TAGE-SC-L Branch Predictors

André Seznec

TAGE-SC-L Branch Predictors*

André Seznec
INRIA/IRISA

Outline

The TAGE predictor [12] is considered as one of the most storage effective global branch/path history predictors. It has been shown that associated with small adjunct predictors like a statistical corrector (SC for short) and/or a loop predictor (L for short) [11, 10], TAGE can even be more effective. In this study, we explore the performance limits of these TAGE-SC-L predictors for respectively 32Kbits storage budget, 256 Kbits storage budget and quasi-unlimited ($<$ 2 Gbits) storage budget.

With a 32Kbits storage budget, only a very limited storage budget can be invested in the adjunct predictors. Then our submitted predictor used most of its storage budget on the TAGE predictor and features only a small loop predictor LP and a simple corrector filter CF. The submitted 32Kbits predictor achieves 3.315 MPKI on the CBP-4 traces.

With a larger storage budget, one can invest more significant storage budget in the adjunct predictors. The submitted 256Kbits TAGE-SC-L predictor features a TAGE predictor, a loop predictor LP and a quite complex ($\approx$ 45 Kbits) statistical corrector SC that exploits local history, global branch history and return-associated branch history. The 256Kbits TAGE-SC-L predictor achieves 2.365 MPKI on the CBP-4 traces.

The no-limit budget allows to use a statistical corrector build with many components exploiting global branch and path histories, local histories and some form of skeleton histories. The submitted predictor achieves 1.782 MPKI on the CBP-4 traces.

1. General view of the TAGE-SC-L predictor

The TAGE-SC-L predictor consists of three components: a TAGE predictor, a statistical corrector predictor and a loop predictor (Figure 1).

The TAGE predictor provides the main prediction. Then this prediction is used by the statistical corrector predictor which role consists in confirming (general case) or reverting the prediction. The statistical predictor reverts the prediction when it appears that, in similar circumstances (prediction, branch histories, branch confidence, ..) TAGE statistically mispredicted. The loop predictor is useful to predict regular loops with long loop bodies.

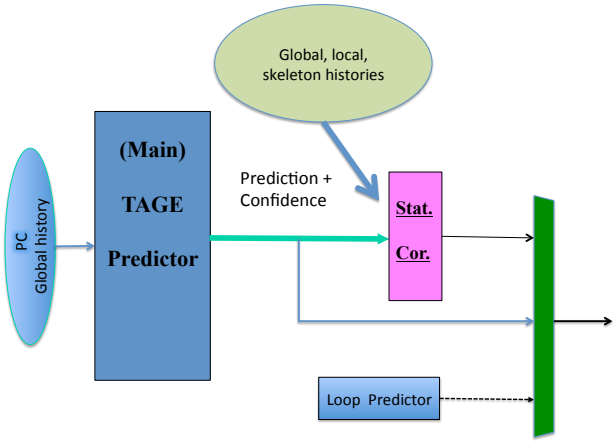

Figure 1. The TAGE-SC-L predictor: a TAGE predictor backed with a Statistical Corrector predictor and a loop predictor

2. The TAGE conditional branch predictor

The TAGE predictor was described in [12] and [9]. Only marginal modifications are introduced in this study.

Figure 2 illustrates a TAGE predictor. The TAGE predictor features a base predictor T0 in charge of providing a basic prediction and a set of (partially) tagged predictor components Ti. These tagged predictor components Ti, $1 \leq i \leq M$ are indexed using different history lengths that form a geometric series [6], i.e, $L(i) = (int)(\alpha^{i-1} * L(1) + 0.5)$.

In practice, the set of history lengths used in the submitted predictors were obtained, first through exploring the use of a geometric series, then through refinements. These refinements led to limited benefits ($< 0.5\%$) on accuracy.

