Vol.2019-ARC-237 No.8
2019/7/24

タイミング調整を行うプリフェッチャに対して投機ミスが与える影響の解析

出川 祐也[1,a)]　中村 朋生[1,b)]　小泉 透[1,c)]　塩谷 亮太[1,d)]　入江 英嗣[1,e)]　坂井 修一[1,f)]

概要：メモリアクセスにかかる時間を短縮するためにキャッシュは重要な役割をはたしており，キャッシュのヒット率の向上のためにはキャッシュの容量の十分な確保のみならず，プリフェッチをはじめとしたキャッシュ制御が有効である．近年では適切なタイミングでプリフェッチを発行するプリフェッチャが提案されている．しかし，これらのプリフェッチャの評価はトレースベースのシミュレーションによって行われているため，分岐予測ミスなどの投機ミスによる影響が考慮されていない．本論文ではサイクル・アキュレートなプロセッサ・シミュレータを用いてシミュレーションを行うことで，これらのプリフェッチャに対して投機ミスが与える影響を解析する．

An Analysis of an Effect of Speculation Mistake on a Prefetcher Controlling Prefetching Timing

Abstract: Cache memory is important to reduce memory access time, and not only sufficient cache capacity but also cache management such as prefetch is effective to improve cache hit rate. Recently prefetchers issuing prefetches timely have been proposed. However, because these prefetcher are evaluated with a trace-based simulator, an effect of speculation mistake such as branch prediction miss on the prefetchers is not taken into consideration. In this paper, we analyze such effect through simulations with a cycle accurate processor simulator.

1. 序論

メモリアクセスには数百サイクルの時間が必要であり，プロセッサの性能のボトルネックとなっている．このため，主記憶と比べて高速小容量のキャッシュをプロセッサに実装し，メモリアクセスにかかる時間を短縮することが一般的である．また，キャッシュのヒット率を向上させるために，アクセスが生じるラインを予測してそのラインをあらかじめキャッシュに挿入しておくためのハードウェアが実装されることがある．このハードウェアをプリフェッチャといい，プリフェッチャがラインをキャッシュに挿入することをハードウェア・プリフェッチという．これ以降，ハードウェア・プリフェッチのことを単にプリフェッチと呼ぶことにする．

アクセスパターンを捉える方法やプリフェッチするラインを決定する方法には様々な手法があり，様々なプリフェッチャが提案されている．また，評価を行うコンフィグレーションを統一し，同一の条件下で複数のプリフェッチャの性能を比較するチャンピオンシップなどが行われており [1], [2], [3]，ここでの評価結果は最新の研究でも広く使われている．これらのチャンピオンシップではトレース・ベースのシミュレータが用いられている．トレース・ベースのシミュレータでは分岐予測ミスによる誤った命令群の実行やメモリ依存予測ミスによる再スケジューリングといった投機ミスの影響は正確に再現されていない．そこで，本論文では The 2nd Data Prefetching Championship [2] で優勝した，タイミング調整を行うプリフェッチャである Best-Offset (BO) prefetcher [4] を取り上げて，投機ミスによって生じる影響によって実際の性能がトレース・ベースのシミュレータにおける評価からどの程度乖離するかをサ

1 東京大学
The University of Tokyo
a) degawa@mtl.t.u-tokyo.ac.jp
b) tomokin@mtl.t.u-tokyo.ac.jp
c) koizumi@mtl.t.u-tokyo.ac.jp
d) shioya@ci.i.u-tokyo.ac.jp
e) irie@mtl.t.u-tokyo.ac.jp
f) sakai@mtl.t.u-tokyo.ac.jp

ⓒ 2019 Information Processing Society of Japan

イクル・アキュレート・シミュレータを利用することで調査する．

以降，第 2 章ではタイミング調整を行うプリフェッチャについて，第 3 章ではトレース・シミュレータとサイクル・アキュレート・シミュレータについてそれぞれ説明する．第 4 章では投機ミスによって生じる影響を調査した結果とその考察を述べる．第 5 章では本論文をまとめる．

