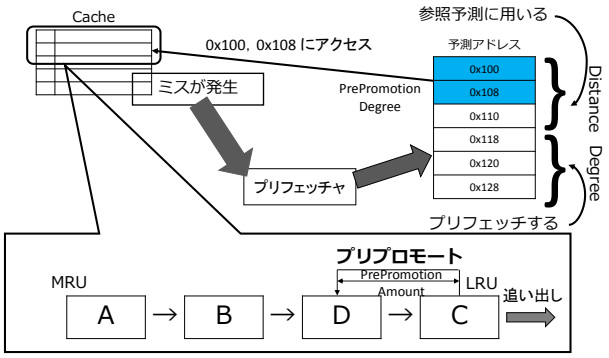

図 3　Behavior of the Pre-Promotion

3. 提案手法

3.1 モチベーション

現在のキャッシュ・マネジメントでは，
(1) 再参照間隔の予測をどうするか
(2) プリフェッチの扱いをどうするか
(3) 追加のハードウェア資源をどう少なくするか
に焦点を当てて研究されている．RRIP，SHiP では主に 1 と 3 を，PACMan では 主に 2 を，PrePromotion では 1，2，3 に焦点を当てている．SHiP では，プリフェッチャのことを考慮しておらず，現在はプリフェッチャが備わったアーキテクチャが一般的となっているため，デマンド・ラインの再参照間隔の予測精度を上げただけでは不十分である．また，PACMan では，プリフェッチ・ラインへの再参照がほとんどない，すなわち再参照間隔が広いことに着目して，プリフェッチ・ラインをなるべく早くキャッシュから追い出すようにしている．しかし，アーキテクチャの構成を**表 1** として，L2，L3 キャッシュの置換アルゴリズムを DRRIP としてとった参照の傾向に関する図である**図 4**，**図 5** を見ると，L2 キャッシュでは，挿入後の参照可能性がプリフェッチ・ラインの方が高く，L3 ではデマンド・ラインの方が高いが，その傾向差は数 % に収まる範囲である．ここで，デマンド・ラインは，挿入時点で利用されたデータであり，プリフェッチ・ラインは，挿入時点では利用されていないデータである．そのため，キャッシュに挿入後の最初の参照が，デマンド・ラインでは再参照，プリフェッチ・ラインでは最初の参照であるため，デマンド・ラインの再参照の予測と，プリフェッチ・ラインの参照の予測は，本質的には同じになる．PrePromotion は，プリフェッチの効果をより引き出す手法として，参照が迫っているラインを保護するのを目的として提案された．しかし，PrePromotion は性能が大きく上がるベンチマークと大きく下がるベンチマークが存在するため，プログラムの実行中にオンとオフを動的に切り替えることが必要である．[17]

以上の点を踏まえて，プリフェッチ・ラインの再参照間隔を予測するキャッシュ・マネジメントを提案する．この手法では，従来の手法と異なり，デマンド・ラインだけでなく，プリフェッチ・ラインの再参照間隔を動的に予測することにより，プリフェッチによるキャッシュ汚染を防ぎつつ，プリフェッチャの性能を最大限引き出すことが期待される．

図 6 では，アーキテクチャの構成を表 1 にし，L2 キャッシュの置換アルゴリズムを LRU，L3 キャッシュの置換アルゴリズムを SHiP + SRRIP (以下，単に SHiP というときは，SHiP + SRRIP を指す) とし，プリフェッチ・ラインの挿入位置を RRPV = 2 で固定したものと，RRPV = 3 で固定したものの Instructions Per Cycle(IPC: 1 サイクルあたりに実行した命令数) を RRPV = 2 とした場合をベースとして比較している．これを見ると，bwaves，mcf，milc，zeusmp，leslie3d，soplex，GemsFDTD，astar では，RRPV = 3，すなわちプリフェッチ・ラインを追い出しやすい位置に挿入すると性能が大きく下がるが，sphinx3，xalancbmk では，プリフェッチ・ラインを少し追い出しにくい位置に挿入する方が良いことがわかる．また，この傾向は，PrePromotion [17] でも見られた傾向であり，プリフェッチ・ラインの参照予測がそのまま PrePromotion にも適用できることが考えられる．

