Journal of Instruction-Level Parallelism 7 (2005) 1-12 Submitted 01/05; published 04/05

Genesis of the O-GEHL branch predictor *

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

Abstract

We detail the genesis of the Optimized GEometric History Length (O-GEHL) branch predictor that was presented at the First Championship Branch Prediction held at Portland in December 2004. The OGEHL predictor efficiently exploits very long global histories in the 100-200 bits range.

The O-GEHL predictor features several predictor tables $T(i)$ (8 for the version presented at CBP) indexed through independent functions of the global branch history and branch address. The prediction is computed through the addition of the predictions read on the predictor tables as on perceptron predictors. GEHL stands for GEometric History Length since the set of used global history lengths forms a geometric series, i.e., $L(j) = \alpha^{j-1}L(1)$. This allows to use the most significant part of the storage budget to capture correlation on recent branch outcomes while still being able to capture correlation on very old branches.

The O-GEHL predictor features dynamic history length fitting and dynamic threshold fitting. These mechanisms improve the ability of the predictor to exploit very long histories.

1 Introduction

Modern processors feature moderate issue width, but deep pipelines. Therefore, any improvement in branch prediction accuracy directly translates in performance gain. In this paper, we present the different steps that lead to the proposal of the O-GEHL predictor that was presented at the 1st Championship Branch Predictor (CBP) [1].

First we recall the rules of the CBP (http://http://www.jilp.org/cbp) and analyze the characteristics of the set of distributed traces. Then we present and comment the three general guidelines that we respected in addition to the rules of CBP, i.e., 1) only global branch or path history information should be used as input to the predictor. 2) the predictor should be close to a possible implementation, 3) the predictor accuracy should be as independent as possible from its initialization state.

Then we present the general view of the (O-)GEHL predictor principles. The GEHL predictor uses multiple predictor tables indexed with distinct history lengths (Figure 1). Each table provides a prediction as a signed counter. As on the perceptron predictor [2, 3], the overall prediction is computed through an adder tree. A threshold-based partial update policy is also borrowed from the perceptron-like predictors. GEHL stands for GEometric History Length since we are using an approximation of a **geometric series** for the history lengths L(i), i.e., $L(i) = \alpha^{i-1}L(1)$. A performance evaluation of a first naive 64 Kbits predictor is then provided.

*. This work was partially supported by an Intel research grant and an Intel research equipment donation

Then we describe and evaluate one by one step that lead to the final O-GEHL predictor proposal. 1) We show that 4-bit counters are cost-effective for the O-GEHL predictor. At constant storage budget, using 4-bit counters allows to use twice as many entries in the predictor tables as using 8-bit counters as on perceptron-like predictors. Using a mix of 4-bit and 5-bit counters is even more cost-effective. 2) The update threshold used for perceptron-like predictors [2] is not adapted to the O-GEHL predictor. We describe and evaluate a dynamic threshold fitting algorithm. 3) The O-GEHL predictor implements a simple form of dynamic history length fitting [4] to adapt the behavior of the predictor to each application. In practice, for demanding applications, most of the storage space in the O-GEHL predictor is devoted to tables indexed with short history lengths. But, the combination of geometric history length and of dynamic history lengths fitting allows the O-GEHL predictor to exploit very long global histories (typically in the 200-bits range) on less demanding applications. 4) We show that some accuracy benefit can be further grabbed through using a more precise representation of the program path than only using conditional branch history.

Section 7 briefly reviews the related works that had major influences in the O-GEHL predictor proposal and concludes the paper.

2 Championship Branch Prediction

2.1 The CBP rule

The following unique rule was given for the contest (see http://www.jilp.org/cbp/):

> "Quantitatively assessing the cost/complexity of predictors is difficult. To simplify the review process, maximize transparency, and minimize the role of subjectivity in selecting a champion, CBP will make no attempt to assess the cost/complexity of predictor algorithms. Instead contestants will be given a storage budget of (64K + 256) bits. All predictors must be implemented within the constraints of this budget. And clear documentation must be provided to assure that this is the case."

The evaluation metric used for the CBP contest is misprediction/KI on a set of distributed traces.

