Vol.2014-ARC-209 No.5
2014/3/7

メモリアクセス履歴を波形解析することによるデータプリフェッチの検討

花田 高彬[1,a)]　村上 和彰[2]

概要：本稿では，反復的および非反復的なメモリアクセス・パタンが混在するワークロードにおいても有効なデータプリフェッチ手法について検討する．将来必要とされるデータを事前にフェッチするデータプリフェッチは，主記憶アクセスの待ち時間を削減する手段として広く知られている．これは，参照先アドレスが規則的に推移する点を活かした手法である．既存プリフェッチ手法は，規則的なアクセスパタンのうち，過去に現れたアクセス履歴が反復的に再出現する特徴に着目している．本稿ではこれを拡張し，規則性はあるが反復的には出現しないアクセスパタンにも対応可能なプリフェッチ手法を検討する．検討手法では，アクセス履歴を合成波形として取扱い，反復的および非反復的パタンが混在する複雑なアクセス履歴をオンタイムに分解する．分類されたパタンに基づいたプリフェッチにより，複雑なアクセスパタンにおいても高精度なプリフェッチが期待される．

1. はじめに

コンピュータ・システムにおいて，データ・プリフェッチ (以降本稿では，単にプリフェッチと呼称) は主記憶アクセスによるプロセッサの待ち時間を削減する有効な手法として広く活用されている [6]．プリフェッチは，参照先アドレスの規則性に着目し，後にロードされると予想されるデータを事前に下位メモリ階層 (主記憶など) から読み出す操作である．プリフェッチを行うことで，上位メモリメモリ階層 (キャッシュ・メモリなど) に事前にデータが保存され，主記憶アクセス待ち時間が削減される [14]．

プリフェッチの中でも，本稿ではハードウェア・プリフェッチに着目する．これは，プリフェッチ対象アドレスをコード中に記述するアプローチ (ソフトウェア・プリフェッチ) とは異なり，アクセス・パタンの規則性に基づきハードウェアが将来の参照先アドレスを予めフェッチする手法である．利用されるアクセス・パタンとしては，参照先アドレスもしくは一つ前の参照先アドレスとの差分 (ストライド値) の履歴である．ハードウェア・プリフェッチャはアクセス・パタンを定期的に保存しておき，プリフェッチが有効であるアクセス・パタンが表れた場合にはプリフェッチが行われる．

ハードウェア・プリフェッチの研究は過去に様々なアプローチで取り組まれている．たとえば，ストライド値が反復的に出現する性質に着目したストライド・プリフェッチやデルタ・コルレーション・プリフェッチ，ストライド値の出現確率に基づくマルコフ・プリフェッチなどが挙げられる [2][5][9][8][10][12][13]．これらの手法では，あるアクセスパタンが始まるとその後一定期間は同じアクセスパタン (反復的アクセスパタン) が継続して出現する傾向に着目し予測を行う．これら既存研究のアプローチでは，予測精度を向上させるために長いアクセスパタン履歴が必要であり，メモリ面積がオーバーヘッドとなる．また，反復的アクセスパタン以外の規則的なアクセスパタンを予測できず，そのようなアクセスパタンが混在する場合では予測精度が低下する．

本稿では，既存研究とは異なるアプローチをとり，省メモリかつ高精度な予測を実現するハードウェア・プリフェッチ手法を検討する．本手法では，複雑なアクセスパタンに対してフィルタリングを行うことで，特性の異なるストライド・ストリームへと分割する．具体的には，自己相関と移動平均を用いることで，アクセスパタン中に存在する反復的ストライド・ストリームと，非反復的ストライド・ストリーム，そしてノイズ成分を抽出する．これら抽出されたストライド・ストリームに基づき，次の参照アドレス予測を行う．検討手法は既存手法に比べ，同等の予測精度を維持しつつ，メモリ面積の削減を期待できる．

[1] 九州大学 大学院システム情報科学府
Graduate School of Information Science and Electrical Engineering, Kyushu University
[2] 九州大学 大学院システム情報科学研究院
Faculty of Information Science and Electrical Engineering, Kyushu University
[a)] hanada@soc.ait.kyushu-u.ac.jp

