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

2005－ARC－161　(10)
2005／1／21

弱偏向状態に着目した分岐予測手法

仲沢 由香里[†]　　斎藤 史子[†]　　山名 早人[‡]

近年，パイプライン段数の深化，命令フェッチ幅・発行幅の増加に伴い，高精度な分岐予測機構が求められている．分岐予測は，分岐の成立・不成立で遷移する 2bit 飽和カウンタ(予測カウンタ)によって予測を行う．予測カウンタの各状態(Strongly Taken, Weakly Taken, Weakly Not-taken, Strongly Not-taken)における予測精度を解析した結果，gshare 予測器の Weakly 状態における予測精度が特に低いことがわかった．本研究では，gshare 予測器の予測カウンタ状態を参照した予測器選択手法を提案する．SPECint95(train 入力)を対象にシミュレーションした結果，Combining 予測器と比較して，12KB のハードウェア容量で平均 0.22%，24KB で平均 0.3%予測ミス率が低減した．

A Branch Prediction Technique focused on Weak States of Prediction Table

Yukari Nakazawa[†], Fumiko Saito[†] and Hayato Yamana[‡]

In recent years, as the pipeline's length gets deeper, and the instruction fetch width and the issue width become wider, more accurate branch predictors are needed. Branch predictors predict with the 2 bit saturating counters (predictor counters) whose state is changed by the execution result of the branch. As a result of analyzing the prediction accuracy in each state (Strongly Taken, Weakly Taken, Weakly Not-taken and Strongly Not-taken) of prediction counters, it turns out that the prediction accuracy in the Weak states of gshare predictor is especially low. We propose the predictor selection technique referring the prediction counter state of gshare predictor. In SPECint95 (train) simulation, the branch prediction miss rate decreases 0.22% in average on 12KB, 0.30% in average on 24KB, compared with Combining predictor.

1．はじめに

プロセッサの性能向上を妨げる原因の一つに制御依存がある．制御依存を緩和するために様々な高精度の分岐予測器が提案されてきた．一方で，フェッチ幅や発行幅，パイプライン段数の増加に伴い，分岐予測ミスペナルティも増加している．そのため，さらに高精度の分岐予測器が求められている．

分岐予測器は，分岐の成立(Taken)・不成立(NotTaken)で遷移する 2bit 飽和カウンタのテーブル(PHT: Pattern History Table)で構成される．本稿では 2bit 飽和カウンタを予測カウンタと呼ぶ．分岐予測器は，1 つの PHT で構成される単体予測器と，複数の PHT で構成されるハイブリッド予測器に大別される．

予測を行う予測カウンタは，Strongly Taken, Weakly Taken , Weakly Not-taken, Strongly Not-takenの4状態を示す．単体予測器の各予測カウンタ状態における予測精度を解析した結果，Strongly TakenとStrongly Not-taken状態 (合わせてStrongly状態と呼ぶ) の場合には予測精度が高く， Weakly TakenとWeakly Not-taken状態 (合わせてWeakly状態と呼ぶ) の場合には予測精度が低いことがわかった．

本稿では，代表的なハイブリッド予測器であるCombining予測器[3]に着目した．Combining予測器は，2つの単体予測器と1つのSelectorから構成される．Selectorは予測の成功・失敗で遷移する2bit飽和カウンタのテーブルで，2つの単体予測器の予測信頼度を判定する．Combining予測器は，2つの単体予測器で正しく予測できる分岐命令が異なるほど予測精度が良い．また，Combining予測器のSelectorは，実際の予測を行うものではなく，予測を採用する単体予測器を選択する役割を

†早稲田大学大学院 理工学研究科
Graduate School of Science and Engineering, Waseda University
‡早稲田大学理工学術院
Science and Engineering, Waseda University

持つ.