Throughout this paper, the base predictor will be a sim-

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

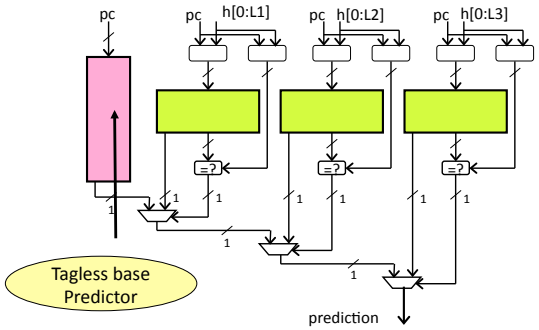

Figure 2. The TAGE predictor synopsis: a base predictor is backed with several tagged predictor components indexed with increasing history lengths

ple PC-indexed 2-bit counter bimodal table; in order to save storage space, the hysteresis bit is shared among several counters as in [7] for the limited size predictor.

An entry in a tagged component consists in a signed counter *ctr* which sign provides the prediction, a (partial) tag and an unsigned useful counter *u*. In our study, *u* is a 2-bit counter and *ctr* is a 3-bit counter for the limited size predictors, and a 5-bit counter for the no-limit predictor.

A few definitions and notations The *provider component* is the matching component with the longest history. The alternate prediction *altpred* is the prediction that would have occurred if there had been a miss on the provider component.

If there is no hitting component then *altpred* is the default prediction.

2.1 Prediction computation

The prediction selection algorithm is exactly the same as described [12] or [9].

At prediction time, the base predictor and the tagged components are accessed simultaneously. The prediction is provided by the hitting tagged predictor component that uses the longest history. In case of no matching on tagged predictor component, the default prediction is used.

However we remarked that when the confidence counter of the matching entry is null, on some traces, the alternate prediction *altpred* is sometimes more accurate than the "normal" prediction. This property is global to the application and can be dynamically monitored through 4-bit counters (*USE_ALT_ON_NA* in the simulator). *For this submission, we use a small array of USE_ALT_ON_NA counters indexed by the program counter (*$< 0.5\%$ *reduction of the misprediction rate).*

Prediction computation summary Therefore the prediction computation algorithm is as follows:

1. Find the longest matching component and the alternate component
2. if (the confidence counter is not weak or USE_ALT_ON_NA is negative) then the provider component provides the prediction else the prediction is provided by the alternate component

2.2 Updating the TAGE predictor

Update on a correct prediction by the longest marching entry The prediction counter of the provider component is updated. When the prediction is low confidence, the alternate provider is also updated.

Update on a misprediction by the longest matching entry First we update the provider component prediction counter.

As a second step, if the provider component Ti is not the component using the longest history (i.e., $i < M$) , we try to allocate entries on predictor, components Tk using a longer history than Ti (i.e., $i < k < M$).

For the limited size predictor, for a medium to large number of TAGE tables, allocating two entries on a misprediction is slightly better than allocating a single entry (due to faster warming of the predictor). However, for a 32Kbits predictor, we also found that when the misprediction ratio is high, one should avoid to allocate many entries (avoiding pollution). Therefore we dynamically decide on the number of entries (one or two) to allocate on a misprediction.

For the no-limit predictor, systematically allocating many entries on a misprediction allows to reduce the warm-up time of the predictor.

The M-i-1 u_j useful bits are read from predictor components Tj, $i < j < M$. The allocation algorithm chose entries for which the useful bits are null, moreover we guarantee that the entries are not allocated in consecutive tables. When an entry is allocated, its prediction counter is set in weak mode and its useful bit is set to null.

Updating the useful counter *u* The useful counter *u* of the provider component is incremented whenever the actual prediction is correct and the alternate prediction *altpred* is incorrect.

In order to avoid the useful bits to stay forever set, we have to implement a reset policy.

In this Championship, we have used a new variation of the reset policy on the small predictor. On allocation of new entries, we dynamically monitor the number of successes and fails when trying to allocate new entries after a misprediction. This monitoring is performed through a single counter called TICK in the simulator. On a failure, the u counter of the associated entry is probabilistically decremented, the decrement probability increases with the TICK counter value.