本稿の残りは以下の構成である．第 2 節にて，既存のプリフェッチ手法を紹介する．次に第 3 節にて，メモリ・アクセスパタンの分類を行い，本稿にて検討するプリフェッチ手法について述べる．第 4 節にて，検討手法の有効性評価を行い，最後に第 5 節にて，本稿をまとめる．

2. ハードウェア・プリフェッチ

2.1 準備

ハードウェア・プリフェッチ問題は，過去の参照先アドレスを入力とし，将来の参照先アドレスを出力とする問題である．上記ハードウェア・プリフェッチ問題に対する解法として，様々なデータ・プリフェッチ手法が過去に研究されている．特に，アクセス・パタンの規則性に基づき，プリフェッチするアドレスを予測する手法について多数研究されている．本稿では，既存のプリフェッチ手法と同様に，入力となる参照先アドレスは過去のキャッシュ・ミス・アドレスを用いるものとする．

ハードウェア・プリフェッチを行う機構 (ハードウェア・プリフェッチャ) は，図 1 に示される構成であると想定する．キャッシュ・ミスが発生すると，ミスが発生したアドレスは予測器に格納される．過去のミス・アドレスと最新のミス・アドレスに基づき次にキャッシュ・ミスが発生するアドレスを予測し，事前に主記憶からデータをフェッチする．フェッチされたデータはプリフェッチャ内のデータ格納バッファ (ストリーム・バッファ) に一時的に格納される．次のキャッシュ・アクセス時，キャッシュ・ミスが発生したとしても，次にストリーム・バッファが参照される．当該アドレスがストリーム・バッファ中に存在するならば，そのデータは上位メモリ階層へとフェッチされる．また，プリフェッチャはそのデータをキャッシュへと転送する．

予測器は各プリフェッチ手法の特色が表れる機構であり，予測精度や実装時の記憶回路面積などに影響を与える．特に予測精度の改善を最大の目的として，過去に複数の研究が取り組まれてきた．次節にて紹介する．

2.2 既存研究

第 2.1 節にて述べたハードウェア・プリフェッチ問題に対する解法として，様々なデータ・プリフェッチ手法が過去に研究されている．過去の手法は，アドレス予測に用いるパラメータ，パタン・マッチング時の仮説，ならびにクラスタリングの有無によって分類できる．

ナイーブなプリフェッチ手法においては，検出するパラメータとして参照アドレスが用いられる [1]．アドレスを用いるケースは，記憶しなければならないアドレスの数が莫大となる．このため，予測精度を高めるためには莫大な記憶領域が必要となる点が欠点となる．それを解決するため，検出するパタンとして参照先アドレスの差分 (ストライド値) を用いる手法が一般的である．

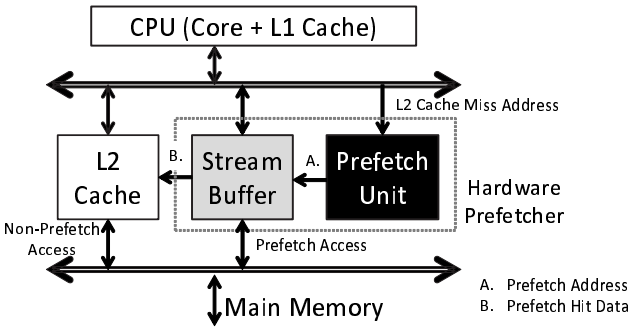

図 1 ハードウェア・プリフェッチ機構の概略図

予測手法は大きくわけて空間的パタンマッチング，時間的パタンマッチング，ならびに確率的予測の 3 手法に大きく分類される．空間的パタンマッチング手法は，参照済みアドレスの傾向に基づき次参照先の予測を行う [4][12]．時間的パタンマッチング手法は，最近の参照先アドレスのパタンに基づき予測を行う [13]．確率的予測手法は，最近の参照先アドレスの遷移グラフに基づき，直近の参照先アドレスから最も次に参照され得るアドレスを決定する [5]．ストライド・プリフェッチ [10] や，デルタ・コルレーション [3] に代表されるプリフェッチ手法の多くは，時間的パタン・マッチングの中に分類される．

