A Deep Dive Into TAGE-SC-L

Alberto Ros
Computer Engineering Department
University of Murcia
Spain
aros@ditec.um.es

Abstract

TAGE-SC-L is the state-of-the-art branch predictor. In this work, we analyze the TAGE-SC-L predictor in depth and propose a number of optimizations on top of it. The key contributions of this work are related to (i) the series used to choose the history lengths, (ii) the selection of predictions from the different predictor components, and (iii) the allocation, update, and replacement of TAGE entries.

Our evaluation using the 105 training traces provided by the championship shows that our proposal obtains an average mispredictions per kilo-instruction (MPKI) of 3.4120 and an average of 145.4 cycles on the wrong path per kilo-instruction (CycWpPKI).

1 Introduction

TAGE-SC-L [3], winner of the 5^{th} Championship Branch Prediction, is the state-of-the-art branch predictor. The 64KB TAGE-SC-L implementation provided on the 6^{th} Championship Branch Prediction achieves for the 105 training traces an average mispredictions per kilo-instruction (MPKI) of 3.7506 and an average of 152.5 cycles on the wrong path per kilo-instruction (CycWpPKI).

The TAGE-SC-L predictor consists of the following prediction components:

- The TAGE predictor, a large structure that includes a bimodal predictor, and several tagged tables using geometrically increasing history lengths. The tagged tables are grouped into two sets of tables: one for low history lengths using fewer bits for the tag field, and another for high history lengths using more bits for the tag.
- The Loop predictor, a small, high-accurate, low-coverage predictor used to detect the end of the loops.
- The Statistical Corrector (SC), a perceptron-like predictor [2] used to cover some of the cases in which TAGE provides a non-negligible fraction of mispredictions.
- A set of counters to decide which of the above described components will provide the final prediction.

In this work, we analyze TAGE-SC-L in depth and propose some optimizations on top of it. Hence, this work focuses on improving MPKI. However, as a consequence of the reduction in MPKI, the CycWpPKI metric is also expected to get lower.

We started by creating a version of TAGE-SC-L using ≈192 KB, as a new baseline predictor used for this work. This new version doubles the size of the TAGE tables and multiplies the bimodal component by 8. It also increases the number of TAGE tables from 10+20 (for low and high history lengths) to 14+30, and uses an extra bit in the low history tables (9 bits in total) to store the tags. The loop prediction size is also doubled. Similarly, all SC components are doubled as well, except for the path history component, which is multiplied by 4. We also made some changes that do not affect the size of the predictor, such as setting a higher number of history lengths to (NHIST=42), having fewer histories assigned to low history length banks (BORN=9), allocating one more entry to TAGE on a misprediction (NNN=2), and increasing the period to reset entries (BORNTICK=512*3). This new baseline achieves 3.4405 MPKI (146.1 CycWpPKI).[1]

Once we have defined a new baseline, we add the proposed modifications on top of it. In particular, this work makes the following contributions.

- We propose a different series for choosing the history lengths. Instead of geometrically increasing history lengths, we advocate starting with a quadratic sequence and ending with a generalized geometric sequence with linearly increasing multipliers.
- We increase the accuracy of the branch prediction with a careful selection of the components that provide the prediction, taking into consideration the confidence of the loop predictor and the use of the alternate prediction.
- We improve the usefulness of the entries in the TAGE tables, by filtering the allocation of entries and enhancing the eviction policy.

2 Enhancements to TAGE-SC-L

This section describes the behavior of key components within the TAGE-SC-L predictor, along with our proposed modifications and optimizations to these components.

2.1 History is not how it was told

TAGE-SC-L uses geometrically increasing history lengths.[2] The TAGE predictor is trained first using the lowest history lengths and, on mispredictions, it allocates entries using larger histories. The rationale behind geometrically increasing lengths is to get precision using many low history lengths, but also to be able to cover patterns that are predictable using few large histories.