3.2 提案機構

主な構成は次のようになる．ベースとする置換アルゴリズムは RRIP とする．再参照間隔の予測には，特徴量をインデックスとする学習テーブルを用いる．Load/Store 由来のキャッシュ・アクセスは，PC を伴うが，プリフェッチ由来の場合は，PC を伴わないため，テーブルにアクセスするための PC 以外の特徴量を考える必要がある．そこで，プリフェッチャをトリガした PC またはアドレスと，プリフェッチしたラインのアドレスをインデックスとする．プリフェッチャとして，Srinath らによる Stream Prefetcher[9] を用いる．この場合，トリガしたアドレスはストリーム・アクセスの先頭を表すことになるため，ストリームごとの特性を学習するのに必要だと考えられるからである．また，学習のためには，デマンド由来のラインかプリフェッチ由来のラインかを区別する必要があるため，prefetched ビットを追加する．この値を参照することで，デマンド用のテーブルと，プリフェッチ用のテーブルのどちらの学習を行うかを判別する．**図 7** に提案の構造を示す．学習の方法は，SHiP と同様に設けた outcome ビットを参照して行う．

予測を適用させる方法は次のようになる．まず，キャッシュ・ミスが発生時に，prefetched ビットを参照し，各テーブルに問い合わせる．次に，インデックスが指し示すエントリの持つ値が 0 の場合は再参照間隔が大きいと予測できるため，デマンド・ラインの場合は SHiP と同様

に RRPV = 3 として挿入する．プリフェッチ・ラインの場合は，図 4，図 5 を見るとわかる通り，大部分がアクセスされず追い出されるので，RRPV = 3 として挿入．エントリの持つ値が正の場合は，デマンド・ライン，プリフェッチ・ライン共に RRPV = 2 として挿入する．また，PrePromotion のオンとオフに関しても，同様にテーブルにアクセスし，エントリの持つ値が 0 の場合はオフ，正の場合はオンにする．図 6 の通り，プリフェッチ・ラインは多くの場合 RRPV = 2 に固定した方が性能が良いため，テーブルの初期化も，初期状態で RRPV = 2 とできるように，1 以上の値で初期化する．

4. 性能評価

性能を評価するために，シミュレータとして，サイクル・アキュレートなシミュレータである鬼斬弐 [20] を用いた．アーキテクチャの構成は表 1 のようにし，L2 キャッシュの置換アルゴリズムは LRU とし，L3 キャッシュの置換アルゴリズムを変えて評価を取った．ベンチマークは SPEC CPU2006[21] の全ベンチマークを用い，各ベンチマークの先頭の 10G 命令をスキップし，1G 命令シミュレーションした．評価を**図 8** に示す．図の横軸は SPEC CPU2006 のベンチマークを表し，縦軸は，LRU + Prefetch をベースとした，相対 IPC である．また，凡例の SHiP と SHiP’，Proposal と Proposal’ との違いは，学習テーブルの初期化を 0 にするか，2 にするかが異なっている．今回評価に用いた SHiP は，PC の下位から 14bit を抜き出したものを特徴量としても用いており，Proposal では，プリフェッチをトリガしたアクセスの PC の下位 14bit と，プリフェッチするアドレスの上位 14bit に XOR をかけたものをインデックスとして，プリフェッチの参照を学習している．

図 8 を見ると， LRU に比べて，全ての手法の性能が下がっているが，今回 SHiP を実装するにあたり，論文では PC からどのようにして特徴量を抽出するか明らかにされていないため，上述したように，考えられるものとして PC の下位 14bit を学習のために用いて実装したが，LRU よりも性能が低下してしまっている．

まず，学習テーブルの初期化の差異について述べる．それぞれ，SHiP も Proposal も，概ねテーブルの初期化を 2 にした方が性能が良く出ており，特に，astar においてはその差が顕著であることがわかる．このことから，学習がある程度進むまでは，RRPV = 3 として，追い出しやすい位置に挿入するよりも，RRIP の考えに従い，RRPV = 2 として挿入し，テーブルが学習するのを待つのが良いことが分かる．次に，SHiP と Proposal について述べる．この 2 つは，幾何平均で見ると大きな差異はないが，第 3 章で述べたプリフェッチ・ラインを RRPV = 2 で挿入したいベンチマークでは，Proposal の性能が下がっており，RRPV = 3 としたいベンチマークでは，Proposal の方が性能が良くなっている．特に，leslie3d では，Proposal の方が性能が大きく低下しており，xalancbmk では，その逆になっている．ここで，leslie3d において，シミュレーション終了する直前のプリフェッチ用のテーブルの学習状況を確認したところ，0 であるエントリが 2611/16384 = 16%，そうでないエントリが 13772/16384 = 84% であり，また，13772 エントリの内，12159 エントリの持つ値が 2 であったため，インデックスが一部に偏ったことを考えると，プリフェッチ・ラインの大部分を RRPV = 3 として挿入していたことが考えられる．今回，プリフェッチ用のテーブルの学習に用いた特徴量は前述したとおりであるが，よりプリフェッチ・ラインの振る舞いを特徴付ける特徴量を考える必要があることがわかった．