2. Best-Offset Prefetcher

Best-Offset (BO) prefetcher [4] の提案以前に Sandbox prefetcher (SBP) [5] が提案されていた．SBP はあるライン A にアクセスが生じると，ライン $A+D$ をプリフェッチするプリフェッチャである．D はオフセットとよばれる．このようなはたらきは Next-line prefetcher [6] と似ているが，Next-line prefetcher では D の値は 1 で固定されていたのに対し，SBP では D の値はアクセスパターンに基づいて動的に決定される．

BO prefetcher は SBP と同様にオフセットに着目したプリフェッチャである．SBP がプリフェッチを発行するタイミングを考慮していなかったのに対して，BO prefetcher はラインの挿入にかかる時間を考慮してプリフェッチを発行できるように工夫がなされている．BO prefetcher には recent requests (RR) table が備わっている．オフセットが D であるときにライン Y がキャッシュに挿入されると，RR table にアドレス $Y-D$ が記録される．ライン X へのアクセスが発生すると，BO prefetcher はアドレス $X-d$ が RR table に存在するかどうかを調べる．ここで d は仮のオフセットである．RR table で アドレス $X-d$ がヒットすれば，$X-d+D$ はすでにキャッシュに挿入されていることになる．ライン $X-d+D$ をキャッシュに挿入するのに要する時間とライン X をキャッシュに挿入するのに要する時間が等しいと仮定すると，ライン $X-d$ へのアクセスが発生した際にライン X のプリフェッチを発行すれば，ライン X へのアクセスが生じる前にライン X をキャッシュに挿入することができるといえる．したがって，d は適切なオフセットの値の 1 つであるといえる．BO prefetcher は上記のような学習を様々な d について行い，その中から最も適した d をオフセットとして採用する．BO prefethcer はこのようなオフセットの決定方法を用いることで，キャッシュの挿入にかかる時間を考慮に入れてプリフェッチを発行することができる．

3. プロセッサ・シミュレータ

3.1 トレース・シミュレータ

トレース・シミュレータはトレースを入力として動作するシミュレータである．トレースは実機でプログラムを実行することによって作成されるダンプである．例えば DPC3 [3] で利用された ChampSim [7] は，Pin [8] によって

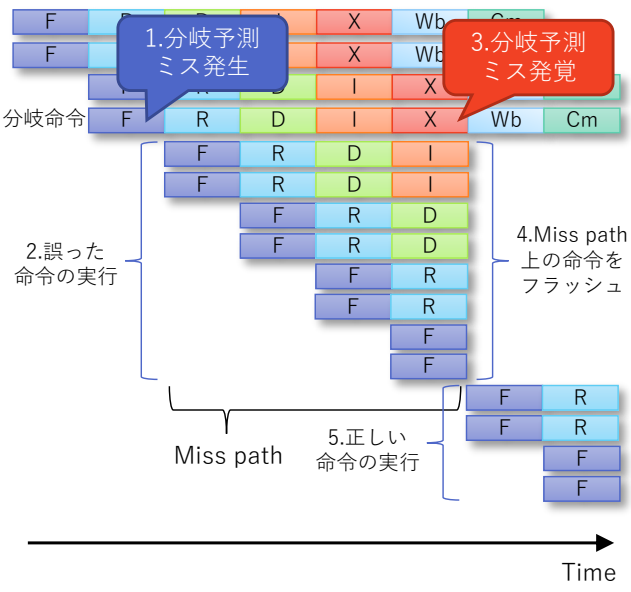

図 1 分岐予測ミス発生時のプロセッサの動作．
Fig. 1 Behavior of a processor when a branch prediction miss occurs.

