Improving the O-GEHL Branch Prediction Accuracy Using Analytical Results

Ekkasit Tiamkaew
Department of Computer Science
Naresuan University, Phitsanulok 65000, Thailand

Angkul Kongmunvattana
Department of Computer Science
Columbus State University, Columbus, GA 31907, USA
ekkasit.tiamkaew@gmail.com

ABSTRACT

The O-GEHL branch predictor has outperformed other prediction schemes using the same set of benchmarks in an international branch prediction contest, CBP-1. In this paper, we present the analysis results on each of the O-GEHL branch predictor tables and also on the optimal number of predictor tables. Two methods are subsequently proposed to help increase the O-GEHL prediction accuracy. The first one aims to increase the space utilization of the first predictor table by dynamically adjusting the lengths of branch history regarding to the type of a benchmark currently in execution. The second one adds an extra table into the O-GEHL predictor using the space saved from the sharing of hysteresis bits. Experimental results have confirmed that both schemes improve the accuracy of two different predictor configurations, leading to two promising research directions for future explorations.

Keywords: Branch Predictor, Neural Network, Perceptron, O-GEHL, Predictor Analysis

1. INTRODUCTION

In the first Championship Branch Prediction competition (CBP-1) [1], the first-placed branch predictor is the O-GEHL (Optimized GEometric History Length) predictor [12], which has modified the perceptron predictor to exploit various lengths of global branch history. It also contains a dynamic mechanism that can adaptively adjust the history lengths used in 3 of its 8 predictor tables, allowing for the use of even longer branch history when necessary.

Figure 1 shows the mechanism of the O-GEHL predictor containing 8 predictor tables, T0-T7. Different weights from each table entries are selected using various forms of indexing parameters, and then added together to generate the prediction outcome, which is the sign of the final sum. Only the first table, T0, is always indexed by a branch address alone. Various combinations of branch address and global branch history are used to index the remaining 7 tables. Let L(i) be the length of global branch history used in the indexing function of the ith table. L(0) and L(1) are initially set to 0 and 3, respectively. The values of other L(i) can be computed using the geometric series equation: $L(i) = \alpha^{i-1} * L(1)$, where a value of α is $\{L(M-1)/L(1)\}^{1/(M-2)}$ and M is the number of predictor tables.

In order to exploit really large branch history, the O-GEHL predictor has set M to 11 and L(10) to 200 even though it has only 8 physical predictor tables. All these tables are initially programmed to use L(0) to L(7) as the table indices. However, T2, T4, and T6, can adaptively change their branch history lengths, switching back and forth between using L(2), L(4), and L(6) to L(8), L(9), and L(10), respectively, with regard to the current branch characteristics. The number of entries in each predictor table is 2K, with the exception of T1, which has only 1K entries. Each entry in T0 and T1 is represented by a 5-bit counter, while an entry in the remaining tables uses a 4-bit counter. These parameters have been empirically selected and therefore proven to optimize the performance of the O-GEHL predictor.

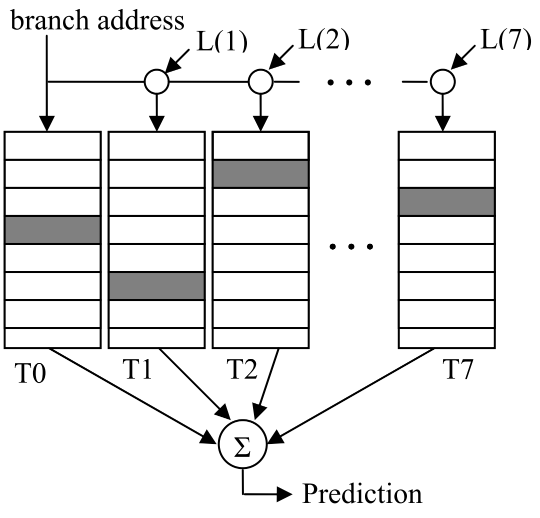

Figure 1. O-GEHL branch predictor

We have performed an extensive analysis on the O-GEHL predictor, mainly searching for its characteristics that can be exploited to further increase the predictor efficiency. The analysis results indicate that, of all predictor table, the first one, T0, is the least utilized due to its use of short branch history as a parameter to the index function, and that more predictor tables generally translate into higher prediction accuracy. Subsequently, we propose two alternate designs of the O-GEHL branch predictor to improve its accuracy, which are: 1) increasing the space utilization of its first predictor table, T0, by dynamically adjusting branch history lengths used in the indexing function, and 2) adding an additional predictor table without requiring extra space by means of sharing hysteresis bits.

