An Analysis of Correlation and Predictability: What Makes Two-Level Branch Predictors Work

Marius Evers Sanjay J. Patel Robert S. Chappell Yale N. Patt
Department of Electrical Engineering and Computer Science
The University of Michigan, Ann Arbor, MI 48109-2122
email: {olaf,sanjayp,robc,patt}@eecs.umich.edu

Abstract

Pipeline flushes due to branch mispredictions is one of the most serious problems facing the designer of a deeply pipelined, superscalar processor. Many branch predictors have been proposed to help alleviate this problem, including two-level adaptive branch predictors and hybrid branch predictors.

Numerous studies have shown which predictors and configurations best predict the branches in a given set of benchmarks. Some studies have also investigated effects, such as pattern history table interference, that can be detrimental to the performance of these predictors. However, little research has been done on which characteristics of branch behavior make predictors perform well.

In this paper, we investigate and quantify reasons why branches are predictable. We show that some of this predictability is not captured by the two-level adaptive branch predictors. An understanding of the predictability of branches may lead to insights ultimately resulting in better or less complex predictors. We also investigate and quantify what fraction of the branches in each benchmark is predictable using each of the methods described in this paper.

1. Introduction

To build high performance microprocessors, accurate branch prediction is required. The correct prediction of branch outcomes and targets is necessary to avoid pipeline bubbles. Over the years, several branch prediction strategies have been proposed [8, 11, 2] to improve prediction accuracy. Several researchers have proposed modifications [2, 3, 7] to these schemes, and there have been studies on which configurations of these work best [6]. However, there has been little work that explains what makes branches predictable. A better understanding of this would likely lead to insights ultimately resulting in better predictors, or in reducing the complexity of current predictors.

Global two-level branch predictors such as GAs [11] and gshare [2] take advantage of the correlation between branches. Pan, So, and Rahmeh [4] identified several cases of such correlated branches in the SPEC89 benchmarks. If two branches are correlated, knowing the outcome of the first branch gives you information about which direction the second branch is likely to take. Consider the example:

```
if (cond1)
...
if (cond1 AND cond2)
```

The first branch depends on a condition (cond1), and the second branch later in the program depends on a condition (cond1 AND cond2) which is related to the condition of the first branch. If the first branch is not taken, we know that the second branch will not be taken. If the first branch is taken, we now know that the second branch only depends on the condition cond2. Clearly, these two branches are correlated even though the outcome of the second branch does not fully depend on the outcome of the first branch. In section 3 of this paper, we will examine and quantify the nature of correlation between branches. We will link this back to the performance of gshare, which is generally considered to be the best performing global two-level branch predictor. We will then show that there is a significant amount of branch correlation that gshare fails to exploit.

In addition to correlation with other branches, many branches are predictable based on the previous outcomes of the branch itself. This is the predictability exploited by per-address two-level branch predictors such as PAs [10]. Branches predictable in this way include loop branches with a regular number of iterations and branches that follow a periodic pattern (such as being taken every other time). As in the branch correlation case, it is possible for a branch to be only partly predictable based on its own history. We will examine branches that are predictable based on their own history in section 4.

Since the best performing branch predictors today are hybrid predictors containing both global and per-address components, we categorize branches in section 5 based on whether they are more predictable using global branch correlation or per-address based methods. We also identify the set of branches where neither global correlation based nor per-address based methods do better than statically predicting the predominant direction of the branch.

In this study, we build on existing work on branch correlation. We try to identify why predictors work well, and identify some areas where the current predictors are not uncovering the predictability that exists.

2. Previous work

2.1. Branch prediction mechanisms

To improve prediction accuracy, various branch prediction strategies have been studied. Smith [8] proposed a branch prediction scheme that uses a table of 2-bit saturating up-down counters to keep track of the direction a branch is more likely to take. Each branch is mapped via its address to a counter. The branch is predicted taken if the most significant bit of the associated counter is set; otherwise, it is predicted not-taken. These counters are updated based on the branch outcomes. When a branch is taken, the 2-bit value of the associated saturating counter is incremented by one; otherwise, the value is decremented by one.

By keeping more history information, a higher level of branch prediction accuracy can be attained [10, 11]. Yeh and Patt proposed the Two-Level Branch Predictor which uses two levels of history to make branch predictions. The first-level history records the outcomes of the most recently executed branches and the second-level history keeps track of the more likely direction of a branch when a particular pattern is encountered in the first level history. The Two-Level Branch Predictor uses one or more k-bit shift registers, called branch history registers, to record the branch outcomes of the most recent k branches. If there is one history register per branch, the predictor is called a per-address (PAs) predictor. If there is one history register to record the outcomes of all branches, it is called a global (GAs) predictor. The two-level predictor uses one or more arrays of 2-bit saturating up-down counters, called Pattern History Tables, to keep track of the more-likely direction for branches. The lower bits of the branch address is used to select the appropriate Pattern History Table(PHT) and the contents of the branch history register select the appropriate 2-bit counter to use within that PHT.

