計算機アーキテクチャ 127－9
設 計 自 動 化 86－9
（1997. 12. 11）

競合による予測精度低下を緩和する分岐予測機構

野口良太† 森 敦司† 小林良太郎†
安藤秀樹† 島田俊夫†

命令レベル並列性を利用する高性能プロセッサにおいては、分岐予測機構の予測精度が性能に大きな影響を及ぼす。この分岐予測機構の予測精度を制限する要因として競合という問題がある。これまで、競合に対する対策としては競合の発生そのものを減少させる試みがなされてきたが、この方策だけでは限界がある。我々は予測に悪影響を与える破壊的競合の削減に焦点を当て、競合が発生しても予測精度を大きく落さないgcshare予測機構を提案する。シミュレーションによる性能評価の結果、gcshare予測機構は既存のgshare予測機構と同程度の予測精度を半分から3分の1程度のハードウェアの規模で構成できることがわかった。

Branch Prediction Scheme with Reducing Destructive Aliasing

NOGUCHI RYOUTA,† MORI ATSUSI,† KOBAYASI RYOUTAROU,†
ANDO HIDEKI† and SHIMADA TOSHIO†

Modern high performance microprocessors using instruction level parallelism requires accurate branch prediction. Aliasing in the branch predictor table is a main factor of limitting prediction accuracy. To solve this problem, reserchers focused on reducing the number of aliasing. Yet only this approach is not enough to gain high prediction accuracy.

In this paper, we focus on reducing destructive aliasing, which causes wrong prediction. We propose the gcshare branch prediction scheme that reduces destructive aliasing. Our experimental results show that the gcshare can reduce the amount of hardware by from approximately 30% to 50% to achieve the same prediction accuracy that the gshare can achieve.

1. はじめに

近年の高性能プロセッサは深いパイプライン構成及びスーパースカラに代表される複数命令の同時発行機構により命令レベル並列性を引き出し性能を向上させている。分岐命令は、こうした命令レベル並列性を利用したプロセッサの性能を厳しく制限する。この影響を緩和するために分岐命令の結果が判明する以前にその分岐方向を予測し後続命令を投機的に実行するという手法が一般的に用いられている。

分岐予測機構はこの分岐方向の予測を行なう機構である。この予測が誤っていた場合には、投機的に発行したすべての命令の実行を取り消してその分岐命令が実行される以前のプロセッサ状態に戻した上で、新たに正しい分岐方向の制御流に従い命令実行を再開しなければならない。現在においてもパイプライン段数がより深く命令発行幅も広がる傾向にあり、予測に失敗することで破棄される命令数は今後も増加すると考えられる。従って、分岐予測機構の予測正確さは今後ますます重要なものとなる。

過去、さまざまな分岐予測機構が提案されてきた[2)7)6)]。これらの多くは、その分岐命令のアドレス情報と過去の分岐の方向の履歴の相関に基づいて予測を行なう。実際の機構としては多数の2ビットカウンタで構成されるテーブルを用意し、これを分岐命令のアドレス及び過去の分岐履歴を記録したシフトレジスタの値と対応づけることで実現している。しかし、分岐命令アドレス、分岐履歴の取り得るすべての組み合わせに対応できる巨大な2ビットカウンタテーブルを用意することは、ハードウェアの制約上現実的でない。その結果、異なる分岐アドレス及び分岐履歴でありながら同じ2ビットカウンタに対応づけられてしまう競合の問題が発生する。

最近の研究では、特にカーネルの振る舞いをも考慮した実際的な環境においては、こうした分岐履歴の競合が頻発し、その結果として予測性能が低下するという問題が注目されている[1)4)]。この問題に対して分岐アドレスと分岐履歴を2ビットカウンタテーブルに対応づけるマッピング関数を工夫し競合を減少させることで対応しようとした研究がある[3)4)]。しかし、このアプローチのみによる性能改善は限界がある。何故ならば現実的なハードウェアの制約のもとで分岐命令アドレスと分岐履歴の取り得るすべての組み合わせに対応できる巨大な2ビットカウンタテーブルを保有することが不可能である以上、いかに優れた競合を回避するマッピング手法を用いたと