Since the contest is performed using traces, immediate update of the predictor is assumed. On a real hardware processor, the effective update is performed later in the pipeline, at misprediction resolution or at commit time. However, for branch predictors using a very long global branch history as the O-GEHL predictor, the difference of accuracy between a delayed updated predictor and an immediately updated predictor is known to be small [5, 6].

2.2 The CBP traces

20 traces selected from 4 different classes of workloads were provided. The 4 workload classes are: server, multi-media, specint, specfp. Each of the branch traces is derived from an instruction trace consisting of 30 million instructions. These traces include system activity.

	FP-1	FP-2	FP-3	FP-4	FP-5	INT-1	INT-2	INT-3	INT-4	INT-5
static	444	452	810	556	243	424	1585	989	681	441
dyn. (x10000)	221	179	155	90	242	419	287	377	207	376
	MM-1	MM-2	MM-3	MM-4	MM-5	SERV-1	SERV-2	SERV-3	SERV-4	SERV-5
static	460	2523	1091	2256	4536	10910	10560	16604	16890	13017
dyn. (x10000)	223	381	302	488	256	366	354	381	427	429

Table 1: Characteristics of the CBP traces

30 million instruction traces were considered short by most of the competitors. However 30 million instructions represent approximately the workload that is executed by a PC under Linux or Windows in 10 ms, i.e., the OS time slice. Moreover, system activity was shown to have an important impact on predictor accuracy [7]. Finally, some traces, particularly server traces, exhibit very large number static branches that are not represented in more conventional workloads such as specint workloads. The characteristics of the traces are summarized in Table 1.

3 Personal guidelines for the O-GEHL predictor design

The computer architecture research community has to propose designs that are close to possible hardware implementation. Our O-GEHL predictor proposal for CBP is respecting a few "best practice" rules [1] that might allow its effective hardware implementation.

3.1 Using only global history or path information

Both using local history and global history have been considered to provide branch prediction [8]. Hybrid predictors using both local and global histories have even been implemented in Alpha EV6 [9].

However, on processors featuring deep pipelines, several branches are speculatively predicted in advance. For accurately predicting branches, one has to manage speculative history. Maintaining and using accurate speculative local history may be very difficult since 1) there may be several occurrences of a single branch inflight at the same time, 2) the response time of predictor may be longer than a single cycle and therefore accurate local history is not even known at the beginning of predictor computation time. On the other hand, maintaining accurate global history or path information is easy through the use of a circular buffer. Moreover a predictor relying only on global history or path information can be ahead pipelined to provide branch prediction in-time [10].

One of our challenges was to show that using only global history on a branch predictor is sufficient to deliver state-of-the-art accuracy, thus relieving the designer from the burden of maintaining accurate speculative local history.

3.2 Implementable design

The rule of the CBP challenge did not put restrictions on the complexity of the logical design of the predictor. Therefore the rule allowed the use of strange sized tables [11, 12, 13],

1. The O-GEHL predictor received the best practice award at CBP.

the use of very complex index computations [11] and/or the use of complex overall prediction computation [11, 12, 13, 14]. However converting these proposals in realistic hardware predictors might require substantial amount of extra work and/or extra storage budgets.

In our proposal, we only use power of two's as numbers of entries in storage tables. Maintaining the logic complexity of the final prediction computation was also one of our goals: the computation of the prediction is performed through a tree adder, but the width of the counters is only 4 or 5 bits, and only 8 counters have to be summed, therefore the prediction computation time remains limited.

On predictors using very long histories, the computation of the indexing functions may also be an issue for the access time to the predictor. Therefore, we demonstrated that simple indexing functions can be efficient. These index functions are described in Section 4.4.

3.3 "Independency" from initialization point of the predictor

In the most branch prediction studies, a "cold" branch predictor is considered at the beginning of the simulation. All counters are therefore initialized to zero. However, branch predictor behaviors might be (very) sensitive to the initial state of the predictor.