また，近年のプリフェッチ研究においては，アクセスパタンをクラスタリングし，複数のアクセス・パタンごとにパタン・マッチングを行う手法も着目されている．代表的な例として，プログラム・カウンタを用いたロード命令ごとの予測手法が挙げられる．ロード命令ごとにアクセス・パタンを管理する事で，他のロード命令のアクセス・パタンが表れる事がなくなり，ノイズの少ない予測が可能となる．登録するアクセス・パタン数に比例して記憶容量が増加するため，実装方法としてはストライド値のみを共有する Global History Buffer (GHB) を用いる手法が一般的である [2][9][8]．

本稿中の検討手法は予測にストライド値を用いる時間的パタン・マッチング手法の一つである．GHB プリフェッチャに代表される既存手法と検討手法は以下の 2 点の差異がある．**1.** 反復的なアクセス・パタンだけでなく，非反復的なアクセスパタンを考慮している．**2.** ストリームの分類にプログラムカウンタなどの特殊な情報は不要である．

3. 波形ベース・プリフェッチ

3.1 着眼点

本稿では，既存研究とは異なるアプローチで高精度なプリフェッチを目指す．まず，既存手法共通の課題について述べる．表 1 に，Lee らによるアクセスパタンの分類 [7] に基づき著者が拡張した分類をまとめる．既存手法の中に

表 1 ストライド・アクセスパタンの分類 (Lee らの分類 [7] に基づく)

パタン	規則的？	反復的？	ストライド列の例	ループ構造コード例
単一ストライド	Y	Y	+4, +4, +4, +4, ...	`S = a[i]; i++;`
複数ストライド	Y	Y	+4, +8, +12, +4, +8, +12, ...	`S = a[i][cons.]; i++;`
直線的増減	Y	N	-4, -6, -8, -10, -12, ...	`S1 = a[i][cons.]; S2 = b[i][cons.]; i++;`
指数的増減	Y	N	+4, +8, +16, +32, +64, ...	`S = A[i]; i = i << 2;`
再帰的参照	N	N	(irregular pattern)	`S = a->next->next;`
ハッシュ値参照	N	N	(irregular pattern)	`S = a[func(i)];i++;`
その他	N	N	(irregular pattern)	

は，各アクセスパタンに対してそれぞれ有効な手法が提案されており，その中でも規則的かつ反復的なアクセスパタンに着目したものは多数報告されている．ストライド・プリフェッチやデルタ・コルレーションがそれにあたる．しかしながら，非反復的なアクセスパタン (直線的増減，指数的増減) を考慮したものはない．仮に，これらのアクセスパタンが混在するアクセスパタンにおいて反復的なアクセスパタンのみに着目したプリフェッチを適用した場合，非反復的なアクセスパタンはノイズとして扱われて予測が行われないか，もしくは，誤った参照アドレス予測を行う．

また，前述の既存手法では予測精度を改善するために大量のアクセス履歴が必要となる．このため，プリフェッチャに搭載されるメモリ・サイズが莫大となる．

本稿では，表 1 にて示している規則的なアクセスパタンをすべて考慮し，かつ少ないメモリ・サイズにて高精度な予測を実現するハードウェア・プリフェッチ手法を検討する．実現するための大きな課題は，特性の異なるストリームの抽出である．一般的に，反復的アクセスパタンと非反復的アクセスパタンは混ざり合って出現する．したがって，あるアクセスパタン履歴から各ストリームを抽出する必要がある．検討手法ではこの抽出を単純なフィルタリング手法を用いることで達成を狙う．

3.2 アクセスパタン・フィルタリング

本検討手法では，反復的アクセスパタンと非反復的アクセスパタンを抽出し，次の参照先アドレスを予測する．具体的な問題設定としては，あるアクセス履歴が与えられたときにそのアクセス履歴から，反復的なストライド・アクセスパタン，非反復的なストライド・アクセスパタン，ならびに規則的ではないストライド値を抽出する．抽出は大きく分けて以下の 2 段階に分けられる．

