Combining Branch History and Value History For Improved Value Prediction

Chirag Sakhuja, Anjana Subramanian, Pawanbalakri Joshi, Akanksha Jain, Calvin Lin
The University of Texas at Austin

ABSTRACT

Context-based value predictors use either control-flow context or data context to predict values. Those based on control-flow context use branch histories to remember past values, but these predictors require lengthy histories or complex hardware to make accurate predictions. Those based on data context use a history of past values to predict a broader class of values, but such predictors require long training times, and they are complex due to speculative value histories.

We observe that the combination of branch and value history provides better predictability than the use of each history separately because the combination can predict values in control-dependent sequences of values. Furthermore, the combination improves training time by enabling accurate predictions to be made with shorter history, and it simplifies hardware design by reducing aliasing in back-to-back instructions, thus eliminating the need for speculative value histories. Based on these observations, we propose a new unlimited budget value predictor, Heterogeneous-Context Value Predictor (HCVP), that when hybridized with E-Stride, achieves a geometric mean IPC of 3.88 (39.6% speedup over baseline) on the 135 public traces, as compared to 3.81 (37.2% speedup over baseline) for the current leader of the Championship Value Prediction.

1. INTRODUCTION

Value prediction is an important means of improving single-thread performance: The basic idea is to break data dependences by predicting values rather than waiting for them to be computed. The recent resurgence in value predictor research stems from Perais' idea of performing misprediction recovery at commit time, which greatly simplifies the logic [1]. This delayed recovery, however, increases the penalty of mispredictions, so modern value predictors require extremely high accuracy, typically over 99%. Thus, for value prediction, the research challenge is to increase predictor coverage while maintaining high accuracy.

Finite Context Method (FCM) based predictors [2], including the DFCM++ predictor [3], use value history to predict correlated values, namely, sequences of values that are not constant or strided but tend to repeat over time. For example, if a linked list traversal repeatedly produces the sequence of values A, B, C, X, then a predictor can learn this sequence and predict that the value X will be produced after it has produced A, B, and C.

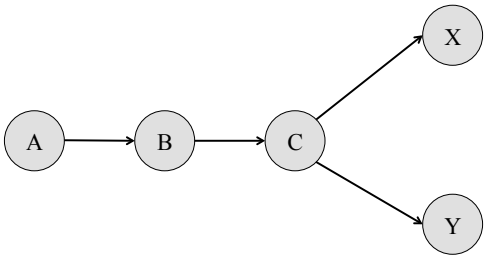


Figure 1: Value history based predictors struggle with control-dependent sequences.

Unfortunately, the use of value history is problematic when the sequence of values is control-dependent. For example, a graph traversal might produce the value sequence shown in Figure 1, where the successor of C depends on a branch outcome. In such scenarios, value histories yield inaccurate predictions, which degrades performance. To improve accuracy, existing solutions [3] use extremely long value histories, hoping that some past value produced by the same instruction correlates with the next value. Long value histories increase hardware overhead, lead to long training times, and are not always successful in learning such divergent patterns.

By contrast, context-based predictors that use control flow information can also achieve high accuracy [4]. Perais uses branch history, in particular, to predict control-dependent values [1]. However, lengthy histories are necessary to capture data-dependent values. For example, to accurately predict the values in a linked list, the branch history must be as long as the linked list.

This paper builds on the insight that control-dependent sequences can be predicted by augmenting value history with control flow information—in particular, the branch history. Thus, our solution decomposes an instruction's value sequence into sub-sequences, such that each subsequence shares the same branch history and is more predictable than the combined sequence. For example, the sequence in Figure 1 can be viewed as two subsequences: (1) A, B, C, X with a branch history of (T,

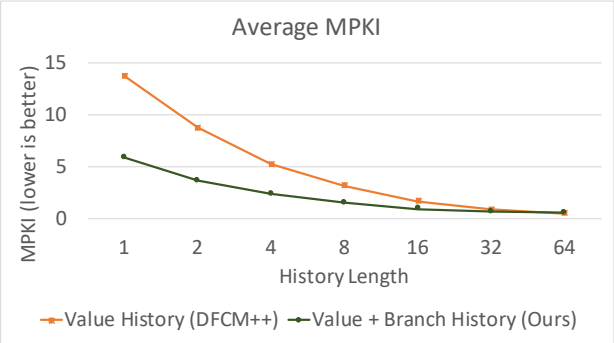


Figure 2: The combination of value history and branch history enables the use of shorter value histories.

