H3VP: History Based Highly Reliable Hybrid Value Predictor

Kenichi Koizumi, Kei Hiraki, and Mary Inaba
The University of Tokyo, Tokyo, Japan
Email: {koiken,hiraki,mary}@is.s.u-tokyo.ac.jp

***Abstract*—We propose the history based highly reliable hybrid value predictor (H3VP) for the 1st Championship Value Prediction. This is a hybrid predictor comprising three individual predictors, used for instructions whose results from either constant, arithmetic, two-periodic, or three-periodic sequences. Each predictor calculates a reliability value, and if this exceeds a certain threshold, the corresponding predictor starts working speculative predictions. The H3VP combines the speculations from each predictor to determine the final prediction. During the tests on the 135 traces provided for the competition, it achieved a geometric mean of 3.582 instructions per cycle in the track with the unlimited budgets, an improvement of about 11.56% compared to the baseline without speculation.**

I. Introduction

In computer architecture, the prediction of the results that will be obtained from executing instructions is important to deal with the dataflow dependencies that can cause a calculation bottleneck. However, these predictions can also result in a very significant penalty when the execution process is rewound. The number of cycles required for execution increases. Therefore, it is important that such speculative predictors introduce as few mistakes as possible.

Value locality has been discussed in previous studies on value prediction [1], [2]. [3] indicates that over 50% of instructions produce the same results as results of the last execution, and 80% will produce results matching one of their previous sixteen. The values themselves are often obtained from computation results that come close to them in time series, but we must consider a broad range of value sequences if we want to avoid mispredictions, consequently improving speculation reliability.

Our proposed value predictor considers very long value histories for each instruction corresponding to each program counter. Our predictor reduces mispredictions by determining the appropriate time to begin speculation based on these histories. Storing the sequences in full would require a large amount of storage; therefore, to save space, we characterize them in terms of four sequence types and make speculations for a particular instruction only if its values conform to one of the following four sequences: (1) constant (one-periodic), (2) arithmetic, (3) two-periodic, and (4) three-periodic sequences. We also use a two-step confidence threshold as an additional optimization technique. In addition, we use a hash function to store the history data related to each program counter (PC) value, dealing with hash collisions by adjusting the PC value assigned to each entry in the History Table as execution progresses, changing the state of the predictor's state machine when a collision occurs.

We also evaluate our predictor's performance using the traces provided for the 1st Championship Value Prediction (CVP-1). We submitted it to all of the competition's three tracks, which had 8 kB, 32 kB, and unlimited storage budgets, respectively.

II. Preliminary Observations

Before designing our predictors, we first evaluated the benchmark traces provided for the CVP-1. In a sample trace comprising 1 million instructions, among with 87% were ALU instructions and 11% were memory load instructions. In addition, value sequences produced by executing the same instruction had periodicities. Indeed, for 49% of the addresses, the instructions always yielded the same values, whereas others had periods of two or three, and a few had periods of four or more. In addition, 92% of the instructions in the trace had two kinds of PCs. These resulted in monotonically increasing or decreasing value sequences. These observations indicated that the most efficient approach would be to design a predictor for instructions that output either constant value or arithmetic sequences and that it would be effective to assign the predictor a storage budget for predicting values with periods of two or three, as much as the resources allowed.

III. History Based Highly Reliable Hybrid Data Predictor

Figure 1 shows a block diagram of our History Based Highly Reliable Hybrid Value Predictor (H3VP). This combines the results of three different types of predictors that use a common history table. Each of the three predictors uses its own independent state machine, whose state is managed for each entry in the history table. Figure 2 shows the state machine's transition diagram. Here, INIT means no instruction execution information has been provided for that entry and that it has not been assigned to any PC. PREPARE indicates that the corresponding entry has yet to obtain sufficient information to determine periodicity. VERIFY means that the corresponding entry has obtained sufficient information, but the prediction reliability has not reached the threshold value for that entry. SPECULATE means that the prediction reliability has reached the threshold and that the corresponding instruction is actively speculating. FAILED represents the occurrence of a misprediction. When this happens, the confidence threshold is raised

