De-aliased Hybrid Branch Predictors

André Seznec, Pierre Michaud

N° 3618

Février 1999

THÈME 1

ISSN 0249-6399

De-aliased Hybrid Branch Predictors

André Seznec, Pierre Michaud

Thème 1 — Réseaux et systèmes
Projet CAPS

Rapport de recherche n3618 — Février 1999 — 23 pages

Abstract: Fixed-size branch predictors tables suffer from a loss of prediction accuracy due to *aliasing* or *interference.* This is particularly true for predictors using a global history vector such as *gshare.* “De-aliased” global history predictors -the skewed branch predictor, the bimode predictor and the agree predictor- were recently proposed. “De-aliased” predictors consistently achieve the same prediction accuracy level as *gshare* or *gselect* using less than half the transistor budget.

However different branches do not require the use of the same vector of information to be accurately predicted. Hybrid predictors combining several branch prediction schemes may deliver higher branch prediction accuracy than a branch predictor using a single branch prediction scheme. Then “de-aliased” branch are natural candidates as hybrid predictors components.

In this paper, we show how cost-effective hybrid branch predictors can be derived from the enhanced skewed branch predictor *e-gskew.* *2Bc-gskew* combines *e-gskew* and a bimodal branch predictor. It consists in four identical *predictor-table banks*, i.e., the three banks from the *e-gskew* -including a bimodal bank- plus a meta-predictor. *2Bc-gskew-pskew* combines a bimodal component, a global history register component and a per-address history component. These hybrid predictors are shown to achieve high prediction accuracy at a low hardware cost.

Key-words: “de-aliased” branch predictor, hybrid branch predictor

(Résumé : tsvp)

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : 02 99 84 71 00 - International : +33 2 99 84 71 00
Télécopie : 02 99 84 71 71 - International : +33 2 99 84 71 71

Prédicteurs de branchements hybrides et désaliasés

Résumé : Les tables de prédiction de branchement souffrent d'une perte de précision du fait des interférences ou "aliasing". Ceci est particulièrement vrai pour les prédicteurs utilisant un vecteur d'historique global. Des prédicteurs "désaliasés" - le "skewed branch" predicteur *e-gskew*, le predicteur bimode et le prédicteur "agree" - ont été proposés récemment. Ces prédicteurs ont la même précision que les prédicteurs classiques à historique global *gshare* ou *gselect*, mais utilisent 2 fois moins de silicium.

Cependant les prédicteurs hybrides utilisant plusieurs vecteurs d'information réalisent une meilleure précision que les prédicteurs à historique global. Les prédicteurs "désaliasés" sont des candidats naturels pour être intégrés au sein d'un prédicteur hybride.

Dans ce rapport, nous montrons comment des prédicteurs de branchements hybrides à coûts d'implémentation réduits peuvent être dérivés du prédicteur "désaliasé" *e-gskew*.

Mots-clé : prédicteur de branchement désaliasé, prédicteur de branchement hybride

1 Introduction

Accurate branch prediction has come to play a major role in microprocessors. Many dynamic branch prediction schemes have been investigated in the past few years, with each offering certain distinctive features [16, 8, 20, 13, 9, 12]. However, fixed-sized predictor tables lead to a phenomenon known as *aliasing* or *interference* [23, 18], in which multiple branch information vectors share the same entry in the predictor table, causing the predictions for two or more branch substreams to intermingle. Aliasing has been shown to seriously impair performance [23, 18], particularly large or multi-process workloads with a strong OS component exhibit very high degrees of aliasing [14, 5]. Therefore, removing aliasing effects has recently received a lot of attention.

"De-aliased" global history branch predictors have been recently introduced: the enhanced skewed branch predictor *e-gskew* [11], the agree predictor [17], the bimode predictor [7] and the YAGS predictor [3]. These predictors have been shown to achieve higher prediction accuracy at equivalent hardware complexity than larger "aliased" global history branch predictors such as *gshare* [9] or GAs [21].

However different branches do not require the use of the same vector of informations to be accurately predicted [22, 10]. Hybrid predictors combining several branch prediction schemes (for instance a *gshare* predictor and the classic bimodal 2-bit counter table) may deliver higher prediction accuracy than a conventional single branch predictor [9]. Therefore, "de-aliased" branch predictors should be included in hybrid predictors to build efficient branch predictors.

In this paper, we show how cost-effective hybrid branch predictors can be derived from the enhanced skewed branch predictor *e-gskew* proposed in [11]. *e-gskew* is constructed from three *predictor-table banks*, each of which functions like a standard predictor table. *e-gskew* offers an opportunity to design cost-effective hybrid predictors using a bimodal component: the bimodal table is already present[1] . We propose *2Bc-gskew*, a hybrid branch predictor combining *e-gskew* and a bimodal branch predictor. *2Bc-gskew* consists in four identical *predictor-table banks*, that is the three banks from the *e-gskew* plus a meta-predictor.

Some branches are better predicted using a per-address history branch predictor than a global history branch predictor. De-aliasing principles can also be applied to per-address history predictors. We define *e-pskew*, a "de-aliased" per-address history branch predictor. While *e-pskew* is less cost-effective than *e-gskew*, it can be combined with *2Bc-gskew* in *2Bc-gskew-pskew*. *2Bc-gskew-pskew* is a 3-component hybrid predictor combining a bimodal component, a per-address history component and a global history component.

The remainder of the paper is organized as follows. In Section 2, we first recall previous work on hybrid predictors and on de-aliasing techniques. Section 3 introduces our experimental methodology. Section 4 introduces and analyzes *2Bc-gskew*, an hybrid branch predictor combining *e-gskew* and a bimodal 2-bit counter predictor. *2Bc-gskew* is shown to be very cost-effective as it outperforms larger *e-gskew* predictors. In Section 5, we show how the principles used for *2Bc-gskew* may be used to design a cost-effective hybrid predictor

[1]The bimode predictor [7] offers the same opportunity.

combining a per-address history component, a bimodal branch component and global history branch predictors. Section 6 summarizes this study.

2 Previous work

2.1 Hybrid predictors

Various vectors of information can be used as input of branch predictors. The branch address was first proposed for indexing branch prediction tables [16]. This predictor is often called the bimodal predictor. More recently, it has been shown that the outcome of a conditional branch is also very correlated to the outcomes of the few past branches [20]. Two families of branch predictors exploiting this correlation are generally distinguished. In global history branch predictors, a single history register records the outputs of the previous branches. On per-address history branch predictors, a history vector is associated with each individual branch.

However, some branches are better predicted by per-address schemes than with global schemes (and reciprocally). This has lead to the introduction of hybrid predictors [9]. An hybrid predictor consists of X branch predictors plus a metapredictor. Each of the X branch predictor components (either static or dynamic [2]) produces a prediction and the metapredictor chooses the prediction which will be used. Ideally a hybrid branch predictor would benefit from the qualities of all its components.

The meta-predictor might be built with the same kind of hardware as branch predictors. For instance, McFarling [9] proposed to use a table of 2-bit saturating counters indexed with the branch address. Other index functions may also be used, for instance the global branch history as on the DEC 21264 [6]. Various hybrid predictors have been studied and proposed. It has been shown [9, 2, 4] that, provided sufficient hardware resources, hybrid predictors outperform single predictors.

