A Dual-Length Path-Based Predictor for Thread Prediction

Niko Demus Barli, Luong Dinh Hung, Hideyuki Miura,
Shuichi Sakai and Hidehiko Tanaka

Graduate School of Information Science and Technology, The University of Tokyo.
7-3-1 Hongo Bunkyo-ku, Tokyo 113-8656, Japan
{niko,hung,hide-m,sakai,tanaka}@mtl.t.u-tokyo.ac.jp

1 Introduction

In speculative multithreading (SpMT) architectures which exploit thread level parallelism from a sequential program, predicting and spawning threads that follow a correct control flow is a major performance factor. Strategies for this control speculation issue can be divided into two approaches. The first approach is to support a limited control speculation, in which threads may only be spawn at specific points in the program. For example: loop iterations, function call and returns, or control equivalent points. This approach minimizes thread misprediction by only speculating at highly predictable places. The second approach is a more aggressive one, in which threads are predicted and spawned as soon as possible. This latter approach bears a larger potential for performance improvement, given that the thread predictor is sufficiently accurate.

This paper targets architectures that use the latter approach. It first studies the predictability of thread addresses using a dynamic path-based predictor. The results show that, in an alias free situation, a high prediction accuracy can be achieved by including sufficiently long path information. However, the requirements for prediction table entries are also increasing almost exponentially when longer path information included. For a finite size table, there are cases when a longer path, due to capacity aliasing, leads to a severely deteriorated accuracy.

To overcome the problem, this paper introduces a dual-length path-based prediction technique. We combine two path-based predictors: one predictor is indexed using a shorter path information, while the other predictor is indexed using a longer path information. A selection table is added to dynamically select which predictor's outcome to use. The rationale behind this hybrid technique is to have the shorter path predictor, which has lower capacity requirements, to redeem the situation when the longer path predictor suffers from a severe capacity aliasing. Furthermore, by manipulating predictor's table update policy, threads that can be accurately predicted by one predictor can be filtered out from contaminating the other predictor. Simulation results show that the combined effect significantly improves the accuracy of thread prediction.

2 Related Work

In general, thread prediction is very similar to branch prediction. There are differences in that there is basically no concept of taken/not-taken and that the information of intra-thread branches is unavailable in thread prediction. Nevertheless, the wealth idea of branch correlation, combining predictors, aliasing reduction, and other branch prediction techniques can be borrowed.

Using pattern of recent branches outcomes has been a successful approach for achieving a high branch prediction accuracy [12,13]. Alternatively, a path-based approach for branch correlation has also been proposed and shown to achieve a similar prediction accuracy [8]. In the context of Multiscalar's thread prediction, Jacobson et. al. has shown that path-based approach achieves higher prediction accuracy compared to the pattern-based approach [4]. In this paper, we also base our thread prediction investigation on a path-based approach.

McFarling introduced the concept of combining different branch predictors to form a hybrid predictor [6]. Using hybrid predictors whose components have different path history length has also been reported [3]. This paper applies the concept of using different path history length components of hybrid predictors in thread predictions, and further explores the possibility of reducing aliasing by manipulating predictor's updating policy.

While hybrid predictors dynamically select one of the components for each prediction, branch classification [1], predictor with elastic history buffer [11], and variable length path-based branch prediction [10], use execution profile for classifying and associating each static branch to an optimal prediction mechanism.

3 Methodology

Throughout this paper, we employed a static method to partition a sequential program into threads. A thread is a connected subgraph of the program's control flow graph with exactly one entry point. Overlapped regions shared by two or more threads may not exist. Thread boundaries were put at function invocations, returns, and at innermost loop iterations by a compiler. For the remaining parts of the program, thread boundaries were placed so that the resulting threads have a maximum size. For Spec95int applications, the resulting threads had an average dynamic size ranging from 14 to 21 instructions.

Simulations were conducted using traces generated by a speculative multithreading simulator. The benchmark used in simulations are eight applications from Spec95int suite. Input for each application was modified to keep the trace size within reasonable size (100M - 200M instructions). When presenting simulation results, we will use harmonic mean for measuring the "average" of prediction accuracy.

