Branch Transition Rate: A New Metric for Improved Branch Classification Analysis

Michael Haungs, Phil Sallee, and Matthew Farrens

Department of Computer Science, University of California, Davis
E-mail: {haungs,sallee,farrens}@cs.ucdavis.edu

Abstract

Recent studies have shown significantly improved branch prediction through the use of branch classification. By separating static branches into groups, or classes, with similar dynamic behavior, predictors may be selected that are best suited for each class. Previous methods have classified branches according to taken rate (or bias). We propose a new metric for branch classification: branch transition rate, which is defined as the number of times a branch changes direction between taken and not taken during execution. We show that transition rate is a more appropriate indicator of branch behavior than taken rate for determining predictor performance. When both metrics are combined, an even clearer picture of dynamic branch behavior emerges, in which expected predictor performance for a branch is closely correlated with its combined taken and transition rate class. Using this classification, a small group of branches is identified for which two-level predictors are ineffective.

1. Introduction

A major obstacle to increasing the instructions per cycle (IPC) for modern architectures are conditional branches. Conditional branches greatly reduce the amount of instruction level parallelism (ILP) that can be utilized by today's modern architectures to increase IPC. To eliminate this negative impact, very accurate branch predictions are needed.

In 1991, Yeh and Patt proposed the Two-Level Adaptive Branch Predictor, and in 1992, they extended this research by proposing alternative implementations of two-level adaptive branch prediction [21, 22]. Two-level adaptive branch predictors or (2-level predictors) provided a new level of branch prediction accuracy and have been a main focal point of branch prediction research. McFarling [17] soon followed with his paper on combining predictors. While this paper is best known for the advent of the gshare predictor, the concept of combining predictors rapidly became prevalent in the research. Chang et al. [3] proposed a method of classifying branches according to their dynamic taken rate and assigning branches in each class to different predictors. They made the important discovery that different history lengths for 2-level predictors performed better for different classes of branches.

Although the design space for branch predictors grew tremendously, an understanding of how and why these predictors worked did not. More recently, however, a number of papers, such as [23], [12], [6], [16], and [4], have provided insights into mechanisms for creating more accurate branch predictors. Our goal is to continue this process towards a deeper understanding about the nature of branch behavior and branch predictor performance.

In this paper, we introduce a new metric for branch behavior, *branch transition rate*. Branch transition rate is defined as the number of times a branch changes direction between taken and not taken over a given number of executions of that branch. We show that transition rate is a better indicator for use in branch classification methods than the taken rate metric proposed by Chang et al [3]. Using a variety of history lengths, we analyze predictor performance for branches in taken and transition rate classes for both per-address and global history 2-level predictors. Given this new understanding, we make general suggestions for branch predictor improvement.

In the next section, we briefly summarize previous research that is relevant to our work. Section 3 describes our method of data collection for the results we present. In section 4, we use both the taken rate and transition rate metrics to analyze groups of branches for the SPEC95 benchmarks. From this analysis, we propose new ways to improve the prediction accuracy of 2-level adaptive predictors in section 5. Section 6 concludes our paper with a summary of our findings and an outline of proposed future work.

2. Related Work

The most closely related work in this area is by Chang et al. which introduced the idea of branch classification using taken rates [3]. These researchers compared various schemes which used taken rates derived from profiling to assign either static predictors or predictors with fewer history bits to heavily biased branches. Their results prompted our investigation into alternative classification schemes.

Since then, others have tried to eliminate interference using some form of either dynamic or static classification. Several proposals have schemes that vary the history length either on a per-branch basis or program basis. Also, work has been done to better understand the branch behavior and dynamics of 2-level branch predictors.

Chen et al. [4] use techniques established in the field of data compression to form a theoretical basis for branch prediction and make suggestions for improvement. They develop this theoretical basis by demonstrating that current correlated predictors are simplifications of the PPM (Prediction by Partial Matching), an optimal predictor in data compression. They reason that since 2-level adaptive predictors are good approximations of the PPM predictor, they are near optimal and suggest improvements may be realized by supplying additional knowledge of the executing program to the predictor, varying the history length, and by using n-bit counters.

The YAGS predictor paper [5] nicely summarizes a variety of predictors (gshare [17], Agree [19], Bi-Mode [14], Skew [18], and Filter [2]) that use varying methods to eliminate interference in pattern history tables. In their paper [5], they address the deficiencies of these predictors and introduce their own predictor, called YAGS (Yet Another Global Scheme). Their focus on interference in 2-level adaptive predictors is in agreement with the work above of Chen et al [4]. Although not always explicit in their classification, most of these predictors use some form of classification to reduce interference. The Agree predictor uses a simple form of bias classification to turn destructive interference (or aliasing) into constructive or neutral aliasing. The Bi-Mode predictor is similar to the Agree predictor, but uses a more dynamic form of bias classification. The Filter predictor behaves much like the bias classification of Chang et al. [3] to reduce interference by removing static branches from the pattern history tables. One difference is that a dynamic counter is used instead of static profiling to determine the bias. The bias counter used by the filter method might also be considered a simple form of transition rate classification since it counts the number of branch executions since the last time a transition occurred.

Juan et al. [11] argued that branch predictors constrained to a fixed history length are sub-optimal. They devised a scheme, DHLF (Dynamic History-Length Fitting), that dynamically selects a history length that works best for the program in execution. This is a coarse grained method of varying history length to achieve prediction accuracy improvement. A finer grained method, as suggest by Chang et al. [3], would vary the history length on a per-address basis.

Stark et al. [20] introduced a method to vary the length of the global path history in a global adaptive 2-level predictor. They profile a program to calculate the best global history length for each branch. Then they map each branch to 1...N hash functions, in the hardware, that use 1...N bits, respectively, of the global history to compute an index into the second level table. They achieved significant improvements, especially when considering small hardware budgets.