5. おわりに

現在のキャッシュ・マネジメントの研究では，ラインの再参照間隔を予測に焦点を当てた研究や，プリフェッチの影響を考慮した研究などがある．本稿では，ラインの再参照間隔の予測をプリフェッチ由来のラインにも適用させることにより，更にプリフェッチャの性能を引き出すキャッシュ・マネジメントを提案した．実際に実装して評価したところ，今回提案した特徴量は，プリフェッチ・ラインの振る舞いを正しく捉えることができていないことがわかったため，より良い特徴量を探す必要があることがわかった．今後の課題は，より良い特徴量を見つけ，それにより性能を上げ，その予測を今度は PrePromotion の動的切り替えに適用させることや，SHiP と異なる学習方式を提案することが考えられる．

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] Belady, L. A.: A study of replacement algorithms for a virtual-storage computer, *IBM Systems Journal*, Vol. 5, No. 2, pp. 78–101 (online), DOI: 10.1147/sj.52.0078 (1966).

[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] 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, *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-44, New York, NY, USA, ACM, pp. 430–441 (online), DOI: 10.1145/2155620.2155671 (2011).

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

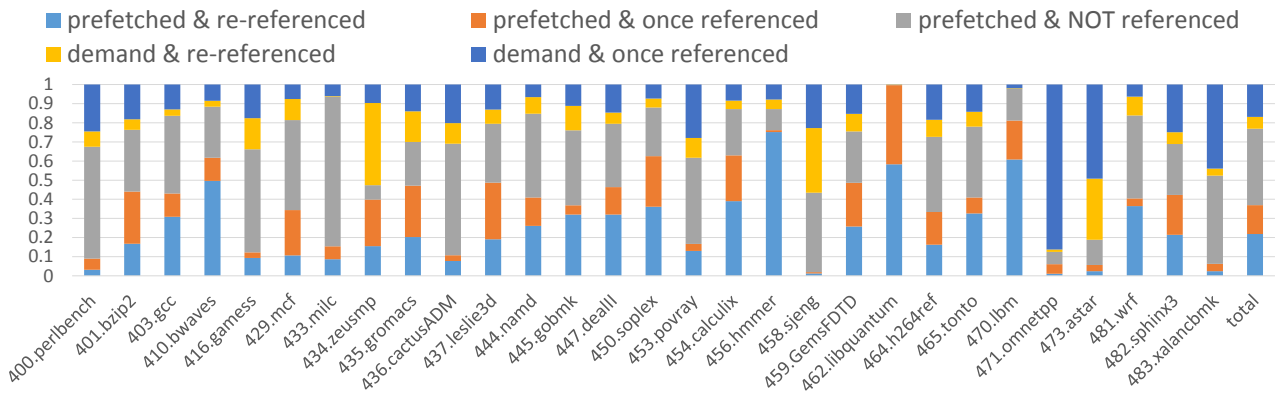

図 4　Reference Characteristic on L2 Cache

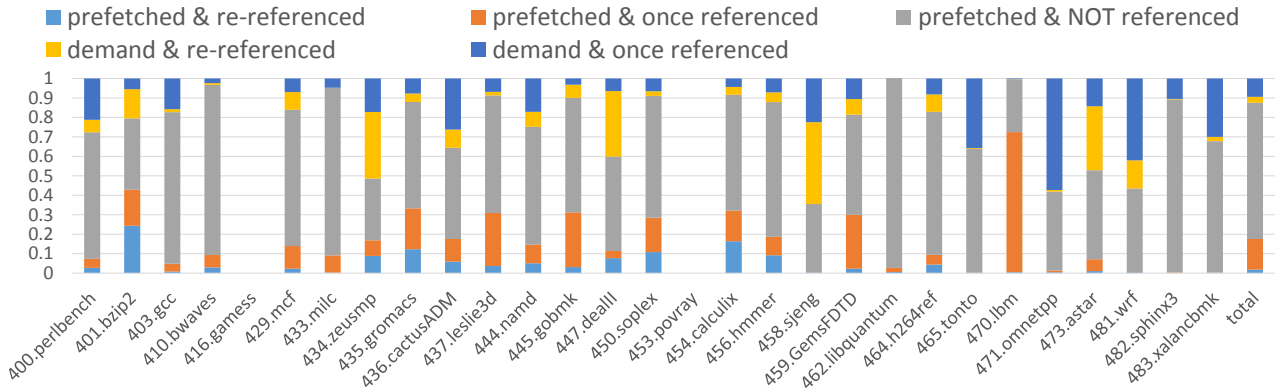

図 5　Reference Characteristic on L3 Cache

(online), DOI: 10.1109/MICRO.2010.24 (2010).

[5] Teran, E., Wang, Z. and Jimnez, D. A.: Perceptron learning for reuse prediction, *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 1–12 (online), DOI: 10.1109/MICRO.2016.7783705 (2016).

[6] Jain, A. and Lin, C.: 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 (online), DOI: 10.1109/ISCA.2016.17 (2016).

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