2.2 "De-aliased" predictors

The accuracy of fixed-sized branch predictor tables suffers from a phenomenon known as *aliasing* or *interference* [23, 18]. Multiple branch information vectors share the same entry in the predictor table, causing the predictions for two or more branch substreams to intermingle. Aliasing has been shown to seriously impair performance [23, 14, 5].

2.2.1 Aliasing classifications

Two classifications of aliasing have been proposed. Young et al. [23] proposed to classify aliasing instances with respect to their consequences: destructive, harmless or constructive. They showed that destructive and harmless aliasing constitute the overwhelming majority of aliasing instances. Michaud et al. [11] proposed to classify aliasing with respect to their origin, i.e using the same 3C's classification as for cache misses: compulsory, conflict or capacity. Conflict aliasing was shown to be the origin of the majority of aliasing instances.

These two classifications lead to two approaches to limit aliasing impact on branch prediction accuracy.

2.2.2 Reducing 3 C's aliasing

The Filter mechanism A mean to reduce capacity aliasing is to reduce the amount of information that must be stored in the predictor table(s) [1]. Highly biased branches can be filtered out from the PHT using a saturating counter and a bias bit for each entry in the BTB (or in a tagless table). The saturating counter is incremented whenever the branch outcome equals the bias bit and else zeroed. If the counter is saturated then the predictor table(s) are not used for the prediction.

It should be noted that the filter mechanism can be used to improve all predictors that will be discussed in the remainder of the paper.

The Enhanced Skewed Branch Predictor The skewed branch predictor *gskew* and the enhanced skewed branch predictor *e-gskew* were introduced in [11].

The basic principle of the skewed branch predictor *gskew* is to use three branch-predictor banks, but to index them by different and independent hashing functions computed from the same vector V of information (e.g., branch address and global history). A prediction is read from each of the banks and a majority vote is used to select a final branch direction.

The rationale for using different hashing functions for each bank is that two vectors, V and W, that are aliased with each other in one bank are unlikely to be aliased in the other banks.. A destructive aliasing of V by W may occur in one bank, but the overall prediction on V is likely to be correct thus removing conflict aliasing impact. An example of such a family of independent hashing functions was taken from [15].

In [11], the enhanced skewed branch predictor *e-gskew* (Figure 1) was further proposed. At equivalent hardware costs *gskew* suffers from more capacity aliasing than *gshare*. In order to limit the impact of capacity aliasing, one of the table in *e-gskew* is indexed using only the address of the branch instead of the combination (global history register-branch address). Experiments showed that *e-gskew* performs equally to *gskew*, when few conflict aliasing were encountered (i.e, for a small history length). But *e-gskew* performs better than *gskew* for long history length (i.e, when capacity aliasing becomes significant). A 3*4K-entry *e-gskew* was shown to outperform a 32 K-entry *gshare*.

2.2.3 Reducing destructive aliasing

The *agree* predictor [17] and the *bimode* predictor [7] aims at decreasing the number of destructive aliasing instances and replacing them by harmless aliasing. For instance, as most branches are biased through a dominant direction, the *agree* predictor records the agreement or disagreement with a bias stored in the BTB (or encoded in the instruction as in the HP 8500). In the bimode predictor, the rationale is to store the predictions associated with biased not-taken branches and with biased taken branches in two distinct predictor tables, the prediction is selected at execution time using a bimodal table.

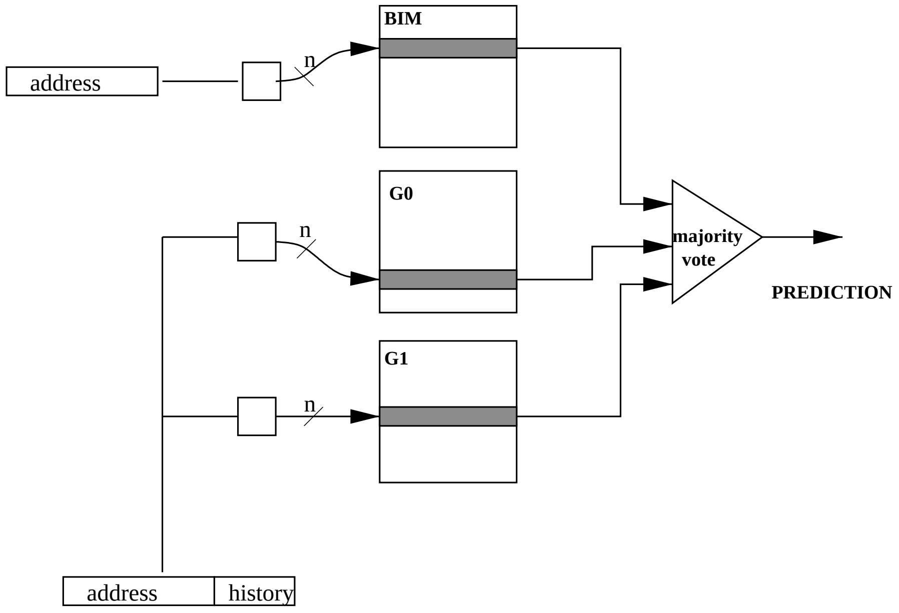


Figure 1: The Enhanced Skewed Branch Predictor

2.2.4 Reducing both destructive aliasing and 3 C's aliasing

Hybrid predictors featuring a bimodal table Some hybrid predictors include a bimodal table (for instance *2Bc-gshare* proposed in [9]). There are two main reasons for using a bimodal predictor component [9, 4].

First, on a two-level branch predictor, an entry has to be associated with each (branch, history) pair. Therefore, a two-level branch predictor requires much more training than a bimodal predictor, i.e suffers from higher compulsory and capacity aliasing. Then, for instance on a context switch incurring a major change in the control flow, a bimodal predictor will be more accurate than a two-level branch predictor on the first thousands (may be hundred of thousands) of branches [5].

At second, on biased branches, the bimodal table is able to deliver the correct prediction using a single entry. This table suffers from less destructive aliasing than a two-level branch predictor.

The YAGS predictor Recently, Eden and Mudge [3] proposed the YAGS predictor to combine benefits from removing conflict aliasing and decreasing the number of destructive aliasing instances. As in the bimode predictor, a bimodal table is used for indicating the bias of a branch. Instances of the branch that do not obey to this bias are recorded in two tagged tables called direction caches. When the bimodal table predicts taken (resp not-taken), the

benchmark	**conditional branch count**	
	dynamic	**static**
groff	11568181	5634
gs	14288742	10935
mpeg_play	8109029	4752
nroff	21368201	4480
real_gcc	13940672	16716
sdet	5221320	4583
verilog	5692823	3918
video_play	5175629	3977

Table 1: Conditional branch counts

not-taken (resp. *taken*) direction cache is searched. On a miss in the searched direction cache, the bimodal table provides the prediction. On a hit, the direction cache provides the prediction.

Only a few bits of the complete tag are recorded thus limiting the storage overhead associated with the tags: 6 bits were shown to provide a good trade-off. The YAGS predictor was shown to outperform other "de-aliased" predictors at equivalent hardware cost [3][2].

It should be noted that the YAGS predictor may be considered as a hybrid predictor. The presence or the absence of an instance in the searched direction cache is used as an implicit choice predictor.

3 Experimental Setup

Propositions of branch predictors need to be validated through trace-driven simulations. We conducted all of our trace-driven simulations using the IBS-Ultrix benchmarks [19]. These benchmarks were traced using a hardware monitor connected to a MIPS-based DECstation running Ultrix 3.1. The resulting traces include activity from all user-level processes as well as the operating-system kernel, and have been determined to be a good test of branch-prediction performance [5, 14]. Conditional branch counts[3] derived from these traces are given in Table 1.

