Hybrid Myths in Branch Prediction

A. N. Eden, J. Ringenberg, S. Sparrow, and T. Mudge
{ane, jringenb, ssparrow, tnm}@eecs.umich.edu
Dept. EECS, University of Michigan, Ann Arbor

Abstract

Since the introduction of the two-level dynamic branch prediction scheme, research into branch prediction has followed two different paths. The first attempted to improve prediction by reducing aliasing in the second level table, which was shown to adversely effect prediction rate. The second attempted to improve prediction rate by combining two or more different components in the branch prediction structure. The assumption was that one component of the hybrid predictor better predicts a certain set of branches, while the second component better predicts a separate set of branches. Most papers proposing an aliasing reduction technique do not compare their new structure with a hybrid one, and vise versa. Hybrid branch predictors added the extra complication of the selection mechanism. Studies have shown the value of incorporating a static and a dynamic selection mechanism into the hybrid predictor, but have failed to identify the underlying reasons for their performance.

We present a study that consolidates the hybrid and aliasing research paths by showing that most of the advantage in combining branch prediction is gained by the selection mechanism ability to reduce aliasing. The study also shows the inability of a selection mechanism to capture the branches' changing best predictor during the programs execution (if such behavior exists). Subsequently, we show that a dynamic and a properly profiled static selection mechanism work well for the same reasons: reducing aliasing. We then highlight the advantages and disadvantages of static and dynamic selection mechanisms. The conclusion that aliasing reduction is paramount to prediction accuracy, and the observation that aliasing still degrades performance for large resource allocation, despite numerous aliasing-reducing structure, lead to the need for further improvements.

Keywords: Branch Prediction, Hybrid predictor, Aliasing, Selection Mechanism

1. INTRODUCTION

Large improvements in underlying chip technology have made available an increasing number of transistors to the microarchitect. In order to improve chips' performance, designers have employed instruction level parallelism (ILP) to fetch and execute multiple instructions per cycle. One of the most conspicuous bottlenecks clogging those wide-issue, deeply pipelined processors is the difficulty of predicting where the instruction stream will go next. It is argued that by the year 2010 branch prediction will become the most limiting factor in processor performance, surpassing even the limitation of memory system [**1**]. Those findings assumed generous resources, implying that simply increasing the size of the predictor will not solve the problem.

The introduction of the two-level adaptive branch prediction [2] resulted in considerable research activity. Different variations of the two-level adaptive branch prediction were introduced [3]. These are the global variation, which has one history register in the first level shared by all branches, and the local variation, which ideally has a history register per branch, but in reality has a set of branches sharing the same history register. It was observed that each of these two-level schemes, and earlier schemes such as the *bimodal* predictor [4][5] has its unique advantages. This led to the idea of combining branch predictors [6]. The main notion is that one set of branches is better predicted by scheme A, while a different set of branches is better predicted by branch prediction scheme B. When it is true, we will refer to it as true hybrid behavior. In such cases, it may be beneficial to combine the two schemes in a hybrid predictor and let each set of branches be predicted by the branch prediction scheme that predicts it most accurately. We will call this line of study the 'hybrid path.'

It was observed that aliasing in the second level of the two-level branch prediction structures can cause considerable degradation in prediction in two-level branch prediction structures [7][8]. This led to the development of numerous branch prediction structures that attempted to reduce the adverse effect of aliasing, particularly in the global two-level branch prediction schemes. We will call this line of study the 'aliasing path.'

This paper will show that most of the gains achieved in hybrid predictor are due to the ability of the selection mechanism to reduce aliasing, not to true hybrid behavior. It follows that hybrid predictors should be compared to aliasing reducing structures and vise versa, because they both achieve their goals by attacking the same problem. We further observe that true hybrid behavior is due to a limited number of branches, and that both dynamic and properly profiled static selection mechanisms map those branches into their respective best component (the different predictors that make up the hybrid predictor). Moreover, we show that both dynamic and static selection mechanisms achieve the same goals, namely, reducing aliasing, in different ways. We also show that the advantages of dynamic selection mechanisms can be applied to static selection mechanisms by a profiling method. We conclude by comparing a well-known aliasing reducing predictor with a hybrid implementation.

The paper is organized as follows: Section 2, surveys related work highlighting respective strengths and weaknesses. In section 3 we explain the simulation methodology used in this paper. Section 4 offers a comprehensive study, which demonstrates in detail the points above. Finally, in section 5 we provide some concluding remarks.

2. PREVIOUS WORK

2.1. Reducing Aliasing

Aliasing is the phenomenon of two unrelated pieces of information sharing the same entry in a table, usually as a result of resource limitations. The lack of a tag in the table hides the source of the information, and as a result this information is used

regardless of its validity. This phenomenon usually degrades prediction. Structures, which reduce aliasing, can be grouped by the underlying method they use. We identified three such methods: reducing destructive aliasing [9][17][11], filtering [12][11], and utilizing associativity [14][11].

While the different aliasing reducing structures have been compared to one another in the literature, there has been no comparison with hybrid branch predictors.

2.2. Hybrid Predictors

The question whether the advantages of different schemes can be combined has been discussed [6]. That author then proposes the *bimodal-gshare* and local-*gshare* hybrid mechanisms. The *bimodal-gshare* hybrid predictor outperforms *gshare*, and the author hints that filtering may be the reason. The local-*gshare* hybrid predictor predicts better than the *bimodal-gshare* for predictors larger than 16KB.