T, T, T), and (2) A, B, C, Y with a branch history of (T, T, T, NT).

The combination of value history and branch history has two added benefits. First, it requires a much shorter value history than previous solutions. Figure 2 compares the accuracy of DFCM++, a state-of-the-art FCM predictor, with our proposed predictor at different history lengths and shows that the combination of branch history and value history achieves acceptable misprediction rates (less than 2 MPKI) at history lengths of 8 and 16, whereas value history alone necessitates the use of history lengths of 32 or longer [3]. Second, it does not require a speculative history of values for in-flight instructions, a complication that has plagued previous value-history-based predictors. As we explain in Section 2.2, by using both the PC and branch history as context, our predictor reduces the odds that one of the sequence elements is in-flight, which allows our predictor to simply ignore such scenarios when they occur.

This paper makes the following contributions.

- We introduce the Heterogeneous-Context Value Predictor (HCVP) value predictor, which combines branch history with a short value history to provide a better tradeoff between coverage and accuracy than the use of local value history alone.
- We show that our predictor does not require speculative value histories, leading to a simpler design for value history based predictors.
- Using the Championship Value Predictor infrastructure, we show that our new value predictor provides better speedup than the previous state-of-the-art. In the unlimited budget category, HCVP outperforms STEVES, the current championship winner by 2.4% (39.6% for our predictor vs 37.2% for STEVES).

2. SOLUTION

Heterogeneous-Context Value Prediction (HCVP) is a context-based predictor that predicts sequences of correlated values. As explained in Section 1, HCVP learns value sequences that share the same load instruction and branch history. For example, for the control-dependent value sequence shown in Figure 1, where C is sometimes followed by X and sometimes by Y, HCVP learns two separate sequences, namely, A, B, C, X and A, B, C, Y.

Additionally, for better predictability, instead of learning correlations between consecutive values, HCVP learns correlations among deltas of consecutive values, where the deltas can be arbitrarily large[1]. The goal is to be able to learn repeating irregular stride sequences, such as, $-1, -2, +3$, and apply them across different base values for higher coverage. Another benefit of separating value history into a base and delta history is that HCVP can quickly learn constant and strided values. For example, constants are represented as a base value equal to the constant value and a delta history of $0, 0, 0, ...$

2.1 High-Level Design

Figure 3 shows HCVP's high-level organization. We see that HCVP has two components. The first component consists of two tables—the Base History Table (BHT) and Stride History Table (SHT)—which track the value history for each program counter and global branch history (PC, BHR) context. In particular, the BHT stores the last seen value for a given (PC, BHR) context, and the SHT stores the sequence of k deltas for this context (k is the value history length). Both tables are indexed using a combination of the program counter (PC) and a global branch history register (BHR). The second component, which includes the Value Prediction Table (VPT), learns next values for a given sequence of values.

Each table entry is augmented with a 4-bit confidence counter. An entry is a candidate for replacement when its confidence counter is 0.

Prediction.

To form a prediction, HCVP first probes the Base History Table (BHT) and Stride History Table with the current (PC, BHR) context, and if both tables have entries, the stride history from the SHT is used to access the Value Prediction Table (VPT). If the confidence associated with the corresponding entry in the VPT is sufficiently high, 10 in our case, HCVP will predict the value by adding the output of the VPT to the base value from the BHT. For example, if the BHT produces a base value of A, and the VPT predicts a stride of $+48$, the next predicted value will be $A + 48$.

If an instruction with the same (PC, BHR) is currently in-flight, no prediction is made.

Training.