The global history register is updated only when a conditional branch is encountered [4].

[2]more precisely *gskew* and *bimode*, our own experiments confirm that it also outperforms *e-gskew*

[3]**beq r0,r0** is used as an unconditional relative jump by the MIPS compiler, therefore we did not consider it as conditional. This explains the discrepancy with the branch counts reported in [5, 14]

[4]Recording all branch directions including unconditional branches was found to lead to higher misprediction rate on the IBS benchmarks.

4 2Bc-gskew

In the previous section, we have recalled that hybrid predictors may deliver a higher prediction accuracy than a single component predictor. *e-gskew* is a very efficient single component branch predictor [11, 7] and therefore a natural candidate as a component for a hybrid predictor. Moreover *e-gskew* includes three predictor tables, and one of them is a bimodal predictor, this bimodal predictor may be used directly.

We first present the principles of the *2Bc-gkew* predictor. Then we present some simulation results illustrating its advantage over previously proposed predictors.

4.1 Basic principle

The hybrid predictor *2Bc-gskew* is illustrated in Figure 2. It combines *e-gskew* and a bimodal predictor. *2Bc-gskew* consists of four 2-bit counters banks.

Bank **BIM** is the bimodal predictor, but is also part of the *e-gskew* predictor. Banks **G0** and **G1** are the two other banks of the *e-gskew* predictor. Bank **Meta** is the meta-predictor.

Compared with the usual design of a hybrid predictor, the bimodal predictor is provided at no hardware cost.

For sake of simplicity, we will consider in the remainder of the paper that the four predictor banks have the same size. However, experiments showed that assuming half size **BIM** and/or **Meta** predictor bank could be also be considered.

Indexing the Meta predictor We perform various simulations in order to determine which information should be used to index the meta-predictor. For small predictor sizes (i.e. $4 * 1K$ entries), the branch address is the best vector of information we found. For medium and large predictor sizes (i.e $4 * 4K$ entries or larger), combining global history and branch address was found to be more efficient[5]. Simulation results reported in this paper assume this latter case.

In fact, branch address + history register would be a better vector of information for indexing the **Meta** bank if infinite predictor tables were used. But a finite bank *Meta* is also experiencing aliasing, in particular conflict aliasing.

Partial updating On a multi-table branch predictor, the update policy has an impact on the prediction accuracy. Total update policy and partial update policy were considered in [11]. Partial update policy was shown to result in a higher prediction accuracy than total update policy for *e-gskew*.

Our simulations showed that applying partial update policy is also worthwhile on *2Bc-gskew*. The different partial update policies we tested on *2Bc-gskew* were roughly equivalent. The partial update policy that was used for the simulation results reported in this paper is described below:

[5]Banks **Meta**, **G0** and **G1** are indexed with three different hashing functions

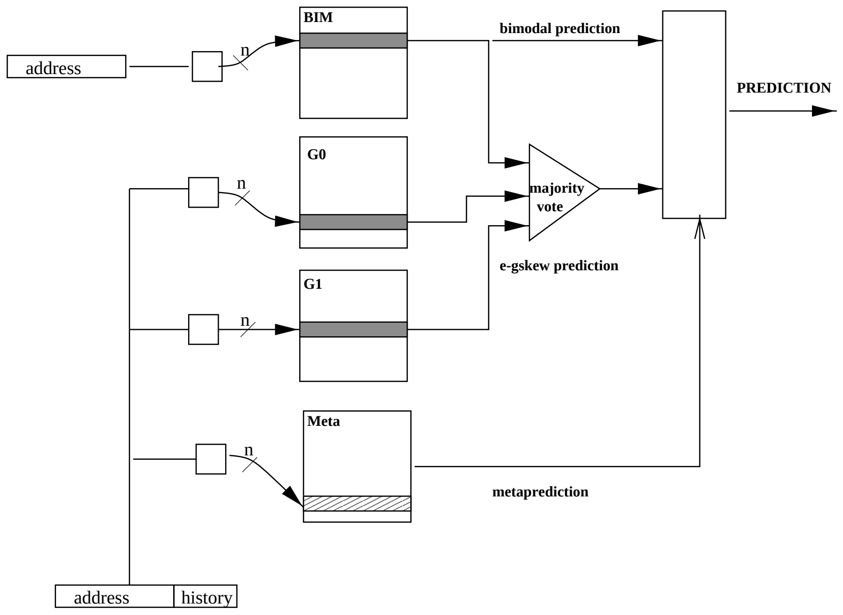


Figure 2: The *2Bc-gskew* predictor

- On a bad prediction, the three banks of the *e-gskew* predictor are updated.
- On a correct prediction, only the banks participating to the correct prediction are updated. That is, if the selected predictor was the bimodal predictor, only the bimodal predictor bank is updated, if the selected predictor was *e-gskew*, only the banks which gave the correct prediction are updated.
- The metapredictor bank **Meta** is updated only when the two predictors disagree.

2Bc-bimode predictor The *bimode* predictor also features an already included bimodal table and also consists of three identical predictor banks. Deriving the the *2Bc-bimode* predictor from the *bimode* predictor can be done in the same way as above for *2Bc-gskew*.

However, on our benchmark set, we found that *2Bc-bimode* is generally less effective than *2Bc-gskew*. We analyze this result as follows: the combined use of a choice predictor and partial update policy removes a significant fraction of the write accesses on the two predictor banks accessed using both address and history register as index function. This allows to remove a significant part of the aliasing on these two banks. This is true for both *2Bc-bimode* and *2Bc-gskew*. However, removing aliasing on *2Bc-gskew* is removing destructive aliasing in half of the cases while this proportion is significantly lower on *2Bc-bimode*.

4.2 Experimental results

We present trace-driven simulation results to illustrate the performance of the *2Bc-gskew* predictor. We first compare the branch prediction accuracy of *2Bc-gskew* with some other branch predictors on the complete IBS traces. Then, we show that *2Bc-gskew* and other hybrid predictors featuring a bimodal component requires a shorter training interval than conventional single component predictors.

4.2.1 Prediction accuracy

Simulations were conducted in order to measure the prediction accuracy of *2Bc-gskew*. Other global history branch predictors were also simulated.

Simulated configurations We illustrate here *gshare*, *e-gskew*, *2Bc-gshare* and the YAGS predictor. *gshare* is generally considered as the most effective "aliased" single component branch predictor. *e-gskew* is one of the most effective "de-aliased" branch predictor [11, 7]. *2Bc-gshare* is the original hybrid predictor proposed by McFarling [9]. A $(N + 2N + N)$-entry *2Bc-gshare* predictor consists in a N-entry bimodal predictor, a $2 * N$-entry *gshare* predictor and a N-entry meta-predictor[6]. This predictor was shown to be quite effective compared with *gshare* [9]. Finally, we also simulate the YAGS predictor. The configuration we simulated is as follows: a N-entry bimodal table, and two $\frac{N}{2}$ direction caches with 6-bit tags, featuring a total storage cost of $10N$ bits. This configuration was shown to represent the best trade-off for the YAGS predictor [7]. We will refer to this configuration as the $(N + 2 * \frac{N}{2})$-entry YAGS predictor.

