5K-1

レジスタ・キャッシュのマルチスレッド・プロセッサへの適用

西川 卓 † 倉田 成己 †† 塩谷 亮太 ††† 五島 正裕 †† 坂井 修一 ††
† 東京大学工学部 †† 東京大学大学院 情報理工学系研究科 ††† 名古屋大学大学院 工学研究科

1 はじめに

プログラムの実行効率を高める方法に，マルチスレッド･プロセッサを用いたものがある．マルチスレッド･プロセッサによる手法には，複数のプログラムを同時に実行しスループットを向上させる **SMT** や，シングルプログラムの実行性能を上げる**ヘルパースレッディング** [1] や **SoF-MT(Swith-on-Future Event Multithreading)**[2] がある．これらの手法で実行効率は上がるが，マルチスレッド・プロセッサであるため，スレッド数に応じたプロセッサ・コンテキストの保持に多くのレジスタが必要となる．それゆえ必然的にレジスタ・ファイルの面積が大きくなるという問題がある．またレジスタ・ファイルが大きくなることによる消費電力の増大と，それに伴う熱量の増加も深刻な問題である．

この問題を解決する手法の１つに，**レジスタ・キャッシュ**がある．レジスタ・キャッシュはレジスタ・ファイルの複雑さを軽減することで，レジスタ・ファイルの抱える上記の問題点を解決する．しかしマルチスレッド・プロセッサは複数のスレッドを実行するので，レジスタ・キャッシュには複数のスレッドのデータが混在する．それゆえマルチスレッド・プロセッサにレジスタ・キャッシュを適用するとレジスタ・キャッシュの利用効率が低下する．

そこで本稿では各種マルチスレッド･プロセッサに，本研究室で提案されたレジスタ･キャッシュの **NORCS(Non-Latency-Oriented Register Cache System)**[3] を適用し，適用した際に予測されるレジスタ・キャッシュの利用効率への影響について考察する．

2 シングル・プログラムを高速化するマルチスレッド・プロセッサ

シングル・プログラムの性能向上を妨げる原因として，delinquent 命令と呼ばれる頻繁にキャッシュ・ミスや分岐予測ミスを起こす少数の静的な命令がある．この delinquent 命令の重要な性質として，その多くが比較的小さなループ中に存在するということが分かっている [2]．ここでは delinquent 命令に注目した二つのマルチスレッド・プロセッサについて説明する．

2.1 ヘルパースレッディング

ヘルパースレッディングとは，**ヘルパースレッド**と呼ばれるスレッドをメインのスレッドに対して先行実行することによって性能を向上させる手法である．このヘルパースレッドは，通常，シングル・プログラムから delinquent 命令に関わる命令のみを動的に抜き出して作成される．ヘルパースレッディングでは，ヘルパースレッドとして抜き出す命令が少ないとメインのスレッドに対して十分に先行できず，逆に抜き出す命令が多いとヘルパースレッドがメインスレッドの実行を妨げ，性能低下を起こしてしまう．そのためヘルパースレッドの作成では最適な数の命令を抜き出す必要がある．

しかし，小さなループ中など delinquent 命令間の距離が小さい場合には，ヘルパースレッド自体の本数が増えるためヘルパースレッド 1 本あたりの命令数を減らしても全体の命令数は多くなる．すると，ヘルパースレッドは十分に先行できず，またメインスレッドの実行を妨げることになり，性能低下を引き起こす原因となる．

2.2 SoF-MT

SoF-MT では，ループの各イタレーションをスレッドとし，SMT と同様の物理構成を持つプロセッサ上でマルチスレッド実行を行う．スレッド内の delinquent 命令をトリガにスレッドの切り替えを行うことで，delinquent 命令による遅延の隠蔽を図る手法である．

図 1 は for ループ中に delinquent な分岐命令が存在した場合の遅延の隠蔽を示す．図中では $if(item[i])$ が delinquent 命令を含む．今，スレッド t_0 の分岐命令をフェッチし，delinquent と判定されたら次のスレッド t_1 にフェッチ先を切り替えて実行を継続する．このスレッドでも，delinquent 命令をフェッチしたら次のスレッドにフェッチ先を切り替える…という動作を繰り返す．t_0 の分岐先が決定したところでフェッチ先のスレッドを t_0 に戻すと，分岐予測ミスによる遅延が隠蔽される．このようなスレッドの切り替えによって行われる命令は

Applying Multithread processor to Register cache
Suguru Nishikawa† Naruki Kurata†† Ryota Shioya†††
Masahiro Goshima†† Shuichi Sakai††
† Faculty of Engineering, The University of Tokyo
†† Graduate School of Information Science and Technology, The University of Tokyo
††† Graduate School of Engineering, Nagoya University

