Vol.2009-ARC-184 No.18
2009/8/5

フィードバックを用いたハイブリッド・プリフェッチ方式

本城剛毅[†1] 石井康雄[†2] 入江英嗣[†1]
稲葉真理[†1] 平木敬[†1]

近年，メモリアクセス性能がプロセッサ性能の大きなオーバーヘッドとなっており，これを解決するためのデータ・プリフェッチが注目されている．データ・プリフェッチとは，ロードされるであろうアドレスを予測し，あらかじめメモリからデータをフェッチしておく手法である．ハイブリッド・プリフェッチは複数のプリフェッチ手法を同時に使う手法で，組み合わせた手法それぞれが持つ特徴を活かすことができる．しかし，メモリからあまりに多くのデータをフェッチしようとすると，資源の枯渇によって性能を低下させてしまう．

本論文では，フィードバック機構付きハイブリッド・プリフェッチ（SHPF）を提案する．SHPF は飽和カウンタと予測されたアドレスを記録しておくテーブルから構成されるフィードバック機構によって，メモリアクセスに使われるアドレスの個数をコントロールすることで資源枯渇の問題を解決する手法である．単純なハイブリッド・プリフェッチと SHPF とを評価した結果，SHPF は IPC の調和平均で約 11.8%改善し，キャッシュミスの平均回数で約 3.86%改善した．

Simple Hybrid Prefetch with Feedback

Goki Honjo,[†1] Yasuo Ishii,[†2] Hidetsugu Irie,[†1]
Mary Inaba[†1] and Kei Hiraki[†1]

In recent years, memory access performance is a large overhead of processor performance, and data prefetch method is attracting. Data prefetch is a method that predicts the addresses to be loaded and fetches the data from memory in advance. Hybrid prefetch is a method that uses multiple prefetch methods at the same time and that makes the most of the methods combined. However, too much fetching of data from memory causes starvation of resources and decreases the total performance.

This paper proposes Simple Hybrid Prefetch with Feedback (SHPF). SHPF is a method that solves the starvation of resources using a feedback mechanism that consists of saturating counters and a table to record all predicted addresses. The result of evaluation of hybrid prefetch and SHPF showed that SHPF improves the IPC by about 11.8% in harmonic mean and improves the cache miss count about 3.86% on average.

1. はじめに

近年，汎用プロセッサの計算力が向上するのに伴い，アプリケーションが必要とするメモリは多くなってきた．そのため，メモリアクセスは大きな問題となっている．しかし，プロセッサ本体に比べてメモリのアクセス性能は向上していないため，メモリのアクセス性能は全体のパフォーマンスの中で大きなオーバーヘッドとなっている．

データ・プリフェッチは，プロセッサから実際にロード命令が発行されるより前に，どのアドレスのデータがロードされるかを予測し，あらかじめメモリからキャッシュにデータをフェッチしておくことで，メモリアクセス時間を隠蔽する手法である．多くのアドレスが予測された場合は，予測されただけ多くのメモリアクセスを行うことができる．

データ・プリフェッチには，大別するとソフトウェア・プリフェッチとハードウェア・プリフェッチがある．ソフトウェア・プリフェッチは，コンパイラが実行バイナリにプリフェッチ命令を埋め込むことで行われるプリフェッチで，ハードウェア・プリフェッチは，キャッシュミスの履歴などを元にアドレスを予測し，行うプリフェッチである．ソフトウェア・プリフェッチは動的な情報を使うことができない，命令オーバーヘッドが発生するなどの問題があるため，本論文では，ハードウェア・プリフェッチを扱う．また，予測に成功した時の利得が大きい「ラストレベルキャッシュへのプリフェッチ」を扱う．

