Value Speculation through Equality Prediction

Kleovoulos Kalaitzidis, André Seznec

Value Speculation through Equality Prediction

Kleovoulos Kalaitzidis
Univ Rennes, Inria, CNRS, IRISA
Campus de Beaulieu, 35042 Rennes, France
kleovoulos.kalaitzidis@inria.fr

André Seznec
Univ Rennes, Inria, CNRS, IRISA
Campus de Beaulieu, 35042 Rennes, France
andre.seznec@inria.fr

***Abstract*—Modern context-based value predictors tightly associate recurring values with instructions and contexts by building confidence upon them. However, when execution monotony exists in the form of intervals, the potential prediction coverage is limited, since prediction confidence is reset at the beginning of each new interval. In this paper, we address this challenge by introducing the notion of *Equality Prediction* (EP), which represents the binary facet of value prediction. Following a twofold decision scheme (similar to branch prediction), EP makes use of control-flow history to determine equality between the last committed result read at fetch time, and the result of the fetched occurrence. When equality is predicted with high confidence, the read value is used. Our experiments show that this technique obtains the same level of performance as previously proposed state-of-the-art context-based predictors. However, by virtue of better exploiting patterns of interval equality, our design complements the established way that value prediction is performed, and when combined with contemporary prediction models, improves the delivered speedup by 19% on average.**

***Keywords*—Microarchitecture, Processor, Speculative Execution, Value Prediction, Equality Prediction.**

I. Introduction & Motivations

Context-based value predictors [1] aim to capture the repetition of the same value in the results of a static instruction by using a specific context. For instance, the state-of-the-art value predictor VTAGE [2] leverages the execution's global branch history, i.e., the same value encountered with the same global branch history. In this study, we aim to capture another form of value regularity: equal values on consecutive occurrences of the same static instruction. Two different kinds of *value equality* can be discriminated:

1. *Uniform*: All dynamic instances of a static instruction always produce the same result.
2. *Interval-style*: Within an interval of repetitive executions, a static instruction produces the same result. But for each interval the result is different.

Equality of the results of two consecutive occurrences of a static instruction is very frequent. We illustrate this event in Figure 1, characterizing *uniform* and *interval* equality of our benchmarks (discussed in Section IV). As reported, interval equality is prevalent across the examined applications, representing on average more than 18% of the instructions. However, to our knowledge, no previous context-based value prediction scheme has been designed with total awareness of interval equality, but rather of uniform equality.

To allow performance gains through value prediction (VP), reaching high confidence value generally necessitates many successive correct predictions (typically more than 100) [2]. On current context-based predictors [2], [3], the prediction confidence is associated with the value to be predicted in the same predictor entry. Therefore, on instructions exhibiting interval equality, prediction reaches high confidence only on long intervals.

In this paper, we introduce a practical solution to extend predictability of instructions that exhibit interval equality, called *Value Speculation through Equality Prediction* (VSEP). In VSEP predictor, we do not associate the confidence with the value in the same predictor entry, but rather predict if the result of the occurrence of an instruction Z is very likely to be equal to the last committed value of instruction Z. When equality exists, value prediction takes place by using the last committed value of the instruction. As a consequence, when VSEP is able to identify the beginning of intervals, it may resume useful (i.e. high confidence) predictions as soon as it can anticipate with high confidence that the first instruction occurrence in the interval has been committed.

Our simulations show that performance improvements of VSEP are in the same range with the state-of-the-art VTAGE, but their prediction coverage is partially orthogonal. This paper describes the microarchitecture details of VSEP and presents a comprehensive performance analysis.

II. related work

Context-based value predictors include the Last Value Predictor [3] indexed by the the program counter and the FCM predictor indexed by a hash of the last values [1]. More recently, VTAGE [2] was introduced. VTAGE leverages global branch history as a context and does not depend on any value history, thus it does not require complex integration in the OoO core. In the same work, Perais and Seznec pointed out that in-order *Validation* at commit coupled with a full pipeline flush on a misprediction is cost-effective provided that predictions are only used with very high confidence. To that end, they proposed the use of forward probabilistic counters (FPC) [4] to track the confidence of value-entries, achieving more than 99% accuracy. Furthermore, Sheikh et. al considered only load instructions by introducing DLVP [5], which forms a different approach of VP through memory address prediction. Finally, Sheikh and Hower presented a composite predictor [6] that combines different predictor models in order to increase coverage in the cost of some additional complexity imposed on the prediction critical path.