† 名古屋大学工学部
School of Engineering, Nagoya University

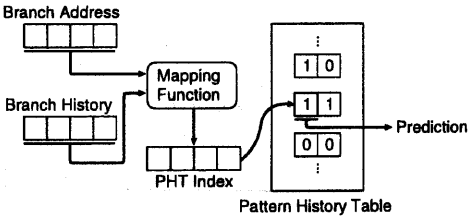


図1　2レベル予測機構の仕組み

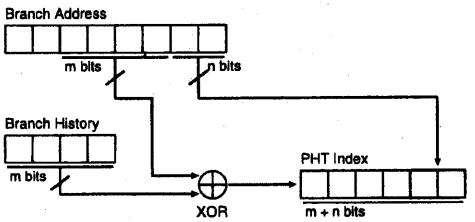


図2　gshare 予測機構のマッピング関数

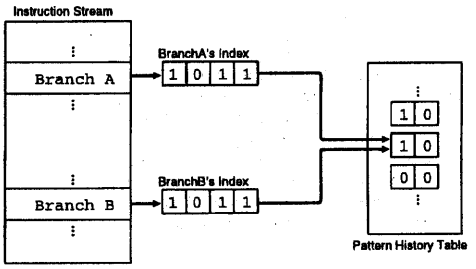


図3　PHT の競合

ころで競合は不可避であるからである。

我々は競合の問題に対して、競合自体を削減するのではなく、競合が生じても予測精度が低下しない方策を考える。具体的には、分岐命令アドレスと履歴の組合せに対し、競合が生じた場合でも、分岐方向の偏りが同一のものが競合するように2ビットカウンタテーブルを構成する。こうすることにより、競合により誤った予測を行う確率を下げる。

2章では、分岐予測機構及び競合についての過去の研究について述べる。3章では、我々の取る競合の問題に対するアプローチについて述べる。4章では、本アプローチに基づく対策を施した予測機構として gcshare 予測機構を提案しその詳細を述べる。5章では同機構の性能の評価を行ない本アプローチの有効性を検証する。6章では本稿をまとめる。

2. 関連研究

2.1 2レベル予測機構

現在の分岐予測方式の主流となっているものが、2レベル予測機構[6][7]と呼ばれるものである。図1にこの機構の概要を示す。この機構は、分岐命令の命令アドレスと、過去の分岐履歴の二つの情報をインデックスとして、パターン履歴テーブル（PHT:Pattern History Table）中のそれに対応する2ビットカウンタを参照しこの最上位ビットの値が予測値として出力される。分岐結果が判明するとこの情報が予測機構へフィードバックされ、分岐履歴を保持するシフトレジスタの更新、及び2ビットカウンタの値の増減を行ない自らの蓄える情報を更新する。

この予測機構には、履歴の取り方や PHT へのマッピング手法などで多くのバリエーションが考えられてきた。McFarling の提案した **gshare 予測機構**[2]はそれらの中でも極めて高精度な予測を行なう機構である。この予測機構では、制御の流れに沿って分岐履歴を蓄える m ビットのグローバル分岐履歴と、予測する分岐命令の命令アドレスの下位 $m+n$ ビットとの XOR を取り、この値に対応する2ビットカウンタを参照して予測結果を出力する（図2）。本稿では、この二つのパラメータのうち m を履歴ビット長、n を追加アドレスビット長と呼ぶ。

2.2 競　合

図3のように、異なるアドレスまたは履歴を持つ分岐が、PHT 中の同じカウンタに対応づけられる現象を PHT の**競合**（Aliasing）と呼ぶ。

カーネルレベルの振る舞いを考慮に入れたより実際的な環境においては、PHT での競合が予測精度に大きな影響を与えることがわかり[1][4]、近年この現象に対する関心が高まっている。

こうした中で、Young らは分岐予測に与える影響の観点から競合を以下の三種類に分類した[8]。

建設的競合 (Constructive Aliasing)　競合しない時と異なる予測結果を出力し、その結果正しい予測結果となった競合。

破壊的競合 (Destructive Aliasing)　競合しない時と異なる予測結果を出力し、その結果誤った予測結果となった競合。

無害な競合 (Harmless Aliasing)　競合しない時と同じ予測結果を出力する競合。

競合しない時の予測結果との比較は、競合の発生しない無限のエントリを持つ理想的な予測機構の予測結果との比較によって求めることができる。Young らは、建設的競合と破壊的競合の発生頻度を調査し、前者に比べ後者の発生頻度が圧倒的に多いという結果を得た。

