Practical Multidimensional Branch Prediction

André Seznec*, Joshua San Miguel†, Jorge Albericio†
*IRISA/INRIA †University of Toronto

***Abstract*—The most efficient branch predictors proposed in academic literature exploit both global branch history and local history. However, local history predictors introduce major design challenges, particularly for the management of speculative histories. The wormhole (WH) branch predictor was recently introduced to exploit branch outcome correlation via multidimensional histories. For some branches encapsulated in a multidimensional loop, their outcomes are correlated with those of the same branch in neighbor iterations, but in the previous outer loop iteration. Unfortunately, the practical implementation of the WH predictor is even more challenging than the implementation of local history predictors. In this paper, we introduce a practical, cost-effective mechanism for capturing multidimensional branch correlations: the Inner Most Loop Iteration (IMLI) counter.**

I. Introduction

Improved branch prediction accuracy directly translates in performance gain through reducing the total overall branch misprediction penalty. It also translates in direct energy consumption reduction through reducing the number of instructions on the wrong path. Therefore replacing the branch predictor by a more accurate one is one of the simplest and energy effective mean to improve the performance of a superscalar processor since it can be done without reopening the design of the overall execution core.

Since the introduction of two-level branch prediction [1], academic branch predictors have been relying on two forms of branch history: global branch or path history and local branch history. However, local history branch predictor components bring only limited accuracy benefit over global history predictors, yet they introduce very complex hardware management of speculative histories. Therefore, most effective hardware designs only use global history components and sometimes a loop predictor [2], [3].

The wormhole (WH) [4], [5] branch predictor was recently introduced to exploit branch outcome correlation in multidimensional loops. For some branches encapsulated in a multidimensional loop, their outcomes are correlated with those of the same branch in neighbor iterations, but in the previous outer loop iteration as illustrated in Figure 1. Unfortunately, the practical implementation of the WH predictor is even more challenging than the implementation of local history predictors.

In this paper, we show that branch output correlations that exist in multidimensional loops can be tracked by cost-effective predictor components: the IMLI-based predictor components (**I**nner **M**ost **L**oop **I**teration). The IMLI-based components can be added to a state-of-the-art global history predictor, and their speculative states can be easily managed. Our experiments show that in association with a main global history predictor such as TAGE [6] or GEHL [7], the two IMLI-based components achieve accuracy benefits in the same range as the ones achieved with local history and loop predictor components. This benefit is obtained at much lower hardware cost and complexity: smaller storage budget, smaller number of tables and much simpler speculative management of the predictor states. Therefore, the IMLI-based components are much better candidates for real hardware implementation than local history predictors and even loop predictors.

Originally "The Inner Most Loop Iteration counter: a New Dimension in Branch History" in the 48th Annual IEEE/ACM International Symposium on Microarchitecture, December 5-9, 2015.

```
for (N=0;i <Nmax;N++)
        for (M=0; M <Mmax; M++){
                if (A[M+N] >0) { ..}              // Branch B1
                if (B[N][M]-B[N-1][M])>0{..}  // Branch B2
                if (C[M]>0)                       // Branch B3
                        if(D[M] >0) {..}          // Branch B4
        }
```

Fig. 1: Branches whose outcomes are correlated with previous iterations of the outer loop

II. Multidimensionality

Albericio et al. [4], [5] recognize that, in many cases, hard-to-predict branches are encapsulated in multidimensional loops. They demonstrate that the outcome of a branch in the inner most loop is correlated with the outcome(s) of the same branch in the same iteration or neighbor iterations of the inner loop but in the previous outer loop iteration. Say B is a branch in the inner loop IL encapsulated in outer loop OL. If Out[N][M] is the outcome of B in iteration M of IL and in iteration N of OL, then Out[N][M] is correlated with Out[N-1][M+D], where D is a small number (e.g -1, 0 or +1).

This is illustrated in Figure 1. We assume that arrays A, B, C and D are not modified by the (not represented) internal code. The outcome of branch B1 in iteration (N,M) is equal to its outcome in iteration (N-1,M+1). The outcome of branch B2 is weakly correlated with its outcome in iteration (N-1,M). The outcome of branch B3 is equal to its outcome in iteration (N-1,M). If executed, the outcome of branch B4 is equal to its outcome in iteration (N-1,M)

Wormhole Predictor. To track these particular cases, Albericio et al. propose the wormhole (WH) predictor. Similar to the loop predictor, WH is intended to be used as a side predictor. WH is a tagged structure with only a few entries (7 in the proposed design optimized for CBP4). For a branch

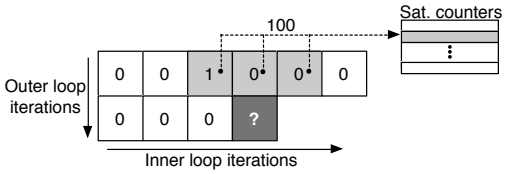

Fig. 2: Example of WH prediction.