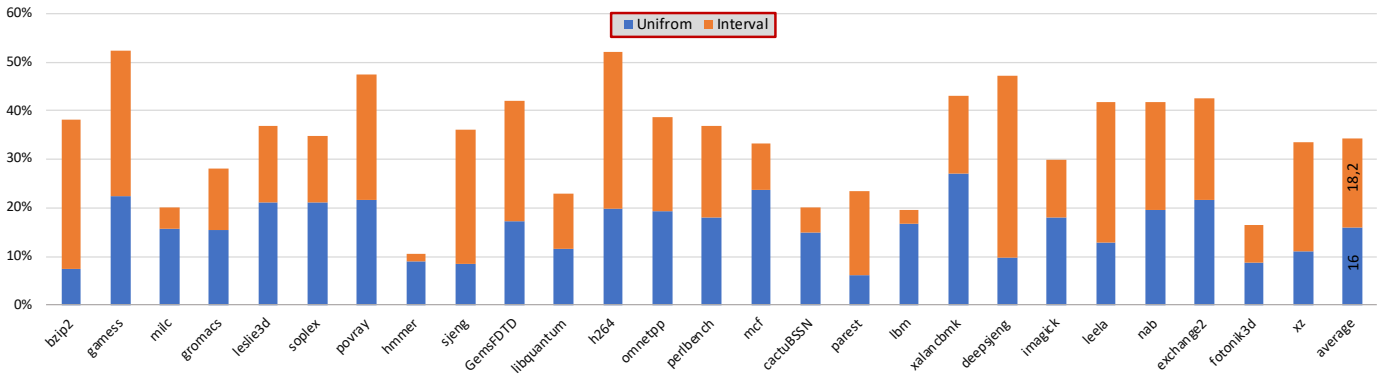

Fig. 1: Breakdown of value equality.

III. VSEP: Value Speculation via Equality Prediction

VSEP consists in two distinct components, ETAGE, the equality predictor, and LCVT, the Last Committed Value Table.

ETAGE is a context-based equality predictor that essentially copies the TAGE branch predictor structure. The prediction of ETAGE is a 1-bit value that defines *equality* or *inequality* between the value to be produced by the current instance of a static instruction and the last committed value of the same instruction. ETAGE employs a series of prediction tables that are indexed by a different number of bits of the global branch history, hashed with the instruction PC. Its main tables are also backed up by a tag-less base table that is directly indexed with the instruction PC. Entries in the tagged tables contain the 1-bit equality information, a partial tag, a 1-bit usefulness indicator and a 3-bit confidence counter. To guarantee high prediction accuracy, we use 3-bit FPC [4]. Experimentally, we found that using the probability vector V = $\{1, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}\}$ to control forward transitions is a good trade-off.

LCVT records the last committed value of instructions. The LCVT that we employ is a 3-way set associative table of no more that 3K entries, since we found that this size is sufficient to track execution. Along with the full 64-bit committed value, each entry includes a 13-bit tag, which is a folded hash of the instruction PC.

1) Prediction: ETAGE delivers a prediction by matching with an entry accessed with the longest history. If no matching entry is found, the base table provides the prediction. In parallel, the LCVT is accessed to provide the last committed value of the relevant instruction. On a miss in the LCVT, inequality is assumed. Note that the predicted value, if any, is used only if equality is predicted with high confidence.

2) Training: At update time, the entry of ETAGE that provided the equality prediction is updated with the resolved equality between the instruction's result and the speculative value retrieved from the LCVT. Their agreement triggers the increment of the entry's confidence counter, according to the probability vector described above. Otherwise, the entry's equality bit is replaced if its confidence was already equal to zero, and the confidence counter is reset. Moreover, a new entry is allocated following a similar allocation policy with TAGE [7]. In addition, the usefulness bit that leads the replacement policy is also handled as in [7]. Finally, the LCVT is updated with the committed result.

