CSE 240A Final Project

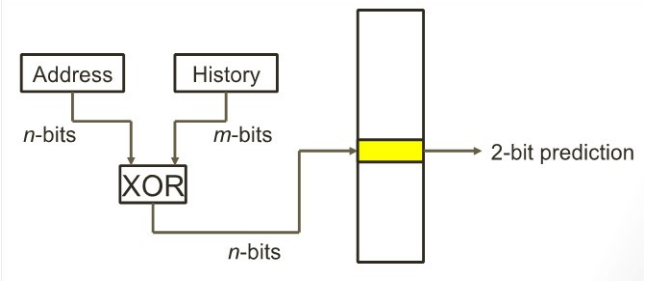
Chuling Ko, Liting Yang

1. INTRODUCTION

Branch Prediction is important to improve performance of instruction pipeline. Nowadays, people use high level of instruction parallelism like superscalar and superpipelining which makes branch becomes more critical determining performance. Predicting correctly allows instructions complete faster without branch hazards. Branch predictor does this by letting more instructions complete within one clock cycle. However, misprediction will costs much more misprediction penalty than origin because we waste time on wrong path and mispredicted instruction need to be flushed and overwritten by proper instructions. Therefore, many people are now engaged in looking for a perfect branch predictor. Some people work hard on offline training data aggregated over multiple application executions in order to deal with H2P and rare branches [1]. In this paper, we discuss three different dynamic branch predictor and compare accuracy numbers obtained for each predictor. We introduce gshare branch predictor , tournament branch predictor, and customized branch predictor. They predict branches base on local history and global history. And we will illustrate their implementations, advantage, disadvantages, and their evolution.

2. IMPLEMENTATION

2.1 Gshare



The Gshare Branch predictor combines a Global History Register (GHR) and the least significant bits of the branch's PC address to index into 2 bit branch history register table. The PC and the GHR are XORed in order to create the index to the branch history register table.

At the beginning, all state are initialized to be Weak Not Taken and all history are initialized to be 0, then repeat the following steps:

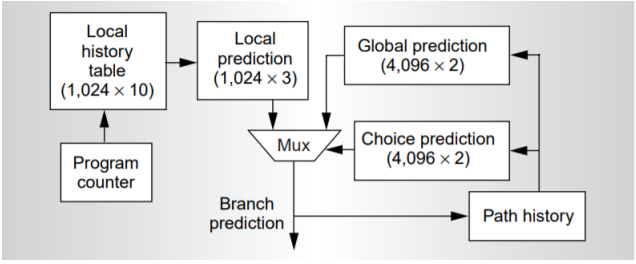
- For every branch instruction, index is that pc address do XOR operation with global History
- Each index has a two bit correlated predictor which have four state (SN = 0,WN = 1,WT = 2,ST = 3). And we make a prediction by the predictor corresponding to index. SN and WN predict NOT TAKEN, otherwise TAKEN.
- If the correct result is TAKEN state will minus one; otherwise, state will plus one
- Then, we update global history by shift one left and do OR operation with outcome.

If the branch prediction mechanism is a one-level mechanism, least significant bits of the PC address are used to index the table. Two irrelevant branches with the same least significant bits might index to the same entries. Two-level adaptive predictors [2] were proposed by Yeh and Patt and later they identified several variations of this mechanism depending on how branch history and pattern history is kept. However, aliasing is a problem for Two-level adaptive predictors.

McFarling [3] states that simply using the address of the branch instruction might be more helpful than using global history information. Using both global history and PC address by hashing them together might be a better way. He thought the value of XOR of global branch history and branch address will contains more information than only global branch history or only branch instruction address. This causes a more uniform distribution and usage of the PHT entries reducing interference. This mechanism seems to need less hardware, by keeping both branch and pattern history globally.

Nevertheless, aliasing is still an unsolved problem for Gshare. Gshare use multiple table entries to track the behavior of any particular branch. This multiplication of entries makes it much more likely that two irrelevant branches will map to the same table entry which in turn makes it much more likely that prediction accuracy will suffer for those branches. For Gshare, aliasing in the predictor table can eliminate any advantage gained through inter branch correlation [4].

2.2 tournament



In this project, we implement Alpha 21264 Tournament pre-

dictor [5]. Tournament branch predictor combines global predictor and local predictor. And use a selector to select prediction outcome from one of them. No matter which predictor we select, update both predictors and the selector based on the correctness of prediction. For global predictor indexed by globalHistory register and local history indexed by least significant bits of PC address.