本研究では，単体予測器の予測カウンタ状態を予測信頼度の判定に利用することで，Combining予測器のSelectorの削減を試みる．提案分岐予測器では，Selectorのような実際には予測を行わないテーブルが不要なため，Selectorに割り当てていたハードウェアを，実際に予測を行う単体予測器に割り当てることができる．

以降，2で分岐予測器の構成，3で実験環境を説明し，4で単体予測器の予測カウンタ状態の解析結果を示し，5で提案手法と実験結果について述べ，6で全体をまとめる．

2. 分岐予測器の構成

本節では，実験で用いた分岐予測器について説明する．

2.1 単体予測器

分岐予測器は，分岐の成立(Taken)･不成立(NotTaken)で遷移する 2bit 飽和カウンタ(図１)のテーブルPHT(Pattern History Table)から構成される．2bit 飽和カウンタは予測カウンタとも呼ばれ，カウンタの状態によって分岐方向を予測する．

単体予測器は，PHT のインデクス方法に基づいて，1 階層予測器と 2 階層予測器に分類される．1 階層予測器は，Bimodal 予測器[1]と呼ばれ，分岐命令アドレスの下位ビットをインデクスとしたPHTから構成される(図 2(a))．2 階層予測器は，1 階層目の表は過去数回の分岐履歴(BHT:Branch History Table)で構成され，2 階層目の表はその分岐履歴をインデクスとした PHT で構成される．Yeh らは BHT 及び PHT のインデクス方法の組み合わせから 9 種類の予測器を提案している[2]. その中から，後の実験で用いる SAg 予測器の構成を図 3 に示す．SAg 予測器の BHT は分岐命令アドレスの下位ビットでインデクスされる．

2 階層予測器の一種である gshare 予測器[3]は，1 階層目の表はグローバルな分岐履歴(BHR:Branch History Register)で構成され，2 階層目の表は単一の PHT で構成される(図 2(b))．PHT は，BHR と分岐命令アドレスの排他的論理和によりインデクスされる．

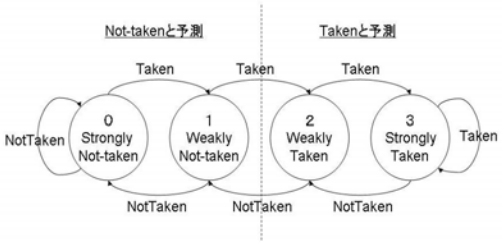
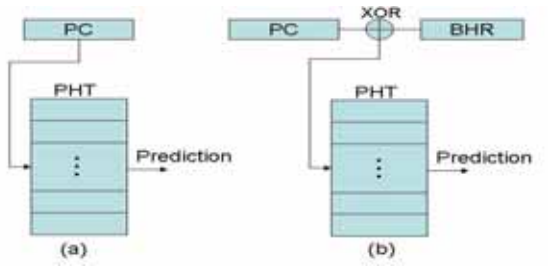


図１　2bit 飽和カウンタ



Bimodal 予測器[1]　　gshare 予測器[3]

図 2

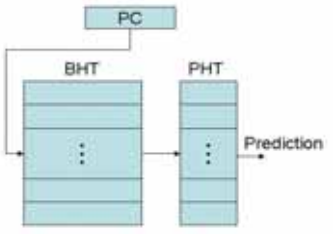


図 3　2 階層予測器 (SAg)[2]

2.2 ハイブリッド予測器

ハイブリッド予測器は，複数の PHT で構成される．代表的なハイブリッド予測器として，予測の信頼度によって PHT を選択する Combining 予測器[3]と，分岐の傾向によって PHT を選択する Bi-Mode 予測器[4]について説明する．

Combining 予測器は，2 つの単体予測器と 1 つの Selector で構成される(図 4)．予測の成功・失敗で遷移する 2bit 飽和カウンタのテーブルである Selector により，予測を採用する単体予測器を決定する．予測の採用・不採用に関係なく，常に予測器を更新する．

Bi-Mode 予測器は，gshare 予測器の破壊的競合を緩和することを目的として提案された. Bi-Mode 予測器は，ChoicePHT, TakenPHT, NotTakenPHT の 3 つの PHT から構成される．ChoicePHT は Bimodal 予測器，TakenPHT と NotTakenPHT は gshare 予測器で構成される．ChoicePHT が Taken と予測する場合は，TakenPHT における予測が採用され，ChoicePHT が NotTaken と予測する場合は，NotTakenPHT における予測が採用される．