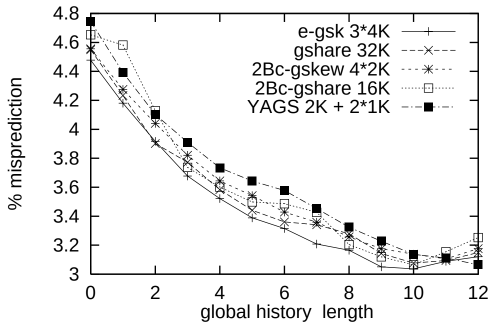
Experimental results We consistantly observed that a $4 * N$-entry *2Bc-gskew* predictor offers a prediction accuracy in the same range as a $(N + 2 * \frac{N}{2})$-entry YAGS predictor, a $3 * 2N$-entry *e-gskew*, a $16 * N$-entry *gshare* predictor or a $(2N + 4N + 2N)$-entry *2Bc-gshare* predictor, and this on all our benchmark set, and for various history lengths. That is *2Bc-gskew* offers equivalent prediction accuracy as YAGS for four fifths of the hardware cost, or as *e-gskew*, for only two thirds of the hardware cost, or as *2Bc-gshare* for one half of the hardware cost, or as *gshare* for one fourth of the hardware cost. Figures 3 and 4 illustrates this phenomenon for *2Bc-gskew* predictors with respectively 4*2K entries and 4*8K entries.

4.2.2 Considering context switches

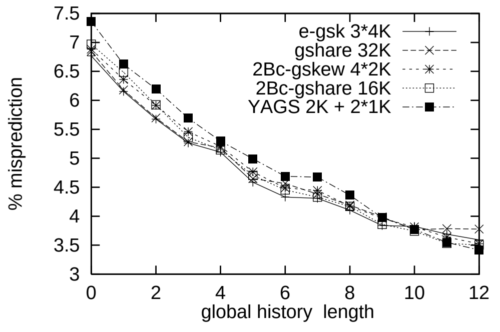
On a time shared workload running on a workstation or a PC, several user applications are competing for the CPU. Switching from a user application to an other incurs a major change in the branches which are executed. When such a major context switch occurs, the branch

[6] The choice of table sizes is certainly not optimal; however the purpose of this paper is not to explore the trade-off between the components in *2Bc-gshare*

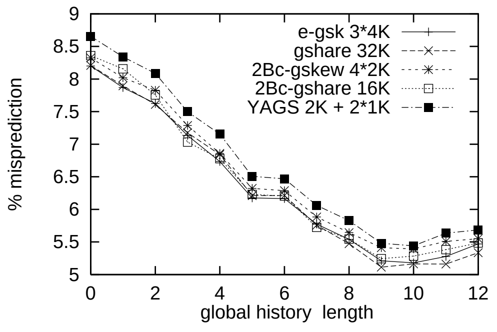
[7] In the original YAGS predictor paper, the length of the history register was arbitrary fixed to the log_2 of the number of entries in the direction cache. We removed this constraint by indexing the direction cache with one the hashing function described in [11]