[8] 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).

[9] Srinath, S., Mutlu, O., Kim, H. and Patt, Y. N.: Feedback Directed Prefetching: Improving the Performance and Bandwidth-Efficiency of Hardware Prefetchers, *2007 IEEE 13th International Symposium on High Performance Computer Architecture*, pp. 63–74 (online), DOI: 10.1109/HPCA.2007.346185 (2007).

[10] 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).

[11] Al-Zoubi, H., Milenkovic, A. and Milenkovic, M.: Performance Evaluation of Cache Replacement Policies for the SPEC CPU2000 Benchmark Suite, *Proceedings of the 42Nd Annual Southeast Regional Conference*, ACM-SE 42, New York, NY, USA, ACM, pp. 267–272 (online), DOI: 10.1145/986537.986601 (2004).

[12] 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).

[13] 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).

[14] Ishii, Y., Inaba, M. and Hiraki, K.: Unified Memory Optimizing Architecture: Memory Subsystem Control with a Unified Predictor, *Proceedings of the 26th ACM International Conference on Supercomputing*, ICS '12, New York, NY, USA, ACM, pp. 267–278 (online), DOI: 10.1145/2304576.2304614 (2012).

[15] 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).

[16] Zhang, K., Wang, Z., Chen, Y., Zhu, H. and Sun, X. H.: PAC-PLRU: A Cache Replacement Policy to Salvage Discarded Predictions from Hardware Prefetchers, *2011 11th IEEE/ACM International Symposium on Cluster,*