3) Pipeline details: VSEP is implemented in a pipeline where validations are done in-order at commit time, since this allows reduced OoO-core complexity by banking the physical register file [2]. One FIFO queue with size equal to that of the instruction window is employed. Each entry stores the 1-bit equality prediction of ETAGE and the value retrieved from the LCVT at prediction time. We assume the following process:

- At Fetch, leverage the ETAGE predictor to generate a high accuracy equality prediction and index the LCVT to acquire the instruction's last committed value. Also, place both the equality prediction and the potentially predicted value in the validation queue.
- At Rename, if *equality* is predicted with high confidence, and a hit on the LCVT is encountered, the predicted value is written to the physical register file. When *inequality* is predicted or *equality* does not have sufficiently high confidence, the predicted value is not used.
- At execution, overwrite the predicted value with the computed one.
- Before Commit, use the validation queue to validate the predicted value against the computed result. Flush the pipeline on a misprediction if the predicted value was inserted in the pipeline. Similarly, validate the equality prediction and update ETAGE predictor adequately. Also, update the LCVT with the just committed result.

IV. Evaluation

The framework that we use in our experiments is a modified[1] version of the *gem5* cycle-level simulator [10] implementing the x86_64 ISA. We consider a quite aggressive 4GHz, 8-wide issue superscalar baseline with a fetch-to-commit latency of 19 cycles (20 when VP is used due to an additional validation stage). The first half of Table I describes the characteristics of the baseline pipeline structure we use in more detail.

[1]Branches are implemented in a single µ-op instead of three.

TABLE I: Simulator configuration overview. *not pipelined.

Front-End	**Fetch through Rename width**: 8 insts/cycle **L1I:** 4-way, 32KB, 1 cycle, 128-entry ITLB, 64B fetch buffer **Branch Pred.:**TAGE-SC-L [8] 64KB, 2-way 8K-entry BTB, 32-entry RAS, 20 cycles min. mis penalty.
Execution	256-entry ROB, 128-entry unified IQ, 48/48-entry LQ/SQ (STLF lat. 4 cycles), 256/256 INT/FP pregs (4-bank PRF), 1K-SSID/LFST Store Sets [9] not rolled-back on squash and cleared every 30K access, **8-issue**, 4ALU(1c), 1Mul/Div(3c/25c*), 2FP(3c), 2FP-MulDiv(5c/10c*), 2Ld/Str Ports, 1Str Port, Full bypass, **8-wide Retire/Validation**
Memory Hierarchy	**L1D**: 4-way, 32KB, 4 cycles load-to-use, 64 MSHRs, 2 reads & 2 writes/cycle, 64-entry DTLB **L2**: Unified private, 8-way 256KB, 11 cycles, 64 MSHRs, no port constraints, Stride Prefetcher, (deg. 1) **L3**: Unified shared, 16-way 2MB, 34 cycles, 64 MSHRs, no port constraints, Stride Prefetcher, (deg. 1) All caches have 64B lines and LRU Replacement Policy **Memory**: Dual Channel DDR4-2400 (17-17-17) 2 ranks, 8banks/rank, 8K row-buffer, tREFI 7.8us, Across a 64B bus Min Read lat.: 75 cycles, Max.: 185 cycles.
LVP [3]	4K-entry LVT, 13bit-tags, *40KB*
VTAGE [2]	4K-entry Base Component, *33,5KB* 6 x 512-entry tagged tables, tags 12+rank bits, *30,5KB*
VSEP	**ETAGE**: 8K-entry Base Component, *4 KB* 13 x 1024-entry tagged tables, tags 8+rank bits, *32KB* **LCVT**: 3K entries, 3-way associative, 13bit-tags, *28KB*
Hybrid VSEP-VTAGE	**ETAGE**: 8K-entry Base Component, *4 KB* 13 x 512-entry tagged tables, tags 8+rank bits, *16KB* **LCVT**: 3K entries, 3-way associative, 13bit-tags, *28KB* **VTAGE Tagged tables**: 6 x 256-entry, tags 12+rank bits, *15KB*

TABLE II: Applications used in our evaluation.

Benchmark Suite	Applications
SPEC2K6 INT/FP	bzip2, gamess, milc, gromacs, leslie3d, soplex, povray, hmmer, sjeng, GemsFDTD, libquantum, h264, omnetpp
SPEC2K17 INT/FP Rate	perlbench, mcf, cactuBSSN, parest, lbm, xalancbmk, deepsjeng, imagick, leela, nab, exchange2, fotonik3d, xz

(a) Performance improvement

(b) Average prediction coverage of uniform and interval equality

Fig. 2: Comparison of the three prediction schemes: LVP, VTAGE and VSEP.