However, the current implementation of TAGE increases performance by not using the same "weight" for all history lengths, meaning that some lengths of the geometric series are even discarded after creating it, while others are granted two ways in the TAGE tables instead of one. Figure 1 shows in the blue line the geometric series (y axis) of 21 elements (x axis). Those history lengths that are discarded are represented without a circle, while those that are granted two ways are represented with a double-sized circle. Note that the y-axis is logarithmic, and hence the geometric series

'CBP 2025', June 21, 2025, Tokyo, Japan
2025. ACM ISBN How.to.remove.this?(Not.ACM)
https://doi.org/How.to.remove.this?(Not.ACM)

[1]Other minor arrangements in the code to simplify the update of the predictor were also done at this step, without significantly impacting MPKI.
[2]The "GE" from TAGE comes from GEometrically.

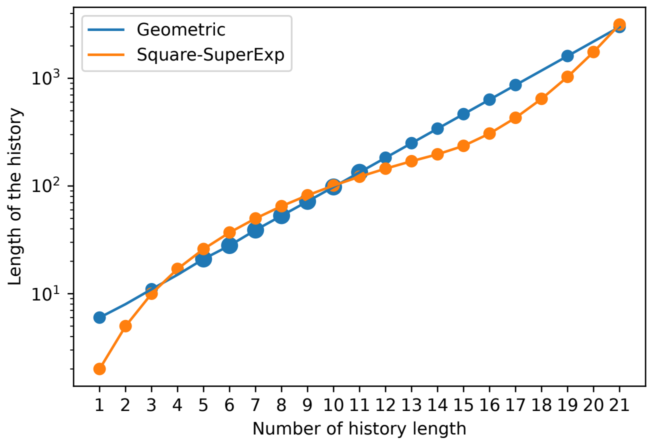

Figure 1: History lengths for the TAGE component

is seen as a straight line. We can observe that the extremes in the blue line have less weight, whereas the middle lengths have more weight.

This decision motivates the choice of a different series, where the low and high lengths increase at a higher speed than in the geometric series, but the middle lengths increase slower. Our proposal is to start with a quadratic sequence, which at the beginning increases faster than the geometric series and at the end slower. Then at some point, it switches to a generalized geometric sequence with linearly increasing multipliers (the growth factor increases at each step), hence growing faster than a geometric series function. Figure 1 shows the resulting series (in orange), using the same weight (one way) for all elements.

The series can be generalized as follows. A quadratic sequence increases by adding to each term a value that itself grows monotonically. For example, in the sequence of squares [1, 4, 9, 16, ...], each element increases by 3, 5, 7, and so on, meaning that 2 is added to the previous increment each time. This constant addition (2 in this example) is known as the second difference, denoted by d. If we let h_1 represent the first term of the sequence and k represent the initial increasing value, the general formula for our quadratic sequence is given in Equation 1.

$$h_n = \begin{cases} h_1, & n = 1 \\ h_{n-1} + d \cdot (n-1) + k, & n > 1 \end{cases} \tag{1}$$

On the other hand, the geometric sequence with increasing multipliers can be generalized as shown in Equation 2, assuming that h_1 represents the first term of the sequence, m represents the initial increasing multiplier and f represents its increasing value.

$$h_n = \begin{cases} h_1, & n = 1 \\ \text{round}(h_{n-1} \cdot (f \cdot (n-1) + m)), & n > 1 \end{cases} \tag{2}$$

We round the resulting value as history lengths are integer numbers.

To combine both functions, we just need to decide on a threshold t to change from quadratic series to super-exponential series, resulting in the Equation 3.

$$h_n = \begin{cases} h_1, & n = 1 \\ h_{n-1} + d \cdot (n-1) + k, & 1 < n < t \\ \text{round}(h_{n-1} \cdot (f \cdot (n-t+1) + m)), & n \geq t \end{cases} \tag{3}$$

Using Equation 3, the values shown in Figure 1 in the orange line are those obtained for the first 21 elements of the series when $h_1 = 2$, $d = 2$, $k = 1$, $f = 0.1$, $m = 1.1$, and $t = 15$. Using this series, the MPKI is reduced to 3.4276 (145.9 CycWpPKI). In addition, the design of TAGE is simplified, since it only requires a direct-mapped table, instead of a set-associative one.

