1H-5

CoreSymphonyにおける分岐予測器の分散化

永塚 智之 †　　吉瀬 謙二 ‡

東京工業大学 工学部情報工学科 †　　東京工業大学大学院 情報理工学研究科 ‡

1 はじめに

CoreSymphony アーキテクチャ[1] は，CMP を対象とした，複数のコアを融合することにより 1 つの強力なコアを形成することで IPC の向上を実現する技術である．

このアーキテクチャでは，自コアに割り当てられた命令と，他コアに割り当てられた命令に関する圧縮された情報を記録する特別な命令キャッシュ（ローカル命令キャッシュ）からステアリング済み命令トレースであるフェッチブロック（以下，FB と表記）をフェッチする．

これまでに提案されている CoreSymphony では分岐予測器のテーブルをインターリーブ化することにより，1 サイクルで複数命令の分岐予測を行い，ローカル命令キャッシュから FB をフェッチする．しかし，このハードウェアは現実的なものではなく，CoreSymphony を実現不可能なものとする要因の一つとなっている．

本稿では，この分岐予測器を見直し，2-way の Out-of-Order スーパースカラ相当のハードウェア複雑度のものに変更し，性能を評価する．

2 従来の CoreSymphony の分岐予測

従来の CoreSymphony では，FB を，協調動作中のコア数 ×2 命令に達するか Taken と予測される分岐命令までの命令ブロックを 1 単位とし，このブロックが 2 つ含まれたものと定義している．協調コア数の最大は 4 なので，この命令ブロックの最大長は 8 命令となる．分岐予測では，この命令ブロックの終端と次の命令ブロックの開始アドレスを 1 サイクルで決定する．よって，1 サイクルに最大 8 命令について分岐予測を行うことになる．この定義では Not Taken である分岐命令は FB 中にいくつ含まれても良いため，FB 中に含まれる基本ブロック数に制限はない．また，FB の最大長は 16 命令となるため，4 コア協調時の仮想的な 8-way のスーパースカラでは，理想的には 2 サイクルで 1 つの FB がフェッチされる必要がある．よって，命令フェッチに 3 サイクル以上かけることは好ましくない．

この FB をフェッチするためのフロントエンドを図 1，インターリーブ化された Gshare と BTB を図 2，図 3 に示す．この Gshare と BTB は，それぞれ現在の PC から最大 8 命令分の分岐予測と分岐先アドレス予測を 8 個に分かれたテーブルに別々に割り当てることにより，1 サイクルで行う．

以下，簡単のため，4 コア協調時の動作についてのみ考える．まず，現在の PC から 8 命令分の分岐予測と分岐先アドレス予測を行う．その結果から，最初に現れる Taken と予測される分岐命令のスロット番号（TKN branch slot）とその分岐先アドレス（next addr）を決定する．その後，next addr を PC にセットし，次のサイクルで分岐先アドレスから始まる 8 命令についてもう一度

