J. Parallel Distrib. Comput. 73 (2013) 284–292

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.

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

Apriori algorithm, where hashing techniques were used to prune the hash search trees. The algorithm is proposed by Park et al. [58]; it describes a hashing technique for fast generation of frequent itemsets (particularly 2-itemsets) and a pruning technique to reduce the size of the transaction data base.

Parallel data mining for association rules: This algorithm is a generalized version of DHP. This algorithm shortly known as PDM, consists of parallel generation of candidate itemsets and parallel determination of large itemsets. It uses a hash table to generate the candidate itemsets in earlier passes and in later passes computes the most frequent itemsets from the candidate itemset directly. Each pass in the algorithm works on the portion of the data that resides in the database partition of the particular node and needs to obtain information from the other processors to obtain global count.

Algorithms based on DIC (Dynamic Itemset Counting): This algorithm was designed to overcome the disadvantages of previous algorithms (Apriori and DHP) that required several passes on the database. It reduces the number of passes required for generating the frequent itemset and number of items scanned per pass.

APM (adaptive parallel mining): It is often called the Asynchronous Parallel Algorithm and is based on DIC (Dynamic Itemset Counting) algorithms. The DIC algorithm divides the database into intervals, performs counts on intervals rather than the entire database. However, the APM algorithm overcomes the problems caused due to uneven distribution of data and ensures homogeneity of distribution within the itemset partition. This algorithm is proposed specifically for a shared memory processor to overcome the performance degradation caused by the synchronization requirement in shared memory multiprocessors, because of the conventional level-wise approach. With this approach all the participating processors are made to work on the computation of support without having to wait for others to finish. The algorithm also achieved less I/O requirements,

Other algorithms:

Fast parallel mining (FPM): The FPM is a modification of the count distribution algorithm, which adopts distributed and global pruning techniques. Simple messaging schemes of the algorithm were the added contributions of this method.

Parallel frequent pattern growth algorithm: This algorithm is based on FPG that extracts frequent patterns from a FP tree, without candidate itemset generation.

B. Noted publications and their contribution to parallel association rule mining

"Efficient parallel data mining for Association rules", by Park et al. [57], in 1995 is one of the earliest efforts on parallel data mining that aims at efficient identification of large itemsets by a hashing technique that collects the count information pertaining to itemsets within a partition.

Yet another attempt on introducing parallelism in association rule mining and frequent itemset mining was described in "Hash-based Parallel Algorithms for Mining Association Rules" by Shintani et al. [67] in 1996. In the present report, the authors proposed four algorithms; NPA (Non-Partitioned Apriori), SPA (Simply Partitioned Apriori), HPA (Hash-Partitioned Apriori) and HPA-ELD (HPA with Extremely Large itemset Duplication). Their experiments showed that HPA and HPA-ELD are efficient in handling very large data sets and also successfully handled the data skew problem. All these algorithms were implemented in a share-nothing environment.

A similar effort on parallel data mining on a share-nothing multiprocessor by Agarwal et al. [2] in 1996, entitled "Parallel Mining of Association Rules" proposed three algorithms, namely, Count Distribution algorithm, Data Distribution algorithm, and candidate distribution algorithm which focus on minimizing communication. These algorithms were modified Apriori algorithms proposed by the same authors in 1993 to suit the parallel data mining context. In another effort authors presented an encoding scheme that facilitated the fast discovery of association rule mining [1]. The count distribution algorithm achieves this at the expense of carrying out redundant computations in parallel, whereas the count distribution algorithm effectively utilized the main memory and the candidate distribution algorithm exploits the semantics of the problem to reduce synchronization and load balancing efforts. In a distributed environment it is expensive to broadcast messages across the sites.

"A Fast Distributed Algorithm for Mining Association Rules" by Cheung et al. [17], 1996 presents a distributed algorithm that makes use of local and global pruning techniques to reduce the number of candidate itemsets. It presents an algorithm that reduces the time complexity of message support from $O(n^2)$ in case of a straight adaptation of Apriori algorithm to $O(n)$.

"Parallel Data mining for Association Rules on Shared Memory Multiprocessors" by Zaki et al. [78] in 1996, describes a parallel approach to data mining on shared memory multiprocessors, which is yet another hash-based approach. The contribution of the present report included optimization of joining, pruning, balancing of hash trees and concluded on considerable performance gains.

"Scalable Parallel Distribution Algorithm" by Han et al. [38] in 1997 came up with the next set of parallel data mining algorithms as an improvement on the data distribution algorithm, namely the "intelligent data distribution algorithm" and "hybrid distribution algorithm", the latter being an improvement over the former. The algorithms address the concerns regarding memory, communication overhead and redundant communication.

In 1997, Zaki et al. [79,80] published "Parallel Algorithms for Discovery of Association Rules", which discusses itemset clustering techniques with respect to parallel data mining. The itemset clustering technique is a new algorithm proposed by the same authors for approximating the potentially large frequent itemset. The algorithms also make use of efficient traversal techniques to generate the candidate itemsets.

Cheung et al. [19] proposed FPM (Fast Parallel Mining) for mining association rules on a shared nothing platform in their report entitled "Effect of Data Skewness in Parallel Mining of Association Rules" in 1998. The FPM employs the count distribution algorithm adoption with two new candidate pruning techniques called the distributed pruning and global pruning. They particularly tested the new approach to data skew issues and found that the distributed pruning is very effective in handling a high degree of skewness in data whereas the global pruning technique is very effective when mild skewness is observed.

Cheng et al. [18] proposed yet another parallel algorithm called the Asynchronous Parallel Algorithm (is also referred as APM-Adaptive Parallel Algorithm) that was studied on a shared memory multiprocessor in their report entitled "Asynchronous Parallel Algorithm". This effort focused on the specific need of the shared memory system, where communication is no longer an issue as in the case of a shared nothing distributed system. However, the shared I/O becomes a bottleneck as the processors seek access to the partitions during the iterations. An adaptive interval configuration technique suggested in the present report generates interval configuration such that the created partitions have a high homogeneity of inter-partition itemset distribution.