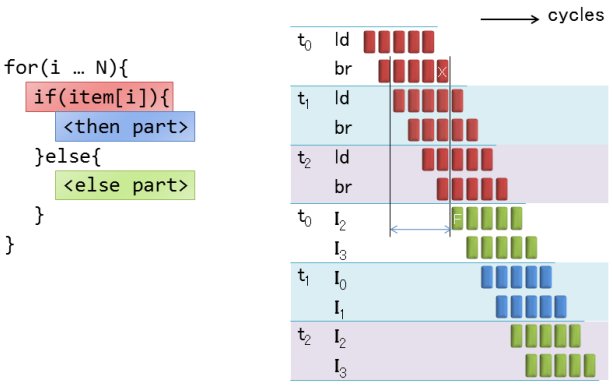

図 1: マルチスレッド実行による分岐命令の遅延の隠蔽

将来起きるイベントなので余分な命令を実行するわけではない．それゆえにヘルパースレッディングの問題は存在しない．

3 レジスタ・キャッシュ

レジスタ・キャッシュとはその名の通りレジスタ・ファイルのキャッシュで，レジスタ・ファイルの複雑さを以下のようにして軽減する．

- **バイパス・ネットワークの簡略化**

  バイパス・ネットワークはレジスタ・キャッシュにのみ必要であり，レジスタ・ファイルには必要ない．このため，バイパス・ネットワークは 1 サイクルでアクセス可能なレジスタ・ファイルのそれと同程度にまで縮小される．

- **レジスタ・ファイルのポート数の削減**

  レジスタ・キャッシュにミスしたオペランドのみがレジスタ・ファイルにアクセスを行うため，レジスタ・ファイルのポート数を削減する．レジスタの面積はポート数の二乗に比例して大きくなるために，レジスタ・ファイルの面積を削減できる．

このように，レジスタ・キャッシュは近年のレジスタ・ファイルの巨大化とそれに伴う消費電力，発生熱量の増大といった問題を同時に解決することができる．

従来のレジスタ・キャッシュではレジスタ・キャッシュ・ミスで引き起こされたパイプラインの乱れによる大幅な IPC の低下がおこる欠点がある．だが当研究室の NORCS はレジスタ・キャッシュ・ミスを前提にした構造であるので，パイプラインの乱れが少なく，前述のように IPC が大幅に低下することはない．

4 考察

マルチスレッド・プロセッサにレジスタ・キャッシュを適用することで，マルチスレッド・プロセッサの抱える面積の問題は軽減される．しかしマルチスレッド・プロセッサは複数のスレッドを実行するため，レジスタ・キャッシュには複数のスレッドのデータが混在する．それゆえにマルチスレッド・プロセッサではスレッド切り替えの増加とともに，レジスタ・キャッシュの利用効率が低下する．ヘルパースレッディングにおいては，ヘルパースレッドが多く生成されることで，スレッド切り替えの頻度が多くなり，レジスタ・キャッシュの利用効率が下がると予測される．その一方で SoF-MT はスレッド切り替えのトリガが少数の delinquent 命令であるため，スレッド切り替えの頻度が低く，レジスタ・キャッシュの利用効率を比較的高く保つことができると予測される．

5 おわりに

本稿では，マルチスレッド・プロセッサの問題を解決する手法として，レジスタ・キャッシュを取り上げた．今後は，本稿で紹介したマルチスレッド・プロセッサにレジスタ・キャッシュを適用したものを，本研究室が開発したサイクル・アキュレートなプロセッサ・シミュレータ「鬼斬弐」[4] 上に実装する．そして IPC やレジスタ・キャッシュ・ヒット率などの評価を行い，上記の考察が適切であったかを検討する．

参考文献

[1] Collins, J.D. and Hong Wang and Tullsen, D.M. and Hughes, C. and Yong-Fong Lee and Lavery, D. and Shen, J.P.: *Speculative precomputation: long-range prefetching of delinquent loads*, *long-range prefetching of delinquent loads,ISCA,*, pp.14-25 (2001)

[2] 塩谷 亮太, 倉田 成己, 五島 正裕, 坂井 修一: *Switch-on-Futrue-Event* マルチスレッディング, 先進的計算基盤システムシンポジウム *SACSIS2010*, pp.157-165

[3] R. Shioya, K. Horio, M. Goshima, S. Sakai: *Register Cache System not for Latency Reduction Purpose*, *MICRO-43*, pp. 301-312 (2010)

[4] 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 SACSIS, pp. 120–121 (2009).