Contents lists available at SciVerse ScienceDirect

J. Parallel Distrib. Comput.

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

Parallel approaches to machine learning—A comprehensive survey

Sujatha R. Upadhyaya

Infosys Technologies, Bangalore, India

ARTICLE INFO

Article history:
Received 1 May 2011
Received in revised form
16 September 2012
Accepted 7 November 2012
Available online 16 November 2012

Keywords:
Distributed and parallel machine learning
GPU
Map reduce

ABSTRACT

Literature has always witnessed efforts that make use of parallel algorithms / parallel architecture to improve performance; machine learning space is no exception. In fact, a considerable effort has gone into this area in the past fifteen years. Our report attempts to bring together and consolidate such attempts. It tracks the development in this area since the inception of the idea in 1995, identifies different phases during the time period 1995–2011 and marks important achievements. When it comes to performance enhancement, GPU platforms have carved a special niche for themselves. The strength of these platforms comes from the capability to speed up computations exponentially by way of parallel architecture / programming methods. While it is evident that computationally complex processes like image processing, gaming etc. stand to gain much from parallel architectures; studies suggest that general purpose tasks such as machine learning, graph traversal, and finite state machines are also identified as the parallel applications of the future. Map reduce is another important technique that has evolved during this period and as the literature has it, it has been proved to be an important aid in delivering performance of machine learning algorithms on GPUs. The report summarily presents the path of developments.

© 2012 Elsevier Inc. All rights reserved.

1. Introduction: parallel machine learning

MACHINE learning offers a wide range of statistical algorithms for analysis, mining and prediction. It includes various techniques such as association rule mining, decision trees, regression, support vector machines, and other data mining techniques. All these algorithms are computationally expensive which makes them the ideal cases for implementation using parallel architecture/parallel programming methods. One of the earliest efforts in this direction dates back to 1995, where K. Thearling [69] discussed the possibilities of enhancing the performance of the popular machine learning approaches such as memory-based reasoning, neural networks, and genetic algorithms by adopting a parallel processing approach. During 1995–2000, there were a number of efforts that focused on improving performance of the association rule mining algorithm by means of parallel programming. A survey report on "Parallel and Distributed Association Mining" by Mohammed Zaki [57], gives a complete summary of efforts made in this period. However, the efforts so far did not focus on performing machine learning tasks on graphic processors. 'Fast matrix multiplication on graphics processors' published in 2001 [43] is one of the first reports that discussed building functions on GPUs. Although efforts like this cannot be marked as machine learning tasks, they in turn helped analyze the machine learning algorithms from the perspective of running them on parallel architecture. Such attempts triggered the efforts that focused on building machine learning techniques on the graphic processors. During the years 2002–2010 there has been a surge of reports that focused on data mining tasks on GPUs. These reports primarily discussed the performance improvement of data mining and other machine learning techniques by algorithm enhancements or even by making use of other popular techniques such as 'map reduce' algorithm. Currently, we see this space buzzing with activity, with focus on text mining, data mining, map reduce and GPU-based implementations.

During the entire stretch of these 15 years, three distinct trends are identified, that record a shift in focus. In the first trend that covered the period from 1995 to date, the focus is on introducing parallelism into Machine learning. These efforts include works that leverage distributed multicore architectures or simply introduce parallelism into the procedure in a different manner. Interestingly, even with the introduction of specialized hardware such as GPUs, similar efforts have continued even until now. However, with the advent of the GPUs in the early 2000s, there was a visible shift of focus in research to machine learning on GPU platforms. The data monster article presented 2009 predicted a paradigm shift being brought about by introduction of GPUs [24]. The last decade has witnessed multiple efforts on GPU processors, however, during the latter 5 years which marked the third trend, most of the efforts were largely influenced by the use of map reduce technique too. The map reduce technique seemed to have influenced all domains of computer science with its growing popularity after its

E-mail addresses: sujatha_upadhyaya@infosys.com, sujatha.upadhyaya@gmail.com.

0743-7315/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2012.11.001

application in web search by Google. The present report sees this entire stretch of fifteen years (1995–2010) as a period before GPU, after GPUs and a period after map reduce popularization, namely

1. General parallel data mining and machine learning approaches: From 1995 until now.
2. Parallel data mining and machine learning on GPUs: From 2000 until now.
3. Parallel data mining and machine learning with map reduce techniques. From year 2005 until now.