Several variations of the Two-Level Branch Predictor have been proposed. McFarling [2] introduced gshare, a variation of the global-history Two-Level Branch Predictor that XORs the global branch history with the branch address to index into the PHT. This leads to better utilization of the PHT. Nair [3] proposed using a path history instead of a pattern history to index into the PHT. This has the advantage of being able to represent the path, albeit imperfectly. It has the disadvantage that information from fewer branches can be captured in the history.

To further improve prediction accuracy, hybrid branch predictors have recently been proposed [2]. A hybrid branch predictor is composed of two or more predictors and a mechanism to select among them. A hybrid branch predictor can exploit the different strengths of its component predictors, enabling it to achieve a prediction accuracy greater than that achieved by any of its components alone.

2.2. Studies on effects seen in branch predictors

There have also been some studies on the behavior of branches and branch predictors. Pan et al. [4] identified several cases of branches being correlated in the source code of the SPEC89 benchmarks.

Chang et al. [1] classified branches based on their taken rates. They proposed a predictor using a static predictor for the strongly biased branches, and a dynamic hybrid predictor for the weakly biased branches.

Sechrest et al. [5] studied the role of adaptivity in two-level branch predictors and determined that, for per-address predictors with short histories, having statically determined values in the PHT performed on par with the adaptive scheme using 2-bit counters.

Talcott et al. [9] and Young et al. [12] studied and classified the effects of pattern history table interference, and showed that it negatively affected the performance of 2-level branch predictors. These two papers used interference-free predictors to aid in the understanding of the potential of 2-level predictors. An interference-free predictor has one PHT for each branch and is therefore prohibitively large, but does not suffer from the negative effects of PHT interference.

Young et al. [12] also showed the advantage of path histories over pattern histories for static branch prediction. Furthermore, they investigated the importance of adaptivity in the PHTs of global 2-level predictors and found that, in some cases, a statically determined PHT, when using the same profiling and testing set, would outperform a PHT using 2-bit counters.

3. Branch correlation

Global two-level branch predictors such as GAs and gshare take advantage of correlation between branches. In this section, we identify several classes of correlation, describe how correlation is detected, and present results quantifying the amount of branch correlation in the SPECint95 benchmarks.

3.1. The general case

There are two reasons for the directions of two branches to be correlated. One is that the conditions of the two branches are based (fully or partly) on the same or related information. An example of this is shown in figure 1a. The other reason is that information affecting the outcome of the second branch is generated based on the outcome of the first branch. An example of this is shown in figure 1b. Since the outcome of the second branch is correlated to the direction of the first branch, we will refer to these two kinds of correlation as $direction$ correlation.

(a)	(b)
branch Y: if (cond1) ... branch X: if (cond1 AND cond2)	branch Y: if (cond1) a = 2; ... branch X: if (a == 0)

```
branch Y: if (cond1)
...
branch Z: if (cond2)
...
branch X: if (cond1 AND cond2)
```

(c)

Figure 1. Correlation examples

In both of these examples, the later branch (branch X) is the one that we are trying to predict. In the rest of this paper, we will refer to this as the $current$ branch. We will refer to the preceding branches (branch Y and Z) whose outcome is correlated with the current branch as the $correlated$ branches. In this study, we will only look for correlated branches within a history of the last n branches leading up to the current branch, where n is in the range 8-32 depending on the experiment.

```
branch Y: if (NOT(cond1)) ...
branch Z: else if (NOT(cond2)) ...
branch V: else if (cond3) ...
...
branch X: if (cond1 AND cond2)
```

Figure 2. Example of in-path correlation

In addition to the direction of the correlated branch, just knowing whether we arrived at that branch on the path leading up to the current branch will give us some information about the outcome of branches preceding the correlated branch. This is illustrated by figure 2. In this case, if we get to branch V, we know that the first two conditions were false, so cond1 and cond2 are both TRUE. Note that the direction of branch V is not related to the condition of branch X, but from knowing that branch V was on the path to branch X we know that the condition of branch X will be satisfied. If a branch is one of the last n branches leading up to the current branch, we will say that it was in the path to the current branch. We will refer to the correlation between a branch being in the path and the outcome of the current branch as $in\text{-}path$ correlation. $In\text{-}path$ correlation is exploited more directly in path based global predictors than in pattern based predictors. If the current branch is at the beginning of a subroutine, its outcome may depend on where the subroutine was called from. $In\text{-}path$ correlation would also account for this effect.

In many cases, the correlation between one pair of branches, such as the pairs shown in figure 1a and 1b, is not strong enough to guarantee the direction of the second branch. Sometimes the correlation is strong only if the correlated branch is taken (or not taken); if the correlated branch is not taken, the correlation is not strong enough to guarantee the direction of the second branch. In these cases, we need to look at the correlation between several branches and the current branch. In the example shown in figure 1c, branch X will be taken if both Y and Z are taken. It will not be taken if either Y or Z is not taken. We do not know whether X will be taken if neither Y nor Z is in the path. Later in this section, we investigate how the predictability of branches increases as a function of the number of correlated branches that are used to determine the prediction.