At the beginning, all state are initialized to be Weak Not Taken, selector state is initialized to be weak global predictor, and all history are initialized to be 0, then repeat the following steps:

- For every instruction, global predictor index is path history ; pc address index into local branch history table and then index into local prediction table.
- Each predictor for each index has a two bit correlated predictor which have four state (SN = 0,WN = 1,WT = 2,ST = 3). And we make a prediction by the predictor corresponding to index. SN and WN predict NOT TAKEN, otherwise TAKEN.
- Then we got the outcomes from two predictor, we use selector to choose our final result. There are four state (SG = 3, WG = 2, WL = 1, SL = 0). SG and WG predict the result from global predictor, otherwise predict from local predictor.
- For both predictor, if the correct result is TAKEN state will minus one; otherwise, state will plus one. For selector, if Global predictor predict correct but Local predictor predicts wrong, then selector state plus one; if Global predictor predict wrong but Local predictor predicts correct, then selector state minus one; id both are wrong or correct, then selector state remains.
- Then, we update global history by shift one left and do OR operation with outcome. And we update local branch table by shift one left and do OR operation with outcome.

Once you have multiple predictors, it is beneficial to arrange that each predictor will have different aliasing patterns, so that it is more likely that at least one predictor will have no aliasing [6]. A Hybrid Branch Predictor as proposed by McFarling is built by taking two branch predictors that work independently from each other and a selector that chooses which branch prediction to take.

Although some branch predictors perform well on particularcases such as floating-point programs containing lots of loops, they are not accurate for other cases that different branch predictors have better prediction. Combining a branch predictor with a large history table with another predictor with a small history table or none is effective to avoid the negative effect of context switches [7]. Thus, it is possible to have multiple properties of a single predictor for a better result by using combined predictor.

Tournament predictors are simply combinations of two branch prediction mechanisms.This approach increase accuracy by picking up the predictor (global or local) which is working better. Also it reduces warm-up time because we use faster-warmup predictor until the slower-warmup predictor warms up. However, tournament predictor needs to use a meta-predictor or a selector and causes longer access latency.

2.3 customized

We implement another hybrid predictor [3] based on Gshare and tournament what we implemented in previous part. Our predictor implement local predictor indexed by least significant bits of PC address and implement global predictor same as Gshare predictor in section 1.1 .

At the beginning, all state are initialized to be Weak Not Taken, selector state is initialized to be weak global predictor, and all history are initialized to be 0, then repeat the following steps:

- For every instruction, global predictor index is that pc address do XOR operation with global History ; pc address index into local branch history table and then index into local prediction table.
- Each predictor for each index has a two bit correlated predictor which have four state (SN = 0,WN = 1,WT = 2,ST = 3). And we make a prediction by the predictor corresponding to index. SN and WN predict NOT TAKEN, otherwise TAKEN.
- Then we got the outcomes from two predictor, we use selector to choose our final result. There are four state (SG = 3, WG = 2, WL = 1, SL = 0). SG and WG predict the result from global predictor, otherwise predict from local predictor.
- For both predictor, if the correct result is TAKEN state will minus one; otherwise, state will plus one. For selector, if Global predictor predict correct but Local predictor predicts wrong, then selector state plus one; if Global predictor predict wrong but Local predictor predicts correct, then selector state minus one; id both are wrong or correct, then selector state remains.
- Then, we update global history by shift one left and do OR operation with outcome. And we update local branch table by shift one left and do OR operation with outcome.

3. OBSERVATION

We compare our custom predictor against a Gshare predictor with 13 bits of global history (–global:13), which is the largest possible Gshare that fits the 16kb budget. We also compare it against a Tournament predictor of about 14kb. This predictor uses 9 bits of global history, 10 bits of local history and 10 PC bits (–tournamet:9:10:10). In our customized predictor, we use 13 bits for global history, 12 bits for local history and 11 bits for pc index. The storage usage is $2*2^{13}$ bits Gshare PHT, $2*2^{13}*2$ bits for selector table, $2*2^{1}1$ bits for local BHT, and $11*2^{11}$ bits for local PHT. Total is 59392 bits which is less than 64K + 256 bits.