Our simulation results show that the first proposed scheme improves the prediction accuracy in almost all hardware budgets, except for 8K and 16K bits. Meanwhile, the second scheme provides best performance when working with branch predictors larger than 64K bits, indicating that when there are abundant hardware resources, adding extra predictor tables is more likely to improve the prediction accuracy than increasing the size of the existing predictor tables. Further analysis of the experimental results reveals that even though the second scheme performs worse than the first one on average, particularly in FP and INT benchmarks, it enjoys more success in some MM and all SERV benchmarks.

The rest of this paper proceeds as follows. Section 2 provides a background on the O-GEHL branch predictor as well as related work, while section 3 describes our experimental methodology. Section 4 shows our analysis results on the O-GEHL predictor. Meanwhile, our proposed mechanisms and the experimental results are discussed in section 5. Finally, we conclude in Section 6.

2. RELATED WORK

Jimenez has first proposed to use the perceptron in neural learning branch predictor for studying and predicting branch outcomes [6, 7]. Despite having higher prediction accuracy than other prediction methods, the predictor's high complexity and long prediction latency have made it rather impractical.

In CBP-1 [1] where complexity is not an issue, two highest ranked branch predictors are proposed by Gao [5] and Seznec [12], both of which have modified the mechanisms to improve the accuracy of the perceptron predictor. Under a distributed set of benchmarks, the O-GEHL predictor [12] scores the highest and therefore is chosen as the subject of our investigation.

The idea of multiple global history lengths used in the O-GEHL predictor was initially introduced in [10], and then refined by Evers et al. [4]. Since then, it has appeared in several branch prediction schemes and been proven to be highly effective in boosting the prediction accuracy. One of our proposed methods has adopted the same concept by increasing the number of the O-GEHL predictor's tables to provide more diversity into the global history lengths.

In order to add another predictor table without wasting up more hardware resources, we follow the footstep of Seznec et al. who proposed to have 2 table entries in the branch predictor share the same hysteresis bit [13]. It has been demonstrated that this sharing mechanism has an insignificant impact on the accuracy of certain branch predictors, especially the gshare branch predictor, in which the sharing only slightly increases the entropy per prediction from 0.18 to 0.19 [9].

Gao's branch predictor, a runner-up in the first round of CBP-1, used an adaptive approach to modify the perceptron predictor to suit each benchmark [5]. We have adopted a similar idea in one of our proposed schemes in order to distinguish individual benchmarks for an appropriate adjustment of the indexing function's parameters.

3. EXPERIMENTAL METHODOLOGY

In order to conduct our experiments, we use a branch predictor framework and input traces from the 1st Championship Branch Prediction competition (CBP) [1], which is sponsored by Intel MRL and IEEE TC-uARCH. This branch prediction contest provides participants with a common evaluation framework, written in C++, and a fixed hardware budget, which is 64K bits, to implement and evaluate their branch prediction algorithms.

In our experiments, we performed a series of simulations using the provided framework with each of the benchmark programs distributed by the competition committee. Benchmarks are classified into 4 categories: FP (floating point), INT (integer), MM (multimedia), and SERV (server). All the simulations have been run until completion. The performance metric used throughout this paper is the output from the framework, which is the number of mispredictions per 1000 instructions.

Because the O-GEHL predictor's original size is 64K bits, enlarging or reducing the predictor requires some modifications to the predictor configurations. We have decided to simply double, or halves, the number of entries in each predictor table when increasing, or decreasing, the predictor size.