The CBP framework was stating that simulations were successively run independently. Therefore the predictors presented at CBP featured the same initial state for all the trace simulations. For the three static predictor configurations presented at CBP [13, 15, 14], using an initial state with essentially setting counters to zero (or weakly taken) was the natural choice. Other initialization choices (e.g. random initialization) lead to slightly higher misprediction rate on these predictors, but remain in the same accuracy range.

The three remaining predictors [1, 11, 12] including the O-GEHL predictor automatically modify their configurations (indexing functions, history length, etc) depending on statistics monitored during simulation. Therefore their behavior may be very sensitive to the initial state of the predictor (configuration, counter values, ...).

Experiments with random initialization of the predictor configuration and storage counters for the predictors in [11, 12] may lead to poor global accuracy. For both predictors, the algorithm for self-configuration were designed assuming a precise initialization state of the predictor, but transition to an alternate configuration (on a context switch or on a program phase change) was not considered.

At all steps during this study, we checked that the O-GEHL predictor behavior is not dramatically impaired by the initialization it may inherit from a previous application (for instance after a context switch) through the following protocol: the simulations of the 20 traces are chained without resetting the predictor counters. Compared with assuming resetting all the counters before simulating a new trace, the discrepancy in prediction accuracy was always marginal (0.03 misp/KI for the final CBP 64Kbits O-GEHL predictor).

4 General principles of GEometric History Length branch prediction

4.1 Rationale

Since the first studies on branch correlation in 1991 [16, 8], it is well known that branch outcomes are strongly correlated with the outcomes of previous branches. The use of multiple predictor components to generate a single prediction has been proposed in 1993 by

McFarling [17]. More recently, neural inspired branch predictors were introduced [2], providing a storage effective "meta"-prediction: meta-prediction is replaced by computation.

The GEometric History Length branch prediction approach leverages this previous knowledge, and tries to exploit the following phenomenom: most branch correlation occurs with recent branches, but some correlation occurs with very old branches.

4.2 General principle

The GEometric History Length (GEHL) branch predictor is illustrated on Figure 1. The GEHL predictor 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), $S = \frac{M}{2} + \sum_{0 \leq i < M} C(i)$ [2]. 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, $1 \leq i < M$ are of the form $L(i) = \alpha^{i-1} * L(1)$, i.e., the lengths L(i) form a **geometric series**.

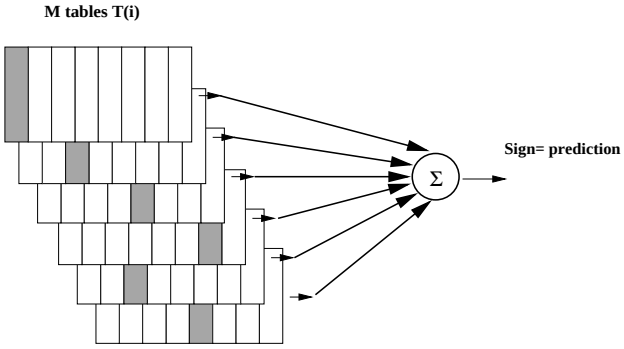

Figure 1: The GEHL predictor

In this article, we only focus on 8-table GEHL predictors as this is the most storage effective choice for a 64 Kbits storage budget. As an example,$\alpha = 2$ and L(1) =2 leads to the following series {0,2,4,8,16,32,64,128}. Remark that 5 tables are indexed using 16 history bits or less while correlation on a 128-bit history might be captured.

4.3 Updating the GEHL predictor

The GEHL predictor update policy is derived from the perceptron predictor update policy [2]. 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 $((p \mathrel{!}= Out)$ or $(|S| \leq \theta))$
for each i in parallel
if *Out* then $C(i) = C(i) + 1$ else $C(i) = C(i) - 1$

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

4.4 Cost effective index functions

The O-GEHL predictor has been defined in order to capture correlation on very long histories in the 100-200 bits range. In Section 6, we will consider that one of the O-GEHL predictor tables is indexed using a 200-bit history, plus a 16-bit path information plus a 32-bit branch address. Fully hashing these 248 bits to compute a 11-bit index would normally require using 23 bit entry functions for computing each index bit (for instance a 23-entry exclusive-OR tree). The delay for computing such functions can be seen as prohibitive.