データ・プリフェッチの性能を向上させる手法の一つに，ハイブリッド・プリフェッチがある．ハイブリッド・プリフェッチは，相補的な特徴をもつプリフェッチ手法を複数組み合わせることで，より多くのアドレスを予測する手法である．多様なプリフェッチ手法を組み合わせることで，それぞれの手法が持つ特徴を活かすことができる．しかし，あまりに多くのアドレスにメモリアクセスを行ってしまうと資源を枯渇させてしまい，全体の性能を低下させてしまう．たとえば，必要なデータがキャッシュから追い出されてしまうキャッシュポ

†1 東京大学
The University of Tokyo
†2 日本電気株式会社
NEC Corporation

リューション[15] を発生させたり，メモリアクセス帯域を使い切ってしまうことでロード命令による必要なメモリアクセスが遅れてしまう問題が発生する．これを防止するためには，プリフェッチを行うアドレスの量をコントロールすることが有効である．同じアドレスへのプリフェッチであっても，アプリケーションごとに，また一つのアプリケーションの中でもフェーズによって，有用か不要であるかは変化するため，動的なフィードバックによってコントロールする必要がある．しかし，今までハイブリッド・プリフェッチに対する効果的なフィードバック手法は提案されていない．

本論文では，プリフェッチを行うアドレスの量をコントロールする手法として，Simple Hybrid Prefetch with Feedback (SHPF) を提案する．これは，手法ごとに，その手法がアドレスを正しく予測できたかどうかをフィードバックし，その手法が正確であるか推測することで，その手法が予測したアドレスをメモリアクセスに使うかどうかを動的に判断する方式である．

本論文は，以下，次のように構成される．2 章では，ハイブリッド・プリフェッチの性能評価を行い，3 章では，SHPF を提案する．4 章で SHPF を評価し，5 章では関連するプリフェッチ手法と，フィードバックについての関連研究について述べ，6 章でまとめを述べる．

2. ハイブリッド・プリフェッチ

本論文では，高性能なプリフェッチを行うためのハイブリッド・プリフェッチ[5] について研究を行う．また，単純なストライドしか検出できないもののより多くのアドレスを予測することができる AMPM Prefetch[7]，複雑なパターンを検出することができるものの予測するアドレスの数が少ない C/DC Prefetch[10]，これら 2 つが検出できないアクセスパターンを検出するものとして，AMPM Prefetch の CZone のサイズと C/DC Prefetch の CZone のサイズよりも大きなストライドに特化した Global Stride Prefetch[4] を使用する．これらは，検出できるアクセスパターンが相補的であるため，これら 3 つの手法が予測するアドレスをメモリアクセスに用いることで，性能向上を期待することができる．

本論文で取り上げるハイブリッド・プリフェッチは，複数のプリフェッチ手法が同時に予測を行い，予測されたアドレスすべてを用いてプリフェッチを行う手法である．ハードウェア・プリフェッチと同様にチップ上でハードウェアによって行われる予測に分岐予測がある．分岐予測は，その予測の結果が複数あった場合，その中の一つだけを用いてしか投機的実行を行うことができないが，ハードウェア・プリフェッチは，予測の結果が複数あれば，予測しただけメモリアクセスを行うことができる．

