社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－EMB－2 (3)
2006／9／27

現実的な構成を保ちながら大幅な予測ミスの削減を図るgshare+分岐予測

吉瀬謙二†

パイプライン段数と命令発行幅の増大により，プロセッサ性能に与える分岐予測の重要性が増している．予測精度を向上させるために，過去数十年の間に様々な分岐予測が提案されている．本稿では，極端な偏りをもつ分岐を検出するBiasテーブルの改良方式を検討し，テーブルの利用効率を向上させるDual-indexed Biasテーブルと呼ばれる方式を提案する．この方式を用いた新しい構成の分岐予測として，現実的な構成を保ちながら大幅な予測ミスの削減を図るgshare+分岐予測を提案する．SpecFP, SpecINT, マルチメディア，サーバの領域を含むベンチマークプログラムを用いた評価からgshare+分岐予測の優れた予測精度を確認する．

The gshare+ branch prediction which improves prediction accuracy maintaining a realistic hardware organization

KENJI KISE†

Accurate branch prediction has come to play an important role for both modern high performance and embedded microprocessors. In order to improve its prediction accuracy, many branch predictions have been investigated. In this paper, we introduce the dual-indexed bias table which enhances the simple bias table in order to detect extremely biased branches. Then, the gshare+ branch prediction using the dual-indexed bias table is proposed as an enhanced version of gshare branch prediction. The gshare+ branch prediction improves its accuracy maintaining a realistic hardware organization. Our experimental results using benchmarks from SpecFP, SpecINT, multi-media and server area show that gshare+ gives better prediction accuracy than gshare and bimode.

1. はじめに

命令発行幅の増大と命令パイプライン長の増大により，プロセッサ性能に与える分岐予測の重要性が増している．また，分岐予測ミスは不必要な命令の処理を発生させるため，消費電力を抑えるためにも分岐予測の精度向上が重要となる．Sengらは，分岐予測ミスを完全に排除することで整数系のベンチマークにおける15%の消費電力を削減できることを報告[4]している．

本稿では，まず，極端な偏りのある分岐[5]を検出して，これらに関して高い精度で予測をおこなうBiasテーブルの改良方式を検討する．Biasテーブルを用いて予測可能となる範囲と予測精度を明らかにし，その後，Biasテーブルの多くのエントリが利用されることがなく利用効率が悪いことを示す．これらの検討に基づいて，テーブルの利用効率を向上されるDual-indexed Biasテーブルと呼ばれる方式を提案し，これにより，予測可能となる範囲を広げることを試みる．

次に，Dual-indexed Biasテーブルを用いた新しい構成の分岐予測として，gshare+分岐予測を提案する．SpecFP, SpecINT, マルチメディア，サーバの領域を含むベンチマークプログラムを用いて予測精度を評価し，既存手法のgshare[3], bimode[2]と比較する．これにより，gshare+の高い予測精度を明らかにする．

本稿では，成立あるいは不成立という1ビットの結果を生成する条件分岐命令のみを予測の対象とし，レジスタの内容に応じて分岐先アドレスを決めるような分岐の予測は扱わない．分岐成立のことをTaken，不成立のことをUntakenと記述することがある．

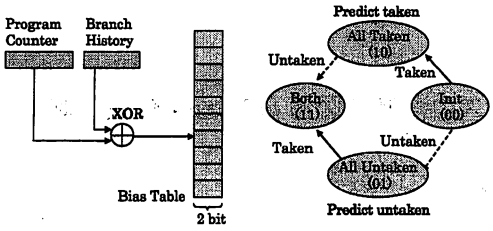
2. 極端な偏りを用いるBiasテーブルの改良

2.1 Biasテーブルの能力

本節では，Biasテーブルを用いて，極端な偏りのある分岐と検出される範囲と予測精度を明らかにする．評価パラメータとして変化させるハードウェア量を除くと，これらBiasテーブルの能力は予測のための状態遷移の方式とインデックスの生成方式に依存する．

まず，予測のための状態遷移の方式を検討する．状態遷移は図1(b)に示したものを考える．ここでは，

† 東京工業大学 大学院情報理工学研究科
Graduate School of Information Science and Engineering, Tokyo Institute of Technology



(a) Bias table indexed by PC and history. (b) The states to detect extremely biased branches.