3.2. Accounting for loop behavior

The discussion in section 3.1 examined the nature of correlation between branches with the assumption that the address of a branch is sufficient to distinguish branches for correlation purposes. However, in tight loops, several iterations can sometimes fit in the history of n branches we are examining, so we must distinguish between multiple instances of the same branch.

There are two straightforward methods to distinguish between multiple instances of a previous branch. For each of these, the branch will be identified by its address along with a "tag" that represents a particular dynamic instance of that static branch.

One of these methods is to number the instances of a branch starting at the current branch. So, if branch A appears in the history 3 times, the most recent occurrence would be A_0, the second most recent would be A_1 and the oldest would be A_2. However, with this method there is no way to clearly identify branch A from a specific iteration of a loop if it does not appear in every iteration.

The other method is to number the instance of a branch by how many backwards branches have occurred between it and the current branch. This enables us to clearly identify the instance of the branch from a certain number of iterations ago. However, with this method you can not easily identify branches from before the current loop. A branch before the beginning of the loop will be tagged differently depending on how many iterations of the loop have passed. [1]

Each of these methods of identifying the instance of a branch have different limitations. Therefore, we tagged all branches using both methods. We considered branches tagged using the two different methods as distinct instances when investigating correlation with the current branch.

3.3. Correlation in two-level branch predictors

Of the branches in the path leading up to the current branch, there will be some that are correlated and some that are not. Ideally, we want to build a history including the outcomes of the branches that are correlated, but excluding the outcomes of branches that are not correlated.

Global two-level branch predictors, such as the gshare predictor shown in figure 3, are able to exploit correlation by basing the prediction on the outcomes of all the recently executed branches. The most recent branch outcomes are recorded in the first level of history, and the second level of history records the most likely outcome when a particular pattern in the first level history is encountered.

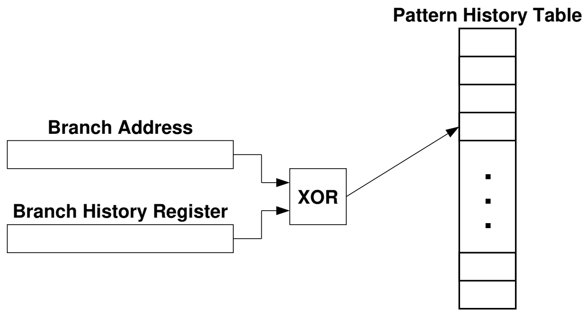

Figure 3. Diagram of a gshare predictor

For each of the branches in the history, if that branch is taken, it will generate a different pattern than if that branch is not taken. These patterns will then use different entries in the second level history table allowing better predictions to be made.

However, not all of the outcomes in the branch history register contain information that is useful for prediction. That is, some of the branches that are recorded in the history are not correlated to the current branch. Different outcomes of these branches will still cause different patterns to be used, but with no beneficial effect on prediction accuracy. However, the added noise, resulting in more interference and longer training times, may have a negative effect.

[1] This could be fixed by only counting the number of backwards branches that branched past the branch in question, but due to subroutine calls, this is difficult to determine

3.4. Correlation using a selective history

To investigate how many of the entries in the history are really needed, we defined a hypothetical predictor. This predictor works in a manner similar to a global 2-level predictor, but only the outcomes of the 1, 2 or 3 most important branches, tagged as described in section 3.2, are included in the history. For this $selective$ history, we used an oracle mechanism to choose the set of 1, 2 or 3 most important branches to include in the history for each branch.

The outcome of each of these branches is recorded in the history as taken, not taken or not in the path of n branches leading up to the current branch. The "not in path" outcome was required in this hypothetical predictor as we are looking at 1, 2 or 3 particular branches, and not all of these appear in the recent path all the time.

The history with 1 branch can have 3 possible patterns (taken, not taken or not exist), the history with 2 branches can have 3^2 patterns, and the history with 3 branches can have 3^3 patterns. Predicting using these history patterns is then done identically to predicting in a global 2-level predictor. The pattern is used to select a counter in the second level table. The upper bit of this counter provides the prediction, and the counter is updated with the outcome of the branch.

3.5. Simulation environment

We simulated the SPECint95 benchmarks to completion using a trace-driven branch prediction simulator. Table 1 lists the 8 SPECint95 benchmarks, the data sets we used, and how many dynamic conditional branches were in each run.

Benchmark	Input	# of Branches
compress	test.in[2]	10661855
gcc	jump.i	25903086
go	2stone9.in[2]	17925171
ijpeg	specmun.ppm[2]	20441307
m88ksim	dcrand.train.big	16719523
perl	scrabbl.pl[2]	10570887
vortex	vortex.in	33853896
xlisp	train.lsp	26422387

Table 1. Summary of the SPECint95 benchmarks along with the input data sets

[2] Abbreviated version of the SPECint input set

3.6. Experimental results

In this section, we examine the prediction accuracy using a selective history of one, two, or three branches as described in section 3.4. We show that only a few branches need to be recorded in the history in order to achieve prediction accuracy comparable to gshare, as long as the most important branches are recorded. We are examining 16 prior branches in all experiments unless otherwise noted.