1. 周期抽出に基づく非規則的ストライド区間の除去

規則的なストライドパタンに分類されるストリームは，共通して周期的に出現する傾向がある．このため，アクセス履歴の中に周期性を検知できれば，そのストライドパタン中に規則的なストライドパタンが存在する事がわかる．一方，周期性を検知できなければ，非規則的なストライドパタンが主な区間であり，プリフェッチによる効果を期待しにくい区間である事がわかる．

周期の獲得には，自己相関を用いる．アクセス履歴に対して様々な周期にて自己相関を算出することで，尤もらしい周期を算出できる．自己相関値が低い値ばかりであれば，周期性は検出できなかったということであり，現在区間は非規則的なストライド値が主な区間であると判定できる．

2. 移動平均に基づくストライド・ストリームの抽出

移動平均は，本来であれば，信号処理などの分野において白色雑音を除去するためのフィルタリング手法の一種である．本来の信号と雑音が混在するサンプルデータに対して，ある間隔でデータのサンプリングを行い平均値を算出する．この平均値算出を連続的に行うことで本来の信号のみが出現するデータを生成でき，信号と雑音を分類できる．

前述のように，各ストライド・ストリームは周期的に出現する傾向がある．先ほど求めた周期の間隔にて各ストライド値の移動平均を算出すると，ストライド・アクセスパタン毎に異なる傾向が表れる．反復的ストライドでは，同じストライド値が連続して出現する．一方，非反復的ストライドでは，直線的に増減，もしくは指数的に増減するストライド列が出現する．これより，移動平均を用いることでストライド・アクセスパタンの分類が可能となる．

3.3 プリフェッチ・フロー

キャッシュ・ミスが発生したとき，そのミス・アドレスと直前のミス・アドレスの差分である最新のストライド値 S_N が得られる．この最新のストライド値はストライド履歴バッファ (History Buffer, HB) に格納される．この時，HB エントリ中もっとも古いストライド値の情報は HB から破棄される．HB のエントリ数を N とすると，HB には N 個のストライド列 $S_1, S_2, \cdots S_{N-1}, S_N$ が格納されていることとなる．

HB 更新の次は，規則性判定，ならびに周期の算出を行う．N 個のストライド列に対して，様々な周期を設定して自己相関 $R_{xx}(k)$ を算出する:

$$R_{xx}(k) = \frac{1}{N}\sum_{i=1}^{N-k} S_i S_{i+k} \quad (1)$$

式 (1) 中の k は仮の周期を示している．この自己相関

表 2 Simualtion Setup

CPU Assumption	
ISA	X86 like
Core Counts	1
L1 (I/D)	32kB, 2ways, 64B
L2 Unified	512kB, 8ways, 64B
Prefetcher Common Assumption	
History Buffer (HB) Entries	128
Prefetch Issue Width	1
Prefetch Degree	1

$R_{xx}(k)$ が最も大きくなる k が，現在のストライド・パタンにおける周期 w であると判断する:

$$\begin{aligned} & w = k \\ Maximize \quad & R_{xx}(k) \\ subjectto \quad & 0 < k < \frac{N}{2} \end{aligned} \tag{2}$$

なお，全ての k であっても自己相関が低い場合，現在のストライド・パタンは規則的ではないと判断し，プリフェッチは行わない．本稿中の評価では，自己相関がある閾値を超えなければプリフェッチは行わないものとする．