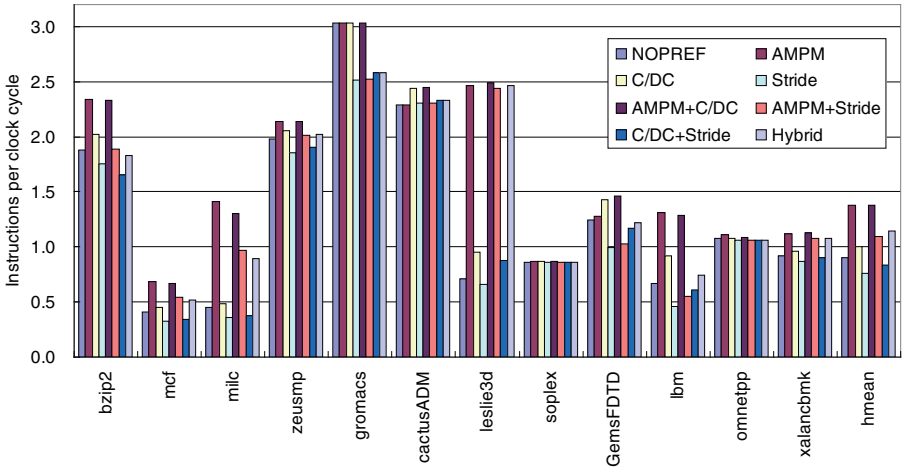

図 1 Hybrid Prefetch の IPC 評価結果
Fig. 1 Result of IPC performance evaluation of hybrid prefetch

図 1 に示したのは AMPM Prefetch，C/DC Prefetch および大きなストライドに特化した Global Stride Prefetch[4] を組み合わせたハイブリッド・プリフェッチの IPC（Instructions Per Cycle count）評価結果である．（細かいパラメーター等は「4. 性能評価」に示されたものと同じである．）性能を比べるため，3 つの手法すべてを組み合わせたハイブリッド・プリフェッチのほかに，何もプリフェッチを行わなかった場合，1 つの手法のみを用いた場合，2 つの手法を組み合わせたハイブリッド・プリフェッチの性能も同時に評価した．

bzip2，mcf，milc，soplex，lbm，omnetpp では AMPM Prefetch が最良の結果を出したが，zeusmp，gromacs，cactusADM，leslie3d，GemsFDTD，xalancbmk では AMPM Prefetch と C/DC Prefetch のハイブリッドが最良の結果を出した．しかし，これらのベンチマークでも，3 つの手法すべてを組み合わせたハイブリッド・プリフェッチは，AMPM Prefetch と C/DC Prefetch のみを組み合わせたハイブリッド・プリフェッチよりも性能が下がる結果となった．

これは，不要なプリフェッチが行われたことで，不要なデータがキャッシュを圧迫するキャッシュ・ポリューションが起きたり，不要なデータをメモリからコピーするためにメモリアクセス帯域を圧迫したりしたことで，有用なプリフェッチが行われるのが遅れてしまったり，

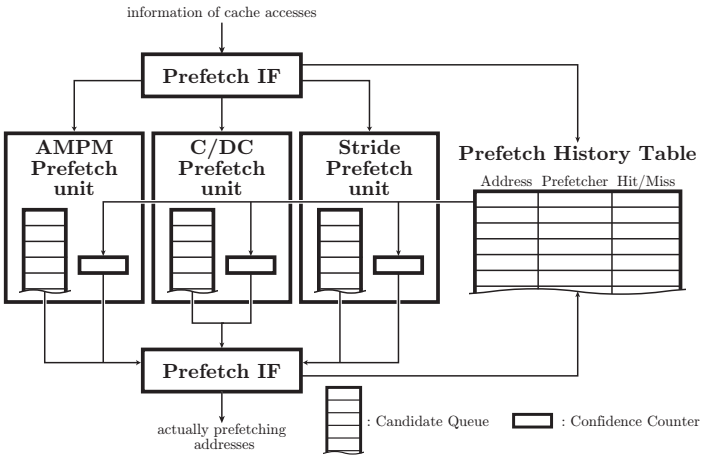

図 2 SHPF のブロックダイアグラム
Fig. 2 Block diagram of SHPF

ロード命令による必要なメモリアクセスが遅れてしまい，ストールが発生してしまったことが原因であると考えられる．よって，フィードバックをかけることで不要なプリフェッチが行われるのを制限すれば，少なくとも図 1 で示された最良の結果までは性能を上げることが期待できる．