Kim and Mudge [12] looked at the dynamic working set characteristics of branches. A working set partition of dynamic branches is based on temporal locality and ordering information. Although radically different than the others, this branch classification method shares the major goal of interference reduction in 2-level predictors. The working set metric, however, targets path-based branch correlation, while both transition rate and taken rate metrics are measurements of per-address branch patterns.

Grunwald et al. [8] argue that hybrid predictors are best guided by static information. They classify branches by which component of the hybrid predictor most accurately predicts each branch. From a purely implementational point of view, simply assigning branches to the component of the hybrid predictor that demonstrates the best accuracy makes sense. Branch transition rate classification, in the general case, provides a richer set of information than a prediction accuracy classification. Independent of a particular predictor, branch transition rate classification provides insight into general branch behavior, the innate predictability of a branch, the distribution of branches in each class, and branch trends. For instance, branch taken and transition rates can be used to determine appropriate history lengths to best predict a given branch. In addition, the construction of a hybrid predictor is a difficult process. We feel that prediction accuracy is a method for evaluating an existing hybrid predictor while our classification method aids in the *design* of a hybrid predictor (see Section 5.4).

2-level adaptive schemes are currently the most accurate branch predictors. Recent work has focused on understanding the underlying dynamics of these predictors and creating new ways to improve their accuracy. Primary methods for improving accuracy have been the reduction of interference, varying history lengths and branch classification.

3. Simulation Environment

For our study, we used version 3.0 of the SimpleScalar tool set [1] provided by T.M. Austin and D. Burger from the University of Wisconsin. We used the sim-bpred simulator, making modifications to track the taken rate and transition rate metrics for each branch address. Simulations were conducted on the SPECint portion of the SPEC95 benchmarks and run to completion.

Table 1. Benchmarks, input sets and number of dynamic conditional branches analyzed.

Benchmark	Input Set	Dynamic Branches
compress	bigtest.in	5641834221
gcc	amptjp.i	194467495
gcc	c-decl-s.i	194487972
gcc	cccp.i	190138561
gcc	cp-decl.i	217997360
gcc	dbxout.i	24944893
gcc	emit-rtl.i	25378207
gcc	explow.i	36513202
gcc	expr.i	153982215
gcc	gcc.i	30394247
gcc	genoutput.i	12971324
gcc	genrecog.i	18202207
gcc	insn-emit.i	20774453
gcc	insn-recog.i	85446679
gcc	integrate.i	33397714
gcc	jump.i	23141650
gcc	print-tree.i	25996412
gcc	protoize.i	76482161
gcc	recog.i	43591736
gcc	regclass.i	18259839
gcc	reload1.i	138706109
gcc	stmt-protoize.i	153772060
gcc	stmt.i	82470825
gcc	toplev.i	65824567
gcc	varasm.i	37656353
go	9stone21.in	3838574925
ijpeg	penguin.ppm	1548835517
ijpeg	specmun.ppm	1392275287
ijpeg	vigo.ppm	1627642253
li	ref/*.lsp	8493447845
m88ksim	ctl.lit	9086543174
perl	primes.pl	1738514158
perl	scrabbl.pl	3150939854
vortex	vortex.lit	9897766691

For each benchmark, the input test set and the number of dynamic branches analyzed are given in Table 3. Both a per-address history predictor, PAs, and a global predictor, GAs, were simulated for each benchmark using history lengths from 0 - 16. Each predictor was limited to 32k bytes. For GAs, the pattern history table (PHT) used contained 2^{17} 2-bit counters. For a given history length k, the remaining 17-k bits of the PHT index were filled with bits from the branch address. For PAs, a PHT with 2^{16} 2-bit counters was used, unless the history length was zero. As much as possible of the remaining space was used in the branch history table (BHT) with the restriction that the number of entries in the BHT be a power of two. The number of BHT entries, then, is given by $2^{\lfloor \log(2^{17}/k) \rfloor}$ where k is the history length. Given a zero history length, PAs and GAs are equivalent to a single table of 2-bit counters referenced by 17 bits of branch address. Only conditional branches were measured.

4. Analysis of Dynamic Branch Behavior

To optimally tailor predictors, researchers must understand the dynamic behavior of branches. In this section we look at two different dynamic branch metrics, *taken rate* and *transition rate*. The advantages of each are described separately. We then discuss the insight gained by combining them.

4.1. Taken Rate Classification

Chang et al [3] classified branches according to their taken rate, which is the number of times a given branch instruction is taken over the total number of times the branch is executed. Using profiling, they assigned branches to 6 different taken rate classes with percent taken ranges of 0-5%, 5-10%, 10-50%, 50-90%, 90-95%, and 95-100% respectively. They discovered that branches with taken rates of 10-90% were more accurately predicted with longer history lengths, but branches in the 0-5% and 95-100% classes were best predicted with very short history lengths. Given this information, they constructed a hybrid branch predictor that used static predictors for branches with the highest bias (0-5% taken or 95-100% taken) and either PAs or gshare for the remaining branches. By removing the most static branches from the pattern history table, interference was greatly reduced.

In Figure 1, we present the distribution for the branches in our benchmark suites divided into 11 equal branch classes. Each bar represents the number of branches in each class, weighted by their dynamic occurrence. Classes are numbered 0 through 10, representing percent taken ranges of 0-5%, 5-10%, 10-15%, etc. Since a majority of branches fall in the first or last branch classes, it makes sense that removing these branches from the predictor tables can decrease interference by a substantial amount.

Figure 3 shows the miss rates for both PAs and GAs predictors for each taken rate class when the optimal history length is used for each class. This shows that branches that have very high or very low taken rates are generally easier to predict than branches that have a weaker bias. Figures 5 and 6 are gray-scale colormap graphs of the PAs predictor miss rates for branches in each taken rate class when

Table 2. Percentage of dynamic branches in each taken rate and transition rate joint class, bold regions indicate branches wrongly classified as hard-to-predict if only taken rate is used.