In 1999, a survey report by Zaki [77], entitled "Parallel and Distributed Associated Rule Mining" presented a comprehensive summary of the similar efforts carried out until 1999. The present report lists the efforts, the differences in approaches and a future outlook thereon.

"Parallel Data Mining for Association Rules on Shared-memory Systems" written in 2001, by Zaki et al. [59] presents a sequential

association mining algorithm that is designed especially for a shared memory multiprocessor environment. Given the fact that the data structures like the hash trees suffer from suboptimal data locality, this algorithm explores novel means for load balancing the computation in the enumeration of frequent associations. They also present an optimization scheme for balancing the hash tree data structure to improve locality and reduce false sharing.

In 2002, Pham et al. [62] presented a distributed algorithm for association rule mining, which adopted the Apriori algorithm for a distributed environment employing the mobile agent (MA) Technology. The present report showed the experimental evaluation of the algorithm to prove advantages of the approach.

In 2003, another effort presented by Veloso et al. [70] described a parallel and distributed frequent itemset mining on dynamic data sets. This is the first effort that considers the presence of a dynamic and distributed data set. The presented algorithm maintains the required information despite data updates, without examining the entire database. This algorithm handles the problem of parallelizing the incremental algorithm. In another attempt to run frequent itemset mining in a parallel manner, the authors modified a trie based algorithm [75]. In this attempt, the input instructions were read by a parallel computer. The authors of the paper [76] presented an effort where they attempted to build on a data structure that was previously used in a sequential algorithm and found that trie structure performed well in the parallel context.

"Parallel Leap" is one of the first efforts to run pattern mining on a massive distributed framework [25]. The research effort presented in VLDB 2007, by Liu et al. [45] presents a FPG (Frequent Pattern Growth) algorithm on a modern multicore processor to overcome the underutilization of the multicore processor. The report presented an overwhelming 400% improvement in speedup compared to the state of the art algorithm. It describes a new technique called cache conscious FP Array and makes use of a lock-free data set tiling technique.

"Toward Terabyte Pattern Mining—An Architecture Conscious Solution" was presented by Buehrer et al. [8] which discussed the need for taking fully into consideration the architectural capabilities of a system for designing an algorithm. The proposed approach adapts the FPGrowth algorithm and leverages its capability. It employs optimization techniques for improving cache, memory and I/O utilization using pruning and tiling techniques, and efficient data placement strategies. The pointer based nature of tree structure implementation makes the algorithm all the more efficient. The strip marshaling and merging mechanism proposed in the present report makes serialization and merging of local trees efficient. It minimizes synchronization between nodes and reduces data size at each node.

Craus et al. [21] presented the generalized parallel algorithm in 2008 that gave itself to parallelism in a much better manner compared to other algorithms. While most of the parallel algorithms were developed on the existing linear algorithms, the authors of this contribution devised new algorithms that would make parallelization much effective. Unlike the Apriori algorithm, in this case the communication pattern would be known before the algorithm starts and that gave a big advantage. The set of transactions can be distributed to processors before the beginning of algorithm. The algorithm strategy makes sure that a processor is free to do the itemset computation without having to wait for other processors to finish.

2.2. Other machine learning approaches

While association rule mining recorded the highest number of research efforts among the parallel machine learning efforts, there were other efforts that focused on other popular machine learning concepts. The initial attempts include approaches to convert the existing machine learning approaches to suit a parallel programming scheme, introducing parallelism into typical tasks related to machine learning such as cross-validation. We also witness efforts on performance optimization, development of frameworks and adaptation of different parallel architectures/environments to machine learning. It is difficult to collect and comprehend relevant information in this section is difficult as one has to bring together all the machine learning efforts (other than ARM) done using parallel approaches. Since there are many techniques that may be labeled as machine learning, it is a tough task to consolidate information under this head. However, it is understood that other machine learning approaches are not widely studied like ARM. An effort is made here to consolidate such research. It has been observed that, usually decision tree classifiers, SVMs and neural networks are tried to adopt parallel approach.

2.2.1. Noted publications and contributions

One of the earliest efforts in this direction was presented in K. Thearling [69] that presented a massively parallel architecture and the algorithms for analyzing time series data. According to the author, massively parallel architectures are useful in both memory-based and computation based algorithms. The report described the method to run memory-based reasoning (MBR) applications. Typical MBR applications that use megabytes of data can easily be processed on this massively parallel architecture. It also presented the method to run computationally complex problems such as neural networks and genetic algorithms.

In 1998, Joshi et al. [42] presented a scalable parallel algorithm for decision tree-based classification. This was designed to overcome the latency involved in making the splitting decision, especially when attributes are continuous. It presents a mechanism to construct and search a distributed hash table, when there are many values to be hashed.

Narlikar [52] presented a parallel, multithreaded decision tree builder that adopted a high level, fine grained, formulation for refining C4.5 classifier. It uses lightweight threads that are dynamically created and destroyed to bring about better performance. The divide and conquer approach of C4.5 is employed to introduce parallelism.

Another report by Zaki et al. [81] presented parallel classification using the decision tree approach on shared memory processors. The data parallelism is based on the attribute scheduling among the processors. The task pipelining and dynamic load balancing also contributed to the performance improvement.

Boxer et al. [7] presented a report on parallel algorithms for pattern recognition that is scalable for certain patterns in a Euclidian space. This effort is the first one to apply a parallel approach to pattern recognition

Bowyer et al. [6], also presented a parallelized version of the C 4.5 decision Tree algorithm, that was ported to work on an ASCII Red Parallel Supercomputer. It allows test results from a validation test set to be stored as weights in the leaves of the tree. The algorithm design allowed the processors to work independently without communicating with each other.

Cross-validation is an important task in machine learning that measures the effectiveness of the algorithm. It is also highly time intensive involving many iterations of learning. Celis et al. [14] presented a parallel approach for cross-validation in Weka, a popular machine learning tool. The n folds of cross-validation are made to run on n different processors and results are then integrated to achieve considerable speed.