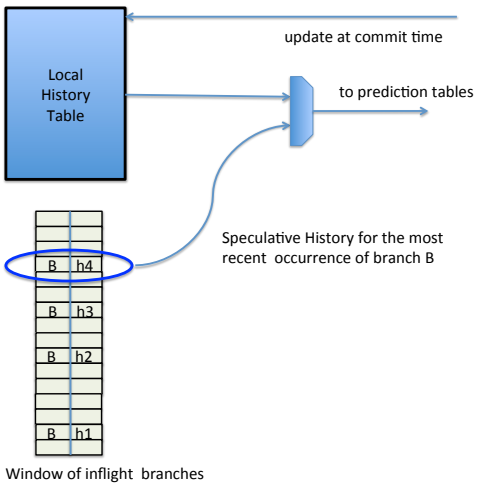

Fig. 3: Retrieving the speculative local history for branch B

B encapsulated in a regular loop IL (i.e., a loop predicted by the loop predictor with a constant number of iterations Ni), an entry is allocated in the WH predictor upon a misprediction. WH then records the local history of branch B. When B is fetched in iteration M of IL and in iteration N of OL, then Out[N-1][M+D] is recovered as bit Ni-D from its associated local history. Figure 2 illustrates the prediction process. WH embeds a small array of prediction counters in each entry. A few bits (grey squares in Figure 2) retrieved from the local history (as just described) are used to index this prediction array.

WH is the first predictor in the literature to track the outcome correlation of a branch encapsulated in a loop nest with occurrences of the same branch in neighboring inner loop iterations, but in the previous outer loop iteration. The number of dynamic instances of these branches can be very significant. When such correlation exists and is not captured by the main predictor, the accuracy benefit can be high.When associated with a state-of-the-art global history predictor, on average WH achieves accuracy improvement on the same range as local history components with a very limited number of entries [4].

Wormhole Limitations. The WH predictor exposes that there is an opportunity to exploit a new form of correlation in branch history. However, the original WH predictor has some limitations that could impair its practical implementation. First, WH only captures the behavior of branches encapsulated in loops with a constant number of iterations. It uses the loop predictor to recognize the loop and extract the number of iterations of the loop. For instance, WH is not able to track any branch if Mmax varies in the example illustrated on Figure 1. Second, the WH predictor captures correlations only for branches that are executed on each iteration of the loop. Branches in nested conditional statements (i.e., branch B4) are not addressed by the WH predictor. Lastly, WH uses very long local histories. The speculative management of these very long local histories is a major design challenge as detailed in the next section. The IMLI-based predictor components proposed in this paper address these shortcomings.

III. Speculative Local History

In order to compute the branch prediction, the predictor states are read at prediction time; they are updated later at commit time. On a wide superscalar core, this read-to-update delay varies from a few tens to several hundreds of cycles. In the meantime, several branch instructions, sometimes tens of branches, would have already been predicted using possibly irrelevant information (i.e., stale branch histories and predictor tables entries).

On the one hand, it is well known that the delayed update of prediction tables has limited prediction accuracy impact for state-of-the-art branch predictors [8], [9]. On the other hand, using incorrect histories leads to reading wrong entries in the predictor tables and is very likely to result in many branch mispredictions [10]. Therefore, accurately managing speculative branch histories is of prime importance. Below, we contrast the simple management of speculative global history with that of speculative local history.

Managing speculative local history is much more complex than managing speculative global history. On a processor with a large instruction window, distinct static branches can have speculative occurrences in-flight at the same time. In practice, speculative history can be handled as illustrated in Figure 3. The local history table is only updated at commit time. At prediction time of branch B, the local history table is read and the window of all speculatively in-flight branches is checked looking for occurrences of branch B (or more precisely of branches with the same index in the local history table). If any in-flight occurrence of branch B is detected, then the (speculative) local history associated with the most recent of these in-flight occurrences is used.

This necessitates an associative search in the window of in-flight branches. Local history must be stored with each in-flight branch in this window. On a misprediction of branch B, the branches fetched after B are flushed from the instruction window.