The index functions used for the simulations presented in this paper can be computed using single three-entry exclusive-OR gate for computing each index bit. We choose to ignore some of the address bits and some of the history bits as follows. For computing the hash function to index Table Ti, 2^n being the number of entries on Ti, we regularly pick 3n bits in the vector of bits composed with the least significant bits of the branch address, the L(i) branch history bits and the min(L(i),16) path history bits. Then we simply hash this 3n bit vector in a n-bit index through a single stage of 3-entry exclusive-OR gates.

Experiments showed very limited accuracy degradation when using these simple hash functions instead of full hash of the branch address, the branch history and path history. Using a single stage of 2-entry exclusive-OR gates (i.e. picking only 2n bits) would result in 2-3 % global increase of the number of mispredictions on a 64Kbits predictor while using no exclusive-OR (i.e., picking n bits and directly using it as an index) would result in 10-11 % average increase of the number of mispredictions.

All experiments illustrated in this paper were realized using these simple indexing functions. See function INDEX in the companion code implementation of the predictor.

4.5 A first performance evaluation

The perceptron predictor study [2] was using 8-bit counters and was using threshold θ= 14+1.93*N, N being the number of weights. Therefore using 8 1-Kentries 8-bit tables and 29 as threshold was a natural choice. The accuracy of this first (naive) GEHL predictor for L(1)= 3 and L(7)= 80 is illustrated in Table 2. While it outperforms both the Path Based Neural Predictor [18] and *2bcgskew* [6] that were often previously considered as state-of-the-art predictors, its average accuracy (3.50 misp/KI) would not have allowed it to qualify as a finalist at CBP.

FP-1	FP-2	FP-3	FP-4	FP-5	INT-1	INT-2	INT-3	INT-4	INT-5
1.650	0.966	0.433	0.106	0.184	1.972	7.159	8.621	1.330	0.339
MM-1	MM-2	MM-3	MM-4	MM-5	SERV-1	SERV-2	SERV-3	SERV-4	SERV-5
7.353	9.201	0.395	1.438	5.643	3.774	3.975	6.019	5.158	4.331

Table 2: Accuracy (misp/KI) of an initial (naive) 64Kbits GEHL predictor: 8-bit counters, $\theta = 29$

5 Step by step to the O-GEHL predictor

In this section, we present the various optimizations we made to derive the O-GEHL predictor presented at CBP from the naive GEHL predictor presented in the previous section.

5.1 Using 4-bit counters

On the perceptron predictor family, using 8-bit counters to represent the weights was shown to be a good tradeoff. However, such a width is a waste of resource on the GEHL predictor. Using 4-bit counters allows to use twice as many entries per predictor table for the same storage budget. However using the initial perceptron update threshold would lead to very poor accuracy. We heuristically found that using the number of tables (i.e., 8) as the update threshold is a good tradeoff.

Table 3 illustrates the accuracy benefit of using 4-bit counters (L(1)=3 and L(7)=80). Average misprediction rate is 3.11 misp/KI. The benefit of halving the counter width, but using twice as many counters is particularly important on the server workloads which exhibit very large footprints.

FP-1	FP-2	FP-3	FP-4	FP-5	INT-1	INT-2	INT-3	INT-4	INT-5
1.627	0.909	0.435	0.104	0.182	1.893	6.303	9.196	1.073	0.348
MM-1	MM-2	MM-3	MM-4	MM-5	SERV-1	SERV-2	SERV-3	SERV-4	SERV-5
7.773	9.532	0.674	1.336	4.866	2.224	2.269	4.565	3.824	3.203

Table 3: Accuracy (misp/KI) of a 64Kbits GEHL predictor: 4-bit counters, $\theta = 8$

5.2 Dynamic threshold fitting for the GEHL predictor

Experiments showed that the optimal threshold θ for the GEHL predictor varies for the different applications. For some of the benchmarks and using a 8-table GEHL predictor, the difference between using $\theta = 5$ or $\theta = 14$ as a threshold results in 0.5 misp/KI variations. However, we remarked that 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.