作成されたトレースを入力として動作する．一般にトレース・シミュレータは後述するサイクル・アキュレート・シミュレータよりも高速に動作するという利点がある．ただし，トレースは実機でプログラムを実行した結果であるため，トレース・シミュレータでは分岐予測やメモリ依存予測といった投機実行を正確に再現することはできない．

3.2 サイクル・アキュレート・シミュレータ

サイクル・アキュレート・シミュレータはプロセッサの動作を 1 サイクルずつ正確に再現するシミュレータである．分岐予測やメモリ依存予測といった，トレース・シミュレータでは正確に再現できない投機実行についてもサイクル・アキュレート・シミュレータでは正確に再現することができる．このため，サイクル・アキュレート・シミュレータでは投機実行のミスがマイクロ・アーキテクチャ・ステートに与える影響を再現することができる．

投機実行のミスがマイクロ・アーキテクチャ・ステートに与える影響について分岐予測を例に説明する．分岐予測ミスが発生した場合には**図 1** のような命令のフラッシュが発生する．本論文では分岐予測ミスの後に誤って実行されるパスのことをミスパスと呼ぶことにする．ミスパス上の命令群はコミットされる前にフラッシュされるため，プログラムの結果に影響を与えることはない．しかし，分岐予測ミスが発生してからミスパス上の命令群がフラッシュされるまでの間に，ミスパス上の命令がプロセッサ内部の状態に影響を与えることがある．例えばミスパス上の命令にロード命令があった場合，要求されたラインが主記憶からキャッシュに挿入されるといったことがありえる．サイ

表 1　シミュレーションしたプロセッサのパラメータ．
Table 1　Simulator parameters.

プロセッサ	
ISA	RV64G
issue width	int:2，fp:2，mem:2
instruction window	int:32，fp:16，mem:16
branch predictor	8KB，8 components TAGE [9]
BTB	2K entries，4-way
LSQ	load:48 entries store:48 entries
page size	4KiB
キャッシュ	
L1 D	32KiB，8-way，64B line 4cycle latency，LRU
L1 I	32KiB，8-way，64B line 4cycle latency，LRU
L2	256KiB，8-way，64B line 16cycle latency，LRU
L3	2MiB，16-way，64B line 44cycle latency，LRU
メインメモリ	97cycle latency
プリフェッチャ	
L1 D	(a) なし (b)BO prefetcher DelayLatency: 70cycle, DQSize: 70, ScoreMax: 3000, RRT: 128 entries
L1 I	Next-line prefetcher
L2	なし
L3	なし

クル・アキュレート・シミュレータではこのようなミスパス上の命令がプロセッサに与える影響を再現することができる．

4.　評価

4.1　評価環境

評価にはサイクル・アキュレート・シミュレータである鬼斬弐 [10] を用いた．鬼斬弐で擬似的にトレース・シミュレータの挙動を再現するために，鬼斬弐にトレース・モードを追加した．トレース・モードでは分岐命令のフェッチ時に分岐予測が正しいかどうかをオラクル的に判断し，分岐予測ミス時にはミスパス上の命令をフェッチしないようにすることで，分岐予測ミスによる命令のフラッシュが発生しないようにしたほか，メモリ依存予測をオラクル的に行うことで，メモリ依存予測ミスによる命令のフラッシュが発生しないようにした．便宜上，トレース・モードに対して無改造の鬼斬弐の実行環境をサイクル・アキュレート・モードと呼ぶことにする．

鬼斬弐を用いて，**表 1** に示すプロセッサをシミュレートした．シミュレーションはサイクル・アキュレート・モードとトレース・モードの両方で行った．BO prefetcher による性能向上の程度を確かめるため，L1D プリフェッチャは (a) なし，(b)BO prefetcher の 2 通りのコンフィグレーションを用いた．ベンチマークとして SPECspeed 2017 [11] に含まれる 20 本のベンチマークを RISC-V 向けにクロス・コンパイルしたバイナリを用いた．最初の 10G 命令をスキップし，それに続く 1G 命令のシミュレーションを行った．

