Journal of Instruction-Level Parallelism 9 (2007) 1-11 Submitted 4/07; published 5/07

The Idealistic GTL Predictor

André Seznec SEZNEC@IRISA.FR
IRISA/INRIA/HIPEAC
Campus de Beaulieu, 35042 Rennes Cedex, France

Abstract

Over the few last years, substantial progress has been made on the accuracy of storage limited conditional branch predictors. In the past years, we proposed the geometric history length predictors, GEHL [10] and TAGE [13]. These predictors rely on several predictor tables indexed through independent functions of the global branch/path history and branch address. The set of used global history lengths forms a geometric series, i.e., $L(j) = \alpha^{j-1}L(1)$.

GEHL and TAGE differ through their final prediction computation functions. GEHL uses a tree adder as prediction computation function. TAGE uses (partial) tag match. For realistic storage budgets (e.g., 64Kbits or 256Kbits), TAGE was shown to be more accurate than GEHL.

In this study, we first look for the accuracy limits of the GEHL and TAGE predictors on the set of distributed traces for the 2nd Championship Branch Prediction (CBP-2). We point out that, for quasi-infinite storage budget, the TAGE predictor achieves a misprediction rate only 9% lower than a 256 Kbits TAGE predictor (3.049 misp/KI vs 3.352 misp/KI). On the other hand, a quasi-infinite storage budget instead of 256 Kbits allows to reduce the misprediction by about 23% (from 3.663 misp/KI to 2.842 misp/KI) on the GEHL predictor. For quasi-infinite storage budgets and very large number of predictor components, the GEHL predictor is more accurate than the TAGE predictor on most benchmarks.

However there exists branches for which the TAGE predictor is more accurate than the GEHL predictor. Moreover our study also indicates that on a few branchs, a loop predictor can capture part of the mispredictions that are not captured by GEHL or TAGE.

Therefore we propose the GTL predictor as an idealistic predictor at CBP-2. The GTL predictor combines a GEHL predictor, a TAGE predictor and a loop predictor. The GTL predictor won the idealistic track at CBP-2.

1. Introduction

Over the few last years, substantial progress has been made on the accuracy of storage limited conditional branch predictors [12, 3, 4, 10, 13].

In the two past years, we proposed the geometric history length predictors, GEHL [10] and TAGE [13]. To the best of our knowledge, these two predictors were the most storage effective branch predictors proposed before the 2nd Championship Branch Prediction. GEHL and TAGE differ through their final prediction computation functions. GEHL uses a tree adder as prediction computation function. TAGE uses (partial) tag match. These two predictors are among the most storage effective conditional branch predictors, TAGE being more efficient than GEHL for limited storage budgets. Both predictors rely on several predictor tables indexed through independent functions of the global branch/path history and branch address. The set of used global history lengths forms a geometric series, i.e.,

$L(j) = \alpha^{j-1}L(1)$. This allows to efficiently capture correlation on recent branch outcomes as well as on very old branches.

GEHL and TAGE are therefore natural candidates to serve as basis for the design of an idealistic branch predictor.

The first objective of this study was to explore the limits of accuracy that can be achieved by the GEHL predictor and the TAGE predictor if one uses a quasi-infinite storage budget. On the set of distibuted traces for CBP-2, the accuracy limit of the TAGE predictor is found to be quite disappointing: only a 9 % reduction in misprediction numbers was achieved compared with a 256 Kbits TAGE predictor. On the other hand, using a huge number of tables on the GEHL predictor is more effective since it allows to reduce the misprediction rate of 256 Kbits GEHL predictor by approximately 23 %.

The second objective of the study was to provide an idealistic limit branch predictor based on "conventional" branch predictors such as GEHL and TAGE. Through combining GEHL and TAGE, we were able to propose an idealistic predictor outperforming both of them. Adding a loop predictor was found to allow to slightly reduce the misprediction rate. Therefore GTL predictor is an hybrid predictor using a very large GEHL predictor as its main component. GEHL is combined with a TAGE predictor and a loop predictor.

The remainder of the paper is organized as follows. Section 3 briefly recalls the principles of the geometric history length predictors GEHL and TAGE. Section 4 presents the respective accuracy limits we found for quasi-unlimited storage budgets GEHL and TAGE predictors. Section 5 presents the structure of the GTL predictor and the configurations used at CBP-2. Section 6 presents the accuracy limits we found for the GTL predictor and its components. Section 7 concludes this study.