IV. IMLI

This section summarizes two IMLI-based components, which are alternative approaches to predicting the class of hard-to-predict branches encapsulated in two-dimensional loops identified by Albericio et al. [4], [5]. These components

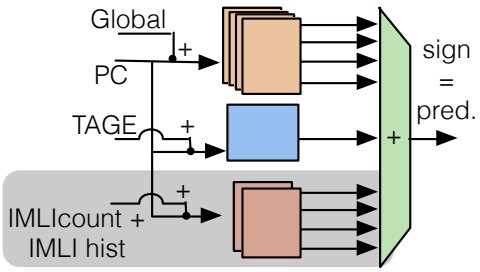

Fig. 4: The Statistical Corrector predictor for TAGE-GSC with IMLI-based components

can be incorporated into any of the two families of state-of-the-art branch predictors: the TAGE predictor family [6] and the neural-inspired predictor family [11], [12], [13], [7]. Figure 4 illustrates adding IMLI components to the statistical corrector in TAGE-GSC [14]. Both components exploit the Inner Most Loop Iteration (IMLI) counter, a simple mechanism that tracks the number of the current iteration in the inner most loop. The first component, IMLI-SIC (**S**ame **I**teration **C**orrelation), captures a completely different correlation than the WH predictor. The second component, IMLI-OH (**O**uter **H**istory), essentially captures the same correlation as the WH predictor. Throughout this section, we will use the following notation when discussing branches in multidimensional loops:

- B is a branch in inner loop IL encapsulated in outer loop OL.
- Out[N][M] is the outcome of branch B in iteration M of IL and in iteration N of OL.

A. Inner Most Loop Iteration Counter

In most cases, a loop body ends by a backward conditional branch. Therefore, for the sake of simplicity, we consider that any backward conditional branch is a loop exit branch. We will also consider that a loop is an inner most loop if its body does not contain any backward branch.

We define the Inner Most Loop Iteration counter, IMLIcount, as the number of times that the last encountered backward conditional branch has been consecutively taken. A simple heuristic allows us to track IMLIcount at fetch time for the inner most loop for any backward conditional branch:

```
if (backward){if (taken) IMLIcount++;
else IMLIcount=0;}
```

In practice, IMLIcount will be 1 or 0 on the first iteration depending on the construction of the multidimensional loop.

The IMLI counter can be used to produce the index of the two IMLI-based predictor components presented below.

B. IMLI-SIC

In some applications, a few hard-to-predict branches encapsulated in loops repeat or nearly repeat their behavior for the same iteration in the inner most loop (i.e., Out[N][M]≡Out[N-1][M]) in most cases. For instance, this occurs when the same expression dependent on the inner most iteration number is tested in the inner loop body. In the example in Figure 1, branches B3 and B4 represent this case.

To capture this behavior, we add a single table to the statistical corrector of TAGE-GSC. We will refer to this table as the IMLI-SIC (Same Iteration Correlation) table. IMLI-SIC is indexed with a hash of the IMLI counter and the PC. With a 512-entries table, we capture most of the potential benefit on this class of branches on our benchmark set. However, the benefit can be further increased by inserting the IMLI counter in the indices of two tables in the global history component of the SC.

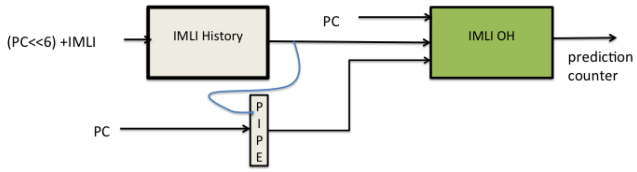

Fig. 5: The IMLI-OH component

C. IMLI-OH

From our experiments, we find that the IMLI-SIC component does not capture all correlations that are captured by the WH predictor. Specifically, when predicting Out[N][M] for a branch B, the outcomes Out[N-1][M-1] and Out[N-1][M] from the previous outer iteration also have to be memorized. In the WH predictor, these outcomes are memorized in the local history associated with branch B in the WH predictor entry. When predicting Out[N][M], these two outcomes are then retrieved as bits Mmax+1 and Mmax of the local history respectively, where Mmax is the number of iterations of the inner loop as predicted by the loop predictor.

The IMLI-OH (**O**uter **H**istory) predictor component, illustrated in Figure 5, is an alternative solution to track Out[N-1][M-1] and Out[N-1][M] for the inner branches in two-dimensional loops using the IMLI counter. It consists of the IMLI-OH predictor table, which is incorporated in the SC part of the TAGE-GSC predictor. It also consists of two structures to store and retrieve the history of the previous outer loop iteration: the IMLI history table and the PIPE vector, described below.

