TAGE Branch Predictor on Sniper Simulator

CS5222 – Project 1

SHEN JIAMIN (A0209166A)

1 INTRODUCTION

In the paper "A case for (partially) TAgged GEometric history length branch prediction" [7] by André Seznec (2006), the author presented his design of a new branch predictor named TAGE predictor. The paper was produced in the background that (1) exploiting several different history lengths is acknowledged, and (2) hybrid predictors combining the prediction result from multiple meta predictors have been developed as well as prediction combination functions such as majority vote, predictor fusion and partial tagging and the adder tree seems to outstanding among them.

The TAGE predictor users a geometric series as the list of history length, which is the main characteristic of O-GEHL[2], while it derives the main structure from PPM-like predictor[1]. The significant difference of TAGE from O-GEHL is the use of partial tags instead of adder tree, which the author believes provides better accuracy and storage efficiency and enables the adaptation of the design for conditional branch prediction to the indirection target branch prediction. Compared with the PPM-like predictor, TAGE proposed a new predictor update algorithm that minimizes the perturbation induced by a single occurrence of a branch. Based on the structure of TAGE, the author also proposed an indirect branch target predictor ITTAGE and a unified predictor COTTAGE, which supports both conditional branch and indirect branch target prediction.

The author evaluated the accuracy of the TAGE predictor with the parameters listed in Table 2 in the paper. He compared the MPKI (mispredictions per kilo instructions) of TAGE, O-GEHL[2] and PPM-like[1] predictors with different prediction tables sizes. The result shows that the TAGE predictor outperforms O-GEHL and PPM-like in all tested conditions, indicating the correctness of the author's proposal. He also evaluated the accuracy regarding the variation of history length, which shows that with the maximum history length growing, the accuracy is better at first but no improvement when the length is too long. The evaluation plot for the impact of the tag width is not as clear as the ones ahead. The variation of interest is the tag width, but the plot's x-axis varies the prediction tables size, which makes it difficult to compare between different tag widths. The result shows that increasing the tag width over a threshold provides little return for accuracy improvement. Besides the explicit evaluations discussed above, the experiment results show that the mispredictions decrease with larger prediction tables but diminishing returns for accuracy.

The work is auspicious in that many variations of TAGE predictor are proposed after it and lead nearly all the Championship Branch Prediction afterwards, for example, L-TAGE[3] in 2007, ISL-TAGE[4] in 2011 and TAGE-SC-L[5, 6] in 2014 and 2016.

However, these following works also indicate the limitations of the TAGE predictor. In [3], the author found that using the alternate prediction, instead of the prediction using the longest history, for newly allocated entries is more efficient in some applications. He also identified that TAGE fails to predict when the control flow path inside the loop body is irregular and augmented TAGE with loop predictor. In [4], it is identified that TAGE sometimes fails to predict branches that are not strongly biased but that are only statistically biased and thus the Statistical Corrector Predictor is

Author's address: Shen Jiamin (A0209166A), shen_jiamin@u.nus.edu.

introduced. He also identified that the number of access to the predictor can be reduced and had an Immediate Update Mimicker as an add-on.

I choose this branch predictor because it's the base of many TAGE-variant branch predictors. Having this implemented in Sniper will make the following implementations for the variants easier. The most difficult part in implementing TAGE is the design of the hash functions used to generate the indices and tags, which is completely not described in the paper.

2 TAGE BRANCH PREDICTOR

2.1 Structure

A N-component TAGE branch predictor consists of one base predictor and $N - 1$ tagged predictor components.

The base predictor is a simple bimodal branch predictor. A bimodal branch predictor is an array of saturating signed counters. The counters are indexed only by the program counter.

A tagged predictor component is an array of (ctr, tag, u) 3-tuples. ctr is a prediction counter like the ones in the bimodal predictor. It is a saturating signed counter indicating the branch tendency. u is a usefulness counter. It is a saturating unsigned counter indicating how useful the entry is. A tagged predictor component uses two difference hash functions to index the table and generate tags. Both of the two hash functions take the program counter and the prediction history as its inputs. Importantly, the tagged components keeps prediction histories of different lengths, where the lengths are a geometric series.

2.2 Prediction Algorithm

The prediction is produced by combining the prediction result from all the components.

The base predictor, which doesn't make prediction based on prediction history, produce the prediction result by directly checking if the indexed counter is non-negative. If $\text{ctr} \geq 0$, the predictor returns a `taken` prediction.

The tagged components, which make prediction based on prediction histories of different lengths, produce the prediction result by first checking if the indexed entry has a matched tag with regards to the current program counter and prediction history, and returns no prediction if the tag doesn't matched. With a matched tag, it returns a `taken` prediction if $\text{ctr} \geq 0$.

