Pushing the branch predictability limits with the multi-poTAGE+SC predictor*

Pierre Michaud André Seznec
Inria

1 Introduction

The multi-poTAGE+SC predictor (Figure 1), which we submit to the unlimited size track, combines the multi-poTAGE and TAGE-SC predictors that have been submitted separately to CBP-4 by the first and second authors respectively[8, 12]. Both multi-poTAGE and TAGE-SC are based on the TAGE predictor. We combine them by replacing the TAGE component in TAGE-SC with the multi-poTAGE. On the CBP-4 traces, the proposed predictor achieves 1.691 mispredictions per thousand instructions (MPKI).

Sections 2 and 3 of this paper describe briefly the multi-poTAGE predictor. Section 4 describes the statistical corrector. Section 5 provides some performance analysis.

2 The poTAGE predictor

TAGE is derived from branch prediction by partial matching (PPM) [1, 7]. The poTAGE predictor, shown in Figure 2, is similar to TAGE [13, 11], except that we replace the USE_ALT_ON_NA mechanism with a *post-predictor* (poTAGE = post-predicted TAGE). In the submitted predictor, the taken/not-taken counter in each TAGE entry is 3-bit wide. Our post-predictor is a 1024-entry table, each table entry holding a 5-bit taken/not-taken counter. The post-predictor is indexed with the u bit and counter value of the longest hitting TAGE entry, and the counter values of the second and third longest hitting TAGE entries (10 index bits total). The 5-bit counter is used and updated like a conventional taken/not-taken counter [15].

In a TAGE predictor, the update is crucial, it must be done very carefully: only the longest hitting counter is updated, and only a few new entries are allocated upon a misprediction for path lengths greater than the longest hitting length [13, 11]. However, because we assume a huge predictor size, it is possible to use a more aggressive update policy to decrease mispredictions due to cold-start effects:

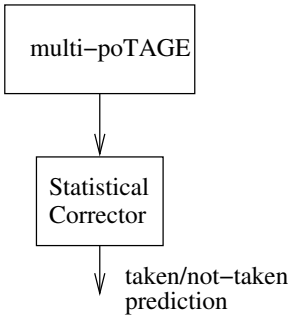

Figure 1: The multi-poTAGE+SC predictor.

- Instead of updating only the longest hitting counter, we update all the hitting counters, whether or not the branch was correctly predicted.
- Instead of allocating a few new entries, and only upon a misprediction, we allocate entries for all the path lengths greater than the longest hitting length, whether or not the branch was correctly predicted, and provided the entries can be stolen (u bit not set). We stop doing aggressive allocation for path lengths greater than 200 branches when all the hitting counters are saturated.

We use this aggressive update policy during what we call the *ramp-up* period. When the ramp-up period is over, we switch to the careful update policy implemented in the ISL-TAGE predictor [11].

The ramp-up period length is roughly proportional to the predictor size. For the submitted predictor, we set the ramp-up period to one million mispredictions.

3 The multi-poTAGE predictor

The multi-poTAGE predictor, depicted in Figure 3, consists of 5 different poTAGE predictors, P0, P1, P2, P3 and

*This work was partially supported by the European Research Council Advanced Grant DAL No 267175

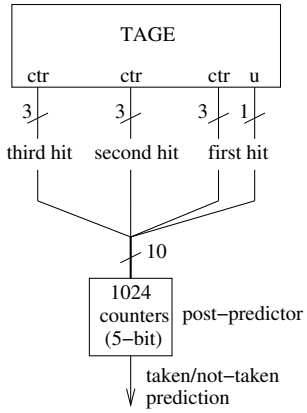

Figure 2: The poTAGE predictor. The post-predictor takes as input the u bit and taken/not-taken counter of the longest hitting TAGE entry, and the taken/not-taken counters of the second and third longest hitting TAGE entries.

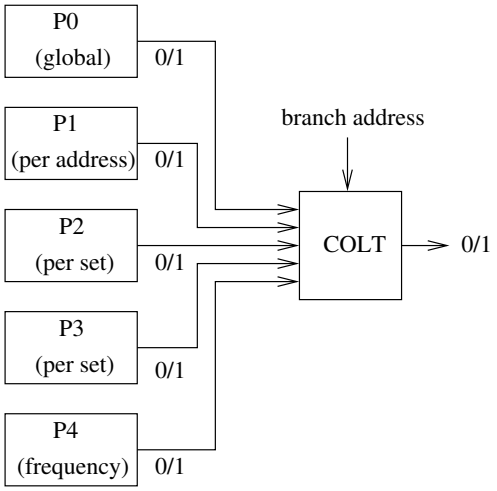

Figure 3: The multi-poTAGE predictor.

P4, combined via COLT fusion [6]. P0 is the global path poTAGE described in Section 2. The other poTAGEs do not use a global path:

- P1 uses 32 per-address subpaths
- P2 uses 16 per-set subpaths with 128-byte sets
- P3 uses 4 per-set subpaths with 2-byte sets
- P4 uses 8 frequency-based subpaths

P2 and P3 are two per-set two-level predictors [16]. Ishii et al. also used per-set schemes in their recent FTL++ predictor [3].