We also examine the distance to the correlated branches that are most important to record in the history. Finally, we demonstrate that there is a significant amount of correlation that gshare, which is generally considered the best correlation based two-level predictor, does not capture.

3.6.1 Correlation with multiple branches

Some branches are strongly correlated with one previous branch, so only information about that branch is needed to predict the current branch. For other branches, we need to capture the correlation with several prior branches to make an accurate prediction. Figure 4 shows the prediction accuracy for each benchmark using a selective history of only the 1, 2 or 3 most important branches. This is compared to the prediction accuracy of an interference-free gshare predictor using a 16 branch history. The accuracy of a regular gshare predictor is shown for reference.

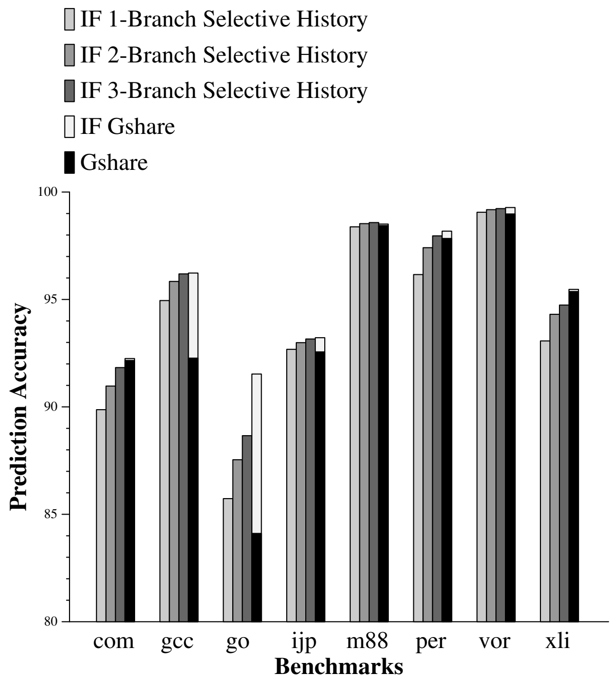

Figure 4. Selective history vs. gshare and interference-free gshare

Most of the benchmarks have the same trend. Even with a selective history containing only one branch, meaning that only correlation with that one branch could be exploited, the prediction accuracy is at a respectable level. When using a selective history of 3 branches, the prediction accuracy is close to the accuracy of an interference-free gshare for most of the benchmarks. This is an important point. The interference-free gshare predictor is using the outcomes of all of the 16 most recent branches to make its prediction. However, this does not result in much better prediction accuracy. Using all 16 outcomes when only a few are needed introduces undesired noise. This noise impacts a gshare predictor in two ways. One is added interference (obviously not a factor in the interference-free gshare). The other is increased training time.

The obvious conclusion that can be drawn from this experiment is that the outcomes of only a very small number of previous branches is needed to make an accurate prediction, as long as the most important branches can be identified.

3.6.2 Distance to correlated branches

We showed in the previous section that only a small selective history is needed to make an accurate prediction. In that experiment, the 16 most recent branches were considered in forming the selective history. In this experiment, we examine how far back the important branches are by considering the n most recent branches, where n is varied from 8 to 32. We will refer to n as the history length. Clearly, the closer the most important branches are to the current branch, the easier it would be to exploit the correlation in a predictor implementation. Figure 5 shows the prediction accuracy using a selective history of 3 branches for history lengths going from 8 to 32 branches in intervals of 4.

The figure shows that examining a history of fewer than 12 branches is limiting. There is a slow but steady growth

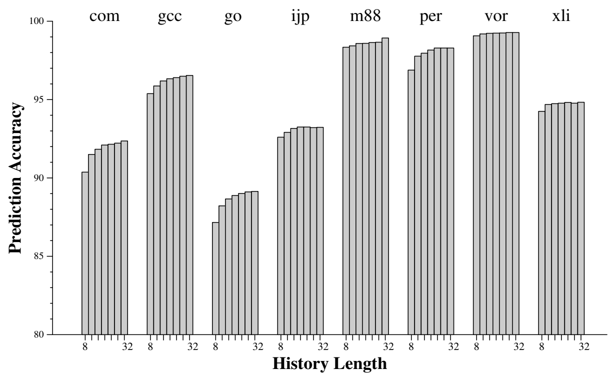

Figure 5. Accuracy as a function of history length using a 3-branch selective history

from 12 up to a history length of 20, but little gain in looking farther back. This indicates that the most correlated branches are close together. The increase in prediction accuracy when going from 12 to 20 entries in the history is approximately 0.5% for compress, gcc and go, and there is almost no increase for m88ksim, vortex and xlisp. However, the improvement in just going from a history length of 12 to 13 for a gshare predictor would be more than this. This is consistent with the notion that the main effect of increasing the history length in a gshare predictor is to reduce interference.