	Taken Rate											
Trans. Rate	0	1	2	3	4	5	6	7	8	9	10	Total
0	26.11%	**0.71%**	**0.01%**	**0.05%**	**0.04%**	**0.02%**	**0.07%**	**0.32%**	**0.69%**	**0.05%**	32.73%	60.81%
1	0.46%	**2.12%**	**0.09%**	**0.09%**	**0.16%**	**0.06%**	**0.07%**	**0.03%**	**0.15%**	**4.00%**	3.59%	10.81%
2	0.00%	2.27%	0.45%	0.11%	0.03%	0.04%	0.99%	0.06%	0.57%	2.97%	0.00%	7.50%
3	0.00%	0.10%	1.01%	0.28%	0.13%	0.20%	0.24%	0.30%	0.87%	0.05%	0.00%	3.18%
4	0.00%	0.00%	0.36%	0.70%	1.08%	0.30%	1.72%	0.52%	0.60%	0.00%	0.00%	5.28%
5	0.00%	0.00%	0.01%	1.77%	0.72%	1.34%	0.16%	0.92%	0.56%	0.00%	0.00%	5.49%
6	0.00%	0.00%	0.00%	0.71%	1.59%	0.45%	0.89%	1.21%	0.00%	0.00%	0.00%	4.85%
7	0.00%	0.00%	0.00%	0.03%	0.13%	0.53%	0.11%	0.40%	0.00%	0.00%	0.00%	1.21%
8	0.00%	0.00%	0.00%	0.00%	0.21%	0.06%	0.02%	0.00%	0.00%	0.00%	0.00%	0.29%
9	0.00%	0.00%	0.00%	0.00%	**0.03%**	**0.07%**	**0.03%**	0.00%	0.00%	0.00%	0.00%	0.13%
10	0.00%	0.00%	0.00%	0.00%	0.00%	**0.44%**	0.00%	0.00%	0.00%	0.00%	0.00%	0.44%
Total	26.57%	5.20%	1.94%	3.76%	4.12%	3.53%	4.30%	3.77%	3.42%	7.06%	36.33%	

each history length is used. The dark areas represent larger miss rates. These graphs show the history lengths that worked best for branches in each taken rate class. The same information in more detail is provided in Figures 9 and 11 in the form of line plots for taken rate classes 0, 1, 9 and 10. While taken rate classes 0 and 10 should definitely be assigned short history lengths, it appears that other classes require longer history lengths. These results support those presented by Chang et al. [3].

4.2. Transition Rate Classification

Our investigation started with the observation that some easy-to-predict branches may be incorrectly classified by taken rate. If a branch execution stream contains long sequences of taken followed by equally long sequences of not-taken, such a branch will fall into a hard-to-predict class when classified using taken rate. However, it is fairly easy to see that such a branch can be well predicted using only a one-bit counter.

To better identify those branches which are easiest to predict with short histories, we chose to classify branches by transition rate. This measures how often a branch switches between taken and not-taken as it is executed. Again, we divided branches into 11 groups 0-10, representing 0-5%, 5-10%, 10-15% etc. Transition class 10 contains branches that transition 95-100% of the time.

Figure 2 shows the percentage of branches that fall into each transition rate class, again weighted by their dynamic occurrence. By definition, branches that are almost always taken will have very low transition rates and very high taken rates. Branches that have very low taken rates must also have very low transition rates. So clearly, transition rate must be able to identify the same easy-to-predict branches that taken rate can. The question, then, is how many branches have moderate taken rates but low transition rates? Also, what happens with branches that transition very frequently? Are these harder or easier to predict?

Figure 4 shows the miss rates for the PAs and GAs predictors when the optimal history length is used for each class. In this graph and in Figure 6, we find the answer to the last question. If a per-address predictor is used, branches in transition rate class 10 are very easily predicted using only a small amount of history. This makes sense, because a branch that transitions every time makes a short repeating pattern alternating between taken and not-taken. As expected, branches in class 0 are also well predicted for either predictor. Similar to taken rate classification, there is a steady decrease in prediction accuracy as transition rate approaches 50%.

In Figures 6, 8, 10 and 12 predictor miss rates are shown for each history length between 0 and 18 for transition rate classes. These show that transition rate classes 0 and 1 are well predicted with short history lengths with either PAs or GAs. With PAs, transition rate classes 9 and 10 are best predicted with short history lengths.

With zero history, branches in classes 9 and 10 are very poorly predicted because they are predicted based only on what they did the last time they were executed. Since a high transition rate means that they are usually changing direction from the previous execution, predicting the same direction as last time is almost always the wrong thing to do. Given only one or two bits of history, however, predictor performance is suddenly close to maximum for these branches.

A quick comparison between taken rate classification

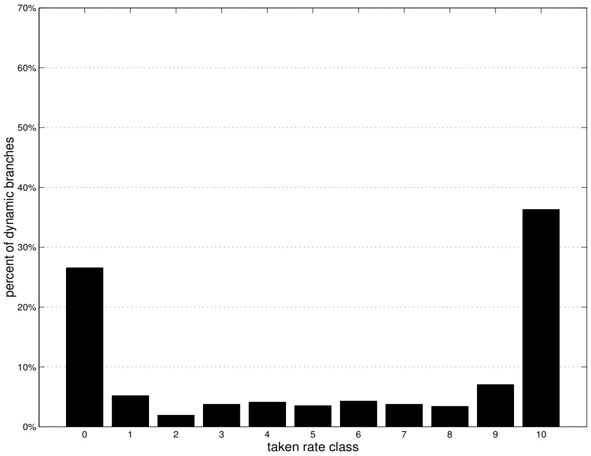

Figure 1. Percent of dynamic branches per taken rate class.

