The O-GEHL branch predictor *

André Seznec
IRISA/INRIA/HIPEAC

Abstract

We introduce the Optimized GEometric History Length (O-GEHL) branch predictor which efficiently exploits very long global histories in the 100-200 bits range. The O-GEHL predictor is derived from the GEometric History Length (GEHL) predictor.

The GEHL predictor features several predictor tables $T(i)$ (e.g. 8) 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. The set of used global history lengths forms a geometric series, i.e., $L(j) = \alpha^{j-1}L(1)$, thus allowing to efficiently capture correlation on recent branch outcomes as well as on very old branches.

The O-GEHL predictor further improves the ability of the GEHL predictor to exploit very long histories through the addition of dynamic history fitting and dynamic threshold fitting.

Presentation outline

In Section 1, we first present the GEHL predictor principles and its main characteristics. Section 2 describes the extra features of the O-GEHL predictor and the configuration of the predictor submitted to CBP. Section 3 briefly reviews the related works that had major influences in the O-GEHL predictor proposition.

Due to space limitations, no performance analysis of the O-GEHL behavior is provided in this write-up.

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

1 The GEometric History Length branch predictor

1.1 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), $S = \frac{M}{2} + \sum_{0 \leq i < M} C(i)$ [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, $1 \leq i < M$ are of the form $L(i) = \alpha^{i-1} * L(1)$, i.e., the lengths L(i) form a **geometric series**.

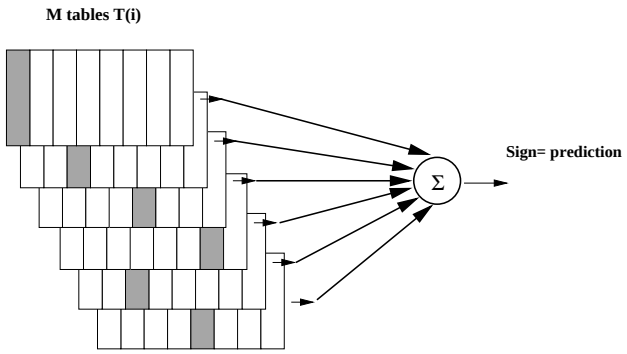

Figure 1. The GEHL predictor

For the CBP submission, we only focus on a 8-table GEHL predictor.

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

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

1.3 Degrees of freedom in the design of GEHL predictors

Numerous degrees of freedom exist in the design of a GEHL predictor. First one can vary the number M of tables in the GEHL predictor. Experiments showed that using 4 to 12 tables provides high level of accuracy. Second, experiments showed that using 4-bit or 5-bit counters with 4 to 12 tables is cost-effective. Third, one can vary the parameters L(1) and L(M-1) [2]. Experiments showed that for storage budgets varying from 32 Kbits to 1 Mbits, the GEHL predictor is able to capture correlation on very long histories in the hundred bits range. Fourth, one can vary the threshold θ for updating the predictor. Using $\theta = M$, the number of tables in the GEHL predictor is a good tradeoff.

1.4 About indexing functions complexity

The index functions used in the submitted predictor involve a single stage of three-entry exclusive-OR gates for computing each index bit. Some of the address bits and some of the history bits are ignored 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 and the L(i) bits of history. Then we simply hash this 3n bit vector in a n-bit vector using a single stage of 3-entry exclusive-OR gates. (see the INDEX function in the predictor code).

[2] $\alpha = (\frac{L(M-1)}{L(1)})^{\frac{1}{M-2}}$

1.5 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 lead to consider some paths 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.

2 The O-GEHL predictor

In this section, we present the O-GEHL predictor. The O-GEHL predictor is derived from the GEHL predictor by augmenting it with dynamic update threshold fitting and dynamic history length fitting.

2.1 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 (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.

2.2 Dynamic history length fitting for the GEHL predictor

Juan et al [3] proposed to continuously adapt the branch history length during execution for global history branch predictors. The 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 of 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) & ($|S| \leq \theta$)){
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.**

Impact of adaptative history length fitting

On the CBP benchmarks, using adaptative history length fitting on the GEHL predictor reduces by 7 % in average the number of mispredictions instead of using a single history length per predictor table.

2.3 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.

The characteristics of the submitted O-GEHL predictor are summarized in Table 1. Using 200 as L(10) the maximum history length and 3 as L(1) is one of the best tradeoffs on the set of the benchmark traces. However using any value in the interval 150-250 for L(10) and any value in the interval 2-6 for L(1) brings very similar simulation results, i.e. the total number

Table	T0	T1	T2	T3	T4	T5	T6	T7
short history length	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 length	-	-	L(8)=79	-	L(9)=125	-	L(10)=200	-
counter width (bits)	5	5	4	4	4	4	4	4
tag bit	-	-	-	-	-	-	-	0.5
entries	2K	1K	2K	2K	2K	2K	2K	2K
storage budget (bits)	10K	5K	8K	8K	8K	8K	8K	8K +1K (tags)

Table 1. Characteristics of the O-GEHL predictor submitted to the CBP: a total of 64Kbits

of mispredictions for these pairs of values are not exceeding the presented results by more than 4%.

2.4 Performances of the O-GEHL predictor

The simulation results obtained with the O-GEHL predictor are summarized in Table 2.

3 The O-GEHL predictor and related works

The use of multiple global history lengths in a single branch predictor was initially introduced in [4], then it was refined by Evers et al. [1] 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 [5, 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 [3] 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.

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

Table 2. Accuracy of the 64Kbits O-GEHL predictor (misp/KI)

References

[1] M. Evers, P.Y. Chang, and Y.N. 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*, pages 3–11, 1996.

[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] 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.

[4] S. McFarling. Combining branch predictors. Technical report, DEC, 1993.

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