4. ANALYSIS OF THE O-GEHL BRANCH PREDICTOR

Space utilization of the predictor tables

After careful and thorough analysis of the predictor to promote a better understanding of its characteristics and to probably discover its shortcomings, we have found quite an interesting statistics regarding the space utilization of each predictor table. As shown in table 1, while the allocated spaces of most predictor tables are used more than 74%, even more than 92% in three of them, the usage percentage of T0 is surprisingly low, just around 52%. This apparently introduces an opportunity for enhancing the predictor's performance.

	T 0	T 1	T 2	T 3
utilization (%)	52.66	74.07	86.90	75.39

	T 4	T 5	T 6	T 7
utilization (%)	92.12	86.88	93.56	92.75

Table 1. Space utilization in each predictor table

Despite its low utilization, we cannot simply reduce T0 size without inflicting considerable damage to the prediction accuracy. This is because each benchmark exhibits a highly different branch distribution pattern across T0. In an attempt to seek for an insight into branch behaviors and eventually find a way to make a better use of T0 space, we have conducted further studies of the way T0 entries are occupied. The results shown in table 2 and 3 are the number of entries that has been used for each benchmark during the entire execution. They are grouped into 4 categories, which are FP, INT, MM, and SERV, as previously mentioned. The average value of each category is also calculated and shown in the table.

From the results, all SERV benchmarks have their branches spread throughout T0, making full use of the allocated table space. This is a level of success other benchmarks unfortunately fail to match. Moreover, several FP and INT benchmarks (even one from MM) have occupied only 20% of the total T0 capacity. Not only do such drastic variations in the table usages undoubtedly call for a more efficient way to distribute branches across the table, but also indicate how difficult it would be to find one.

FP 1	400
FP 2	405
FP 3	682
FP 4	489
FP 5	231
Average	441.4

INT 1	386
INT 2	1105
INT 3	791
INT 4	568
INT 5	392
Average	648.4

Table 2. The number of T0 entries used FP and INT

MM 1	409
MM 2	1434
MM 3	852
MM 4	1385
MM 5	1823
Average	1180.6

SERV 1	2037
SERV 2	2040
SERV 3	2047
SERV 4	2047
SERV 5	2045
Average	2043.2

Table 3. The number of T0 entries used MM and SERV

However, higher space utilization in other predictor tables suggests that the simplest and most intuitive scheme to improve T0 usage percentage probably is to use global branch history as another parameter for the indexing function of T0. In the other word, L(0) of the new scheme is no longer zero.

We have performed various experiments with different values of L(0). One of the best configurations is to set L(0) to 3 and L(1) to 5 while other L(n) are recalculated using to the same equation given in the section 2.1. Figure 2 shows the effects this configuration has on the prediction accuracy for each benchmark set. FP and INT benchmarks benefit from larger L(0), gaining 5.22% and 1.99% of prediction accuracy, respectively. Unfortunately, the overall prediction accuracy is decreased because MM and SERV benchmarks suffer 7.24% and 10.24% loss in prediction accuracy respectively. This scenario gets worse when other configurations with even larger L(0) have been used in the experiments. The results reveal that while an indexing function with larger L(0) improves the prediction accuracy of the benchmarks that exhibited low utilization percentage, specifically FP and INT, it degrades that of the others.

Apparently, short global branch history still plays a significant role in maintaining high branch prediction accuracy in the O-GEHL predictor, especially for both MM and SERV benchmarks. This makes the task of improving the predictor's efficiency and accuracy much more complicated since simply increasing the length of global branch history alone, as done in traditional perceptron predictor [7], is not going to accomplish the job anymore.

