Available online at www.sciencedirect.com
ScienceDirect
Microprocessors and Microsystems 31 (2007) 168–177
MICROPROCESSORS AND MICROSYSTEMS
www.elsevier.com/locate/micpro

Clustered indexing for branch predictors

Veerle Desmet *, Hans Vandierendonck, Koen De Bosschere

Ghent University – UGent, Department of Electronics and Information Systems (ELIS), Parallel Information Systems (PARIS) Group, Member HiPEAC, Sint-Pietersnieuwstraat 41, 9000 Gent, Belgium

Available online 8 September 2006

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. We present several ways to incorporate this cluster information as an additional information source in branch predictors.

Our profile-based results demonstrate that cluster information is useful in various branch prediction schemes. When *clustered indexing* is applied, the same performance can be obtained with 2–8 times less hardware budget. For small predictor budgets, clustered indexing is very cost-effective, e.g., the misprediction rate in an 8 Kib gshare is reduced 12.3% on average for SPEC CPU2000 INT. For large budgets up to 4 Mib, clustered indexing reduces the number of mispredictions by 3–5%, or stated otherwise only half the hardware budget is required to obtain the same performance as the original gshare scheme.

Keywords: Branch prediction; Aliasing; Clustering

1. Introduction

Past research has emphasized the aliasing problem in dynamic branch predictors and has pointed at the mostly destructive nature of this interference of branches [18]. We measured that the misprediction rate in a 8 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 uncontrolled 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 systematically selecting clusters of branches that can share prediction information. Therefore, we applied a cluster algorithm in which the time varying taken rate behavior of branches serves as similarity criterion during cluster formation. Kim and Tyson [8] already suggested to reuse prediction resources only for branches that interleave in time. In addition, our clustered indexing strategy allows static branches with similar dynamic behavior to share resources. In other words, we additionally convert destructive interference into neutral interference, providing an even more effective predictor solution.

Further, this study explores the feasibility to guide various dynamic branch predictors according to this static branch cluster information. The results show that clustered indexing is extremely useful in today's range of practical branch prediction implementations. Typically, those implemented branch predictors are simple and small, often based on bimodal, global and gshare components. Although we focus on branch prediction, clustered indexing is a general technique that can be used in various other hardware

* Corresponding author. Tel.: +32 9 264 3594; fax: +32 9 264 3405.
E-mail addresses: veerle.desmet@elis.UGent.be (V. Desmet), hans.vandierendonck@elis.UGent.be (H. Vandierendonck), koen.debosschere@elis.UGent.be (K. De Bosschere).

0141-9331/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.micpro.2006.08.003

structures in processors, e.g., branch target buffers, trace caches, pre-fetchers, value predictors, etc.

The remainder of this paper is organized as follows. Section 2 starts with the basic concept of our alternative indexing technique and Section 3 continues with the discussion of how static branch clusters are determined. After providing more details on the methodology in Section 4, the results of clustered indexing are presented in Section 5 for various branch predictors. Section 6 discusses related work and in Section 7 we conclude the paper.

2. Clustered indexing

The aim of clustered indexing is to structure the information kept in branch prediction tables. At the same predictor budget, this information structuring process results in higher branch prediction accuracies. Otherwise stated, the cost-effectiveness of branch predictors can be improved when branch prediction is guided by additional cluster information. This section deals with various ways for using cluster information in a predictor, and the next section details on how branches can be divided in branch clusters.

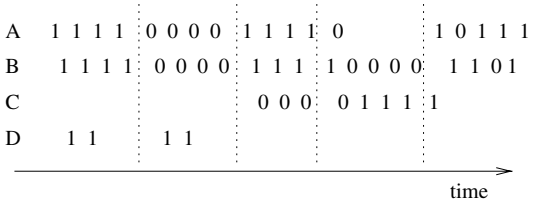
Cluster information provides an alternative information source that is useful for indexing a structure. As branches within a cluster have some behavioral properties in common, we can decide to use a specific subset in the prediction tables per cluster of branches. This means that a given prediction table is partitioned into a set of smaller tables or *subtables*, each of those only used by a predefined cluster of branches. Fig. 1 illustrates different methods to use the cluster information in the indexing process. The term 'original index' is left unspecified as the technique is generally applicable in every prediction scheme. We compare the original indexing scheme (left) against clustered indexing where the cluster information is used as subtable identifier (center). Since identifying an entry within a subtable requires less bits compared to the index for the entire prediction table, the cluster information is used in exchange for some bits in 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. One of the contributions of this paper is to cluster branches such that using the cluster information in a setup with subtables augments the prediction accuracy by eliminating destructive aliasing.

Another approach to incorporate the cluster information is shown in the right of Fig. 1. In this case, the original index and the cluster identifier are hashed together to select an entry in the prediction table. This hashing strategy preserves the original index length, which is important to explore correlation in e.g., long histories. In the next section we discuss the identification of such branch clusters.

3. Identifying static branch clusters

In this section, we describe our experimental setup to obtain clusters of static branches that can share prediction state without degrading prediction accuracy. The identification of branch clusters is based on the following two observations. First, branches with identical time varying outcome behavior can use the same table section while benefiting from constructive aliasing (e.g., A and B in Fig. 2). Second, two branches sharing resources yet executing in strictly separable time intervals do not influence each other, except for some initialization period (e.g., C and D in Fig. 2). To incorporate both the outcome behavior and the time notion, we have chosen to quantify the behavioral properties of static branches by *the taken rate behavior in time slices*.

Our results indicate this is an appropriate metric, although we do not claim this is the best possible one. We have also analyzed the transition rate but it turned out to perform worse then the taken rate. The reason is that the transition rate distribution sharply peaks at 'no



A	100%	0%	100%	0%	80%
B	100%	0%	100%	20%	75%
C	NE	NE	0%	75%	100%
D	100%	100%	NE	NE	NE

Fig. 2. Up, branch outcome behavior for 4 static branches A, B, C and D with '1' and '0' for a taken and not-taken branch outcome, respectively. Down, representation by means of time varying taken rate behavior; NE, 'not executed'.

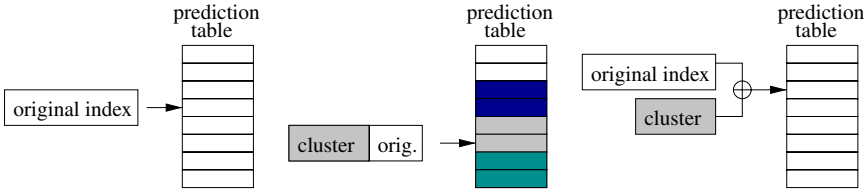


Fig. 1. Original indexing (left) compared to clustered indexing with subtables (center) and clustered indexing with hashing (right).