2. Experimental Framework

2.1. Set of Traces

The simulation results presented in this paper were obtained on the set of distributed traces for CBP-2. This set of traces was collected on the SPEC suite. In the table of simulation results, we refer to each trace by its SPEC number.

2.2. Information for Indexing the Branch Ppredictor

2.2.1. Path and Branch History

In order to avoid misprediction due to path aliasing, on the presented limit predictors, the predictor components are indexed using a hash function of the program counter, the global history combining branch direction and path (7 address bits). Non-conditional branches are included in these histories.

2.2.2. Discriminating Kernel and User Branches

Kernel and user codes appear in the traces. In practice in the traces, we were able to discrimate user code from kernal through the address range. In order to avoid history pollution by kernel code, on jumps in kernel code, we save the global branch history and path histories and restore them when jumping back in user code. Such use of two different

histories for kernel and user branches was also independently proposed in [7]. In practice, using two global histories resulted in a reduction of 2-3 % of the overall misprediction number.

3. The Geometric History Length Predictors

3.1. Common Features on TAGE and GEHL

The GEHL predictor [10] and the TAGE predictor [13] rely on several predictor tables indexed through independent functions of the global branch/path history and branch address. On both predictors, the set of used global history lengths forms a geometric series, i.e, $L(i) = (int)(\alpha^{i-1} * L(1) + 0.5)$; table T0 is indexed with the PC.

The use of a geometric series allows to use very long branch histories in the hundred of bits range for realistic size predictors (e.g., 64Kbits or 256Kbits). As pointed out in [10, 13], the exact shape of the series is not important. The important factor is to be able to catch correlation with very old branchs while still using most of the storage budget on short histories.

The TAGE predictor and the GEHL predictor essentially differ through their final prediction computation function.

3.2. The GEHL Branch Predictor [10]

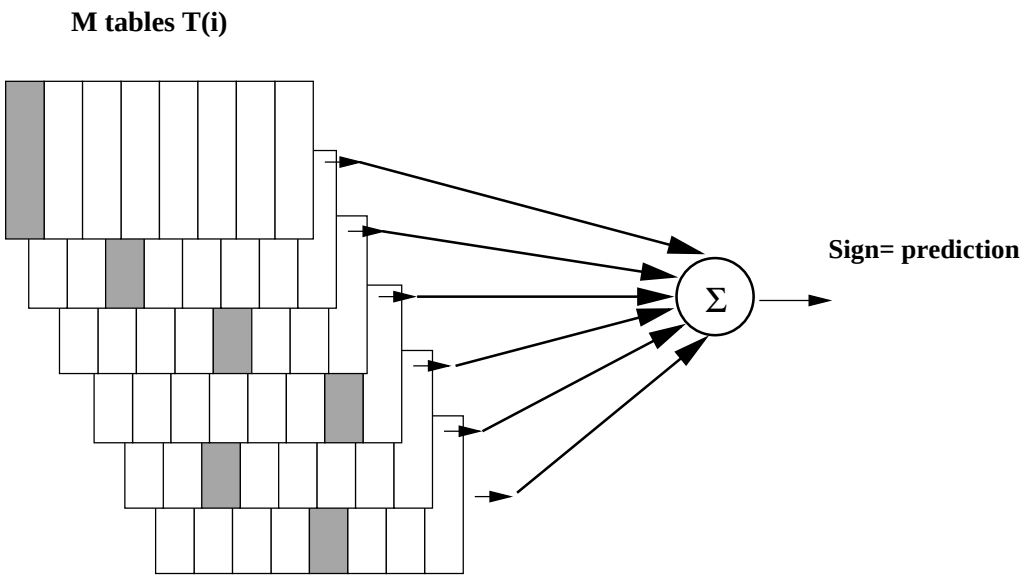

Figure 1: The GEHL predictor.

3.2.1. General Principle

The GEometric History Length (GEHL) branch predictor is illustrated on Figure 1. It features M distinct predictor tables Ti, $0 \leq i < M$ indexed with hash functions of the branch address and the global branch history. The predictor tables store predictions as

signed counters. To compute a prediction, a single counter C(i) is read on each predictor table Ti. The prediction is computed as the sign of the sum S of the M counters C(i), $S = \frac{M}{2} + \sum_{0 \leq i < M} C(i)$ [1]. The prediction is taken if S is positive and not-taken if S is negative.

