C-012

Control Independenceアーキテクチャに適した分岐合流点の動的予測手法

西川直樹 † 孟林 † 小柳滋 ††
† 立命館大学大学院理工学研究科 †† 立命館大学情報理工学部

1 はじめに

近年のスーパースカラプロセッサは，高い命令レベル並列性を抽出するために，パイプラインが深化傾向にある．分岐予測ミスが発生した場合，パイプラインにおける分岐命令以降の命令をすべてフラッシュする．そのため，分岐予測ミスによるペナルティが，性能向上を阻害する大きな要因となっている．この問題を解決する手法として，近年 Control Independence(CI) アーキテクチャが注目されている．CI アーキテクチャでは，分岐に依存しない命令 (CI 命令) をフラッシュせずに再利用する．これにより，ペナルティの軽減が期待できる．

しかし再利用を実現するには，分岐合流点 (CI 命令ブロックの先頭) の予測が必要となる．CI 命令ブロックの存在を分岐予測時に把握し，キューに保存しなければならないからである．

分岐合流点は Reconvergence Point(RP) と呼ばれる．RP 予測手法には，コンパイラによる静的予測と文献 [1] のような動的予測がある．後者には，前者のように事前にプログラムを分析する必要がないというメリットがある．しかし既存手法は，DMT アーキテクチャ [6] などの投機的マルチスレッド処理にも対応できる汎用性の高い手法であるため，CI にとって有益な RP を考慮していないというデメリットを持つ．

そこで本研究では，より有益な RP を予測として出力し，再利用命令数の増加を図るための CI アーキテクチャに適した動的 RP 予測手法を提案する．

2 RP 予測手法

CI の先行研究 [2][3][4][5] では，各々が独自の RP 予測手法を用いている．TCI[2] でのみ，文献 [1] の手法を基に，他のパラメータの付与などをした動的予測器を用いているが，その他は独自の静的予測を行っている．

本章では，動的予測の既存手法である文献 [1] の概要を説明し問題点を明らかにした後，提案に移る．

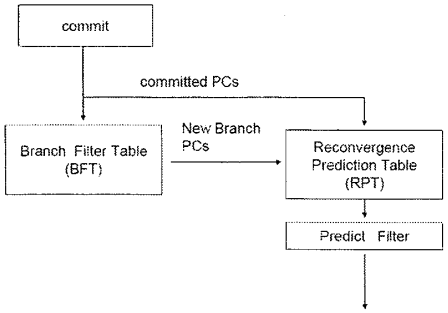

図 1: 動的予測器の構成 [1]

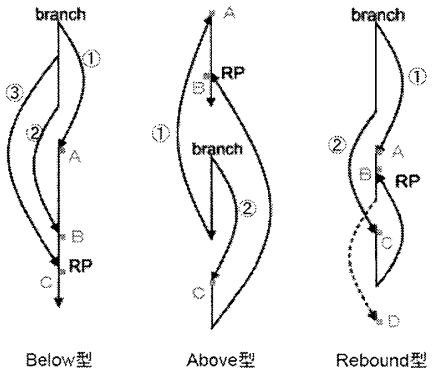

図 2: 合流タイプと RP 候補 [1]

2.1 既存の動的 RP 予測手法概要

図 1 は全体のモジュール構成を示している．予測器は，新規の分岐命令のみを取り出す BFT(Branch Filter Table) と，予測値を格納する RPT(Reconvergence Point Table)，複数の予測候補から 1 つを選別して出力するフィルターで構成されている．予測器の入力は，プロセッサのバックエンド部からコミットされた命令のみである．命令のプログラムカウンタ (PC) 値と命令内容のみが予測の材料となる．RP の候補となるのは，分岐先またはジャンプ先のアドレスのみであるため，これらの情報で十分に予測できる．

Collins らは多数のベンチマークを分析し，図 2 のような 3 つの合流タイプを定義した．Below 型は分岐命令より下で合流するタイプであり，Above 型はその逆である．Rebound 型は Below 型において，上へ飛んで合流するタイプである．さらに，複雑な分岐をしている場合は例外型となる．

各分岐命令は，コミット情報を得ることで動的にア

Dynamic reconvergence prediction for control independence architecture

† Naoki NISHIKAWA
† Lin MENG
†† Shigeru OYANAGI
Graduated School of Science and Engineering, Ritsumeikan University (†)
College of Information Science and Engineering, Ritsumeikan University (††)

クティブな合流タイプへ変更される．各タイプが持つことのできる RP 予測値は 1 つだけである．

図 2 の Below 型を例に説明する．まず，1 のパスを通ると分岐先である A が Below 型の RP 候補となる．続いて 2 のパスを通ると，ジャンプ先である B も Below 型の候補となる．この場合，下のアドレスを優先するという制約がある．そのため，予測値は A から B へ上書きされる．3 のパスを通ると，同様に B から C へと上書きされる．よって最終的に RP は C と予測される．