図 1 Bias テーブルの構成 (a) と予測のための状態遷移 (b)

Init, All Untaken, All Untaken, Both という 4 つの状態を利用する．我々は，プログラムの開始からの全ての実行において必ず成立する分岐，あるいは必ず不成立する分岐，すなわち，プログラムの開始から予測の時点までの全て挙動が同じ分岐のことを**極端な偏りのある分岐**と呼んでいる．それぞれの分岐の状態は図 1(b) の Init から始まり，分岐成立のみが続く場合には All Taken の状態に留まることになる．このため，All Taken に留まっている場合には，極端な偏りのある成立分岐として Taken と予測する．同様に，分岐不成立のみが続く場合には All Untaken の状態に留まることになる．このため，All Untaken に留まっている場合には，極端な偏りのある不成立分岐として Untaken と予測する．Init の時には Untaken と予測する☆．

次に，インデックスの生成方式を検討する．コンパイラが生成する個別の分岐命令によって挙動が大きくことなるので，分岐命令のアドレス (プログラムカウンタ) を利用してインデックスを生成することが基本となる．これに加えて，グローバルな分岐履歴を利用することで極端な偏りをもつ分岐と検出する範囲を拡大できること[1] がわかっている．よって，図 1(a) の様に，プログラムカウンタとグローバル分岐履歴を用いてインデックスを生成することとする．Bias テーブルは 2 ビットのデータのテーブルなので，1KB のハードウェア量の時に，12 ビットのインデックスを必要とする．この時に 1 ビットの分岐履歴を利用し，ハードウェア量の増加に伴って分岐履歴のビット数が増加するようにインデックスを決める．具体的には，C 言語の演算子を利用して，次式によりインデックスを生成する．

```
index = (pc ^ (bhr << 11)) % TABLE_SIZE;
```

ただし，分岐履歴 bhr は左シフトの後に最近の分岐結果を LSB に保存する構成とし，プログラムカウンタ pc のワードオフセットは除去されているものとする．

図 2 に，ハードウェア量を変化させて測定した Bias テーブルの能力をまとめる．ベンチマークは，2004 年に開催された分岐予測のコンテスト Championship

☆ 利用した分岐トレースにおいて僅かに分岐不成立の回数が多かったため，Init 状態の時に不成立と予測することとした．成立と予測してもほとんど結果に変化はない．

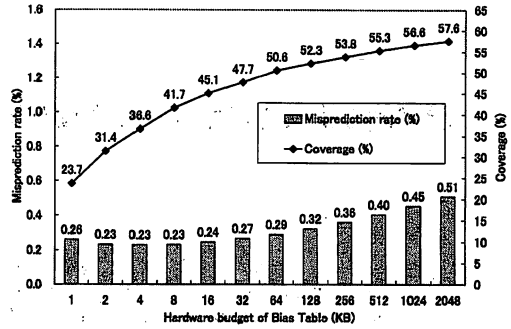


図 2 ハードウェア量を変化させて測定した Bias テーブルのミス率 (Misprediction rate) と適用範囲 (coverage)

Branch Prediction のトレースを利用し，トレース駆動のシミュレータを用いてデータを得る．横軸は Bias テーブルのハードウェア量で，1〜2048KB まで変化させた．Coverage は Bias テーブルを用いて予測をおこなうことができる割合で，図 1(b) の場合には，インデックスで選択された 2 ビットの値が Init, All Taken, All Untaken の状態にある割合となる．本稿では，coverage のことを**適用範囲**と呼ぶことにする．Misprediction rate は，予測をおこなった回数と予測ミスの回数との比率として計算するミス率である．

図 2 の結果から，ハードウェア量の増加に伴って適用範囲が増加することがわかる．4KB の適用範囲は 36.6%であるが，64KB まで大きくすることで全体の半分を超える 50.6%の分岐の予測が可能となる．一方，5%程度の一般的な分岐予測のミス率と比較して，Bias テーブルを用いて予測する時のミス率は 0.23%〜0.51%という低い値を維持できる．ハードウェア量を大きくするに従ってミス率が増加する傾向にあるが，特に，2〜8KB の現実的なハードウェア量の構成において低いミス率を達成する．