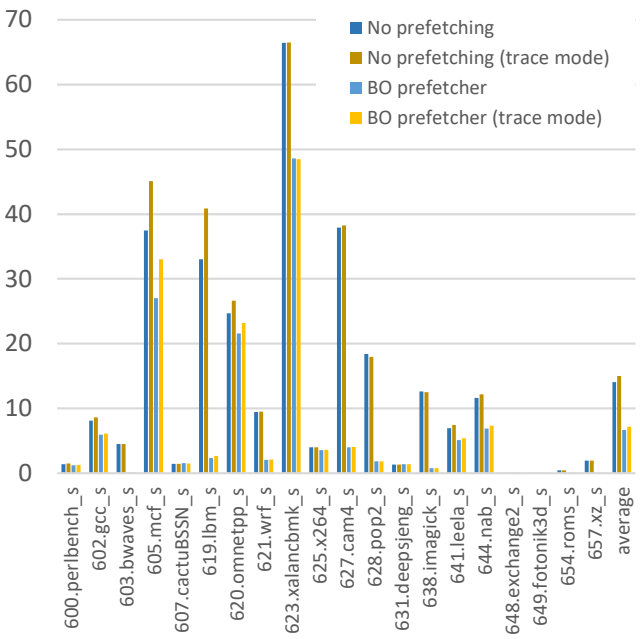

図 2　L1D キャッシュのロードの MPKI.
Fig. 2　Load MPKI on the L1D cache.

4.2　MPKI

図 2 に L1D キャッシュへのロードの Miss per kilo instructions (MPKI) を示す．この MPKI にはミスパス上の命令で発生したロードミスは含まれていない．いくつかのベンチマークにおいてはサイクル・アキュレート・モードで測定したときと比べてトレース・モードで測定したときに MPKI が大きく増加しており，*605.mcf_s* において 20.5%から 22.2%，*619.lbm_s* において 15.4%から 27.1%増加した．*600.perlbench_s*，*602.gcc_s*，*620.omnetpp_s*，*641.leela_s*，*644.nab_s* においても同様の傾向が見られた．つまり，これらのベンチマークにおいては，投機ミスによって生じるミスパス上の命令が MPKI を減少させるはたらきをしたといえる．

その原因として，これらのベンチマークではミスパス上の命令と真のパス上の命令が同じラインにアクセスしていたということが考えられる．そのような場合，ミスパス上に存在するロード命令がキャッシュミスを起こすことでキャッシュにラインが挿入され，ミスパス上の命令群がフラッシュされた後に真のパス上のロード命令がそのラインを参照することでキャッシュヒットを起こす．

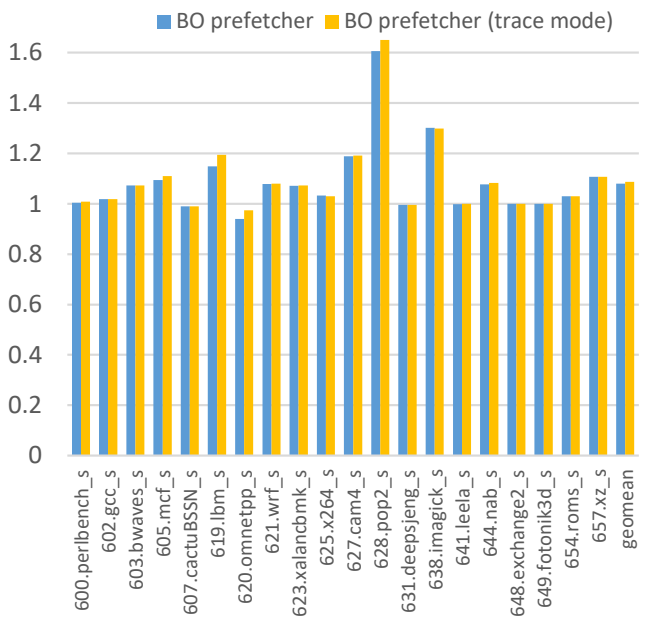

図 3　BO prefetcher を用いたときの相対 IPC.
Fig. 3　BO prefetcher speedup versus no prefetching.