上記より求めた周期 w を用いて，反復的ストライド・ストリームと非反復的ストライド・ストリームを抽出する．抽出には，次式に示す移動平均を用いる:

$$y(p,t) = \frac{1}{3}\sum_{j=-1}^{1} S_{(t+j)w+p} \tag{3}$$

式 (3) 中の y は t, p を変数とする移動平均値である．t は時間成分を表しており，p は各ストライド・ストリームの個別番号を意味している．この式では，$tw+p$ を中心，w を間隔として 3 点のストライド値の平均を算出する．この平均値が t の値に応じてどのように推移していくかを観測することにより，ストライド・ストリーム p の特性を判別できる．具体的には，平均値が t の値に依存せず一定の値 s をとり続ける場合，そのストライド・ストリームは周期 w の反復的ストライド・アクセスパタンであり，そのストライド値は s であると判断する．また，平均値が t の増加に応じて単純増減するのであれば，直線的に増減するストライド・アクセスパタンであると判断する．

最後に，最新のストライド値 S_N と周期 w より，次に現れるストライドがどのストライド・ストリームであるのかを求める．その後，該当のストライド・ストリームの特性に基づき，次のストライド値を予測する．最後に，最新のミス・アドレスと予測ストライド値を加算し，プリフェッチするアドレスを求める．

4. 定量的有効性評価

4.1 評価環境

検討手法の有効性を定量的に評価するため，本稿ではトレース・ベースでの評価を実施する．まず，プロセッサ・シミュレータを用いてキャッシュ・アクセス・トレースを生成する．トレースは，プリフェッチを行わなかった場合の参照先アドレスとヒット／ミスが参照毎に保存されている．次に，このトレースをプリフェッチャ・シミュレータに与える．プリフェッチ・シミュレーションより，仮にプリフェッチが行われていた場合に，予測が成功していたか否かを評価する．

表 3 プリフェッチャに要するメモリ量

プリフェッチャ	総メモリ量	主要なメモリ機構
STRIDE	16 B	Most Recent Address
MARKOV	128 kB	HB, Markov-Chain Table
PC/DC/MG	2176 B	HB, Index Table
WAVE	1024 B	HB

評価にあたり，本稿では 4 つの指標にて比較を行う．**Coverage**: 非プリフェッチ時ミス数に対するプリフェッチ発行率．**Accuracy**: 発行プリフェッチ数に対する予測成功率．**SalvageRate**: 非プリフェッチ時ミス数に対する予測成功率．**OverPrefetchRate**: 非プリフェッチ時ミス数に対する予測ミス率．第 4 の指標は，プリフェッチ・ミスによる主記憶アクセスの増加率の評価に用いられる．

シミュレーションにおけるプロセッサ想定を表 2 に示す．本評価では，単純なシングル・コア・プロセッサを想定する．プリフェッチを行うメモリ階層は L2 キャッシュである．L2 キャッシュにおけるキャッシュ・ミスに基づき，予測は行われると想定する．ベンチマーク・プログラムについては，ベンチマーク・プログラム・セット SPEC CPU 2006 より 20 種類のプログラムを選択した．なお，ベンチマーク・セットの中から L2 キャッシュ・ミスがある程度発生するプログラムを選択している．また，プロセッサ・シミュレータにはミシガン大学が提供しているプロセッサ・シミュレータ Gem5 を使用する．

なお，本評価はトレース・ベースのシミュレーションであるため，プリフェッチによる新たなキャッシュ・ミス [*1] を観測することはできない．より現実的な条件における評価は今後の課題である．

また，本評価では検討手法の有効性を示すため既存プリフェッチ手法を利用した場合と比較する．評価対象となる予測手法は，ストライド・プリフェッチ，マルコフ・プリフェッチ，ならびに Diaz らが提案するストリーム間遷移率を考慮したデルタ・コルレーション・プリフェッチ (PC/DC/MG) である [2]．検討手法を含めた 4 種のハードウェア・プリフェッチャをプリフェッチ・シミュレータに実装し，評価を行う．各予測手法において共通する要素

[*1] 例えば，プリフェッチしたデータがあるキャッシュ・ラインに新たに保存され，その時追い出されたデータに対して後ほど読出しが発生した場合に，プリフェッチによる新たなキャッシュ・ミスが発生する