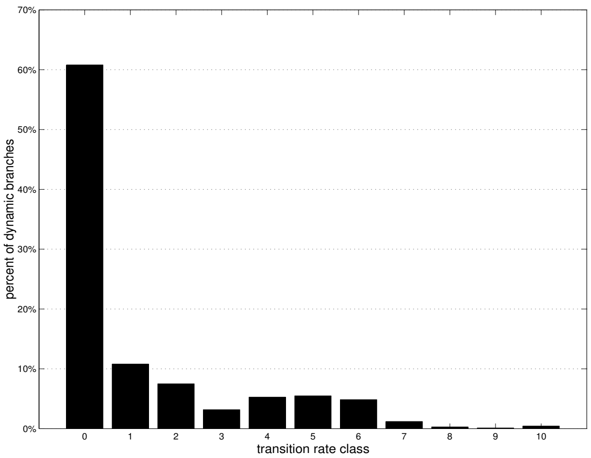

Figure 2. Percent of dynamic branches per transition rate class.

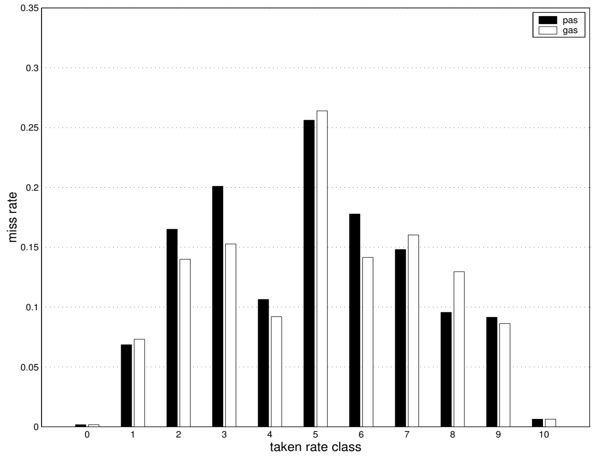

Figure 3. Miss rates by taken rate class for optimal history length per class.

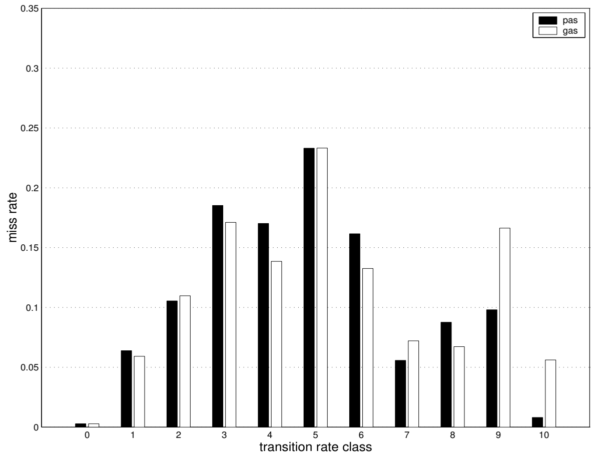

Figure 4. Miss rates by transition rate class for optimal history length per class.

and transition rate classification reveals that transition rate classification is slightly more advantageous. If the goal of such a classification is to identify and remove branches that are best predicted using short history lengths, transition rate can be used to locate more of these branches.

Suppose we accept taken rate branch classes 0 and 10 as mentioned earlier into this group of well predicted branches and assign them predictors that use little or no history. This is similar to the method employed by Chang et al. [3]. Then we have identified by taken rate 26.57 + 36.33 = 62.90 percent of the dynamic branches as belonging to this group. But using transition rate we can identify more branches that can benefit from lower history lengths, potentially improving prediction accuracy for these additional branches and further reducing interference if resources are freed for other branches. For GAs, transition rate classes 0 and 1 perform best with shorter history lengths. This totals 60.81 + 10.81 = 71.62 percent of the dynamic branches, showing that taken rate wrongly classifies 71.62 - 62.90 = 8.72 percent of the dynamic branches. And for PAs, we can identify by transition rate 60.81 + 10.81 + .13 + .44 = 72.19 percent. That is 9.29 above the 62.90 percent identified by taken rate, almost a 15% improvement in classification. The 9.29% of the total dynamic branches that are misclassified when using taken rate are highlighted in Table 2, which lists the percentage of dynamic branches in each joint class.

4.3. Combined Classification

Both taken rate and transition rate are easy to calculate for branch profiles, so there is no reason not to do both. Al-

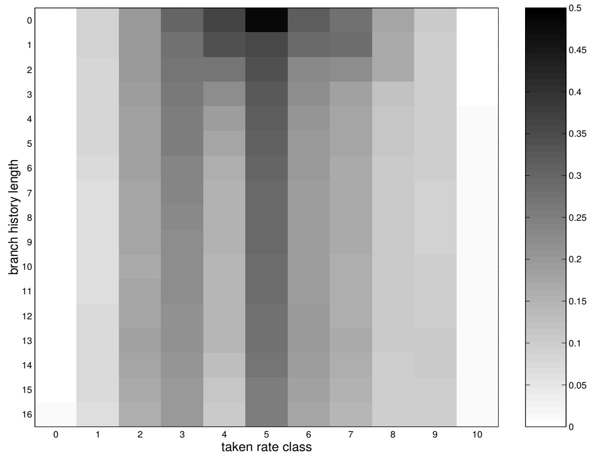

Figure 5. Miss rates for PAs by taken rate class and branch history length.

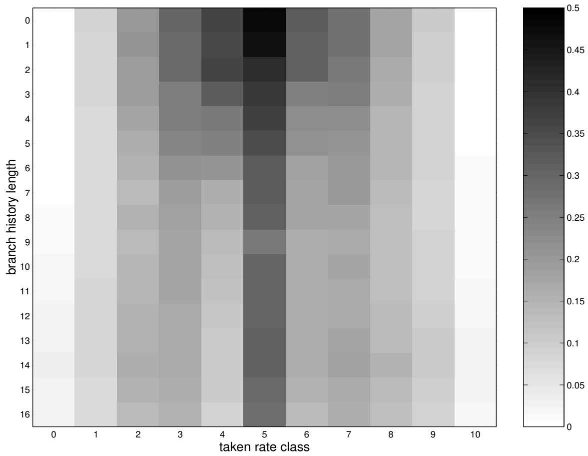

Figure 7. Miss rates for GAs by taken rate class and branch history length.