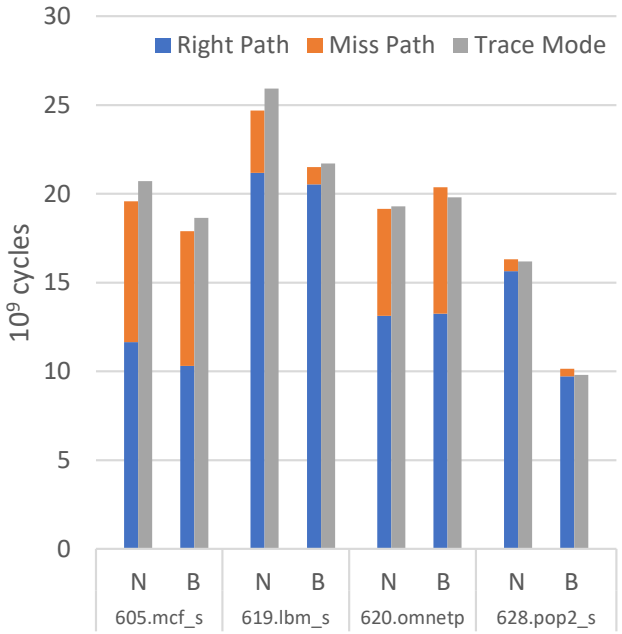

図 4　サイクル数．“N” は “No prefetching”，“B” は “BO prefetcher” を表す．
Fig. 4　A number of cycles. “N” represents “No prefetching” and “B” represents “BO prefetcher”.

4.3　相対 IPC とサイクル数

各コンフィグレーションについて，プリフェッチャがないときに対して BO prefetcher を用いたときの相対 Instruction per cycle (IPC) を図 **3** に示す．いくつかのベンチマークにおいてはサイクル・アキュレート・モードで測定した相対 IPC と比べてトレース・モードで測定した相対 IPC が大きく変化した．具体的には，*605.mcf_s* において 1.5%，*619.lbm_s* において 4.0%，*620.omnetpp_s* において 3.7%，*628.pop2_s* において 2.8%増加している．

サイクル・アキュレート・モードで測定した相対 IPC と比べてトレース・モードで測定した相対 IPC が大きく変化した *605.mcf_s*，*619.lbm_s*，*620.omnetpp_s*，*628.pop2_s* の各ベンチマークにおいて，1G 命令の実行に要したサイクル数を図 **4** に示す．サイクル・アキュレート・モードで測定したサイクル数はミスパスを実行していたサイクル数とそれ以外のサイクル数に分けている．ミスパス上の命令による影響がなければ，サイクル・アキュレート・モードで測定したサイクル数とトレース・モードで測定したサイクル数は等しくなることになる．

サイクル・アキュレート・モードで測定したプリフェッチャなしのときのサイクル数を C_{n}，トレース・モードで測定したプリフェッチャなしのときのサイクル数を T_{n}，サイクル・アキュレート・モードで測定した BO prefetcher を用いたときのサイクル数を C_{b}，トレース・モードで測定した BO prefetcher を用いたときのサイクル数を T_{bt} とする．*605.mcf_s*, *619.lbm_s*, *620.omnetpp_s*, *628.pop2_s* の各ベンチマークにおいては $(T_{\mathrm{n}}-C_{\mathrm{n}})/C_{\mathrm{n}} > (T_{\mathrm{b}}-C_{\mathrm{b}})/C_{\mathrm{b}}$ なる関係が成り立っていた．この関係は変形すると $C_{\mathrm{n}}/C_{\mathrm{b}} < T_{\mathrm{n}}/T_{\mathrm{b}}$ となる．つまり，これらのベンチマークにおける BO prefetcher の性能向上をトレース・シミュレータで測定すると，本来の性能向上よりも過大な性能向上が測定されることになる．