2. General parallel machine learning approaches

In this category, we take into consideration every parallel machine learning effort that does not particularly refer to GPU architecture or map reduce technique. The time period observed is 1995 until now. It is interesting to note that most of the efforts were related to data mining, particularly frequent itemset mining and association rule mining. However, there have been several other efforts focusing on performance issues, and other machine learning tasks/algorithms like text mining,

2.1. Association rule mining and frequent itemset mining

Association rule mining (ARM) and frequent itemset mining (FIM) are closely related topics. Finding frequent itemsets is deemed to be a prerequisite to ARM and is the most critical step in association rule mining. An association rule mining is a problem of arriving at the rules of the form *A* implies *B* where *A* and *B* are itemsets, with good frequency and strength. Support of a rule is defined as the joint probability of transactions containing both *A* and *B* and the confidence of the rule is the conditional probability that a transaction contains *B* given that it contains *A*. As mentioned in the Introduction, Zaki et al. [77] cover all the technical aspects of parallel association rule mining/frequent itemset mining. It summarizes the similar efforts between 1996 and 1999 and brings out issues varying from types of algorithm to characteristic features of algorithms. This present report looks into almost every aspect of data mining although it does not trace the individual contribution of the reports. This is one of the landmark reports that summarize the efforts until 1999. Mueller et al. published a report on the comparison of fast sequential itemset mining algorithms with parallel approaches [51] in 1995 that summarizes the state of art.

2.1.1. Major algorithms and other observations—ARM

A. *Algorithms*: Association rule mining was first introduced in 1993, although the parallel data mining paradigm was introduced much later. Many of the then existing algorithms were tried and modified to work on the parallel platforms. The Apriori algorithm, DHP (Direct Hash Pruning) algorithm and DIC(Dynamic Itemset mining) are some of the algorithms that have greatly influenced parallel association rule mining space. In fact, the Apriori algorithm forms the basis for almost all the algorithms including DIC and DHP. The categorization is chosen for the sake of simplicity of presentation. The following section will discuss the algorithms that were developed in parallel data mining contexts.

Algorithms based on the Apriori algorithm: The Apriori algorithm was first proposed by Agarwal et al. in 1993. Although this algorithm was originally proposed for a sequential context, it was later adopted in many parallel contexts. Many algorithms were developed and tried on 'shared memory' and 'shared nothing' architectures and later modified to suit other platforms too.

The following section discusses the parallel algorithms influenced by the Apriori algorithm and these algorithms represent a sequential improvement on the previous one. They differ in the methods adopted for partitioning and distributed mining of large itemsets. A substantial improvement on performance and scalability factors was shown through these implementations.

Count distribution: This algorithm is designed to keep the processors busy even at the cost of performing redundant calculations. Each processor generates the complete candidate *k*-itemset, using the frequent itemset generated in the previous pass. Since the frequent itemset generated is common, all the processors will be generating identical candidate itemsets for each pass. Each processor then independently generates the local support count for candidates in the candidate itemset and at the end all local counts are taken into consideration to get the global count and build the frequent itemset.

Data distribution: Quite contrary to the count distribution algorithm data distribution algorithm each processor processes mutually exclusive candidate itemsets. The disadvantage is that each processor will have to broadcast the local data to other processors in each pass unlike the count distribution algorithm that broadcasts only the counts in each pass.

Candidate distribution: This algorithm attempts to avoid synchronizing at the end of each pass. In each pass the algorithm divides the frequent itemset into such a way that each processor can generate a unique candidate set independent of other processors, while the data is selectively replicated.

Intelligent data distribution: The algorithm uses the aggregate memory of the parallel computer employing an intelligent data partition scheme and an efficient communication system. The algorithm improves over the data distribution algorithm.

Hybrid distribution: This is an improvement over the previous one that aims at load balancing by dynamic partitioning. In order to ensure this, the locally stored portion of the databases is sent to other processors by a ring based all-to-all broadcast.

Fast distribution algorithm: In this algorithm, the process of generation of candidates remains the same as the Apriori algorithm. The relationship between the local and global large sets is used to generate a smaller set of candidates, thus reducing the number of messages to be passed. Two pruning techniques local and global hash tree pruning techniques that ensure that $O(n)$ messages are sufficient against the typical requirement of $O(n\hat{}2)$ messages.

NPA: In the Non-Partitioned Apriori algorithm the candidate itemsets are partitioned such that each piece fits into the local memory of a processor and is copied into all the processors. Each of the processors proceeds individually to identify the *k*-itemsets and the hash tables are then updated to determine the consolidated support strength across the processors.

SPA: In SPA or the Simply-Partitioned Apriori the data is shared or broadcast to all the processors.

HPA: The Hash-Partitioned Apriori algorithm partitions the candidate itemset using a hash function and this reduces the broadcasting efforts and the comparison workload.

HPA-ELD: The HPA works well, however, if the size of the candidate itemset is smaller than the system memory, HPA does not make use of the remaining space. The HPA-ELD (HPA with Extremely large itemset duplication) does utilize the memory by copying some of the itemsets. It chooses the frequently occurring itemsets and copies them over the processors so that all the space is used.

Algorithms based on DHP algorithm: Direct Hashing and Pruning or DHP is one of the earliest ARM algorithms that focused on fast generation of itemsets and reduction in database size. Although the above-mentioned algorithms based on the Apriori algorithm did use hash-based partitions, they did not use hashing techniques to prune search trees. The DHP algorithm is extended on the

Download English Version:

https://daneshyari.com/en/article/431862

Download Persian Version:

https://daneshyari.com/article/431862

Daneshyari.com