Combining Branch Predictors

1 Abstract

1.1 instruction-level parallelism： key factor of computer performance

1.2 conditional branch： critical limit to instruction-level parallelism

1.3 predictiing condition branch：uses the history of previous branches

1.4 a method：combining the advantages of serveral predictors

1.4.1 uses a history mechanism

1.4.2 hasing branch history with branch address

1.4.3 outperforms previous approach

1.4.4 accuracy reaches 98.1%

1.4.5 a factor of two smaller than other schemes for the same accuracy

2 Introduction

2.1 branch instructions are important

2.1.1 Reason1：instruction level parallelism is used for a higher levels of performance

2.1.2 Reason2：supersclar and superpipelining techniques are popular

2.1.3 Reason3：branch instructions are important in determining overall machine performance

2.2 compiler assisted techniques are les appropriate

2.2.1 Reason1：the number of delay slots increases

2.2.2 Reason2：the use of delay slot problematic

multiple implementations of an architeccture with different superscalar or superpiplining

2.2.3 Efftect：increase the importance of hardware methods of reducing brach cost

2.3 branch performance problem

2.3.1 prediction of the branch direction：the theme of this paper

2.3.2 minimal delay to get the branch target

Branch Target Buffer：a special instruction cache to store target instruction

referred to Lee and Smith for more information

2.4 hardware branch prediction stratedies

2.4.1 bimodal branch prediction

2.4.2 local branch prediciton

2.4.3 global branch prediction

2.5 a new technique：combines the advantage of different branch predictors

2.6 a method ：increases the utility of branch history by hasing it together with the branch address

2.7 organization of this paper

2.7.1 Section 2：previous work in branch prediction

2.7.2 Section 3：bimodal predictors

2.7.3 Section 4：local predictors

2.7.4 Section 5：global predictors

2.7.5 Section 6：predictors indexed by both global hitory and branch address information

2.7.6 Section7：hashing global history and branch addres information before indexing the predictors

2.7.7 Section8：the technique for combining multiple predictors

2.7.8 Section9：concluding remarks

2.7.9 Section10：suggestions for future work

3 Related Work

3.1 J.E.Smith：serveral hardware schemes ect. bimoal schenem

3.2 Lee and A.J.Smith：evaluated several branch prediction scheme

3.3 McFarling and Hennesy：compared hardware and software approaches

3.4 Hwu，Conte, Chang：performed a similar study for a wider range of pipeline length

3.5 Fisher and Freudenberger：studied the stabiliy of profile information

3.6 Yet and Patt：described both the local and global branch predictor

3.7 Pan, So, and Rahmeh：described hwow both global history and branch address information can be used in one predictor

3.8 Ball ad Larus：described several techniques for guessing the most common branches directions at compile time using static information

4 Bimodal Branch Prediton

4.1 background：the behavior of typical branch is order taken or not taken

4.2 simplest approach：a table of counters indexed by the low order address bits in PC

