Oracle: A 16KB EVES-based Value Predictor

Alenkruth Krishnan Murali (jht9sy) [1]
Khyati Kiyawat (vyn9mp) [2]

[1]Ph.D. student, Department of Computer Engineering, University of Virginia

[2]Ph.D. student, Department of Computer Science, University of Virginia

alenkruth@virginia.edu, khyati@virginia.edu

Outline – In this study, we explore the performance of a 16KB value predictor, Oracle, on a set of 10 traces (compute_int and srv). Considering that this is part of a competition (with no secret traces), we tune the 8KB EVES (Enhanced VTAGE[1] Enhanced Stride)[2] predictor to extract the best prediction on loads for the provided traces. With knowledge of how the predictor works, we observe that we can extract maximum performance by increasing the storage allocated to the Enhanced VTAGE predictor by 216% and reducing the storage allocated to the Enhanced Stride predictor by 94%. With this configuration, Oracle achieves an IPC of **3.2616** on the provided traces with a storage of **15.9954KB**.

1 Block diagram

The implemented predictor referred to as Oracle from now onward is a 16KB value predictor based on the EVES predictor proposed in [2]. Fig.1 shows the block diagrams of different components of Oracle. It is a 14-component VTAGE predictor with a Stride based value predictor. The base predictor is backed up by 13 tagged predictors indexed with the hash of different lengths of the Global History Register(GHR) and the Program Counter(PC). The predictor entries contain a pointer indexing to a value in the value table. The stride predictor predicts based on the previous result of the instruction. Sections 3 and 4 explain the details of the VTAGE and Stride predictors.

2 Size calculation

The VTAGE predictor is implemented using a large VTAGE array that has 48 interleaved banks with 64 entries in each bank. Each entry consists of a 10-bit pointer or hash, a 14-bit tag, a 1-bit usefulness counter (u_a), and a 5-bit confidence counter (c_a), i.e. 28 bits. 8 bits of the pointer are used to index 2^8 lines of the value table shown in Fig.1. The value table is a 3-way skewed associative data value table and therefore the remaining 2 bits of the pointer are used to index the way. Each entry of the value table consists of a 57-bit value and a 1-bit usefulness counter (u_v). The number of bits used by the VTAGE array and the value table is shown below:

Size of VTAGE array:
$Nbanks * Nentries * Nbits/entry$ Here, $Nbanks = 48$, $Nentries = 64$, and
$Nbits/entry = pointer + tag_a + u_a + c_a$

$$= 10 + 14 + 1 + 3 = 28$$

Thus, the VTAGE array uses 48*64*28 = **86016** bits.

Size of Value table:
$Nways * Nentries * Nbits/entry$
Here, $Nways = 3$, $Nentries = 256$, and
$Nbits/entry = data_valve + u_v = 57 + 1 = 58$

Thus, the value table uses 3*256*58 = **44544** bits.

Size of Stride predictor:
$Nways * Nentries * Nbits/entry + SafeStrideCounter$
Here, $Nways = 1$, $Nentries = 4$,
$SafeStrideCounter = 16$, and
$Nbits/entry = last\_value + stride + tag_s + u_s + c_s +$

$$NoFirstOcc = 64 + 2 + 5 + 2 + 5 + 1 = 79$$

Thus, the stride predictor uses (1*4*79)+16 = **332** bits.

We have a few more registers in the predictor i.e. an **8**-bit TICK counter, an **8**-bit counter to track the last misprediction, and a **127**-bit Global history register.

The **total storage** required by Oracle is
$86016 + 44544 + 332 + 8 + 8 + 127 =$ **131035** bits, i.e. **15.9954 KB**, which fits within the 16KB storage budget.