Distinct history lengths are used for computing the index of the distinct tables. Table T0 is indexed using the branch address. The history lengths used in the indexing functions for tables Ti, $1 \leq i < M$ form a geometric series.

3.2.2. Updating the GEHL Predictor

The GEHL predictor update policy is derived from the perceptron predictor update policy [5]. The GEHL predictor is only updated on mispredictions or when the absolute value of the computed sum S is smaller than a threshold θ. Saturated arithmetic is used. More formally, the GEHL predictor is updated as follows, *Out* being the branch outcome:

if ((p != Out) or ($|S| \leq \theta$))
for each i in parallel
if Out then $C(i) = C(i) + 1$ else $C(i) = C(i) - 1$

3.2.3. Dynamic Threshold Fitting for the GEHL Predictor

Experiments showed that the optimal threshold θ for the GEHL predictor varies for the different applications. In [10], dynamic threshold fitting was introduced to accomodate this difficulty. For most benchmarks there is a strong correlation between the quality of a threshold θ and the relative ratio of the number of updates on mispredictions NU_{miss} and the number of updates on correct predictions $NU_{correct}$: experimentally, in most cases , for a given benchmark, when NU_{miss} and $NU_{correct}$ are in the same range, θ is among the best possible thresholds for the benchmark. This property was verified for the CBP-2 traces as well as for the CBP-1 traces.

A simple algorithm allows to adjust the update threshold while maintaining the ratio $\frac{NU_{miss}}{NU_{correct}}$ close to 1. This algorithm is based on a single saturated counter TC (for threshold counter).

if ((p != Out) {TC= TC + 1; if (θ is saturated positive){$\theta = \theta + 1$; TC=0;} }
if ((p == Out) & ($|S| \leq \theta$)) {TC= TC - 1; if (θ is saturated negative){$\theta = \theta - 1$;
TC=0;}}

Using a 7-bit counter for TC was found to be a good tradeoff.

3.3. Counter Width

For storage limited GEHL predictors, one must trade the counter width with the number of entries. For a quasi-infinite storage budget GEHL predictor, we found that using 7 to 9 bits counters results in the same accuracy level. 8-bit counters are used in this paper.

1. For p-bit signed counters, predictions vary between -2^{p-1} and $2^{p-1} - 1$ and are centered on $-\frac{1}{2}$

3.4. The TAGE Predictor [13]

The TAGE predictor [13] is derived from Michaud's PPM-like tag-based branch predictor [8]. The TAGE predictor is illustrated on Figure 2. 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.