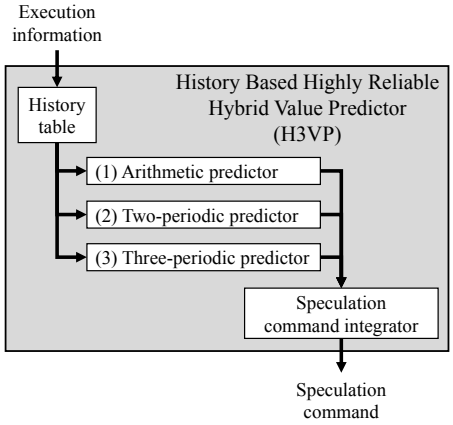

Fig. 1. Overview of History Based Highly Reliable Hybrid Value Predictor (H3VP)

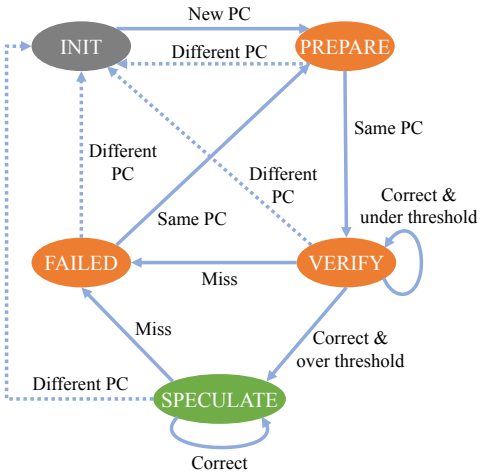

Fig. 2. State machine for an individual value predictor.

for that entry, as discussed below. In addition, when a PC that is different from the current PC for a given entry is executed three times in succession, the machine discards the information about the current PC and allocates the entry to the new PC.

Figure 3 shows a detailed block diagram of our H3VP. The current PC is hashed to generate an index value with a smaller bit width that is used to access the history table, thereby storing data about this PC (except possibly in the event of a hash collision, as discussed below). The history table and predictor statuses are updated as follows: when the target entry is in the INIT state, meaning that it is not yet tied to any PC, it is immediately assigned to the current PC. Every time the instruction at this address is executed, the result (actual value) is stored in the entry's Hist0 field. The values produced by the previous execution of the instruction and the one preceding that are stored in Hist1 and Hist2, respectively, and the difference between the latest actual value and the value previously stored in Hist0 is stored in Diff field. The predicted value for the next execution of the instruction is calculated based on the Hist0, Hist1, Hist2, and Diff fields. For example, the arithmetic predictor outputs the value obtained by adding the values in the Hist0 and Diff fields. If this prediction turns out to be equal to the next actual value, the confidence counter (Cnf) will be incremented.

The predictors decide the performance of speculation as follows: for each of the three predictors (arithmetic, two-periodic, three-periodic), speculation will only be active if the confidence counter corresponding to the current instruction's PC exceeds the current confidence threshold. Regardless of whether speculation is active, the entry's status is set to FAILED if the predicted and actual values differ. The integration block combines the speculation results from the three individual predictors. Here, if at least one of the predictors make a speculation, it is executed for that instruction. If the prediction fails, the predictor responsible is updated accordingly.

In the event of a hash collision, i.e., two different PCs being assigned to the same table entry, the PC assignments are updated as follows: suppose that we have two different program counters, PC_1 and PC_2, with the same hash $[hash(PC_1) = hash(PC_2)]$, and the corresponding table entry has been assigned to PC_1. If an instruction is executed at address PC_2, "PC_2" is temporarily stored in that entry's NewPC field in the history table. Every time the instruction at PC_2 is executed, the Cnt field is incremented. If its value reaches the threshold of 3, the entry is reassigned to PC_2, i.e., the PC field is changed to PC_2. We used three as the threshold, but there is room for optimization in determining this value. However, if an instruction other than PC_2 is executed before the counter (Cnt) reaches three, the counter is reset to zero.