For the 256Kbits predictor, we implement the smooth resetting all the u counters proposed in [10].

For the no-limit predictor, the replacement policy is not a real issue.

Tag width tradeoff Using a large tag width leads to waste part of the storage while using a too small tag width leads to false tag match detections [12]. Systematic experiments confirmed that using narrower tags on the tables with smaller history lengths leads to the best storage tradeoff.

For the submitted limited size predictors, increasing the tag width by 1 bit on all tables would result in unsignificant accuracy benefit, while decreasing this width results in about 1% misprediction rate increase.

For the no-limit predictor, we set the tag width to 16 for all tables; experiments with larger width did not lead to higher accuracy.

The global history vector For the limited size predictors, the usual combination of the global branch history vector and a short path history (limited to 1 bit per branch) was found to be slightly outperformed by a global history vector introducing much more information for unconditional branches (4 bits per unconditional branch). Using 7 bits per branch would be better, but consumes storage bits. This is used for the no-limit predictor.

3 The loop predictor component

The loop predictor simply tries to identify regular loops with constant number of iterations.

The implemented loop predictor provides the global prediction when a loop has successively been executed 7 times with the same number of iterations. The loop predictor used in the submission features only 64 entries and is 4-way skewed associative.

Each entry consists of a past iteration count on 10 bits, a retire iteration count on 10 bits each , a partial tag on 10 bits, a confidence counter on 4 bits, an age counter on 4 bits and 1 direction bit i.e. 39 bits per entry. The loop predictor features only a limited number of entries 16 for the 32Kbits predictor and 32 for the 256Kbits predictor.

Replacement policy is based on the age. An entry can be replaced only if its age counter is null. On allocation, age is first set to 7. Age is decremented whenever the entry was a possible replacement target and incremented when the entry is used and has provided a valid prediction. Age is reset to zero whenever the branch is determined as not being a regular loop.

The loop predictor reduces the misprediction rate by approximately 1% for the 256Kbits predictor and 1.2 % for the 32Kbits predictor and only 0.4 % on the no-limit predictor.

4 Statistical Corrector Predictor

The TAGE predictor is very efficient at predicting very correlated branches even when the correlation is with very remote branches. However TAGE fails at predicting statistically biased branches e.g. branches that have only a small bias towards a direction, but are not strongly correlated with the history path. On some of these branches, the TAGE predictor performs sometimes worse than a simple PC-indexed table of wide counters.

In [10, 11], we introduced the Statistical Corrector predictor to better predict this class of statistically biased branches, The correction aims at detecting the unlikely predictions and to revert them: the prediction flowing from TAGE as well as information on the branch (address, global history, global path, local history) are presented to Statistical Corrector predictor which decides whether or not to invert the prediction. Since in most cases the prediction provided by the TAGE predictor is correct, the Statistical Corrector predictor can be quite small.

4.1 32Kbits predictor: The Corrector Filter

For the 32Kbits predictor, only a small storage can be devoted to the statistical corrector, since the benefit of increasing the TAGE predictor size is important. The Corrector Filter is an associative table used to monitor the mispredictions from TAGE. The mispredictions (PC and predicted direction) are stored in the Corrector Filter with a low probability. On a hit on the Corrector Filter, the prediction is inverted if the entry indicates to revert the prediction with a high confidence. Replacement policy favors the replacement of entries with low confidence or which indicate not to invert the prediction.

A 64-entry 2-way skewed associative Corrector Filter is used with a 13-bits entry (6-bit counter + 7-bit tag). As an optimization, high confidence branches flowing from TAGE are not inverted by the Corrector Filter.

The Corrector Filter reduces the overall misprediction rate by about 1.5%.