Jin and Agarwal [41] presented a generic approach to parallel decision tree classification that could be adopted in other machine learning approaches such as association rule mining. The method is based on the AVC groups (combination of AVC (attribute value class label) sets for all attributes) that comprise of three subgroups,

namely, small, concise and partial AVCs. The parallelization is brought about by working independently on the subgroups.

There are quite a few efforts on training and building an SVM classification engine by employing parallel approaches. Cascade SVM, presented by Graf et al. [36] is a parallel approach to SVM training that needs a mention here. This approach spoke about splitting the large data into smaller sets and then running multiple SVMs on them. The partial results are combined and filtered in a cascade of SVMs until the global optimum is reached. Another similar objective was presented in the report [71]. The authors proposed a hierarchical and parallel approach for training SVMs. The approach was successful in achieving speedup in training while reducing the number of support vectors and maintaining generalization accuracy. A book [5] published in 2007 discussed in depth the algorithmic and computational issues associated with optimization strategies for solving the SVM dual problem. The authors also examine the parallel approaches in two chapters of the same book.

A parallel approach to evidence propagation in a probabilistic inference in a Bayesian network was shown in Xia et al. [74]. The authors employed a re-rooting approach along with collaborative scheduling, exploiting both task and data parallelism. On an eight core machine the speedup achieved was almost eight times.

An attempt run SVM on Java Optimized Platform (JOP) was demonstrated by Pederson et al. [60]. This work adopted a class-based approach.

A recent report on a streaming parallel decision tree algorithm that is capable of processing possibly infinite data is presented by Ben-Haim et al.. It processes streaming data in a breadth first mode using horizontal parallelism. At the core it builds histograms in an online manner and then these histograms are used for decision making on new tree nodes at the master processor [3].

A significant contribution is observed in the area of Neural Networks too. The reports [65,47,72,29,53,23,5,48] have recorded such attempts. A survey of ANNs on massively parallel architectures has been presented in [65] in the year 2002. Report [47] presents an attempt to implement a scalable NN on a massively parallel architecture, with an objective to carry out pattern recognition applications. This attempt is a fairly recent one reported in the year 2005 (an updated version was published in 2008). In this approach, neurons in each layer are distributed equally over all processors, however, all input is fed to each processor. Most of the work presented in this section exploits the ability of parallel platforms to perform matrix/vector manipulations quickly. Lastly, the work in [23], talks about a parallel implementation for a cluster system. This used a network parallel training approach to implement the same. Task decomposition for addressing a K-class classification problem was achieved by dividing it into a series of ${}^{n}C_{r}$ 2-class problems. Each of the two 2-class problems can be learned in parallel and later the trained modules are integrated according to the module combination principles.

3. Data mining, machine learning and related efforts on GPUs

In the time frame 1999–2000, scientists toyed around with the idea of using graphics processors for general purpose computing. The machine learning research was quick to respond and we see a number of interesting and innovative research initiatives. The survey report [54,55] on the adaptation of GPU for general purpose computation identifies the map reduce paradigm, matrix manipulation, query processing, databases and data mining as the areas of traction. In this section we make an effort to look at the machine learning aspect since the year 2000.

3.1. Major observations

Using graphics processors for general purpose computation seemed to be a definite trend during this decade. Among the several other applications, popular use of graphics processors included large matrix/vector operations molecular dynamics, signal processing, ray tracing, simulation, sequence matching, speech recognition, databases, sorting, searching and medical imaging. As is very obvious, many of the above-mentioned tasks such as large matrix/vector operations sequence matching, databases, sorting, searching and medical imaging are closely related to machine learning and the contributions in these areas in turn took the machine learning research a step forward. Although the initial years until 2004 did not see the adaptation of GPU architecture for machine learning as a major trend, the later years saw some of the very significant work in machine learning and data mining.

In this section, a summary of the work that contributed to machine learning research indirectly is presented. Fast string matching, matrix operations, stream mining, searching, sorting and database operations are typical problems addressed in this category. Matrix manipulation being a very important part of image processing is one of the usually discussed problems on GPUs. While some emphasized on speedy matrix manipulation for visualization/image/pixel processing, certain efforts focused on matrix manipulation for numeric methods, which are useful from the point of view of machine learning implementations.

3.2. Machine learning and related techniques on GPUs: major publications and their contributions

Amongst the general purpose computation applications of GPUs, machine learning is considered to be the most prominent one. In this section, an attempt is made to present all the research efforts made to explore the possibilities of using GPUs for machine learning until now. Given that, the GPUs understand only graphics data, data related to general purpose computation problems needs to be expressed in terms of graphics data in order to be solved. It is observed that in the initial years, the basic functionalities such as matrix multiplication and other matrix manipulation, string matching, sorting, query processing and other database operations etc., required for running the complex algorithms were implemented. These efforts are a step toward machine learning adoption. In order to show the gradual progress over the years these efforts are also mentioned in the present report, although they do not discuss the machine learning algorithms directly. The forthcoming years saw the development of complete machine learning algorithms on the GPU platform. All of these efforts presented in this Section do not utilize the map reduce framework as it is intended to present such efforts separately. It is interesting to observe that a lot of work spanning across a large spectrum of algorithms has been done in a short span. In depth studies relating to a particular subject are not observed. These studies cover a broad spectrum of topics rather than a deep understanding of a certain topic.

One of the earliest attempts on graphics processors closely related with machine learning is the attempt to run matrix multiplication, a basic computation element of most of the machine learning algorithms [43]. In the present report the authors make an attempt to adapt the technique from parallel computing and perform a portion computation at each processor. In fact, this was the first attempt to use the graphics processor for general computation, when it was believed that GPUs are not meant for such operations. It was shown that GPUs gave a competitive performance and it triggered many attempts in that direction.