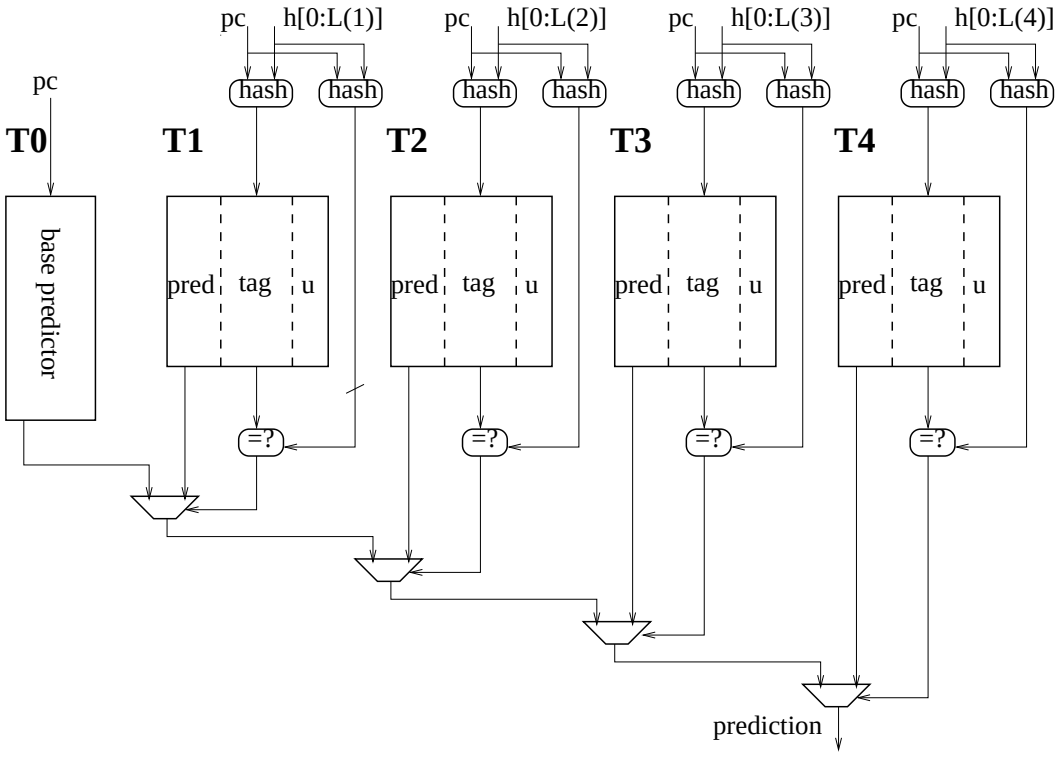

Figure 2: A 5-component TAGE predictor.

When the TAGE predictor is used as stand-alone the base predictor T0 is typically a bimodal PC indexed predictor [13].

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 this paper, u is a 2-bit counter and *ctr* is a 3-bit counter when TAGE is used as a standalone predictor and a 5-bit counter when TAGE is used as part of the GTL predictor.

3.4.1. Prediction Computation on TAGE

At prediction time, the base predictor and the tagged components are accessed simultaneously. The base predictor provides a default prediction. The tagged components provide a prediction only on a tag match.

In the general case, the overall prediction is provided by the hitting tagged predictor component that uses the longest history, or in case of no matching tagged predictor component, the default prediction is used.

However,on several applications, newly allocated entries provide poor prediction accuracy. We remarked that on several applications it is more efficient to use the alternate prediction rather than using the prediction of a newly allocated entry. This property was found to be essentially global to the application and can be dynamically monitored through a single 4-bit counter (*USE_ALT_ON_NA* in the simulator).

Moreover, no special memorization is needed for recording the "newly allocated entry": we consider that an entry is newly allocated if its prediction counter is weak (i.e. equal to 0 or -1). This approximation was found to provide results equivalent to effectively recording the "newly allocated entry" information.

Therefore the prediction computation algorithm is as follows:

1. Find the longest matching component

2. if (the prediction counter is not weak or USE_ALT_ON_NA is negative) then the prediction counter sign provides the prediction else the prediction is the alternate prediction.

3.4.2. Updating the TAGE Predictor

We present here the various scenarios of the update policy that we implement on the TAGE predictor.

Updating the Useful Counter u The useful counter u of the provider component is updated when the alternate prediction *altpred* is different from the final prediction *pred*.

Updating the Prediction Counters The prediction counter of the provider component is updated. For large predictors also updating the alternate prediction when the useful counter of the provider component is null results in a small accuracy benefit.

Allocating Tagged Entries on Mispredictions For a 64 Kbits 8-component predictor [13] or 256 Kbits TAGE predictors [11], allocating a single entry is the best tradeoff. For the very large predictor considered here, allocating all the available entries was found to be more effective [2]: this reduces the cold start mispredictions.

The allocation process of a new entry on a misprediction is described below.

The M-i u_j counters are read from predictor components Tj, $i < j \leq M$. Then we apply the following rules.

(A) Avoiding ping-pong phenomenon: in the presented predictor, the search for free entries begins on table Tb, with b=i+1 with probability 1/2, b=i+2, with probability 1/4 and b=i+3 with probability 1/4.

(B) For all components Tk, $k \geq b$, if $u_k = 0$ then the entry is allocated.

(C) Initializing the allocated entry: An allocated entry is initialized with the prediction counter set to weak correct. Counter u is initialized to 0 (i.e., *strong not useful*).

2. An entry is considered as free if its useful counter u is null.

4. Accuracy Limits of the Geometric History Length Predictors

In this section, we present the accuracies of the GEHL and TAGE predictors using quasi-infinite storage budgets. These accuracy results are compared with the respective accuracies of 256 Kbits TAGE and GEHL predictors.

The 13-components 256 Kbits TAGE configuration is described in [11] The 256 Kbits GEHL configuration features 12 4Kentries tables. Counters of tables T0 to T3 are 6-bit wide. Counters of tables T4 to T11 are 5-bit wide. Maximum history length is 500.