Among the predictions from the components, TAGE takes the one based on the longest prediction result as the prediction result. That is, the last component that provides a prediction. The component that provides the final prediction is the **provider component**, and the component that would have provided the prediction, if the provider component does not, is the **alternative component**.

It should be noted that TAGE can always give a prediction because it will fallback to bimodal predictor if all the tagged components fail to match the tag, while a bimodal predictor can always provide a prediction.

2.3 Update Algorithm

Assume that the prediction provided by provider component is hitpred and the one provided by alternative component is altpred. The actual branch result is actual.

2.3.1 Update Base Predictor. The bimodal predictor is always updated by incrementing its ctr if the branch result is taken or decrementing the ctr if the branch result is not taken.

2.3.2 Update Usefulness Counter. The usefulness counter is updated on two cases.

(1) If hitpred ≠ altpred and the provider component is a tagged component, the indexed entry from the provider component should update its usefulness counter. The counter is incremented if the prediction is correct (hitpred = actual), which means it's more useful, and decremented otherwise.
(2) Periodically, the usefulness counter should be reset to imitate the behavior of Least Recently Used policy. Concretely, every call to update function will increment a `tick` counter. When it hits a threshold, it clears the least significant bit (or most significant bit, in turn) of all the usefulness counters in all components.

2.3.3 Update Prediction Counter. If the provider component is a tagged component, the indexed entry in the provider component will update its ctr. It increments ctr if the branch result is taken and decrementing the ctr otherwise.

2.3.4 Allocate New Entry. The predictor attempts to allocate a new entry if the prediction is incorrect (hitpred ≠ actual) and the provider component didn't keep the longest prediction history. When allocating a new entry, the predictor looks for an available component, where the entry indexed has a usefulness count of 0, from the components taking longer history than the provider component.

If there are multiple components available, the predictor chooses one randomly. The components taking longer history has a higher probability of being chosen. If there is no available components, all the usefulness counter in these components are decremented.

3 DEBUGGING

As the build system for sniper is confusing for me, I chose to use "Caveman Debugging". This is done by printing the critical states during execution and examine their behavior. One successful example is when I printed out the distribution of hit component, I found that the prediction is heavily based on the bimodal predictor, while only less than 1/1000 go to the tagged components. This is very strange and inspired me to examine the allocating behavior and the calculation of indices and tags. It turns out that I incorrectly implemented the global prediction history buffer, which caused a significant loss of history information.

4 EVALUATION

4.1 Parameter Search

The first part of the evaluation aims to search for a configuration that makes TAGE works well. The two selected number of tagged components are 5 and 8, and the bit lengths of tag are 9 and 11 respectively. The bit length of ctr is set to 4, With $n \in [15, 24)$, we set the log length of bimodal prediction table as $n - 4$ while the log length of tagged prediction table as $n - 6$. and the minimun length of history is set to 4. The space for maximum length of history is $\{32, 64, 128, 256, 512, 1024, 2048\}\}$. The results are plot in Figure 1 and Figure 2, whico show that the TAGE predictor works best with the configuration listed in Table 1.

The plot also shows that if the maximum history length is too long, which leads to inclusion of unrelated prediction history during prediction, the prediction performance may goes worse.

4.2 Comparision

We also have a comparison between the implemented TAGE branch predictor and other predictors existing in the Sniper. We configured two different of TAGE predictors, named TAGE-5 and TAGE-8 as shown in Table 1. Then we run

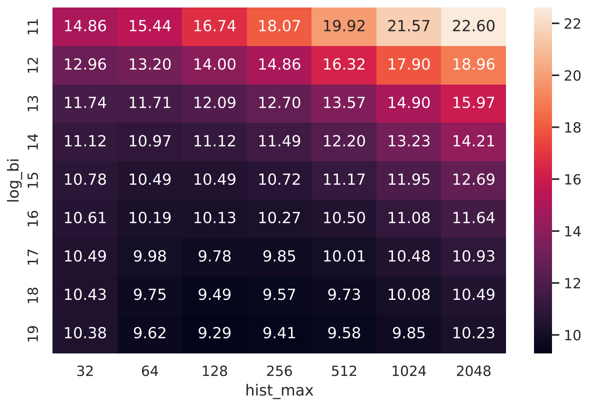

Fig. 1. Result of parameter search with 5 tagged components (The annotated number is MPKI)