2.2 Bias テーブルの利用効率

競合が発生することで Bias テーブルの適用範囲が減少する．例えば，All Taken の状態と All Untaken の状態をうまく分離することができず 1 つのエントリを共有すると，Both の状態となり，Bias テーブルによる予測ができなくなる．同様に，Both の状態とその他の状態との競合によっても，Bias テーブルの適用範囲が減少する．このような競合を削減するためには Bias テーブルのエントリ数を増やせばよいが，ハードウェア量や回路遅延の制約などから容易ではない．

そこで，本節では，限られたハードウェアを有効に利用することを目指して，Bias テーブルの多くのエントリが利用されていない点を指摘するとともに Bias テーブルの改良の方向性を導き出す．

図 3 は，4KB の Bias テーブルにおいて，ベンチマークの実行が終わった時点でテーブルの全てのエントリの状態を数え上げ，全エントリに対するそれぞれ

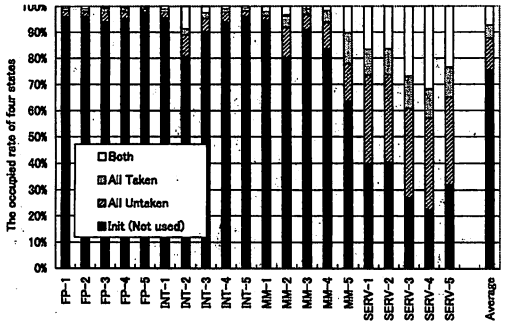


図 3 4KB の Bias テーブルにおけるプログラム終了時の各状態の割合

の状態の割合を示したグラフである．20 個のベンチマークに加えて，右端に平均値を示した．

FP-1 から FP-5 として示した浮動小数点演算のベンチマークでは，プログラムに含まれる分岐命令の種類が少ないので，90%を超える多くのエントリが参照されることなく初期値 Init の状態に留まっている．図 1(b) に示した様に，1 回でも参照されたエントリは，分岐結果に応じて Init から推移するため，Init に留まるエントリは 1 度も参照されていないことに注意する．SERV-1 から SERV-5 として示したサーバ系のベンチマークでは，分岐命令の種類が多いために All Untaken の割合が 30%程度と高くなっているが，全体の平均をみると，一度も参照されることのないエントリが 75%に及ぶことがわかる．このことから，利用されていないエントリを有効に活用することで，Bias テーブルの能力を改善できる余地があることがわかる．

2.3 Dual-indexed Bias テーブルの提案

本節では，洗練された構成を持つ Dual-indexed Bias テーブルを提案する．

図 4(a) に示す様に，まず，図 1(b) に示した 4 つの状態において，Init と All Untaken という 2 つの状態を 1 つにまとめることを考える．この変更により，従来の 4 状態を利用する場合には許さなかった All Untaken から All Taken への推移を 1 回だけ許すことになるが，これが適用範囲と予測精度に与える影響は小さい*．

続いて，削減した 1 つの状態を有効に利用することを考える．前節の評価から，Bias テーブルのエントリの多く (プログラム終了時の平均で 75%) は Init 状態に留まることがわかっている．加えて，Init と All Untaken をまとめた状態を考えると，このエントリに留まる割合は図 3 の Init の割合と All Untaken の割合を加えたものとなり，プログラム終了時の平均で

* 例えば，4KB の Bias テーブルの比較では，状態数を 4 から 3 に変更することで，予測ミス率が 0.04 ポイント上昇して 0.27%に悪化する程度である．このとき，適用範囲は 0.3 ポイント改善される．

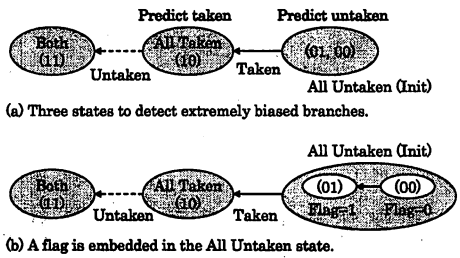


図 4 3 状態への変更 (a) とフラグの埋め込みによる有効活用 (b)

