名古屋大学大学院工学研究科博士前期課程
修士学位論文

容量効率を意識したソース・タグ値に基づくセグメント化による発行キューの電力削減

令和 3 年 3 月
情報・通信工学専攻

森　健一郎

概要

近年のサーバー・アプリケーションや，JavaScript で書かれた Web アプリケーションでは，従来のアプリケーションと比べて命令キャッシュ・ミスが特に多く発生することが知られている．これに対し，命令キャッシュ向けのプリフェッチャが多く研究されており，非常に高いキャッシュ・ヒット率が達成されている．しかし，それらのプリフェッチャでは高い性能をもつものほど大きな追加資源が必要になる．例えば，最先端の命令プリフェッチャである Proactive Instruction Fetch では，L1 命令キャッシュそのものより大きなテーブルを必要とする．また，プリフェッチのミスに対するカバー率こそ高いものの，無駄なプリフェッチを多く実行してしまい，電力を無駄に消費してしまう場合がある．

これに対し，本論文では命令プリフェッチのアプローチではなく，フェッチ・ステージのパイプライン構造の工夫によりフェッチ・スループットを向上させる手法を提案する．本提案手法はプリフェッチャと異なり，複雑な機構や大きなテーブルも必要なく，無駄なメモリアクセスを全く行わない．さらに，性能低下のデメリットなしに命令キャッシュ・ミスによるストールを削減し，フェッチ・スループットを向上させることができる．提案手法をサーバー向けのベンチマークを用いて評価したところ，プリフェッチを行わない場合と比較して最大 25.8%，平均で 13.0%の性能向上を達成した．また，最先端の命令プリフェッチャと比較して平均 4.8%の性能向上が得られることを確認した．

目 次

第 1 章　はじめに

近年のプロセッサは，命令キャッシュ・ミスによる性能低下が問題となっている．これは，現代のアプリケーションの命令ワーキング・セットが大きくなっていることに由来する．このようなアプリケーションの例として，オンライントランザクション処理などのサーバー向けのアプリケーション [?,?,?] や，JavaScript を用いた Web アプリケーション [?,?]，クラウドのアプリケーション [?,?] がある．これらのアプリケーションは，命令フットプリントが膨大であり，L1 命令キャッシュに収まらないため，命令キャッシュ・ミスが多く発生する．命令キャッシュ・ミスが発生すると，プロセッサに実行できる命令を供給できなくなるため，それにより性能が低下する．これは近年の高性能なアウト・オブ・オーダー実行方式のプロセッサにおいても命令キャッシュ・ミスにおけるストールは隠蔽することが難しいため，重要な問題である．

この問題に対し，命令キャッシュ・ミスを減らす手法として，命令プリフェッチャがある．命令プリフェッチャは，ある命令が要求される前に，その要求を先読みして下位レベルのメモリへアクセスを行い，あらかじめその命令をキャッシュへと転送する．これを命令プリフェッチという．命令プリフェッチが成功すると，命令キャッシュ・ミスを回避できるため，性能低下を抑えることができる．

命令プリフェッチャには様々なものが提案されている．例えば，単純なものとしてミスしたラインの次のラインをプリフェッチするネクストライン・プリフェッチャがある．また，分岐予測器が持つ情報を使用する Fetch Directed Instruction Prefetching [?] や，命令キャッシュ・ミスのストリームを記録して命令プリフェッチを行う Temporal Instruction Fetch Streaming [?]，及びそれを改良した非常に高いプリフェッチ効果をもつ Proactive Instruction Fetch (PIF) [1] などが提案されている．

しかし，これらのプリフェッチャは，性能が高いものほど非常に大きなコストが必要になる．なぜなら，有効なプリフェッチを行うためには，複雑なキャッシュ・ミス・パターンを予測するアルゴリズムをハードウェアで実現する必要があるからである．特に PIF は，命令キャッシュ・ミスを 90%以上削減することが可能であるが，必要なストレージのサイズは一般的な L1 命令キャッシュよりも非常に大きい（L1 命令キャッシュが 32KB に対し，200KB 程度)．さらに，このようなプリフェッチャは，プリフェッチのミスに対するカバー率こそ高いものの，無駄なプリフェッチを実行することもあり，その分電力を余分に消費してしまう．

