A Hybrid Stride and Finite Context Method Based Value Predictor

S. Deepak Narayanan† and S. Vinu Sankar†

***Abstract*— There are various types of predictors that are usually involved in value prediction. In this particular work, we have worked on stride-based value prediction and Finite Context Method (FCM) based value prediction on the workload provided for the contestants (*public kit*) and have noted down the results for the same. The IPC (geomean over provided 135 traces) comes out to be 2.552241.**

I. Introduction

The predictor implemented here has mainly two parts. Firstly, we have a Finite Context Method based prediction, which is made with the construction of a hash function generator. Secondly, we have a stride-based value predictor, which makes use of an extra structure for prediction. Typically, the patterns of values may remain constant or differ by a stride or may have some other sequence. Here we make use of regulairty in the pattern. Section 2 below is about the FCM based value prediction and Section 3 is about the stride-based value prediction.

II. Finite Context Method Based Value Prediction

This method works with the help of a hash function generator. The basic predictor structure given by the committee has two entries, the firstLevelEntry and the secondLevelEntry. The first level table has attributes predict time index, commit time index and inflight, whereas, the second level table has attributes predictions and confidence. We here implement a hash function generator that generates the index for accessing the first level table. The first level index f is given by
f = (PC << sizeof(GHR) + GHR) mod ($4096 \times 4 - 1$)

Here PC is the program counter value, GHR is the global history register and 4096×4 - 1 is the first level mask, which is the size of the table structures we have. Taking modulus will ensure that indexing will not go out of the desired range.

The predict time index value from the first level table is accessed using the index value *f*. A modulo operation is performed over the value obtained and the first level mask again. The value obtained now is used to as the second level index. This index value is used to access the prediction value from the second level table.

III. Stride-Based Value Prediction

For storing the stride, a new structure is formed which has three attributes namely PC, stride and value. The PC stores the program counter value, stride that of the difference between the last PC and the current PC and the value column has the value of the actual address. After each instruction, the structure will be appended with the values corresponding to the PC values, if the stride value is found to be same. The structure will be made empty, if the stride misses the pattern for ten consecutive times. While value prediction, if stride pattern is found, then the structure is used for prediction instead of the FCM based prediction.

Let the last two values stored in the structure be *A* and *B* in the order of index. If a pattern is observed and the PC is more than the previous PC value, then we predict the value to be $B + \|B - A\|$.

If the PC is less than the previous PC value, then we predict the value to be that stored in the structure at index LAST-INDEX - 1 - (PC - *B*) BIT-WISE-AND (Stride Value).

IV. Structure of the Predictor

The Predictor is Hybrid - it tracks the steps taken by the Program Counter and selectively takes a decision. If a pattern is observed in the stride, then the stride based predictor is chosen. Else, we go ahead with the Fixed Context Method Predictor. The size of the structure that was implemented for storing the stride is calculated as follows. We are using Vector Data Structure, from the standard template library. We have totally 300 entries, each having three columns, each of which, independently stores 64 bits of data. This totally amounts to 300×364 bits = $300 \times 192 = \frac{57600}{8}$ Bytes = $\frac{7200}{1024} = 7.03125$ Kilo Bytes. Therefore, we are competing in 8KB Category this way.

Now for the 32KB category, we have a vector of size 1200, implemented in the exact same way. The calculation for the 32KB Category follows from above and is just $4 \times 7.03125 = 28.125$ KB. The calculation.

For the unlimited category, we are using a vector with 4800 entries, again implemented in the exact same way, and thefore alculation just multiplies the above value for the 32 KB Category by 4 = $28.125 \times 4 = 112.5$KB. The structure for storing stride values in the mypredictor.h file

† Both the authors have contributed equally.

[1]S. Deepak Narayanan, Department of Computer Science and Engineering, Indian Institute of Technology Gandhinagar, India deepak.narayanan@iitgn.ac.in

[2]S. Vinu Sankar, Department of Computer Science and Engineering, Indian Institute of Technology Gandhinagar, India vinu.sankar@iitgn.ac.in

Name of the Category	Size of Structure
8 KB	7.03125 KB
32 KB	28.125 KB
Unlimited	112.5 KB

```
struct Stride {
uint64_t pc;
uint64_t stride;
uint64_t value;
};
  std::vector<Stride> strideTable;
uint32_t strideCount;
```

V. CONCLUSIONS

The overall IPC count (geomean over 135 traces) comes out to be **2.552241**.

ACKNOWLEDGMENT

The authors of this paper are grateful to Prof. Manu Awasthi for helping them out at various stages of the project. The authors are also thankful to Vinesh Srinivasan and Arthur Perais for their much needed support and help as and when it was needed. The authors finally would like to thank all the people who have helped them during the course of this project.

REFERENCES

[1] T. Nakra, R. Gupta, Mary Lon Soffa, Global Context-Based Value Prediction

[2] B. Calder, G. Reinman and D. M. Tullsen, "Selective value prediction," Proceedings of the 26th International Symposium on Computer Architecture (Cat. No.99CB36367), Atlanta, GA, 1999, pp. 64-74.

[3] Mikko H. Lipasti, Christopher B. Wilkerson, and John Paul Shen. 1996. Value locality and load value prediction. In Proceedings of the seventh international conference on Architectural support for programming languages and operating systems (ASPLOS VII). ACM, New York, NY, USA, 138-147.