4.2 256Kbits: Multi-GEHL Statistical Corrector Predictor

For the 256 Kbits predictor, we use an enhanced version of the statical corrector predictor introduced in [10] and in [11].

The Multi-GEHL Statistical Corrector, MGSC, used in the submitted predictor is composed of 6 different components:

- The Bias component: indexed through the PC and the TAGE predicted direction. The Bias component is made of two tables indexed with distinct hashing functions to limit entry conflict impact.
- 5 GEHL-like components respectively indexed using the global conditional branch history (4 tables), the return-stack-associated branch history (4 tables), a 256-entry local history (3 tables) and 2 16-entry local history (4 tables and 3 tables). The GEHL components do not feature a PC indexed table. Only the global conditional branch history component uses the TAGE output in its indices.

The return stack associated branch history work as follows. History is stored in the same return stack as calls. On a call, the top of the stack is copied on the newly allocated entry in the stack, after the associated return, the branch history reflects the branches that were executed before the call. This allows to capture some correlations that could be lost during the execution of the called function. The benefit of the associated GEHL component is only of 0.7 %. Investing its storage budget in doubling the global history GEHL component would bring back about half of this benefit.

The use of two distinct local histories was introduced by [1]. The benefit of the 2nd local history is marginal (about 0.5 % misprediction reduction). However adding a 3rd local history table with also 16 history entries, with a different hashing function on the local history table induces an extra accuracy benefit in the same range.

All tables are 5 or 6 bit counters. The Bias tables as well the tables with the shorter history of each GEHL component are 6 bits wide, the other tables are 5 bits wide. The prediction is computed as the sign of the sum of the (centered) predictions read on all the Statistical Corrector tables, moreover we integrate the result of the TAGE prediction through adding twice the number of multi-GEHL tables multiplied by the direction (+1,-1).

The MGSC predictor tables are updated using a dynamic threshold policy as suggested for the GEHL predictor [6]. *As suggested in [3], a PC-indexed table of dynamic threshold is used (32 entries), enabling marginal benefit (0.1% misprediction decrease).*

The role of the Bias component indexed with the PC **and** the TAGE output is fundamental. Most branches are correctly predicted in most cases by the TAGE predictor. The associated Bias tables entries tend to saturate on these branches. This allows to use relatively small other components in the MGSC.

The output of MGSC is generally more accurate than the output of the TAGE predictor. However when the output of TAGE is high confidence (see [**?**]) and the output of MGSC is low confidence (i.e. $|sum| < threshold$), TAGE is generally a little bit more accurate. Then, in the general case we select the MGSC output, apart when TAGE output is high confidence and MGSC output is low confidence then the prediction is chosen based on 3 counters respectively monitoring the behavior of very low MGSC confidence ($|sum| < threshold/3$), medium low MGSC confidence($threshold/3 < |sum| < 2*threshold/3$), and high low MGSC confidence ($2 * threshold/3 < |sum| < threshold$). This simple policy enables about 0.7% misprediction reduction.

In practice, the MGSC used in the submitted 256Kbits predictor features less than a total of approximately 46 Kbits storage bits. It reduces the total misprediction rate by about 6.8%.

4.3 Statistical Corrector for the no-limit predictor

For the no-limit predictor, we use a perceptron-inspired [2] Statistical Corrector, that combines multiple components:

- the Bias component: indexed through the PC and the TAGE predicted direction.
- 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[5]; a counter is associated with 6 consecutive bits of history. Each of these components features 10 tables.
- 2 perceptron-derived using respectively global history and global path history: 10 tables each.
- a global history GEHL component: 209 tables. The 5 shortest history components are indexed using the TAGE output in the index (¡0.1% gain through this trick).
- a global history component inspired from the MAC-RHSP predictor [5]; 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 which 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. Each of these components features 15 tables.

- Two path skeleton history perceptron-derived components: 10 tables each.