Classifying branches into sets with similar bias behavior was suggested in work on branch classification [9]. Different sets of branches are predicted by different predictors. Specifically, highly biased branches are predicted with a short history, while less biased branches are predicted with a long history. Selecting the hybrid component is determined by profiling. In [9], a static-local-global predictor was also presented. The selection of whether to use the static component was determined by profiling, while a *bimodal* selection mechanism was used to choose between the local and global components. This structure improved on the prediction of the global-local hybrid structure.

Although filtering was explicitly mentioned in some of the hybrid predictors, they were not presented as a method to reduce aliasing. While most hybrid studies compared their results with the *gshare* structure, they do not compare themselves to a more sophisticated aliasing reducing structure.

2.3. Selection Mechanisms

The first hybrid structure [6] used an array of 2-bit saturating counters indexed by the branch address as the selection mechanism. The counter is increment or decrement depending on which one of the hybrid's components gave the correct prediction; it is not updated if both components gave the same prediction (either correct or incorrect). It was proposed that using a global two-level scheme as the selection mechanism might improve prediction [10]. A static selection mechanism using profiling was compared with a dynamic selection mechanism in [15]. Although profiling for selection mechanisms was investigated before in [9], the authors in [15] use an improved profiling method to achieve better prediction for the local-*gshare* hybrid predictor. Two reasons were credited for the improvement. First, using a static selection mechanism doesn't require hardware resources for the selection mechanism, freeing hardware for use in the hybrid components. Second, only one component needs to be updated, because each branch uses only one component. This reduces contention in the second level structures.

3. SIMULATION METHODOLOGY

Performance of each hybrid configuration was measured by trace-driven simulations performed on the SPECint95 benchmarks, the PowerPC and S390 benchmarks. The SPECint95 traces were obtained by running SimpleScalar's bpred program [18] and extracting the branch address, target address, and branch outcome. The S390 and PowerPC traces were provided by IBM. IBM provided no further information about the traces, but their distinguishing characteristic is a large number of branches. This promotes aliasing, which considerably degrade prediction for these benchmarks. Table 1 displays the number of static branches for the different benchmarks.

	static branches	
	real	**train**
SPECint95		
gcc	13763	14085
compress	495	704
go	7401	7749
jpeg	2760	2854
li	1701	1457
m88ksim	1646	2199
perl	3443	2721
vortex	7581	11132
IBM Traces		
s390	21727	N/A
powerPC	16710	N/A

Table 1: traces used in simulations

In all the simulations performed in our studies, the depth of correlation (the size of the history register/s) follows directly from the size of the PHT. For example, if the global component in the hybrid predictor had 1K entries in its PHT, the history register size will be 10 bits. We used the McFarling local-*gshare* hybrid predictor because in preliminary simulations it exhibited the best true hybrid behavior. The McFarling predictor was used exactly as described in [6].

In simulations with a real structure (limited to different sizes), we used a two way set associative BTB with 4K entries. This is large enough to prevent it from being a performance bottleneck and enables us to concentrate on the tradeoffs in the PHTs.

All hybrid predictors simulated had two components, a *gshare* structure implementing the global branch prediction scheme, and a PAs structure implementing the local branch prediction scheme. In cases where a dynamic selection mechanism was employed, the *bimodal* structure was used.

Profiling was done on the same data sets that were used for simulation, unless stated otherwise. This enabled us to obtain an upper limit on the prediction accuracy. It is expected that using a different data set (the more realistic situation) for profiling will degrade the performance of the hybrid predictor with a static selection mechanism, as one of the studies in the paper shows.

Throughout the discussion, we present only four graphs due to lack of space: the arithmetic average of the SPECint95, the gcc benchmark from the SPECint95 suite, the PowerPC benchmark, and the S390 benchmark. The gcc benchmark is presented because it is relatively hard to predict compared to the other SPECint95 programs. The PowerPC and S390 are presented because of their large number of branches. When the SPECint95 average is not applicable, we present the results of the go benchmark instead. Simulations were conducted for all other SPEC95 benchmarks, and the results were similar.