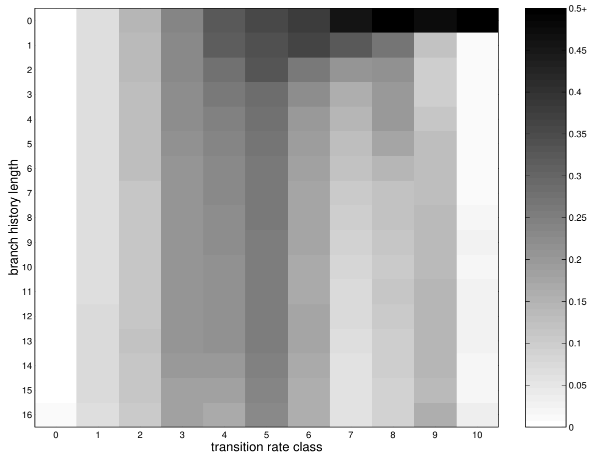

Figure 6. Miss rates for PAs by transition rate class and branch history length.

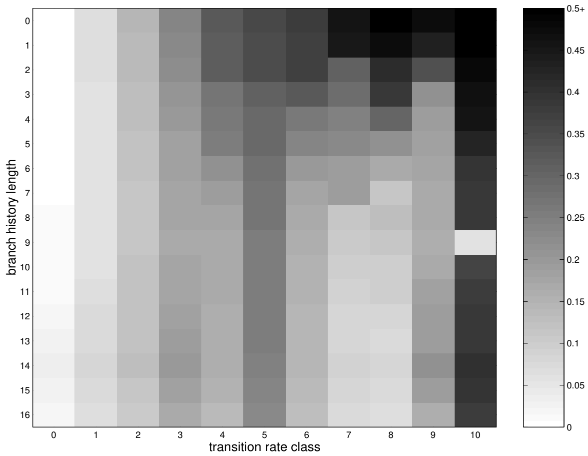

Figure 8. Miss rates for GAs by transition rate class and branch history length.

though there is some overlap, the metrics are not redundant. Classifying by both taken rate and transition rate provides some valuable insights into the behavior of branches and the performance of two-level branch predictors. Table 2 lists the percentage of static branches in each combined class, weighted by their dynamic occurrence. The distribution of branches appears to form a smooth arc connecting the static classes and passing through the 5/5 class.

Figures 13 and 14 show predictor miss rates for each taken and transition class given the optimal history length in the form of a gray-scale colormap. The most obvious feature of these graphs is the spot in the middle representing the near 50% prediction rate of both predictors for branches having taken and transition rates near 50%. Both types of predictors fared the worst on these branches.

Further work should be done to determine what kind of branches fall into this 5/5 class, and how predictor accuracy may be improved for these branches. The branches in this group may be data dependent branches for example, which represent a fundamental limit to predictor performance. These branches may be prime candidates for alternative techniques such as dual path execution, predication, or any software techniques which might reorder code in order to remove or make these branches more predictable. In fact, such techniques will probably only be as successful as the degree to which they are able to target these hard-to-predict branches.

Additional observations may be made from Figures 13 and 14. Note that, for the per-address predictor, the outer edge of the ”triangle” appears to be well predicted while

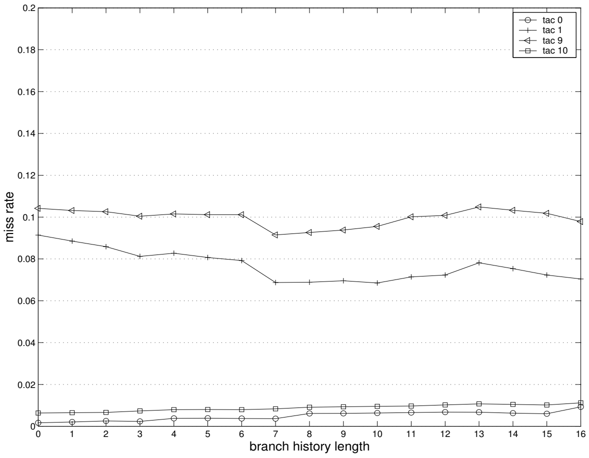

Figure 9. Miss rates for PAs by history length for taken classes 0,1,9, and 10.

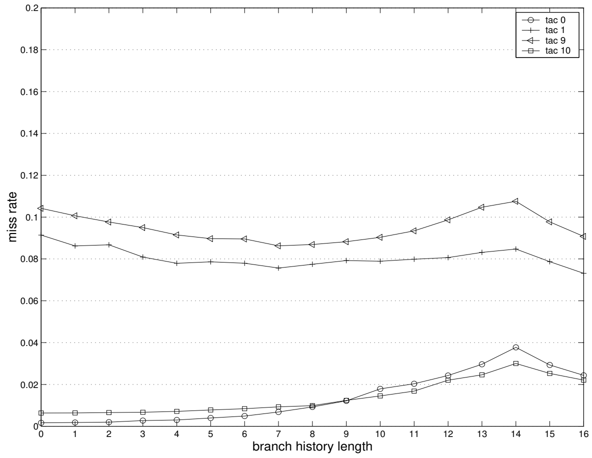

Figure 11. Miss rates for GAs by history length for taken classes 0,1,9, and 10.