All tables are 8 bit counters, wider counters do not bring any significant benefit (less than 0.1 %). The prediction is computed as the sign of the sum of the (centered) predictions read on all the Statistical Corrector tables: a total of 460 counters are summed.

In practice, we tried every combination of information from the control flow that we expected to have a possible correlation with the direction output; some brought extra accuracy and were retained, others only polluted the prediction and were rejected.

As for 256Kbits, the Statistical Corrector predictor tables are updated using a dynamic threshold policy. [6] and a PC-indexed table of dynamic threshold [3] is used enabling marginal benefit.

Choosing between TAGE and statistical corrector outputs The prediction flowing out from the statistical corrector is generally more accurate than the TAGE prediction. However using a chooser results in a slightly better 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 TAGE prediction, the confidence (high or not) of the TAGE prediction, the confidence (high or not) of the SC prediction.

This chooser marginally outperforms the chooser described for the 256 Kbits predictor (0.005 misp/KI).

5 The submitted limited size predictors

5.1 32Kbits TAGE-SC-L predictor

The TAGE component consists in a (4Kbits prediction, 1Kbits hysteresis), 10 tagged tables with respective histories $\{4, 9, 13, 24, 37, 53, 91, 145, 226, 359\}$, $\{256, 256, 256, 256, 128, 256, 128, 64, 64\}$ entries and $\{7, 7, 7, 8, 9, 10, 10, 11, 13, 13\}$ tag widths and 32 entries USE_ON_ALT_NA and a few counters for a total of 31639 bits.

The loop predictor features 16 39-bits entries, i.e 624 bits.

The corrector filter features 64 13-bits entries, i.e 832 bits.

Including the 359 bits of global branch history, the 27-bit global path history, the submitted TAGE-SC-L predictor features a total of **33,465** storage bits.

5.2 The 256Kbits TAGE-SC-L predictor

The TAGE component consists in a (16Kbits prediction, 4Kbits hysteresis), 15 tagged tables with respective history lengths $\{6, 10, 18, 25, 35, 55, 69, 105, 155, 230, 354, 479, 642, 1012, 1347\}$, with $\{1K, 1K, 1K, 2K, 1K, 1K, 1K, 1K, 1K, 512, 512, 512, 256, 128, 128\}$ entries and $\{7, 9, 9, 9, 10, 11, 11, 12, 12, 12, 13, 14, 15, 15, 15\}$ tag widths and 2x256 entries 5-bit USE_ON_ALT_NA counters and a few counters. This leads to a total of 214,376 bits including the 1347 bits of global branch history, and the 27-bit global path history.

The loop predictor features 32 39-bits entries, i.e 1,248 bits.

The MGSC features a Bias component with two tables(128 entries each) a global conditional branch history GEHL (2 256-entries + 2 512-entries tables), a return-stack-associated history GEHL (2 256-entries + 1 512-entries tables, a 16-entry history stack), a 256-entries local history GEHL (2 512-entries + 1 1024-entries table) and two 16-entry local history GEHL (2 256-entries + 2 512-entries tables). All tables are 5 or 6 bit counters as mentioned in the previous section. It also features a 32 entries dynamic threshold table. Including local history tables, and the three selection counters, the MGSC features a total of 46,809 storage bits.

The submitted 256Kbits TAGE-SC-L predictor features a total of **262,433** storage bits.

Performance analysis of the statistical corrector As already mentioned, the statistical corrector reduces the misprediction rate of the TAGE-L predictor by 6.8% (2.538 MPKI).

If one considers only the three local components (about 30 Kbits total) then the misprediction rate is increased by about by 1.5 % (2.401 MPKI).

If one considers only the global history component and the return-stack associated history (about 15 Kbits total) then the misprediction rate increase by 3.6 % (2.452 MPKI).

If one doubles the size of all the GEHL components in the statistical corrector (i.e. adds about extra 40 Kbits) then the global misprediction rate is decreased by only 1.2 % (2.338 MPKI).