2.2 How to be more confident

When multiple components generate predictions, selecting the most appropriate one for each branch becomes of utmost importance. The confidence level associated with each component plays a pivotal role in guiding this decision.

2.2.1 The loop predictor. The loop predictor is the most confident of all components [6]. Once it takes a decision, it should not be overridden. In TAGE-SC-L, the loop decision may be chosen over the TAGE prediction. Later, the selected prediction *fights* with the decision of the statistical correction. However, this fight does not consider whether the loop predictor was selected over TAGE as the prediction provider, but the decision is taken only considering the TAGE confidence. Instead, we propose that the loop prediction, if selected, remains the final prediction.

In addition, we refine the confidence of the loop predictor using values from 0 to 7. This confidence is given by the number of significant bits (in binary representation) of the product between the number of loop iterations predicted and the prediction counter (number of consecutive times that prediction was found correct). For example, if the product is 30 (11110_2), the confidence of that prediction is set to 5.

After refining the loop confidence, we use a set of saturating counters to decide between the TAGE-SC prediction and the loop prediction. These counters (*choosers* from now on) are similar to the ones employed in TAGE-SC-L: they are moved (increased or decreased) to the side of the component that in a mismatch of predictions provides the correct one. To select the chooser, we use information regarding loop confidence, TAGE confidence (i.e., confidence of the longest matching entry), and the decision taken between the TAGE and SC components. The hash used to select the chooser among the 32 choosers available for the loop predictor is depicted in Figure 2.

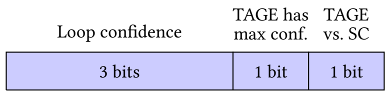

Figure 2: Hash used to select loop prediction choosers

Actually, the loop predictor is only chosen when its confidence is higher than two, leaving some of the counters unused.

2.2.2 The alternate prediction. The alternate prediction is the second longest matching history length. It is used in TAGE-SC-L when the longest matching history length is seen as newly allocated (i.e., it has low confidence −0 considering values from 0 to 3), and a chooser indicates that it should be used.

We improved the accuracy of the selection of the alternate prediction by (i) considering also to select it when the confidence is less than 2 (i.e., it is 0 or 1) and (ii) increasing the number of choosers, selecting them based on the confidence of the hit (longest match) entry. The resulting hash to select among the 256 choosers is depicted in Figure 3.

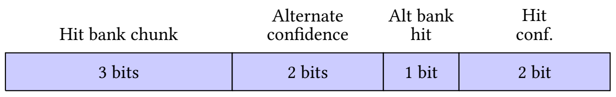

Figure 3: Hash used to select alternate prediction choosers

The hit bank chunk keeps the most significant bits of the hit bank. The alternate bank hit is needed to distinguish whether the prediction comes from the bimodal table or from an alternate bank. The hit confidence can only take two values, as only when the confidence is 0 or 1, the alternate bank can be selected (requiring only 128 choosers although 256 appear in the code).

We also keep track of a second alternate prediction in order to update its confidence counter when the branch is resolved. In this way, we can keep more confidence counters updated.

2.2.3 The statistical corrector (SC). In TAGE-SC-L, the decision to use TAGE or SC depends on the confidence of the hit bank in TAGE and the position of the sum from the perceptron-like tables [2] relative to a threshold. TAGE-SC-L defines three levels of confidence: high, medium, and low. If TAGE has high confidence and SC has low confidence, TAGE is selected. If one predictor has medium confidence and the other has high confidence, two choosers are used to determine which component to select. In all other cases, SC is chosen.