For TAGE, we illustrate an idealistic predictor with a total of 18 components and maximum history length of 2000. For GEHL, we illustrate an idealistic predictor with a total of 97 components and maximum history length of 2000.

Simulation results are displayed per application in Table 1. The idealistic TAGE predictor achieves 3.049 misp/KI against 3.352 misp/KI for a 256Kbits TAGE predictor. The idealistic GEHL predictor achieves 2.842 misp/KI against 3.663 misp/KI on a 256Kbits GEHL predictor.

The simulation results shows that the TAGE predictor is very storage effective: a 256K bits TAGE predictor is within 10 % of the misprediction rate of a quasi-infinite TAGE predictor. Through this analysis, we also discovered that increasing the number of components in TAGE over 15-20 leads to a decrease of accuracy.

A contrario, the idealistic GEHL predictor achieves better accuracy than the idealistic TAGE predictor while the 256Kbits GEHL predictor is only achieving accuracy in the range of the one of a 64K TAGE predictor. This is achieved through a very high number of components[3].

One can also note that on a few benchmarks the idealistic TAGE slightly outperforms the idealistic GEHL predictor. *This indicates that there exists some correlations that are captured by the TAGE predictor that are not captured by the GEHL predictor.*

	164	175	176	181	186	197	201	202	205	209
ideal GEHL	10.166	7.919	2.490	7.663	1.630	3.969	5.480	0.321	0.241	2.234
256K GEHL	10.298	9.250	4.503	10.405	2.595	6.046	6.200	0.467	0.525	2.362
ideal TAGE	10.749	8.467	2.630	7.846	2.056	4.342	5.787	0.321	0.254	2.372
256K TAGE	10.781	8.990	3.224	9.006	2.415	5.141	5.853	0.368	0.349	2.345
	213	222	227	228	252	253	254	255	256	300
ideal GEHL	1.043	1.035	0.291	0.485	0.179	0.183	1.257	0.108	0.044	10.113
256K GEHL	1.172	1.157	0.588	0.721	0.223	0.378	1.767	0.178	0.043	14.386
ideal TAGE	1.052	1.052	0.284	0.456	0.202	0.163	1.418	0.088	0.040	11.414
256K TAGE	1.119	1.114	0.397	0.590	0.218	0.325	1.547	0.141	0.041	13.269

Table 1: Per benchmark accuracy in misp/KI.

3. Using 257 components resulted in a marginal better accuracy, but simulation time was close to 5 hours.

5. The GTL Predictor

In this section, we present the idealistic predictor GTL predictor. The GTL predictor is illustrated on Figure 3. The GTL predictor features a GEHL predictor [10], a TAGE predictor [13] and a loop predictor as components.

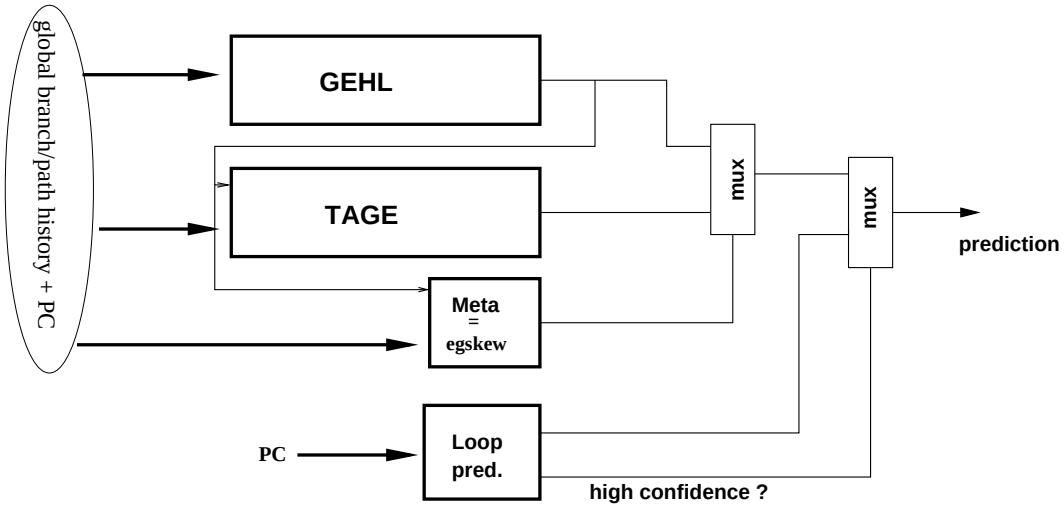

Figure 3: The GTL predictor.