The outcome of branches are stored in the IMLI history table. We found that a 1 Kbit table is sufficient. The outcome of a branch at address B is stored at address (B*64) + IMLIcount. This allows us to recover Out[N-1][M] when predicting Out[N][M]. However, when predicting the next iteration (i.e., Out[N][M+1]), Out[N-1][M] would have already been overwritten with Out[N][M]. Therefore, the PIPE (**P**revious **I**nner iteration in **P**revious **E**xternal iteration) vector is used to intermediately store Out[N-1][M]. This vector only contains 16 bits, corresponding to the 16 distinct branch addresses that the 1K-entry IMLI outer history table is able to track.

The IMLI-OH predictor table is indexed with the PC hashed with bits Out[N-1][M] and Out[N-1][M-1] retrieved as described above. Using a 256-entry IMLI-OH predictor table was found to be sufficient to cover all the practical cases in our set of 80 traces.

D. Speculative Management of IMLI

After the fetch of a given instruction block, the new speculative IMLI counter is derived from the previous speculative IMLI counter as well as the presence/absence of any forward branches in the instruction fetch block and their predicted directions. Checkpointing the speculative IMLI counter allows for resuming branch prediction and instruction fetch with the correct IMLI counter after a branch misprediction.

For IMLI-OH, the IMLI PIPE vector is a small structure (16 bits in our study). It can be checkpointed for each instruction fetch block. In practice, precise management of the IMLI outer history is not required. Analysis of simulations shows that the IMLI-OH component essentially captures correlation for branches that are encapsulated in loops with a large number of iterations. In practice, for these branches, when iteration M of IL in iteration N of OL is fetched, the occurrences around iteration M of the inner most loop IL and iteration N-1 of the outer loop OL have been committed for a long time. For these branches, the correct Outer History is used. For the other branches that do not exhibit IMLI counter correlations, using the incorrect Outer History has very limited impact.

V. Methodology

Throughout this paper, trace-based simulations of the branch predictors are used in order to motivate and validate the proposed designs. Misprediction rates measured as Mispredictions Per Kilo Instructions (MPKI) will be used as a metric of accuracy.

Trace-based branch prediction simulations are assuming immediate updates of the prediction tables and branch histories. On real hardware, branch histories are speculatively updated ensuring that the same prediction tables entries are read at fetch time and updated at commit time. The prediction tables are updated at commit time; thus in a few cases, a prediction table entry is read at prediction time before a previous branch in the control flow commits and updates it. However, for the state-of-the-art global history predictors considered in this paper, the delayed update of predictor tables has very limited impact on accuracy [9], [8]. Moreover, this impact can be mitigated [9].

Application Traces. To allow reproducibility of the experiments presented in this paper, all the simulations are performed using the two sets of traces that were distributed for the two last branch prediction championships in 2011 (CBP3) and 2014 (CBP4). Each set of traces features 40 traces. Traces from CBP3 were transformed in order to be compatible with simulations through the CBP4 framework. These 80 traces cover various domains of applications including SPEC integer and floating-point applications, servers, client and multimedia applications.

Branch Predictors. The IMLI predictor components presented in this paper improve branch accuracy when combined with any of the two families of state-of-the-art branch predictors: the TAGE predictor family [6] and the neural-inspired predictor family [11], [12], [13], [7]. We consider one global history predictor from each of the two families as base references. As a representative of TAGE-based global history predictors, we use the TAGE-GSC predictor (i.e. the global history components of TAGE-SC-L [14], the winner of CBP4). As a representative of neural-inspired global history predictors, we use a GEHL predictor [7].

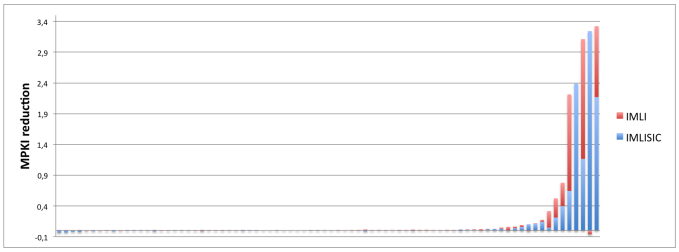

Fig. 6: IMLI-induced MPKI reduction on the 80 benchmarks; TAGE-GSC predictor

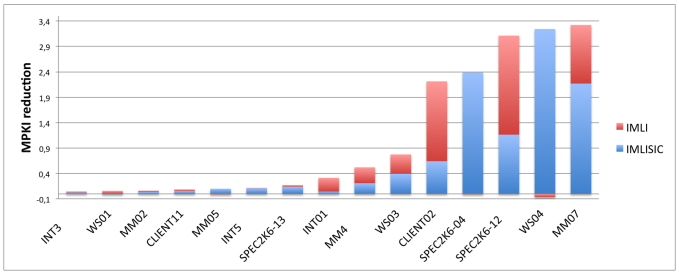

Fig. 7: IMLI-induced MPKI reduction on the 15 most benefitting benchmarks; TAGE-GSC predictor