On a correct prediction, the confidence of the corresponding entries in all three tables is incremented, the BHT is updated with the current value, and the SHT is updated with the delta between the current value and the base value that was used for prediction. For the above example, the BHT will be updated to carry a base

[1] This idea of separating values into base and deltas has been applied to prior FCM-based predictors and is called the Differential Finite Context Method (DFCM) [5].

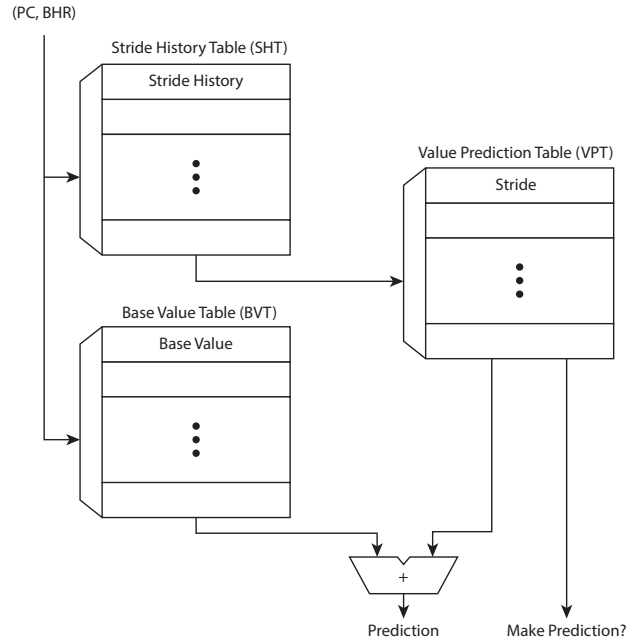


Figure 3: High-level organization of HCVP.

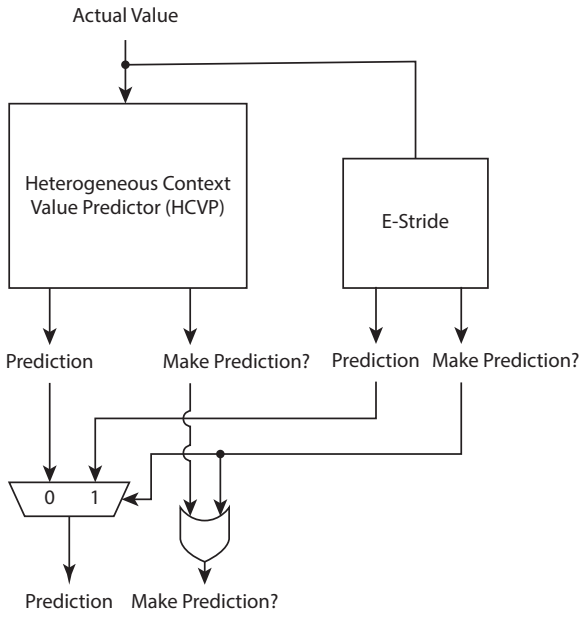


Figure 4: Hybridizing HCVP with E-Stride.

value of $A + 48$, and the SHT entry will be augmented with a stride of +48. On an incorrect prediction, the confidence for corresponding entries in all three tables is set to 0.

If a miss in any of the three tables inhibits a prediction, then the tables are initialized with appropriate information.

2.2 The Update Problem

The update problem occurs when table entries become stale when making back-to-back predictions, often in tight loops. Fundamentally, the update problem is caused by having in-flight requests to the same entry in the table, leading to out-dated branch or value history when a prediction must be made.

Our solution is to incorporate branch history. Since branch history changes on each iteration of a tight loop, repeated instances of instructions access different entries of the table, greatly reducing the likelihood of the update problem. We find that the incorporation of branch history reduces the number of collisions with in-flight instructions to the point that HCVP can still achieve good performance while suppressing predictions altogether on a collision.

2.3 E-Stride Hybrid

Though value history-based predictors can predict constant and strided values, these patterns are not space efficient. For example, to predict a constant value, the stride history must consist of a sequence of 0s, while a computation-based or branch history based predictor could store these patterns in a single entry.