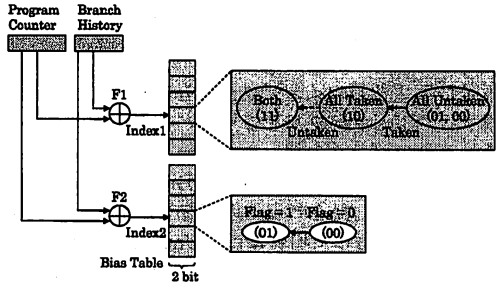


図 5 Dual-indexed Bias テーブル

約 87%と高くなる．このことから，この状態に 1 ビットのフラグを埋め込むことで 4 状態として，テーブルを構成するエントリの 2 ビットを有効に活用する．この様子を図 4(b) に示す．これにより，約 87%のエントリにおいて，1 ビットのフラグを利用できるようになる．

次に，埋め込まれた 1 ビットのフラグの使い方を検討する．ある分岐の予測をおこなう時に，従来と同様のインデックスを採用するだけでは，利用するエントリが変わらないためにテーブルの利用効率は向上しない．そこで，ある分岐の予測をおこなう時に，異なるエントリを指し示す 2 つのインデックスを生成し**，選択された 2 つのエントリを利用する方式を考える．2 つのインデックスを index1, index2 と記述する．index1 を用いて参照したエントリは，図 4(a) に示す 3 状態の構成により極端な偏りのある分岐を検出する．一方，index2 を用いて参照したエントリは，組み込まれたフラグを用いて極端な偏りのある分岐を検出する．このように，2 つのインデックスを用いてテーブルを参照する方式を **Dual-indexed Bias テーブル**と呼ぶことにする***．

図 5 に，提案する Dual-indexed Bias テーブルの構成を示す．プログラムカウンタとグローバル分岐

** 2 つより多い数のインデックスを利用することで予測精度が向上することを確認しているが，本稿では，2 つのインデックスを利用する方式のみを議論の対象とする．

*** 埋め込まれた 1 ビットのフラグを活用する方式に関しては検討の余地がある．これは今後の課題である．

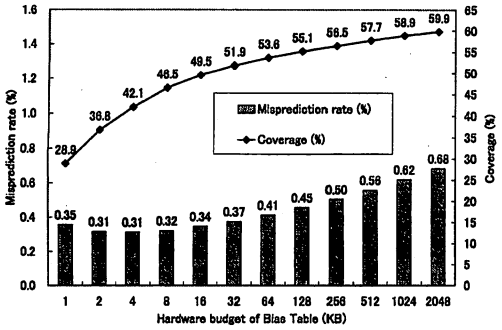


図 6 ハードウェア量を変化させて測定した Dual-indexed Bias テーブルのミス率と適用範囲

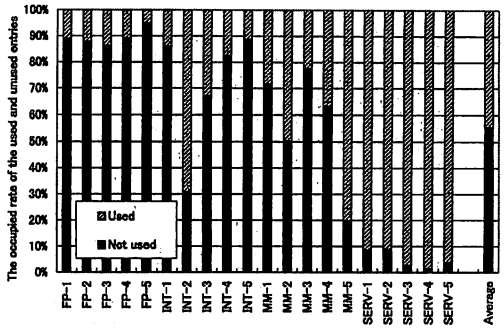


図 7 4KB の Dual-indexed Bias テーブルにおけるプログラム終了時の利用エントリの割合

履歴から 2 種類のインデックス index1, index2 を作成する．Bias テーブルを 2 つにバンク化することで，それぞれのバンクの読み出しポート数を 1 に維持できる．まず，index1 を利用して Bias テーブルを参照する．読み出した 2 ビットの値を 3 状態として捉え，All Untaken だった場合には，Untaken と予測する．All Taken だった場合には Taken と予測する．続いて，index2 を利用して Bias テーブルを参照する．読み出した 2 ビットの状態が All Untaken (Flag=0 あるいは Flag=1) の場合には，埋め込まれたフラグの値を利用する．この時，極端な偏りのある不成立分岐が多いことから，Flag=0 を All Untaken, Flag=1 を Both の状態と捉え，Flag=0 の時に Untaken と予測する．これらに当てはまらない場合には極端な偏りはないとして，他の機構により予測をおこなう．

Dual-indexed Bias テーブルの更新方法をまとめる．index1 により選択したエントリは，分岐結果に応じて，図 4(a) の状態推移に応じて値を変更する．index2 により選択したエントリは，Flag=0 の状態で，かつ分岐結果が成立の時に，Flag=1 の状態に変更する．

