Implementation and Analysis for G-Share, Tournament and Perceptron Branch Predictor for CSE240A SP19

Xiaofan Yu
PID#: A53276743
Email: x1yu@ucsd.edu

1. INTRODUCTION

Branch prediction has raised great interests after the appearance of pipeline, which is proposed to enhance instruction throughput by processing multiple instructions simultaneously at different stages. However, the cost we have to pay for this inter-instruction parallelism is the extra hardware and software to prevent various hazards, e.g. data, structural and control hazard. Among all types of hazards, control hazard is directly caused by branches. On the program level, branch enables more complex control flow. There is no way for the consecutive instruction to make the right fetch excepting stalling until the judging stage finishes. To deal with this issue, branch prediction is proposed to boost throughput performance while assure correct execution.

With branch prediction, the pipeline will make a guess and start fetching that predicted instruction anyway. If the prediction is correct, the machine can continue executing as if no branch exists. However, if the guess is wrong, our pipeline will flush every mistaken instruction and their temporary results, then migrate to the right segment. In this case, the pipeline seems to stall for the same time as with no branch prediction. The benefit of branch prediction can be explicitly shown in the Cycles Per Instruction (CPI) equation:

$$CPI = 1 + p_{mis\_predict} \cdot c_{penalty} \tag{1}$$

Here we assume the base CPI is 1. $p_{mis\_predict}$ represents the misprediction rate while $c_{penalty}$ denotes for extra stalling cycles. It can be observed that, a sophisticated branch predictor with very little misprediction rate can achieve close-to-best throughput performance. The disadvantage of integrating branch prediction into pipeline is the hardware and memory overhead caused by judging circuit, Branch History Table (BHT), etc.

Since Branch Prediction first came into the field in 90s, a large amount of research has been proposed until 2010s. All the designs of branch predictor focus on two aspects: *(a)* how to capture the local (PC, i.e. Program Counter) and history (BHT) information efficiently to make better prediction. *(b)* How to reduce hardware implementation cost while retain the accuracy. Or in other words, given a memory constraint, which is the best branch predictor.

Traditional G-Select and G-Share predictors glue or perform XOR operation on PC and global history bits in order to index BHT [1]. Later on, the hybrid Tournament Predictor popularized by the Alpha 21264 [2] makes a better trade-off between local and global history, reaching better accuracy than gselect and gshare. Recently, novel predictors such as Perceptron [3] and TAGE [4] leverage more complex data processing techniques. Perceptron branch predictor borrows the idea from machine learning while TAGE adds data values to better capture the correlation between branch outcome and history content. However, there is not much exciting work recently.

In this project, I implement G-Share, Tournament and Perceptron predictors under a $(16k + 256)$ bits limitation. The implementation details will be discussed and compared in Section 2. After that, Section 3 analyzes the evaluation results of all 3 implementations on the given 6 traces from time- and memory-varying dimensions. Finally, the report is concluded in Section 4.

2. IMPLEMENTATION

In this Section, I will briefly explain the high-level ideas of 3 implemented predictors: G-Share, Tournament and Perceptron. Their advantages, disadvantages and technical evolution pattern are gathered in Table 2.

2.1 G-Share

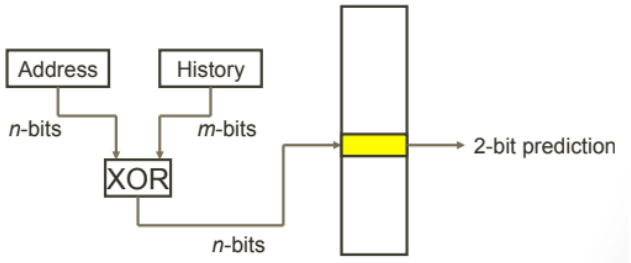


Figure 1: Diagram of G-Share predictor from slides of Prof. Jishen Zhao.