P4 uses a new sort of first-level history, frequency-based subpaths. In a two-level predictor, the first-level history consists of a set of subpath that we call a *path spectrum*. At prediction time, a subpath is selected from the path spectrum. The selected subpath is used to access the second-level history (here, poTAGE), yielding a prediction. The spectrum of P4 consists of 8 subpaths. The subpath is selected as follows. The branch address is used to index a Branch Frequency Table (BFT). If the BFT is large enough, each static branch uses a distinct BFT entry. Each BFT entry holds a counter indicating the current *frequency* of the static branch. The frequency of a branch is the number of times the branch has been executed until now since the counter was reset[1].

Predictor P4 seeks to exploit correlations between branches having (roughly) the same frequency. Each of the 8 subpaths $S[0]$ to $S[7]$ corresponds to a distinct frequency bin. Let F_{max} be the maximum branch frequency so far. Branches whose frequency lies in $[F_{max}/2, F_{max}]$ are predicted with subpath $S[0]$. Branches whose frequency lies in $[F_{max}/4, F_{max}/2]$ use subpath $S[1]$. Branches whose frequency lies in $[F_{max}/8, F_{max}/4]$ use subpath $S[2]$. And so forth.

The multi-poTAGE paper provides a more detailed description [8].

4 The Statistical Corrector

The Statistical Corrector predictor was introduced as an adjunct predictor to TAGE in [14, 11]. It allows to better predict the class of branches that are not completely determined by the path, but are globally biased by path. The correction aims at detecting the unlikely predictions from TAGE and to revert them: the prediction flowing from TAGE, its confidence level, as well as information on the branch (address, global history, global path, local history) are presented to the Statistical Corrector predictor which decides whether or not to invert the prediction.

In this submission, we use the output of the multi-poTAGE predictor and the prediction outputs of its internal components as inputs of a Statistical Corrector.

We use a statistical corrector similar to the one used in the companion submission [12]. It is a perceptron-inspired [4] Statistical Corrector [14, 11], that combines multiple components:

- the Bias table: indexed through the PC and the multi-poTAGE predicted direction (after selection through

[1] In the submitted branch prediction algorithm, we reset the frequency counters only once, when the simulation starts.

the COLT component).

- A second Bias table indexed with the PC, the multi-poTAGE output and the 5 poTAGE component outputs.
- 4 LGEHL components, 3 using a 16-entry history table, and one using 64K-entry history table. Each one features 15 tables.
- 4 perceptron-derived local history components using similar history tables. In these perceptron-derived components, we use the MAC representation of the counters[9]; a counter is associated with 6 consecutive bits of history. Each of these components features 10 tables.
- 2 perceptron-derived components using respectively global branch history and global path history: 10 tables each.
- a global history GEHL component: 209 tables
- a global history component inspired from the MAC-RHSP predictor [9]; a counter is associated with 6 consecutive bits of history and part of the global branch history (1/3) is hashed with the PC: 80 tables.
- Two path skeleton history GEHL components. The first path skeleton are the taken branches whose targets are not too close to the branch source. By too close, we mean 16 bytes for backward branches and 128 bytes for forward branches. The second path skeleton registers the branch in the path only if it was not among the last 8 encountered branches. These components feature 15 tables each.
- Two path skeleton history perceptron-derived components: 10 tables each.

All the tables hold 8 bit counters. The prediction is computed as the sign of the sum of the (centered) predictions read on all the Statistical Corrector tables: a total of 461 counters are summed.

The Statistical Corrector predictor tables are updated using a dynamic threshold policy as suggested for the GEHL predictor [10]. As suggested in [5], we use a PC-indexed table of dynamic threshold, which yields marginal benefit.

Any of these components has only a limited accuracy impact, but if one removes all the components exploiting local history or exploiting global branch/path history the impact on accuracy is more significant.

configuration	MPKI increase
MP+SC (base)	0
PPM	+30.8%
P0	+18.7%
MP	+5.7%
P0+SC	+5.6%
MP+SCg	+2.5%
P0+SCg	+12.2%
PPM+SCg	+13.7%

Table 1: **Average MPKI increase for various configurations. MP is the multi-poTAGE (5 poTAGEs + COLT); SC is the statistical corrector; SCg is the statistical corrector reduced to its bias and global-history components; P0 is the global-path poTAGE alone; PPM is P0 without the post-predictor. The base MPKI for all configurations is that of MP+SC.**

4.1 Choosing between multi-poTAGE and the statistical corrector ouputs

The prediction flowing out of the statistical corrector is often more accurate than the poTAGE prediction. However using a chooser results in a higher accuracy than just using the statistical corrector output.

The best chooser we have experimented uses a GEHL + LGEHL structure with limited number of tables (i.e. short histories). It is indexed with the PC, the multi-poTAGE prediction, the confidence (high or not) of the SC prediction.