A conjugate gradient and a multi grid solver were implemented on a GPU in [4] and it was shown that GPUs can be successfully used like a streaming processor with high floating point performance. In the year 2009, an attempt to fast string matching on GPU was presented by Schats et al. This attempts the marks the beginning of text processing on GPUs.

Among the first sorting algorithms to be implemented on a GPU is the work presented in [31], where a cache efficient sorting algorithm that can be used in database and data mining contexts was presented. The authors used the texture mapping and blending capabilities of the GPUs to the bi-tonic sorting. The parallelism and the vector processing capabilities of the GPUs were exploited to establish this. Fat string matching algorithm given by [64] is one of the first attempt to use GPUs for string processing.

Fang et al. [35] presented a fast and approximate stream mining algorithm for construction of €- approximate quantile and frequency summaries. They used sorting as the computational element for histogram construction. The sorting algorithm developed in this context used the periodic balanced sorting network and it exploited the high computational power and the memory band width of GPUs. This is one of the earliest efforts where massive data collected by logs, sensor networks and web tracking is processed as streams generated by periodic queries and to compute numerical statistics.

Guha et al. [37] presented an informative seminar on utilizing the prowess of GPUs for data mining and visualization to KDD audience. This tutorial presented the methods of exploiting the parallel architecture of GPUs for typical clustering and classification algorithms, regression and other typical statistics analysis, matrix/vector operations, data streaming, analysis and view. It also familiarized the audience with the tools and applications for processing data as streams with pixel level parallelism on GPUs.

''Scalable Clustering Using Graphics Processors'' by Cao et al. [9] is the first to modify a clustering algorithm for GPU, making it scalable to process huge data. The authors identify the components in clustering algorithms that are computationally expensive and modify them suitably to run on graphics processors. The work presented is based on a K means algorithm, where distance computation and comparison are expensive operations. The authors propose a new distance measuring scheme based on the fragment vector processing and multi-pass rendering. The algorithm minimizes the data transmission between the CPU and GPU taking into account the low bandwidth.

A new document clustering algorithm inspired by nature called, 'the flocking-based algorithm' is presented in the report by Charles et al. [15] in the Journal of Undergraduate Research. In this agent based approach, the organization of the document emerges through an interaction amongst a group of agents. The similar documents flock together and loosely organize themselves according to the subject. The authors implemented and tested the algorithm on both sequential and parallel platforms and compared the performances. Although GPU outdid the single core performance, the method posed certain restrictions on the number of documents that could be processed.

GPUTeraSort, an application that performed sorting on billions of database records was presented by Govindaraju et al. [32] in ACM SIGMOD in 2006. This is one of the reports that boosted the large scale computational efforts on GPUs. The sorting algorithm used both data and task parallelism and achieved high performance on GPU and showed that terabytes of data can be processed at the cost of few pennies. In the year 2007 too, a database query processing effort on GPU was presented in ACM SIGMOD [26]. The same research group presented another report in the next ACM SIGMOD in 2008 on performing relational joins on GPUs. These three reports are marked as milestones in database processing on GPUs. This group started their work on GPUs as early as 2005 where they presented efficient sorting algorithm for database operations [33,34]. One more effort on efficient scatter and plot operations on GPU was also presented in 2007 [40].

Another report presented in the Journal of Parallel and Distributed Computing authored by Che et al. [16] discussed a performance leap achieved by an implementation of a K Means clustering algorithm on GPUs among the several other general purpose applications of GPU.

GPUminer presented by Fang et al. [27] discussed a parallel data mining system that implemented K means clustering and Apriori frequent pattern mining algorithms. Although the authors did not compare the performance of their K means algorithm with the previous implementations, they presented the performance studies on a quad-core CPU and GPU. This attempt specially focused on architecture focused implementation where the data storage and buffer management is based on CPU, parallel mining is based on co-processing and visualization is based on GPU.

An SVM training and prediction method was implemented by A. Carpenter and was presented in [10]. The second order heuristic employed significantly reduces iterations required for the running the module for decomposing and solving the quadratic program. However, the method proves to be efficient as the number of data points increase. Surprisingly, this attempt preceded by Catanzaro et al. [11], that used the map reduce paradigm to develop an SVM solver. This used CUDA, the specialized language for programming on GPU.

Latent Semantic Analysis, a technique usually used for reducing the dimension of term-document data sets using singular value composition (SVD) was designed to run on GPU by Cavanagh et al. [13]. They chose the Lancozs algorithm that tridiagonalizes a matrix for computing SVD faster as it involves a matrix-vector. This attempt was also one of the earliest CUDA implementations. The limitations of the system and the used language were found to be the constraints for leveraging the benefits of such an implementation to the fullest.

An effort to optimize the K means clustering algorithm was presented in Zechner et al. [82]; this time leveraging the benefits of coding on CUDA. The present report leveraged the architectural aspects of the platform to overcome the shortcomings of the previous implementations. The distance calculations were parallelized on GPUs and the sequential updating of centroids were done on a CPU.

An effort to implement the Apriori algorithm for frequent itemset mining (FIM) on GPU hardware was shown in Fang et al. [28]. The authors demonstrated two ways of implementing the FIM; one of which used the GPU alone while the other uses both the GPU and CPU memory. They used a bitmap-based approach that made use of a trie for demonstrating both the above-mentioned approaches. They also implemented a pure bitmap-based approach to run a GPU implementation.

Building the TF/IDF representation from raw text data is an important task involved in text mining. The problem of ranking documents based on TFIDF search was accelerated using GPU by Zhang and Potok [85]. The report was one of the first reports to work on text mining. It mainly dealt with introducing parallelism to the process of building a token frequency hash table for each document. This requires that preprocessing tasks like tokenizing, stemming, etc. be made to run in a parallel manner. A global token frequency hash table is then built and TF/IDF values are calculated to find the relevance of a document with respect to a topic.

According to [73], one can achieve a fair amount of success in processing the very large data sets that do not fit in GPU memory. Here, the authors primarily investigated the viability of processing very large data sets on GPUs for practical implementations. They demonstrated the popular K means clustering algorithm and measured the performance against the 'Minebench' benchmark to prove the efficiency of their approach.