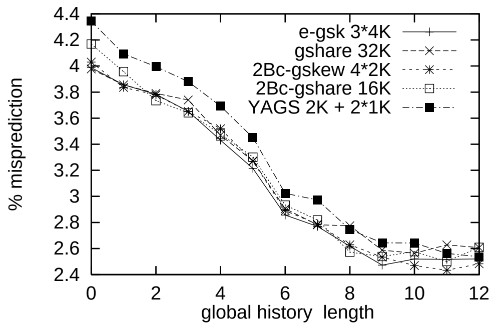
groff



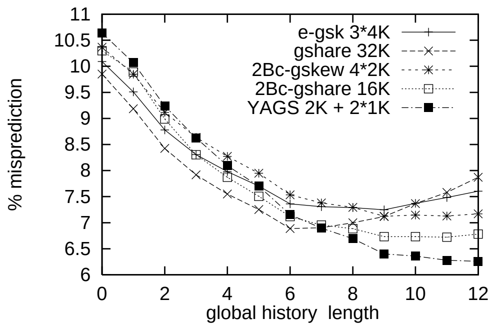
gs



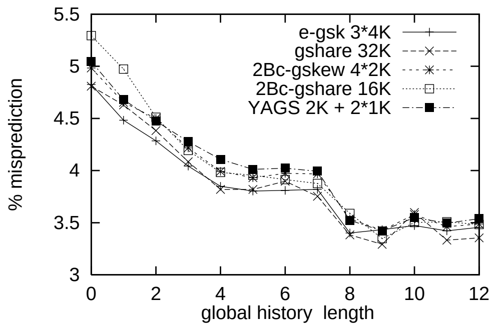
mpeg play



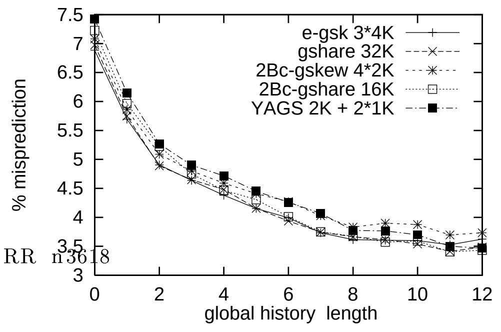
nroff



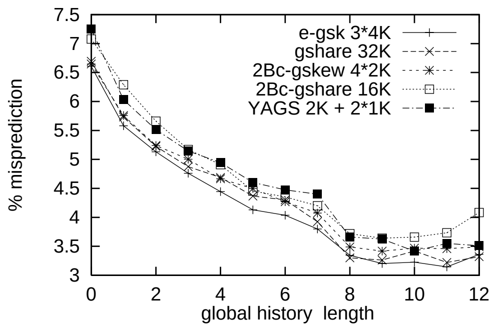
real gcc



sdet



verilog



video play

Figure 3: Misprediction rate of $4 * 2K$-entry 2Bc-gskew predictor

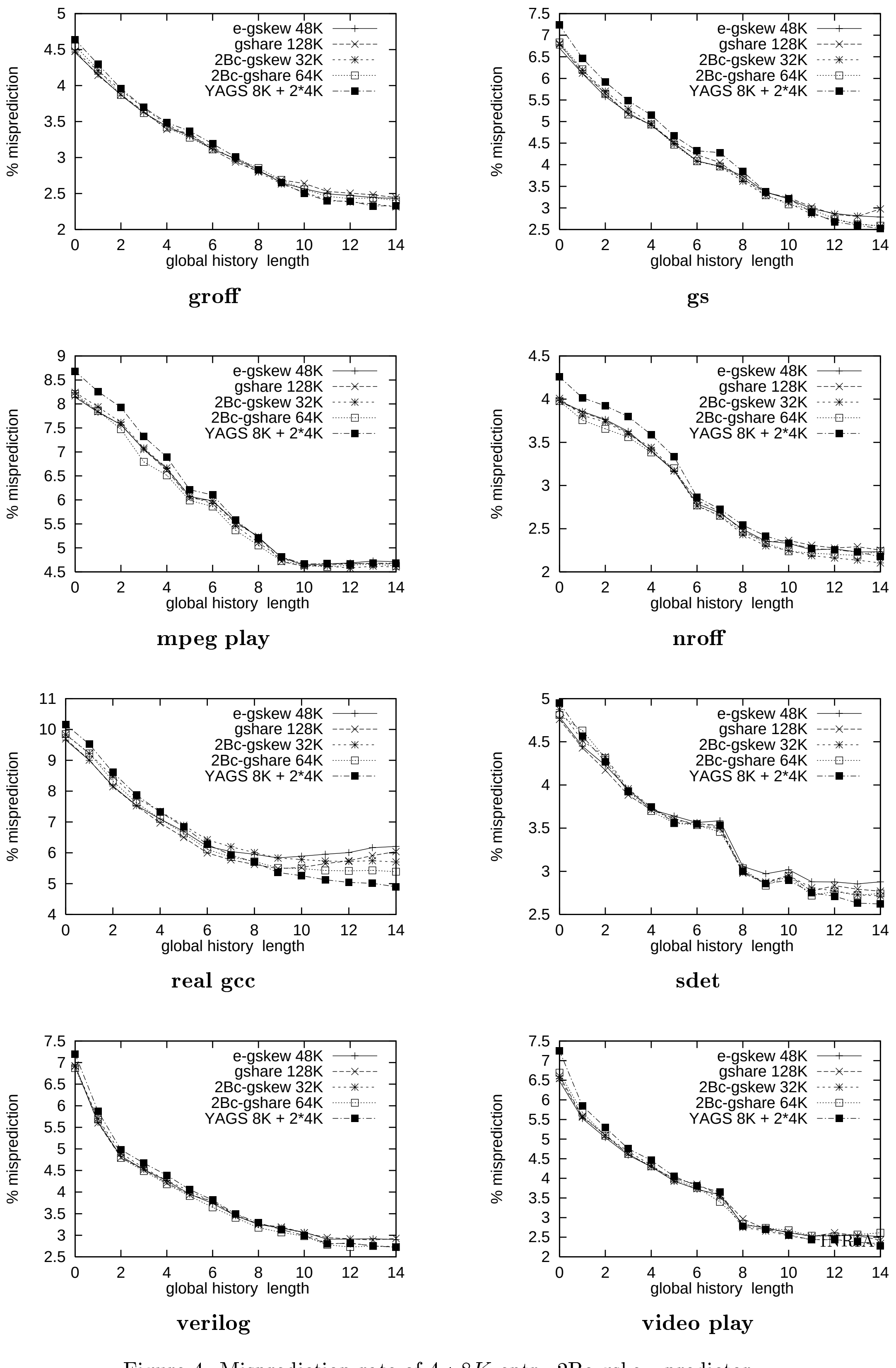


Figure 4: Misprediction rate of $4 * 8K$-entry 2Bc-gskew predictor

predictor has to be trained again before delivering its full potential prediction accuracy: it suffers from compulsory aliasing [11]. In Section 2, we have recalled that hybrid predictors featuring a bimodal component will benefit from a better prediction accuracy during training than single component global history branch predictor [4]. In this section, we intend to show this phenomenon for *2Bc-gskew*.

Experimental set-up IBS traces [19] include both user and operating systems instructions and include some context switches. However, they were collected while a single user application was running.

We did not have access to any realistic traces of time-shared workloads including both user and operating systems instructions. Then we chose to artificially mimic the loss of prediction information on major context switches as follows: the information loss was simulated through a random initialization of all the predictor entries every Q branches[8]. Q was varied from $50,000$ to $500,000$. This does not precisely characterize the behavior of branch predictors on context switches, but is sufficient to compare their training times.

Experimental results Figure 5 illustrates this experience for a $4 * 8K$-entry *2Bc-gskew* predictor assuming an history length of 10. On this figure, we also plot the corresponding simulation results for the same predictors as previously.

As expected, the bimodal components embedded in *2Bc-gskew* and *2Bc-gshare* allow them to reach a high prediction accuracy faster than *gshare*: *2Bc-gskew* fastly reaches the same prediction accuracy as its bimodal component. At the same time the embedded *e-gskew* predictor is trained. On the other hand, on *gshare*, each individual (address,history) pair must train back its target 2-bit counter.

e-gskew also requires a longer training interval than *2Bc-gshare* and *2Bc-gskew* but smaller than *gshare*. *e-gskew* is not an hybrid predictor - there is no meta-predictor -. However its included bimodal table is used to make the decision when the two other banks disagree. Such a situation appears quite often during the training phase after a major context switch.

Our simulation results also point out that the YAGS predictor is particularly effective on context switches. On the first occurence of a couple (branch, history) who does not complain with the bimodal prediction, the corresponding entry in the direction cache is set to weakly taken or weakly not-taken depending on the branch outcome. Therefore, on a misprediction associated with a compulsory aliasing encountered after a context switch, the inertia of the direction cache is only a single access.

4.2.3 2Bc-gskew versus e-gskew

The advantage of *2Bc-gskew* over *e-gskew* can be explained as follows. Depending on the benchmark, around 65-80 % of the predictions were provided by the bimodal component. The application of the partial update policy then limits the updates on banks *G0* and *G1*

[8]Note that even on context switching between two user applications such a brutal loss is not really encountered, information for the operating system is conserved for instance