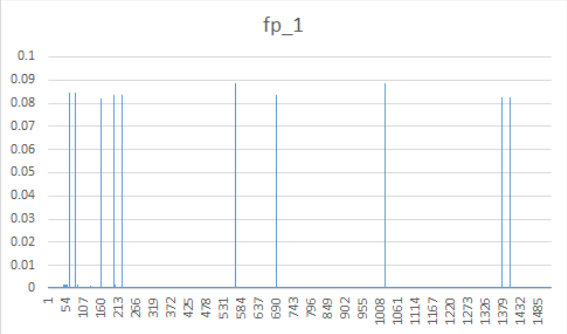
3.1 Gshare

From the observation of Gshare, we can tell that if the distribution of the indices is concentrated, then Gshare can have a good performance, like fp_1 and int_2 ; on the other

Table 1: Accuracy Rate(%)

Traces	Gshare	Tournament	customized
fp1	99.175	99.009	99.181
fp2	98.322	96.754	99.614
int1	86.161	87.378	90.116
int2	99.58	99.574	99.711
mm1	93.304	97.419	98.845
mm2	89.862	91.517	93.034

Figure 1: The index distribution on trace fp_1



hand, if the indices used in Gshare are spread out over a wide area, then the state of the predictor should not be stable enough, thus the performance won't be so good, like int_1 and mm_2.

Figure 1 to Figure 6 are the graph of each traces. The y-axis is the percentage of each index, and the x-axis are the indices.

3.2 Tournament

We can tell that when there is a wide gap between the proportion of selecting local and selecting global, the performance of the tournament predictor is better among all the traces. Like trace fp_1, which has 83% selecting local, and trace int_2, which has 95% selecting global. Note that these two traces are also the ones with concentrated index distribution in the analysis of gshare prediction.

From Table 1 and Table 2, we can also observe that when tournament predictor select the local prediction more than the global prediction, it can beat gshare predictor, like trace int_1, mm_1, mm_2. That is why we decide to use tournament with local predictor and gshare in our custom predictor.

Figure 2: The index distribution on trace fp_2

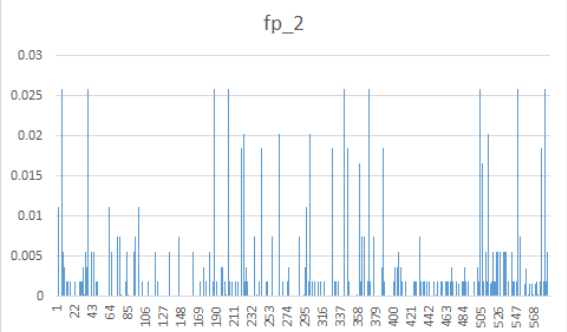


Figure 3: The index distribution on trace int_1

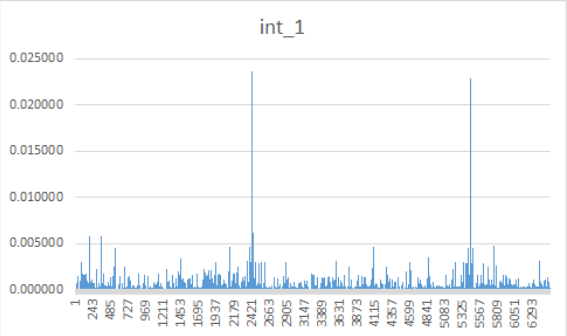


Figure 4: The index distribution on trace int_2

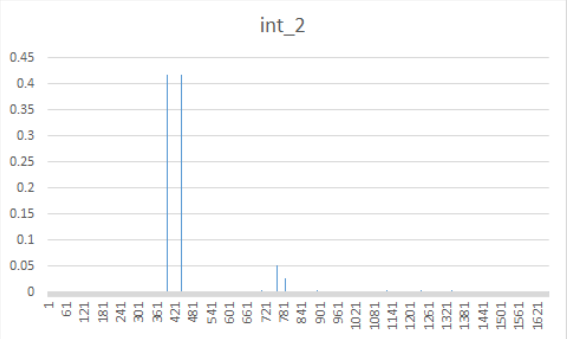


Figure 5: The index distribution on trace mm_1

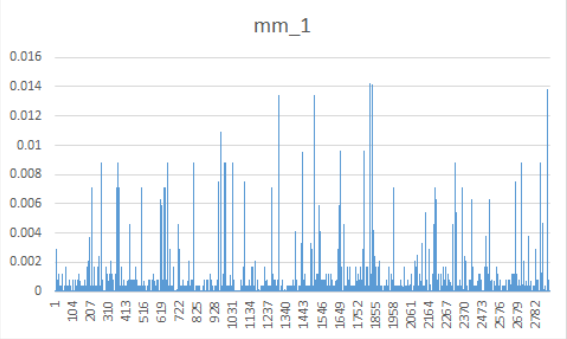


Figure 6: The index distribution on trace mm_2

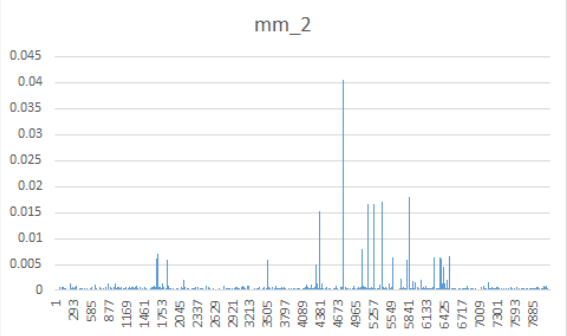


Table 2: Selected rate for tournament(%)