In 2010, the data/document clustering problem was addressed by three reports [22,83,84]. It is interesting to note that these contributions that appeared in noted journals/conferences came from the same group. The large scale data clustering solution presented in [22] was based on a flocking algorithm mentioned in one of their previous works [15]. The present report described methods to overcome the shortfalls experienced with the previous work where the large size of the document information caused frequent GPU global device memory reads. The global memory device is not cached and has a delay of hundreds of cycles per read. In order to encounter this problem, some document terms were cached in shared memory for faster access. The initial approached was also replaced by a new method that used a document similarity matrix that could be used to read directly for document comparison.

The flocking algorithm used in [15] is not regarded as one of the machine learning techniques; however; the purpose of document clustering is typically addressed by text mining, a machine learning approach. A highly parallelized approach for calculating the TF/IDF values for gathering similarity of documents as in text mining was demonstrated by the same research group in [85]. They employed the flocking-based simulation algorithm for document clustering using the TF/IDF values for similarity comparison in [83] and showed a performance lift. Another similarity measure TF/ICF proposed in yet another report was also employed here as computation of TF/IDF on parallel architecture did not improve the performance to a great extent. In summary, the present report archived several approaches and combined machine learning aspects with simulation algorithms and suggested alternates for document clustering.

The best performing combination in the above-mentioned effort; namely the flocking-based clustering similarity measurement calculated using the TF/ICF was implemented on a four node GPU cluster and presented in [84]. The ML algorithms based on kernel methods scale poorly as the data size increases. This problem was addressed by Srinivasan et al. [68] by employing GPUs partially. The GPUs are typically used to accelerate matrix operations such as decomposition, product and their iterative formulations, which are amply used in kernel methods.

Three attempts on implementing decision trees (of different nature) on GPUs are presented in [66,61,30]. The report presented in [66] demonstrated an object recognition motive. They developed a strategy to transform forest data structure from a list of binary trees to a 2D texture. The implementation was a hundred times faster compared to the CPU counterpart. A tree-based context clustering is demonstrated in [61]. Node splitting in the decision tree is carried out independent of all other splits, gaining considerable speed. A tree level parallelism was adopted in [30] to build extremely randomized trees.

An attempt to implement Neural Networks [49] on GPUs is reported in Ly et al.. Unlike the normal parallelization approaches, this approach focused on getting the element matrix operations optimized so that the overall gain in performance is met. The report showed that the implementation gain was about 66 fold, although the element operations sped up more than 1000 fold, because of the communication overhead.

The work reported by Gneuron [63] is yet another attempt to build neural networks on GPUs, where the authors made an attempt to identify the hotspots that are critical and tried to parallelize the same. The focus was on the elements of operations within the training cycle, but the training cycle was not parallelized.

4. Data mining and machine learning using map reduce technique

The map reduce technique popularized by Google Inc. has brought a paradigm shift to the way in which data is handled in distributed/parallel environments. We see the adoption of map reduce technique right from the multicore era. Clearly, with the advent of GPUs, the map reduce technique turned out to be an effective tool to elevate parallel performance. In fact, the number of reports we find in this section is quite small, indicating the scope for research in this space. This section covers the efforts that engage the map reduce technique in a parallel environment including multicores and GPUs. The literatures studied under this Section were published from 2005 onwards.

4.1. Major observations

The map reduce technique has been popular for quite some time. However, it has been explored in a very limited manner. Most of the efforts have been triggered from a need to perform better in a certain application context. Therefore, there remains a lot of scope to explore the algorithmic aspects.

4.2. Parallel data mining/machine learning using map reduce technique—major publications and their contributions

The very first implementation of a map reduce framework on a multicore environment and subsequent demonstration of a machine learning algorithm was done by Chu et al. [20]. The authors presented a broadly applicable map reduce framework for running most of the machine learning algorithms. The framework is based on the premise that all algorithms can be expressed in summation form. The computations are done by different mappers and the final summation is carried out by the reducer. The authors demonstrated the working of LWLR, LR, Naïve Bayes, PCA, Support Vector Machine, ICA, GDA, ICA and Neural network etc. on this framework.

Both He et al. [39] and Catanzaro et al. [12] developed a map reduce framework on GPUs using CUDA. While Catanzaro et al. demonstrated use of this framework by implementing SVM on it; He et al. demonstrated functions such as string matching, matrix multiplication, building of inverted index, etc.. He et al. developed a set of APIs similar to those of a CPU-based map reduce implementation and hid the programming complexities on GPU considerably. The framework developed by Catanzaro et al. required the user to define a map function, a set of reduce operators and a cleanup function operates on the results of reduction. DisMarc is one of the recent efforts that exploits the computing prowess of GPUs to process huge data on a map reduce framework [50]. It successfully hides the complexity of GPU programing behind a simple map-reduce interface.

A parallel FP Growth algorithm was presented by Li et al. [46]. An attempt to parallelize a FP Growth, a popular FIM algorithm was done here. The algorithm partitions computations in such a way that each machine carries out a set of independent mining tasks, eliminating the computational dependencies between the machines. A linear speed up was shown.

Lin et al. [44] published a book on "Data-Intensive Text Processing with map reduce" in 2010. In this book the author provides a detailed explanation of the map reduce system and the method to implement various text processing approaches on them. They discuss the whole approach on Hadoop distributed architectures to emphasize the parallel advantage brought about by map reduce.

An attempt to implement a massively parallel ensemble of decision trees with map reduce has been presented in [56]. A map reduce job to find the best split when there is too much data to fit in the memory was defined. The map reduce framework was also employed to grow the entire tree as long as it fits in the memory.