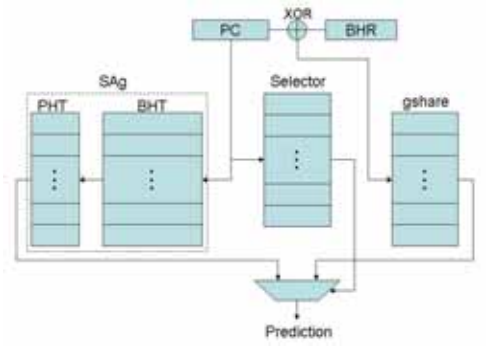


図 4 Combining 予測器(SAg と gsahre の組み合わせ)[3]

表 1 各ベンチマークの動的命令数と動的条件分岐命令数
(100 万命令)

プログラム	入力	全命令	条件分岐
099.go	50 9 2stone9.in	554	63
124.m88ksim	Ctl.in	114	14
126.gcc	Amptjp.i	1,280	194
129.compress	1000 q 2131	36	4
130.li	train.lsp	182	24
132.ijpeg	vigo.ppm	1,407	87
134.perl	jumble.pl	2,293	299
147.vortex	persons.250	2,608	287

3. 実験環境

シミュレータは，SimpleScalar 3.0d/PISA sim-bpred[6]，ベンチマークは SPECint95(train 入力)(表 1)を用いた．プログラムは gcc2.7.2.3-O2 –funroll-loops でコンパイルされている．

本実験では，BTB(Branch Target Buffer)のエントリ有無を参照した分岐予測手法[7]をすべての分岐予測器に適用した．この手法の適用により，BTB にエントリのない分岐は NotTaken と予測され，予測表の競合を削減できる．

4. 単体予測器の予測カウンタ状態の解析

分岐方向を予測する予測カウンタは，分岐の成立(Taken)，不成立(NotTaken)に応じて，Strongly Taken, Weakly Taken, Weakly Not-taken, Strongly Not-taken の 4 状態を遷移する(図 1)．Weakly 状態は前回の予測が失敗していることを示す．逆に，Strongly 状態は前回の予測が成功していることを示す．つまり，この 4 状態は予測方向を決定するだけでなく，予測の信頼度を示していると推察できる．そこで，単体予測器の予測カウンタの各状態における予測ミス率を調査した．

本稿では，8KB(2^{15}エントリ)の gshare 予測器と Bimodal予測器の結果を示す．なお，1.0KB(2^{12}エントリ)の場合も同様の傾向を示したので，1.0KBの結果は省略する．

Bimodal 予測器では，全予測ミスの約 62～75%が Strongly 状態で発生する．また Strongly 状態における予測ミス率は平均 11.24%, Weakly 状態における予測ミス率は平均 26.6%であった(全体の予測ミス率は平均 13.2%)(図 5).

gshare 予測器では，全予測ミスの約 35～45%が Weakly 状態で発生する．Strongly 状態における予測ミス率は平均 5.0%, Weakly 状態における予測ミス率は平均 38.02%であった(全体の予測ミス率は平均 7.36%)(図 6).

以上から，予測カウンタが Weakly 状態の場合には，予測ミスしやすいこと，つまり，予測の信頼度が低いことがわかった．特に gshare 予測器では，Strongly 状態と Weakly 状態の予測ミス率に平均 33.02%と大きな差がある．

5. 予測カウンタ状態を利用した分岐予測手法

第 4 節で示したように，gshare 予測器において，予測カウンタが Weakly 状態である時の予測信頼度が特に低いことがわかった．gshare 予測器は，グローバルの分岐履歴を利用した予測手法(2.1 節参照)であるが，グローバル履歴で予測困難な分岐は，ローカル履歴(各 PC の過去の分岐履歴)で予測できるものが多い[5]. そこで，gshare 予測器の予測カウンタが Weakly 状態の場合には，gshare 予測器の予測を採用せず，ローカル分岐履歴に基づく予測を行う．

5.1 提案手法

gshare 予測器の予測カウンタが Strongly (Strongly Taken/Strongly Not-taken)状態の場合には，gshare 予測器の予測をそのまま利用する(図 7). つまり，Strongly Taken 状態の時は「Taken(分岐する)」，Strongly Not-taken 状態の時は「NotTaken(分岐しない)」と予測する．

gshare 予測器の予測カウンタが Weakly (Weakly Taken / Weakly Not-taken)状態の場合には，各 PC のローカル履歴に基づく予測を採用する(図 7)．ローカル履歴に基づく予測には，2.1 節で説明した SAg 予測器を採用する．ただし，SAg 予測器の BHT は常時更新する