To improve the accuracy and coverage of HCVP, we choose to hybridize with E-Stride, which allows HCVP to focus resources on correlated patterns that E-Stride does not detect. Figure 4 shows how HCVP and E-Stride interact: If E-Stride can make a prediction, it is given priority. Both HCVP and E-Stride train on the actual value, regardless of the source of the prediction.

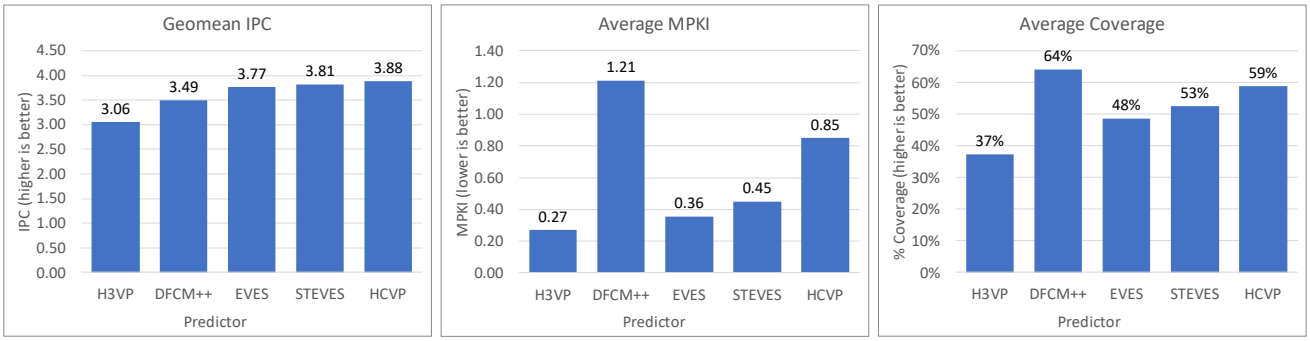
Instruction Window Size	256
Fetch Width	16
Branch Prediction	Two-level predictor
Memory Disambiguation	Perfect
L1 cache	32 KB, 4-way, 64B, 2c
L2 cache	1 MB, 8-way, 64B, 12c
L3 cache	8 MB, 16-way, 128B, 60c
Main memory	150c fixed latency

Table 1: Baseline Microarchitecture

3. EVALUATION

We follow the methodology prescribed by Championship Value Prediction (CVP) organizers for a year-round competition that was started in 2018 [6]. In particular, we use the simulator and benchmarks provided for the competition. The simulator models an out-of-order pipeline and is configured according to the CVP rules. The baseline configuration is listed in Table 1.

We present results for all 135 public traces provided by the CVP committee. The traces, each with 30M dynamic instructions, include a mix of compute (integer and floating point benchmarks) and server-class benchmarks. These samae traces were used in CVP 2018. We compare HCVP against the leading submissions on the CVP leaderboard in the unlimited hardware budget category. Specifically, we compare against H3VP, the first predictor to achieve improvement over the baseline in CVP in all budget categories, EVES [7], which won the championship in 2018 in all budget categories,



(a) IPC (b) Accuracy (c) Coverage

Figure 5: Comparison of value predictors with unlimited budget across 135 public traces.

DFCM++ [3], the runner up to EVES in the unlimited budget category, and STEVES [8], a recent submission that outperforms EVES in the unlimited budget category.

H3VP focuses on predicting hand-selected sequences that were found to be useful. EVES combines VTAGE, a state-of-the-art branch history based value predictor for constant values, and Enhanced Stride, or E-Stride, is a state-of-the-art branch history-based value predictor for strided values. DFCM++ uses an FCM-based value predictor, and STEVES is a hybrid between EVES and DFCM++.

We configure each of the predictors according to their original source code submission to the CVP website. EVES uses a variable branch history register, ranging from 0 to 512 bits, and DFCM++ uses value history lengths of 32 and 64. HCVP uses a branch history register of length 128 and a value history length of 16. All of the predictors, including HCVP, are augmented with the E-Stride predictor.