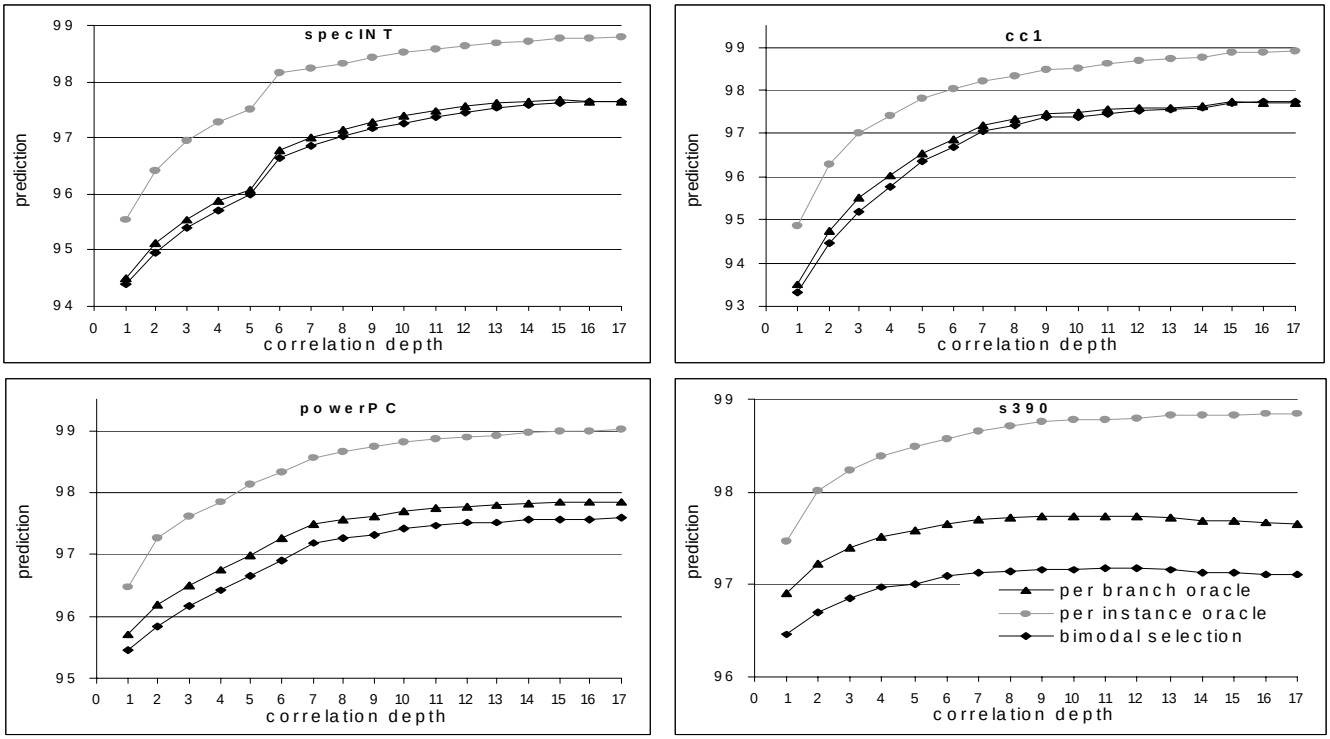

Figure 1 - Testing the potential of static vs. dynamic selection mechanisms in a unlimited resource environment

4. EXPERIMENTAL RESULTS

4.1. Static vs. Dynamic Selection Mechanism

We begin by examining the relative merits of using a static versus a dynamic selection mechanism to choose between the different components of a hybrid predictor. As noted earlier, a static selection mechanism requires less information to be stored in the predictor structure, because each branch utilizes only one component. This reduces contention, which reduces aliasing and helps the prediction rate. Moreover, hardware resources that would have been used for the selection mechanism are now available for increasing the size of the predictor's components. The main problem with static selection is the additional bits needed in the ISA. Although some ISAs have this bit in place, others will require that the ISA be altered. Dynamic selection mechanisms are claimed to have an edge over static ones, because it has been suggested that the best component for predicting a branch can change during the execution of a program.

It is unclear, however, whether there is an inherent benefit in choosing the component used by a specific branch dynamically. If the best component to predict a branch is dynamically changing during the program run, it will be beneficial to dynamically select the component used by a branch. However, if there is no inherent benefit in choosing the component used by a branch dynamically, it is beneficial to choose it statically and avoid the extra cost of using both components for each branch, and the cost of the selection mechanism.

Figure 1 shows the prediction accuracy for an unlimited resources global-local hybrid predictor. The three plots represent three types of selection mechanisms: per-branch oracle, per-instance oracle, and an implementation of a real selection mechanism – the *bimodal*. The per-branch oracle records the prediction rate for both components and when the program terminates, it chooses the best component as the predictor for each branch. The per instance oracle get a prediction from both components, and if any of them is correct, it records a correct prediction. (Notice that the per-instance oracle is an overestimation and even for a randomly generated prediction, probability dictates a 75% correct prediction.) Determining whether the best component to predict a branch changes during program execution is difficult. One approach is to slice the dynamic stream of a specific branch into *n* subsets of branch instances, and then to choose the best component for each set [16]. The problem is that a small *n* leads to an optimistic outcome, while a large *n* might erase the benefit of having a dynamic selection mechanism. Using either large *n* or small *n* can lead to the erroneous conclusions. Clearly, it does not matter whether the best component for each branch changes throughout the program run if a known selection mechanism cannot identify the best component dynamically. In our experiments we used an unbounded hybrid predictor with an unbounded *bimodal* selection mechanism. This eliminated the adverse effect of aliasing and allows a check on whether the *bimodal* selection mechanism can capture the changing best predictor throughout the program execution. Figure 1 shows that there is no inherent gain in using a dynamic selection mechanism. In other words, if there is a gain to be made in changing the component used for each branch during the program execution, the *bimodal* selection mechanism does not capture it[1]. This is made clear in the graphs where it can be seen that the *bimodal* selection mechanism always under-performs the per branch oracle. Moreover, it appears that the bimodal selection mechanism makes mistakes in selecting the proper

1. The global selection mechanism was considered as well, but provided similar results.

component, which degrades the overall performance. This phenomenon is accentuated in programs with a large number of branches like the S390 and PowerPC. They display a significant gap between the prediction of the oracle static selection mechanism and the prediction when using the bimodal selection mechanism.

Figure 1 depicts the inability of the dynamic selection mechanism to dynamically adapt to the changing behavior of branches, even if such a transient behavior exists. Therefore, there does not appear to be an advantage to employing dynamic selection mechanisms instead of static ones. We would thus expect that in a limited resource setting a static selection mechanism will outperform a dynamic selection mechanism for the reasons mentioned above (less aliasing and more resources dedicated to the prediction components). Figure 2, however, shows the exact opposite. In a limited resources setting, the hybrid predictor with a dynamic selection mechanism (dynamic) outperforms a hybrid with a perfect static selection mechanism (static). The other plots (static limited and static unlimited) will be discussed later and can be ignored for moment.