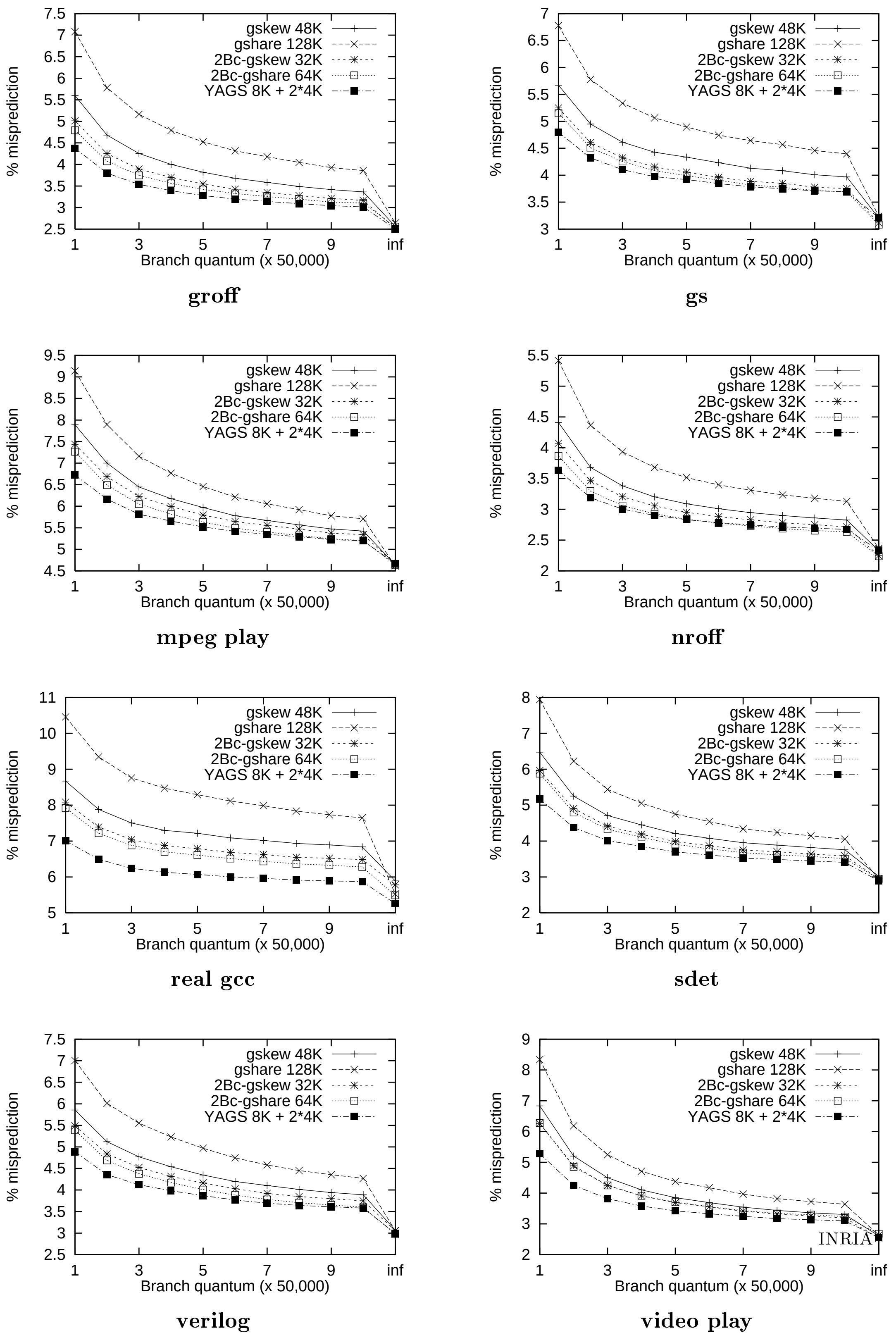


Figure 5: An estimation of major context switches impact on prediction accuracy

of the predictor, this further limits the capacity aliasing on the *e-gskew* predictor: entries in banks *G0* and *G1* are not polluted by branches who do not require using a global history register to be accurately predicted.

In the illustrated experiments, the metapredictor was indexed with a hashing function combining the branch address and the history register. The same degree of aliasing should then be encountered on banks *G0* and *G1* and on the metapredictor bank *Meta*. However, the impact of aliasing on bank *Meta* on prediction accuracy is very limited because in most cases, *e-gskew* and the bimodal predictor agree, and then the metaprediction does not matter and is not even updated.

4.2.4 2Bc-gskew versus YAGS

From our experiments, the YAGS predictor and *2Bc-gskew* exhibits approximately the same level of performance for very close hardware budgets. For a real design, chosing among these two predictors would probably depend on practical implementation considerations.

The main advantage of the YAGS predictor over *2Bc-gskew* is a a shorter training time on context switches. On the other hand, the *2Bc-gskew* predictor presents the advantage of being completely symmetric involving four identical 2-bit counter tables while the YAGS predictor is slightly more complex as it features both tagged and untagged 2-bit counter tables.

5 Combining local and global history

When de-aliased predictors are considered, global history predictors outperform local history predictors on IBS traces for hardware budgets lower or equal to 64 Kbits[9].

However there exist branches that are better predicted using a local history rather than using a global history [10, 21]. As pointed out previously, in a hybrid predictor a bimodal component is also very useful on context switches. Therefore provided a sufficient hardware budget, a hybrid predictor combining a per-address component, a bimodal component and a global address component should outperform a predictor relying on only a global history.

The possible design space for such a hybrid predictor is very large. Its exploration is beyond the scope of this paper and would require the exploration of many parameters -local history length, global history length, trade-off between hardware budgets devoted to each predictor component, ...

However, complex hybrid predictors are generally considered as an ultimate solution that can be used when a very large hardware budget is available for branch prediction. In this section, we show that the principles used in the design of *2Bc-gskew* may be used also in a three components hybrid branch predictor. First we apply the "de-aliasing" technique to propose a per-address history branch predictor, *e-pskew*. Then we show how to combine *e-pskew* and *2Bc-gskew* in *2Bc-gskew-pskew*. Some simulation results are provided to show its cost-effectiveness.

[9]for "aliased" predictors, this is not always true [5]

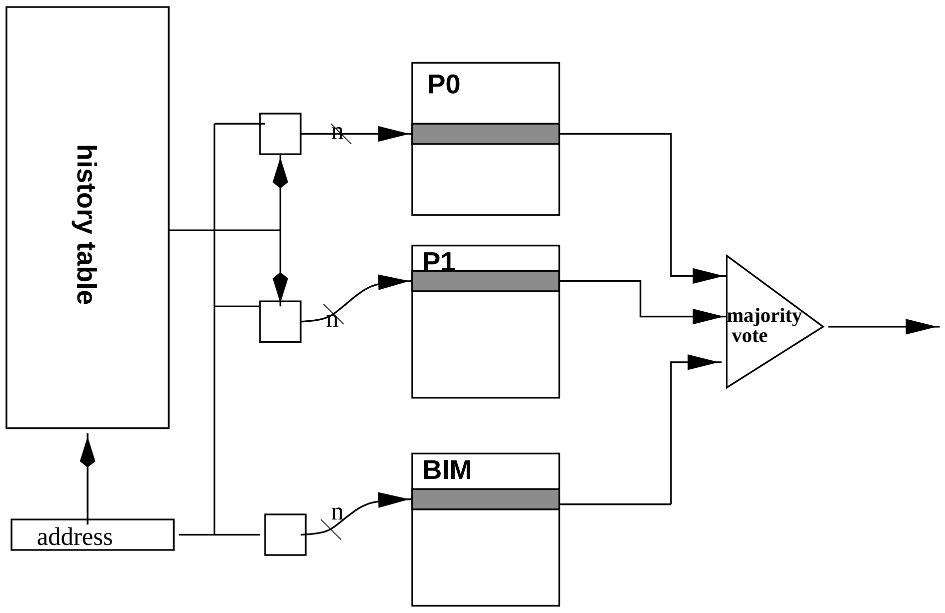


Figure 6: *e-pskew* predictor

5.1 A “de-aliased” per-address predictor

A per-address history predictor consists of a history table and a predictor table (generally saturating 2-bit counters). The address of the branch is used to index the history table, then the history and the branch address are then combined to index the predictor table. Aliasing can significantly reduce the accuracy of the prediction as for global history predictors. *e-pskew*, illustrated in Figure 6, is a “de-aliased” per-address history branch predictor. It consists of a history table, and three predictor tables. Two of the three predictor tables are indexed with independent hashing functions of the (address, history). The third predictor table is indexed with the branch address. As for *e-gskew*, a majority vote determines the global prediction. Partial updating can also be used on *e-pskew*.

We checked that, on our benchmark set, *e-pskew* outperforms other “aliased” per-address predictors at equivalent hardware complexity, but is outperformed by *e-gskew* on our benchmark set. However, *e-pskew* is the natural per-address history component to be included in a hybrid predictor as shown below.

5.2 The 2Bc-gskew-pskew predictor

2Bc-gskew-pskew is a hybrid predictor combining a per-address history predictor component, a global history predictor and a bimodal predictor. It combines *e-pskew* and *2Bc-gskew*. *2Bc-*

gskew already includes a meta-predictor. A second meta-predictor is used to chose between these two predictors. More complex meta-prediction could be implemented to select among the three basic predictor components as suggested in [4]. However, we did not find such an extra hardware complexity to be cost-effective.

2Bc-gskew-pskew may be built using a history table and seven identical 2-bit saturating counter predictor banks illustrated in Figure 7. Banks **BIM** is the bimodal predictor, but is also one of the banks of *e-pskew* and of *e-gskew*. Banks **G0** and **G1** (resp. **P0** and **P1**) are the two other banks in *e-gskew* (resp. *e-pskew*)predictor. Bank **Meta1** is is the meta-predictor used for *2Bc-gskew*, and bank **Meta2** is the global meta-predictor.