3 Fine-tuned EVTAGE

Oracle has a VTAGE predictor which is directly derived from the indirect target branch predictor (ITTAGE)[3]. The VTAGE predictor primarily makes use of the PC and global branch history to predict the instruction result. VTAGE has multiple banks which are indexed with different lengths of the global branch history, hashed with the PC of the instruction. Each VTAGE bank entry has a tag which is obtained from a hash of

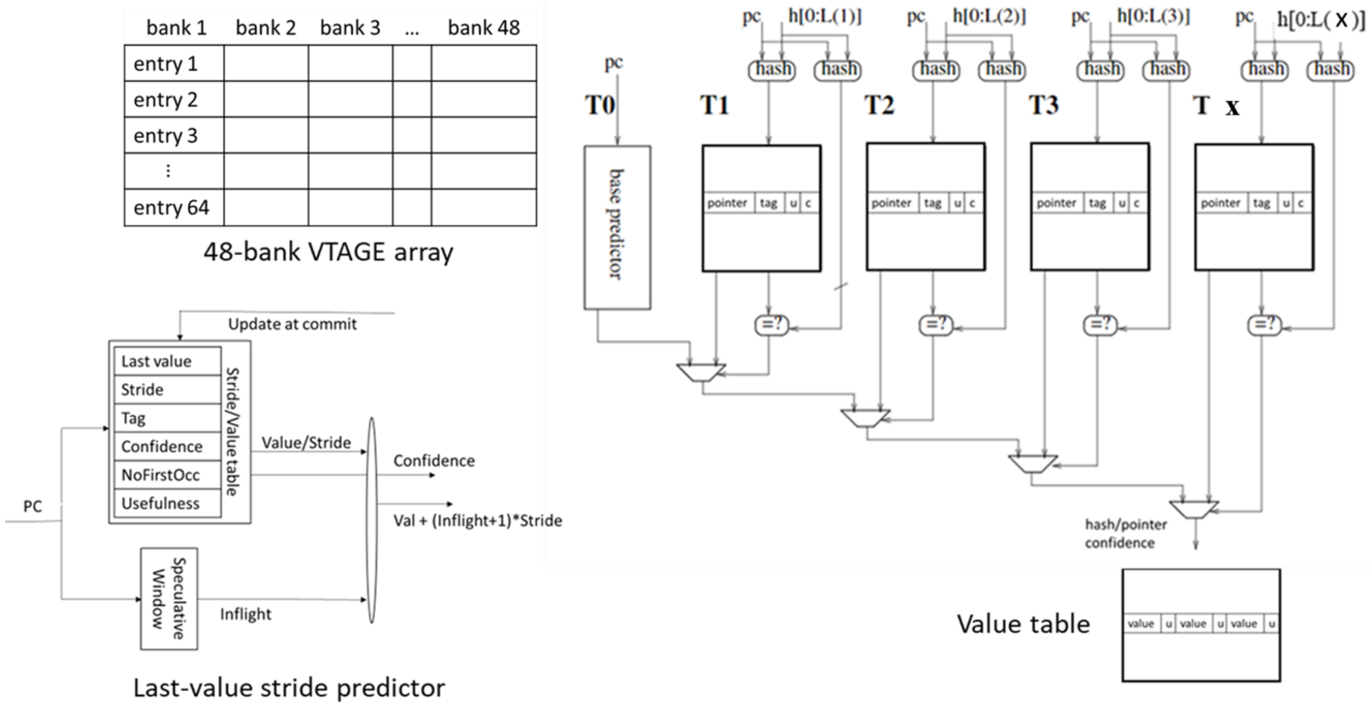

Figure 1: Block diagram of Oracle. Here, x=13.

the global history and PC. This section describes the changes made to the baseline VTAGE predictor and the reason(s) behind them. More information about the VTAGE predictor and the EVTAGE predictor can be found in their respective papers [2][4].

3.1 Tag Length and Predictor Bank Entries

Since VTAGE performs better with longer tags and more entries in the predictor banks, we increased their sizes to observe the potential speedup. We observed that the length of the tags was more critical to performance since we could avoid the aliasing of VTAGE predictor bank entries. Oracle has 48 VTAGE banks with 64 entries each, and a 14-bit tag.