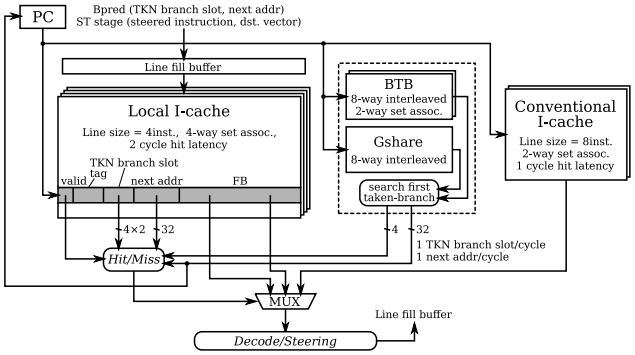

図 1: 現状のフロントエンド．

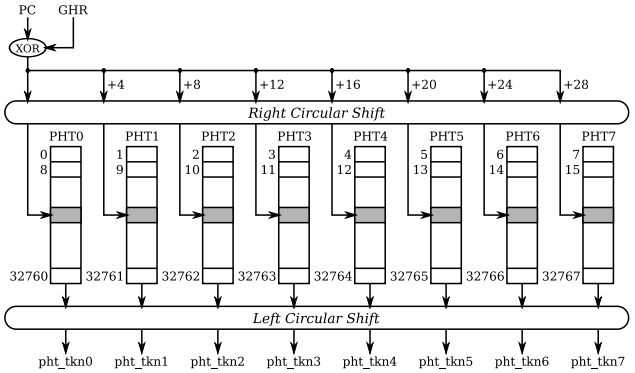

図 2: 8 命令分の分岐予測を行う Gshare．

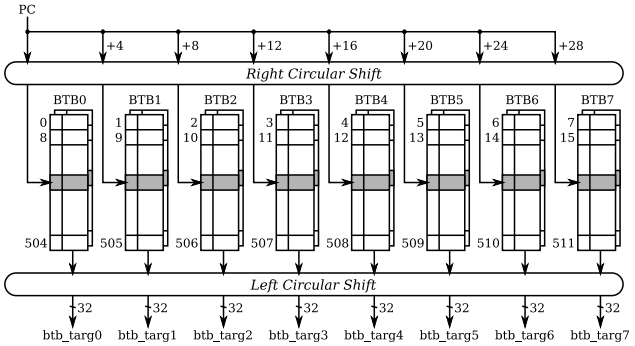

図 3: 8 命令分の分岐先アドレス予測を行う BTB．

予測を行う．そうして得られた 2 つの TKN branch slot と 1 回目の予測で得られた next addr について，ローカル命令キャッシュのタグが一致したラインと比較し，Hit/Miss 判定を行う．よって，命令フェッチには 2 サイクルを要する．ミス時にはその結果を用いて従来型命令キャッシュから命令をフェッチし，予測結果と同じ FB を生成し，ローカル命令キャッシュに格納する．

本来の Gshare の動作であれば，2 つ目以降の分岐命令の予測は GHR を投機的に更新して行う必要があるが，先行する命令の分岐予測結果が得られてから，連続して 8 回テーブルを参照するのはクリティカルパスの問題から現実的ではない．よって，8 命令分の予測は同一の GHR を使用することで並列に行う．だが，実際には正

Decentralization of Branch Predictor on CoreSymphony Architecture.

Tomoyuki NAGATSUKA†, Kenji KISE‡

†Depart. of Computer Science, Tokyo Institute of Technology

‡Graduate School of Information Science and Engineering, Tokyo Institute of Technology

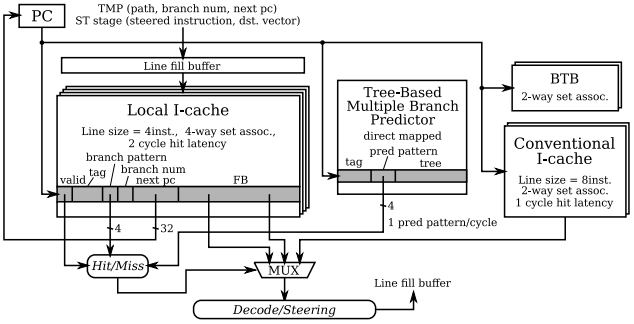

図 4: 変更後のフロントエンド.

しい GHR を用いて予測される分岐命令は 8 命令中，最初の分岐命令のみであるため，2 つ目以降の分岐命令の分岐予測精度は悪くなる．

全ての命令の分岐予測は並列で行われるが，Gshare からは最大 1 × 8bit，BTB からは最大 32 × 8bit が分岐予測結果として出力される．これは全ての命令に関して分岐予測を行った結果であり，現実的とは言えない．

3 提案する分岐予測

1 回のテーブル参照で複数の分岐予測を行う技術として複数分岐予測器が存在する．現実的な分岐予測器を実現するため，その一つである，Tree-Based Multiple Branch Predictor[2]（以下，TMP と表記）を用いることにする．それに伴いローカル命令キャッシュおよび FB に関する仕様を変更する．変更後のフロントエンドを図 4 に示す．

FB は命令トレースであるが，命令トレースの一般的な定義では，トレースは，最大長に達するか規定の基本ブロック数を含んだ位置で終端する [4]．しかし，この定義ではトレースの終端位置が曖昧になりやすく，同じ命令を含むトレースのパターンが増加するため，トレースキャッシュのハードウェア効率が悪くなり，ヒット率が低下する．よって，文献 [3] を参考に FB の定義を，4 命令に達するか Taken/Not Taken 問わず分岐命令に出会う位置までの命令ブロックを 1 単位とし，このブロックが協調動作中のコア数分含まれたものを FB とするように変更する．