Therefore, we implement a simple algorithm that adjusts 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 (TC is saturated positive){$\theta = \theta + 1$; TC=0;} }
if ((p == Out) & ($|S| \leq \theta$)) {TC= TC - 1; if (TC is saturated negative){$\theta = \theta - 1$; TC=0;}}

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

Table 4 illustrates the misprediction rate when using the threshold fitting algorithm (L(1)=3 and L(7)=80). Average misprediction rate is 3.05 misp/KI. The benefits of this

dynamic threshold fitting for smaller or larger predictors and for different counter widths are further analyzed in [19].

FP-1	FP-2	FP-3	FP-4	FP-5	INT-1	INT-2	INT-3	INT-4	INT-5
1.599	0.919	0.435	0.101	0.304	1.445	6.250	9.328	1.032	0.348
MM-1	MM-2	MM-3	MM-4	MM-5	SERV-1	SERV-2	SERV-3	SERV-4	SERV-5
7.691	9.427	0.649	1.348	4.840	2.041	2.029	4.532	3.661	3.058

Table 4: Accuracy (misp/KI) of a 64Kbits GEHL predictor: 4-bit counters, dynamic threshold fitting

5.3 Dynamic history length fitting for the (O)-GEHL predictor

Juan et al [4] proposed to continuously adapt the branch history length during execution for global history branch predictors. The (O)-GEHL predictor offers an opportunity to implement such an adaptative history length fitting. We consider a predictor featuring 8 tables, but using 11 history lengths L(j) forming a **geometric** series. For three predictor tables, Tables T2, T4 and T6, two possible history lengths are used: Table T2 is indexed using either L(2) or L(8), Table T4 is indexed using either L(4) or L(9), Table T6 is indexed using either L(6) or L(10).

The algorithm we propose to select the history length for indexing the predictor makes a rough estimation of the aliasing ratio encountered on Table T7, i.e., the predictor component using the longer history apart L(8), L(9) and L(10). Intuitively, if Table T7 experiences a high degree of aliasing then short histories should be used on Tables 2, 4 and 6, if Table T7 encounters a low degree of aliasing then long histories should be used.

To compute this estimation of the aliasing ratio, we add a tag bit to some entries of Table T7 and we use a single saturating 9-bit counter AC (for aliasing counter). On a predictor update, the tag bit records one bit of the address of the branch and the following computation is performed:

```
if ((p!=out) & (|S| ≤ θ)){
if ((PC & 1) == Tag[indexT[7]]) AC++; else AC= AC - 4;
if (AC == SaturatedPositive) Use Long histories
if (AC == SatutedNegative) Use Short Histories
Tag[indexT[7]] = (PC & 1);}
```

When the last update of the corresponding entry in Table T7 has been been performed using the same (branch, history) pair, AC is incremented. When the last update has been performed by another (branch, history) pair, AC is incremented on false hits and decremented by 4 on misses.

In average, AC will stay positive if the ratio of conflicting updates on Table T7 by distinct branches remains below 40 %.

Using a 9-bit counter and flipping from short to long histories and vice-versa only on saturated values guarantees that such flippings are very rare.

Remark Associating a tag bit per entry in predictor table T7 is not needed. For instance one can associated only a tag bit to one out of N entries and ignore the other entries in the algorithm to update the AC counter. **For the CBP challenge predictor, we use only 1 K tag bits for a 2 Kentries Table T7.**

Fitting in a 64 Kbits storage budget The dynamic history length fitting presented above requires extra storage space in addition of the predictor tables. A O-GEHL predictor featuring 8 tables would normally lead to 8 2K 4-bit counters tables and a 1K 1-bit tag table associated with Table T7, i.e. a total of 65 Kbits. For fitting in the 64Kbits storage budget of CBP, Table T1 uses only 1K counters, thus reducing the storage budget to 61 Kbits. Experiments showed that using 5-bit counters on the tables using short history is slightly beneficial (tables T0 and T1). Therefore while respecting the storage budget constraints, the predictor submitted to the CBP mixes 5-bit counters and 4-bit counters.

Impact of adaptative history length fitting On the CBP benchmarks, using adaptative history length fitting on the O-GEHL predictor reduces by more than 5 % in average the number of mispredictions (i.e. 2.89 misp/KI) instead of using a single history length per predictor table as illustrated in Table 5 for L(1)=3 and L(10)=190.