そこで，本論文では命令プリフェッチのアプローチではなく，命令フェッチ部のパイプライン構造を工夫することによって命令フェッチのスループットを向上させる以下のような手法を提案する．

1. 従来の命令フェッチ･パイプラインでは，L1 命令キャッシュがヒットすることを前提としてパイプラインが設計されており，命令がフェッチされるとその命令は直ちに次のステージが送られる．これに対し，本研究では Miss-assuming Pipeline (MAP) と呼ばれる新しいパイプライン構造を提案する．MAP は L1 命令キャッシュのミスを前提としてパイプラインが設計されており，命令キャッシュがヒットしてもミスしても常に一定のレイテンシでフェッチを行う．この動作により，MAP では命令キャッシュ・ミスが発生してもフェッチのスループットを損なうことなく，実行を継続できる．

2. しかし，MAP は従来のパイプラインと比べてパイプライン段数が増加なるため，分岐予測ミス・ペナルティが増加してしまう．MAP の 利点を最大限活用しつつこの欠点に対処するために，本研究ではフェッチのパイプライン構造を，従来のパイプラインと MAP の間で動的に切り替えて使用するアーキテクチャを提案する．同時に，本論文ではこのアーキテクチャによって得られる恩恵を最大化するための最適なパイプラインの切り替えアルゴリズムを提案する．このアルゴリズムを用いると，MAP に

よる分岐予測ミス・ペナルティ増加の影響を最小にすることができ，得られる恩恵を最大化することができる．

提案手法は，プリフェッチャとは異なり無駄なメモリ・アクセスを全く行わない．また，従来のプリフェッチャのような複雑な機構や大きなテーブルも必要なく，非常に低コストで構成することができる．さらに，本提案手法を適用すると，性能低下のデメリットなしに命令キャッシュ・ミスによるストールを削減でき，命令キャッシュ・ミスが多く発生するような状況においては，大きな性能向上が期待できる．

本論文の構成は次の通りである．まず，第?? 章で関連研究を示す．第 2 章では MAP について説明する．その後，第?? 章で従来の構成と MAP を組み合わせたアーキテクチャについて述べ，第?? 章では提案するパイプラインの切り替えアルゴリズムについて述べる．第?? 章では提案手法の評価を行い，最後に第 3 章でまとめる．

第 2章　Miss-assuming Pipeline (MAP)

本章では，提案手法の一つである MAP について説明する．まず，2.1 節で MAP の概要を述べた後，?? 節で MAP の構造を示す．その後，?? 節で MAP の動作を従来のパイプラインと比較しながら示し，?? 節で従来のパイプラインに対するトレードオフについて述べる．

2.1　概要

本研究では，命令キャッシュ・ミスによる性能低下を抑制するために，Miss-assuming Pipeline (MAP) という新しい命令フェッチのパイプライン構造を提案する．従来のプロセッサは，命令キャッシュ・アクセス時に，キャッシュがヒットすることを前提としてパイプラインが設計されている．これに対し，MAP は命令キャッシュ・アクセス時にキャッシュ・ミスが発生することを前提としてパイプラインが設計されている．このことを図 ??を用いて説明する．同図はプロセッサのパイプライン・ステージを図示したものであり，L1, L2, ID，EX, WB はそれぞれ L1 キャッシュ・アクセス，L2 キャッシュ・アクセス，デコード，実行，ライトバック・ステージを表しており，NOP は何もしない（no-operation）で命令を次のステージに渡すステージを表している．なお，説明を簡単にするために，本論文の説明では L1 キャッシュのレイテンシを 1 サイクル，L2 キャッシュのレイテンシを 2 サイクルと仮定する．また，L2 キャッシュは必ずキャッシュ・ヒットすると仮定する．