A more realistic 256 Kbits TAGE-SC-L predictor Through uncommenting "#define REALISTIC" on the first

line of the *predictor.h* file of the submission , one gets a more realistic configuration of the TAGE-SC-L predictor with statistical corrector featuring a global branch GEHL (4 tables) and one local history GEHL (4 tables), a TAGE predictor with 12 equal size tagged tables, achieving only 2.7 % extra mispredictions (2.430 MPKI).

In practice, benefits associated with using three local history GEHL components instead of one and the return-stack associated history GEHL component are not worth: less than 1 % of extra mispredictions if one invest the freed storage budget in the remaining components of the statistical corrector.

6 Accuracy analysis of the no-limit TAGE-SC-L predictor

The submitted predictor features many components. We analyze here the benefits of the components in terms of extra mispredictions over the submitted predictor when the component is removed.

6.1 Exploring the best configuration

We systematically explored several parameters for finding the best possible configuration. Most tables were dimensioned with 1 million or $\frac{1}{2}$ million entries, and we varied the number of tables and the history lengths to find the best configuration. Through this process, we reduce the misprediction rate by only 0.7 %.

The simulator footprint does not exceed 500 Mbytes, tests with larger tables did not result in any visible accuracy benefit.

6.2 The TAGE predictor

The TAGE component presented in the submission would observe 23.5 % more mispredictions than the complete predictor. In [8], we mentioned that the TAGE predictor accuracy saturates around 18-20 tagged components. We verify the same behavior on the distributed traces. The TAGE predictor with only 20 tagged tables observes only 19.5 % extra mispredictions, but the complete predictor with only 20 tagged tables would encounter 0.4 % more mispredictions than the submitted predictor.

However the extra mispredictions due to extra tagged components are essentially captured by the statistical corrector.

6.3 Removing the TAGE predictor

If one removes the TAGE predictor from our submission then the predictor becomes a perceptron-inspired predictor exploiting local and global branch/history. This predictor achieves 4.7 % more mispredictions than the submitted predictor.

6.4 The loop predictor

Without the loop predictor, the misprediction rate is increased by 0.4 %.

6.5 The chooser

Systematically using the SC prediction instead of choosing through our chooser between SC output and TAGE output would result in an accuracy loss of 3.2 %.

6.6 The Bias component

The bias component tends to record the statistical bias of the output of TAGE on branch B and a predicted direction D. That is when branch B is predicted in a direction D and it has been observed that in most similar cases, B was mispredicted then the bias component will indicate to revert the prediction.

Removing the use of the TAGE output in the index of the bias component would change our submission in a more classical hybrid predictor, it would also result in an increase of 1.6 % more mispredictions: this is the most influent single table in the set of 460 tables of the statistical corrector.

6.7 Local history components

Ignoring local history in the statistical corrector would result in a 3.9% increase of the total misprediction rate. Further eliminating the use of local history from the chooser has no significant impact (4.1%).

Using several local histories Using several local history tables is beneficial. Using a single local history with 32K-entry history table would result in an accuracy loss of 1.7 %. The use of several distinct local histories was introduced by [1]. In practice,

3 16-entry local history table sizes indexed with different hashing functions, we divide the con flow sttsome correlation between branches that would map on a single entry of 16-entry history table.

Perceptron-derived local history components Removing the perceptron-derived local history components would result in a 0.4 % increase of the misprediction rate.

Local history GEHL components Removing the local history GEHL components would result in a 1.3 % increase of the misprediction rates.

6.8 The global history/path components

Several global history components are used in the SC predictor. Globally removing these components results in an accuracy loss of 3.7%.

However, removing only one of them has small impact on accuracy as listed below, that is these components are essentially capturing the same kind of correlation.

The long history components Removing only the global history GEHL component from our submitted predictor would result in 0.3 % more mispredictions. Removing the MAC-RHSP derived component would result in an increase of 0.25 % of the misprediction rate. Removing both components results in an increase of 1.8 % of the mispredictions, that is the MAC-RHSP component and the GEHL component are essentially capturing similar correlation on branch outputs.