We improve that mechanism by using an array of choosers that are selected based on the confidence of TAGE, the outcome of the chooser of the alternate bank (see previous subsection), and the value of the perceptron sum (with respect to the threshold). By having more choosers and also considering the alternate bank selection, the accuracy is improved. The hash to choose among the 16 TAGE-SC choosers is depicted in Figure 4.

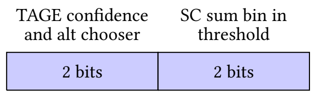

Figure 4: Hash used to select between TAGE and SC

If the TAGE confidence is 2 or 3, its value is used for the first two bits. Otherwise, the decision between alternate or hit bank (0 or 1) is used. To fill the last two bits, only three values are used, exactly as the three SC confidence options in the TAGE-SC-L code, leaving again some counters unused.

When all the techniques presented in this section to improve the choosers are enabled, the MPKI is reduced to 3.4216 (145.7 CycWpPKI).

2.3 Be careful when feeding the beast

TAGE-SC-L allocates entries in the TAGE tables even when the loop predictor is providing the prediction. As previously noted, the loop predictor typically exhibits very high confidence. We propose not to waste valuable TAGE entries when the loop predictor is selected as the prediction component and the prediction is successful, and in that case to completely bypass the TAGE allocation.

On the other hand, optimizing the replacement policy used in the TAGE tables is also critical, as mentioned in the TAGE cookbook by Seznec [5]. Our small contribution here is to increase (set) the useful (u) bit −used for replacement in TAGE− of the alternate bank when it correctly predicts the direction of the branch and the hit bank does not.

With these two techniques, the MPKI is further reduced to 3.4156 (145.5 CycWpPKI).

2.4 Other optimizations and fine tuning

We also implemented a minor tweak in the loop predictor to make an early change of the branch direction when the allocation is done using the wrong direction. In other words, if the direction expected to be the one of the loop exit repeats after allocation, we assume that it is not the exit direction and we update it, as well as the number of iterations seen (to 2). We also reset the age of the loop entry when there is a mismatch in the number of iterations between the new detected value and the stored one (last number of iterations detected).

Finally, a bit of fine-tuning has been performed to reach the size limit of the predictor and perform minor optimizations. The small IMLI SC table has been removed, the hash function of a bias SC component has been modified, and the second and third history tracking tables have been increased.

After these changes, the final MPKI is 3.4120 (145.4 CycWpPKI).

3 Discussion

The optimizations presented in this work are simple and do not increase the complexity of the predictor. They do not require multiplications or other complex operations. In some cases, the predictor is even simplified with respect to TAGE-SC-L, e.g., no associative searches in the TAGE tables or the use of a simple policy for feeding TAGE (removing the need of using random numbers in several cases).

Optimizations are consistent across different baselines and are expected to have more impact on baselines with less storage requirements or more realistic (e.g., lower number of history lengths). In fact, as we improved the baseline predictor, the effectiveness of the presented optimizations was reduced.

In addition, the proposed sequence for choosing history lengths can be integrated in any branch predictor that uses increasing history lengths, such as perceptron-based branch predictors [1].

We also believe that a loop predictor can be practically implemented in a real processor by updating it at commit time. To ensure correct predictions, it is necessary to track the number of branches