2 つのインデックスの生成方法についてまとめる．index2 は，より積極的に分岐履歴を利用するように調整する．index1 は従来手法と同様に求める．具体的には，C 言語の演算子を利用して次式により 2 つのインデックスを求める．

```
index1 = ( pc       ^ (bhr << 11)) % TABLE_SIZE;
index2 = ((pc >> 1) ^ (bhr <<  5)) % TABLE_SIZE;
```

ただし，異なるバンクを指し示すように，index1 の最上位ビットを反転したものを index2 の最上位ビットとする．

図 6 に，ハードウェア量を変化させて測定した Dual-indexed Bias テーブルのミス率と適用範囲をまとめる．縦軸，横軸は図 2 と同じである．4KB のハードウェア量で比較すると，図 2 に示した Bias テーブルの適用範囲は 36.6%であるが，dual-indexed Bias テーブルでは 42.1%と 5.5 ポイントの改善を得る．一方，予測精度は 0.23%から 0.31%へと悪化する．

図 7 に，4KB の Dual-indexed Bias テーブルにおけるプログラム終了時の利用エントリの割合を示す．このデータは，テーブルのそれぞれのエントリに，それが利用されたかどうかを示すフラグを付加し，これを利用して，プログラム終了時に利用されていないエントリの割合 (Not used) とそうでないエントリの割合 (used) を計測したものである．図 3 にまとめた様に，従来手法における未使用エントリの割合は 75%であるが，Dual-indexed Bias テーブルでは未使用エントリの割合を 55%へと改善できている．しかしながら，依然として半数を超えるエントリが未使用のままである．これらを有効に利用する方式の開発は，今後の課題である．

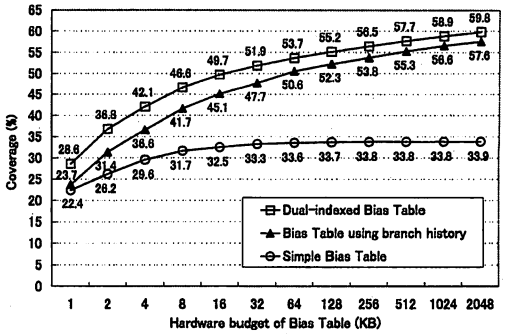


図 8 3 つの Bias テーブルにおける適用範囲 (coverage) の比較

2.4 幾つかの Bias テーブルの比較

図 8 に，3 つの Bias テーブルにおける適用範囲の比較をまとめる．Dual-indexed bias テーブルと図 1 に示した Bias テーブル (Bias Table using branch history) に加えて，図 1(a) のインデックス生成に分岐履歴を利用しないシンプルな Bias テーブル (Simple Bias Table) の 3 つの適用範囲を示した．

図 8 の結果から次の知見を得る．分岐履歴を利用しないシンプルな Bias テーブルでは，32KB 程度のハードウェア量で適用範囲が飽和する．他の方式との比較から，分岐履歴を利用することの利点が大きいことが

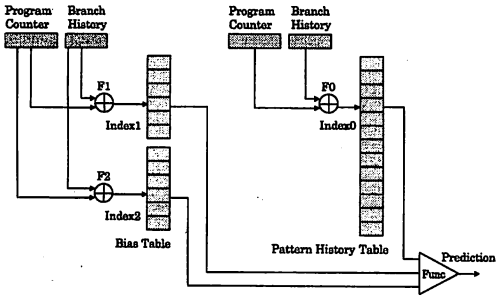


図 9 Gshare+分岐予測

わかる．次に，Dual-indexed Bias テーブルと分岐履歴を利用する Bias テーブルとを比較する．全てのハードウェア量において，提案方式の Dual-indexed Bias テーブルが高い適用範囲を示す．特に 2〜8KB の現実的なハードウェア量の設定において，Dual-indexed Bias テーブルの利得が大きい．また，例えば，4KB の Dual-indexed Bias テーブルは，8KB の分岐履歴を利用する Bias テーブルより性能が高いことから，Dual-indexed Bias テーブルを用いることで同様の適用範囲を得るためのハードウェア量を半分以下に削減できる．

3. Gshare+分岐予測の提案

本章では，Dual-indexed Bias テーブルを利用する改良版の gshare 分岐予測として，gshare+分岐予測を提案する．

図 9 に，gshare+分岐予測の構成を示す．gshare+分岐予測は，図 9 右に示した gshare 分岐予測に，図 5 の Dual-indexed Bias テーブルを追加した構成を持つ．