なかには，複数のタイプが同時にアクティブになる分岐命令が存在する．その場合，フィルターを介して 1 つのタイプが選別される．つまり，1 つの予測値が出力される．

CI アーキテクチャにとって，大きな問題はこの点にある．各タイプが 1 つの RP しか保持しないために，有益な RP が無視される場合が発生するのである．例えば，図 2 の below 型では，C よりも B の方が有益である場合が存在する．つまり 2 から 1 のパスへ変更した際，RP が B だと，より多くの命令を再利用できる．RP 候補は下を優先する方が予測成功率が高い．しかし CI アーキテクチャの場合，より再利用命令数の多い，有益性が高い候補を選別する必要がある．

次節ではこの点について考慮した，CI アーキテクチャに適した提案手法を説明する．

2.2 提案手法

CI アーキテクチャでは図 3 のように，再利用命令を一時的に保存する Re-dispatch & Re-Issue キューを設けている．

DMT は複数のパスを並列処理するアーキテクチャであるため，RP は処理の終点となるが，一方 CI では RP は命令を保存，再発行する始点となる．そのため保存する始点をできるだけ前にすれば，複数の始点がキューに含まれるようになる．再発行する始点は，キューに格納されている複数の始点からの選択が必要であるが，それはフェッチした命令との比較により解決できる．

そのためタイプ分類は必要なく，分岐成立・不成立両方のパスを通る確実性だけあればよい．そこでエントリの主キーである分岐命令に分岐成立ビットを設け，各 RP 候補がどちらのパスを通ったのか記憶する．両方のパスを通った RP 候補のみ，予測として出力する．

前節と同様に図 2 の below 型を例に考える．両方のパスを通る RP 候補は B と C のみであるため，A は予測として出力されない．まず 1 のパスを通ったとすると，予測器は CI 機構に B 以降の保存を要求する．次に，フェッチされた命令を常に RP 候補と比較する．

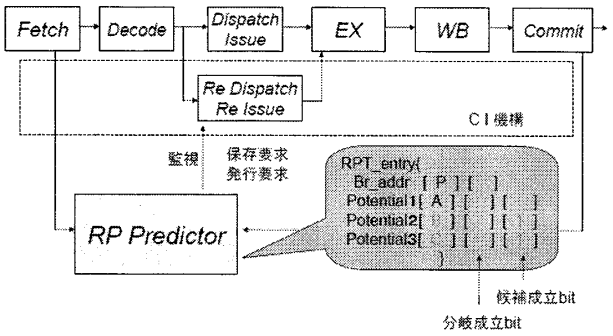

図 3: 提案する予測器

そうすると 2 のパスを通った場合は B 以降を，3 の場合は C 以降の再発行を要求できる．

このように複数の RP 候補とフェッチされた命令を比較し，一致した候補を予測として出力することで，再利用命令数の増加に加えて予測成功率の向上も期待できる．

3 おわりに

本研究では，CI アーキテクチャの性能向上を補助する目的として，アーキテクチャに適した分岐合流点の動的予測手法を提案した．

今後，同じ CI アーキテクチャ上に，既存手法と提案手法を SimpleScalar Tool Set を用いて実装し，比較評価を行う予定である．評価指標に IPC 向上比と，予測成功率を用いて有用性を検証する．

参考文献

[1] J.D.Collins, D.M.Tullsen, Hong Wang, "Control Flow Optimization Via Dynamic Reconvergence Prediction" 37th Int'l Symposium on Microarchitechrure, pages 129-140,2004.

[2] A.S.AL-Zawaki,V.K.Reddy,E.Rotenberg and H.H.Akkary "TCI: Tranceparent Control Independence" In Proc. 35th Int'l Symposium on Computer Architecture, pages 470-481, May 2007.

[3] A.Hilton, and A. Roth "Ginger: Control Independence Using Tag Rewriting" In Proc. 35th Int'l Symposium on Computer Architecture, pages 436-447, May 2007.

[4] A.Gandhi, H. Akkary, and S.Srinivasan "Reducing Branch MispredictionPenalty via Selective Branch Recovery" In Proc. 10th Int'l Symposium on High Performance Computer Architecture, pages 254-265, Feb.2004.

[5] C.Cher and T.Vijaykumar "Skipper: A Microarchitecture for Exploiting Control-Flow Independence" In Proc. 34th Int'l Symposium on Microarchitecture, pages 4-15, Dec.2001.

[6] H.Akkary and M.A.Driscall,"A dynamic multithreading processor" In Proc. 31th Int'l Symposium on Microarchitecture, pages 226-236,1998.