The GEHL predictor is the main predictor, since it is the most accurate component as was shown in the previous section. The GEHL predictor is used as the base predictor component in TAGE, i.e. when there is no partial hit on any of the TAGE components, the TAGE prediction is the GEHL prediction. Moreover, the GEHL prediction is used as part of the indices for the tagged components of the TAGE predictor as well as for the components of the metapredictor. Using a (partial) prediction as part of the index of another table was already used on the YAGS predictor [1] and the bimode predictor [6].

GTL uses a metapredictor to discriminate between the TAGE and the GEHL predictions. We use a metapredictor derived from the skewed predictor [9].

Finally, the prediction features a loop predictor. The loop predictor provides the prediction when a loop has been successively executed with 8 times the same number of iterations as in [2].

5.1. The Loop Predictor

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

As in [2], the loop predictor provides the global prediction when the loop has succesively been executed 8 times with the same number of iterations. The loop predictor used in the submission features 512K entries.

5.2. GEHL Configuration in the GTL Predictor

We leveraged the different degrees of freedom in the design of the GEHL predictor to get the best predictor that we could simulate with a memory footprint in the range of 768 Mbytes.

Experiments showed that, for a GEHL simulator with a memory footprint in the range of 768 Mbytes, using 97 tables provides a high level of accuracy [4] Experiments showed that using 8-bit counters leads to good prediction accuracy.

We also slightly improve the update policy by incrementing/decrementing twice the counters when they are between -8 and 7. This results in a gain of 0.009 misp/KI on the overall GTL predictor.

5.3. TAGE Configuration in the GTL Predictor

We leveraged the different degrees of freedom in the design of the TAGE predictor to get the best predictor that we could simulate while maintaining the total memory footprint of the simulator smaller than 1 gigabyte.

The TAGE component in the submitted predictor feature a total of 19 tagged components, the GEHL predictor is used as the base predictor. Each table feature 1M entries. The tag width is 16 bits on the tagged tables.

5.4. Selecting between TAGE and GEHL Predictions

As a meta-predictor to discriminate between TAGE and GEHL predictors, we found that a skewed predictor [9] works slightly better than a bimodal table (by 0.004 misp/KI). The respective history lengths of the three tables are 0, 4 and 22. As for the TAGE predictor component, the output of the GEHL predictor is used to index the meta-predictor.

For one benchmark, the GEHL+TAGE predictor exhibited a slightly higher misprediction rate than the GEHL predictor alone. Therefore to avoid this situation, we use a single safety counter that monitors that GEHL+TAGE against GEHL alone.

5.4.1. Indexing the TAGE Predictor Component and the Metapredictor

The TAGE predictor is used to try to correct predictions of the GEHL predictor. Therefore the output of the GEHL predictor is used as part of the indices of the TAGE predictor components and the metapredictor.

5.4.2. History Lengths

Geometric length allows to use very long history lengths. Experiments illustrated in the previous section showed that using 2,000 as the maximum history length for both TAGE and GEHL predictors is a good choice, but that using respectively 400 for GEHL and 100,000 for TAGE is marginally better (by 0.014 misp/KI)[5].

For the GEHL predictor, we force the use of distinct history lengths by enforcing the property $L(i) > L(i+1) + 1$.

5.5. Static Prediction

On the first occurence of a branch, a static prediction associated with the branch opcode is used: this allows to reduce the overall misprediction rate by 0.005 misp/KI.

4. 257 would have been a marginally better choice but the simulation time was too long for CBP2 execution time constraints.
5. 100,000 is 0.002 misp/KI better than 10,000

6. Simulation Results

The average predictor accuracy of GTL is **2.717 misp/KI** on the distributed set of traces. Removing the loop predictor and the static prediction, i.e., GEHL+TAGE, one will still obtain 2.774 misp/KI, the accuracy of the GEHL predictor component alone being 2.891 misp/KI. As already mentioned, a GEHL predictor alone using a 2,000 branch history length achieves 2.842 misp/KI.

Results for the GTL and GEHL+TAGE (row G+T) are displayed per application in Table 2. Results for GEHL predictor using a 2,000 branch history length are also displayed as a reference. One can note that the benefit of loop prediction is marginal apart on *164.gzip* and to a less extent on *201.compress*.