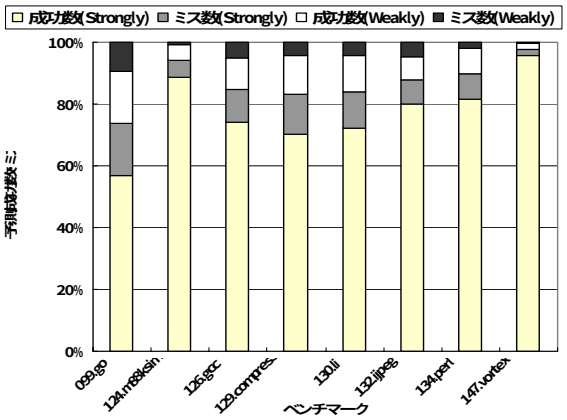


図5　Bimodal 予測器の Strongly/Weakly 状態における予測成功数・ミス数

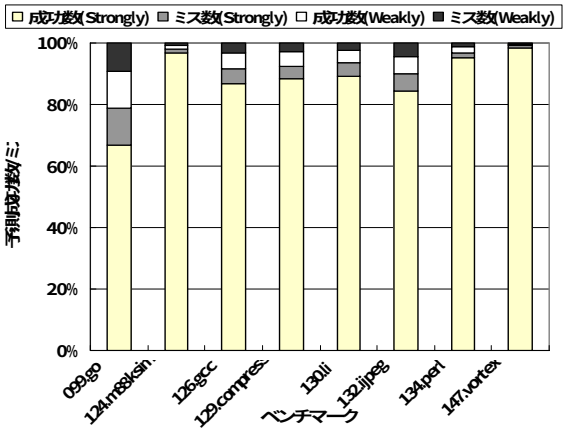


図6　gshare 予測器の Strongly/Weakly 状態における予測成功数・ミス数

が，PHT は予測が採用された時だけ更新する．また，gshare 予測器は常時更新する．提案手法の予測器構成を図 8 に示す．

提案手法では，gshare 予測器が，実際の予測を行う予測器と，予測を採用する予測器を選択する Selector の両方を兼ねる．そのため，Combining 予測器[3]の Selector を省け，Selector に割り当てていたハードウェア量を gshare 予測器や SAg 予測器に割り当てられる．

5.2 実験結果

ハードウェア容量 1.5KB，12KB，24KB を基準に評価を行った．比較対象の予測器は，gshare 予測器，Bi-Mode 予測器，SAg と gshare を組み合わせた Combining 予測器である．各予測器のエントリ数等のパラメータを表 2 に示す．

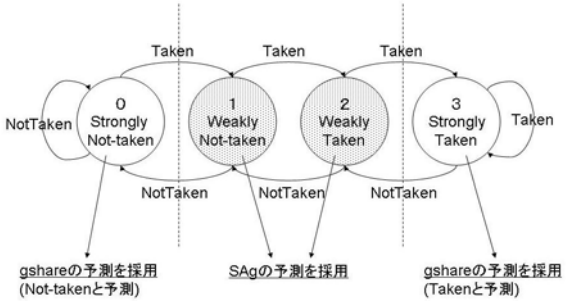


図 7　gshare の予測カウンタの各状態における予測採用方法

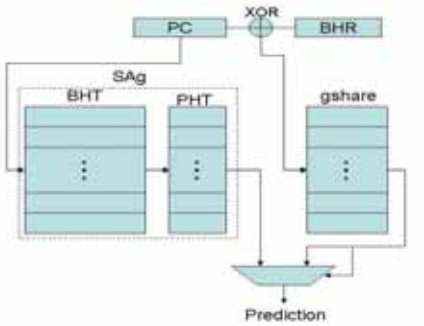


図 8　gshare 予測器の予測カウンタ状態を利用した予測器の構成

Combining 予測器では，テーブル数の多い SAg 予測器を利用するため，Selector の容量を小さめにした．Combining 予測器と提案手法に対しては，複数のパラメータの組み合わせの中から，基準のハードウェア量以下で，かつベンチマーク全体で一番精度の良いパラメータを選択した．12KB と 24KB のハードウェア量では，Combining 予測器は 11.75KB と 23KB，提案手法は 11.68KB と 23.5KB で比較を行った．次節以降では，これらの容量を便宜的に 12KB，24KB と表現する．