As shown is Figure 1, G-Share uses the XOR result of PC and history register to locate a 2-bit saturating counter in BHT. Predictions will be made based on the 2-bit state, which will be updated later according to the branch outcome. In all, G-Share uses the XOR operation to magically record

both local and global information. The size of the BHT is computed as follows:

$$Size = 2 \cdot 2^{N_{bits}} \tag{2}$$

where N_{bits} is the number of used bits in PC and history register. Thus, under the $16k+256$ bits memory bound, the maximum allowed bits is 13.

2.2 Tournament

Due to the inflexible XOR operation, G-Share does not generate well in some cases that highly rely on local address or global history compared with the other. Triggered by this issue, Tournament predictor splits BHT into 3 routes (Figure 2). Local and global prediction trajectory are completely separated, taking either PC or history bits. A chooser is employed to record which one works better with certain path history. Note the local prediction route here is a two-level addressing: a local prediction table writes down the history of each PC address, and the final local prediction is indexed by that local history record. With Tournament predictor, both fine-grained local and global predictions are enabled, while choice prediction makes the final prediction the better one from two candidate predictions.

Obviously, the cost of Tournament predictor is the large required memory space. Using Equation (3), the best possible tournament predictor under our memory constraint is to have 9, 10, 10 bits for local history, global history and PC respectively, with a total size of 14k bits.

$$Size = N_{local\_history} \cdot 2^{N_{PC}} + 2 \cdot 2^{N_{local\_history}} + 2 \cdot 2 \cdot 2^{N_{global\_history}} \tag{3}$$

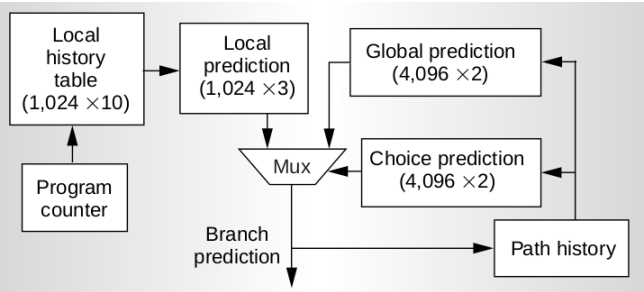


Figure 2: Block diagram of Tournament predictor [2].

2.3 Perceptron

Perception is the simplest linear regression heuristic. The key idea is to train a weight for each history bit, and the table of weights F is indexed by PC [4]. All weights are intialized to zero. As shown in Figure 3, the prediction value is calculated as follows:

$$y = F_0 + \sum_i F_i \cdot BHR_i \tag{4}$$

The ultimate prediction will be TAKEN if y is larger than a pre-set threshold, and vice versa. The training process only takes place if mis-predicted. For example, if the true outcome is TAKEN, the weight associated with history bit 1 will be increased by 1. The total space required by Perceptron

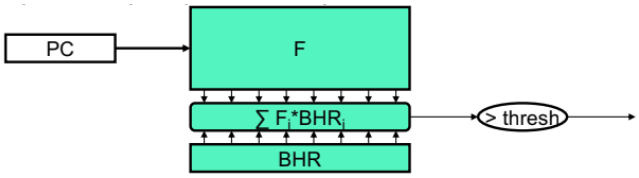


Figure 3: Block diagram of Perceptron predictor from Prof. Jishen Zhao.

Table 1: Parameters Setting of Hybrid Perceptron Predictor

	PC	Global History	F	G-Share	Chooser
Notation	N_{PC}	$N_{history}$	N_{F_bits}	N_{gs_bits}	N_{chsr_bits}
Number of Bits	6	21	8	11	9

predictor mainly depends on the size of F:

$$size = 2^{N_{PC}} \cdot N_{history} \cdot N_{F\_bits} \tag{5}$$

Notice that, unlike previous design, the size is linearly related to the number of history bits. Therefore, Perceptron predictor can make use of long history bits. But they also possess several shortcomings as written in Table 2.