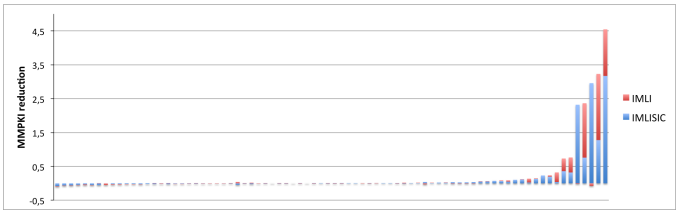

Fig. 8: IMLI-induced MPKI reduction on the 80 benchmarks; GEHL predictor

VI. Evaluation

Figures 6 and 7 illustrate the accuracy benefit obtained from augmenting TAGE-GSC with the two IMLI-based components on the whole set of 80 benchmarks and the 15 most improved benchmarks respectively. The benefit of IMLI-SIC alone is illustrated in the lowest bar.

IMLI-SIC. IMLI-SIC reduces the average misprediction rate from 2.473 to 2.373 MPKI for CBP4 and from 3.902 to 3.733 MPKI on CBP3 traces. This benefit is essentially obtained on a few benchmarks: two CBP4 benchmarks — SPEC2K6-04 (-2.37 MPKI) and SPEC2K6-12 (-1.16 MPKI) — and three CBP3 benchmarks — WS04 (-3.20 MPKI), MM07 (-2.17 MPKI and CLIENT02 (-0.64 MPKI). The accuracies of a few other benchmarks (MM4 and WS03) are marginally improved while the other benchmarks remain mostly unchanged.

The impact of adding the IMLI-SIC table to GEHL is very similar, reducing misprediction rate from 2.864 MPKI to 2.752

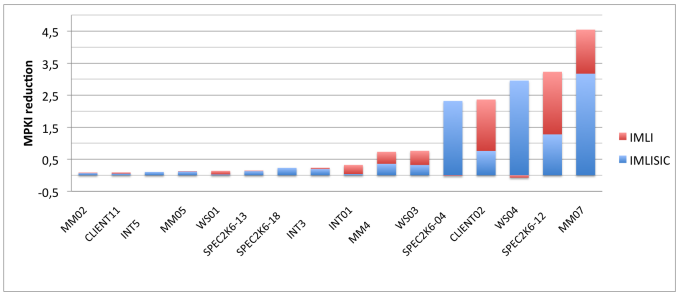

Fig. 9: IMLI-induced MPKI reduction on the 15 most benefitting benchmarks; GEHL predictor

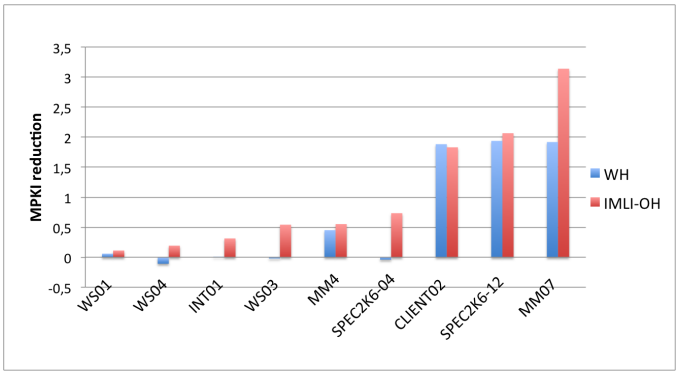

Fig. 10: IMLI-OH vs WH prediction accuracy on top of the GEHL predictor

MPKI for CBP4 traces and from 4.243 MPKI to 4.053MPKI for CBP3 traces. The same benchmarks as for TAGE-GSC are improved by IMLI-SIC. This is illustrated in Figures 8 and 9,

Interestingly, SPEC2K6-04 and WS04 are benchmarks that were not improved by the WH predictor. In practice, as already pointed out, the structure of the WH predictor only captures correlations for branches that are encapsulated in regular loops with constant iteration numbers and are executed on each iteration of the inner loop. IMLI-SIC does not suffer from these restrictions. On the other hand, benchmarks that are improved by WH — SPEC2K6-12, CLIENT02, MM07 and MM4 — are not as significantly improved by IMLI-SIC as with WH.