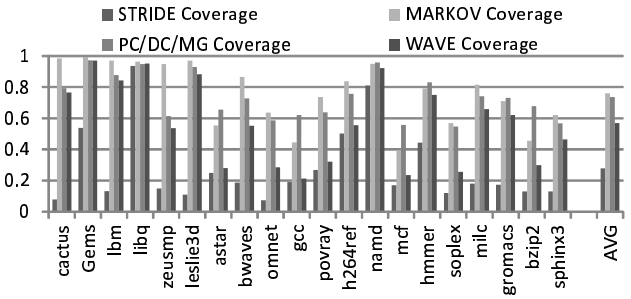

図 2 L2 キャッシュ・ミスに対するプリフェッチ発行率 (Coverage)

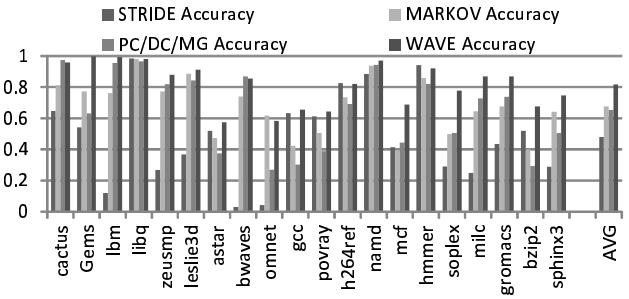

図 3 予測ヒット率 (Accuracy)

を表 2 に示す．これらの値は既存研究における設定値を参考にした．また，各プリフェッチ手法の特性，ならびに個別設定を以下に示す．

- **ストライド・プリフェッチ (STRIDE)**: 過去 2 区間の L2 キャッシュ・ミス時のストライド値が一致している場合，そのストライド値が将来も連続すると仮定してプリフェッチを行う．この設定は Palacharla らの設定 [10] と同じである．
- **マルコフ・プリフェッチ (MARKOV)**: L2 ミスが発生するたびに，過去のストライド履歴を格納している履歴バッファ (HB) 中を解析し，ストライド値を状態として扱ったマルコフチェインを生成する．マルコフチェインに基づき，現在のストライド値 (状態) から最も次に出現しやすいストライド値を求め，プリフェッチを行う．マルコフチェインを物理的に実現するため，HB のエントリ数の 2 乗のメモリサイズが必要となる．
- **PC/DC/MG**: アドレス履歴バッファ (GHB) を用いたデルタ・コルレーション・プリフェッチャの一種．ストリームごとにデルタ・コルレーション・プリフェッチが可能となる．省メモリかつ高精度なデルタ・コルレーション・プリフェッチを実現できる．元来の GHB 利用デルタ・コルレーションではストリーム間の推移は考慮していなかったが，Diaz らの提案によるこのプリフェッチャでは，ストリーム間遷移率をマルコフ・プリフェッチャに似た方法で計算しており，複数のストリームが高頻度に遷移する場合においても高精度に次参照アドレスを予測できる．

なお，本評価では予測精度を主題に置いているが，検討手法はハードウェア・プリフェッチャに要するメモリの観点からみても既存手法に比べて優位性がある．前述の想定における各プリフェッチ手法に要するメモリ量を比較した結果が表 3 である．なお，この表中にはストリーム・バッファは含まれていない．表 3 より，既存手法の中でも省メモリかつ高精度なプリフェッチ手法である PC/DC/MG に比べ，検討手法 (WAVE) は少ないメモリで予測機構を実現できることが示されている．