3.6.3 Correlation not exploited by gshare

The predictor using a 1-branch selective history was less accurate than gshare for half of the benchmarks. However, this predictor uses only a single-branch selective history and still outperforms gshare for two benchmarks and equals gshare for the remaining two. Although gshare has the potential to utilize correlation with 16 branches, it appears from this result that sometimes it fails to capture the correlation with the one branch used in the selective history.

To quantify the inability of gshare to utilize this correlation, we created a hypothetical predictor. This predictor uses the 1-branch selective history predictor for branches where it achieves a higher accuracy than gshare. Otherwise, gshare is used. If gshare could consistently exploit the correlation with the one branch used in the selective history, the accuracy of this hypothetical predictor should be the same as the accuracy of gshare.

As shown in table 2, the prediction accuracy of this hypothetical predictor, shown as "gshare w/ Corr" was approximately 4% higher than that of gshare for gcc and go and on average 0.23% higher for the other benchmarks. This shows that gshare is not always able to exploit even the one strongest correlation for each branch.

We also performed the same experiment with a similar hypothetical predictor using an interference-free gshare. As shown in table 2, the prediction accuracy of this hypothetical predictor was 0.5-0.6% higher than that of interference-free gshare for gcc and go (representing 13% of the mispredictions for gcc and 7% of the mispredictions for go) and on average 0.1% higher for the other benchmarks. This indicates that although interference limits the ability of gshare to exploit available correlation, other factors such as increased training time are also keeping gshare from fully exploiting even the one strongest correlation for each branch.

Benchmark	gshare	gshare w/ Corr	IF gshare	IF gshare w/ Corr
compress	92.16	92.40	92.25	92.41
gcc	92.27	95.95	96.23	96.73
go	84.11	88.54	91.53	92.14
ijpeg	92.56	93.12	93.22	93.31
m88ksim	98.44	98.58	98.51	98.59
perl	97.84	98.29	98.18	98.34
vortex	98.98	99.29	99.28	99.32
xlisp	95.37	95.52	95.47	95.52

Table 2. Accuracy of gshare w/ and w/o additional correlation

4. Per-address predictability

This section discusses branches that are predictable based on the recent outcomes of the branch itself.

4.1. Classes of per-address predictability

We identified three classes of per-address based predictability: loop-type branches, branches having repeating patterns, and branches having non-repeating patterns.

For each of these three classes, we used a predictor based on the premise of that class. It follows that any branch having behavior indicative of one of these classes will be very well predicted by the predictor for that class. Therefore, it is reasonable to use the prediction accuracies of these predictors to classify branches. We consider a branch as belonging to the class for which the prediction accuracy was the highest. The specific predictors for the classes are described in the sections below.

However, there are some branches that are either very strongly biased or do not exhibit the characteristics of any of these classes. For these branches, using the predictors based on the premises of these classes does no better than predicting a single direction throughout the program, where that direction is the direction taken most often by the branch during the run. This is the best one can achieve with a static predictor, hence we refer to it as the "ideal" static predictor. Branches for which the ideal static predictor is best are not considered to belong to any of the classes. It is not necessarily true that these branches can not be predicted better by a different dynamic predictor. However, they can not be predicted better using the methods described here.

4.1.1 Branches having loop-type behavior

The *loop-type* class contains "for-type" and "while-type" branches. For-type branches are taken n times followed by not-taken once. While-type branches are not-taken n times followed by taken once. n is expected to stay the same or change infrequently.

We designed a predictor that captures this loop-type behavior. It makes n predictions in a row of one direction, then a single prediction of the opposite direction. The value of n is determined from the previous number of consecutive same-direction outcomes. Since the for-type and while-type branches only differ in the directions predicted, the predictor maintains a direction bit to differentiate between the two. The counts of n are kept in a perfect BTB to prevent interference from affecting our classification. For these experiments, we assumed $n < 256$.

4.1.2 Branches having repeating patterns

The *repeating pattern* class consists of branches that follow a repeating pattern of outcomes, but are more general than loop branches. We group this class of branches into two subsets. The first subset, "fixed-length patterns", contains branches repeating any arbitrary pattern of outcomes of length k. The second subset, "block patterns", contains branches that are taken n times, then not-taken m times, then taken n times, and so on.

To capture the fixed-length pattern subset, we constructed a set of predictors. Since a fixed length pattern of length k repeats every k branches, then each branch should have the same outcome as k times ago. A predictor based on capturing this type of behavior need only predict the same direction as the branch took k times ago. We simulated 32 different variations of this predictor that each predicts using a different value of k between 1 and 32. For each branch, the best prediction accuracy of these 32 predictors was used as the fixed-length prediction accuracy.

The predictor for the block pattern subset was implemented similarly to the loop predictor in the previous section. It predicts that, after the nth consecutive taken branch, the branch will be not-taken the same number of times (m) as before the first of these taken branches. Similarly, after the mth consecutive not-taken branch, the prediction is that the branch will be taken the same number of times (n) as before the first of these not-taken branches. For these experiments, we assumed $n < 256$, $m < 256$, and stored the counts in a perfect BTB.