この変更により，FB の最大長は協調コア数が 1–4 コア全ての場合において以前のものと変わらず，FB 中に存在する分岐命令数の最大は，4 コア協調時の場合の 4 つとなる．また，FB の終端は，分岐命令であるか，または，ある分岐命令の分岐先アドレスから 4 命令目の位置であることが保証され，トレースの境界が明瞭になることにより，ローカル命令キャッシュの利用効率が改善される．

TMP による複数分岐予測では，次にフェッチすべき FB に現れる分岐命令の分岐方向のパターンを 0，1 で表したもの（pred pattern）が予測結果として出力される．FB の新しい定義では 1 つの FB の中に最大 4 つまで分岐命令が含まれるので，TMP は 4 つ先までの分岐命令の予測を 1 サイクルで行えるようにする．ローカル命令キャッシュのラインには，その FB の辿る 4bit の分岐パターン（branch pattern）を含め，それと予測結果を比較し，Hit/Miss 判定を行うように変更する．

次に，TMP で用いる GHR の投機的な更新のために，ライン中に，FB 中に含まれる分岐命令数（branch num）を含めることにする．ある FB がフェッチされたとき，branch num だけ GHR をシフトし，branch pattern を

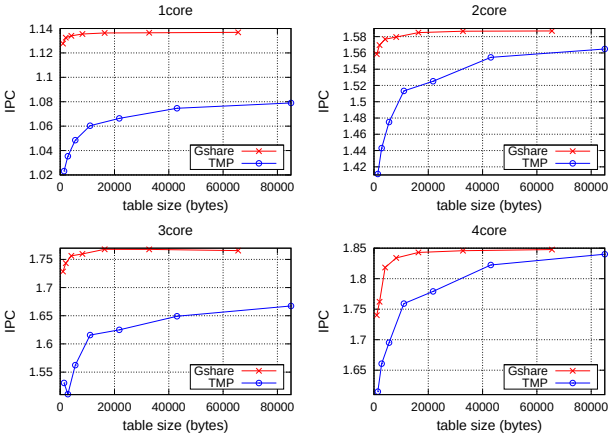

図 5: 評価結果

挿入する．

最後に，現状のアーキテクチャにおいて最も実装上問題のある BTB について再考する．新しく提案する方法では FB のフェッチにおいて各分岐命令の分岐先アドレスは考慮しない．よって，FB 中の全ての分岐命令の飛び先は必要としないため，次回の命令フェッチのために，最後に出現する分岐命令の分岐先アドレスが分かれば十分である．しかし，これは BTB で予測する必要はなく，ローカル命令キャッシュのライン中に含めることで実現できるため，ローカル命令キャッシュから FB をフェッチする際には BTB は必要としない．この変更により，BTB はローカル命令キャッシュにミスしたときにのみ使用されるため，以前より小さくすることが可能である．従来型命令キャッシュのフェッチ幅は 2 命令なので，BTB は 1 サイクルに 2 命令の分岐先アドレス予測ができれば十分である．

4 性能評価

シミュレータによる IPC の評価結果を図 5 に示す．評価には SPEC2006 INT から 6 種類（bzip2，gcc，mcf，gobmk，hmmer，h264ref）を用いた．シミュレーションでは 1G 命令スキップ後の 100M 命令について測定を行った．評価の結果，IPC の低下を 4 コア協調時に 8KB 程度のテーブルサイズで 4.1%に抑えることができた．

提案した分岐予測では Return Address Stack（RAS）による復帰アドレスの予測は含まれていないため，まだ改善の余地があると考えられる．

5 まとめ

本稿では，実現するのが困難であった CoreSymphony の分岐予測器を見直し，現実的なものにしたときの性能の低下を評価した．今後の課題としては，RAS を含めた分岐予測器の構成が挙げられる．

参考文献

[1] 若杉祐太, 他．協調可能スーパースカラ CoreSymphony. 情報処理学会論文誌 ACS, Vol.3, No.3, 2010.

[2] Ryan Rakvic, et al. Completion Time Multiple Branch Prediction for Enhancing Trace Cache Performance. In Proc. ISCA-27, 2000.

[3] Bryan Black, et al. The Block-based Trace Cache. In Proc. ISCA-26, 1999.

[4] Eric Rotenberg, et al. Trace Cache: a Low Latency Approach to High Bandwidth Instruction Fetching. In Proc. MICRO-29, 1996.