3.1 Comparison Against Prior Art

We now compare HCVP against the baselines in in the unlimited category. This evaluation is useful for exploring the limits of prediction for different value prediction algorithms.

Figure 5a compares HCVP against H3VP, DFCM++, EVES, and STEVES under unlimited hardware budgets. Each bar represents the geometric mean IPC over a baseline with no value predictor. We see that HCVP achieves the highest IPC. DFCM++, which is most directly comparable to HCVP due to its use of value history, achieves only a 3.49 IPC. Thus, HCVP significantly advances the state-of-the-art for both branch history-based predictors and value history-based predictors.

Figures 5b and 5c show the MPKI (mispredictions per thousand instructions measured over all instructions) and coverage, respectively. HCVP sees a higher MPKI than EVES and STEVES because it targets hard-to-predict value sequences. However, this high MPKI is compensated by HCVP's 6.2% higher coverage than STEVES and 10.3% higher coverage than EVES.

Compared to DFCM++, HCVP reduces MPKI significantly, even though it uses much shorter history lengths. DFCM++ tries to predict correlated values, but it cannot handle divergence. Furthermore, DFCM++ speculatively updates its value history, which can lead to cascading mispredictions. However, HCVP achieves significantly less coverage than DFCM++ because it does not predict in-flight (PC, BHR) pairs, which we measure to make up roughly 31% of prediction-eligible instructions. These results show that even though HCVP sacrifices some coverage, it achieves a better tradeoff between accuracy and coverage that results in an overall performance boost. By incorporating branch history when accessing the BHT and SHT, HCVP significantly reduces the impact of stale entries under back-to-back predictions.

4. CONCLUSIONS

In this paper, we have introduced a method of identifying a new class of value patterns, namely, control-dependent value sequences, that have not been previously explored. We have shown that it is possible to predict these values by combining control-flow and value history information, and we have shown that this combination addresses many of the complications associated with FCM predictors. For example, it reduces the amount of value history that is needed for high accuracy, and it does not require speculative values for in-flight instructions.

Leveraging these insights, we have proposed a new value predictor, called the Heterogeneous-Context Value Predictor. Prior FCM predictors consume megabytes of storage, so HCVP is the most space-efficient design in the FCM literature. We hope that our findings will encourage future work that explores hardware optimizations that reduce the cost of using value histories for practical deployment.

Acknowledgments.

This work was partially funded by the Samsung Global Research Outreach program. This work was also funded in part by NSF Grant CCF-1823546 and a gift from Intel Corporation through the NSF/Intel Partnership on Foundational Microarchitecture Research.

5. REFERENCES

[1] A. Perais, *Increasing the performance of superscalar processors through value prediction*. PhD thesis, INRIA, 2015.

[2] Y. Sazeides and J. E. Smith, "Implementations of context based value predictors," tech. rep., Technical Report ECE-97-8, University of Wisconsin-Madison, 1997.

[3] N. Deshmukh, S. Verma, P. Agrawal, B. Panda, and M. Chaudhuri, "Dfcm++: Augmenting dfcm with early update and data dependence-driven value estimation," *First Championship Value Prediction, CVP 2018*, 2018.

[4] T. Nakra, R. Gupta, and M. L. Soffa, "Global context-based value prediction," in *Proceedings Fifth International Symposium on High-Performance Computer Architecture*, pp. 4–12, IEEE, 1999.

[5] B. Goeman, H. Vandierendonck, and K. De Bosschere, "Differential fcm: Increasing value prediction accuracy by improving table usage efficiency," in *Proceedings HPCA Seventh International Symposium on High-Performance Computer Architecture*, pp. 207–216, IEEE, 2001.

[6] "Championship value prediction," 2018. `https://www.microarch.org/cvp1/index.html`.

[7] A. Seznec, "Exploring value prediction with the eves predictor," 2018.

[8] A. Gupta, P. Mor, H. Taneja, and B. Panda, "Steves: Pushing the limits of value predictors with sliding fcm and eves," *Championship Value Prediction*, 2019.