3. SHPF（Simple Hybrid Prefetch with Feedback）

前章の評価結果から推測されるとおり，複数のハードウェア・プリフェッチ手法を単純に組み合わせるハイブリッド・プリフェッチは，不要なプリフェッチによってキャッシュ・ポリューションが起きたり，メモリアクセス帯域を使い切ってしまうことで性能を下げてしまうことがある．これを解決するには，不要なメモリアクセスが行われる量を減らすことが必要である．そこで，プリフェッチ手法の予測精度をフィードバックによって推測し，その手法が予測したアドレスが正確かどうかを動的に判断してフィルタリングし，メモリアクセスに使う SHPF（Simple Hybrid Prefetch with Feedback）を提案する．

図 2 は SHPF の仕組みを表したブロックダイアグラムである．

SHPF では，予測精度推測のため，各プリフェッチ機構に信頼度カウンタを持たせる．この信頼度カウンタはビット数の少ない飽和カウンタで，この値が一定の閾値（各プリフェッチ手法ごとに定められている）よりも大きければその手法が予測したアドレスが正確であると判断し，メモリアクセスに使用する．また，各プリフェッチ手法が予測したアドレスをすべて Prefetch History Table に記録しておく．そして，Prefetch History Table 上で実際にロード命令によって発行され，ラストレベルキャッシュへアクセスされたメモリアクセスのアドレスと照合し，信頼度カウンタの値をインクリメント／デクリメントさせる．

Prefetch History Table は，Address フィールド，どのプリフェッチ機構によって予測されたかを記録する Prefetcher フィールド，そのアドレスにロード命令によるアクセスがあったかを記録する Hit/Miss フィールドで構成されている．ハードウェア実装上の容易さにより，Prefetch History Table は FIFO ポリシーによって置き換えられるものとした．また，複数のプリフェッチ機構が同じアドレスを予測した場合は予測された数だけ，Address フィールドは同じだが Prefetcher フィールドが異なる行が挿入されるものとした．SHPF の機構すべてを取り仕切る Prefetch IF は，候補キューにためられたアドレスをすべて Prefetch History Table にコピーする．コピーされた段階では，Hit/Miss フィールドは `false` である．

ロード命令によるメモリアクセスが行われるたびに，Prefetch IF は各プリフェッチ機構にその情報を渡すと同時に Prefetch History Table にも渡す．Prefetch History Table では，テーブル全体を検索し，同じアドレスがそれまでに予測されていなかったかどうかを調べる．予測されていれば，対応する Hit/Miss フィールドを `true` に書き換える．この際，実際にプリフェッチが完了してデータがキャッシュに載る前であっても，Hit/Miss フィールドは `true` に書き換わる．

FIFO ポリシーによって Prefetch History Table からエントリーが追い出されるとき，Hit/Miss フィールドが `true` であれば，対応するプリフェッチ機構の信頼度カウンタの値をインクリメントし，`false` であれば，信頼度カウンタの値をデクリメントする．ハードウェア実装上の容易さにより，信頼度カウンタは「インクリメント／デクリメント」されるものとした．信頼度カウンタの値は，Prefetch IF によってチェックされ，これがプリフェッチ機構ごとに定められた閾値より大きければ，そのプリフェッチ機構の予測は正確であると推測し，そのプリフェッチ機構が予測し候補キューに記録したアドレスはメモリアクセスに用いられる．予測が不正確であると推測されたプリフェッチ機構が予測したアドレスは，メモリアクセスには一切用いられない．

ビット数の少ない飽和カウンタと閾値による判断を行うことで，プリフェッチ機構の予測が正確／不正確であるという判定が変わりやすくなる．これによって，プログラムのフェーズが変化し，得意とするプリフェッチ手法が変化した場合に素早く対応することができる．

閾値は，より低ければそのプリフェッチ機構は正確であると判定されやすくなり，高ければ不正確であると判定されやすくなる．このため，性能を上げやすいプリフェッチ機構の閾値は低めに，性能をあまり上げることができないプリフェッチ機構の閾値は高めに設定する必要がある．また，より多くのアドレスを予測するプリフェッチ機構は大幅に信頼度カウンタの値を動かす事ができるため，プリフェッチ機構の予測が正確／不正確であるという判定を変えやすい．