Predictor	Local	Global
fp1	83.1116	16.8884
fp2	39.7181	60.2819
int1	62.4415	37.5585
int2	5.0827	94.9173
mm1	67.5743	32.4257
mm2	71.2684	28.7316

3.3 Custom

When we select gshare(global) predictor, our accuracy rate for gshare predictor is usually better for single gshare predictor in section 1.1. Because we use gshare predictor only when we expect Gshare has better performance than local predictor. Tournament performs better when there are many proportion of selecting local. Our customized predictor has same local predictor as tournament in section 1.2 which means our custom has similar ability to handle those branches. In those cases Tournament predictor performs better than Gshare, our local predictor also have a better accuracy rate than Tournament.For trace int1, we have lower accuracy than other traces because in this case we select half global predictor and half local predictor and we are likely to change our selector which might cause a misprediction.

Table 3: Accuracy Rate(%)

Predictor	Local	Global	select local rate
fp1	99.51	99.10	18.51
fp2	99.99	99.43	32.63
int1	89.97	90.26	50.16
int2	99.76	99.62	65.21
mm1	97.82	99.52	39.68
mm2	94.17	90.73	66.89

4. RESULTS AND CONCLUSION

From Table 1, the final result is that our custom predictor is the best overall. As for the performance comparison between Gshare and Tournament, in trace fp_2, Gshare predictor beats Tournament predictor; in trace int_1, mm_1, mm_2, Tournament predictor beats Gshare predictor; and in trace fp_1 and int_2, the performances are similar between Tournament predictor and Gshare predictor.

From Gshare, the observation is that when a trace has a concentrated distribution of the index XORed by address and history, the performance of Gshare on that trace is good. That is because the prediction state of each index is stable enough.

From Tournament, we have two observations. First, if the proportion difference between selecting global and selecting local is wide, the performance of Tournament on that trace is good. Second, if local is selected more often than global, the performance of Tournament on that trace is better than Gshare.

For our customized predictor, if the distribution of the indices is concentrated, then we can select Gshare as a global predictor which preforms well in these cases. if there are many branches predictions need to rely on local information such as those cases tournament performs well, we can select local predictor. Therefore, our customized predictor always has a better accuracy than Gshare predictor and tournament predictor.

In conclusion, we introduce and implement Gshare predictor and Tournament predictor, and our custom predictor which is hybrid based on Gshare and Tournament. And then we observe the performance of the three predictors and the six traces, finding out that the simpler trace can have the better performance, and our custom predictor is the best, and the tournament predictor beats Gshare predictor when the local predictor is selected more.

5. CONTRIBUTION

In this project, Chuling is responsible for implementing Gshare predictor and Tournament predictor, and run the experiments on the observation section for Gshare predictor and Tournament predictor. And for the report, Gshare and Tournament part in Observation and Result and Conclusion is written by Chuling.

Liting is responsible for designing and implementing Custom predictor, and run the experiments on the observation section for Custom predictor. Also,for the report, Introduction and Implementation and Customized part in Observation and Result and Conclusion are written by Liting.

6. REFERENCES

[1] A.-K. Lin and S. J. Tarsa, "Branch prediction is not a solved problem: Measurements, opportunities, and future directions," Jun. 2019.

[2] T.-Y. Yeh and Y. N. Patt, "Two-level adaptive training branch prediction," in *24th ACM/IEEE International Symposium on Microarchitecture*, Nov. 1991.

[3] S. McFarling, "Combining branch predictors," in *Technical Report TN-36m*, Jun. 1993.

[4] C. L. S. Serest and T. Mudge, "Correlation and aliasing in dynamic branch predictors.," in *Proceedings of the 23rd Annual International Symposium on Computer Atxhitecture*, May. 1996.

[5] R. E. Kessler, "The alpha 21264 microprocessor," 1999.

[6] S. S. A. N. Eden, J. Ringenberg and T. Mudge, "Hybrid myths in branch prediction," in *Conf. on Information Systems Analysis and Synthesis (ISAS 2001)*, Jul. 2001.

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