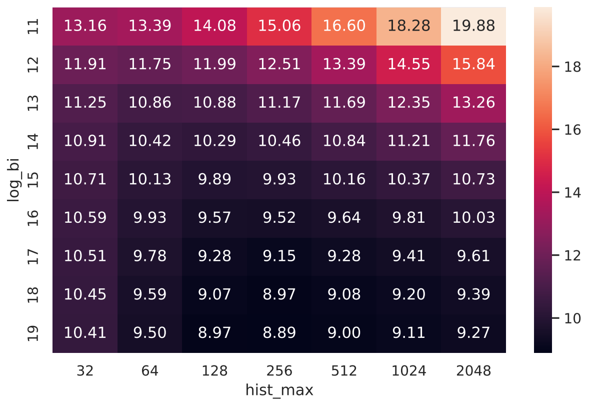

Fig. 2. Result of parameter search with 8 tagged components (The annotated number is MPKI)

Table 1. Configuration of TAGE predictors in comparison

Parameter	TAGE-5	TAGE-8
# Tagged Components	5	8
# of entries in Bimodal Predictor (log)	19	19
ctr + tag width (bits)	3 + 9	3 + 11
# of entries in Tagged Components (log)	17	17
History length (bits)	4 − 128	4 − 256
Misprediction Penalty	8	8

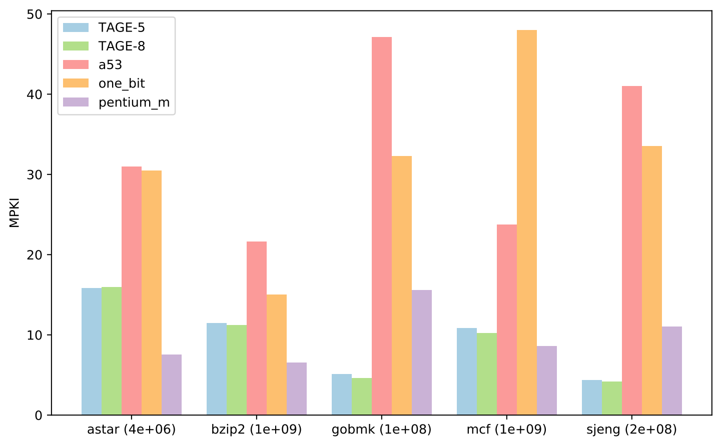

Fig. 3. Comparision of different predictors on the pinballs

1B instructions based on "nehalem-lite" configuration of Sniper and simulate the five provided pinballs. The simulation result is shown as Figure 3.

Due to some unknown causes, the pinballs exit earlier without completing the 1B-instruction simulation. Among the five pinballs, only bzip2 and mcf can complete the simulation for 1×10^9 instructions. The number of instructions simluated for the pinballs is attached in the parentheses.

By comparing the MPKI (misses per 1K instructions), we find that TAGE-5 and TAGE-8 have similar performance. They outstand in gobmk and sjeng simulation and worse only than Pentium M in other simulations.

ACKNOWLEDGMENTS

I'm willing to get my work merged into the Sniper codebase.

REFERENCES

[1] Pierre Michaud. 2005. A PPM-like, Tag-based Predictor. *The Journal of Instruction-Level Parallelism* 7 (2005), 10 pages. https://jilp.org/vol7/

[2] André Seznec. 2005. Analysis of the O-GEometric History Length Branch Predictor. In *32nd International Symposium on Computer Architecture (ISCA'05)*. IEEE Computer Society, Madison, WI, USA, 394–405. https://doi.org/10.1109/ISCA.2005.13

[3] André Seznec. 2007. The L-TAGE Branch Predictor. *The Journal of Instruction-Level Parallelism* 9 (2007), 13 pages. https://jilp.org/vol9/index.html

[4] André Seznec. 2011. A 64 Kbytes ISL-TAGE branch predictor. In *JWAC-2: Championship Branch Prediction*. JILP, San Jose, United States, 4 pages. https://hal.inria.fr/hal-00639040

[5] André Seznec. 2014. TAGE-SC-L Branch Predictors. In *JILP - Championship Branch Prediction*. Minneapolis, United States, 8 pages. https://hal.inria.fr/hal-01086920

[6] André Seznec. 2016. TAGE-SC-L Branch Predictors Again. In *5th JILP Workshop on Computer Architecture Competitions (JWAC-5): Championship Branch Prediction (CBP-5)*. Seoul, South Korea, 4 pages. https://hal.inria.fr/hal-01354253

[7] André Seznec and Pierre Michaud. 2006. A case for (partially) TAgged GEometric history length branch prediction. *The Journal of Instruction-Level Parallelism* 8 (2006), 23 pages. https://jilp.org/vol8/