The IMLI-SIC table allows for predicting the number of iterations of the inner loop whenever the inner loop has a constant iteration number. As a result, activating the loop predictor when IMLI-SIC is enabled has limited impact. For instance, with TAGE-GSC, the benefit of the loop predictor is reduced from 0.034 MPKI to 0.013 MPKI on CBP4 and from 0.094 MPKI to 0.010 MPKI on CBP3.

IMLI-OH. First, we compare the benefits of IMLI-OH and WH when added to the base predictors. This is illustrated in Figure 10 for the GEHL predictor; results for TAGE-GSC are similar. As expected, the two predictors enhance the accuracy of the benchmarks that were enhanced by WH. IMLI-OH slightly enhances the accuracy of a few other benchmarks (e.g., SPECK6-04 and WS03) that are also enhanced by IMLI-SIC.

The benefits from IMLI-OH over the base predictors augmented with IMLI-SIC are proportionally smaller than the ones from IMLI-SIC alone: 2.0 % MPKI reduction on CBP4 traces (resp. 2.2 %) and 2.3 % (resp. 2.3 %) on CBP3 traces for TAGE-GSC (resp. GEHL).

IMLI Overall. The total benefit of IMLI-SIC and IMLI-OH is illustrated as the full bar in Figures 6 and 7 for TAGE-GSC and in Figures 8 and 9 for GEHL. These benefits are obtained only on a few benchmarks but are significant for these benchmarks. Note that the benefits of IMLI-OH and IMLI-SIC are not always cumulative, as illustrated by SPECK6-04.

For TAGE-GSC, the misprediction rate is improved by 6.8 % from 2.473 MPKI to 2.313 MPKI on CBP4 traces and by 6.1 % from 3.902 MPKI to 3.649 MPKI on CBP3 traces. For the GEHL predictor, the misprediction rate is improved by 6.0 % from 2.864 MPKI to 2.694 MPKI on CBP4 traces and 6.5 % from 3.902 MPKI to 3.649 MPKI on CBP3 traces. This misprediction reduction is most prominent for seven benchmarks: SPEC2K6-04, SPEC2K6-12 and MM-4 from CBP4 as well as CLIENT02, MM07, WS04 and WS03 from CBP3 (Figures 7 and 9). Most of the other benchmarks neither benefit nor suffer from the IMLI components as illustrated in Figures 6 and 8.

These two predictor components can be simply added as extra tables in the statistical corrector predictor of TAGE-GSC or in the GEHL predictor. The overall storage budget for implementing the two IMLI-based components is low: a total of 708 bytes (i.e., 384 bytes for the IMLI-SIC table, 128 bytes for the IMLI outer history table, 192 bytes the IMLI OH predictor table, 4 bytes for the PIPE vector and the IMLI counter). Moreover, managing the speculative states of IMLI-SIC and IMLI-OH is as simple as that of speculative global history; it can be implemented by checkpointing only two small structures: the IMLI counter (10 bits) and the IMLI PIPE vector (16 bits). Despite this low storage budget and hardware complexity, the IMLI-based components significantly reduce the misprediction rate for several benchmarks when added to TAGE-GSC and GEHL.

VII. Impact on Local History

Up to now, we have considered IMLI-based components for branch predictors featuring only global history components. State-of-the-art academic branch predictors feature both local and global history components, but most real hardware processors only use global history predictors. In this section, we show that the potential accuracy benefit from using local history is further limited when using IMLI-based components.

The two base predictors, TAGE-GSC and GEHL, can be augmented with local history components. These local history components can be inserted in the SC predictor of TAGE-GSC and can be added as a local history GEHL predictor in GEHL, which yields FTL [13]. We consider augmenting both predictors with a local history component. For TAGE-GSC, we activate the local history components and the loop predictor in TAGE-SC-L [14]. For GEHL, we add 1) 4 tables of 2K 6-bit counters and a 256-entry table of 24-bit local history counters, and 2) a 32-entry loop predictor, thus yielding a FTL predictor [13].