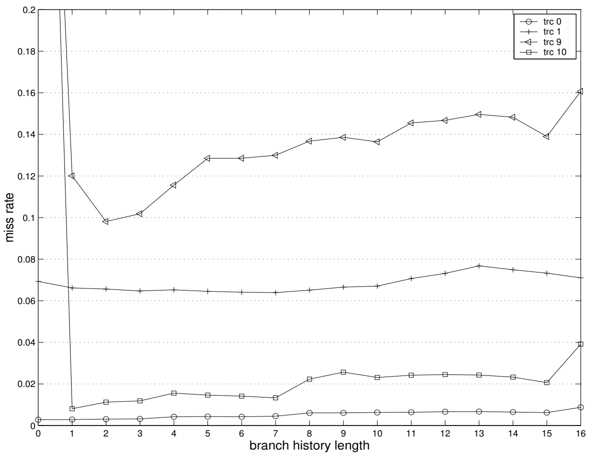

Figure 10. Miss rates for PAs by history length for transition classes 0,1,9, and 10.

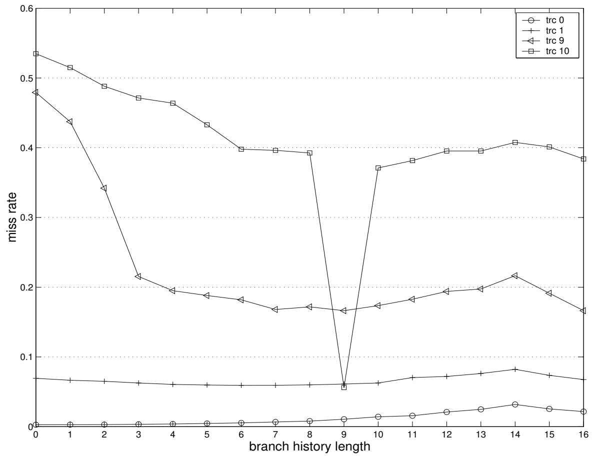

Figure 12. Miss rates for GAs by history length for transition classes 0,1,9, and 10.

the center of the triangle is the worst predicted. For both predictors, the 5/5 class has by far the worst miss rate, almost 50%. This shows that for most branches both taken rate and transition rate are important indicators of the expected prediction accuracy.

5. Opportunities for Predictor Improvements

The data presented on the transition rate and taken rate metrics for the first time clearly identifies and classifies branches that account for a majority of the misses in 2-level adaptive branch predictors. To improve the accuracy of current branch predictors, researchers need to find new methods for reducing the miss rate of these branches. Based on our prior analysis, we will suggest several methods.

5.1. Classification Methods

Branch classification will be an increasingly important aspect of future branch prediction schemes. Branches vary widely in their dynamic behavior, and predictors that work well on one type of branches may not work as well on others. Since interference poses a significant problem for state of the art branch prediction schemes, branches which require fewer resources need to be identified so that resources may be reallocated accordingly. This identification requires some form of classification, either through static profiling or some kind of dynamic method.

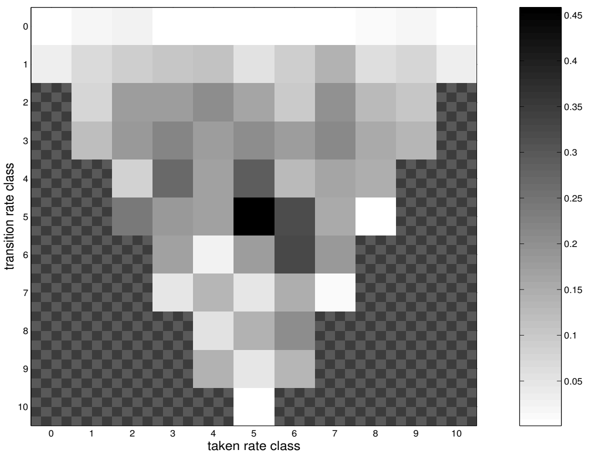

Figure 13. PAs miss rates for each joint class (for optimal history length per class)

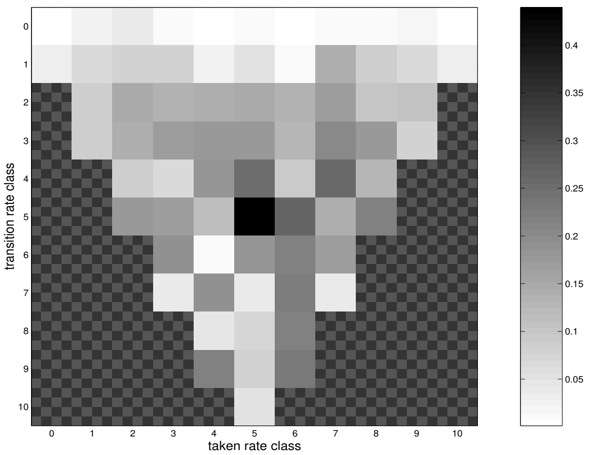

Figure 14. GAs miss rates for each joint class (for optimal history length per class).

5.2. Hard-To-Predict Branches

Through the use of transition rate and taken rate classification, we have identified a set of hard-to-predict branches that have near 50% taken and transition rates. These branches may comprise a fundamental limit to traditional 2-level adaptive branch predictors. Non-predictive techniques may be needed to handle these branches, such as predication, dual path execution, or other alternative methods.

5.2.1 Dual Path Execution

Dual path execution is a method that tries to reduce the penalty of mispredicting a branch by executing down both the taken and fall-through paths following a conditional branch. The disadvantage to this method is it is hardware prohibitive to attempt for every branch. Various research has been done to make dual path execution a viable method for reducing the misprediction penalty. Lick and Tyson [15] assigned a confidence level for each conditional branch, using it to decide whether to use dual path execution or the outcome from the branch predictor. Conditional branches that had a high misprediction rate (low confidence level) were executed down both paths. Similarly, Heil and Smith [9] used a confidence mechanism to make decisions on whether dual path execution was needed for each conditional branch. The PolyPath architecture described in [13] performs as a normal single path machine for conditional branches that are easily predicted and performs dual path execution for poorly predicted branches. These researchers all used some measure of prediction accuracy to determine whether to use dual path execution for each branch. In Section 5.3, we discuss how our taken/transition rate classification can be used as a measure of prediction accuracy for conditional branches.