SHPF では，Prefetch History Table にアドレスが記録されてから FIFO によって追い出されるまでにアクセスされたかどうかを信頼度カウンタにフィードバックすることでプリフェッチ手法の予測精度を推測する手法である．

4. 性能評価

4.1 評価環境

シミュレーションに用いたシミュレータは，x86 プロセッサ用にコンパイルされたバイナリを trace driven にシミュレートできる DPC Kit[⋆1]で，先頭の 4G 命令をスキップし，次の 100M 命令をシミュレーションに用いた．その他の細かいパラメーターは**表 1** に示した通りである．また，シミュレーションに用いたベンチマークは，SPEC CPU2006[⋆2]の中からメモリアクセスが性能を大きく左右する 12 個（bzip2，mcf，milc，zeusmp，gromacs，cactusADM，leslie3d，soplex，GemsFDTD，lbm，omnetpp，xalancbmk）を選んだ．

表 1 シミュレーションのパラメータ
Table 1 Parameters of simulation

cache line	64 bytes
L1 cache	8-way set associative, 32 kbytes, Latency: 1 cycle
L2 cache	16-way set associative, 512 kbytes, Latency: 20 cycles
Main Memory	Latency: 200 cycles, 1 access per 10 cycles
Prefetch History Table	maximum 1024 entries
Confidence Counter	4bits for each prefetch method
Thresholds	3 (AMPM Prefetch), 5 (C/DC Prefetch), 11 (Stride Prefetch)
Stride Prefetch	depth: 8, minimum stride: 4096bytes
C/DC Prefetch	256 Index Table entries, 256 Global History Buffer entries, 4096bytes CZone
AMPM Prefetch	4096 bytes CZone, 32 hot zones, maximum 16 candidates
compile	gcc 4.2 with option “-O3 -fomit-frame-pointer -funroll-all-loops”

4.2 評価結果

図 3 と**図 4** に示したのが，SHPF の IPC と MPKI（Misses Per Kilo Instructions）性能評価の結果である．

全体で見ると，SHPF の性能は単純なハイブリッド・プリフェッチに比べ，IPC の調和平均で約 11.8%，キャッシュミスの平均回数で 3.86%改善した．しかし，milc，leslie3D，xalancbmk ではハイブリッド・プリフェッチよりも SHPF の方が性能が低い．

これらのベンチマークは，図 1 で示されているように，C/DC Prefetch や Stride Prefetch よりも AMPM Prefetch が高い性能を出すベンチマークである．7) に示されている通り，AMPM Prefetch は，より多くのアドレスを予測して，高い coverage を達成しようとすることで，予測精度が下がっても性能を上げる手法である．予測精度のみを基準としてフィードバックする SHPF では，coverage が高くなることで性能を上げ，予測精度が低いプリフェッチ手法はその信頼度カウンタが低くなりがちであるために，そのような手法のみが有効であるベンチマークでは SHPF は性能をあげることができなかったと思われる．

zeusmp と cactusADM に注目すると，SHPF は MPKI がハイブリッド・プリフェッチよりも高くなっているのに，IPC もハイブリッド・プリフェッチより高くなっている．これは，SHPF によって不正確な予測アドレスがフィルタリングされ，不要なメモリアクセスが減ったために，正確なプリフェッチ手法による有用なメモリアクセスやロード命令によるメモリアクセスをより早く行うことができるようになり，プリフェッチによるメモリアクセス時間の隠蔽をより効率的に行えたり，ストールが減ったりしたことが原因であると考えられる．

5. 関連研究

Block Prefetch[3),12)] は，キャッシュミスが発生したキャッシュライン直後の数ラインにメモリアクセスを発行する手法である．また，Stream Prefetch[8),11),14)] は，Sequential Stream