We cast a wide range of workloads from SPEC2K6 [11] and SPEC2K17 [12] to expose as many value equality patterns as possible. All benchmarks are statically compiled to target x86_64 architecture by using GCC compiler version 5.5 or newer with -O3 optimization enabled. To get relevant numbers, we identify a region of interest in the benchmarks using Simpoints 3.2 [13], as VP is highly sensitive to phase behavior. We simulate the resulting slice of 150M instructions by warming up the processor (caches, predictors) for 50M instructions and then collecting statistics for 100M instructions.

We evaluate VSEP by comparing it with two context-based value predictors, namely, the classic LVP and the state-of-the-art VTAGE predictor. The second half of Table I summarizes the structural details of all the predictors examined in this study. Due to space constraints, we present results only for benchmarks that benefit from any of the the VP models we examine, but we consider all the applications when we report average results (geometric mean for the relative speedup).

A. Performance of VSEP vs VTAGE and LVP

Figure 2 reports our simulation results by comparing the three different value predictors that we consider. In particular, Figure 2a shows the relative IPC of the three variants normalized to the baseline and Figure 2b shows the fraction of predicted instructions from those exposing either uniform or interval value equality. That is, *uniform coverage* describes the portion of instructions that exhibit uniform equality and their result was predicted, while *interval coverage* similarly expresses the predicted portion of the instructions that exhibit interval equality. Note that a prediction is used in the pipeline only in high confidence.

Essentially, VSEP benefits the same benchmarks with respect to typical context-based predictors and heavily competes with the state-of-the-art VTAGE. It delivers up to 45% of speedup (on *imagick*) and an average speedup of 5.8%. Also, VSEP succeeds in always obtaining equal or higher performance to the fundamental LVP.

In terms of coverage, VSEP and VTAGE achieve similar uniform coverage of around 54% and higher than 48% of LVP. Since LVP does not leverage control-flow history to distinguish instructions, it can not capture same levels of uniform equality due to higher address aliasing among its entries. Concerning interval coverage, VSEP successfully surpasses both LVP and VTAGE, achieving 50%, versus 35% for VTAGE and 29% for LVP. As expected, LVP is the least efficient to capture value equality in intervals. It mispredicts at the beginning of each interval (triggering a pipeline flush) and then it needs to re-train its identified value-entry before re-reaching high confidence and becoming used in the pipeline again.

Evidently, VSEP is able to cover cases of interval equality that both LVP and VTAGE miss. Practically, if we assume an instruction that exhibits interval equality, after a value switch from X to Y, VSEP can predict equality with high confidence

as soon as the first occurrence of the new value Y has been committed. On the other hand, VTAGE has to reconstruct high confidence for each of the entries that has been predicting X. Therefore, on several benchmarks, VSEP outperforms VTAGE e.g on *leslie3d* and on *GemsFDTD*. Nonetheless, VTAGE can also predict independent value patterns, e.g. strides, and therefore evenly outperforms VSEP on some other applications, e.g. on *h264*, and *xz*.

Readers will notice that speedup of VTAGE is lower than that of LVP for two applications, namely for *leslie3d* and for *GemsFDTD*. After further exploration we found that the potential performance gain of VTAGE for these benchmarks is limited by bursts of mispredictions associated with the interval transitions. When applications expose very long value equality intervals, VTAGE may establish the confidence of the same recurrent value in several of its tagged-entries. Hence, upon an interval step, a burst of mispredictions will occur when these entries will be successively hit.[2] Contrary, VSEP does not suffer from this phenomenon as it separates the decision for performing VP (i.e. equality) from the predicted value.

B. Combining VSEP with VTAGE

Our experimental analysis verifies that VSEP accounts for meaningful gains, either comparable or higher to an established value predictor as VTAGE, by eliminating the essential weakness of conventional VP methods to make use of sporadic value equality. Although both VSEP and VTAGE can individually obtain noticeable speedups, none of the two can plainly outperform the other, as they can capture different cases. Consequently, since the two methods can be considered as partially orthogonal, we study the combination of VSEP with a short-scale adaptation of VTAGE.