We assumed an idealized predictor update timing: before predicting a succeeding thread, the predictor is properly updated using the most recent thread information. When a thread exited from a return instruction, we excluded the succeeding thread from the trace. Threads that follow a thread exiting from a return instruction is best predicted using a return address stack, which is however beyond the scope of this paper.

Direct-mapped tagless tables were used during experiments. For an n-length path-based predictor table, we concatenate address bits from the latest n threads, and fold (using bitwise XOR) the concatenated bits to form index for the table. The notation is, for example, 12-8-8-8|3|: we concatenate 12-bit address from the current thread, 8-bit address from the next three threads, and fold the resulting bits by three, generating a 12-bit index.

4 Dual-Length Path-Based Predictor

4.1 Predictability

Fig. 1 shows prediction accuracy achieved using path-based approach, given an alias free table and complete thread address information for the path. High prediction accuracy is theoretically achievable if sufficiently long path information is incorporated. However, when we lengthen the path, the number of table entries required for prediction is also increasing almost exponentially (fig. 2). For three applications of Spec95int, using a path length of four requires more than 10k entries. Assuming that each entry holds a 32-bit address, a 4k-entry table has already occupied 32-kB of space. Thus, although the predictability is high, limitation on table size may restrict the final accuracy achieved.

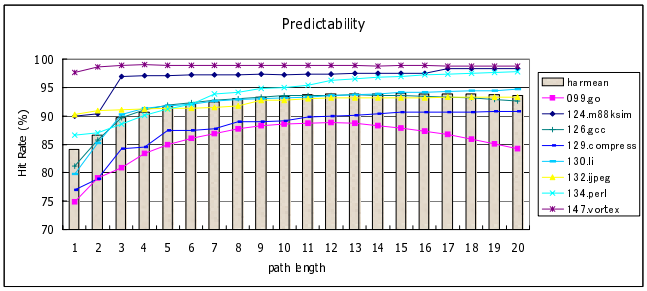

Figure 1: Predictability using path information

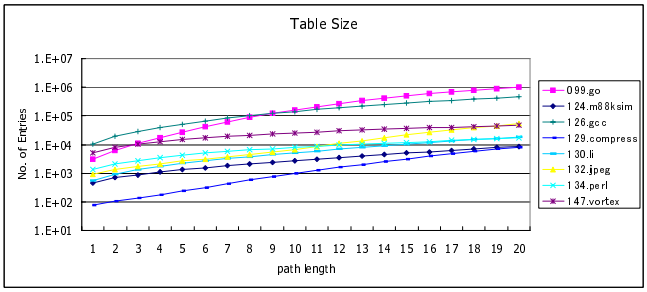

Figure 2: No. of entries used in prediction table

For a finite size table, the accuracy can be considered as a function of path length and application's footprint. While longer path is advantageous for small applications, severe capacity aliasing may occur in large applications. Fig. 3 shows prediction accuracy for two representative applications using a 4k-entry table. For n-th thread in the path m bits of address is concatenated, where m=max(12-2n,4). The concatenating result is folded as needed to form a 12-bit index. The figure shows that, although in *li* lengthening the path from one to four increased the prediction accuracy by 11%, in case of *go*, it decreased the accuracy by 7.5%.

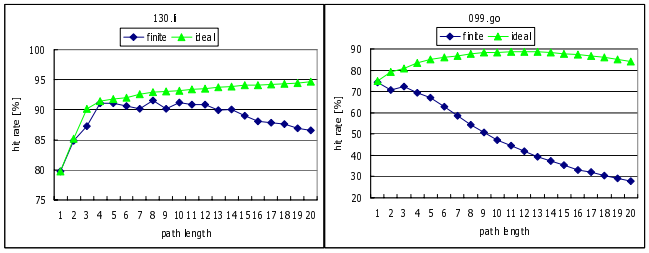

Figure 3: Accuracy for finite size table

4.2 Combining Predictors

Predictability study in the previous section has indicated that, for a finite size table the optimal path length varies following footprint characteristics of the program. The penalty incurred when missing the optimal point is not negligible. Thus, it is difficult to optimize a fixed single-length predictor to cover a wide range of applications.