Michaud らは、キャッシュミスの分類にヒントを得て、Young とは別に以下の3種類のような競合の分類を行った[3]。

初期競合 (Compulsory Aliasing)　初めてテーブルを参照した際に発生する競合

容量競合 (Capacity Aliasing)　テーブルの容量不足に起因する競合

対立競合 (Conflict Aliasing)　マッピングに起因する

競合

3. 破壊的競合の削減

競合という問題に対するこれまでのアプローチは主にマッピング関数を工夫して、競合の発生頻度を減少させるというものであった[3)4)]。この解決アプローチでは、対立競合の削減に効果があるものの、初期競合や容量競合の発生頻度を削減することはできない。そのため、この解決アプローチのみでは、競合による性能低下を取り去るには不充分である。

しかし、初期競合と容量競合の出現頻度を削減することは難しい。初期競合は初めて参照される際には必ず発生するのでこの競合の出現を妨げることはできない。容量競合についても、この出現頻度を下げるためにはテーブルの容量を大きくする以外に有効な手段が見つからない。このことは、現実的なハードウェアの制約のもとでは競合の発生は本質的に不可避であることを意味する。

このように、競合の発生がある程度は避けられないものであるならば、より高精度な予測を行う機構を構成するためには、たとえ競合が起きても予測精度を下げない仕組み、すなわち競合が破壊的なものとなりにくい仕組みを構築する必要がある。

ところで、多くの分岐は、Taken と Not-Taken のどちらかに偏っている。このため、多くの競合は互いに異なる方向に偏った分岐の間か、同じ方向に偏った分岐の間で発生する。互いに異なる方向に偏っている分岐が競合を起こした場合、対応づけられたカウンタは Taken を出力する状態と Not-Taken を出力する状態の間を激しく振動することになり正確な予測を行うことができない。こうして、競合状態になければ正確に予測できる分岐の予測が阻害されることになる。これは破壊的競合である。逆に、同じ方向に偏っている分岐が競合を起こした場合には、カウンタは同じ方向へ遷移するため競合が発生しない状態と同じく正確な予測を行うことができる。

このことから、競合を引き起こす分岐の方向が同じになるような仕組みを設けることで、予測精度を悪化させる破壊的競合を削減できる。

4. gcshare 予測機構

前節では、競合を引き起こした分岐の分岐方向をそろえることで、破壊的競合が削減できることを述べた。

これを実現するために、我々は PHT を分岐結果が Taken に偏った分岐用のテーブルと、Not-Taken に偏った分岐用のテーブルに分離し別々に管理する手法を提案する。PHT を Taken に偏った分岐用と Not-Taken に偏った分岐用とに分離することは、仮に競合が発生したとしても、同じ分岐方向同士の分岐情報の競合、すなわち無害な競合となる可能性を高め、逆の分岐方向同士の情報の競合、つまり破壊的競合となる可能性を下げる働きを持つ。

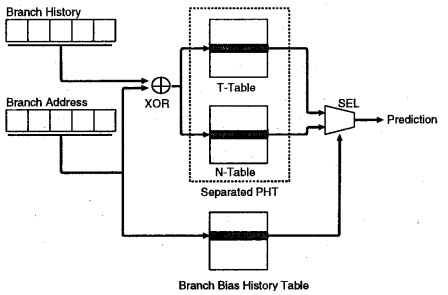


図4 gcshare 予測機構

gshare 予測機構にこの手法を適用したものを **gcshare** 予測機構と呼ぶ。以下では、この予測機構の仕組みの詳細について述べる。

4.1 gcshare 予測機構の構成

gcshare 予測機構は、T-テーブル、N-テーブル、及び分岐偏り履歴テーブル (**BBHT:Branch Bias History Table**) の 3 つテーブルで構成される（図 4)。

T-テーブルと N-テーブルは、これまでの 2 レベル予測機構における単一の PHT を Taken に偏った分岐用のものと、Not-Taken に偏った分岐用のものとに分離したものである。どちらも、通常の gshare 予測機構と同じく分岐アドレスと分岐履歴の XOR を取った情報をテーブルのインデックスとする。

BBHT は分岐の偏り情報を保持し、T-テーブルと N-テーブルのうちどちらのテーブルの結果を使用するかの選択を行うのに用いる。テーブルのインデックスには、分岐命令のアドレスのみを用いる。

予測結果は以下の手順で出力される。まず、3 つのテーブルを同時に参照する。この時、出力される T-テーブル及び N-テーブルの出力値を BBHT の出力結果により選択し、これを最終的な予測結果とする。それぞれのテーブルのカウンタの更新は、BBHT と T-テーブルと N-テーブルのうち選択されたものについてのみ行う。