5 Performance analysis

The multi-poTAGE and statistical corrector configurations used here are very close to those described in the companion papers [8, 12], where they were analyzed separately[2]. Table 1 provides some performance analysis about the combined multi-poTAGE+SC predictor. We disabled some features of multi-poTAGE+SC to see how they impact the overall performance.

When we apply the statistical corrector on the multi-poTAGE predictor, more than half of the accuracy gain comes from the bias and global-history components of the statistical corrector (cf. Table 1, compare MP, MP+SC and MP+SCg).

The local-history statistical corrector components have more effect when statistical correction is applied on a single global-path poTAGE: P0+SCg has 6.3% (12.2% vs.

[2]The configuration parameters used here are slightly different from those used in [8, 12] because of the memory size constraint.

5.6%) more mispredictions than P0+SC. That is, the non-global components of the multi-poTAGE (P1,P2,P3,P4) and the non-global components of the statistical corrector have partially redundant effects. Still, the non-global components of multi-poTAGE bring a 5.3% prediction accuracy gain (MP+SC vs. P0+SC).

It is interesting to notice that when statistical correction is applied to P0, removing the post-predictor from P0 has little impact: P0+SCg and PPM+SCg have close MPKIs (respectively +12.2% and +13.7% more mispredictions than MP+SC). However, removing the post-predictor from an isolated poTAGE degrades prediction accuracy significantly: PPM has about 10% more mispredictions than P0 (+30.8% vs. +18.7%).

In a PPM-like predictor, the longest matching context is not always the most accurate [13, 2]. It is this problem that the post-predictor and the statistical corrector try to solve, but the statistical corrector is a more effective solution.

It is interesting to notice that statistical correction brings the same relative improvement over a single poTAGE (P0+SCg vs. P0) than over the multi-poTAGE (MP+SC vs. MP).

6 Conclusion

On the CBP-4 traces, the multi-poTAGE+SC predictor achieves about 5 % fewer mispredictions than the multi-poTAGE predictor or the TAGE-SC predictor that are presented in two other submissions by respectively the first and the second author.

The non-global components of the multi-poTAGE are responsible for most of the prediction accuracy gain of multi-poTAGE+SC over TAGE-SC. The effectiveness of the statistical corrector lies in the fact that, in a PPM-like predictor such as TAGE or poTAGE, the longest matching context is not always the most accurate.

References

[1] I.-C.K. Chen, J.T. Coffey, and T.N. Mudge. Analysis of branch prediction via data compression. In *Proc. of the 7th Int. Conference on Architectural Support for Programming Languages and Operating Systems*, 1997.

[2] H. Gao and H. Zhou. PMPM: Prediction by combining multiple partial matches. *Journal of Instruction Level Parallelism*, May 2007.

[3] Y. Ishii, K. Kuroyanagi, T. Sawada, M. Inaba, and K. Hiraki. Revisiting local history for improving fused two-level branch prediction. In *Proc. of the 2nd JILP Workshop on Computer Architecture Competitions*, June 2011.

[4] D.A. Jiménez and C. Lin. Neural methods for dynamic branch prediction. *ACM Transactions on Computer Systems*, 20(4), November 2002.

[5] Daniel A. Jiménez. OH-SNAP: Optimized hybrid scaled neural analog predictor. In *Proc. of the 3rd Championship Branch Prediction*, 2011.

[6] G. H. Loh and D. S. Henry. Predicting conditional branches with fusion-based hybrid predictors. In *Proc. of the Int. Conf. on Parallel Architectures and Compilation Techniques*, 2002.

[7] P. Michaud. A PPM-like, tag-based predictor. *Journal of Instruction Level Parallelism*, April 2005.

[8] P. Michaud. Five poTAGEs and a COLT for an unrealistic predictor. In *Proc. of the 4th Championship Branch Prediction*, 2014.

[9] A. Seznec. Revisiting the perceptron predictor. PI-1620, IRISA, May 2004.

[10] A. Seznec. Analysis of the O-GEHL branch predictor. In *Proc. of the 32nd Int. Symp. on Computer Architecture*, june 2005.

[11] A. Seznec. A new case for the TAGE branch predictor. In *Proc. of the 44th Int. Symp. on Microarchitecture*, December 2011.

[12] A. Seznec. Exploring the branch prediction accuracy limits for the TAGE-SC branch predictor. In *Proc. of the 4th Championship Branch Prediction*, 2014.

[13] A. Seznec and P. Michaud. A case for (partially) tagged geometric history length branch prediction. *Journal of Instruction Level Parallelism*, February 2006.

[14] André Seznec. A 64 kbytes ISL-TAGE branch predictor. In *Proc. of the 3rd Championship Branch Prediction*, June 2011.

[15] J. E. Smith. A study of branch prediction strategies. In *Proc. of the 8th Int. Symp. on Computer Architecture*, 1981.

[16] T.-Y. Yeh and Y. N. Patt. A comparison of dynamic branch predictors that use two levels of branch history. In *Proc. of the 20th Int. Symp. on Computer Architecture*, 1993.