When comparing against the other classes, we used the higher accuracy of the subset predictors.

4.1.3 Branches having non-repeating patterns

The *non-repeating pattern* class consists of branches that do not have a repeating pattern, but are still predictable based on their previous outcomes. These are branches with outcomes that can be predicted based on specific previous outcomes in the history. Many data dependent branches fall into this category, as the input to a program commonly has some pattern to it. PAs is a predictor that works on this premise. To prevent interference from affecting our classification, we used an interference-free PAs predictor with a very large BTB as a predictor for this class.

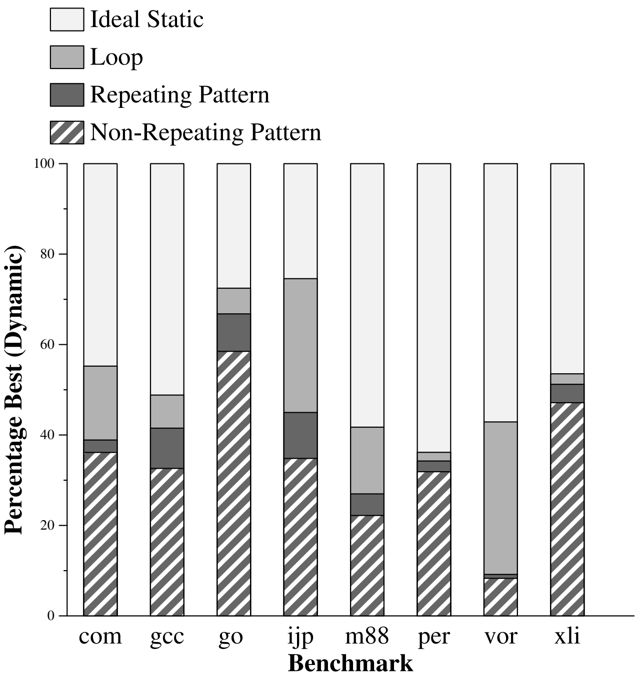

Figure 6. Fraction of branches each per-address class was dominant weighted by the dynamic execution frequencies of the branches

4.2. Experimental results

4.2.1 Distribution of per-address predictability classes

In figure 6, we show how branches fall into each class of per-address predictability. We also show the set of branches where the accuracy of an ideal static predictor is greater than or equal to the predictability of any of the classes.

Figure 6 shows that about half of the branches, those at least equally well predicted using an ideal static predictor, were not classified as belonging to any of the specific classes of per-address predictability. 88% of these branches are more than 99% biased, while the rest are simply not predictable using any of the per-address based methods described in this paper.

Approximately a third of the branches were classified as having non-repeating patterns, meaning that a 2-level per address predictor is needed to predict them. Most of the remaining branches, about a sixth, were classified as loop-type. Repeating patterns that were not captured in the two loop-type categories were infrequent.

4.2.2 Unexploited per-address predictability in PAs

We showed in the previous experiment that loop-type branches accounted for almost one sixth of the branches in the SPECint95 benchmarks. These branches are predicted better by a loop predictor than PAs, but the previous experiment does not show how much better.

To examine whether the behavior of these loop-type branches is captured sufficiently well by PAs, we created a hypothetical predictor. This hypothetical predictor uses the loop predictor described earlier for all branches in the loop class, and PAs for all other branches.

As shown in table 3, the prediction accuracy of this hypothetical predictor, shown as "PAs w/ Loop", was 0.8 and 1.4% higher than that of PAs for gcc and go respectively and 0.1-0.6% higher for the other benchmarks. This shows that PAs is not necessarily the best choice for all per-address predictable branches.

Benchmark	PAs	PAs w/ Loop	IF PAs	IF PAs w/ Loop
compress	93.46	93.49	94.41	94.42
gcc	92.08	92.91	91.86	93.20
go	82.16	83.53	84.81	85.84
ijpeg	94.87	95.50	95.86	96.28
m88ksim	98.58	99.14	99.09	99.35
perl	96.83	96.96	97.79	97.87
vortex	98.86	99.14	99.03	99.23
xlisp	95.46	95.54	96.70	96.73

Table 3. Prediction accuracy of PAs w/ and w/o loop enhancement

We also performed the same experiment with a similar hypothetical predictor using an interference-free PAs. As shown in table 2, the prediction accuracy of this hypothetical predictor was 0.3 and 1.0% higher than that of interference-free gshare for gcc and go respectively (representing 4% of the mispredictions for gcc and 7% of the mispredictions for go) and on average 0.2% higher for the other benchmarks. Although interference limits the ability of PAs to predict loops, an interference free PAs will still not be able to predict the exits of loops longer than its history length.

5. Branch correlation vs. per-address predictability

We have examined the behavior of branches both in terms of branch correlation based and per-address based predictability. However, the highest performance is achieved by hybrid predictors, typically consisting of a global component and a per-address component. To better understand the cooperation between the component predictors that takes place in a hybrid, we would like to know the individual benefits of both global and per-address predictors in that context. In this section, we examine this both using the classes of global and per-address predictability defined earlier, and using PAs and gshare.