- Q1: use the branch address or the branch instruction address ?
-

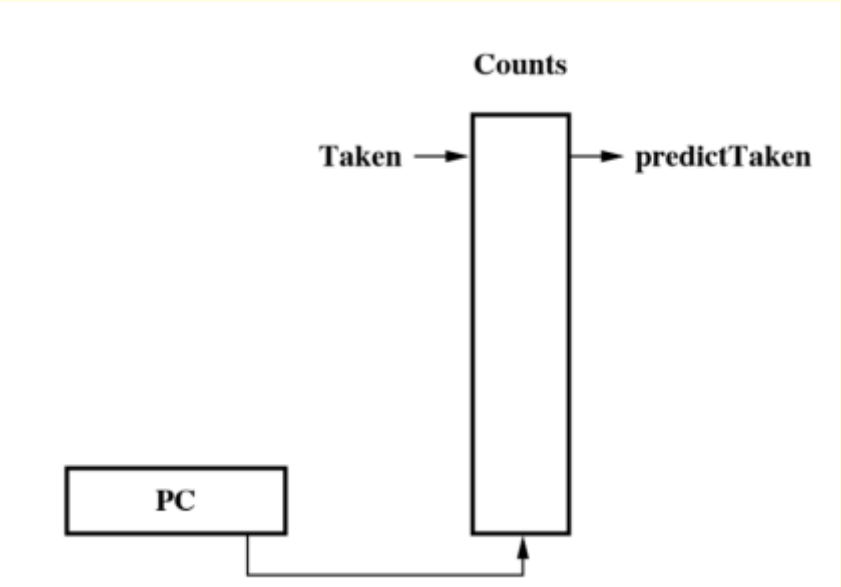


Figure 1: Bimodal Predictor Structure

4.2.1 if taken， counter is incremented and the counter is saturating

4.2.2 if not taken, counter is decremented

4.2.3 the most significant bit determines the prediction

- Q2: if the counter use to many bits, then it will take a long time to predict the later branch ？

4.2.4 2-bits long counter can tolerate a branch going an unusual direction one time

4.2.5 small table results in degraded prediction accuracy

Reason： multiple branches may share the same counter

4.2.6 alternative implementation：store a tag with each counter and use a set-associative lookup to match

disadvantage： if the size of tags is accounted for, a simple array of counters is better

if tags were already needed to support branch target buffer, it will not be the case

4.3 quantitative evaluation： use SPEC'89 benchmarks

-

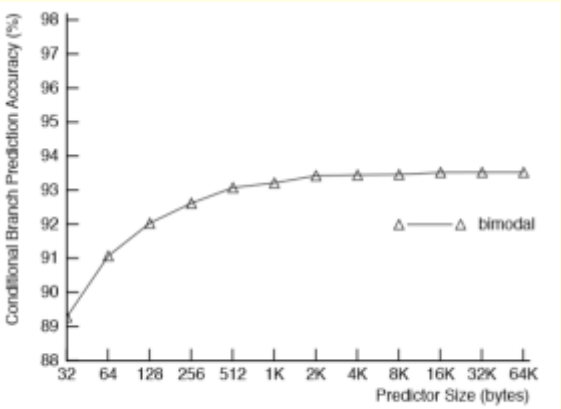


Figure 3: Bimodal Predictor Performance

4.3.1 only the first 10 million instructions was simulated

4.3.2 Execution traces were obtained n a DECstation 5000 using the pixie tracing facility

4.3.3 all counters are initially set as if all previous branches were taken

4.3.4 The accuracy increases with predictor size increases

4.3.5 the accuracy saturates at 93.5% once each branch maps to a unique counter

4.3.6 a set-associative predictor would saturate at the same accuracy

5 Local Branch Prediction

5.1 backgound：one way to improve on bimodal prediction：recognize that many branches execute repetitive patterns

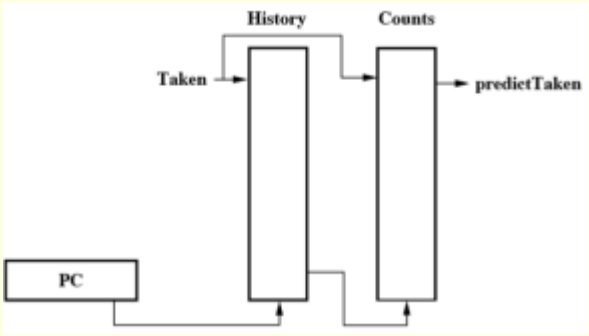
5.1.1 most commom example ： for(i = 1; i <= 4; i++)

5.1.2 execute pattern：(1110)^n, 1--taken, 0-- not taken, n-- the loop times

- Q3：th loop time is the hole loop or the i < n

5.2 A branch prediction uses this pattern --- per-address scheme：named by Yeh and Patt

5.2.1 uses two tables



1. Hisory table： records the history of recent branches

- Q4: will the entry value of history table change dynamiclly if there is only a loop?

an array indexed by the low-order bits of branch address

a set-associative branch history table

each ectry records the recent n branches whose branch address is mapped , n--the bits of entry bits

2. Counts table：like the bilodal branch prediction

- Q5：what the point of pattern like (1110) if we use the significant bit of counter to determine branch?