FP-1	FP-2	FP-3	FP-4	FP-5	INT-1	INT-2	INT-3	INT-4	INT-5
1.411	0.904	0.356	0.111	0.071	0.678	5.854	8.968	1.007	0.336
MM-1	MM-2	MM-3	MM-4	MM-5	SERV-1	SERV-2	SERV-3	SERV-4	SERV-5
7.495	9.154	0.140	1.361	4.637	2.088	2.065	4.542	3.632	3.033

Table 5: 4-bit counters, dynamic threshold fitting, dynamic history length fitting

5.4 Last optimization: Information for indexing a global history branch predictor

For computing the indexes for global history predictors, most studies consider either hashing conditional branch history with the branch address or hashing path history with the branch address. These solutions cause different paths to appear as equal even before computing the effective index in the predictor tables. The impact of this phenomenon on predictor accuracy is important when using short history. On the GEHL predictor, it impairs the accuracy of the predictions provided by the tables indexed with short histories.

In order to limit this phenomenon on the O-GEHL predictor, we include the non-conditional branches in the branch history *ghist* (inserting a taken bit) and we also use a path history, *phist* consisting of 1 address bit per branch. Since confusion on paths decreases when the history length increases, we use a maximum path history length of 16 on the O-GEHL predictor submitted to CBP.

As illustrated in Table 7, this last optimization brings a 2-3 % decrease in misprediction numbers, but this decrease is not uniform among applications.

6 Performances of the CBP O-GEHL predictor

The characteristics of the submitted O-GEHL predictor are summarized in Table 6. Using 200 as L(10) the maximum history length and 3 as L(1) is one of the best tradeoffs

Table	T0	T1	T2	T3	T4	T5	T6	T7
short history	L(0)=0	L(1)=3	L(2)=5	L(3)=8	L(4)=12	L(5)=19	L(6)=31	L(7)=49
long history	-	-	L(8)=79	-	L(9)=125	-	L(10)=200	-
counter width)	5	5	4	4	4	4	4	4
tag bit	-	-	-	-	-	-	-	0.5
entries	2K	1K	2K	2K	2K	2K	2K	2K
budget (bits)	10K	5K	8K	8K	8K	8K	8K	8K +1K

Table 6: Characteristics of the O-GEHL predictor submitted to the CBP: a total of 64Kbits

on the set of the benchmark traces. However using any value in the interval 125-300 for L(10) and any value in the interval 2-6 for L(1) brings very similar simulation results, i.e. the total number of mispredictions for these pairs of values are not exceeding the presented results by more than 4%.

The simulation results obtained with the O-GEHL predictor are summarized in Table 7.

FP-1	FP-2	FP-3	FP-4	FP-5	INT-1	INT-2	INT-3	INT-4	INT-5
1.408	0.906	0.413	0.181	0.041	0.694	5.519	8.998	0.940	0.343
MM-1	MM-2	MM-3	MM-4	MM-5	SERV-1	SERV-2	SERV-3	SERV-4	SERV-5
7.218	9.019	0.229	1.358	4.427	1.999	1.912	4.422	3.407	2.956

Table 7: Accuracy of the 64Kbits O-GEHL predictor (misp/KI)

7 Related works and conclusion

The use of multiple global history lengths in a single branch predictor was initially introduced in [17], then it was refined by Evers et al. [20] and further appeared in many proposals. By using several short history components, the O-GEHL predictor suffers from very limited aliasing impact on short histories.

As neural inspired predictors [21, 2], the O-GEHL predictor does not use storage based metapredictors, but computes the prediction through an adder tree. This adder tree does not "waste" storage space for meta prediction. As the perceptron predictor, the O-GEHL predictor also uses a specific partial update policy considering a threshold. We improved this update policy through proposing dynamic threshold fitting.

The O-GEHL predictor implements dynamic history length fitting [4] and can adapt its behavior to each application.

