Interference in Branch Predictors: A Systematic Approach

Veerle Desmet*,[1]
Hans Vandierendonck*,[2],
Koen De Bosschere*

* *ELIS, Ghent University – UGent, Sint-Pietersnieuwstraat 41, 9000 Gent, Belgium*

ABSTRACT

As a result of resource limitations, state in branch predictors is frequently shared between uncorrelated branches. This interference can significantly limit prediction accuracy. In current predictor designs, the branches sharing prediction information are determined by their branch addresses and thus branch groups are arbitrarily chosen during compilation. This feasibility study explores a more analytic and systematic approach to classify branches into *clusters* with similar behavioral characteristics, and we evaluate several ways to incorporate this additional source of information in branch predictors.

Our profile–based results illustrate the potential of using clustering information in various types of branch predictors. In particular for small predictor budgets, clustered indexing is a very cost effective solution, e.g. the misprediction rate in a 1 KiB gshare predictor is reduced by 19% and 12.3% for SPECint2000 in a self and cross profiling setup respectively.

KEYWORDS: branch prediction, aliasing, clustering

1 Introduction

Past research has emphasized the aliasing problem in dynamic branch predictors and has pointed out the mostly destructive nature of this interference of branches [YGS95]. We measured that the misprediction rate in a 1 KiB gshare predictor would be reduced by 34% if all destructive aliasing could be avoided.

Today, the pairs of branches that cause aliasing are determined by their branch addresses. As these addresses are arbitrarily chosen by the compiler, this results in a variable and un-

[1]E–mail: veerle.desmet@elis.UGent.be
Veerle Desmet is supported by a grant from the Flemish Institute for the Promotion of the Scientific-Technological Research in the Industry (IWT).
[2]E–mail: hans.vandierendonck@elis.UGent.be
Hans Vandierendonck is a postdoctoral researcher of the Fund for Scientific Research-Flanders (FWO).

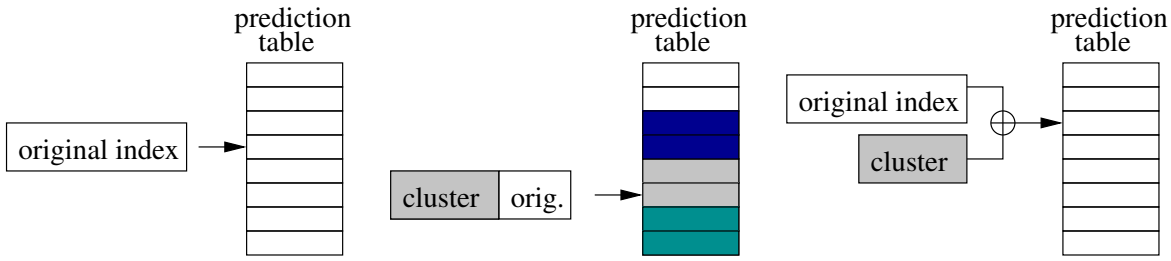


Figure 1: Left: Traditional Indexing. Center: clustered indexing with subtables. Right: clustered indexing with hashing.

controlled amount of aliasing. The key idea behind this research is to fix the pairs of branches sharing prediction information in a systematic way. Therefore, a cluster technique is used to form branch clusters based on their behavioral characteristics. The paper focuses on profile–based techniques for carefully selecting clusters of branches that can share prediction information without affecting performance.

Further, this study explores the feasibility to guide various dynamic branch predictors according to these static branch cluster information. Although we focus on branch prediction, clustered indexing is a general technique and can be applied to various other structures in processors, e.g. branch target buffers, trace caches, prefetchers, and value predictors.

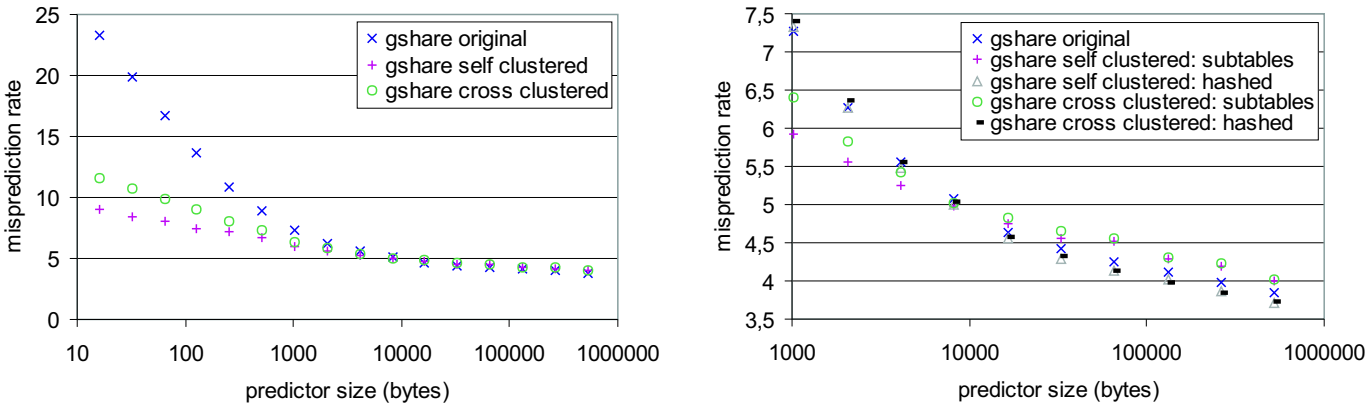
2 Clustered Indexing

Clustering provides an alternative information source that is useful for indexing a structure. The aim in branch prediction is to structure the information kept in prediction tables so that different branches benefit from looking into a particular table entry, guided by cluster information. In particular, we partition a given prediction table into a set of smaller tables or *subtables*, each of those only used by a predefined group or *cluster* of branches. Figure 1 illustrates the use of a subtable identifier in exchange for part of the original index. Although smaller prediction tables increase aliasing, the overall performance would not degrade if the branches in a same subtable are chosen so that they rarely negatively interact. Another approach to incorporate the cluster information is shown in the right of Fig. 1 which hashes the cluster identifier together with the original index to form an index in the prediction table.