This observation leads us to the idea of combining two path-based predictors with different path length. A selection table indexed by the address of current thread is added. This table contains 3-bit saturating counters with the counter's MSB bit serves as a selection bit. When one of the predictor predicted correctly and the other predictor missed the prediction, the counter is incremented or decremented accordingly. The logic behind this hybrid approach is to prepare both a longer path (high-predictability high-aliasing) predictor and a shorter path (low-predictability low-aliasing) predictor. Applications that can take advantage of a longer path length can be predicted with high accuracy, while the ones suffering from severe aliasing may still be rescued.

4.3 Reducing Aliasing

Aliasing, specifically capacity aliasing, is the origin of the problem that prevents the path-based predictor from achieving higher prediction accuracy. While combining predictors of different path length helps to adaptively adjust the prediction to its optimal performance, another possible approach to increase prediction accuracy is to reduce the aliasing itself.

A number of dealiased branch predictors have been proposed in the past. Skewed predictor [7] uses majority vote from odd number of component predictors to produce final prediction. This predictor exploits the characteristics that in a lightly aliased situation, if an entry in one predictor is aliased, there is little possibility that the other tables are also aliased. Agree [9] and Bimode [5] predictor avoid destructive aliasing by trying to store identical prediction bits in the same table. Filter predictor [2] filters highly biased branches and uses BTB instead of PHT to predict these branches, thus, reducing aliasing in the PHT.

We explored the possibility of reducing aliasing in the hybrid predictor by manipulating prediction table update policy. In the original hybrid predictor approach, both component predictors are always checked for prediction miss and updated when a miss occured. However, for branches that can be predicted correctly in one predictor, updating the other table introduces unnecessary contamination. To reduce this contamination, we

investigated the following two update policies in addition to the original update policy (we call the original policy as **Hybrid-Always**).

- **Hybrid-Partial**: Confidence counter (2-bit resetting counter) is added into each entry of the selection table. The counter is incremented in the case of prediction hit and the selection counter is already saturated. When the confidence counter is also saturated, only the currently selected prediction table is checked and updated. The counter is reset when a miss occurred.
- **Hybrid-Lazy**: Prediction tables are checked and updated only when the selected prediction missed. In case of hit, the unselected table is not updated even if its prediction would have missed.

5 Evaluations

Fig. 4 shows prediction accuracy for single scheme predictors with path length of one (4k-entry, index=12|1|) and four (4k-entry, index=12-8-8-8|3|), and prediction accuracy for variations of dual-length hybrid predictors that combined predictors with path length of one (2k-entry, index=11|1|) and four (2k-entry, index=11-8-7-7|3|). A 4k-entry selection table is used in hybrid predictors constituting approximately 10% additional bits. Hit rate for an ideal predictor of path length four is also shown as references.

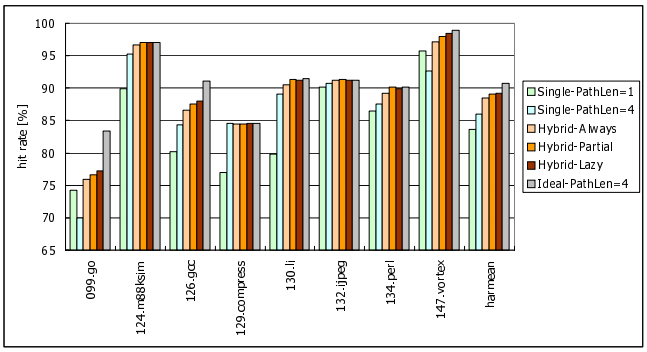

Figure 4: Hit rate for 4k-entry prediction table

The results show considerable improvement achieved by hybrid predictors over the single scheme predictors. Especially in the case of *go*, *gcc*, and *vortex*, in which capacity aliasing is prohibitive, improvement over single scheme predictor with path length of four is significant. In average, **Hybrid-Always** outperforms single scheme predictor (path length of four) by 2.4%. **Hybrid-Partial** and **Hybrid-Lazy** further improve the prediction accuracy by 0.6% and 0.7% respectively.