Various indexing functions can be used to index these banks, in particular the vector of information used for indexing banks **Meta1** and **Meta2**. In the experiments illustrated below, bank **Meta2** was indexed with the branch address while bank **Meta1** was indexed using a hashing function combining the branch address and the global history.

Partial updating may also be used on *2Bc-gskew-pskew* in order to limit pollution on prediction banks.

5.3 Cost-effectiveness of complex hybrid predictors

In Figure 8, we plot simulation results for a *2Bc-gskew-pskew* predictors using 16Kbits of memory: 7 banks of 1K 2bit saturating counters and a 2Kbits history tables. The considered local history length was 8, while the global history length was varied from 8 to 16. The local history length was chosen because it allows easy comparison of the hardware budgets for the predictors. The simulation results are compared with a $4 * 2K$-entry *2Bc-gskew* predictor and a $4 * 4K$-entry (i.e., 32Kbits) *2Bc-gskew*.

For a 16Kbits hardware budget, on six of our benchmarks, *2Bc-gskew-pskew* outperforms *2Bc-gskew*. A 16Kbits *2Bc-gskew-pskew* even outperforms a 32Kbits *2Bc-gskew* on two applications.

As already mentioned, the design space for a hybrid predictor combining a per-address component, a bimodal component and a global address component is very large. Many parameters may be varied: local history length, global history length, trade-off between hardware budgets devoted to each predictor component, ... We have not explored this design space. But simulations results presented here indicate that, even for relatively modest hardware budget dedicated to branch predictions[10], 3-component hybrid predictors such as *2Bc-gskew-pskew* are worth to be considered.

6 Conclusion

Decreasing the impact of *aliasing* in global history branch predictors has received a lot of attention for a few years [5, 18, 11, 7, 17, 3]. “De-aliased” global history predictors -the skewed branch predictor, the bimode predictor and the agree predictor- were introduced to

[10]i.e, in the same range as on currently announced microprocessors such as the DEC 21264