⋆1 http://www.jilp.org/dpc/
⋆2 http://www.spec.org/cpu2006/

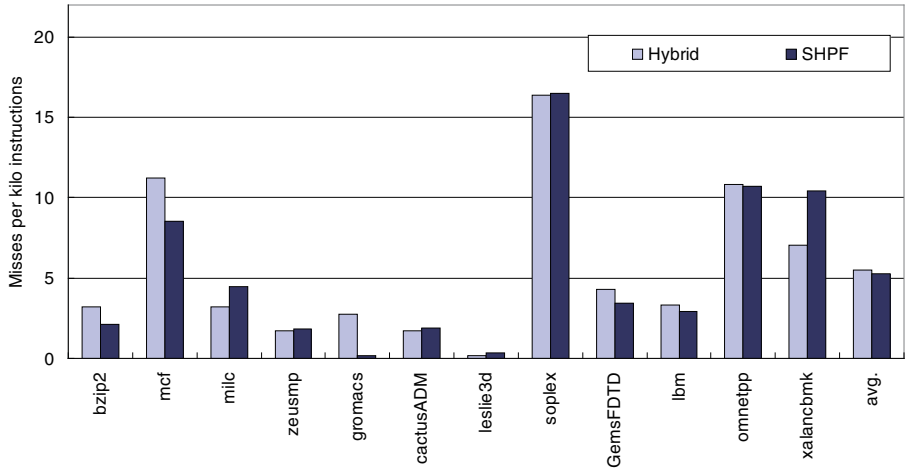

図 3 SHPF の MPKI 評価結果
Fig. 3 Result of MPKI performance evaluation of SHPF

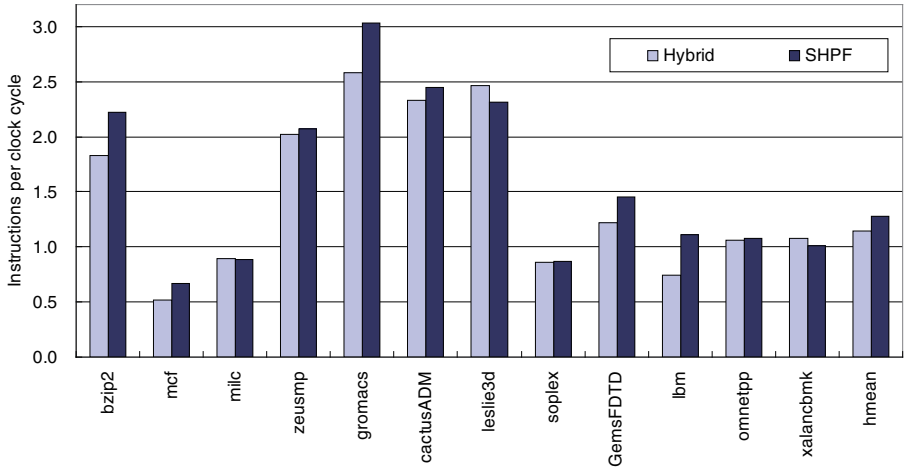

図 4 SHPF の IPC 評価結果
Fig. 4 Result of IPC performance evaluation of SHPF

を検知し，検知できたときのみプリフェッチを行う手法である．

Stride Prefetch[2] は同じプログラムカウンタによるキャッシュミスが起きたときに Stride を計算してプリフェッチを行う手法である．また，DStride[4] は，プログラムカウンタごとに Stride を計算せず，グローバルに Stride の検出を行う．

Block Prefetch と Stride Prefetch は，消費するハードウェア資源に比べて性能を効率的に向上することができるため，実際のプロセッサの中にはこれらを実装しているものもある[1),14)]．