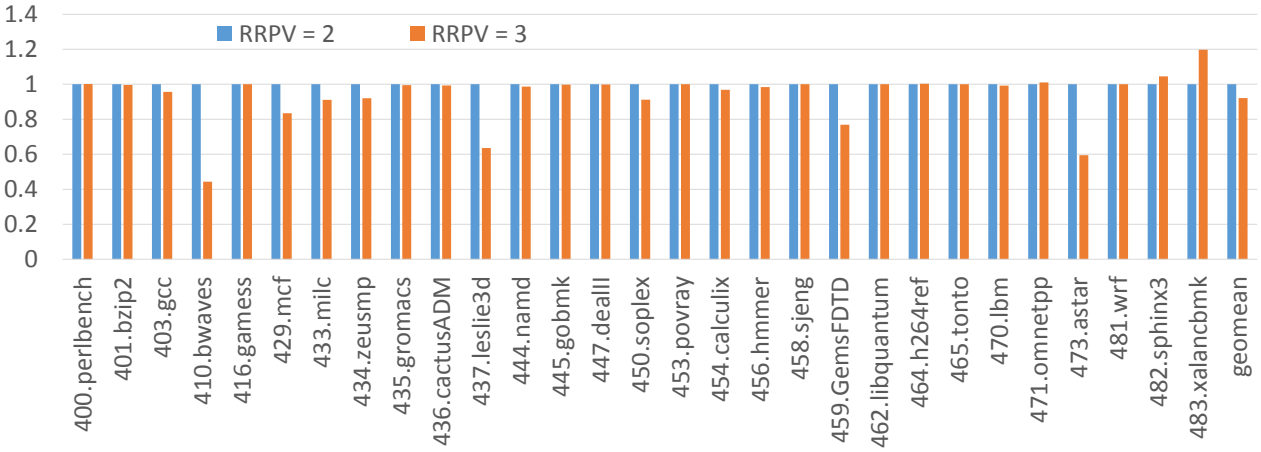

図 6 Change Prefetched Lines' Insertion Position

表 1 Architecture

Processor	
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
Cache	
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, 64B line, 24cycle latency
メモリ・アクセス	200cycle latency
L2 プリフェッチャ	Stream Prefetcher, Distance:16, Degree:16

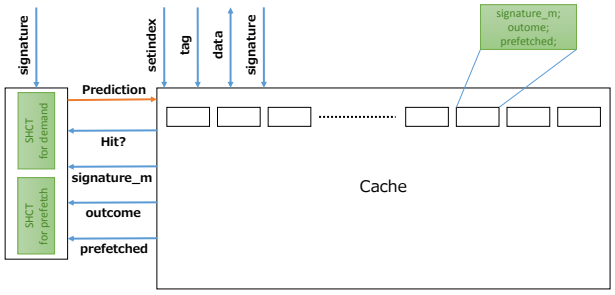

図 7 Proposal Structure

Cloud and Grid Computing, pp. 265–274 (online), DOI: 10.1109/CCGrid.2011.27 (2011).

[17] 甲地弘幸，入江英嗣，坂井修一：Pre-Promotion の動的切り替え手法の検討 (コンピュータシステム)，電子情報通信学会技術研究報告 = IEICE technical report : 信学技報， Vol. 116, No. 177, pp. 103–107（オンライン），入手先 ⟨http://ci.nii.ac.jp/naid/40020932172/⟩ (2016).

[18] 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).

[19] Qureshi, M. K., Jaleel, A., Patt, Y. N., Steely, S. C. and Emer, J.: Adaptive Insertion Policies for High Performance Caching, *Proceedings of the 34th Annual International Symposium on Computer Architecture*, ISCA '07, New York, NY, USA, ACM, pp. 381–391 (online), DOI: 10.1145/1250662.1250709 (2007).

[20] 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).

[21] The Standard Performance Evaluation Corporation: SPEC CPU2006 suite. `http://www.spec.org/cpu2006/`.

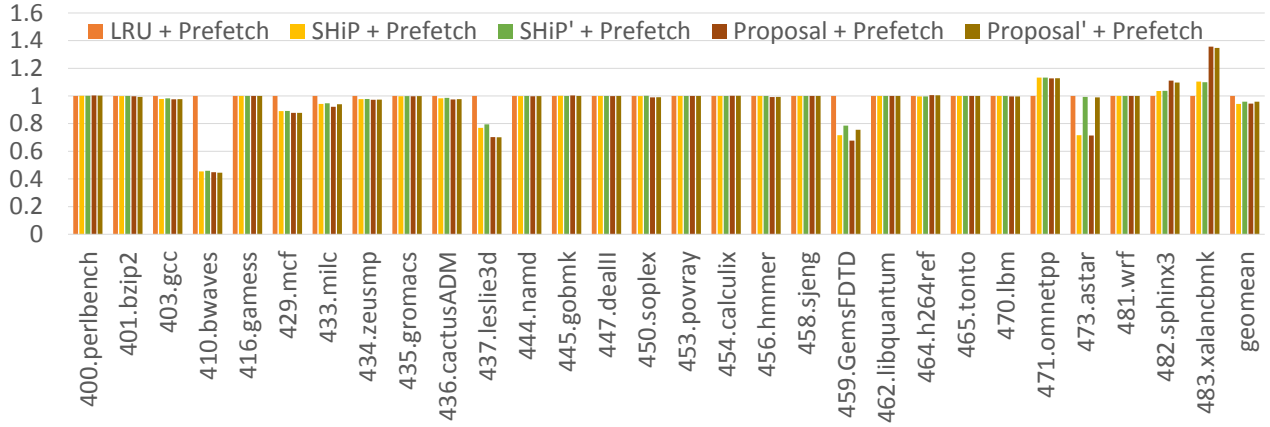

図 8 Evaluation