We used a two-step threshold for the confidence counters. The 2nd confidence threshold is larger than the 1st confidence threshold. A small threshold value (1st) was used as long as no mispredictions occurred for the instruction at the corresponding PC, but a larger threshold value (2nd) was used for entries that caused even one prediction miss. Based on the results of our preliminary evaluation, we set the 2nd confidence threshold to 112, and the 1st threshold value to 24 for CVP-1's 8kB track and 10 for its 32 kB and unlimited tracks.

IV. Performance

TABLE I
AVERAGE IPCS FOR OUR H3VP FOR CVP-1'S THREE TRACKS

	8 kB	**32 kB**	**Unlimited**
No prediction	3.211		
H3VP	3.341	3.378	3.582

In this section, we present our predictors' performance results. Table I shows the H3VP's scores for CVP-1's three tracks, provided as the geometric means of the instructions per cycle (IPC) over the 135 trace benchmarks provided. Compared with the no-predictor baseline, it achieved a 4.06%, 5.19%, and 11.56% improvement in IPC for the tracks with

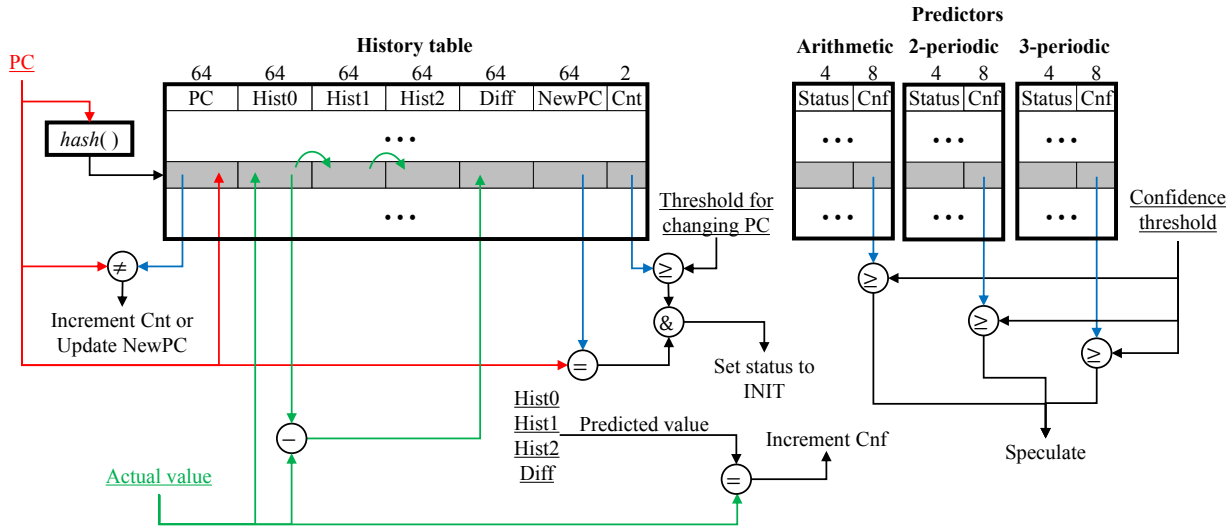

Fig. 3. Block diagram for the History Based Highly Reliable Hybrid Value Predictor (H3VP), showing the bit widths of each field.

8 kB, 32 kB, and the unlimited budgets, respectively. Figure 4 shows the (geometric) mean IPC performance for the individual predictors and H3VP for the 8 kB, 32 kB, and unlimited tracks. Among individual predictors, the arithmetic predictor's IPC is the highest. Four individual predictors have situations where each works effectively. For example, the constant predictor conforms to instructions generating an immediate value. The arithmetic predictor conforms to instructions related to a control flow on an application (e.g., an incremental counter of iterations). The two-periodic predictor conforms to instructions processing tuples of data. The three-periodic predictor conforms to instructions related to coordinate calculation in applications handling three-dimensional space. At this time, the arithmetic predictor conforming to instructions related to flow control operations achieves high performance in many traces. On the other hand, two- and three-periodic predictors achieve high performance only in applications including specific processes. Note that the value sequences that can be predicted by the constant predictor are a subset of those that can be predicted by the other three predictors, namely, the arithmetic, two-periodic, and three-periodic predictors. Such scores as the H3VP could not be achieved by any of the predictors alone, but only by integrating the output of the three individual predictors. Instructions that can be predicted only with the two- or three-periodic predictors do exist. However, many value sequences predictable by the two- or three-periodic predictors can also be predicted with the constant predictor. Therefore, in realistic microarchitecture design, employment of the constant and arithmetic predictors should be given priority based on the limitation of storage budget.