3.2 Usefulness counter

We also reduce the usefulness counter length to 1-bit, removing the hysteresis. We additionally reduce the width of the TICK counter to 8 bits and modify the TICK counter update formula. This accelerates the eviction of unused VTAGE bank entries making space for newer entries. This also retains the *hot* entries in the tables improving prediction.

3.3 History length

VTAGE makes use of global history to make predictions. We can vary the length of the history register to provide more or less history context to the predictors. Oracle makes use of 14 history lengths; up to 127 bits. We observe that a longer history length always does not result in better performance. As the history length increases, the TAGE components with the longer history lengths need a longer time to warm up, thereby delaying prediction. A history length longer than 127 bits provided marginal speedups. We decided to select a history length of 127 bits due to storage limitations. The tags for the predictor banks entries and the indexes for the banks are obtained by hashing the global branch history with the PC of the instruction. We also observed that adding some redundant history length(s) helped with the hashing function. This alleviates aliasing and improves performance. The history lengths utilized by Oracle are listed below:

$$h[14] = \{0, 0, 1, 3, 6, 12, 18, 30, 63, 90, 127, 127, 127, 127\}$$

3.4 Hashing Formula for Index and Tag

The length of the tags in the VTAGE predictor banks is critical to the performance of the predictor. It is essential that the tags are unique in order to avoid aliasing. We hash the global history with the PC to obtain the tag. The tag contains context from the PC as well as from the global history since it is a product of the hash. By increasing the tag length, we were able to avoid aliasing to some extent. Likewise, we index into the predictor banks with an index obtained from hashing the PC and history registers. This necessitates the need to avoid aliasing in index generation as well. Therefore, we update the index and tag generation function by tweaking the hash formula. We update the formula in such a way as to provide the indexes and tags more context from the PC and global history while keeping the index and tag hashes orthogonal to each other.

Oracle's VTAGE obtains an average IPC of 3.2559 for the given traces.

4 Minimal E-Stride

The stride predictor in EVES computes the predicted value as a result of the addition of the last result of the instruction and a stride. While designing Oracle, we observed that the given traces did not benefit from the stride predictor as it did from the VTAGE. Hence, we decided to reduce the storage allocated to the stride predictor in the EVES 8KB implementation by 94%. Oracle has a stride predictor with 4 entries with a 5-bit tag and a 2-bit stride.

The stride predictor obtains an average IPC of 3.2102 which is still better than the baseline EVES 8KB predictor.

5 Performance

Predictor	Size (KB)	Average IPC
only VTAGE	15.95	3.2559
only E-Stride	0.04	3.2102
Oracle	15.9954	3.2616

Table 1: Average IPC obtained using Oracle

Oracle can be found at `https://github.com/Alenkruth/value_predictor/releases/tag/oracle_v1`

Note: Oracle has been heavily tuned to predict values present in the given set of 10 instruction traces. We have not tested the performance of Oracle on traces other than the ones provided. While Oracle outperforms EVES with twice the storage (32KB) we expect EVES (of comparable storage) to perform better on a diverse set of traces.

References

[1] A. Perais and A. Seznec, "Revisiting value prediction," Ph.D. dissertation, INRIA, 2012.

[2] A. Seznec, "Exploring value prediction with the eves predictor," in *CVP-1 2018-1st Championship Value Prediction*, 2018, pp. 1–6.

[3] A. Seznec and P. Michaud, "A case for (partially) tagged geometric history length branch prediction," *The Journal of Instruction-Level Parallelism*, vol. 8, p. 23, 2006.

[4] A. Perais and A. Seznec, "Practical data value speculation for future high-end processors," in *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, 2014, pp. 428–439. DOI: 10.1109/HPCA.2014.6835952.