605.mcf_s，*619.lbm_s* においては，BO prefetcher を用いたときのサイクル数はトレース・モードで測定したときよりもサイクル・アキュレート・モードで測定したときのほうが小さかった．このことから，これらのベンチマークにおいては BO prefetcher のはたらきがミスパスによって妨げられているとはいえない．これらのベンチマークでは図 3 で示されるようにトレース・シミュレータで測定した相対 IPC よりもサイクル・アキュレート・シミュレータで測定した相対 IPC が小さかったが，それは BO prefetcher のはたらきがミスパスによって妨げられたからではないということになる．

一方で *620.omnetpp_s*，*628.pop2_s* においては，BO prefetcher を用いたときのサイクル数はトレース・モードで測定したときよりもサイクル・アキュレート・モードで測定したときのほうが大きかった．このことから，これらのベンチマークにおいては BO prefetcher のはたらきがミスパスによって妨げられた可能性があるといえる．

5.　まとめ

本論文では BO prefetcher に対して投機ミスが与える影響をサイクル・アキュレート・シミュレータを用いることで解析した．その結果，ベンチマークによってはプリフェッチャがないときと BO prefetcher を用いたときの両

ⓒ 2019 Information Processing Society of Japan

方で MPKI がトレース・シミュレータで測定されるものよりも小さくなることがわかった．また，いくつかのベンチマークでは BO prefetcher による性能向上幅はトレース・シミュレータで測定されるものよりも小さくなることがわかった．ただし，そのようなベンチマークにおいて投機ミスによる影響が BO prefetcher のはたらきを妨げているといえるかどうかは，ベンチマークによって異なることもわかった．今後の課題はタイミング調整を行う他のプリフェッチャについても同様の解析を行うことである．

参考文献

[1] The 1st JILP Data Prefetching Championship (DPC-1): Data Prefetching Championship, https://www.jilp.org/dpc/.

[2] The 2nd Data Prefetching Championship (DPC2): The 2nd Data Prefetching Championship (DPC2), http://comparch-conf.gatech.edu/dpc2/.

[3] The 3rd Data Prefetching Championship: The 3rd Data Prefetching Championship, https://dpc3.compas.cs.stonybrook.edu/.

[4] Michaud, P.: Best-offset hardware prefetching, *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 469–480 (online), DOI: 10.1109/HPCA.2016.7446087 (2016).

[5] Pugsley, S. H., Chishti, Z., Wilkerson, C., Chuang, P., Scott, R. L., Jaleel, A., Lu, S., Chow, K. and Balasubramonian, R.: Sandbox Prefetching: Safe run-time evaluation of aggressive prefetchers, *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, pp. 626–637 (online), DOI: 10.1109/HPCA.2014.6835971 (2014).

[6] Smith, A. J.: Cache Memories, *ACM Comput. Surv.*, Vol. 14, No. 3, pp. 473–530 (online), DOI: 10.1145/356887.356892 (1982).

[7] ChampSim: ChampSim, https://github.com/ChampSim/ChampSim.

[8] Luk, C.-K., Cohn, R., Muth, R., Patil, H., Klauser, A., Lowney, G., Wallace, S., Reddi, V. J. and Hazelwood, K.: Pin: Building Customized Program Analysis Tools with Dynamic Instrumentation, *Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '05, New York, NY, USA, ACM, pp. 190–200 (online), DOI: 10.1145/1065010.1065034 (2005).

[9] Seznec, A. and Michaud., P.: A case for (partially)-tagged geometric history length predictors, *Journal of Instruction Level Parallelism*, Vol. 8 (online), available from ⟨https://www.jilp.org/vol8/v8paper1.pdf⟩ (2006).

[10] 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，pp. 120–121 (2009).

[11] Standard Performance Evaluation Corporation: SPEC CPU® 2017, https://www.spec.org/cpu2017/.

ⓒ 2019 Information Processing Society of Japan