In my final implementation, I use hybrid Perceptron predictor which uses a chooser table to select from Perceptron and G-Share. The total memory usage is calculated by Equation (6). Important parameters shown in Table 1 are set to meet the $(16k+256)$ bits memory requirement.

$$size = 2^{N_{PC}} \cdot N_{history} \cdot N_{F\_bits} + 2 \cdot 2^{N_{gs\_bits}} + 2 \cdot 2^{N_{chsr\_bits}} \tag{6}$$

3. OBSERVATION

3.1 Prediction Accuracy

Misprediction rate of all three implemented predictors on the 6 traces are summarized in Table 3. In this experiment under the $(16k+256)$ bits memory limitation, I use the G-Share predictor with 13-bit PC and global history, the Tournament predictor with 9-bit global history, 10-bit local history and 10-bit PC, the hybrid Perceptron predictor with settings in Table 1.

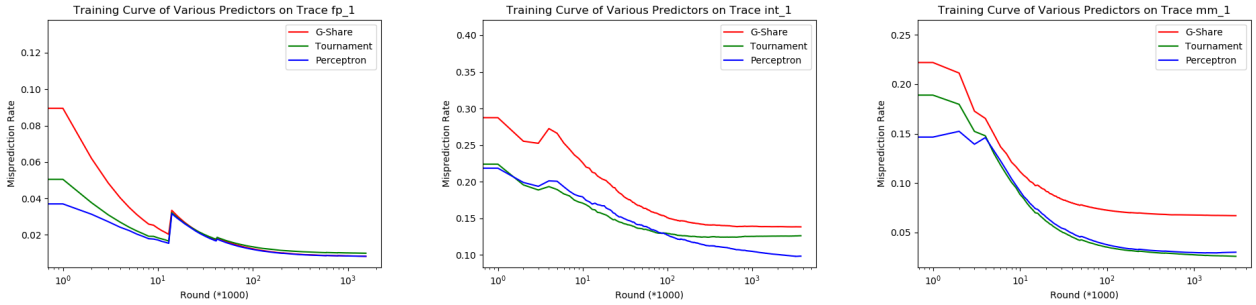
It can be observed that, the implemented hybrid Perceptron predictor beats the rest two in 3 out of 6 traces, while performing close to the best predictor in the other 3 traces. However, it is hard to examine the property of each predictor simply from the misprediction numbers. In the following lines, comparisons are made from two critical design considerations of current branch predictors: learning capability and memory limitation.

3.2 Learning Capability

The first important factor is the information capture ability over time. In other words, how does the prediction accuracy or misprediction rate evolve as time increases? Figure 4a, 4b and 4c show the visualization of misprediction rate of

Table 2: Comparison of G-Share, Tournament and Perceptron Predictors

Predictor	Advantage	Disadvantage	Evolution
G-Share	Efficiently combine local and global history together	Inflexible in making trade-off between local and global information	Improved from G-Select, better utilize space by XORing rather than gluing PC and history bits
Tournament	Use choice prediction to balance local and global prediction	Larger memory overhead	Solve the inflexibility in G-Share
Perceptron	Able to use long history bits	Not as fine-grained as Tournament predictor	Novel predictor that employs machine learning algorithms to save space and better capture history



(a) Trace fp_1. (b) Trace int_1. (c) Trace mm_1.

Figure 4: Learning curve of G-Share, Tournament and hybrid Perceptron predictors on different traces.

Table 3: Misprediction Rate of G-Share, Tournament and Hybrid Perceptron Predictors

Trace	G-Share	Tournament	Perceptron
fp_1	**0.825254**	0.991016	0.896886
fp_2	1.677959	3.245971	**0.986231**
int_1	13.838811	12.622250	**10.678960**
int_2	0.420098	0.425530	**0.376879**
mm_1	6.695889	**2.580626**	3.404680
mm_2	10.138043	**8.483219**	9.892091

G-Share, Tournament and hybrid Perceptron predictors over three traces respectively. The setting of each predictor is the same as in previous section. Comparing over the three figures, following observations can be spotted:

(a) The same predictor presents varied learning characteristics on different traces. This is attributed to the difference in workload pattern.