5.1. Global / per-address distribution

We have examined branch correlation (section 3) and classes of per-address based predictability (section 4). In this section, we present a distribution showing how many branches fall into each of these categories, weighted by execution frequency. Once again, we do not classify branches which are predicted at least as accurately with an ideal static predictor.

Figure 7 shows the distribution of branches best predicted by gshare and PAs. The proportion of branches that were predicted at least as accurately with an ideal static predictor, 55% on average, is also shown for reference. In this case, 83% of these branches were more than 99% biased. On average, gshare was best for 29% of the branches, and PAs was best for 16% of the branches.

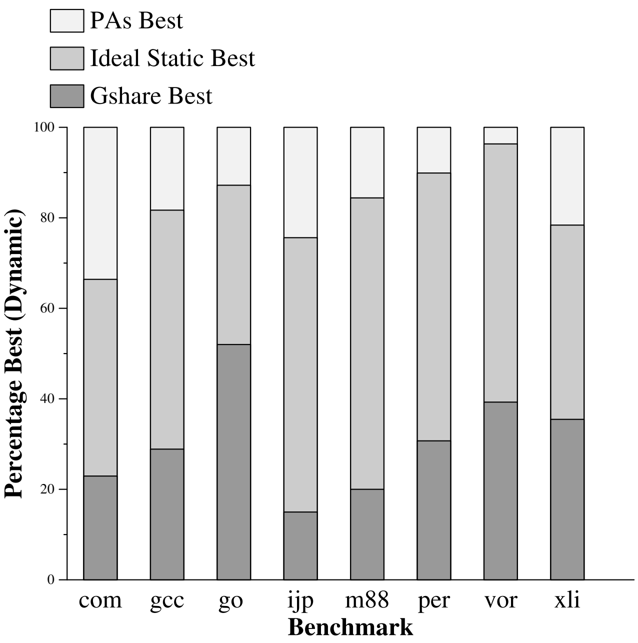

Figure 7. Distribution of branches best predicted using gshare, PAs, and an ideal static predictor, weighted by execution frequency

Figure 8 shows the distribution of branches best predicted using the branch correlation and per-address based predictors given in this paper. In this case, the branch correlation fraction included branches best predicted using interference-free gshare or using a 3-branch selective history (see section 3.4). The per-address fraction included branches best predicted using any of the per-address based predictors de-

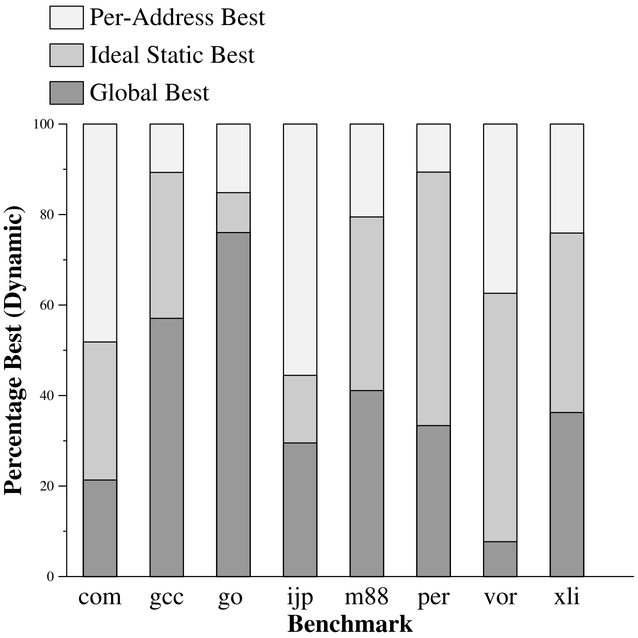

Figure 8. Distribution of branches best predicted using global correlation, per-address based predictors, and an ideal static predictor, weighted by execution frequency

scribed in section 4.1. The proportion of branches that were predicted at least as accurately with an ideal static predictor, 40% on average, is also shown for reference. 92% of these were more than 99% biased. 38% of the branches were best predicted using branch correlation. 22% of the branches were best predicted using per-address based predictors.

Two conclusions that can be drawn from figure 7 and 8 is that (1) there is a significant set of branches for which we can do better than PAs and gshare, and (2) there are 40% of the branches for which we are still unable to do better than an ideal static predictor. However, of these, on average 92% are more than 99% biased. For the other 8%, more research is needed to discover better dynamic predictors.

5.2. Individual importance of gshare and PAs

We showed in the previous section that there is a large number of branches for which gshare is the better predictor, and a large number of branches for which PAs is better. To fully understand the importance of having both a global and a per-address predictor, we need to know how much better each predictor is for the set of branches where it is best.

Therefore, we studied the difference in prediction accuracy between gshare and PAs for all branches. For branches where the difference is small, either of the predictors could be used without a significant loss in prediction accuracy.