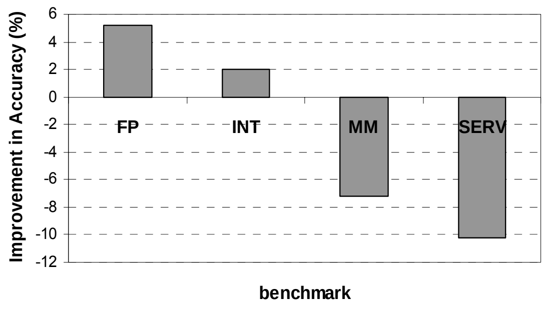

Figure 2. Improvement in prediction accuracy when L(0) = 3 and L(1) = 5

The effects of T8, T9, and T10 after being allocated their own space.
Due to a space limitation in the O-GEHL predictor, table T8, T9, and T10 are arranged to collocate with T2, T4, and T6, respectively. These extra tables are physically assigned their own spaces in our experiments for a study of their impact on the prediction accuracy. As shown in table 4, approximately 1% improvement on prediction accuracy is observed each time an extra table is added. The third table (T10) however is an exception as it causes a slight drop in the prediction accuracy. These results show that only 9 or 10 tables are sufficient to obtain higher predictor efficiency.

	O-GEHL (64K)	9 tables (72K)	10 tables (80K)	11 tables (88K)
Mispredict (%)	2.82	2.79	2.75	2.76

Table 4. The effects on prediction accuracy when more tables are added into the predictor

We have also performed other experiments where the traditional O-GEHL predictor is added with a rather small table (1-2K bits). Various versions of global branch history with different lengths are used to index to the extra table. Even with four-way associativity being implemented on the table to increase its space utilization, the consequent prediction accuracy is not improved. This result, along with the finding from table 3 that most of MM and all SERV benchmarks use up almost all the table entries, reveals that an extra table requires a large number of entries, preferably 2K entries, to have a positive impact on the overall prediction accuracy.

5. OUR PROPOSED APPROACHES AND THE EXPERIMENTAL RESULTS

Increasing space utilization in T0
One interesting fact about T0 is that it is under utilized when used with particular benchmarks but almost fully utilized with the others. Therefore, static modifications to table size or indexing function cannot be made without sacrificing the prediction accuracy in some benchmarks. An appropriate approach is to find a means to dynamically adjust the length of global branch history used in the indexing function of each predictor table, which is L(n) where $0 \le n \le 10$, with regard to what kind of benchmark is being used at the time.

Distinguishing between benchmarks that need a small L(n) and those that need a larger one is a challenging problem. We have decided to use the number of conditional branches as a deciding factor in this process since in most cases there are more of them in FP and INT than in MM and SERV benchmarks. In our approach, each L(n) is initially given a default value as done in the O-GEHL predictor. After a certain time period, *t*, has passed, the number of conditional branches, *c*, is then compared to a pre-specified value, *v*. If *c* is less than *v*, it is likely that the currently running benchmark program is in either MM or SERV category and these L(n) values are not changed. Otherwise, L(0) is set to 3, L(1) to 5, and L(10) to 200, which is the same value, while all other L(n) values will be re-calculated using the given geometric series equation. These values have been proven to have biggest positive impact on the prediction accuracy for FP and INT benchmarks, as previously shown in figure 2. Once this process is done, regular prediction mechanism of the O-GEHL predictor can go on without any other interruption. As a result, an overhead cost is kept very small and each benchmark is likely to run in the predictor configuration that suits it the most.

Not only should *t* be small enough to allow for a timely adjustment of the L(n) values, but should also be sufficiently large for an efficient classification of benchmark programs. Therefore, a 7-bit counter responsible for tracking the number of all branch instructions, both conditional and unconditional ones, is chosen to represent the amount of time that has passed. The counter overflow signals that the time *t* is reached, and triggers the L(n) adjustment process. The use of this counter allows the time *t* to be extremely short, compared to the whole execution time, while the value of *v* can be set to, based on the information we have gathered on previous runs of all benchmark programs, approximately 84% of the counter's maximum value possible. With this scheme, another 7-bit counter is required to track the number of conditional branches, and a comparator is needed for the comparison process.

We have added this mechanism into the O-GEHL predictor and run the experiments with various budget sizes, ranging from 8K-1M bits. The results are shown in figure 3.