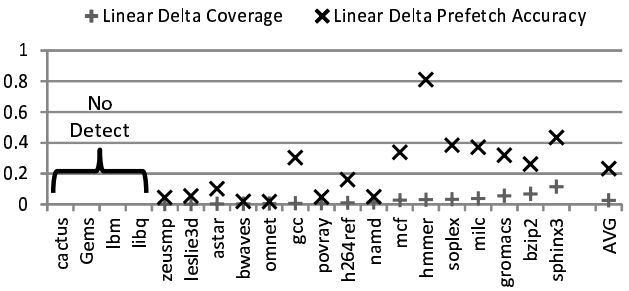

図 4 直線的ストライド増減パタンの出現率ならびに予測ヒット率

4.2 結果ならびに考察

はじめに，各手法における Coverage と Accuracy を比較する．図 2, 図 3 はそれぞれ，ベンチマーク・プログラムごとの実行時の各手法における Coverage と Accuracy を示している．縦軸はそれぞれプログラム実行全体で見た Coverage と Accuracy を示している．横軸は各ベンチマーク・プログラムを意味している．結果より，STRIDE は Coverage, Accuracy が共に低い．MARKOV と PC/DC/MG は Coverage が高く，Accuracy が STRIDE より高い．検討手法 (WAVE) は，比較的 Coverage が低いが Accuracy が高い．特に Accuracy はどのプログラムにおいても各手法に比べて高い値を示している．これより，検討手法は予測が困難な非規則的な区間ではプリフェッチを行わず，予測可能な規則的な区間では積極的にプリフェッチを行い高い予測精度を示していることが表れている．特に他の手法に比べて高い予測精度が表れているのは非反復的なストライド・ストリームの出現確率が高いプログラムであることが図 3 ならびに図 4 より明らかである．図 4 は，各プログラムにおける直線的ストライド値増減パタンの出現率 (Linear Delta Coverage) と，検討手法 (WAVE) にて，そのパタンに対してプリフェッチを実施した際にどれだけ予測が成功したか (Linear Delta Prediction Accuracy) を示している．図 4 中にて出現率ならびに予測成功率が高いプログラムでは，図 3 中にて検討手法は突出した Accuracy 値を示している．これより，非反復的なストライド・ストリームの予測が成功していることが示されている．

次に，SalvageRate 値の評価結果である図 5 に着目する．

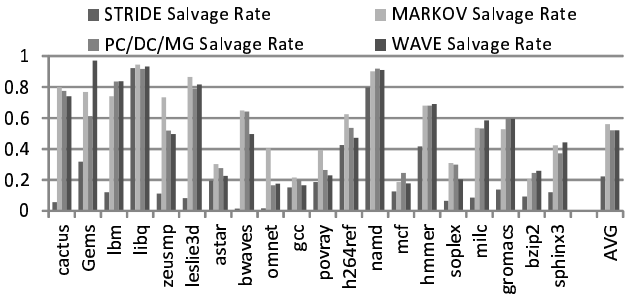

図 5 L2 ミス全体に対するプリフェッチヒット率 (SalvageRate)

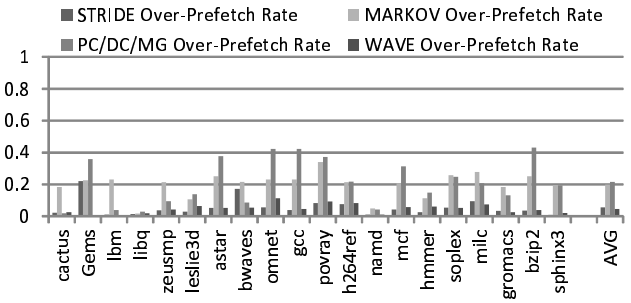

図 6 予測ミスによる主記憶アクセス増加率 (OverPrefetchRate)

図 5 より，検討手法は他のメモリ履歴を必要とする既存手法 MARKOV, PC/DC/MG とほぼ同等に，プリフェッチによりキャッシュ・ミスを削減できることが示されている．特に，非反復的ストライド・パタンの出現頻度の高いケースではより良い SalvageRate 値を期待できる．しかしながら，プログラム *soplex* のように Coverage 値が極端に低くなるケースでは，たとえ Accuracy が高くても SalvageRate 値を高められない．このようなケースにでは，Accuracy を維持しつつ Coverage を高めることが重要であると考えられる．これは今後の課題である．

本検討手法は，正規的ではないアクセス・パタンの区間ではプリフェッチは行わず，少ないプリフェッチ回数で多くのキャッシュ・ミス回数を削減する．これにより，予測ミスによる主記憶への無駄なフェッチ回数を抑制できる．最後に OverPrefetchRate 値の評価結果を図 6 に示す．結果より，Coverage が高く Accuracy が低い既存手法 (MARKOV, PC/DC/MG) に比べ，Coverage が低く Accuracy が高い検討手法 (WAVE) は無駄なフェッチ回数を抑制できている．本評価では比較していないが，無駄なフェッチ回数の少ない検討手法は既存手法に比べ，メモリバンド幅を圧迫することなくプリフェッチが行われ，プログラム実行時間の改善に貢献すると予想される．