Holding the heel of this observation a question is born: What is it about the dynamic selection mechanism that boosts the performance of a hybrid predictor with a dynamic selection mechanism when working in a size-restricted structure? Alternatively, what is it about the static selection mechanism that in a limited-resource setting degrades the performance of a hybrid predictor?

One possible hypothesis is that a dynamic selection mechanism reduces aliasing. For example, consider the case where two branches *A* and *B* are both better predicted by the global component of the hybrid predictor. In an unlimited resource setting, a dynamic selection mechanism will choose the global component to predict them. In a resource limited setting, branch *A* will suffer from aliasing, which considerably degrades the prediction of its global component. As a result, the dynamic selection mechanism chooses the local component to predict branch *A*'s outcomes. Although both branches *A* and *B* are inherently better predicted by a global component, branch *A* will be better predicted by the local component in a limited resources environment. We next examine how much aliasing reduction helps a hybrid predictor.

4.2. Aliasing Reduction in Hybrid Predictors

Figure 3 shows the extent to which reducing aliasing helps boost the performance of hybrid prediction. It compares a resource bound local-global hybrid predictor (hybrid), with a resource bound local-global hybrid (aliasing hybrid), where the selection mechanism doesn't take into consideration the effect of aliasing. To simulate this effect, a run of the local-global hybrid predictor was made with no limits on resources. The selection pattern for the entire run was logged and later served as the selection mechanism in the limited hybrid version. The selection mechanism in this case is that for the true hybrid behavior with no regards to aliasing, since it was recorded in an aliasing free setting. From figure 3, we conclude that a large portion of the benefits brought by hybrid predictors with dynamic selection mechanism come from reducing aliasing. Moreover, comparing the hybrid predictor to an unlimited version of the global scheme (UL global), shows that the local-global hybrid predictor never fulfils its promise of improving prediction beyond that of a single scheme, even for generous resource allocation. Notice that the different between UL Hybrid and UL global is the potential difference between the hybrid predictor (global-local) and the global scheme. This difference pales in comparison to the difference between UL global and hybrid that represents the remaining aliasing after the *bimodal* selection mechanism was able to reduce some of them (the difference between hybrid and aliasing-hybrid).

4.3. Prediction Potential of a Hybrid Structure

Next we investigated whether there is an inherent gain in the local-global hybrid predictor over a single scheme, or whether the gain realized by the hybrid predictor is limited to reducing aliasing rather than to true hybrid behavior. Figure 4 shows the improvement of the program's prediction for each branch (x-axis) when using the local predictor versus the global predictor with no limits on resources. Positive percentages indicate the branch is better predicted by the local scheme, while negative percentages indicate the branch is better predicted by the global scheme. The branches are sorted on the x-axis according to the percentage improvement. Figure 4 shows that the number of branches that contribute to the true hybrid behavior of the local-global hybrid predictor is small. Here and after, these small number of branches will be reffered to as the hybrid branches. For most branches the improvement obtained by using the global component instead of the local component or vice versa is insignificant. Only a few branches (the hybrid branches) are responsible for the improvement of a local-global hybrid predictor over a single scheme predictor. If the predictor component for the other branches (the majority) changes dynamically to reduce aliasing, it remains to make sure that the hybrid branches are predicted by the component that does it best. This will allow to take advantage of both alias reduction and true hybrid behavior. When employing a static selection mechanism, this can be done at profile time. In the case of a dynamic selection mechanism it seems that an explicit way of indicating the appropriate component for the hybrid branches is needed. However, in a study we conducted, it was shown that the dynamic selection mechanism is already performing that task of mapping the hybrid branches into their respective best component. Attempting to lock the hybrid branches into their respective best component, while letting the rest of the branches' component to be chosen dynamically, resulted in degraded performance.

Despite the potential embedded in hybrid predictors, and the ability to the selection mechanism to identify the hybrid branches, this potential is not fulfilled. Performance degradation due to aliasing dominates the hybrid potential that as a result is never fulfilled.

4.4. Aliasing Aware Static Selection Mechanism

At this point we have shown that both static and dynamic selection mechanisms reduce aliasing in hybrid branch predictors. The former by reducing contention in the structure (updating only one component) and eliminating the hardware cost in the selection mechanism, and the later by dynamically distributing the branch stream across the two components while alleviating contention in the PHT. Dynamic selection mechanism performs much better than an ideal static selection mechanism. In the ideal static selection mechanism, profiling was done with no limitation on resources. This led to branches better predicted by the global scheme to be mapped to the gshare component, and branches better predicted by the local scheme, to be mapped to the PAs component. Notice that the ideal static selection mechanism does not take aliasing into consideration. One way of