Hybrid *VSEP-VTAGE* employs an ETAGE equality predictor accompanied by a LCVT, and a moderately-sized VTAGE, which does not encompass its base component (i.e., a LVT). The model works as follows: both predictor components are indexed in parallel in the early pipeline front-end. No specific policy is necessary to clarify which predictor makes the final prediction. Simply, if *value equality* is predicted by ETAGE, the prediction of VSEP is the one that proceeds, otherwise, the prediction of VTAGE is considered for use to cover the cases missed from VSEP. For validation/update, each entry of the FIFO queue in VSEP (as described in Section III) is augmented with the predicted value of VTAGE.

In Figure 3 we present the performance gain of the compound model by comparing it with VTAGE and VSEP alone at similar storage budgets. Evidently, our hybrid solution can efficiently combine the two schemes by retaining their independent benefits. For any benchmark individually, the speedup is in the range of the highest between VSEP and VTAGE, and on average is augmented by 19% from VSEP/VTAGE, obtaining an overall average of 7.1%.

[2]Findings from conducted experiments that confirm this behavior are not included due to space constraints.

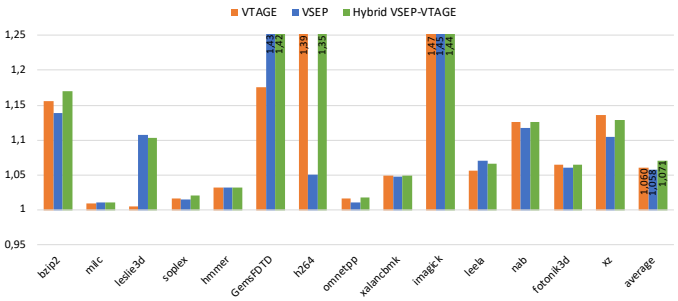

Fig. 3: Performance improvement of the compound model.

V. Conclusion

Context-based value predictors represent an important class of value predictors [1]. They exploit the recurrent occurrences of the same result by a single instruction. Some instructions deliver always the same result while others produce the same result on intervals. Overall, these instructions represent the most significant part of the predictions covered by a state-of-the-art context-based predictor, such as VTAGE. The VSEP predictor presented in this paper was introduced to specifically target interval-style value equality, while certainly covering the uniform category as well. Our conducted experiments reveal that VSEP is a technique that enhances performance gains of modern context-based VP schemes and paves the way for the development of novel approaches to efficiently extend the prediction range of established VP.

Acknowledgment

This research was partially supported by an Intel research grant.

References

[1] Y. Sazeides and J. E. Smith, "The predictability of data values," in *Proc. 30th Annual Int. Symp. Microarchitecture*, pp. 248–258, Dec. 1997.
[2] A. Perais and A. Seznec, "Practical data value speculation for future high-end processors," in *HPCA*, 2014.
[3] M. H. Lipasti and J. P. Shen, "Exceeding the dataflow limit via value prediction," in *MICRO 29*, 1996.
[4] N. Riley and C. Zilles, "Probabilistic counter updates for predictor hysteresis and stratification," in *HPCA-12*, 2006.
[5] R. Sheikh, H. W. Cain, and R. Damodaran, "Load value prediction via path-based address prediction: Avoiding mispredictions due to conflicting stores," in *MICRO-50*, 2017.
[6] R. Sheikh and D. Hower, "Efficient load value prediction using multiple predictors and filters," in *HPCA*, 2019.
[7] A. Seznec and P. Michaud, "A case for (partially) tagged geometric history length branch prediction," *Journal of Instruction Level Parallelism*, vol. 8, pp. 1–23, 2006.
[8] A. Seznec, "TAGE-SC-L Branch Predictors Again," *in JWAC-5: Championship on Branch Prediction, https://www.jilp.org/cbp2016/*, 2016.
[9] G. Z. Chrysos and J. S. Emer, "Memory dependence prediction using store sets," in *Computer Architecture News*, vol. 26, 1998.
[10] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, *et al.*, "The gem5 simulator," *Computer Architecture News*, vol. 39, no. 2, 2011.
[11] Standard Performance Evaluation Corporation, "The SPEC CPU 2006 Benchmark Suite," *in http://www.spec.org*.
[12] Standard Performance Evaluation Corporation, "The SPEC CPU 2017 Benchmark Suite," *in http://www.spec.org*.
[13] E. Perelman, G. Hamerly, M. Van Biesbrouck, T. Sherwood, and B. Calder, "Using simpoint for accurate and efficient simulation," *SIGMETRICS Perform. Eval. Rev.*, vol. 31, no. 1, 2003.