5. おわりに

近年の計算機においては，主記憶へのアクセス待ち時間の削減は重要な課題となっており，その課題を解決する手法としてプリフェッチは様々な計算機システムおいて活用されている．既存のストライド・パタンに基づくプリフェッチ手法では，特性の異なるストライド・ストリームが混在するケースでは高精度なアドレス予測は困難であった．本稿では，既存研究とは異なるアプローチ検討し，この問題に取り組んだ．検討手法では，ストライド・パタンにフィルタリング処理を施すことで特性の異なるストライド・ストリームを抽出する．検討手法の有効性評価の結果，既存手法では予測精度の低いケースにおいて高い予測精度を達成できることが明らかとなった．また，提案手法は既存手法に比べて予測に要するメモリ，ミス率を少なくできることも明らかとなった．

本稿における評価では，トレース・ベースの評価を行っている．このため，プリフェッチによる新たなキャッシュ・ミス (Cache Pollution) の影響が表れていない．この影響も考慮した性能評価が第一の今後の課題である．

参考文献

[1] T. Alexander, and G. Kedem: Distributed Prefeth-buffer/Cache Design for High Preformance Memory Systems, *HPCA*, pp.254–263 (1996).
[2] P. Diaz, and M. Cintra: Stream Chaining: Exploiting Multiple Levels of Correlation in Data Prefetching, *ISCA*, pp.81–92 (2009).
[3] Z. Hu, M. Martonosi, and S. Kaxiras: TCP: Tag Correlating Prefetchers, *HPCA*, pp.317–326 (2003).
[4] Y. Ishii, M. Inaba, and K. Hiraki: Access Map Pattern Matching for High Performance Data Cache Prefetch, *JILP*, vol. 13, (2011).
[5] D. Joseph, and D. Grunwald: Prefetching using Markov Predictors, *ISCA*, pp.252–263 (1997).
[6] H. Q. Le, W. J. Starke, J. S. Fields, F. P. O ' Connell, D. Q. Nguyen, B. J. Ronchetti, W. M. Sauer, E. M. Schwarz, and M. T. Vaden: IBM POWER6 Microarchitecture, *IBM J. Res. Dev.*, vol. 51, no. 6, pp.639–662 (2007).
[7] J. Lee, H. Kim, and R. Vuduc: When Prefetching Works, When It Doesn 't, and Why, *ACM TACO*, Vol. 9, No. 1, pp.2- (2012).
[8] K. J. Nesbit, A. S. Dhodaplar, and J. E. Smith: AC/DC: An Adaptive Data Cache Prefetcher, *PACT*, pp.135–145 (2004).
[9] K. J. Nesbit, and J. E. Smith: Data Cache Prefetching Using a Global History Buffer, *IEEE Micro*, Vol. 25, No. 1, pp.90–97 (2005).
[10] S. Palacharla, and R. E. Kessler: Evaluating Stream Buffers as a Secondary Cache Replacement, *ISCA*, pp, 24–33 (1994).
[11] A. Sharif, and H-H. S. Lee: Data Prefetching by Exploiting Global and Local Access Patterns, *JILP*, vol. 13, (2011).
[12] S. Somogyi, and T. F. Wenisch: Spatial Memory Streaming, *ISCA*, pp.252–263 (2006).
[13] S. Srinath, O. Mutlu, H. Kim, and Y. N. Patt: Feedback Directed Prefetching: Improving the Performance and Bandwidth-Efficiency of Hardware Prefetchers, *HPCA*, pp.63–74 (2007).
[14] S. P. Vanderwiel, and D. J. Lilja: Data Prefetch Mechanism, *ACM CUSR*, Vol. 32, No. 2, pp.174–199 (2000).