We run simulations selectively activating the different components: Base, Base+I (I for IMLI), Base+L (L for local) and

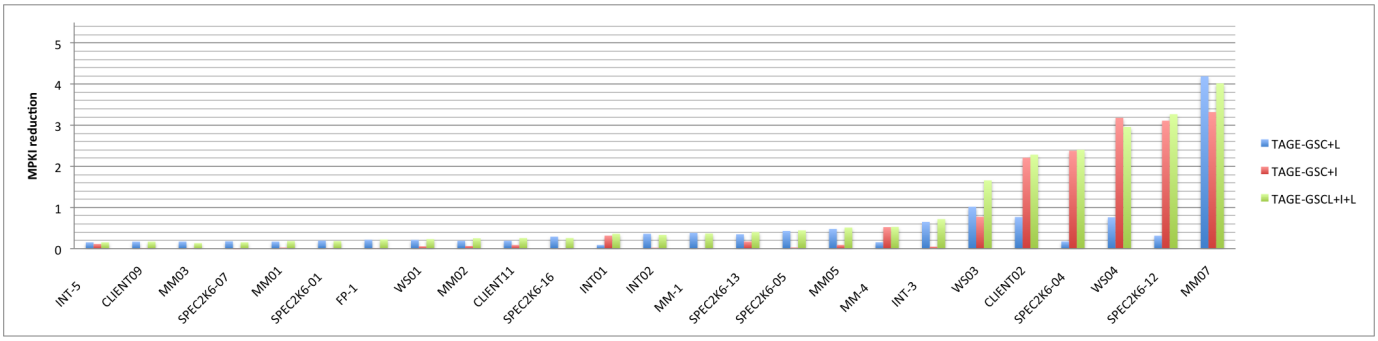

Fig. 11: Benefits of local history components on TAGE; the 25 most benefitting benchmarks

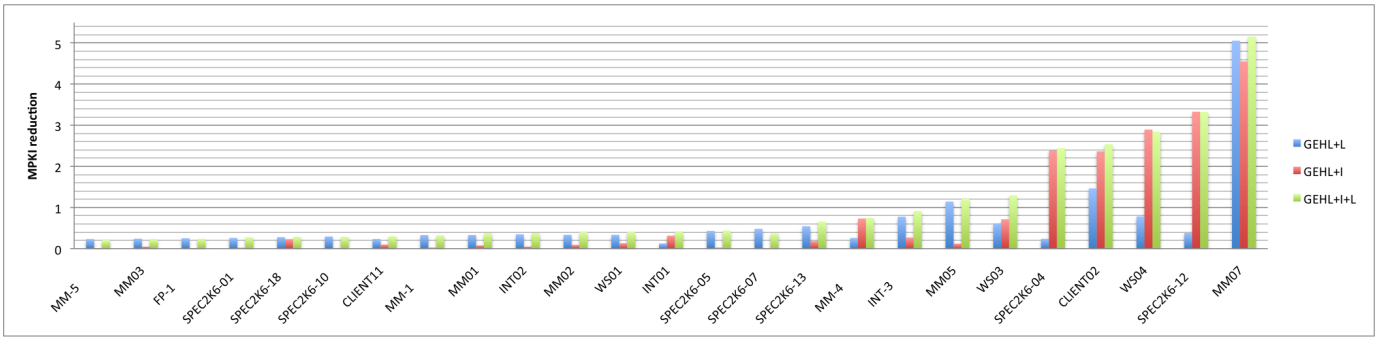

Fig. 12: Benefits of local history components on GEHL; the 25 most benefitting benchmarks

Base+I+L. The results for the 25 most affected benchmarks (out of 80) are illustrated in Figures 11 and 12. The average misprediction rates are reported in Tables I and II.

	TAGE-GSC	+L	+ I	+I + L
size (Kbits)	228	256	234	261
CBP4	2.473	2.365	2.313	2.226
CBP3	3.902	3.670	3.649	3.555

TABLE I: Average misprediction rate (MPKI) for TAGE-GSC-based predictors.

	GEHL	+L	+ I	+ I + L
size (Kbits)	204	256	209	261
CBP4	2.864	2.693	2.694	2.562
CBP3	4.243	3.924	3.958	3.827

TABLE II: Average misprediction rate (MPKI) for GEHL-based predictors.

Overall, adding the local history predictor components and the loop predictor to the IMLI-augmented base predictors leads to lower accuracy gains than adding them to the base predictors. For TAGE-GSC, the benefit shrinks from 0.108 MPKI without IMLI to 0.087 MPKI for CBP4 traces and from 0.232 MPKI to only 0.094 MPKI for CBP3 traces. For GEHL, we observe a very similar trend with an accuracy benefit of 0.132 MPKI instead of 0.171 MPKI on CBP4 traces and 0.131 MPKI instead of 0.319 MPKI on CBP3 traces.