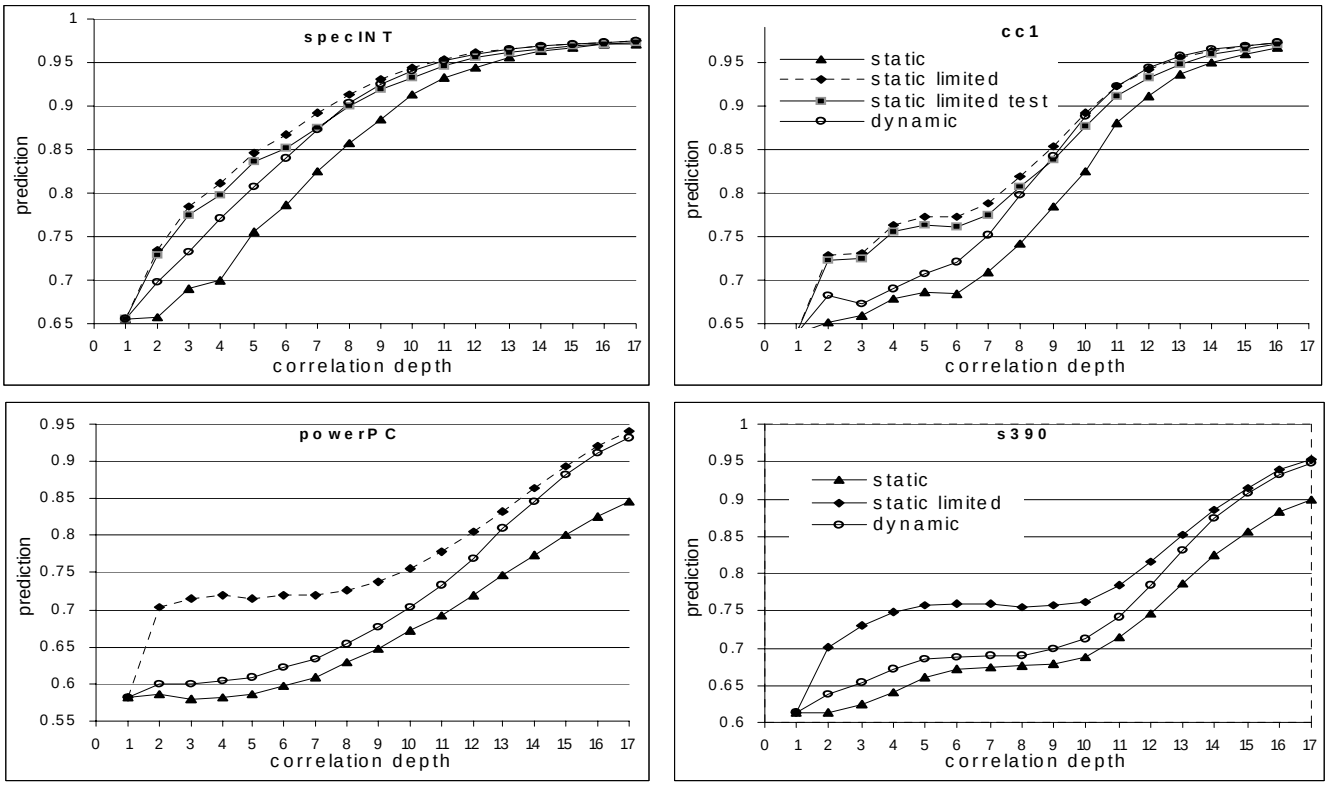

Figure 2 – Dynamic vs. perfect static selection mechanism in hybrid predictors

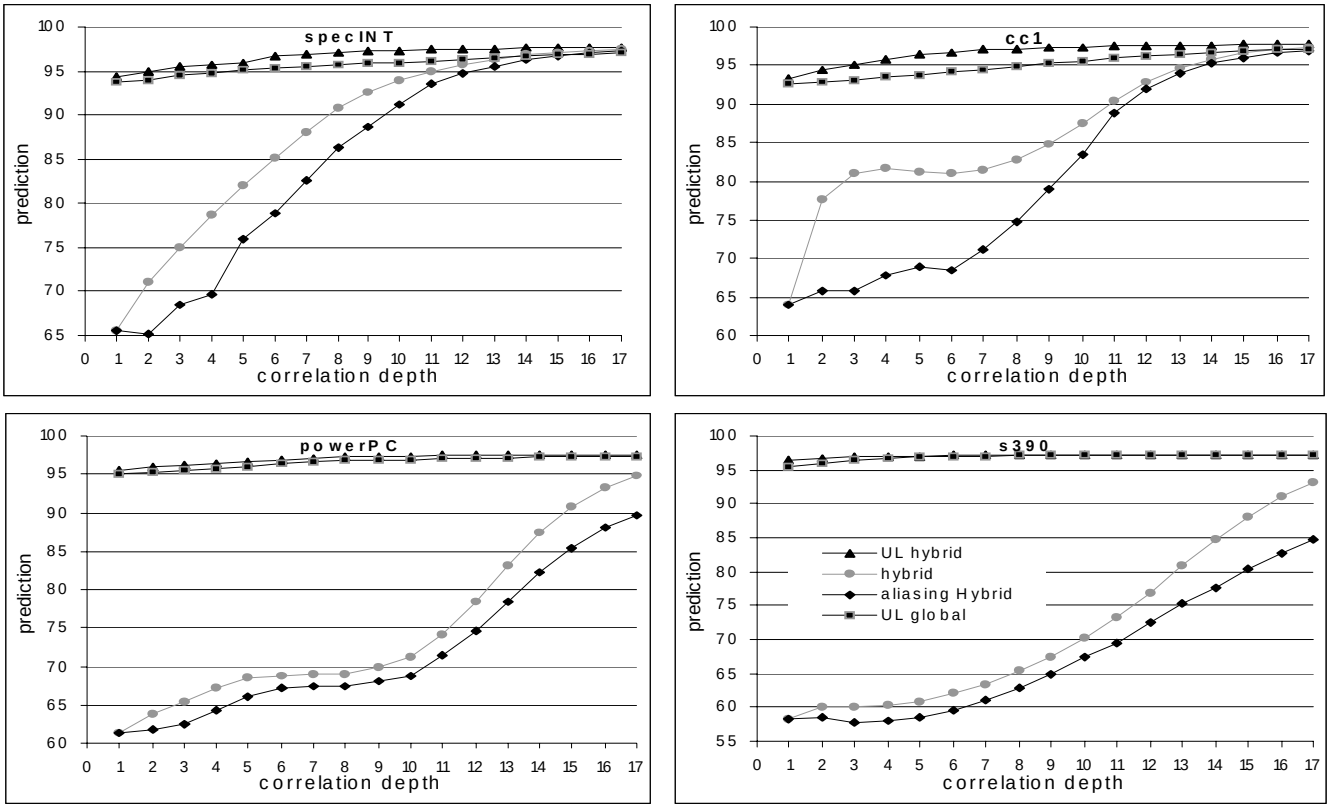

Figure 3 – The role of Hybrid predictors in reducing aliasing

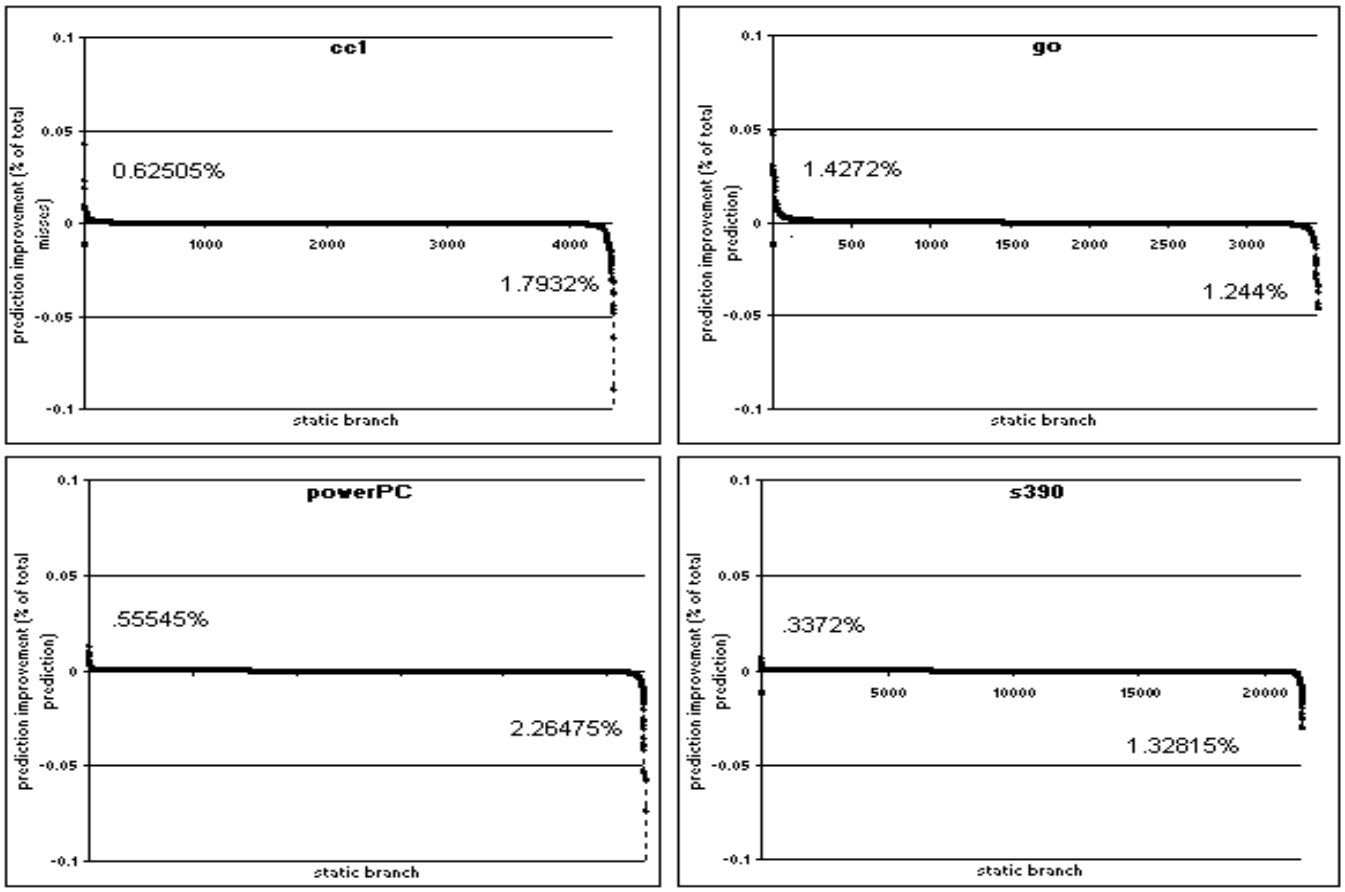

Figure 4 – per branch potential of a hybrid component

achieving this is to take the actual table size into consideration while profiling. Figure 2 shows the importance of taking into consideration the size of the predictor structure when profiling. When taking size into consideration during profiling, the branches are distributed not just by their true hybrid behavior, but also by taking aliasing into consideration. Figure 2 shows that while dynamic selection mechanism is better than a static selection mechanism with perfect profiling, employing profiling that takes the size of the structure into consideration (static limited) results in even better performance than dynamic selection. The fact that the difference between the prediction percentages diminishes with size indicates that the difference is mostly due to better aliasing reduction. Using this profiling method combines the advantage of static and dynamic selection mechanisms as we explained before.