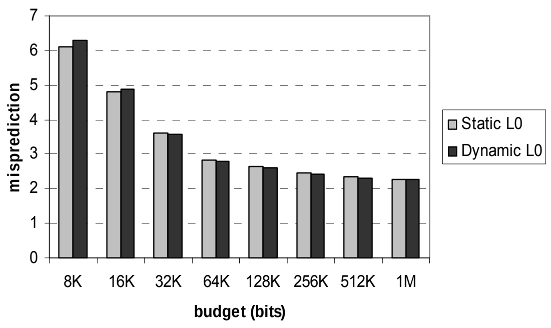

Figure 3. Misprediction rates when the dynamic adjustment of L(n) is incorporated into the predictor

The prediction accuracy is improved in every case except for 8K- and 16K-bit predictors. This is not surprising since T0 in these budgets is 4-8 times smaller than usual and the task of increasing its space utilization becomes almost impossible to accomplish. However, a 0.025 average decrease in misprediction rate can be observed in larger predictors, which is equal to 0.96% improvement in prediction accuracy. In fact, the most impressive improvement occurs in the 128K-bit predictor, which exhibits an almost 2% gain in prediction accuracy. 512K- and 1M-bit predictors are also enjoying more than 1% increase in prediction accuracy. This is all because higher space utilization can be achieved when T0 is larger.

The fact that the O-GEHL predictor is specifically optimized for a 64K-bit budget suggests that the predictors with larger budget may be able to experience even higher rise in prediction accuracy with different combination of L(n) or different space allocation method for each predictor table. This requires extensive examinations of the predictor with an extremely large set of parameters, and therefore is outside the scope of our paper. However, one can easily begin the study by concentrating on T0 space utilization since our analysis and experimental results have already confirmed that it certainly is an important factor in deciding the prediction accuracy.

It is also inevitable that further performance improvement of the highly optimized 64K-bit O-GEHL predictor will be highly difficult to achieve. Nevertheless, our scheme still has reduced the misprediction rate by 0.02, which is roughly a 0.71% improvement. Though this number seems rather small, it is almost 7 times the difference in prediction accuracy between the first-placed O-GEHL predictor and the second-placed Gao's proposed predictor [5], which is only 0.003 apart (a tiny 0.11% increase). The fact that we has improved the accuracy of the already highly-optimized branch predictor makes the results even more impressive.

	FP	INT	MM	SERV
8K	7.42	-6.55	-1.03	-1.27
16K	15.30	2.63	-0.38	-3.07
32K	18.58	5.63	0.02	-4.95
64K	13.24	2.06	0.03	-1.13
128K	27.6	0.22	0.69	-0.04
256K	14.65	0.83	-0.22	-1.28
512K	13.17	1.30	0.37	-1.25
1M	20.75	0.79	0.27	-2.28
Average	16.34	0.86	-0.03	-1.91

Table 5. Improvement percentage in the prediction accuracy for each benchmark group

To elaborate the contributions made by our approach, we have performed an analysis to evaluate its impact on the prediction accuracy of each benchmark category, as shown in table 5. The decrease in prediction accuracy is displayed in a negative number.

FP benchmarks are undoubtedly the biggest beneficiary with an average of 16.34% increase in prediction accuracy. The lowest improvement, 7.42%, lies in the predictor with 8K-bit budget while the highest one, 27.6%, is enjoyed by the128K-bit predictor. Meanwhile, almost all INT benchmarks are experiencing various levels of improvement in prediction accuracy, particularly with the highest rise of 5.63% belonging to the 32K-bit predictor. The average increase in prediction accuracy of all INT benchmarks is disappointedly only 0.86%. This is mainly because of a 6.55% deficit in the 8K-bit predictor, which, among the predictors of all sizes, is the only one that suffers a prediction accuracy loss. These improvements are the direct consequences from heightening T0 space utilization, which is however not the case for SERV benchmarks that unfortunately have not benefited from our proposed scheme at all. Insignificant changes around or less than 1% in prediction accuracy can also be observed in all MM benchmarks.