Table 1: Storage requirements of the proposed predictor

Component	Fields per entry [variable] and number of entries	Size	
Bimodal	Table [bimodal.table] ⟶ Prediction: 2^{16} 1-bit entries, hysteresis: 2^{14} 1-bit entries	10 KB	10 KB
TAGE	Low-history tables ⟶ Tag: 9 bits, conf: 3 bits, use: 1 bit, $2^{11}\times$ 14 entries	45.5 KB	166.0775 KB
	High-history tables ⟶ Tag: 12 bits, conf: 3 bits, use: 1 bit, $2^{11}\times$ 30 entries	120 KB	
	Global hist. register: 3157 bits, path hist. register: 27 bits, replacement counter: 11 bits	0.39 KB	
	Hit/alternate bank chooser [use_alt_vs_hit] ⟶ Counter: 6 bits, 256 entries	0.1875 KB	
Loop	Table [loop_predictor.table] ⟶ Tag: 10 bits, numIter: 10 bits, currentIter: 10 bits, ctr: 4 bits, age: 4 bits, dir: 1 bit, 2^6 entries	0.3047 KB	0.3320 KB
	Loop chooser [use_loop_vs_tagesc] ⟶ Counter: 7 bits, 32 entries	0.0273 KB	
SC	Thresholds and sum weights	0.0815 KB	15.4907 KB
	Three bias tables	1.125 KB	
	Global history table and history register	3.0049 KB	
	Path history table	3 KB	
	Fist local history table and local histories	3.6934 KB	
	Second local history table and local histories	1.6309 KB	
	Third local history table and local histories	1.5410 KB	
	IMLI table	1.0684 KB	
	SC chooser [use_sc_vs_tage] ⟶ Counter: 7 bits, 16 entries	0.0137 KB	
TOTAL			**191.90 KB**

in flight for each table entry, allowing the predictor to be adjusted accordingly. This approach is similar to the technique used in stride-based value prediction [4].

4 Conclusion

This work has proposed several optimizations to TAGE-SC-L, consisting of improving the sequence of the history lengths, performing better decisions to choose among prediction components, and improving the use of the large, but still limited, TAGE tables. The end result is an updated predictor, named TASQ-SC-L (from Sequence Quadratic), that obtains an average MPKI of 3.4120 and an average CycWpPKI of 145.4 cycles.

Acknowledgments

To Linnea and Marc, because when we play math, I also learn with them. To Alexandra, for her patience and support.

I would like to thank André Seznec and Daniel Jimenez for visiting the University of Murcia and delivering insightful talks on branch prediction, which motivated me to work on this topic.

This work has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation program (ECHO, grant agreement No 819134), from the MCIN/AEI/10.13039/501100011033/ and the “ERDF A way of making Europe”, EU (DAMAS, grant PID2022-136315OB-I00), and from the MCIN/AEI/10.13039/501100011033/ and the European Union NextGenerationEU/PRTR (HEEDA, grant TED2021-130233B-C33).

References

[1] Daniel A. Jiménez. 2016. Multiperspective Perceptron Predictor. In *5th JILP Workshop on Computer Architecture Competitions (CBP-5).* https://jilp.org/cbp2016/paper/DanielJimenez1.pdf

[2] Daniel A. Jiménez. 2005. Improved Latency and Accuracy for Neural Branch Prediction. *ACM Transactions on Computer Systems* 23, 2 (2005), 197–228. https://doi.org/10.1145/1062247.1062250

[3] André Seznec. 2016. TAGE-SC-L Branch Predictors Again. In *5th JILP Workshop on Computer Architecture Competitions (JWAC-5): Championship Branch Prediction (CBP-5).*

[4] André Seznec. 2018. Exploring value prediction with the EVES predictor. In *1st Championship Value Prediction (CVP-1).* 1–6. https://microarch.org/cvp1/papers/Seznec.pdf

[5] André Seznec. 2024. *TAGE-SC, an engineering cookbook.* Technical Report RR-9561. Inria. Available as Inria Research Report RR-9561.

[6] Sawan Singh, Arthur Perais, Alexandra Jimborean, and Alberto Ros. 2024. Alternate Path Âμ-op Cache Prefetching. In *51st International Symposium on Computer Architecture (ISCA).* 1230–1245. https://doi.org/10.1109/ISCA59077.2024.00092

A Cost Analysis

The storage cost of the proposed predictor is 191.82 KB. Details are given in Table 1. The cost is indeed a bit lower since some of the choosers are never employed, but the values in this table have been selected as described in the code of the predictor.