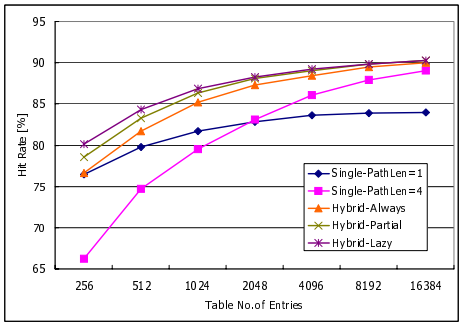

Figure 5: Average hit rate vs. prediction table size

Fig. 5 shows the average hit rate when prediction table size varied from 256 to 16k entries. The effect of aliasing reduction, especially in **Hybrid-Lazy** predictor, is increasing for smaller table size. When the table gets larger, the advantages of hybrid approach over single scheme predictor with path length of four is diminishing. At this point, we should consider longer path length for the components of the hybrid predictors.

6 Conclusion

The predictability of threads achieved using path-based approach is fairly high if sufficiently long path information is incorporated. However, long path information may lead into severe aliasing problem in programs with large footprint. To solve this problem, this paper introduced a dual-length path-based predictor in which two predictors with different path length are combined to form a hybrid predictor. The hybrid approach allows us to exploit higher predictability of the longer path predictor when the aliasing is not prohibitive. At the same time, the shorter path predictor provides backup when the aliasing is becoming severe. Furthermore, aliasing can be reduced by filtering unnecessary update to prediction tables of the hybrid predictor. Evaluation results showed the combined effect significantly improved the prediction accuracy over single scheme predictors.

References

[1] P.-Y. Chang. Branch Classification: a New Mechanism for Improving Branch Predictor Performance. In *Proc. of the 27th MICRO*, pages 22–31, 1994.

[2] P.-Y. Chang, M. Evers, and Y. N. Patt. Improving Branch Prediction Accuracy by Reducing Pattern History Table Interference. In *Proc. of the 1996 PACT*, pages 48–57, 1996.

[3] K. Driesen and U. Holzle. Accurate Indirect Branch Prediction. In *Proc. of the 25th ISCA*, pages 167–178, 1998.

[4] Q. Jacobson, S. Bennett, N. Sharma, and J. E. Smith. Control Flow Speculation in Multiscalar Processors . In *Proc. of the 3rd HPCA*, pages 218–229, 1997.

[5] C.-C. Lee, I.-C. K. Chen, and T. N. Mudge. The Bi-mode Branch Predictor. In *Proc. of the 30th MICRO*, pages 4–13, 1997.

[6] S. McFarling. Combining Branch Predictors. Technical Report TN-36 Digital Western Research Lab., 1993.

[7] P. Michaud, A. Seznec, and R. Uhlig. Trading Conflict and Capacity Aliasing in Conditional Branch Predictors. In *Proc. of the 24th ISCA*, pages 292–303, 1997.

[8] R. Nair. Dynamic Path-Based Branch Correlation. In *Proc. of the 28th MICRO*, pages 15–23, 1995.

[9] E. Sprangle, R. S. Chappell, M. Alsup, and Y. N. Patt. The Agree Predictor: A Mechanism for Reducing Negative Branch History Interference. In *Proc. of the 24th ISCA*, pages 284–291, 1997.

[10] J. Stark, M. Evers, and Y. N. Patt. Variable Length Path Branch Prediction. In *Proc. of the 8th ASPLOS*, pages 170–179, 1998.

[11] M.-D. Tarlescu, K. B. Theobald, and G. R. Gao. Elastic History Buffer: A Low-Cost Method to Improve Branch Prediction Accuracy. In *Proc. of the 1997 ICCD*, pages 82–87, 1997.

[12] T.-Y. Yeh and Y. N. Patt. Two-Level Adaptive Branch Prediction. In *Proc. of the 24th MICRO*, pages 51–61, 1991.

[13] T.-Y. Yeh and Y. N. Patt. Alternative Implementations of Two-Level Adaptive Branch Prediction. In *Proc. of the 19th ISCA*, pages 124–134, 1992.