A primary hindrance for success of dual path execution schemes results when branches chosen for dual path treatment occur too close together. In that case, the number of paths that must be simultaneously explored may multiply beyond a manageable limit.

To determine if dual path execution could be feasible for the hard-to-predict branches we identified, we calculated the relative distribution of branches from this class within a 8 branch window. At each occurrence of a hard-to-predict branch, the distance in dynamic branch executions to the previous hard-to-predict branch was measured, and a counter associated with that distance was incremented. These results are shown in Figure 15. With the exception of ijpeg, these branches seldom occur within a few branch instructions of each other.

By using transition rate in addition to taken rate, the number of branches that are truly hard-to-predict can be more closely identified, possibly making dual path a viable option for these branches.

5.2.2 Predication

Predication is a way of eliminating conditional branches. By predicating instructions guarded previously by a condition, a potential misprediction is avoided. Along with other considerations (such as nested branches, the number of branches guarded by a condition, etc), our classification scheme could help to identify those branches that would the best candidates for predication. For easily predicted branches (see Section 4), predication will probably not be useful and may lengthen execution time of the program. For example, those branches that are in taken rate class 1 and in transition rate class 1 would be poor candidates for predica-

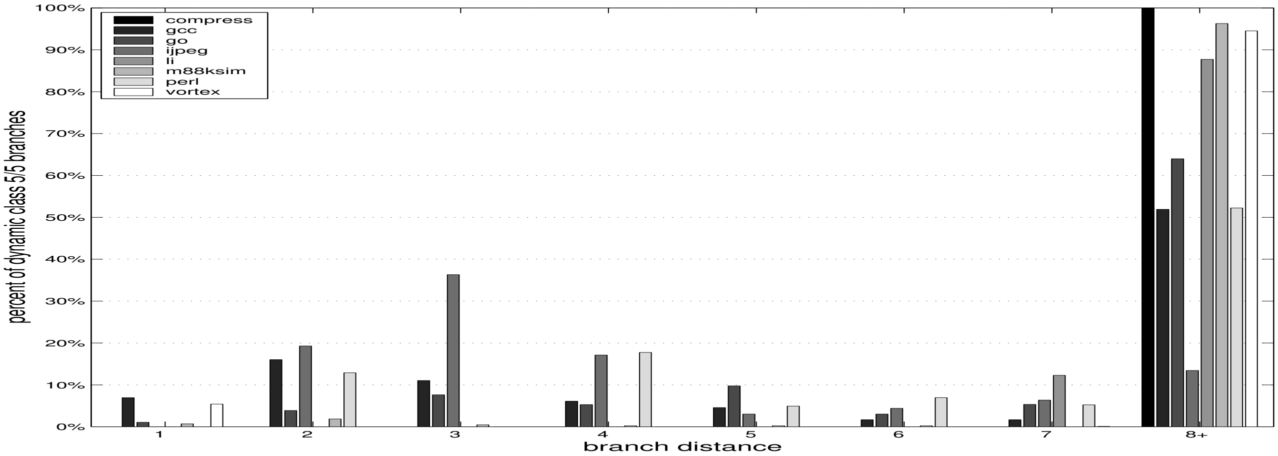

Figure 15. Relative distribution of class 5/5 branches.

tion. Predicating those instructions would greatly increase the number of instructions executed by the CPU (due to their dynamic number as seen in Figure 14) and substantially increase the running time of the program. However, the branches with near 50% taken and transition rates would be good candidates for predication. Predication could reduce mispredictions for these branches with only a slight increase in program running time, due to their low occurrence rate.

5.3. Confidence Levels

In section 5.2.1, we mentioned the importance of assigning confidences to the prediction of a conditional branch when considering whether to perform dual path execution. Grunwald et al. [7] cite the following applications as examples of needing confidence estimation: simultaneous multithreading, bandwidth multithreading, power conservation, eager execution, and improving branch predictors. In general, assigning confidence to branch predictions is a method for controlling speculation. In [10], they compare the performance and hardware costs of different confidence estimation methods and make suggestions for improvement.

Jacobsen et al. [10] classified branches by the dynamic execution rate and the branch misprediction rate. Using this classification they identify branches as having low confidence (hard to predict) or high confidence (easier to predict). They then describe two dynamic methods for confidence estimation: one level and two level.

Our results indicate that branch prediction accuracy is closely correlated with taken and transition rates. Confidence levels could be assigned to each branch according to its class. These rates might be used as a direct indicator of the confidence level to assign, without needing to measure prior predictor accuracy for each branch.

5.4. Hybrid Predictors

Through branch classification, we have observed that branches have different dynamic behavior. Some branches are predicted well by static methods or by 2-level adaptive branch predictors using a history length of a few bits. Others are predicted well using a global history, while still others are predicted well using a per-address history. It would be very difficult to create a single predictor that can perform optimally for every class of branches. Therefore, hybrid predictors are needed. Hybrid predictors can more easily be constructed to optimally handle the different classes of branches discussed in this paper.

The design space for developing new hybrid predictors is both large and complicated. Our work suggests that an ideal hybrid predictor would need to classify branches (either statically or dynamically), provide for both global and per-address histories, and vary history lengths per branch.

By classifying branches as described in this paper, the design space can be explored more efficiently and succinctly. Decisions about the composition of the hybrid predictor can be directly correlated to the classes of branches generated by our technique. For example, we have demonstrated that the optimal history length for predicting a branch is dependent upon its taken and transition rate class, and the performance of global versus per-address predictors on different classes has been shown. Also, the dynamic occurrence of branches in each category is useful in deciding the relative size of each component of the predictor and whether to predict statically or dynamically.