The decrease of prediction accuracy in MM and SERV benchmarks takes place because the process of distinguishing between each benchmark category is not efficient enough. As a result, a few MM and SERV benchmark programs sometimes have to be run with large L(n), causing more mispredictions. To perfectly classifying the benchmarks, either a highly complicated mechanism must be used or a thorough profiling must be conducted. The former is very likely going to slow down the prediction process while requiring more die space. On the other hand, despite being faster and requiring no extra space, the latter is highly dependent on the profiled benchmarks and cannot adapt well to a new type of benchmarks.

The results also reveal that our proposed scheme works particularly well with most FP and INT benchmarks and is therefore most suitable for microprocessors that run scientific and computation-intensive applications. For computers whose jobs involve running applications with multimedia or server workloads, a traditional O-GEHL predictor is more appropriate. Nonetheless, because its overall misprediction rate is still lower than the O-GEHL predictor's, our proposed approach is considered a better alternative in microprocessors running general applications with unknown workloads.

Adding an extra predictor table without incurring the space required

Prediction accuracy usually increases when a predictor is allocated larger die space, or more predictor tables in a case of perceptron predictor. However, the analysis results in section 4.2 have demonstrated that only up to 2 predictor tables are actually required for the O-GEHL predictor. Our proposed approach is to add an extra table into the predictor without ever increasing the allocated table space. With the introduction of hysteresis bits being shared between 2 table entries, the number of entries in each predictor table does not even need to be reduced to make room for an extra table.

In a 2-bit counter, the most significant bit is a direction bit, which provides the prediction result, while the least significant bit is a hysteresis bit, which prevents the direction bit from immediately changing after just a single misprediction. Seznec et al. have proposed an approach of sharing a single hysteresis bit between 2 adjacent table entries [13] to save up more die space of the branch predictor in an EV8 microprocessor and then use it in a more fruitful fashion,. This strategy has been proven to work efficiently by Loh, who shows that hysteresis bits are strongly biased and that it is unnecessary to waste an entire bit for hysteresis in a single table entry [9]. His experimental results also illustrate that, by having 2 entries in gshare predictor share a hysteresis bit, the entropy per prediction has increased only slightly, implying that its prediction accuracy is unlikely to be significantly affected.

We have adopted the hysteresis sharing policy to save the predictor space and create room for an extra predictor table. Since higher degree of aliasing is an inevitable consequence to the hysteresis-bit sharing scheme and is likely to cause more damages with small predictor tables, T1 is exempted from being implemented the mechanism. As a result, the overall die space saved from using this approach in the 7 remaining predictor tables, each of which contains 2K entries, is exactly 7K bits. The area of this size is precisely what another predictor table, T8 in this case, actually needs as it will contain 2K entries of 3-bit counters and 1K entries of a hysteresis bit.

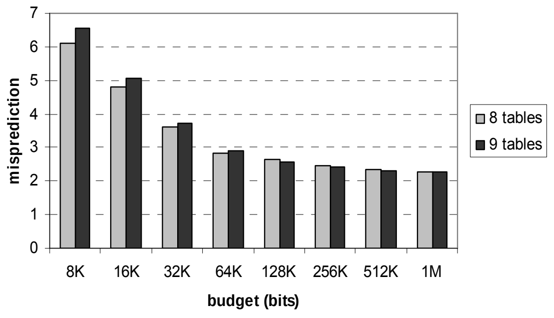

Figure 4. Misprediction rates when 9 predictor tables with shared hysteresis bits are used in the predictor

The misprediction rates, before and after the sharing of hysteresis bits, across various predictor budgets are shown in figure 4. Listing of the improved percentages in prediction accuracy for each predictor is also presented in table 6. The results get worse with the predictor of size 8K-64K bits while some improvement can be seen in the predictors larger than 64K bits. This is simply because small branch predictors tend to already have high aliasing rate. Sharing hysteresis bits is likely to multiply increase the interference to the extent that it cannot be compensated by the benefits gained from having an extra predictor table anymore.