まず，従来のパイプラインの動作について説明する．従来のパイプラインでは，命令キャッシュがヒットした場合，図 ?? (??) の上側に示すように命令が得られると即座にその命令をデコード・ステージへ送る．一方，命令キャッシュがミスした場合は，L2 キャッシュへのアクセスが必要になるため，図 ?? (??) の下側に示すように L1 キャッシュ・アクセス後

に 2 サイクルかけて L2 キャッシュ・アクセスを行い，得られた命令をデコード・ステージへ送る．

これに対し，MAP では，命令キャッシュがヒットした場合でも，得られた命令を即座にデコード・ステージへ送らず，L2 キャッシュのレイテンシと同じ数の NOP ステージを経由してから命令をデコード・ステージへ送る．これは，命令キャッシュがヒットしてもミスしても，命令をデコード・ステージに送るタイミングを同じにするためである．詳細は **??** 節で述べるが，この動作により MAP は命令キャッシュ・ミスが発生してもパイプラインの乱れが生じず，性能が低下しないという特徴を持つ．命令キャッシュがミスした場合の動作は，従来のパイプラインと同じである．

なお，命令キャッシュのミスを前提としている MAP に対し，従来のパイプラインは命令キャッシュのヒットを前提としているため，本論文では以降，従来のパイプラインのことを MAP と対比して Hit-assuming Pipeline (HAP) と呼ぶ．

第 3章　まとめ

本論文では，パイプライン構造の工夫によって命令キャッシュ・ミスによる性能低下を抑制する手法を提案した．本論文ではまず，MAP と呼ばれる独自のパイプライン構造を提案した．MAP は命令キャッシュ・ミスが発生しても性能が低下しない代わりに，分岐予測ミス・ペナルティが増加するという特徴を持つ．そして，本論文では，MAP と従来の構成を組み合わせ，それらを動的に使い分けることで性能向上を図るアーキテクチャと，このアーキテクチャの性能を最大化することができるパイプラインの切り替えアルゴリズムを提案した．提案手法の利点は，命令プリフェッチャと異なり，非常に小さなコストで構成できる上，無駄なメモリ・アクセスを全く行わないことである．

サーバー向けベンチマークで提案手法の評価を行ったところ，提案手法はプリフェッチなしのモデルと比較して最大 25.8%，平均で 13.0%の性能向上を達成し，最先端の命令プリフェッチャと比較して平均 4.8%の性能向上が得られることを確認した．また，提案手法を適用することで，性能へ大きく影響を与えることなく，命令キャッシュのサイズを小さくできることを確認した．

発表実績

- 松尾玲央馬, 塩谷亮太, 安藤秀樹, “パイプライン構造の動的制御による命令フェッチ・スループットの向上”, 情報処理学会研究報告, Vol.2018-ARC-232, No.3, 2018 年 7 月
- R. Matsuo, R. Shioya and H. Ando: ”Improving Instruction Fetch Throughput with Dynamic Control of Pipeline Structure”, MICRO-51 ACM Student Research Competition, Poster no.10, Fukuoka, Japan, Oct. 2018

受賞歴

- 情報処理学会システム・アーキテクチャ研究会 若手奨励賞, 2018 年 8 月

謝辞

本研究を進めるにあたり，多大なる御指導と御鞭撻を賜わりました名古屋大学大学院工学研究科 情報・通信工学専攻 安藤秀樹教授に心より感謝いたします．また，本研究の遂行を支えてくださいました，名古屋大学大学院工学研究科情報・通信工学専攻安藤研究室の諸氏に深く感謝します．

参 考 文 献

[1] M. Ferdman, C. Kaynak, and B. Falsafi, "Proactive instruction fetch," in *Proceedings of the 44th International Symposium on Microarchitecture*, December 2011, pp. 152–162.