C/DC Prefetch[10] は，アクセスされたアドレスの間隔に注目した手法である．メモリを CZone と呼ばれる一定の大きさに区切り，それぞれの CZone の中で GHB Prefetch[9] を行おうとする．すなわち，アドレスの間隔に規則性を見つけようとする．この手法は，Block Prefetch や Stride Prefetch に比べ複雑なパターンを予測するのが得意であるが，予測するアドレスの数はとても少ない．

AMPM Prefetch[7] は，CZone ごとに作られたアクセスマップ上にキャッシュラインの状態をマッピングし，そのマップを見て予測する手法である．キャッシュラインの状態は「I（Initial）」，「A（Accessed）」，「P（Prefetched）」，「S（Successed）」の 4 種類あり，初期値は「I」である．キャッシュラインにアクセスがあると，そのラインの状態が「I」ならば「A」に，「P」ならば「S」に書き換えられる．そして，キャッシュにアクセスがあるたびに，そのライン（アドレスを a とする）から N 離れたライン（$a+N$）と $2\times$ N 離れたライン（$a+2\times N$）の状態がともに「A」もしくは「S」になっているかどうかを $1 \leq N \leq$ (CZone size/2)，$-1 \geq N \geq -1\times$ (CZone size/2) の範囲で並列に調べ，なっていれば $a-N$ のアドレスにプリフェッチを出す．この手法は，単純なストライドしか認識しない上に，Stride Prefetch よりも conservative にしか予測ができないが，アウト・オブ・オーダー実行やループ展開による予測精度の低下に耐えることができる．また，この手法は他の手法に比べて多くのアドレスを予測することができるが，予測精度は低い．

Feedback Directed Prefetching[13] は，単位時間ごとにプリフェッチの精度・遅さ・ポリューションを動的に計測し，その結果によってプリフェッチの積極度とプリフェッチしたデータのキャッシュへの挿入ポリシーを変化させる．Feedback Directed Prefetching は SHPF に比べて複雑な処理を必要とする．

Adaptive Stream Detection[6] は，単位時間ごとに stream の長さを測ることで stream の平均的な長さを動的に予測し，その長さまでプリフェッチを出すことでプリフェッチを長

く出し過ぎることを防ぐ手法である．

6. おわりに

本論文では，SHPF（Simple Hybrid Prefetch with Feedback）を提案した．SHPF は，Prefetch History Table の Hit/Miss フィールドによってプリフェッチ手法がアドレスを正しく予測できたかを信頼度カウンタにフィードバックする．そして，信頼度カウンタの値を元にプリフェッチ手法の予測精度を動的に推測することで，その手法が予測したアドレスが正確であるかどうかを判断してフィルタリングする手法である．これによって，不要なメモリアクセスを防ぐことができ，キャッシュポリューションやメモリアクセス帯域の圧迫による性能低下を防ぐことができる．

SHPF を DPC Kit によって評価した結果，単純なハイブリッド・プリフェッチに比べて IPC で約 11.8%，キャッシュミス回数で約 3.86%の性能向上を達成した．また SHPF によって不要なメモリアクセスを防ぐことで有用なプリフェッチやロード命令によるメモリアクセスをより早く行うことができるため，ベンチマークによっては，キャッシュミス回数はハイブリッド・プリフェッチよりも高いものの IPC はハイブリッド・プリフェッチよりも高い結果となるものがあった．

また，予測精度ではなく，予測アドレスの量が多いことで性能を向上しているプリフェッチ手法が効果的なベンチマークでは，SHPF は性能を上げることができなかった．これは，SHPF は予測精度のみによってフィードバックする手法であることが原因なので，予測精度だけでなく予測アドレスの量を考慮に含めたフィードバックは今後の課題としたい．

SHPF は，プリフェッチ手法によって予測するアドレスの数が異なること，特定の組み合わせでプリフェッチ手法を選ぶとメモリアクセス帯域を使い切ってしまいやすいことが考慮されていない．これらを考慮し，予測されたアドレスのうちのいくつかをプリフェッチに用いるなど，細かい制御を行う必要がある．

