Computer Languages, Systems & Structures 53 (2018) 143–162

Contents lists available at ScienceDirect

Computer Languages, Systems & Structures

journal homepage: www.elsevier.com/locate/cl

Data access skipping for recursive partitioning methods☆

Orhan Kislal*, Mahmut T. Kandemir

The Pennsylvania State University, State College, PA 16802, USA

ARTICLE INFO

Article history:
Received 4 January 2018
Revised 18 March 2018
Accepted 19 March 2018
Available online 27 March 2018

Keywords:
Memory
Machine learning
Compiler optimization
Parallel programming

ABSTRACT

The memory performance of data mining applications became crucial due to increasing dataset sizes and multi-level cache hierarchies. Recursive partitioning methods such as decision tree and random forest learning are some of the most important algorithms in this field, and numerous researchers worked on improving the accuracy of model trees as well as enhancing the overall performance of the learning process. Most modern applications that employ decision tree learning favor creating multiple models for higher accuracy by sacrificing performance. In this work, we exploit the flexibility inherent in recursive partitioning based applications regarding performance and accuracy tradeoffs, and propose a framework to improve performance with negligible accuracy losses. This framework employs a data access skipping module (DASM) using which costly cache accesses are skipped according to the aggressiveness of the strategy specified by the user and a heuristic to predict skipped data accesses to keep accuracy losses at minimum. Our experimental evaluation shows that the proposed framework offers significant performance improvements (up to 25%) with relatively much smaller losses in accuracy (up to 8%) over the original case. We demonstrate that our framework is scalable under various accuracy requirements via exploring accuracy changes over time and replacement policies. In addition, we explore NoC/SNUCA systems for similar opportunities of memory performance improvement.

1. Introduction

Recursive partitioning is a well-studied predictive modeling approach that is widely used in data mining and related fields. While prior research [2–5] has investigated different implementations of decision trees, random forests and their variants, evaluating and optimizing its performance in the context of emerging multi-level on-chip cache hierarchies and modern main memory systems took much less attention. Most decision tree learning implementations, like many other data mining algorithms, are memory-bound since they typically apply computationally-trivial operations to large amounts of data. In doing so, they require multiple accesses to the same data in order to create the tree levels. This increases the dependency on memory performance and, consequently, on the effectiveness of data access optimizations.

Prior work suggested different strategies to handle costly memory accesses in data-intensive applications, including memory request scheduling [6–8], on-chip network optimizations [9–11], and aggressive cache optimizations (at both architectural and software levels) [12–15]. While these optimizations are successful in certain cases, their effectiveness

☆ An earlier version of this article was presented at the annual conference of IPDPS [1]. There are numerous additions in this paper, including, but not limited to, the pooling optimization, the extension into random forests and their respective experimental results. Various chapters are revised and extended as well.

* Corresponding author.
E-mail address: kislal.orhan@gmail.com (O. Kislal).

https://doi.org/10.1016/j.cl.2018.03.003

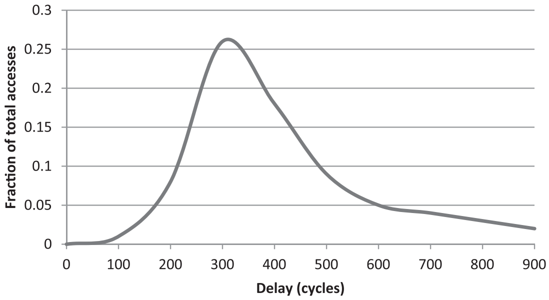

Fig. 1. The data access distribution for our decision tree learning implementation used in this work on the NoC/SNUCA system (details of our experimental platform will be given later in Section 5). We see that some data accesses are much costlier compared to others.

is ultimately limited as they try to preserve the *correctness of execution*. Approximate computing [16–19], an emerging research area, instead suggests that, by relaxing the exact correctness requirement, it may be possible to reach a faster (and possibly more energy efficient) execution in certain applications.

An important question then, in the context of recursive partitioning algorithms, is whether approximate computing can be employed to reduce their memory access latencies without significantly affecting the correctness of the original applications. Focusing on one particular implementation of decision tree learning [5] and skipping some portion of costly data accesses (as our approximate computing strategy), this paper starts with two critical observations:

- The inaccuracy resulting from skipping data accesses depends more on the number of data accesses skipped, rather than which specific data accesses are skipped.
- In contrast, the performance benefits achieved through data access skipping depends strongly on which specific data accesses are skipped.

The reason for the first observation is the fact that every data point is accessed multiple times throughout the execution. Even if a data point is skipped and replaced with a (possibly incorrect) prediction, there is a very high probability that the same point will not be skipped again, and it will eventually contribute to the decision tree. The reason for the second observation is the fact that costs of different data accesses (even in the case of data-parallel applications) are not uniform in current multicore and manycore systems. Specifically, a data access may hit at any level in the on-chip cache hierarchy (composed of L1, L2 and L3 in modern systems), or miss in all of them. In the latter case, a main memory request is made, whose cost depends on the on-chip network latency, memory queuing latency, and whether we hit in the row-buffer or not. A distribution of data access latencies in a typical execution of our target decision tree application is plotted in Fig. 1 (note that according to prior research, such patterns are not unexpected [20]). One can observe from this graph that, data accesses exhibit a great deal of variations in their latencies. Consequently, when considering the two observations above, two alternate approximate computing strategies (for a given algorithm) that skip the *same* amount of data accesses are expected to result in similar inaccuracies but *significantly different* performance improvements.

Motivated by this result, our goal in this work is to explore the potential performance improvements that can be achieved via data access skipping. Specifically, our proposed approach tries to skip a fraction of costly data accesses (last-level cache misses) in an attempt to maximize the performance benefits under a given inaccuracy bound. In addition, we explore a pooling technique that has been shown to improve performance in data mining algorithms and adapt it to work in conjunction with our optimization. Our detailed experiments with this approach applied to a sample decision tree learning implementation show that:

- A small portion of the data accesses has a negligible impact on accuracy but a significant impact on performance. Specifically, skipping 3% of the data accesses reduces the accuracy by only 4–6%, but improves the memory performance by 50–55%. These savings translate to an average execution time improvement of 15%.
- Randomly skipping data accesses (i.e., without considering their latency costs) offers considerably smaller improvements compared to our scheme. For example skipping 3% of the data points “randomly” improves the memory performance only by 4–5%.
- Our cache miss skipping scheme is applicable for both uniform and non-uniform cache hierarchies. In fact skipping 3% of the data points improves the memory performance by 45–48% and the overall performance by 13–14% on the NoC/SNUCA based system.
- Applying the pooling technique causes a negligible (1%) performance overhead but improves the accuracy by approximately 3% on more aggressive skipping ratios.
- Constructing multiple trees for random forest training reduces the loss of accuracy even further. Specifically, skipping 3% of the data access reduces the accuracy by only 2%, but improves the execution time by 12%.

The remainder of this paper is structured as follows. A general overview of decision tree and random forest methods are provided in Section 2. Section 3 explains the details of our data access skipping strategy. The pooling technique and