indexed by the branch history stored in the first table

5.2.2 local predictor can suffer from two kinds of contention

1. the branch history may reflect a mix of histories which map to the same entry

solution: increase bits of entry

2. one array for counter results in conflict between patterns

5.3 quantitative evaluation：use SPEC'89 benchmarks

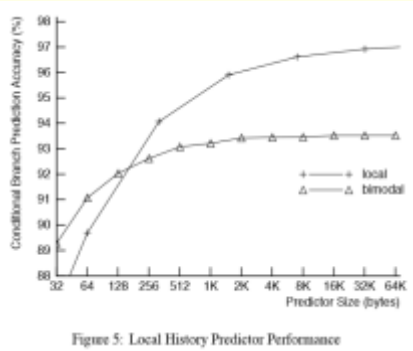
- 

Figure 5: Local History Predictor Performance

5.3.1 if the size of predictor is too samll, there will be of no value

5.3.2 above 128byte the local predictor is better than bimodal

5.3.3 the accuracy approaches 97.1%

5.3.4 less than half as many misspredictions as the bimodal scheme

6 Global Branch Prediction

6.1 background：local branch predictor only considers the current branch

6.2 A scheme：take consideration of other recent branches to make a prediction

6.2.1 A single shift register GR：records the direction taken by the most recent n conditional branches

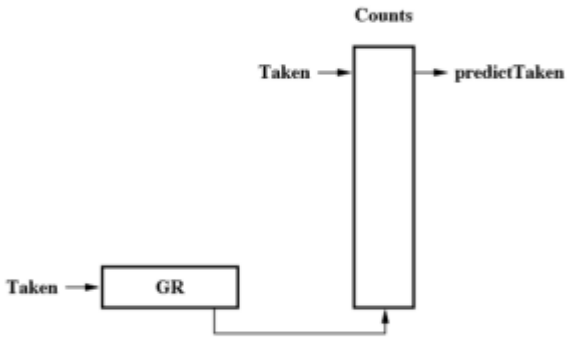


Figure 6: Global History Predictor Structure

6.2.2 sut for two types of patterns

1st：the direction take by current branch may depend on other recent branches. ect if (x > 1)... if (x < 1)...

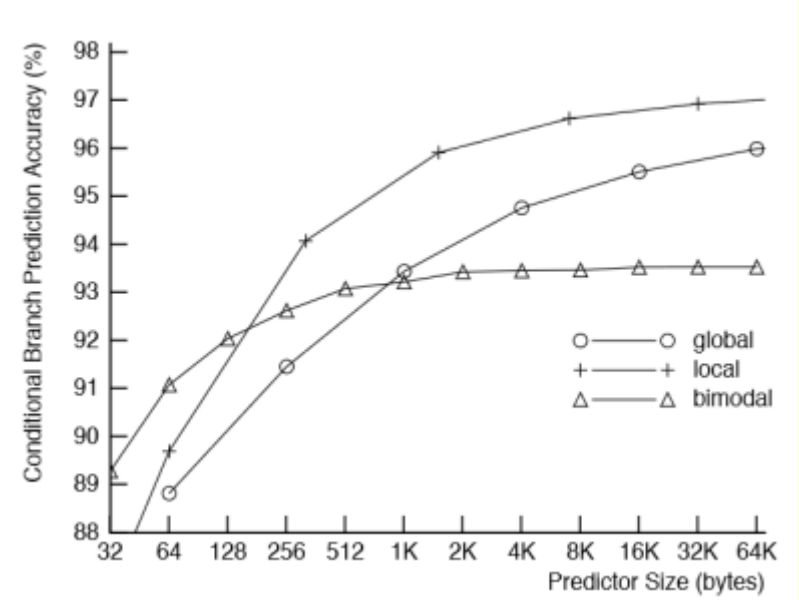
2nd：by duplicating the behavior of local branch prediction ect for( ){ for( ){ } }

test	value	GR	result
j<3	j=1	1101	taken
j<3	j=2	1011	taken
j<3	j=3	0111	not taken
i<100		1110	usually taken

- Q6: how does the GR change? use the significant bit of GR to judge result?

6.3 quantitative evaluation：use SPEC'89 benchmarks

- for small predictors, the bimodal is better than others，as the size increase, the global is better
-
-



6.3.1 the information content in each additional address bit decline to zero for increasingly large counter tables.