The advantages of using a static selection mechanism with aliasing bound profiling are as follows: 1) The branches are distributed amongst the component according to contention in the structure. 2) The selection hardware is eliminated. 3) Only one component is used per branch, which further reduces contention. The question arises whether such good prediction can be achieved when profiling from a test data set. As figure 2 shows, when using a different data set to profile the program the static selection mechanism (static limited test) suffers degradation in performance. For small predictors the static selection mechanism still performs better than the dynamic selection mechanism, but the dynamic selection mechanism eventually surpasses it.

4.5. Comparing Hybrid and Aliasing-Reducing Structure

Finally, after discovering that the main strength of hybrid predictors is reducing aliasing, we made a direct comparison between one of the most used aliasing reduction implementations, the *bi-mode* predictor, and the McFarling hybrid predictor (figure 5). If the size of the local history registers is ignored (McFarling), the McFarling predictor does better than the *bi-mode* predictor for small size predictors, while the *bi-mode* predictor performance catch up for larger predictors and eventually surpasses the McFarling predictor. This phenomenon is accentuated for traces with a large branch signature like the S390 and the PowerPC. For the PowerPC trace, predictors larger than 1K bytes should use the *bi-mode* structure. This is also true for the SPECFP traces (not shown). When taking the size of the local history registers into account (McFarling adjusted), the McFarling predictor performs poorly. However, it should be noted that the BTB used in the simulations was large to prevent it from being a bottleneck. We hypothesize that much smaller BTB (and therefore smaller size local history registers) can be used without compromising performance. However, factors like ease of implementation and smaller access time further the favors a *bi-mode* implementation.

It would be acceptable if the hybrid predictor produced poor prediction compared to the *bi-mode* predictor for smaller size predictors. The hybrid scheme produces more information than the global scheme (since it implements both the global and local scheme), and results in more aliasing, which degrades performance. Conversely, the fact that for larger predictors the *bi-*

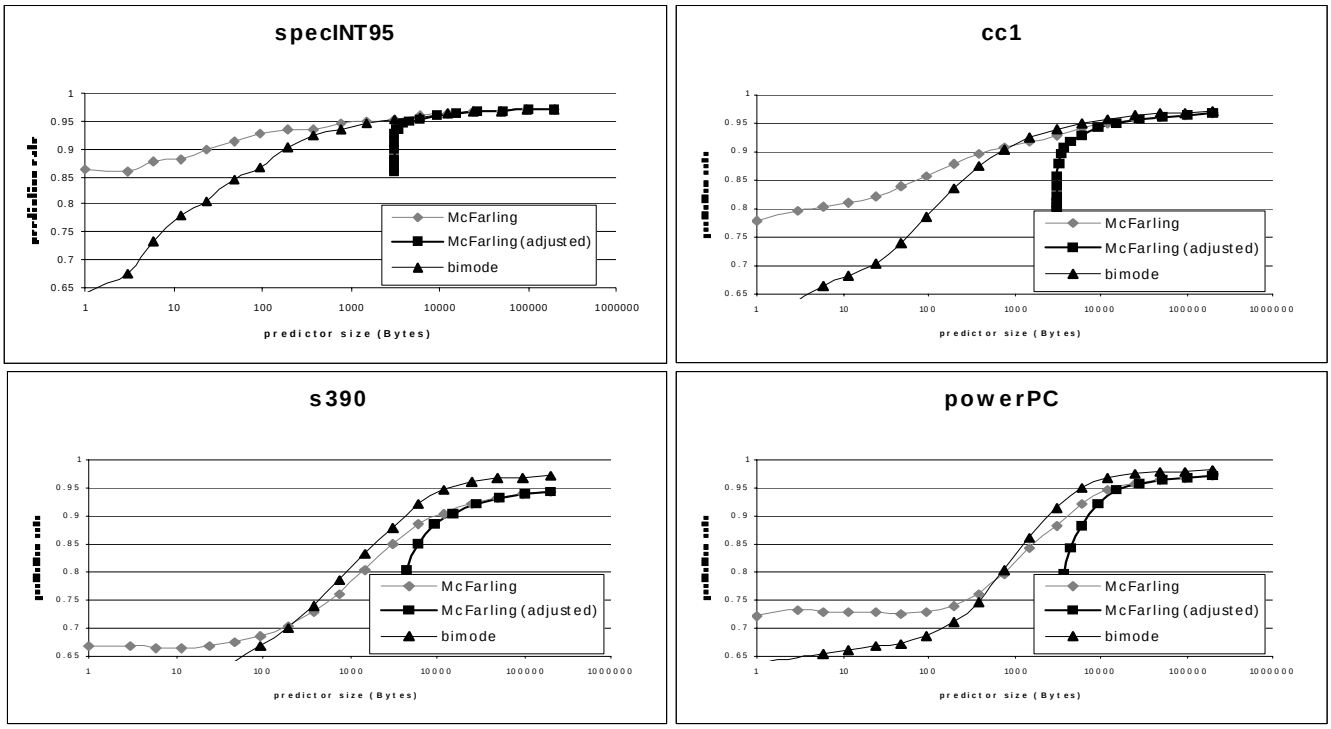

Figure 5 – McFarling vs. Bimode predictor

mode predictor outperforms the McFarling predictor further proves that current implementation of the hybrid predictor are not capable of taking advantage of the hybrid scheme's potential.