4.2 gcshare 予測機構のコスト配分

gcshare を実現する際、与えられたハードウェアコストを BBHT, T-テーブル、N-テーブル間で最適に配分する構成を見出す必要がある。本節では、このコスト配分に関して議論する。

4.2.1 BBHT

BBHT は PHT で破壊的競合を抑えるための分岐のマッピングを行うので、BBHT で競合が起きると gcshare の機能が著しく低下すると考えられる。したがって、BBHT には PHT に優先してコストを配分すべきと考えられる。

BBHT のコストは、エントリ数とカウンタビット長の

積によって決まるので、これらの間にトレードオフが存在する。これに関しては、カウンタビット長に優先しエントリ数にコストを割り当てるべきと考えられる。なぜなら、先に述べたようにBBHTでの競合はgcshareの機能を著しく低下させると考えられるため、競合が生じないようにエントリ数は充分でなければならない。

分岐方向の偏りは、PHTに2ビットカウンタを用いて管理していることからわかるように、高々2ビットあれば良いことが知られている。しかし、コスト制約が非常に厳しく、充分なエントリ数が得られない場合は、カウンタのビット長を1ビットにする選択が考えられる。逆に、コスト制約が緩く、充分なエントリ数が確保できるならば、カウンタのビット長を伸ばし、より多くの状態を管理することで精度の高い分岐偏り情報を得るという選択も存在する。

4.2.2 T-テーブル、N-テーブル

通常、Taken側に偏った分岐の方が出現頻度が高い。よって、N-テーブルよりもT-テーブルにより多くのコストを割くことで、より効率的なコスト配分ができる。

T-テーブル、N-テーブルの大きさは、そのインデックスのビット長で決定される。N-テーブルがT-テーブルより小さくなるように実装することは、N-テーブルのインデックスのビット長をT-テーブルのそれより短くすることで実現できる。インデックスビットをnビット小さくした時のT-テーブルとN-テーブルの大きさの比は$2^n:1$となる。

インデックスをどのようにして短くするかについては、追加アドレスビット長を削減する方法と履歴ビット長を削減する方法の二つが考えられる。

5. 性能評価

5.1 評価環境

トレース駆動シミュレーションによる性能評価を行った。ベンチマークにはIBS-Ultrixを用いた[5]。表1に各ベンチマークに含まれる動的な分岐数と静的な分岐数の詳細を示す。これらの各トレースはDECstation(MIPS R3000)にハードウェアモニタをつなぐことで、OS(Ultrix3.1)の実行トレースをも含む形で採取された。そのため、このベンチマークは、アプリケーションのみの実行トレースに比べ、より実際の環境に近い実行トレースを提供する。

5.2 gcshare予測機構のコスト配分

gcshare予測機構におけるパラメータを以下に列挙する。

- BBHTのエントリ数
- BBHTのカウンタのビット長
- PHTの履歴ビット長
- PHTの追加アドレスビット長
- T-テーブルとN-テーブルの大きさの比

表1 IBS-Ultrix Benchmark

benchmark	dynamic branches	static branches
groff	11,568,181	5,634
gs	14,288,742	10,935
jpeg_play	20,926,069	6,716
mpeg_play	8,109,029	4,752
nroff	21,368,201	4,480
real_gcc	13,940,672	16,716
sdet	5,221,321	4,583
verilog	5,692,823	3,918
video_play	5,175,630	3,977

これらのすべてのパラメータの組み合わせについて性能評価を行うことは困難である。そこで、この節ではBBHTのエントリ数とカウンタのビット長、及びTテーブルとNテーブルの大きさの比の2点について性能に与える影響を調査し最適なコスト配分を検討する。

この節の結果は、以降の節で詳細な性能評価を行う際の評価モデルのパラメータ設定に用いる。

5.2.1 BBHTのエントリ数、カウンタビット長

まず、BBHTについての検討を行なう。図5は履歴ビット長が10で追加アドレスビット長が0のgcshare予測機構のBBHTのエントリ数、及び構成するカウンタのビット数を変化させた時の全ベンチマーク平均での予測ミス率を示したものである。

図5からエントリ数が2K以下とそれほど大きくない時には、2ビットや3ビットのカウンタを用いるよりも、1ビットのものを用いた方が性能が良いことがわかる。これは、競合が発生した場合状態数の少ない1ビットカウンタの方が早く状態の遷移ができ競合時の影響が小さいためと考えられる。