The perceptron-derived global history and global path components Removing these two components results in an insignificant increase of 0.1 % of the misprediction rate.

6.9 Skeleton path history components

Removing only the skeleton path history components would result in a marginal 0.4 % increase of the misprediction rate.

However, if we remove all the global branch/path history components **and** the skeleton path history components then a 5.1 % misprediction rate increase is encountered. This shows that the skeleton path history components essentially capture correlations that are also captured by the global history components.

6.10 Comparison with the GTL predictor

The GTL predictor [8] was the most accurate published "unlimited storage size" predictor so far. The TAGE-SC-L significantly outperforms it due to several reasons: 1) use of alternate forms of global history components (MAC-RHSP component, perceptron-inspired components), 2) introduction of local history components 3) use of the Bias component.

However TAGE-SC-L achieves about 5% more mispredictions than the poTAGE-SC predictor submitted to same championship [4].

7 Conclusion

The TAGE-SC-L predictor can be adapted to various predictor storage budgets.

For small storage budgets (e.g. 32Kbits), large accuracy benefits is obtained through increasing the number or the sizes of the TAGE tagged tables, therefore only very small loop predictors and statistical correctors can be used.

For larger storage budgets, the adjunct predictors can implement more complex schemes as illustrated by the multi-GEHL statistical corrector used for the 256 Kbits predictor.

The submitted predictors were optimized to achieve the ultimate accuracy at the fixed storage budgets of the championship. This leads to use unrealistic number of tables, different predictor table sizes, tag widths and different schemes in the multi-GEHL statistical corrector. However, simpler designs with storage budgets in the same range, but using much smaller number of tables would achieve prediction accuracy in the same range (less than 3 % extra mispredictions).

Using the TAGE-SC-L scheme with a quasi unlimited budget allows to reach substantially better accuracy than the one of the 256Kbits predictor (more than 20 % less mispredictions). However this is achieved through using a particularly unrealistic number of components in the predictor and trying to exploit all forms of branch and path history that we could imagine.

References

[1] Y. Ishii, K. Kuroyanagi, T. Sawada, M. Inaba, and K. Hiraki. Revisiting local history for improving fused two-level branch predictor. In *Proceedings of the 3rd Championship on Branch Prediction, http://www.jilp.org/jwac-2/*, 2011.

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

[3] Daniel A. Jiménez. Oh-snap: Optimized hybrid scaled neural analog predictor. In *Proceedings of the 3rd Championship on Branch Prediction, http://www.jilp.org/jwac-2/*, 2011.

[4] Pierre Michaud and André Seznec. Pushing the branch predictability limits with the multi-potage+sc predictor. In *Proceedings of the 4th Championship on Branch Prediction, http://www.jilp.org/cbp2014/*, 2014.

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

[6] A. Seznec. Analysis of the O-GEHL branch predictor. In *Proceedings of the 32nd Annual International Symposium on Computer Architecture*, june 2005.

[7] A. Seznec, S. Felix, V. Krishnan, and Y. Sazeidès. Design tradeoffs for the ev8 branch predictor. In *Proceedings of the 29th Annual International Symposium on Computer Architecture*, 2002.

[8] André Seznec. The idealistic gtl predictor. *J. Instruction-Level Parallelism*, 9, 2007.

[9] André Seznec. The L-TAGE branch predictor. *Journal of Instruction Level Parallelism (http://wwwjilp.org/vol9)*, May 2007.

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

[11] André Seznec. A new case for the tage branch predictor. In *Proceedings of the MICR0 44*, 2011.

[12] André Seznec and Pierre Michaud. A case for (partially)-tagged geometric history length predictors. *Journal of Instruction Level Parallelism (http://www.jilp.org/vol8)*, April 2006.