	164	175	176	181	186	197	201	202	205	209
GTL	9.393	7.783	2.434	7.312	1.591	3.891	5.260	0.293	0.234	2.168
G+T	9.992	7.785	2.472	7.350	1.602	3.901	5.397	0.302	0.247	2.175
GEHL	10.166	7.919	2.490	7.663	1.630	3.969	5.480	0.321	0.241	2.234
	213	222	227	228	252	253	254	255	256	300
GTL	0.946	0.943	0.271	0.425	0.177	0.128	1.129	0.087	0.033	9.858
G+T	0.991	0.990	0.287	0.442	0.180	0.137	1.211	0.093	0.041	9.890
GEHL	1.043	1.035	0.291	0.485	0.179	0.183	1.257	0.108	0.044	10.113

Table 2: Per benchmark accuracy in misp/KI.

7. Conclusion

The geometric history length predictors TAGE and GEHL represent the current state-of-the-art of hardware implementable predictors.

In this study, we found that, at a a very large storage budget and for very large number of components, the GEHL predictor outperforms the TAGE predictor. The accuracy of TAGE reaches a plateau with medium number of components (in the range of 15-20) and medium storage budget (around 64 Mbytes): a 256 Kbits 13-component TAGE predictor exhibits only approximatelly 9% more mispredictions than the best no-storage limit TAGE predictor we found. The GEHL predictor accuracy improves with the number of components and using around 100 components for a limit GEHL predictor is effective.

We found that combining the GEHL predictor and the TAGE predictor allows to grab (part of) the last pieces of predictability contained in global branch/path history. A loop predictor can improve a little bit the prediction accuracy of the GEHL+TAGE predictor. It can be remarked that, on the set of benchmarks for CBP2, the loop predictor has only small return apart on 164.gzip. On the set of benchmarks for CBP-1, this return was found to be even smaller. Apart the loop predictor, and despites our best efforts, we have not found any way to integrate a local history predictor component bringing any benefit to the GTL predictor.

We hope that the GTL predictor will serve as a predictor reference to a new generation of researches to test and validate new branch prediction algorithms.

Acknowledgements

This work was partially supported by an Intel research grant, an Intel research equipment donation and by the European Commission in the context of the SARC integrated project #27648 (FP6).

References

[1] A. N. Eden and T.N. Mudge. The YAGS branch predictor. In *Proceedings of the 31st Annual International Symposium on Microarchitecture*, Dec 1998.

[2] H. Gao and H. Zhou. Adaptive information processing: An effective way to improve perceptron predictors. *Journal of Instruction Level Parallelism (http://www.jilp.org/vol7)*, April 2005.

[3] D. Jiménez. Fast path-based neural branch prediction. In *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, dec 2003.

[4] D. Jiménez. Piecewise linear branch prediction. In *Proceedings of the 32nd Annual International Symposium on Computer Architecture*, june 2005.

[5] D. Jiménez and C. Lin. Dynamic branch prediction with perceptrons. In *Proceedings of the Seventh International Symposium on High Perform ance Computer Architecture*, 2001.

[6] C-C. Lee, I-C.K. Chen, and T.N. Mudge. The bi-mode branch predictor. In *Proceedings of the 30th Annual International Symposium on Microarchitecture*, Dec 1997.

[7] T. Li, L.K. John, A. Sivasubramaniam, N. Vijaykrishnan, and J. Rubio. Os-aware branch prediction: Improving microprocessor control flow prediction for operating systems. *IEEE Trans. Computers*, 56(1):2–17, 2007.

[8] P. Michaud. A ppm-like, tag-based predictor. *Journal of Instruction Level Parallelism (http://www.jilp.org/vol7)*, April 2005.

[9] P. Michaud, A. Seznec, and R. Uhlig. Trading conflict and capacity aliasing in conditional branch predictors. In *Proceedings of the 24th Annual International Symposium on Computer Architecture (ISCA-97)*, June 1997.

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

[11] A. Seznec. The cbp-2 l-tage branch predictor. *Journal of Instruction-Level Parallelism, vol. 9*, May 2007.

[12] 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.

[13] A. Seznec and P. Michaud. A case for (partially)-tagged geometric history length predictors. *Journal of Instruction Level Parallelism (http://www.jilp.org/vol7)*, April 2006.