6. Conclusions and Future Work

We have introduced a new metric for branch classification, *branch transition rate*, and showed that it can increase the effectiveness of current classification schemes which only employ *taken rate*. When both transition rate and taken rate metrics are used an even clearer understanding of branch behavior can be realized. We believe that classification methods will become an increasingly important aspect of future branch prediction schemes, performed either by static profiling or by dynamic methods. It is our hope that the analysis done in the paper will be useful in many aspects of branch prediction. Confidence assignments, predication, and dual path execution are branch prediction methods that we feel could potentially benefit from this kind of classification.

This research has identified a small set of problem branches. What program structures cause these branches to occur? Are these branches data dependent branches, and are they predictable or avoidable through other techniques? We are currently trying to address these questions.

Further study is also needed to explore ways of implementing transition rate classification methods in prediction schemes. The information presented here can be readily used in schemes which employ profiling. It may also be possible to perform classification based on transition rate using some form of dynamic counter. If pattern history is already maintained for each branch, it would be easy to also maintain the local transition and taken rates for this history window. Otherwise, some kind of dynamic transition rate counter could be employed. It remains to be investigated whether improvements in accuracy gained from such additions would support the hardware cost necessary to implement them. Our desire is that the information presented here will assist other researchers build better predictors through a deeper understanding of how branches behave and why predictors work.

References

[1] T. Austin and D. Burger. *The SimpleScalar Tool Set, Version 3.0*, 1998.

[2] P.-Y. Chang, M. Evers, and Y. Patt. Improving branch prediction accuracy by reducing pattern history table interference. In *Proceedings of the Intl. Conf. on Parallel Architectures and Compilation Techniques*, October 1996.

[3] P.-Y. Chang, E. Hao, T.-Y. Yeh, and Y. Patt. Branch classification: A new mechanism for improving branch predictor performance. In *Proceedings of the 27th Annual ACM/IEEE Intl. Symposium on Micro.*, pages 292 – 302, 1994.

[4] I.-C. K. Chen, J. Coffey, and T. Mudge. Analysis of branch prediction via data compression. In *Proceedings of the 7th Intl. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS VII)*, pages 128–137, Cambridge, MA, October 1996.

[5] A. Eden and T. Mudge. The yags branch prediction scheme. *IEEE*, 1998.

[6] M. Evers, S. Patel, R. Chappell, and Y. Patt. An analysis of correlation and predictability: What makes two-level branch predictors work. In *Proceedings of the 25th Annual Intl. Symposium on Computer Architecture*, pages 52–61, 1998.

[7] D. Grunwald, A. Klauser, S. Manne, and A. Pleszkun. Confidence estimation for speculation control. In *25th Intl.. Symposium on Computer Architecture*, June 1998.

[8] D. Grunwald, D. Lindsay, and B. Zorn. Static methods in hybrid branch prediction. In *Proceedings of the Intl. Conf. on Parallel Architecture and Compilation Techniques*, 1998.

[9] T. Heil and J. Smith. Selective dual path execution. Technical report, University of Wisconsin-Madison, Nov 1996.

[10] E. Jacobsen, E. Rotenberg, and J. Smith. Assigning confidence to conditional branch predictions. In *Proceedings of the 29th Annual Intl. Symposium on Microarchitecture*, pages 142–152, Paris, France, December 1996.

[11] T. Juan, S. Sanjeevan, and J. Navarro. Dynamic history-length fitting: A third level of adaptivity for branch prediction. In *Proceedings of the 25th Annual Intl. Symposium on Computer Architecture*, pages 155–166, 1998.

[12] S. Kim and G. Tyson. Analyzing the working set characteristics of branch execution. *IEEE*, 1998.

[13] A. Klauser, A. Paithankar, and D. Grunwald. Selective eager evaluation on the polypath architecture. *ISCA*, 1998.

[14] C.-C. Lee, I.-C. Chen, and T. Mudge. The bi-mode branch predictor. In *Proceedings MICRO 30*, December 1997.

[15] K. Lick and G. Tyson. Hybrid branch prediction using limited dual path execution. Technical report, University of California, Riverside, Riverside, CA, July 1996.

[16] S. Mahlke, R. Hank, R. Bringmann, J. Gyllenhaal, D. Gallagher, and W. Hwu. Characterizing the impact of predicated execution on branch prediction. In *Proceedings of the 27th Intl. Symposium on Microarchitecture*, pages 217–227, December 1994.

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

[18] P. Michaud, A. Seznec, and R. Uhlig. Trading conflict and capacity aliasing in conditional branch predictors. In *Proceedings of the 24th Annual Intl. Symposium on Computer Architecture*, May 1997.

[19] E. Sprangle, R. Chappell, M. Alsup, and Y. Patt. The agree predictor: A mechanism for reducing negative branch history interference. In *Proceedings 24th Annual Intl. Symposium on Computer Architecture*, May 1997.

[20] J. Stark, M. Evers, and Y. Patt. Variable length path branch prediction. In *Proceedings of the 8th Intl. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS VIII)*, Cambridge, MA, October 1998.

[21] T.-Y. Yeh and Y. Patt. Two-level adaptive branch prediction. In *Proceedings of the 24th Annual ACM/IEEE Intl. Symposium on Microarchitecture*, 1991.

[22] T.-Y. Yeh and Y. Patt. Alternative implementations of two-level adaptive branch prediction. In *Proceedings of the 19th Annual Intl. Symposium on Computer Architecture*, pages 124–134, 1992.

[23] C. Young, N. Gloy, and M. Smith. A comparative analysis of schemes for correlated branch prediction. In *Proceedings 22nd Annual Intl. Symposium on Computer Architecture*, pages 276–286, Santa Magherita Ligure, Italy, June 1995.