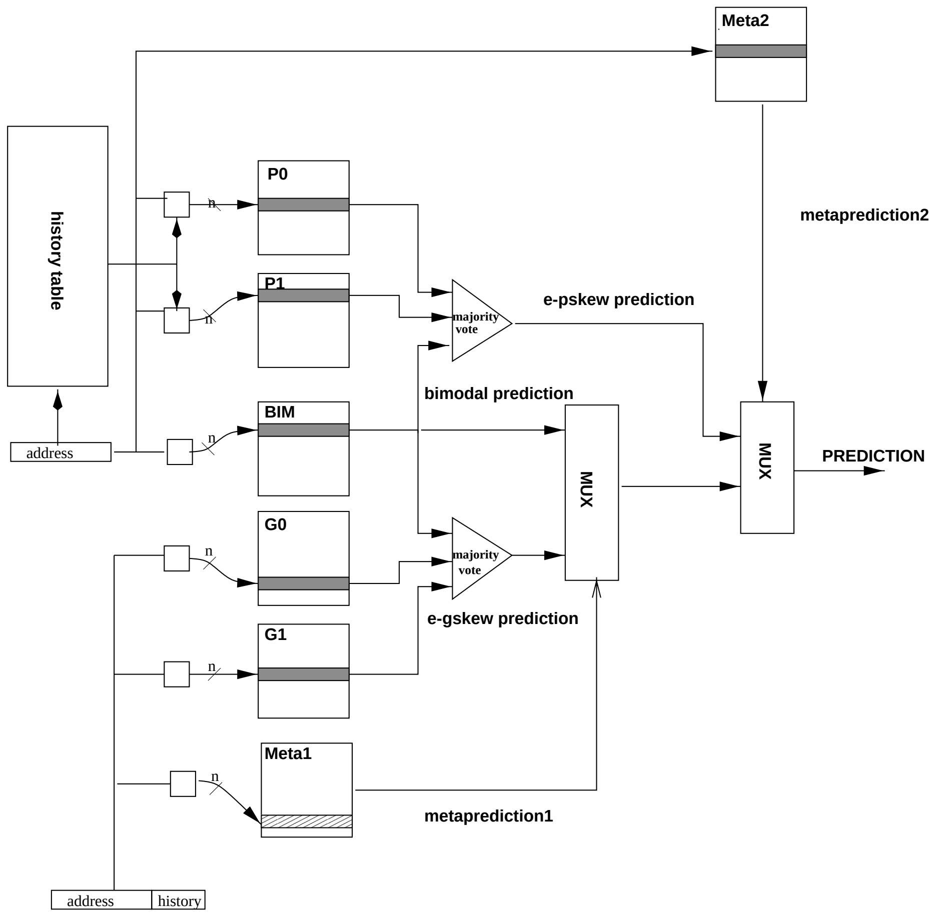


Figure 7: The *2Bc-gskew-pskew* predictor

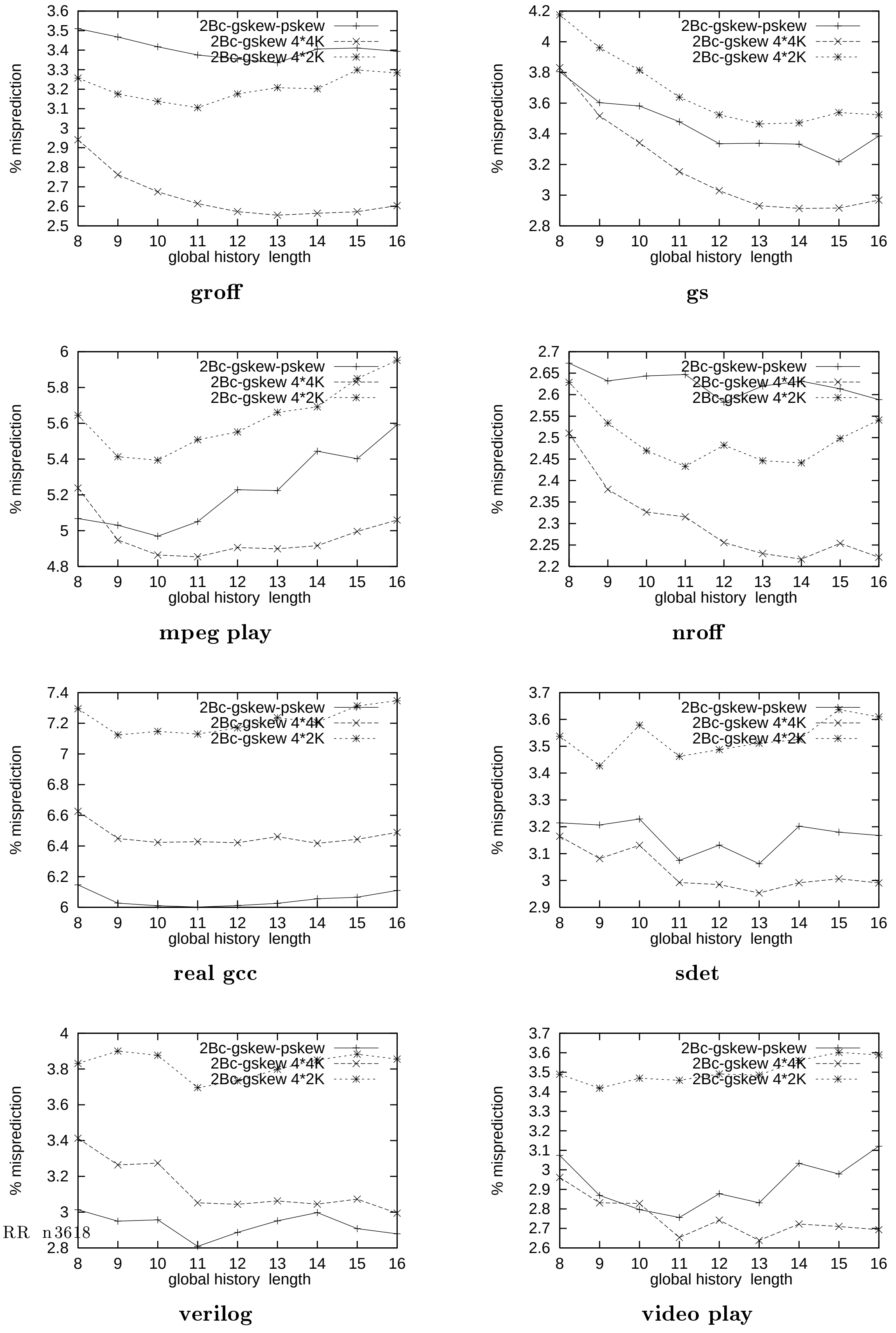


Figure 8: *2Bc-gskew-pskew vs. 2Bc-gskew*

limit this impact. At equivalent hardware budgets, they consistantly outperform "aliased" predictors.

However, it is well known that some branches are well predicted using a global history while others are better predicted using a local history. Provided a sufficient hardware budget, hybrid predictors combining several branch prediction schemes deliver higher branch prediction accuracy than a branch predictor using a single branch prediction scheme. Therefore, "de-aliased"global history predictors are natural candidates as components for hybrid branch predictors.

The enhanced skewed branch predictor *e-gskew* naturally already includes a bimodal branch predictor component. The design of *2Bc-gskew* takes part of this particularity. *2Bc-gskew* combines *e-gskew* and a bimodal predictor. It consists of four identical *predictor-table banks*, i.e., the three banks from *e-gskew* -including a bimodal bank- plus a meta-predictor. Trace driven simulations shows that *2Bc-gskew* provides the same prediction accuracy as a four times larger *gshare* predictor or a two times larger *2Bc-gshare predictor* [9], a one half larger *e-gskew* or a five fourths larger YAGS predictor. It also exhibits a low training interval after prediction information loss such as those encountered on context switches between two user applications.

Provided sufficient hardware budget, complex hybrid predictors combining more than two predictors may outperform 2-component hybrid predictors. However these predictors were not considered as cost-effective till now. *2Bc-gskew-pskew* combines 3 predictor components, *e-gskew*, a bimodal a table and *e-pskew* a "de-aliased" per-address history branch predictor. A cost-effective design of this predictor is enabled by the reuse of the bimodal table as a part of the three predictor components. This hybrid predictor has been shown to be very cost-effective, even more than *2Bc-gskew*.

References

[1] P.-Y. Chang, M. Evers, and Y.N. Patt. Improving branch prediction accuracy by reducing pattern history table interference. In *International Conference on Parallel Architectures and Compilation Techniques*, 1996.

[2] P.-Y. Chang, E. Hao, T.-Y. Yeh, and Y.N. Patt. Branch classification: a new mechanism for improving branch predictor performance. In *Proceedings of the 27th International Symposium on Microarchitecture*, 1994.

[3] A. N. Eden and T.N. Mudge. The YAGS branch predictor. In *Proceedings of the 31st Annual International Symposium on Microarchitecture*, Dec 1998.

[4] M. Evers, P.-Y. Chang, and Y.N. Patt. Using hybrid branch predictors to improve branch prediction accuracy in the presence of context switches. In *Proceedings of the 23rd Annual International Symposium on Computer Architecture*, May 1996.

[5] N. Gloy, C. Young, B. Chen, and M.D. Smith. An analysis of dynamic branch prediction schemes on system workloads. In *Proceedings of the 23rd Annual International Symposium on Computer Architecture*, May 1996.

[6] L. Gwennap. Digital 21264 sets new standard. *Microprocessor Report*, October 1996.

[7] C-C. Lee, I-C.K. Chen, and T.N. Mudge. The bi-mode branch predictor. In *Proceedings of the 30th Annual International Symposium on Microarchitecture*, Dec 1997.

[8] J.K.F. Lee and A.J. Smith. Branch prediction strategies and branch target buffer design. *IEEE Computer*, pages 6–22, January 1984.

[9] Scott McFarling. Combining branch predictors. Technical report, DEC, 1993.

[10] M.Evers, S.J. Patel, R.S. Chappell, and Y.N. Patt. An analysis of correlation and predictability: What makes two-level branch predictors work. In *Proceedings of the 23rd Annual International Symposium on Computer Architecture*, June 1998.

[11] P. Michaud, A. Seznec, and R. Uhlig. Trading conflict and capacity aliasing in conditional branch predictors. In *Proceedings of the 24th Annual International Symposium on Computer Architecture (ISCA-97)*, June 1997.

[12] R. Nair. Dynamic path-based branch correlation. In *Proceedings of the 28th Annual International Symposium on Microarchitecture*, 1995.

[13] S.T. Pan, K. So, and J.T. Rahmeh. Improving the accuracy of dynamic branch prediction using branch correlation. In *Proceedings of the 5th International Conference on Architectural Support for Programming Languages and Operating Systems*, 1992.

[14] S. Sechrest, C.C. Lee, and T. Mudge. Correlation and aliasing in dynamic branch predictors. In *Proceedings of the 23rd Annual International Symposium on Computer Architecture*, May 1996.

[15] A. Seznec and F. Bodin. Skewed associative caches. In *Proceedings of PARLE' 93*, May 1993.

[16] J.E. Smith. A study of branch prediction strategies. In *Proceedings of the 8th Annual International Symposium on Computer Architecture*, May 1981.

[17] E. Sprangle, R. S. Chappell, M. Alsup, and Y.N. Patt. The agree predictor: A mechanism for reducing negative branch history interference. In *Proceedings of the 24th Annual International Symposium on Computer Architecture (ISCA-97)*, pages 284–291, June 1997.

[18] A.R. Talcott, M. Nemirovsky, and R.C Wood. The influence of branch prediction table interference on branch prediction scheme performance. In *Proceedings of the 3rd Annual International Conference on Parallel Architectures and Compilation Techniques*, 1995.

[19] R. Uhlig, D. Nagle, T. Mudge, S. Sechrest, and J. Emer. Coping with code bloat. In *Proceedings of the 22nd Annual International Symposium on Computer Architecture*, June 1995.

[20] T.-Y. Yeh and Y.N. Patt. Two-level adaptive branch prediction. In *Proceedings of the 24th International Symposium on Microarchitecture*, Nov. 1991.

[21] T.-Y. Yeh and Y.N. Patt. Alternative implementations of two-level adaptive branch prediction. In *Proceedings of the 19th Annual International Symposium on Computer Architecture*, May 1992.

[22] Tse-Yu Yeh. *Two-level adaptative branch prediction and instruction fetch mechanisms for high performance superscalar processors.* PhD thesis, University of Michigan, 1993.

[23] C. Young, N. Gloy, and M.D. Smith. A comparative analysis of schemes for correlated branch prediction. In *Proceedings of the 22nd Annual International Symposium on Computer Architecture*, June 1995.

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399