予測方法は次の通りである．まず，次式により 3 つのインデックスを求め，Bias テーブルとパターン履歴表を参照する．

```
index1 = ( pc       ^ (bhr << 11)) % TABLE_SIZE;
index2 = ((pc >> 1) ^ (bhr <<  5)) % TABLE_SIZE;
index0 = ( pc       ^  bhr_sel   ) % TABLE_SIZE;
```

ただし，パターン履歴表のインデックス index0 の分岐履歴 bhr_sel は，通常の分岐履歴 bhr と index1 により極端な偏りのある分岐と判断されない場合に更新する BHR_NOB[5] の 2 つから適切に選択する．bhr の半分以上のビットが極端な偏りのある分岐と判断された時に，bhr の情報量が少ないと判断して，BHR_NOB を選択する．

もし，index1 で参照した Bias テーブルのエントリの状態が All Untaken であれば Untaken と予測し，All Taken であれば Taken と予測する．そうではなく，index2 で参照した Bias テーブルのエントリの状態が Flag=0(All Untaken) だった場合には，極端な偏りのある不成立分岐と判断し，Untaken と予測する．これらに当てはまらない場合には極端な偏りを持つ分岐ではないと判断し，index0 により参照したパターン履歴表の状態に基づいて gshare と同様に予測をおこなう．

テーブルの更新方法は次のとおりである．Bias テーブルの更新方法は 2.3 節と同じである．Bias テーブルによって極端な偏りのある分岐と判断されずに，gshare のパターン履歴表に基づいて予測をおこなった場合にのみ，gshare と同様の方式でパターン履歴表の状態を更新する．

4. 評　価

4.1 評価方法

トレース駆動のシミュレータを利用して提案方式の gshare+の予測精度を，従来手法の gshare, bimode と比較する．

ベンチマークプログラムは，Intel MRL と IEEE TC-uARCH の支援で 2004 年に開催された分岐予測のコンテスト Championship Branch Prediction のために作成された 20 本のトレースを利用する．これら 20 本のトレースは，SpecFP, SpecINT, マルチメディア，サーバという異なる 4 つの応用分野の実行履歴を集めたものである．

それぞれの分岐予測は，予測精度が高くなるように調整した結果から，次に示す構成を利用する．Gshare+分岐予測は，Bias テーブルとパターン履歴表のエントリ数を同じとする．Bimode 分岐予測は，Choice PHT のエントリ数を Direction PHT のエントリ数の 2 倍に設定する．例えば，8KB の場合，Choice PHT を 4KB, Taken PHT を 2KB, Untaken PHT を 2KB の合計 8KB とする．

4.2 評価結果

図 10，**図 11** に，ハードウェア量を 8KB と 16KB に設定した場合の予測精度をまとめる．SpecFP, SpecINT, マルチメディア (MM), サーバ (SERV) と全体の平均 (Average) の予測精度を示す．Gshare+, gshare, bimode と，比較のために図 1 の Dual-index を利用しない Bias テーブルを追加した gshare (Gshare with Bias Table) のデータをまとめる．

図 10，図 11 から，ほとんどのベンチマークにおいて，gshare+が最も高い予測精度を達成することがわかる．8KB のハードウェア量の時，gshare+の予測ミスは 3.92%で，gshare と比較して 1.00 ポイント，bimode と比較して 0.45 ポイントの予測ミスを削減する．この時，gshare+は，gshare を基準に 20.0%，bimode を基準に 10.3%の予測ミスを削減する．同様に，16KB のハードウェア量の時，gshare+の予測ミスは 3.50%で，gshare と比較して 1.04 ポイント，bimode と比較して 0.52 ポイントの予測ミスを削減する．この時，gshare+は，gshare を基準に 22.9%，bimode を基準に 13.0%の予測ミスを削減する．