5.2.1 予測精度

各予測器の予測ミス率を表 3 に示す．提案手法は，Combining 予測器と比較して，1.5KB では平均 0.05%，12KB では平均 0.22%，24KB では平均 0.30%予測ミス率が低減した．

129.compress と 124.m88ksim は，全容量で Combining 予測器よりも予測ミス率が増加した．これらのベンチマークでは，gshare 予測器が Strongly 状態の場合でも，SAg 予測器の予測を採用した方が，予測が成功する分岐が多いと考えられる．

表 2 各予測器のパラメータ(PHT，BHT エントリ数)

予測器	容量(KB)	パラメータ
Gshare	1.0	PHT 1K
	8.0	PHT 32K
BiMode	1.5	Choice/Taken/NotTakenPHT 各 2K
	12	Choice/Taken/NotTakenPHT 各 16K
	24	Choice/Taken/NotTakenPHT 各 32K
Combining	1.5	SAg: PHT 1K, BHT 0.5K*8bit gsahre: 2K, Selector: 1K
	11.75	SAg: PHT16K, BHT 1K*14bit gshare: 16K, Selector: 8K
	23	SAg: PHT16K, BHT 4K*14bit gshare: 32K, Selector: 16K
提案手法	1.5	SAg: PHT1K, BHT 0.5K*4bit gshare: 4K
	11.68	SAg: PHT8K, BHT 1K*13bit gshare: 32K
	23.50	SAg: PHT16K, BHT 2K*14bit gshare: 64K

099.go と 126.gcc の予測ミス率は，Combining 予測器と比較すると低減したが, Bi-Mode 予測器と比較すると増加した(23.5KB の 099.go を除く)．これらのベンチマークは，gshare 予測器におけるエントリ競合が激しいため，分岐の偏向を利用した Bi-Mode 予測器の方が予測精度が高いことが知られている[4].

Combining 予測器と比較して予測ミス率が増加した 129.compress と 128.m88ksim は，Bi-Mode 予測器と比較すると予測ミス率は低減した.

5.2.2 Selector の予測精度への影響

Combining 予測器の Selector と，提案手法の予測器選択手法の精度を比較する．Combining 予測器と提案手法で，SAg 予測器と gshare 予測器の各パラメータを同一に設定する．Combining 予測器は，さらに数 KB の Selector を追加して予測を行い，提案手法は，追加のハードウェアなしで gshare 予測器の予測カウンタ状態に基づいて予測を行う.

SAg 予測器と gshare 予測器のパラメータを①11.75KB と②23KB の Combining 予測器と同一(表 2 参照)に設定した．提案手法で必要なハードウェア量は Selector 部分を除いた①9.75KB と②19KB となる.

Combining 予測器と提案手法の予測ミス率を表 4 に示す．提案手法の予測ミス率は，Combining 予測器と比較して，①では平均 0.18%，②では平均 0.14%増加した．しかし，これは比較しているハードウェア量が異なることに注意する必要がある．Combining 予測器とハードウェア量を揃えた場合には，①で平均 0.22%，②で 0.3%予測ミス率は低下する(5.2.1 節参照)．Combining 予測器の Selector を取り除いただけの構成で提案手法を適用しても予測精度は低下するが，Selector に割り当てていたハードウェアを SAg 予測器や gshare 予測器に追加することで，Combining 予測器よりも予測精度を向上できる.

099.go，130.li，134.perl は，①②どちらの場合でも平均 0.16%予測ミス率が低減した．これらのベンチマークでは，Combining 予測器の Selector を取り除き，提案手法を適用するだけで予測精度が向上する．つまり，予測精度を低下させずに必要なハードウェア量を削減できる.

6. まとめ

分岐予測器は，分岐の成立(Taken)，不成立(NotTake によって遷移する2bit飽和カウンタ(予測カウンタ)を用いて予測を行う．予測カウンタの各状態(Strongly Taken, Weakly Taken, Weakly Not-taken, Strongly Not-taken)における予測精度を解析した結果，gshare 予測器のWeakly状態における予測精度が特に低いことがわかった.