(b) Three predictors exhibit similar trends on the same trace. G-Share is the one starting from highest misprediction rate and usually ending up with the less good prediction performance. Although G-Share shows the best result on fp_1, it performs much worse than the rest two predictor on the rest traces. This proves the above-mentioned statement that G-Share is simple but not flexible.

(c) The prediction accuracy of hybrid Perceptron predictor is close to, if not better than, the Tournament one. This shows that both Tournament and hybrid Perceptron predictors make a good balance between global and local information.

(d) Another detail to notice is that, hybrid Perceptron predictor usualy has better results at the beginning of training, compared with the other two options. In fact, this phenomenon surprises me as Perceptron Machine Learning algorithms usually require longer time to converge. The outstanding initial performance here is probably because using coefficients is able to capture pattern faster than XORing or looking up tables.

3.3 Memory Limitation

Apart from learning capability, another key factor to care about is the memory limitation. Branch predictor is an additional block in pipeline. Large predictor will impede the speed of prediction, which finally degrades the overall throughput. Besides, different system have non-uniform memory budget. Therefore, which implementation approach to select given a certain memory limitation becomes a key issue in branch predictor design.

To investigate into this problem, I record the changing of misprediction rate when bearing different memory constraints. I alter the number of bits of each predictor to meet the 1k, 2k, 4k, 8k, 16k, 32k, and 64k bits memory requirements. The experiment result is shown in Figure 5. The misprediction rate here refers to the ratio of total mispredicted branches

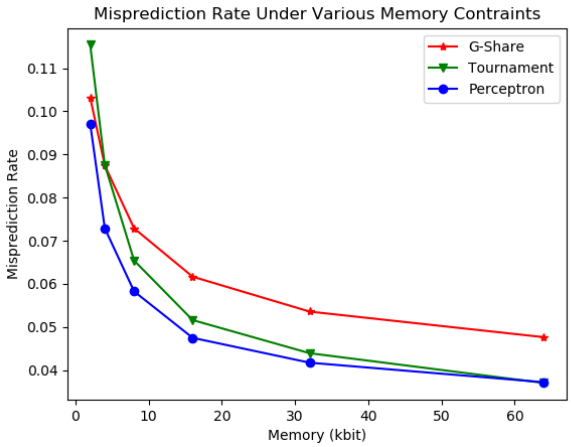


Figure 5: Misprediction rate of G-Share, Tournament and Perceptron predictors under various memory upperbound.

over all decisions across 6 traces.

As we expected, an overall decreasing trend can be seen as the memory space increases. However, different predictors show different characteristics when considering from the memory perspective. Perceptron is the overall best predictor. It is able to fully-utilize the history bits thus having advantage when considering memory constraint. G-Share is the worst among the three candidates, as XORing is not flexible in capturing local and global history information. Tournament predictor shows impressive result with higher memory budget. The reason for this is that although Tournament predictor makes fine-grained balancing between local and global data, it has high memory overhead, requiring two-level local history table and an extra chooser.

4. RESULTS AND CONCLUSION

In this project, I implement three branch predictors based on existing literature: G-Share, Tournament and hybrid Perceptron predictors. Evaluations are conducted on accuracy, learning capability and memory limitation. Although the misprediction rate of hybrid Perceptron predictor is only able to beat the rest two candidates in 3 out of 6 given traces, I show that hybrid Perceptron predictor presents faster learning and better prediction accuracy under memory constraints.

5. REFERENCES

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

[2] R. E. Kessler, "The alpha 21264 microprocessor," *IEEE micro*, vol. 19, no. 2, pp. 24–36, 1999.

[3] D. A. Jiménez and C. Lin, "Dynamic branch prediction with perceptrons," in *Proceedings HPCA Seventh International Symposium on High-Performance Computer Architecture*, pp. 197–206, IEEE, 2001.

[4] A. Seznec, "A 256 kbits l-tage branch predictor," *Journal of Instruction-Level Parallelism (JILP) Special Issue: The Second Championship Branch Prediction Competition (CBP-2)*, vol. 9, pp. 1–6, 2007.