Size (bits)	Improv (%)
8K	-7.75
16K	-5.73
32K	-3.50
64K	-2.38
Average	-4.84

Size (bits)	Improv (%)
128K	2.54
256K	0.95
512K	1.08
1M	1.05
Average	1.40

Table 6. Improved prediction accuracy across all predictor budgets

Table 6 shows that the average decrease in prediction accuracy of the predictors sized 8K-64K bits is 4.84%, indicating that the strategy does not work well in small branch predictors. Meanwhile, the average increase in prediction accuracy of the predictors with hardware budget between 128K and 1M bits is 1.40%, which is better than 1.17% improvement obtained from the equal-sized predictors with dynamic-L(n) scheme. The highest improvement percentage of prediction accuracy from the hysteresis-sharing scheme is 2.54% in 128K-bit predictor and the lowest is 0.95% in 256K-bit predictor, while in the dynamic L(n) scheme the highest is only 1.96% in 128K-bit predictor and the lowest only 0.57% in 256K-bit predictor. These results reveal to us a very interesting discovery: Even though the hysteresis-sharing scheme incurs, on average, higher misprediction rate when all hardware budgets are considered, it outperforms the dynamic-L(n) scheme in large branch predictors. Consequently, it can be concluded that a larger number of tables definitely are an essential factor in enhancing the O-GEHL prediction accuracy when hardware budget is not limited.

Size (bits)	FP	INT	MM	SERV
128K	24.88	-2.94	0.47	8.64
256K	10.08	-0.43	-0.18	3.98
512K	14.55	0.77	-1.92	0.20
1M	16.05	1.00	4.14	-1.55
Average	16.39	-0.40	0.63	2.82

Table 7. Improvement percentage in prediction accuracy for each benchmark category, for predictors with budget larger than 64K bits

To further elaborate the contributions made by our approach, we have performed an analysis to evaluate its impact on the accuracy of large predictors (128K-1M bits) with each benchmark category, as shown in table 7. Even though the hysteresis-sharing scheme has had a reasonable success with all FP benchmarks, with an impressive average improvement of 16.39% in prediction accuracy, it is still slightly lagging behind the dynamic-L(n) scheme that exhibits an average of 19.04% increase in prediction accuracy over the same range of hardware budget. However, on average for predictors of size 128K-1M bits, it manages to improve 0.63% and 2.82% of prediction accuracy in MM and SERV benchmarks, respectively, while the dynamic-L(n) scheme can only achieve 0.28% improvement in MM benchmark and even suffer 1.21% deficit in SERV benchmark. These results confirm the hysteresis-sharing scheme as an attractive alternative in improving the prediction accuracy in large predictors, with a wide range of applications.

Combining the two proposed approaches

In term of prediction accuracy, the combination of the two mechanisms is only slightly better than the dynamic-L(n) scheme and is still inferior to the hysteresis-sharing scheme when the predictor size is larger than 64K bits. For the predictors of smaller size, it performs worse than each individual scheme. Hence, we choose to omit the discussion of its experimental results.

6. CONCLUSIONS

We have performed an analysis on the O-GEHL predictor and, based on the results, proposed and evaluated two new mechanisms to help improve its prediction accuracy. Since T0 is underutilized, our first proposed scheme is to increase its space utilization by dynamically adjusting the lengths of global branch history to best suit the benchmark type. The second scheme proposes to add an extra predictor table without incurring any hardware budget, using the space saved from sharing a single hysteresis bit between 2 table entries

The first scheme increases the prediction accuracy in almost every hardware budget but 8K and 16K bits, which are too small for the approach to be effective. In a budget of 64K bits, a regular O-GEHL predictor size, our approach improves the prediction accuracy by a much larger margin than what the O-GEHL predictor does over the second-placed predictor in the CBP competition. It also works particularly well with most FP and INT benchmarks and is thus suitable for microprocessors that usually run scientific and computation-intensive applications. Furthermore, its negative impact on MM and SERV benchmarks suggests that one of the possible future research directions is to find a more efficient way to classify the benchmarks to prevent the predictor from using inappropriate global branch history lengths.