The main contribution of the O-GEHL predictor over previously proposed predictors is its ability to efficiently exploit very long history. Due to the use of **geometric** history lengths associated with dynamic history length fitting, the O-GEHL predictor is able to achieve high accuracy on a wide range of applications. For some applications, the number of different (branch, history) pairs explodes when the history length increases. On these applications, the O-GEHL predictor achieves high accuracy because 5 out of 8 predictor tables are using history lengths shorter than 12. On other applications, correlation exists

with very old branches. The O-GEHL predictor is able to capture this correlation since some of its tables are indexed using history lengths in the 100-200 bits range.

This article only focuses on the design of a 64Kbits O-GEHL predictor. However the design space of cost-effective O-GEHL predictors is very large. For instance, depending on implementation tradeoffs, one can use from 4 to 12 tables, one can use 4-bit,5-bit and even 3-bit counters. High level of accuracy can be obtained for a broad spectrum of maximum history lengths, for instance any length in the 125-300 range for a 64 Kbits O-GEHL predictor. These tradeoffs are further analyzed in [19].

References

[1] A. Seznec, "The o-gehl branch predictor," in *The 1st JILP Championship Branch Prediction Competition (CBP-1)*, 2004.

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

[3] D. Jiménez and C. Lin, "Neural methods for dynamic branch prediction," *ACM Transactions on Computer Systems*, November 2002.

[4] T. Juan, S. Sanjeevan, and J. J. Navarro, "A third level of adaptivity for branch prediction," in *Proceedings of the 25th Annual International Symposium on Computer Architecture*, June 30 1998.

[5] D. Jiménez, "Reconsidering complex branch predictors," in *Proceedings of the 9th International Symposium on High Perform ance Computer Architecture*, 2003.

[6] 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.

[7] N. Gloy, C. Young, J. B. Chen, and M. D. Smith, "An analysis of dynamic branch prediction schemes on system workloads," in *ISCA '96: Proceedings of the 23rd annual international symposium on Computer architecture*, pp. 12–21, ACM Press, 1996.

[8] T.-Y. Yeh and Y. Patt, "Two-level adaptive branch prediction," in *Proceedings of the 24th International Symposium on Microarchitecture*, Nov. 1991.

[9] R. Kessler, "The Alpha 21264 microprocessor," *IEEE Micro*, vol. 19, no. 2, pp. 24–36, 1999.

[10] A. Seznec and A. Fraboulet, "Effective ahead pipelining of the instruction addres generator," in *Proceedings of the 30th Annual International Symposium on Computer Architecture*, June 2003.

[11] D. A. Jiménez, "Idealized piecewise linear branch prediction," in *The 1st JILP Championship Branch Prediction Competition (CBP-1)*, 2004.

[12] H. Gao and H. Zhou, "Adaptive information processing: An effective way to improve perceptron predictors," in *The 1st JILP Championship Branch Prediction Competition (CBP-1)*, 2004.

[13] G. Loh, "The frankenpredictor," in *The 1st JILP Championship Branch Prediction Competition (CBP-1)*, 2004.

[14] V. Desmet, H. Vandierendonck, and K. D. Bosschere, "A 2bcgskew predictor fused by a redundant history skewed perceptron predictor," in *The 1st JILP Championship Branch Prediction Competition (CBP-1)*, 2004.

[15] P. Michaud, "A ppm-like, tag-based predictor," in *The 1st JILP Championship Branch Prediction Competition (CBP-1)*, 2004.

[16] S. Pan, K. So, and J. Rahmeh, "Improving the accuracy of dynamic branch prediction using branch correlation," in *Proceedings of the 5th International Conference on Architectural Support for Programming Languages and Operating Systems*, 1992.

[17] S. McFarling, "Combining branch predictors," tech. rep., DEC, 1993.

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

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

[20] M. Evers, P. Chang, and Y. Patt, "Using hybrid branch predictors to improve branch prediction accuracy in the presence of context switches," in 23^{rd} *Annual International Symposium on Computer Architecture*, pp. 3–11, 1996.

[21] L. N. Vintan and M. Iridon, "Towards a high performance neural branch predictor," in *IJCNN'99. International Joint Conference on Neural Networks. Proceedings.*, 1999.