本稿では, gshare予測器の予測カウンタ状態に基づいて，予測を採用する予測器を選択する手法を提案した．Gshare予測器を実際の予測を行う予測器としてだけでなく，Selectorとしても利用することで，従来の Combining予測器のSelectorを削減できる.

本稿では，SPECint95(train)を用いてSimpleScalar 3.0d/PISA sim-bpredシミュレータで実験を行った．11.68KBの本手法では，11.75KBのCombining予測器に対して平均0.22%予測ミス率が低減した．また，23.5KB の本手法では，23KBのCombining予測器に対して0.3%，予測ミス率が低減した.

謝辞

本研究の一部は，日本学術振興会21世紀COEプログラム「プロダクティブICTアカデミア」の支援により行われた.

表 3　各予測器の予測ミス率(%)

	099.go	124.m88ksim	126.gcc	129.compress	130.li	132.ijpeg	134.perl	147.vortex	平均
1.0KB gshare	27.22	2.03	10.28	8.06	6.08	10.5	3.74	3.94	8.98
1.5KB Bi-Mode	23.40	2.06	8.40	8.00	6.14	9.91	3.38	0.82	7.76
1.5KB Comb*	25.49	1.71	10.46	5.14	6.03	9.52	3.15	0.92	7.80
1.5KB 提案手法	24.66	1.76	9.06	6.90	5.82	9.72	3.32	0.73	7.75
8.0KB gshare	19.78	1.67	6.89	7.17	5.11	9.39	2.31	0.63	6.62
12KB Bi-Mode	17.65	1.67	5.97	7.51	5.17	9.23	2.34	0.59	6.27
11.75KB Comb*	20.21	1.07	6.68	4.42	4.54	8.76	2.40	0.57	6.08
11.68KB 提案手法	17.93	1.32	6.07	5.73	4.29	8.75	2.25	0.53	5.86
16.0KB gshare	16.86	1.65	6.22	6.66	4.93	9.18	2.08	0.59	6.02
24KB Bi-Mode	15.36	1.61	5.46	7.08	5.01	9.01	2.21	0.55	5.79
23KB Comb*	17.68	0.96	5.81	4.41	4.45	8.64	2.30	0.49	5.59
23.5KB 提案手法	15.15	1.18	5.47	5.36	3.99	8.58	2.05	0.49	5.29

*Comb = Combining 予測器

表 4　SAg 予測器と gshare 予測器のパラメータ(容量)を同じにした場合の予測ミス率(%)

	099.go	124.m88ksim	126.gcc	129.compress	130.li	132.ijpeg	134.perl	147.vortex	平均
①11.75KB Comb	20.21	1.07	6.68	4.42	4.54	8.76	2.40	0.57	6.08
①9.75KB 提案手法	19.94	1.37	6.7	5.89	4.31	8.95	2.38	0.57	6.26
②23KB Comb	17.68	0.96	5.81	4.41	4.45	8.64	2.30	0.49	5.59
②19KB 提案手法	17.35	1.3	5.93	5.62	4.18	8.75	2.26	0.48	5.73

*Comb = Combining 予測器

参考文献

[1] Smith J. E.,"A Study of Branch Prediction Strategies", Proc. of 8th ISCA, pp. 135-148, 1981

[2] Yeh T. Y. and Patt Y. N.,"A Comparison of Dynamic Branch Predictors that use Two Levels of Branch History", Proc. of 20th ISCA, pp. 257-266, 1993

[3] McFarling S., "Combining branch predictors", Technical Report TN-36, Digital Western Research Laboratory, 1993

[4] Lee C. C., Chen I. K. and Mudge T. N., "The Bi-Mode Branch Predictor", Proc. of MICRO-30, pp. 4-13, 1997

[5] M. Evers, S. J. Patel, R. S. Chappel, Y. N. Patt, "An Analysis of Correlation and Predictability: What Makes Two-Level Branch Predictors Work", Proc. of 25th ISCA, pp. 52-61, 1998

[6] Burger D. and Austin T. M., "The SimpleScalar Tool Set, Version2.0", Technical report, 1997

[7] 斎藤史子, 山名早人, "BTB のエントリ有無を参照した分岐予測器", 情報処理学会 ACS 論文誌 Vol.45, No. SGI 11(ACS 7), pp. 71-79, 2004