- Q7: what is the "information coontent"?

Reason1： the branch address bits used efficiently distinguish different branches

Reason2：As the numbers if counters double, half of branch will share the same counter

Reason3：as more counters added, each frequent branch will map to unique counter

6.3.2 The information content of GR continue to grow for larger sizes

over 90% of the time each branch goes the same direction which allows a global predictors to identify different branches

GR can capture more information than just idnetifying which branch is current

7 Global Predictor with Index Selection

7.1 background：global history information is less efficient at identifying the current branch than using the branch address

7.2 gselect：uses both the branch address and the global history

7.2.1

7.2.2

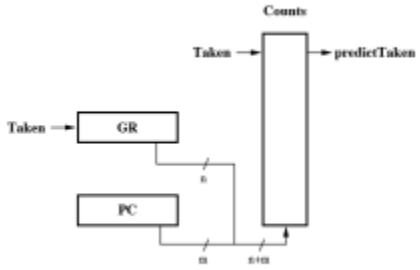


Figure 8: Global History Predictor with Index Selection

7.3 there is a tradeoff betweenusing more history bits or more address bits

7.4 quantitative evaluation

- a gselect predictor have less delay and be easier to pipeline than a local predictor
  -
  -

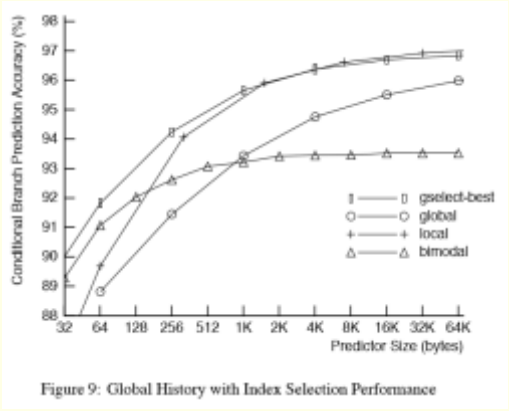


Figure 9: Global History with Index Selection Performance

7.4.1 better than simple global prediction or bimodal

7.4.2 The storage space required for global history is negligible

7.4.3 gselect requires only a single array aceess whereas local prediction requires two array accesses

8 Global History with Index Sharing

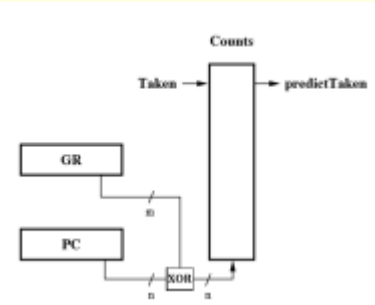
8.1 backgroud

8.1.1 global history information weakly identifies the current branch

8.1.2 there is a lot of redundancy in the counter index used by gselect

8.1.3 if there are enough address bits to identify the branch, the frequent global history cobinations can be sparse

8.2 gshare: hasing the branch address and the global history

- 

8.2.1 hasing it by the exclusive OR of the branch address with the global history

8.2.2 gshare is able to separate more cases than gselect

8.3 quantitative evaluation

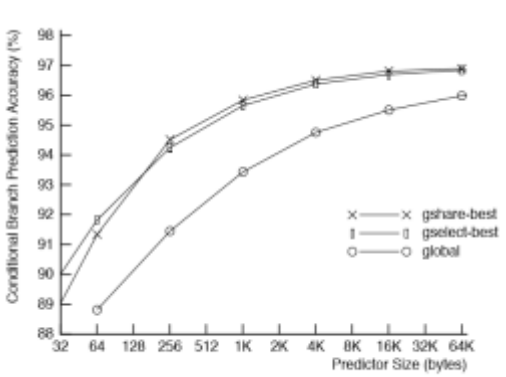
- 

Figure 11: Global History with Index Sharing Performance

8.3.1 for small predictors, gshare underperform gselect

Reason：there is already too much contention for counters between different branches and adding global information just makes if worse