Figure 6 shows the (geometric) mean IPC performance for the H3VP for the 8 kB, 32 kB, and unlimited tracks, where the x-axis indicates the large threshold value for the confidence counter. The reason why we carefully selected the

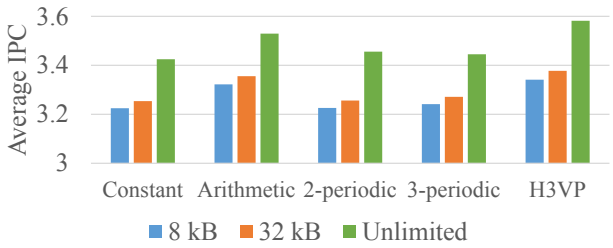

Fig. 4. Average IPC scores for the individual predictors and H3VP, for the unlimited track.

optimal value for the large threshold is that the performance varies greatly depending on the value of this threshold. The value of the confidence threshold to achieve the highest IPC was 112 in all three tracks. If the threshold value is too small, it is impossible to avoid failure of speculation due to an appearance of an irregular value with a period length longer than the threshold value. Then, the IPC performance significantly decreases. On the other hand, if the threshold value is too large, the beginning of speculation delays and the predictor will miss out rewards for speculation success. Then, the IPC performance gently decreases. The optimal confidence threshold is determined by the ratio of the acceleration rewards on speculation success to the rewinding penalty on speculation failure. These factors depend on an architecture used for the evaluation. In architectures with more significant rewinding penalty, it is necessary to set the confidence threshold to a larger value.

Figure 5 shows the IPC scores for CVP-1 trace, indicating that the H3VP yielded an improvement for 125 of the traces. In many *srv* traces, the H3VP achieves high IPC improvements, while improvements for *compute_fp* traces is not good. One of the reasons is that the arithmetic predictor in the H3VP sees the sequences of 64-bit floating point values as integer se-

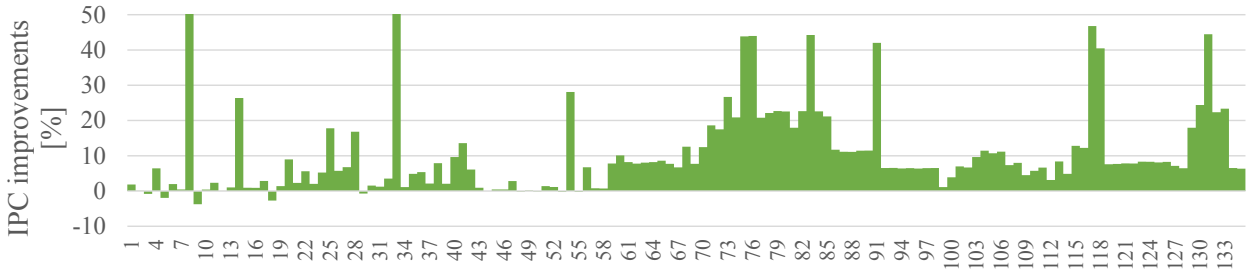

Fig. 5. IPC improvements yielded by the H3VP for each of the 135 CVP-1 traces. Trace number 1–45, 46–58, and 59–135 are *compute_int*, *compute_fp*, and *srv* traces, respectively. The improvement of trace *compute_int_7* (#8) is 633% and that of trace *compute_int_34* (#33) is 169%.