BBHTを1ビットカウンタで構成する場合は、8Kエントリ程度でその性能は飽和する。このため、これ以上エントリ数を大きくすることはあまり意味がない。この8Kエントリ時に要するハードウェア量は1Kbyteに相当する。

BBHTにより多くのハードウェアが割けるならば、BBHTを2ビットカウンタで構成する選択もある。2ビットカウンタで構成する場合には、16Kエントリ程度で性能は飽和していることから、このエントリ数が適当であると考えられる。この時のBBHTのハードウェア量は4Kbyteに相当する。

BBHTのカウンタビット長を3ビットにまで伸ばした場合は、32Kエントリ与えても同じエントリ数で2ビットカウンタで構成したものの性能を上回ることができない。この32Kエントリの時のハードウェア量は12Kbyteにもなることから、3ビット以上のカウンタを用いることは良い選択ではない。

以上のことから、低コストの予測機構においては1ビットのBBHTを、大規模な予測機構では2ビットのBBHTを用いることが適当である。

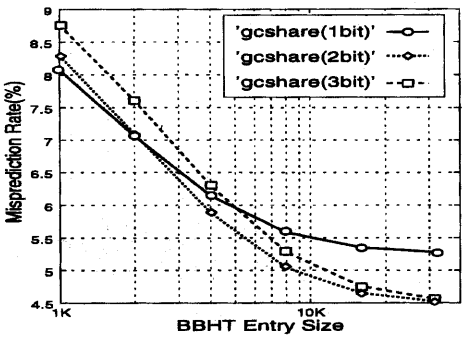
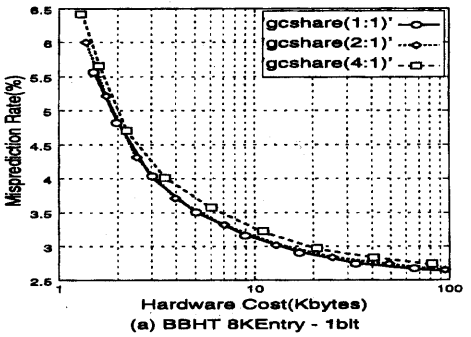


図5　BBHTのエントリ数とカウンタビット長の性能に与える影響



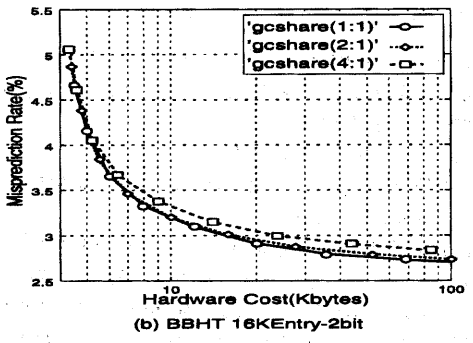


図6　T-テーブル、N-テーブルのコスト配分

以降の性能評価では、BBHTに8Kエントリの1ビットカウンタで構成したものと、16Kエントリの2ビットカウンタを用いたものについてのみ評価する。

5.2.2　T-テーブル、N-テーブルのコスト配分

次にT-テーブルとN-テーブルの大きさの比についての検討を行なう。図6(a)はBBHTが8Kエントリの1ビットカウンタで構成された10履歴のgcshare予測機構で、T-テーブルとN-テーブルのエントリ数の比を1対1、2対1、4対1と変化させた時の予測性能の変化を示したものである。図6(b)は、同様の測定をBBHTが16Kエントリの2ビットカウンタで構成したもので行なった場合の結果である。

ここでのインデックスの削減手法には、ともに履歴ビット長を削減する手法を用いた。追加アドレスビット長を削減する手法についても同様の測定を行ったが、性能差がほとんどないためこちらについては図中から省略する。

結果から、1ビット、2ビットともに1対1、2対1の構成比は性能差がほとんどない。4対1以上の構成比はハードウェア削減効果よりも性能の低下の方が大きく良い選択ではないことがわかる。

この結果に基づき、以下ではPHTを1対1に分割したものについてのみ詳細な性能評価を行なうことにする。

5.3　PHT分離の効果

この節では、PHTをT-テーブルとN-テーブルに分離することで、予測性能と競合内容にどのような効果が現れるかを調査する。

PHTの大きさを32Kエントリとし、以下の3つの予測機構について性能評価を行った。

- gshare予測機構(gshare)
- BBHTを1ビット×8Kエントリで構成しPHTを等分に分割したgcshare予測機構(gcshare1)
- BBHTを2ビット×8Kエントリで構成しPHTを等分に分割したgcshare予測機構(gcshare2)