3 Identifying static branch clusters

The identification of branch clusters is based on following two observations. First, branches with identical time varying outcome behavior can use the same table section while benefiting from constructive aliasing. Secondly, two branches sharing resources yet executing in strictly separable time intervals do not influence each other, except for some initialization period. We incorporate these aspects in our metric to quantify the behavioral properties of static branches: *the taken rate behavior in time slices of 10M conditional branch executions*.

With behavioral properties of branches as input we use the k–means clustering algorithm [PM00] to assign each static branch to exactly one cluster. Two important examples are the ever recurring clusters ‘always taken’ and ‘always not taken’, both reflecting that many branches have a tendency to be most of their time either taken or not taken.



(a) Subtables

(b) Hashing

Figure 2: Original indexing compared to clustered indexing for gshare.

4 Evaluation

Here, we use the branch clusters identified based on profile information and we distinguish *self* clustering from *cross* clustering. Self clustering is somewhat optimistic, whereas cross clustering is a realistic approach in a profiling setup.

4.1 Gshare: clustered indexing with subtables

Consider a gshare branch predictor [McF93]. The comparison of its original indexing to clustered indexing in a subtable setup is illustrated in Fig. 2(a). Clustered indexing outperforms the originally indexed gshare for bit budgets up to 8KiB. For a 1KiB gshare the misprediction rate is decreased from 7.3% downto 5.9%, a reduction of the misprediction rate by 19% in case of self profiling clusters. Cross profiling attains 5.4% and preserves 12.3% reduction of the miprediction rate. For large predictor sizes clustered indexing with subtables performs slightly worse.

4.2 Hashing as alternative to subtables

Side effect of using subtables is the smaller portion of the original index being used (smaller tables require less index bits), which also motivates the slightly higher alias rate. In case of gshare, subtabling shortens the effective length of the used global history. As alternative to subtables, we propose *hashing* or the concatenation of the cluster identification number with the branch address after which this whole bit string is XOR–ed with the global history. This is in contrast with the previous section where the branch address and the global history are XOR–ed after which the cluster identification is concatenated. Hashing does not need subtables, so the entire table is indexed which allows the use of histories as long as in the original predictor. Fig. 2(b) shows that for large predictors hashing with the cluster identification number can reduce the misprediction rate by 5% on average for a 256 KiB gshare predictor (from 4% downto 3.8%).

4.3 Other predictors

At small budgets, a clustered gshare outperforms (results not shown) several more powerful predictors like bi–mode [LCM97], gskew [MSU97]. Interestingly to point out is that even a clustered version of the bi–mode predictor performs worse than a clustered gshare.

Moreover, the proposed technique is applicable to any other prediction scheme that uses table indexing. In particular, a very accurate 1 KiB perceptron predictor [JL01] improves 15% (self) and 6.7% (cross) over the original scheme by using clustered indexing (from 6% downto 5.1%). At 4KiB, the misprediction rate is still reduced by 2.2% (self) when using clustered indexing but at 16 KiB we measure 4% mispredictions for both original and clustered indexing, so we can state that clustered indexing improves a perceptron predictor until the point where the misprediction rate saturates.

5 Conclusion

This paper describes clustered indexing, an alternative indexing mechanism that eliminates destructive aliasing by carefully selecting branch clusters that share prediction information. These branch clusters are identified by a k–means clustering algorithm based on the time varying taken rate behavior of the static branches.

In various dynamic branch predictors, clustered indexing with subtables can significantly reduce misprediction rates in small predictors; larger predictor sizes benefit from hashing the cluster identification in exchange for less branch address bits. For SPECint2000, this compile time mapping between branches and table entries reduces 19% and 12% of the mispredictions in a self–profiled and cross–pofiled setup respectively for an 1 KiB gshare.

References

[JL01] Daniel A. Jiménez and Calvin Lin. Dynamic branch prediction with perceptrons. In *Proceedings of the 7th International Symposium on High Performance Computer Architecture*, pages 197–206, January 2001.

[LCM97] Chih-Chieh Lee, I-Cheng K. Chen, and Trevor N. Mudge. The bi–mode branch predictor. In *Proceedings of the 30th Annual International Symposium on Microarchitecture*, pages 4–13, December 1997.

[McF93] Scott McFarling. Combining branch predictors. Technical Report TN-36, Digital Western Research Laboratory, June 1993.

[MSU97] Pierre Michaud, André Seznec, and Richard Uhlig. Trading conflict and capacity aliasing in conditional branch predictors. In *Proceedings of the 24th Annual International Symposium on Computer Architecture*, pages 292–303, June 1997.

[PM00] Dan Pelleg and Andrew Moore. X-means: Extending k–means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 727–734, 2000.

[YGS95] Cliff Young, Nicolas Gloy, and Michael D. Smith. A comparative analysis of schemes for correlated branch prediction. In *Proceedings of the 22nd Annual International Symposium on Computer Architecture*, pages 276–286, May 1995.