The IMLI components capture part of the correlations that are captured by the local history components and the loop predictor. Figures 11 and 12 show this phenomenon. When IMLI components are very effective (i.e., MM-4, SPECK2-04, SPECK6-12, CLIENT02, WS04 and MM07), the local history components are often somewhat effective, (e.g., MM07, WS04, WS03 and CLIENT02). However, their impact is only partially cumulative. On the other hand, Figures 11 and 12 also show that the benefit of local history components is more evenly distributed on the overall set of benchmarks than that of the IMLI-based components.

Summary. The accuracy benefits of using local history components and a loop predictor on top of a predictor implementing global history and IMLI-based components is limited. These reduced benefits further argue against the cost-effectiveness of local history predictor components when the predictor already features IMLI-based components.

VIII. Acknowledgements

This work was partially supported by the European Research Council Advanced Grant DAL No 267175. This work is also supported by a Queen Elizabeth II/Montrose Werry Scholarship in Science and Technology, Bell Graduate Scholarship, a Discovery grant, and a Strategic grant from the Natural Sciences and Engineering Research Council of Canada.

References

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

[2] T. Sherwood and B. Calder, "Loop termination prediction," in *High Performance Computing, Third International Symposium, ISHPC 2000, Tokyo, Japan, October 16-18, 2000. Proceedings*, pp. 73–87, 2000.
[3] D. Morris, M. Poplingher, T. Yeh, M. Corwin, and W. Chen, "Method and apparatus for predicting loop exit branches," June 27 2002. US Patent App. 09/169,866.
[4] J. Albericio, J. San Miguel, N. Enright Jerger, and A. Moshovos, "Wormhole: Wisely predicting multidimensional branches," in *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-47, (Washington, DC, USA), pp. 509–520, IEEE Computer Society, 2014.
[5] J. Albericio, J. San Miguel, N. Enright Jerger, and A. Moshovos, "Wormhole branch prediction using multidimensional histories," in *Proceedings of the 4th Championship on Branch Prediction, http://www.jilp.org/cbp2014/*, 2014.
[6] A. Seznec and P. Michaud, "A case for (partially)-tagged geometric history length predictors," *Journal of Instruction Level Parallelism (http://www.jilp.org/vol8)*, April 2006.
[7] A. Seznec, "Analysis of the O-GEHL branch predictor," in *Proceedings of the 32nd Annual International Symposium on Computer Architecture*, june 2005.
[8] D. Jiménez, "Reconsidering complex branch predictors," in *Proceedings of the 9th International Symposium on High Performance Computer Architecture*, 2003.
[9] A. Seznec, "A new case for the tage branch predictor," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-44, (New York, NY, USA), pp. 117–127, ACM, 2011.
[10] E. Hao, P.-Y. Chang, and Y. N. Patt, "The effect of speculatively updating branch history on branch prediction accuracy, revisited," in *Proceedings of the 27th Annual International Symposium on Microarchitecture*, (San Jose, California), 1994.
[11] D. Jiménez and C. Lin, "Dynamic branch prediction with perceptrons," in *Proceedings of the Seventh International Symposium on High Performance Computer Architecture*, 2001.
[12] D. Jimenéz and C. Lin, "Neural methods for dynamic branch prediction," *ACM Transactions on Computer Systems*, vol. 20, Nov. 2002.
[13] Y. Ishii, "Fused two-level branch prediction with ahead calculation," *Journal of Instruction Level Parallelism (http://wwwjilp.org/vol9)*, May 2007.
[14] A. Seznec, "Tage-sc-l branch predictors," in *Proceedings of the 4th Championship on Branch Prediction, http://www.jilp.org/cbp2014/*, 2014.

Dr André Seznec is a senior research director at Inria in Rennes. He has been working on computer architecture for more than 30 years. His main research interests are speculative execution, pipeline design, and memory hierarchy and system. Dr Seznec is an IEEE fellow.

Joshua San Miguel is a PhD candidate at the University of Toronto. His research spans a wide range of computer architecture topics including branch prediction, approximate computing and networks-on-chip.

Dr Jorge Albericio is a Postdoctoral Fellow at the University of Toronto. He completed his PhD at the University of Zaragoza. His research interests include branch prediction, architectures for machine intelligence algorithms, memory hierarchy, and approximate computing.