5.3.1　予測性能

それぞれの評価モデルについて履歴ビット長を変化させた時の予測ミス率の変化を示したのが図7である。ここでは、PHTのエントリ数は固定されているので、履歴ビット長を1ビット伸ばすと追加アドレスビット長は1ビット減ることになる。

履歴ビット長が0から2ビット程度の極めて短い場合を除いてBBHTを加えPHTを分割したことで、予測性能はもとのgshare予測機構より予測精度は0.5%程度向上していることがわかる。

どの予測機構もある程度までは、履歴ビット長が増加すると予測精度が向上するが、それ以上になると予測精度は低下する。これは、追加アドレスビット長が短くなったことで競合が頻発し、これによる予測精度低下が、履歴ビット長が伸びたことによる性能向上の効果よりも大きなものとなるためである。

単一のPHTで構成したgshare予測機構では、9履歴を越えると競合による予測精度低下が大きく、性能はこれ以上向上しない。これに対して、同じ大きさのPHTを分離したgcshare予測機構では10履歴以上の長い履歴を与えても、競合の悪影響は大きくなく性能を大きく落とさない。このことは、より小さいPHTでもgshareと同程度の分岐履歴を運用した高精度予測機構を構築できることを示唆している。

5.3.2　競合内容

図8は、各予測機構の競合の発生頻度とその種類の内訳を示したものである。縦軸の競合頻度(Aliasing Rate)

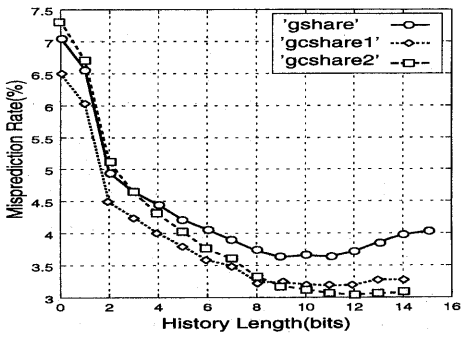


図7　32K エントリ PHT の時の予測精度

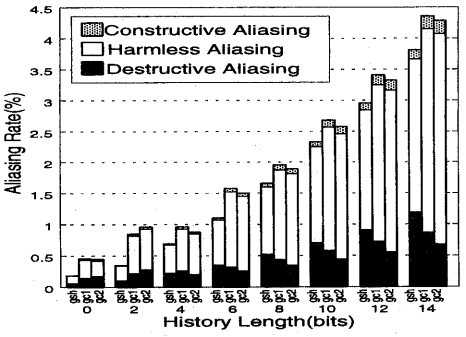


図8　競合の発生頻度とその内訳

は競合の発生回数を実行した条件分岐数で割って算出したもので、どれくらいの割合で競合が発生するかを示している。横軸は履歴ビット長を示す。0から14までの各履歴ビット長ごとにある3つの帯グラフは、上記の3つの予測機構の競合頻度を建設的競合、無害な競合、破壊的競合の3つに分類して示している。これらの競合の分類は、同じ履歴ビット長を持ちPHTのサイズが無限である競合のない予測機構との競合時の予測結果の比較によって競合を分類している。予測結果が同じならば無害な競合であり、競合のない予測機構が予測失敗で評価対象の予測機構が予測に成功していれば建設的競合、逆は破壊的競合と分類される。

いずれのグラフについても、PHT を分離した gcshare 予測機構では競合の発生頻度そのものは 0.2%から 0.3%程度増加している。これは、gcshare 予測機構では、BBHT の出力する分岐の偏り情報が変化することによって全く同じアドレスと分岐履歴を持つものであっても別の場所へ対応づけられることがあるためである。こうして、単一の PHT の時に比べ広い範囲に分岐の情報が分布することが、競合発生の増加につながっている。

しかし、競合自体は増加しているものの、予測に悪影響を与える破壊的競合については履歴ビット長が6を越えたところではむしろ減少している。特に履歴ビット長が長く、競合の発生頻度が高くなった時の効果は大きく、単一の PHT である gshare 予測機構での破壊的競合のおよそ 20%から 40%を削減している。

履歴ビット長が6以下と少なく競合自体があまり発生することのない場合は競合全体の発生頻度が高く、結果として破壊的競合は増加している。しかし、前節での予測性能の結果から見てもわかるように、これらの履歴ビット長が極めて短い時は競合がなくとも高い予測精度を達成することができず、こうした配分の時の結果は重要ではない。