Even though the second scheme causes the prediction accuracy to drop when a hardware budget is 8K-64K bits, it outperforms the first scheme when a hardware budget is 128K-1M bits. A larger number of tables are hence an essential factor in improving the O-GEHL prediction accuracy when hardware resources are not limited. Moreover, for branch predictors of size 128K-1M bits, it is even capable of increasing the prediction accuracy in MM and SERV benchmarks, which is a task the first scheme fails to accomplish. This is therefore an attractive scheme in improving the accuracy of large branch predictors in microprocessors running a wide range of applications.

For branch predictors sized 32K and 64K bits, the first scheme using dynamic L(n) is the best option, while the O-GEHL predictor with regular configurations is still the best when hardware resources are very limited, 8K-16K bits in this case.

7. ACKNOWLEDGEMENT

This work was supported in part by NSF under Grant EIA-0116759 and a gift from Sun Microsystems, Inc.

8. REFERENCES

[1] The 1st JILP Championship Branch Prediction Competition (CBP-1), Journal of Instruction-Level Parallelism, 2004, http://www.jilp.org/cbp/.

[2] P-Y. Chang, M. Evers, and Y. N. Patt, "Improving Branch Prediction Accuracy by Reducing Pattern History Table Interference", *Proceedings of the 1996 ACM/IEEE Conference on Parallel Architectures and Compilation Techniques*, October 1996.

[3] A. N. Eden and T. N. Mudge, "The YAGS Branch Prediction Scheme", *Proceedings of the 31st ACM/IEEE International Symposium on Microarchitecture*, November 1998, pp. 69-77.

[4] M. Evers, P.Y. Chang, and Y.N. Patt, "Using Hybrid Branch Predictors to Improve Branch Prediction Accuracy in the Presence of Context Switches", *Proceedings of the 23rd International Symposium on Computer Architecture*, 1996, pp. 3–11.

[5] H. Gao and H. Zhou," Adaptive Information Processing: An Effective Way to Improve Perceptron Predictors", *The 1st JILP Championship Branch Prediction Competition (CBP-1) in conjunction with MICRO-37*, December 2004.

[6] D. A. Jimenez, "Fast Path-Based Neural Branch Prediction", *Proceedings of the 36th International Symposium on Microarchitecture*, December 2003, pp. 243–252.

[7] D. Jimenez and C. Lin, "Dynamic branch prediction with perceptrons", *Proceedings of the 7th International Symposium on High Performance Computer Architecture*, 2001.

[8] C-C Lee, I-C. K. Chen, and T. N. Mudge, "The Bi-mode Branch Predictor", *Proceedings of the 30th ACM/IEEE International Symposium on Microarchitecture*, December 1997, pp. 4-13.

[9] G. H. Loh, D. S. Henry, and A. Krishnamurthy, "Exploiting Bias in the Hysteresis Bit of 2-bit Saturating Counters in Branch Predictors", *Journal of Instruction-Level Parallelism*, vol. 5, June 2003, pp. 1-32.

[10] S. McFarling, "Combining Branch Predictors", Technical Report TN-36, Digital Western Research Laboratory, June 1993.

[11] P. Michaud, A. Seznec, and R. Uhlig, "Trading Conflict and Capacity Aliasing in Conditional Branch Predictors", In *Proceedings of the 24th International Symposium on Computer Architecture*, May 1997, pp. 292-303.

[12] A. Seznec, "The O-GEHL Branch Predictor", *The 1st JILP Championship Branch Prediction Competition (CBP-1) in conjunction with MICRO-37*, December 2004.

[13] A. Seznec, S. Felix, V. Krishnan, and Y. Sazeides, "Design Tradeoffs for the Alpha EV8 Conditional Branch Predictor", In *Proceedings of the 29th International Symposium on Computer Architecture*, May 2002.

[14] E. Sprangle, R. S. Chappell, M. Alsup, and Y. N. Patt, "The Agree Predictor: A Mechanism for Reducing Negative Branch History Interference", *Proceedings of the 24th International Symposium on Computer Architecture*, June 1997, pp. 284–291.

Received: Jun. 2006. Accepted: Oct. 2006.