5. Critical gaps, open problems and scope for research

One important observation is that breadth and depth of research in ARM space is very high and it follows a continuous record. This seemed to be natural in a way; as ARM is required to process huge databases and indicated the need adopt parallel approaches to reduce processing time. However, other machine learning approaches have been studied rather intermittently, although decision trees and neural networks seem to have a fair share too. More than academic innovation, application scenarios seem to have driven research in this space. It is important to note that there is plenty of scope to work on these areas.

We note that an effort to consolidate these efforts with respect to performance improvement, memory utilization, processor utilization etc., would give a comprehensive picture. It requires a humongous effort to realize this, as these reports correspond to efforts on varied hardware with different specifications. It may not make sense to compare each of them against the other, however, it would be worth consolidating them under broader categories.

A lot of work has been done on the theoretical front and most of the performance issues are well addressed. However, the most important gap the author sees is at the application front, where these algorithms are deployed in actual business contexts. The business world witnesses very few scenarios where the benefits of parallel machine learning are actually realized. With the advent of cloud environment, there is a need for real life applications to be modified for the cloud context. The need for working with a lot of data should open up a lot of opportunities for deploying parallel algorithms.

The map reduce framework has not yet been fully exploited in the machine learning context. Given that the map reduce framework can greatly enhance the performance of applications on GPUs, such a combination would be a boon to the machine learning age. This would make analysis, data visualization and prediction run on desktops without latency, irrespective of the amount of data that needs to be handled. The greatest advantage is the reduction of cost brought about by GPUs.

The next natural step is to work on distributed environments such as GPU or GPU/CPU clusters and the cloud using a distributed framework for map reduce such as Hadoop. In a practical sense, GPUs have performed better than multicores and are much less expensive. In the context of the cloud too, where the cost of hardware is a concern, it makes sense to adopt GPUs to take care of the cost and speed concerns.

Big data analytics is gaining prominence today. While the techniques for handling the size of data are mature, techniques for deriving semantics from a large amount of data are conspicuously absent. It is very important to invest efforts in bringing out the semantics of the data, where both data mining and parallel approaches have a great role to play. Knowledge extraction powers of machine learning need to be employed to learn from large, unstructured data in a distributed manner. This is another direction yet unexplored. This is a big step toward realizing the semantic web and knowledge extraction by mining the unstructured sources.

6. Conclusion

Machine Learning has become more relevant today than ever, given the availability of a huge amount of information and the need to analyze and predict business insights. This necessitates that the speed of processing information is increased exponentially. Needless to say, parallel approaches make a lot of sense in this context. An effort to understand the parallel approaches to machine learning has been made here.

We have attempted to present a holistic picture of parallel machine learning efforts for the past one and a half decade. We have grouped the efforts based on the influence of new technologies that were introduced in this period. Accordingly, we have classified these efforts as; the ones that did not consider both map reduce technique and GPUs (Year 1999–2000 and beyond), the ones that were employed on GPUs (Year 2000–2005 and beyond) and the ones that used the map reduce technique (Year 2005 onward). A few of the efforts that discussed the map reduce technique on GPU were included in the last category.

We observe that there are a good number of efforts on realizing greater performance both in terms of speedup and memory utilization. However, there are fewer application contexts. Our study shows that there are many opportunities to work on both algorithmic and application aspects of machine learning on GPU and on map reduce frameworks and on a combination of both. Of course, the cloud-based aspects of machine learning are almost unexplored and this definitely is the direction to look forward to. Last but not least, aspects of knowledge extraction from large and unstructured data, particularly the web, for realizing semantic web offers open opportunities for research. This capability will go a long way into benefitting from the collaborative efforts that work on realizing the semantic web.

References

[1] R. Agrawal, et al., Fast discovery of association rules, in: Advances in Knowledge Discovery and Data Mining, AAAI Press, 1996.
[2] R. Agrawal, J. Shafer, Parallel mining of association rules, in: IEEE TKDE, 1996.
[3] Y. Ben-Haim, E. Yom-Tov, A streaming parallel decision tree algorithm, Journal of Machine Learning (2010).
[4] J. Bolz, I. Farmer, E. Grinspun, P. Schr¨oder, Sparse matrix solvers on the GPU: conjugate gradients and multigrid, in: ACM SIGGRAPH, 2003.
[5] L. Bottou, O. Chapelle, D. DeCoste, J. Weston (Eds.), Large Scale Kernel Machines, The MIT Press, 2007.
[6] K.W. Bowyer, L. Hall, T. Moore, N. Chawla, A parallel decision tree builder for mining very large visualization datasets, in: IEEE Conference on Systems, Man and Cybernetics, 2000.
[7] L. Boxer, R. Miller, A. Rau-Chaplin, Scalable parallel algorithms for geometric pattern recognition, Journal of Parallel and Distributed Computing (1999).
[8] G. Buehrer, S. Parthasarathy, S. Tatikonda, Toward terabyte pattern mining—an architecture conscious solution, in: 12th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, 2007.
[9] F. Cao, A.K.H. Tung, A. Zhou, Scalable clustering using graphics processors, in: WAIM 2006, 2006.
[10] A. Carpenter, CUSVM: a CUDA implementation of support vector classification and regression, 2009. http://patternsonascreen.net/cuSVMDesc.pdf.
[11] B. Catanzaro, N. Sundaram, K. Keutzer, Fast support vector machine training and classification on graphics processors, in: ICML 2008, 2008.
[12] B. Catanzaro, N. Sundaram, K. Keutzer, A map reduce framework for programming graphics processors, in: PACT 2008, 2008.
[13] J.M. Cavanagh, T.E. Potok, X. Cui, Parallel latent semantic analysis using graphics processor unit, in: 11th Annual Conference Companion on Genetic and GECCO'09, Evolutionary Computation Conference: Late Breaking Papers, 2009.
[14] S. Celis, D.R. Musicant, Weka parallel: machine learning in parallel, Technical Report, Carlton College, CS TR 2002, 2002.
[15] J.ST. Charles, R.M. Patio, T.E. Potok, X. Cui, Flocking-based document clustering on the graphics processing unit, US department of energy, Journal of Undergraduate Research (2006).
[16] S. Che, M. Boyer, J. Meng, D. Tarjan, J.W. Sheaffer, K. Skadron, A performance study of general-purpose applications on graphics processors using CUDA, Journal of Parallel and Distributed Computing (2008) 350–371.
[17] D. Cheung, et al., A fast distributed algorithm for mining association rules, in: International Conference Parallel and Distributed Information Systems, 1996.
[18] D. Cheung, K. Hu, S. Xia, Asynchronous parallel algorithm for mining association rules on shared-memory multi-processors, in: ACM Symposium on Parallel algorithms and Architectures, 1998.
[19] D. Cheung, Y. Xiao, Effect of data skewness in parallel mining of association rules, in: Pacific-Asia Conference Knowledge Discovery and Data Mining, 1998.
[20] C. Chu, S.K. Kim, Y. Lin, Y. Yu, G. Bradski, A.Y. Ng, K. Olukotun, Map-reduce for machine learning on multicore, in: NIPS, 2006.
[21] M. Craus, A. Archip, A generalized parallel algorithm for frequent itemset mining, in: 12th WSEAS International Conference on Computers, 2008.
[22] X. Cui, J. Beaver, J.S. Charles, T. Potok, The GPU enhanced parallel computing for large scale data clustering, in: Book Chapter of GPU Computing Gems 2010, Springer, 2010.