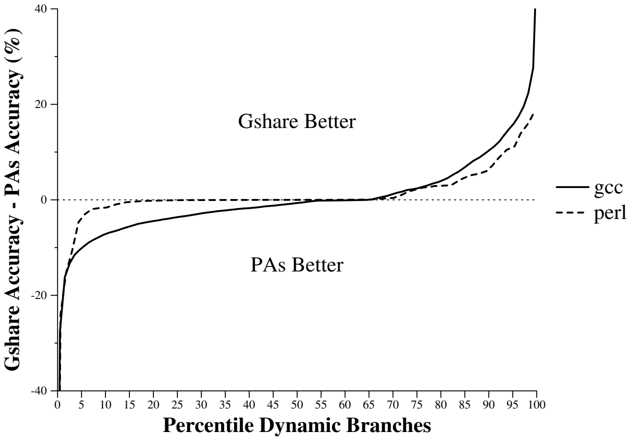

Figure 9. Difference between gshare and PAs accuracy

However, for branches where the difference is large, it is important to use the better predictor for that branch.

Figure 9 shows the distribution of the difference between the accuracy of gshare and PAs for gcc and perl. The curve for go is very similar to that of gcc, while perl is representative of the other benchmarks. For gcc, we see from the left end of the figure that for 10% of the dynamic branches, PAs is more than 7.0 percentage points better than gshare. Similarly, we see from the right end of the figure that for 10% of the dynamic branches, gshare is more than 10.4 percentage points better than PAs.

The area between the curve and the horizontal line in the "PAs Better" region indicates the amount of prediction accuracy that would be lost if gshare were the only predictor. The area between the curve and the horizontal line in the "Gshare Better" region indicates the amount of prediction accuracy that would be lost if PAs were the only predictor.

We can see from figure 9 that there is a large number of branches for which PAs is much better than gshare, and there is a large number of branches for which gshare is much better than PAs. This explains why hybrid predictors are capable of achieving much higher performance than individual predictors.

6. Conclusions

In this paper, we have explained and quantified reasons why global two-level branch predictors work. We showed that only information about a very few previous branches is needed for a correlation based predictor to be accurate, and that these branches can generally be found close to the branch that is being predicted. We further showed that the gshare two-level predictor is not fully exploiting this correlation. Capturing correlation with two or three branches was needed to in most cases surpass the prediction accuracy of gshare. However, if gshare could take advantage of only the single strongest correlation for each branch, it would achieve a 3.7% higher prediction accuracy for gcc, and 1.2% higher accuracy on average.

We also examined per-address predictability. We identified three classes of per-address predictability and showed how often each of these occurred. We showed that there is a large set of branches with loop-type behavior in the benchmarks. A PAs predictor using a separate loop predictor for these branches would achieve a 0.8% higher prediction accuracy than PAs for gcc, and 0.5% better on average.

Furthermore, we showed the frequency of branches for which each type of predictability was best. We presented this both using the two-level predictors gshare and PAs, and using the classes of predictability defined in this paper. We showed that 55% of the branches were at least equally well predicted with an ideal static predictor as with PAs or gshare. For the 17% of these branches that are not heavily biased, more accurate dynamic predictors are still needed. However, when using our classes, we identified that several of these had higher predictability, and could potentially be improved beyond the accuracy of an ideal static predictor. Of the branches that were still at least equally well predicted with a static predictor, 92% were more than 99% biased.

Finally, we showed that there is a large set of branches for which PAs is significantly better than gshare, and a large set for which gshare is significantly better than PAs, confirming the importance of using both global and per-address predictors in hybrid branch predictors.

7. Acknowledgments

This paper is one result of our ongoing research in high performance computer implementation at the University of Michigan. The support of our industrial partners, in particular, Intel and HAL computer systems is greatly appreciated.

References

[1] 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 Annual ACM/IEEE International Symposium on Microarchitecture*, pp. 22–31, 1994.

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

[3] R. Nair, "Dynamic path-based branch correlation," in *Proceedings of the 28th Annual ACM/IEEE International Symposium on Microarchitecture*, pp. 15–23, 1995.

[4] 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*, pp. 76–84, 1992.

[5] S. Sechrest, C.-C. Lee, and T. Mudge, "The role of adaptivity in two-level adaptive branch prediction," in *Proceedings of the 28th Annual ACM/IEEE International Symposium on Microarchitecture*, 1995.

[6] 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*, 1996.

[7] A. Seznec, "Trading conflict and capacity aliasing in conditional branch predictors," in *Proceedings of the 24th Annual International Symposium on Computer Architecture*, 1997.

[8] J. E. Smith, "A study of branch prediction strategies," in *Proceedings of the 8th Annual International Symposium on Computer Architecture*, pp. 135–148, 1981.

[9] 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 1995 ACM/IEEE Conference on Parallel Architectures and Compilation Techniques*, 1995.

[10] T.-Y. Yeh and Y. N. Patt, "Two-level adaptive branch prediction," in *Proceedings of the 24th Annual ACM/IEEE International Symposium on Microarchitecture*, pp. 51–61, 1991.

[11] 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*, pp. 124–134, 1992.

[12] 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*, pp. 276–286, 1995.