また、予測に好影響を与える建設的競合は、gcshare 予測機構では競合全体の発生頻度が増加したため、すべての履歴ビット長の配分において、gshare 予測機構より

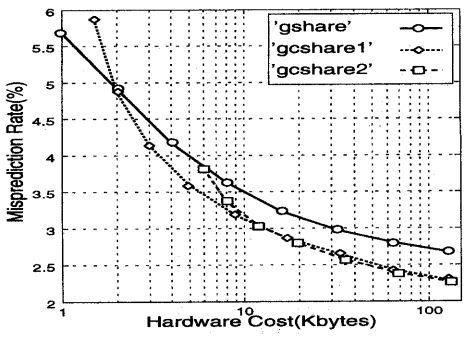


図9　予測精度の比較

高い。

以上より、PHT を分割したことで破壊的競合を無害な競合や建設的競合に変えるという仕組みは有効に機能していることがわかる。

5.4　コスト性能比の評価

この節では、gcshare 予測機構と gshare 予測機構のコスト性能比について調査する。

gshare 予測機構と2種類の gcshare 予測機構のそれぞれについて PHT の大きさを変化させて前節と同様に履歴ビット長と追加アドレスビット長の配分を変化させて性能評価を行い、そのそれぞれについて最も予測精度の良くなる最適な履歴ビット長を求めた。この時の各予測機構のハードウェア量と全ベンチマーク平均の予測ミス率の関係を図9に示す。

図9から、ハードウェア量が 2Kbyte 未満の時には、gcshare 予測機構は同規模の gshare 予測機構の性能に劣っていることがわかる。8K エントリの1ビットカウンタで構成した BBHT は 1Kbyte、16K エントリの2ビットカウンタで構成した場合にはこれだけで 4Kbyte ものハードウェア量を要する。そのため、こうした小規模時においては破壊的競合削減の効果よりも BBHT を加えたことによるコスト増加の影響が大きく予測精度が向上しない。

表2　各予測機構の履歴ビット長の配分

Predictor(m,n)	Cost(bytes)	Mispred(%)
gshare(4,8)	1K	5.67
gshare(5,8)	2K	4.92
gshare(9,5)	4K	4.17
gshare(9,6)	8K	3.63
gshare(11,5)	16K	3.23
gshare(12,5)	32K	2.97
gshare(12,6)	64K	2.81
gshare(16,3)	128K	2.67
gcshare1(7,3)	1.5K	5.87
gcshare1(6,5)	2K	4.87
gcshare1(9,3)	3K	4.14
gcshare1(9,4)	5K	3.59
gcshare1(11,3)	9K	3.18
gcshare1(11,4)	17K	2.87
gcshare1(14,2)	33K	2.65
gcshare1(17,0)	65K	2.43
gcshare1(18,0)	129K	2.29
gcshare2(10,2)	6K	3.82
gcshare2(11,2)	8K	3.36
gcshare2(12,2)	12K	3.04
gcshare2(13,2)	20K	2.79
gcshare2(16,0)	36K	2.56
gcshare2(17,0)	68K	2.40
gcshare2(18,0)	132K	2.26

逆に費やすことのできるハードウェア量が大きくなると、BBHTを追加することによるコスト増加は相対的に小さなものとなる。こうして、ハードウェア量がほとんど増加することなく破壊的競合が削減された結果、gshare予測機構に比べ約0.4%予測ミス率が減少している。

表2は、図9に使用した各予測機構の履歴ビット長(m)と追加アドレスビット長(n)の組み合わせを示したものである。

この表からgcshare予測機構は同程度の予測精度をもつgshare予測機構と比較して追加アドレスビット長が短くできることがわかる。

例えば、予測ミス率約3.2%を実現するのにgshare予測機構、1ビットのgcshare予測機構ともに11の履歴ビット長を要しているが、追加アドレスビット長はgshare予測機構が5であるのに、gcshare予測機構では2ビット少ない3で同程度の予測精度を達成している。また、予測ミス率約3.0%を達成するには、gshare予測機構、2ビットのgcshare予測機構ともに12の履歴ビット長を要しているが、追加アドレスビット長は前者は5、後者は2である。

これは、gcshare予測機構では破壊的競合が削減された結果、gshare予測機構と同じ履歴長を運用して同程度の予測精度を達成するのに必要とするPHTのハードウェア量が半分から4分の1で済むことを意味する。