[23] G. Dahl, A. McAvinney, T. Newhall, Parallelizing neural networks training for cluster systems', in: PDCN, 2008.
[24] A. DI Blas, T. Kaldeywey, Why graphics processors will transform database processing? Data monster, 2009. http://spectrum.ieee.org/computing/software/data-monster/0.
[25] M. El-Hajj, O. Zaiane, Parallel leap: large-scale maximal, pattern mining in a distributed environment, in: ICPADS, 2006.
[26] R. Fang, B. He, M. Lu, K. Yang, N.K. Govindaraju, Q. Luo, P.V. Sander, Gpuqp: query co-processing using graphics processors, in: ACM SIGMOD 2007.
[27] W. Fang, K.K. Lau, M. Lu, X. Xiao, C.K. Lam, P.Y. Yang, B. He, Q. Luo, P.V. Sander, K. Yang, GPUMiner: parallel data mining on graphics processors, Technical Report HKUST-CS08-07, Oct 2008.
[28] W. Fang, M. Lu, X. Xiao, B. He, Q. Luo, Frequent itemset mining on graphics processors, in: 4th International Workshop on Data Management On New Hardware, 2009.
[29] P. Farber, K. Asanovic, Parallel neural network training on multi-spert, in: IEEE 3rd International Conference on Algorithms and Architectures for Parallel Processing, 1997.
[30] M. Geary, H. Lee, D. Signorelli, J. Vaughan, Implementing extremely randomized trees in CUDA, Technical Report, Stanford University, 2009.
[31] A. Ghoting, G. Buehrer, S. Parthasarathy, D. Kim, A. Nguyen, Y. Chen, P. Dubey, Cache-conscious frequent pattern mining on a modern processor, in: VLDB, 2005.
[32] N. Govindaraju, J. Gray, R. Kumar, D. Manocha, GPUTeraSort: high performance graphics co-processor sorting for large database management, in: ACM SIGMOD, 2006.
[33] N.K. Govindaraju, B. Lloyd, W. Wang, M. Lin, D. Manocha, Fast computation of database operations using graphics processors, in: ACM SIGMOD, 2004.
[34] N.K. Govindaraju, N. Raghuvanshi, M. Henson, D. Tuft, D. Manocha, A cache-efficient sorting algorithm for database and data mining computations using graphics processors, Technical Report, University of North Carolina, 2005.
[35] N.K. Govindaraju, N. Raghuvanshi, D. Manocha, Fast and approximate stream mining of quantiles and frequencies using graphics processors, in: ACM SIGMOD, 2005.
[36] H.P. Graf, E. Cosatto, L. Bottou, I. Durdanovic, Vladimir Vapnik, Parallel support vector machine: the cascade SVM, in: NIPS, 2005.
[37] S. Guha, S. Krishnan, S. Venkatasubramanian, Tutorial.: data visualization and mining using the GPU, KDD, 2005.
[38] E.H. Han, G. Karypis, V. Kumar, Scalable parallel data mining for association rules, in: Proceeding of the ACM Conference Management of Data, 1997.
[39] B. He, W. Fang, Q. Luo, N.K. Govindaraju, T. Wang, Mars: a map reduce framework on graphics processors, in: PACT 2008, 2008.
[40] B. He, N.K. Govindaraju, Q. Luo, B. Smith, Efficient gather and scatter operations on graphics processors, in: SC 2007, 2007.
[41] R. Jin, G. Agarwal, Communication and memory efficient parallel decision tree construction, in: Third SIAM Conference on Data Mining, 2003.
[42] M.V. Joshi, G. Garypis, V. Kumar, ScalParC: a new scalable and efficient parallel classification algorithm for mining large datasets, in: International Parallel Processing Symposium, 1998.
[43] E.S. Larsen, D. Mcallister, Fast matrix multiplies using graphics hardware, in: SC 2001, 2001.
[44] J. Lin, C. Dyer, Data-Intensive Text Processing with Map Reduce, Morgan Kaufmann Book, 2010.
[45] L. Liu, E. Li, Y. Zhang, Z. Tang, Optimization of frequent itemset mining on multiple-core processor, in: VLDB 2007, 2007.
[46] H. Li, Y. Wang, D. Zhang, M. Zhang, E. Chang, PFP: parallel FP-growth for query recommendation, in: ACM Recommender Systems 2008.
[47] L.N. Long, A. Gupta, Scalable massively parallel artificial neural networks, in: AIAA 2005, 2005.
[48] B. Lu, M. Ito, Task decomposition and module combination based on class relations: a modular neural network for pattern classification, IEEE Transactions on Neural Networks 10 (5) (1999).
[49] D.L. Ly, V. Paprotski, D. Yen, Neural networks on GPUS: restricted Boltzmann machines, Technical Report, University of Toronto, 2008.
[50] A. Mooley, K. Murthy, H. Singh, DisMaRC: a distributed map reduce framework on CUDA, Technical Report, University of Texas, 2009.
[51] A. Mueller, Fast sequential and parallel algorithms for association rule mining, a comparison, Technical Report, 1995.
[52] G.J. Narlikar, A parallel, multi-threaded decision tree builder, Technical Report, CMU, CMU-CS-98-184, 1998.
[53] M. Oldroyd, Optimization of Massively Parallel Neural Networks, Fultus Publishing, ISBN: 1596820101, 2004.
[54] J.D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Kruger, A.E. Lefohn, T.J. Purcell, A survey of general-purpose computation on graphics hardware, in: EUROGRAPHICS 05, 2005.