Assuming that combining the two paths of research, and reaping. the benefits of each, is easy, is partially to blame for the misunderstanding of the hybrid path. In our pursuit of integration between hybrid structures and aliasing-reducing structures, we experimented with a McFarling hybrid predictor where each of its component is a *bi-mode* predictor. The expectation was that this would gain from both true hybrid behavior and reduce aliasing. In fact the oracle selection mechanism with these predictors failed to achieve this, and the best predictor was the version with the profiling obtained by simulating the structure size. Once again, the benefit of reducing aliasing overwhelms the benefits of the hybrid scheme.

5. SUMMARY

We have shown that the major contribution of hybrid predictors to enhance prediction is their ability to reduce aliasing. In a sense the true hybrid behavior is insignificant. Therefore, studies that work on structures to reduce aliasing should be compared to known hybrid predictors and vise versa. We also refuted the belief that dynamic selection mechanisms can capture the branch's changing behavior. Instead we showed that the dynamic selection mechanism works well because of its ability to serve as a load balancer to reduce aliasing. This can be achieved with static selection mechanism as well if proper profiling is done. Different data sets, however, degrade a static selection mechanism's performance. For smaller predictors the static selection mechanism performed best, while for larger predictors the dynamic selection mechanism has an edge.

Since every aspect of improvement in prediction accuracy we investigated turned out to be due to aliasing reduction[2], we urge future studies to consider that and perform appropriate limit studies to confirm that this is not the case with new structures/ schemes. The consolidation of the different research paths and the fact that aliasing still degrades performance, even for large resource allocation, should lead to further study of predictors that reduce aliasing. Current branch predictors are not able to take advantage of the potential in the hybrid scheme. Future predictors, however, should investigate ways to do so.

6. ACKNOWLEDGEMENT

The authors would like to thank P. Emma, M. Charney and T. Puzak of IBM T. J. Watson research center for the S390 and PowerPC database traces. We also like to thank Amy Claire Harfeld for help in editing this paper.

7. REFERENCES

[1] Parthasarathy Ranganathan and Norman Jouppi. "The relative impact of memory latency, bandwidth and branch limit to micrprocessor performance." Presented at 1st Workshop on Mixing Logic and DRAM: Chips that Compute and Remember, held in conjunction with the 1997 International Symposium on Computer Architecture, Denver Colorado, June 1997.

[2] Yeh, T-Y. and Patt, Y. "Two-Level Adaptive Training Branch Prediction," Proceedings of the 24th International Symposium on Microarchitecture, 51-61, Nov. 1991.

[3] 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

2. In a set of studies not presented here, we discovered that the 'third-level of adaptivity' research path improve prediction accuracy for the mere reason that it filters information, and consequently reduce aliasing.

Support for Programming Languages and Operating Systems, pp. 76-84, 1992.

[4] Smith, J.E. “A Study of Branch Prediction Strategies,” Proceedings of the 8th International Symposium on Computer Architecture, 135-148, May 1981.

[5] Nair, R. Optimal 2-bit branch predictors. IEEE Trans. on Computers, Vol. 44, No. 5, May 1995.

[6] McFarling, S. “Combining Branch Predictors,” WRL Technical Note TN-36, Jun. 1993.

[7] Talcott, A.R., Nemirovsky, M., and Wood, R.C., “The Influence of Branch Prediction Table Interference on Branch Prediction Scheme Performance,” Proceedings of the 3rd International Conference on Parallel Architectures and Compilation Techniques, Jun. 1995.

[8] Young, C., Gloy, N., and Smith, M. “A Comparative Analysis of Schemes for Correlated Branch Prediction,” Proceedings of the 22nd International Symposium on Computer Architecture, Italy, Jun. 1995.

[9] Chang, P., Hao, E., Yeh, T., and Patt, Y., “Branch Classification: a New Mechanism for Improving Branch Predictor Performance,” IEEE Micro-27, Nov. 1994.

[10] Yeh, T-Y. and Patt, Y. “Alternative Implementations of two-level adaptive branch predictions,” Proceedings of the 19th International Symposium on Computer Architecture, 124-134, May 1992.

[11] A. N. Eden and T. N. Mudge. “The YAGS branch predictor.” In Proceedings of the 31st International Symposium on Microarchitecture, December 1998.

[12] Chang, P., Evers, M., and Patt, Y., “Improving Branch Prediction Accuracy by Reducing Pattern History Table Interference,” International Conference on Parallel Architecture and Compilation Techniques, Oct. 1995.

[13] Andre Seznec and Francois Bodin, “Skewed-associative Caches.” In proceedings of the PARLE’93, May 1993

[14] Michaud, P., Seznec, A., and Uhlig, R., “Trading Conflict and Capacity Aliasing in Conditional Branch Predictors,” Proc. of the 24th Ann. Int. Symp. on Computer Architecture, May 1997.

[15] Grunwald D., Lindsay D, and Zorn B. “Static Methods in Hybrid Branch Prediction” Proceedings of the International Conference on Parallel Architecture and Compilation Techniques, 1998.

[16] Evers, M. Improving Branch Prediction by Understanding Branch Behavior. Ph.D. Thesis, The University of Michigan, 1999.

[17] C.C. Lee, I.C. K. Chen, and T. N. Mudge, ``The *bi-mode* branch predictor,'' in Proceedings of the 30th Annual ACM/ IEEE International Symposium on Micro-architecture, pp. 4 -- 13, 1997.

[18] D. Burger, T. Austin, “The SimpleScalar Tool Set, Version 2.0”, Technical Report TR 1342, University of Wisconsin, June 1997