8.3.2 for predictor sizes 256 bytes and over, gshare outperforms gselect

9 Combining Branch predictors

9.1 background

9.1.1 the different branch prediction schemes have different advantages

9.1.2 the combination can have a better prediction accuracy

9.2 A method: containd two predictors P1 and P2

9.2.1 P1 and P2 can be any kind of branch predictor

9.2.2 contains an additional counter array to select the best predictors to use

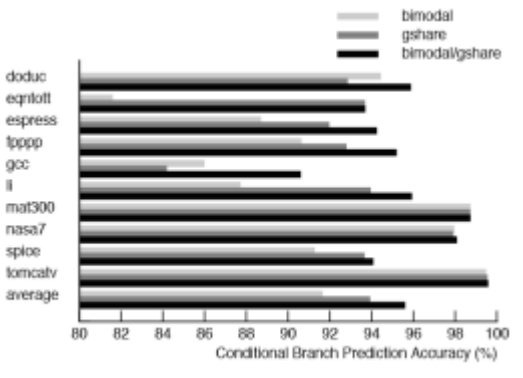
9.2.3 use 2-bit up/down saturating counters

P1c	P2c	P1c-P2c	
0	0	0	(no change)
0	1	-1	(decrement counter)
1	0	1	(increment counter)
1	1	0	(no change)

- P1c P2c means if the predictor are correct respectively

9.3 quantitative evaluation：use SPEC'89 benchmarks

-



9.3.1 the combination of bimodal/gshare is useful

9.3.2 all the benchmarks were run to completion

9.3.3 the predictors array has 1K counters

9.3.4 the combination of local/gshare predictor outperforms bimodal/gshare from 2kb size

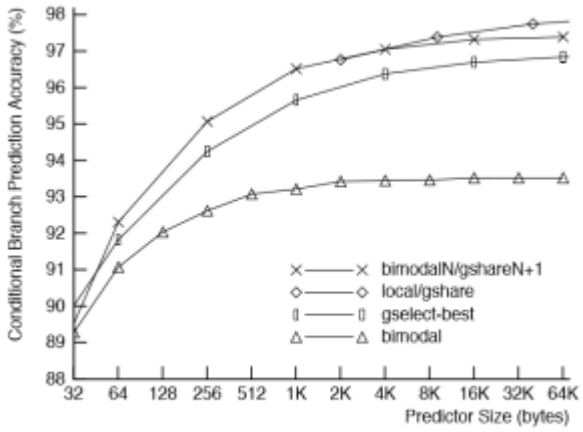


Figure 16: Combined Predictor Performance by Size

9.3.5 the accuracy approaches 98.1%

10 Conclusions

10.1 this paper presents two new methods for improving branch prediction performance

10.1.1 1st. using the bit-wise exclusive OR of the global branch history and the branch address to access predictor counters

10.1.2 2nd. combining advantages of multiple branch predictors

combined predictors using local and global branch information reach a accuracy of 98.1%

11 Suggestion for Future Work

11.1 1st. parameters like sizes, associativities and the pipeline costs can be explored

11.2 2nd. the information such as whether the branch target is forward or backward might be useful

11.3 3rt. the sparse branch history might be compressed to reduce the number of counters needed

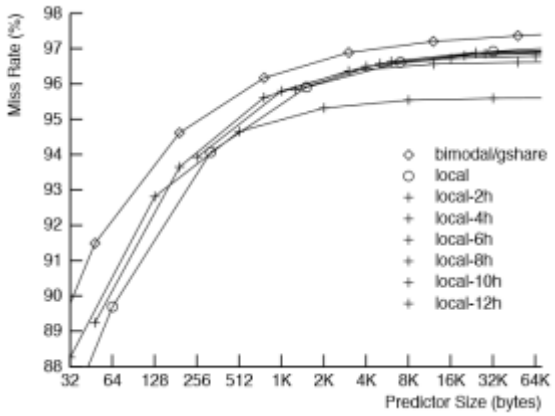
11.4 4th. a compiler with profile support might be able to reduce or eliminate the need for branch predictor

12 A Appendix

12.1 for section 4, the local prediction. discuss two variations and show that the combined predictor has better performance than these alternatives

12.1.1 1st. indexing the counter array with both the branch address and the local history

the number of history bits used to index the counter array is held constant



- Q8: why I cannot understance these two figures?

12.1.2 2nd. change the number of history entries

using the same number of history entries of history table entires as counters is usually a good choice