このため、前述の2つの例では、それぞれ43.8%、67.5%も少ないハードウェア量でgshare予測機構と同程度の予測精度を達成している。

表3　ベンチマーク別の予測精度の比較

Benchmark	gshare(9,6)	gcshare2(11,2)
groff	3.27	2.80
gs	4.19	3.55
jpeg_play	0.90	0.84
mpeg_play	5.18	4.84
nroff	2.76	2.43
real_gcc	6.95	6.27
sdet	3.38	2.97
verilog	3.68	3.30
video_play	3.23	2.90

このことから、gcshare予測機構は非常にコスト性能比の高い予測機構であることがわかる。

5.5　ベンチマーク別の性能評価

これまで予測性能の評価には各ベンチマークの予測ミス率の平均を用いてきた。この節では、ハードウェアが同規模であるgshareとgcshareの各ベンチマーク別の性能を評価する。

表3にハードウェア量が8Kbyteの時のgshareと16Kエントリの2ビットカウンタで構成したBBHTを持つgcshareの各ベンチマーク別の予測性能を示す。履歴ビット長、追加アドレスビット長の配分は前節と同様のものでベンチマーク平均での性能が最も良くなるものである。

この結果から、8Kbyte時ではgcshare予測機構はすべてのベンチマークでgshareの性能を上回っている。これらのうちで特に性能向上が大きいのが、real_gccとgsで、それぞれ0.68%、0.64%向上している。これらのベンチマークはいずれも静的分岐数が多く、競合が発生しやすいという特徴を持つ。gcshare予測機構は競合の悪影響を緩和するため、これらのベンチマークでは特に性能向上が大きいと考えられる。

6. ま と め

競合はハードウェア資源の限られた2レベル予測機構の性能を制限する大きな要因である。扱う分岐アドレス情報や分岐履歴情報に比べて小さいテーブルしか用意できないのであれば本質的に競合は不可避な現象である。我々は、分岐方向を考慮したPHTの運用を行うことで破壊的競合が削減できることを示した。そして、この対策を施した予測機構としてgcshare予測機構を提案した。この予測機構では、Takenに偏った分岐とNot-Takenに偏った分岐を別のPHTに対応づけることで破壊的競合を削減する。この結果、gshare予測機構に比べ競合の悪影響が少なく、同じ大きさのPHTでより長い履歴を活用することができる。トレース駆動シミュレーション上で性能評価を行った結果、同程度の予測精度を構成するのに必要なハードウェア量がgshare予測機構の半分から3分の1程度で済み、ハードウェア量8Kbyte時において最大約0.7%の予測精度が向上することがわかった。

謝辞　本研究の一部は、文部省科学研究費補助金基盤研究 (B)「マルチスレッド型並列計算機方式の研究」の支援により行った。

参 考 文 献

1) N. Gloy, C. Young, J. B. Chen, and M. D. Smith, "An Analysis of Dynamic Branch Prediction Schemes on System Workloads," In *Proc. 23rd Annual International Symposium on Computer Architecture*, pp.12-21, May 1996.

2) S. McFarling, "Combining Branch Predictors, " *WRL Technical Note TN-36*, Digital Equipment Corporation, June 1993.

3) P. Michaud, A. Seznee and R. Uhlig, "Trading Conflict and Capacity Aliasing in Conditional Branch Predictors," In *Proc. 24th Annual International Symposium on Computer Architecture*, pp.292-303, May 1997.

4) S. Sechrest, C-C. Lee, and T. Mudge, "Correlation and Aliasing in Dynamic Branch Predictors," In *Proc. 23rd Annual International Symposium on Computer Architecture*, pp.22-32, May 1996.

5) R. Uhlig, D. Nagle, T. Mudge, S. Sechrest, and J. Emer, "Instruction Fetching: Coping with Code Bloat," In *Proc. 22nd Annual International Symposium on Computer Architecture*, pp.345-356, May 1995.

6) T-Y. Yeh and Y. Patt, "A Comparison of Dynamic Branch Predictors that use Two Levels of Branch History," In *Proc. 20th Annual International Symposium on Computer Architecture*, pp.257-266, May 1993.

7) T-Y. Yeh and Y.Patt, "Two-Level Adaptive Branch Prediction,"In *Proc. 24th Annual International Symposium and Workshop on Microarchitecture*, pp.55-61, November 1991.

8) C. Young, N. Gloy, and M. D. Smith, "A Comparative Analysis of Schemes for Correlated Branch Prediction," In *Proc. 22nd Annual International Symposium on Computer Architecture*, pp.276-286, May 1995.