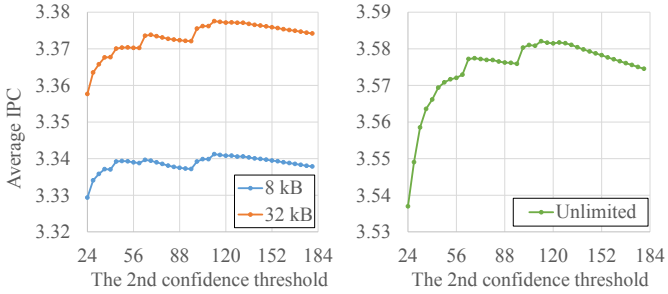

Fig. 6. Average IPC scores for the H3VP for the 8 kB, 32 kB, and the unlimited tracks. The 1st confidence threshold is set to 24 in the limited tracks, and 10 in the unlimited track.

quences. If applications with many floating-point instructions are targeted for the speculation such as *compute_fp* traces, an arithmetic predictor with floating-point adders and subtractors will be helpful as a way of increasing coverages.

V. Storage Budget

Now, we consider the storage consumed by each of our individual predictors and for the H3VP overall. Table II shows the single-entry storage consumption for the history table and each predictor. The number of bits for fields of the history table and predictor indicates actually required bit widths for the members of the history table and predictor structure in the submitted source code. Table III shows the composition, entry size, and total storage consumed by the predictors we submitted for each track. We deploy three predictors, arithmetic, two-periodic, and three-periodic predictors, except the constant predictor. In 8 kB, 32 kB, and the unlimited tracks, the bit widths of hashed index value are 7, 9, and 20, respectively. Since each predictor shares the history table, we can reduce the required storage. The H3VP can change the required storage amount only by increasing or decreasing the number of entries.

VI. Summary

In this paper, we proposed the History Based Highly Reliable Hybrid Value Predictor (H3VP). We achieved IPC improvements for the CVP-1 trace benchmarks by focusing on speculation targets that can be characterized by arithmetic or periodic sequences. This approach can effectively suppress the increase in storage consumption while allowing us to maintain a large history table. When applied to the traces provided for the competition, our predictor improved the IPC by an average of 4.06%, 5,19%, and 11.56% for the 8 kB, 32 kB, and unlimited traces, respectively, provided in the competition.

TABLE II
Single-entry storage consumption for the history table and each predictor.

Struct	Pointer	Storage [bits]
HistoryTable	*pc* (PC)	64
	history (Hist0–2)	64 × 3
	diff (Diff)	64
	newpc (NewPC)	64
	newpc_cnt (counter for NewPC)	2
	Total	386
MyPredictor	*status* (a portion of Status)	3
	count (confidence counter)	8
	failed (a portion of Status)	1
	Total	12
Combined total (history table and three predictors)		422

TABLE III
Storage consumption of our submissions

Track	History table size	Storage
8 kB	128	54016 bits (6.59 kB)
32 kB	512	216064 bits (26.4 kB)
Unlimited	1048576	442499072 bits (52.8 MB)

References

[1] Y. Sazeides and J. E. Smith, "The Predictability of Data Values," in *Proceedings of the 30th Annual ACM/IEEE International Symposium on Microarchitecture*, ser. MICRO 30. Washington, DC, USA: IEEE Computer Society, 1997, pp. 248–258. [Online]. Available: http://dl.acm.org/citation.cfm?id=266800.266824

[2] K. Wang and M. Franklin, "Highly Accurate Data Value Prediction Using Hybrid Predictors," in *Proceedings of the 30th Annual ACM/IEEE International Symposium on Microarchitecture*, ser. MICRO 30. Washington, DC, USA: IEEE Computer Society, 1997, pp. 281–290. [Online]. Available: http://dl.acm.org/citation.cfm?id=266800.266827

[3] M. H. Lipasti, C. B. Wilkerson, and J. P. Shen, "Value Locality and Load Value Prediction," in *Proceedings of the Seventh International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS VII. New York, NY, USA: ACM, 1996, pp. 138–147. [Online]. Available: http://doi.acm.org/10.1145/237090.237173