SHPF では，ハードウェア資源について一切考慮しなかったが，ハードウェア・プリフェッチである以上，消費するハードウェア資源は必ず考慮しなければいけない．簡素化した Prefetch History Table や，正確であると推測したプリフェッチ手法が予測したアドレスのスロットリングについては，今後の課題である．

参 考 文 献

1) Doweck, J.: Inside Intel® Core™ Microarchitecture and Smart Memory Access, White Paper, Intel Corporation (2006).
2) Fu, J. W.C., Patel, J.H. and Janssens, B.L.: Stride directed prefetching in scalar processors, *MICRO-25: Proc. of the 25th Annu. Int. Symp. on Microarchitecture*, Los Alamitos, CA, USA, IEEE Computer Society Press, pp.102–110 (1992).
3) Gindele, J.: Buffer Block Prefetching Method, *IBM Technical Disclosure Bulletin*, Vol.20, No.2, pp.696–697 (1977).
4) Hariprakash, G., Achutharaman, R. and Omondi, A.: DStride: data-cache miss-address-based stride prefetching scheme for multimedia processors, *ACSAC 2001: Proc. of the 17th Annu. Comp. Security Applications Conference*, pp.62–70 (2001).
5) Hsu, W.-C. and Smith, J.: A performance study of instruction cache prefetching methods, *IEEE Transactions on Computers*, Vol.47, No.5, pp.497–508 (1998).
6) Hur, I. and Lin, C.: Memory Prefetching Using Adaptive Stream Detection, *MICRO-39: Proc. of the 39th Annu. IEEE/ACM Int. Symp. on Microarchitecture*, IEEE Computer Society, pp.397–408 (2006).
7) Ishii, Y., Inaba, M. and Hiraki, K.: Access map pattern matching for data cache prefetch, *ICS '09: Proc. of the 23rd Int. Conf. on Supercomputing*, New York, NY, USA, ACM, pp.499–500 (2009).
8) Jouppi, N.P.: Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers, *ISCA 1990: Proc. of the 17th Annu. Int. Symp. on Computer Architecture*, New York, NY, USA, ACM, pp.364–373 (1990).
9) Nesbit, K. and Smith, J.: Data cache prefetching using a global history buffer, *Micro, IEEE*, Vol.25, No.1, pp.90–97 (2005).
10) Nesbit, K.J., Dhodapkar, A.S. and Smith, J.E.: AC/DC: An Adaptive Data Cache Prefetcher, *PACT-2004: Proc. of the 13th Int. Conf. on Parallel Architectures and Compilation Techniques*, IEEE Computer Society, pp.135–145 (2004).
11) Palacharla, S. and Kessler, R.E.: Evaluating stream buffers as a secondary cache replacement, *ISCA 1994: Proc. of the 21st Annu. Int. Symp. on Computer architecture*, Los Alamitos, CA, USA, IEEE Computer Society Press, pp.24–33 (1994).
12) Smith, A.: Sequential Program Prefetching in Memory Hierarchies, *Computer*, Vol.11, No.12, pp.7–21 (1978).
13) Srinath, S., Mutlu, O., Kim, H. and Patt, Y.: Feedback Directed Prefetching: Improving the Performance and Bandwidth-Efficiency of Hardware Prefetchers, *HPCA-13: IEEE 13th Int. Symp. on High Performance Computer Architecture*, pp.63–74 (2007).

Vol.2009-ARC-184 No.18
2009/8/5

14) Tendler, J., Dodson, J., Fields, J., Le, H. and Sinharoy, B.: POWER4 system microarchitecture, *IBM Journal of Research and Development*, Vol.46, No.1, pp.5–25 (2002).
15) Zhuang, X. and Lee, H.-H.: A hardware-based cache pollution filtering mechanism for aggressive prefetches, *ICPP-2003: Proc. of the 38th Int. Conf. on Parallel Processing*, pp.286–293 (2003).