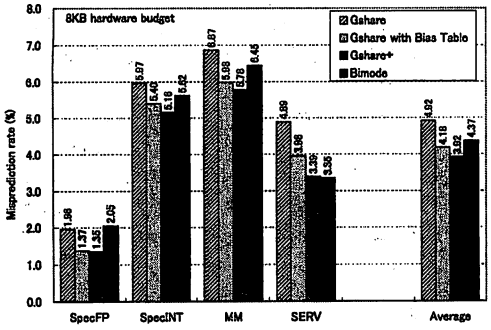


図 10 8KB のハードウェア量における分岐予測の予測精度の比較

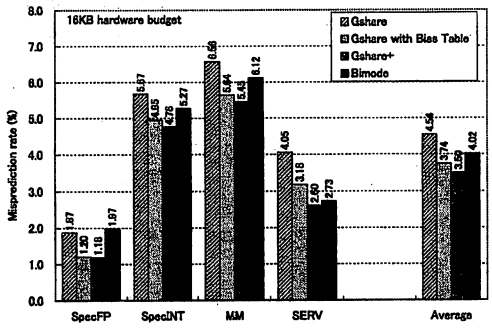


図 11 16KB のハードウェア量における分岐予測の予測精度の比較

gshare+と gshare with Bias Table の比較から，2 つのインデックスを用いることによる利得を知ることができる．8KB の時，平均で 0.26，サーバ系のベンチマークで 0.57 ポイントの改善を示す．このように 2 つのインデックスを利用することで一定の精度向上を得るが，一方で，ハードウェアが複雑になる．これらのトレードオフを考慮して適切な方式を選択することが重要となる．

図 12 に，ハードウェア量を変化させた時の予測精度をまとめる．ハードウェア量が 2KB と少ないときには，bimode が最も良い精度を示す．一方で，2KB を超える所では gshare+の精度が最も良く，8KB の gshare+は 64KB の gshare より高い精度を示す．また，16KB の gshare+は 64KB の bimode を超える精度を達成する．このように，gshare+は，同様の精度を得るためのハードウェア量を大幅に削減できる．

5. おわりに

極端な偏りをもつ分岐を検出する Bias テーブルの改良方式を検討し，2 つのインデックスを利用してテーブルの利用効率を向上させる Dual-indexed Bias テーブルを提案した．4KB のハードウェア量の Dual-indexed Bias テーブルを用いることで，0.32%のミス率という非常に高い精度で，42%の分岐を予測できる．

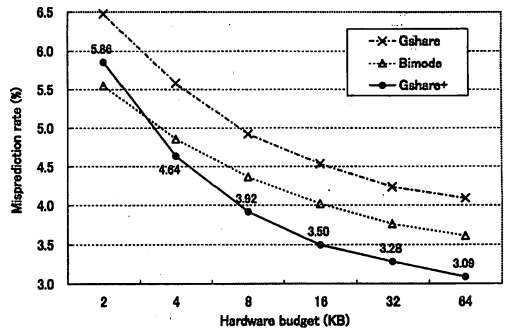


図 12 ハードウェア量を変化させた時の予測精度の比較

Dual-indexed Bias テーブルを用いた新しい構成の分岐予測として，現実的な構成を保ちながら高い予測精度を達成する gshare+分岐予測を提案した．ベンチマークプログラムを用いて gshare+, gshare, bimode を比較した結果から，2KB を超えるハードウェア量において，gshare+分岐予測が最も高い予測精度を達成することを示した．8KB のハードウェア量の gshare+の予測精度は 64KB の gshare より高く，16KB の gshare+は 64KB の bimode より高いことを明らかにした．

謝　辞

本研究の一部は，科学研究費補助金基盤研究 (B) 課題番号 17360178 の助成による．

参考文献

1) Kise, K., Katagiri, T., Honda, H. and Yuba, T.: The Bimode++ Branch Predictor, *Proceedings of the 8th International Workshop on Innovative Architecture for Future Generation High-Performance Processors and Systems (IWIA-2005)*, pp. 19–26 (2005).
2) Lee, C.-C., Chen, I.-C. K. and Mudge, T. N.: The bi-mode branch predictor, *Proceedings of International Symposium on Microarchitecture*, pp. 4–13 (1997).
3) McFarling, S.: Combining branch predictors, Technical report, Digital Equipment Corporation WRL TN-36 (1993).
4) Seng, J. S. and Tullsen, D. M.: Exploring the Potential of Architecture-Level Power Optimizations, *Third International Workshop on Power-Aware Computer Systems* (2004).
5) 吉瀬謙二，片桐孝洋，本多弘樹，弓場敏嗣: Bimode-Plus 分岐予測器の提案，情報処理学会論文誌コンピューティングシステム，Vol.46, No.SIG 7(ACS 10), pp. 85–102 (2005).