[55] J.D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Kruger, A.E. Lefohn, T.J. Purcell, A survey of general-purpose computation on graphics hardware, Computer Graphics Forum (2007).
[56] B. Panda, J.S. Herbach, S. Basu, R.J. Bayardo, Massively parallel learning of tree ensembles with map reduce, in: VLDB, 2009.
[57] J.S. Park, M. Chen, P.S. Yu, Efficient parallel data mining for association rules, in: ACM International Conference Information and Knowledge Management, 1995.
[58] J.S. Park, M. Chen, P.S. Yu, An effective hash-based algorithm for mining association rules, in: ACM SIGMOD1995, 1995.
[59] S. Parthasarathy, M.J. Zaki, M. Ogihara, W. Li, Parallel data mining for association rules on shared memory systems, Journal of Knowledge and Information Systems (2001).
[60] R.U. Pedersen, M. Schoeberl, Short paper: object oriented machine learning with a multicore real-time java processor, in: JTRES 10, 2010.
[61] N. Pilkington, H. Zen, An implementation of decision tree-based context clustering on graphics processing units, in: Interspeech 2010, 2010.
[62] A.H. P. Nguyen, T.B. Ho, A distributed algorithm for mining association rules, in: The Third International Conference on Parallel and Distributed Computing, Applications and Technologies, 2002.
[63] R.D. Prabhu, Gneuron: parallel neural networks with GPU, in: HiPC 2007, 2007.
[64] M.C. Schats, C. Trapnell, Fast exact string matching on the GPU, Technical Report, Stanford, 2009.
[65] U. Seiffert, Artificial neural networks on massively parallel computer hardware, in: ESANN'02, 2002.
[66] T. Sharp, Implementing decision trees and forests on a GPU, in: ECCV 2008, 2008.
[67] T. Shintani, M. Kitsuregawa, Hash-based parallel algorithms for mining association rules, in: International Conference on Parallel and Distributed Information Systems, 1996.
[68] B.V. Srinivasan, Q. Hu, R. Duraiswami, GPUML: graphical processors for speeding kernel machines, in: Workshop on High Performance Analytics, Algorithms, Implementations and Applications, Seoul Conference on Data Mining, 2010.
[69] K.K. Thearling, Massively parallel architectures and algorithms for time series analysis, in: L. Nadel, D. Stien (Eds.), Lectures in Complex Systems, Addison-Wesley, 1995.
[70] A. Veloso, M. Erick Otey, S. Parthasarathy, W. Meira Jr., Parallel and distributed frequent itemset mining on dynamic datasets, in: HiPC, 2003.
[71] Y. Wen, B. Lu, in: J. Wang, X. Liao, Z. Yi (Eds.), Hierarchical and Parallel Method for Training Support Vector Machines, in: ISNN LNCS, vol. 3496, 2005, pp. 881–886.
[72] I. Wesley-Smith, A parallel artificial neural network implementation, in: NCUR, 2006.
[73] R. Wu, B. Zhang, M. Hsu, GPU accelerated large scale analytics, Technical Report, 2009-38, 2009.
[74] Y. Xia, X. Feng, V.K. Prasanna, Parallel evidence propagation on multicore processors, in: PACT 2009, 2009.
[75] L. Yang, Pruning and visualizing generalized association rules in parallel coordinates, in: IEEE TKDE, 2005.
[76] Y. Ye, C.-C. Chiang, A parallel apriori algorithm for frequent itemsets mining, in: SERA, 2006.
[77] M.J. Zaki, Data mining parallel and distributed association mining: a survey, in: IEEE Concurrency, 1999.
[78] M.J. Zaki, et al., Parallel data mining for association rules on shared-memory multi-processors, in: Supercomputing'96, 1996.
[79] M.J. Zaki, et al., Parallel algorithms for fast discovery of association rules, in: Data Mining and Knowledge Discovery, 1997.
[80] M.J. Zaki, et al., Parallel algorithms for discovery of association rules, in: KDD, 1997.
[81] M.J. Zaki, C. Ho, R. Agrawal, Parallel classification for data mining on shared-memory multiprocessors, in: ICDE, 1999.
[82] M. Zechner, M. Granitzer, Accelerating K means on the graphics processor via CUDA, in: First International Conference on Intensive Applications and Services, INTENSIVE'09, 2009.
[83] Y. Zhang, F. Mueller, X. Cui, T.E. Potok, Data-intensive document clustering on GPU clusters, Journal of Parallel and Distributed Computing (2010).
[84] Y. Zhang, F. Mueller, X. Cui, T. Potok, Large-scale multi-dimensional document clustering on GPU clusters, in: IEEE International Parallel & Distributed Processing Symposium IPDPS 2010, 2010.
[85] Y. Zhang, F. Mueller, X.T. Potok, GPU accelerated text mining, in: Workshop on Exploiting Parallelism using GPUs and Other Hardware-Assisted Methods, March 2009.

Further reading

[1] A. Hoisie, O. Lubeck, H. Wasserman, Performance and scalability analysis of teraflop-scale parallel architectures using multidimensional wavefront applications, International Journal on High Performance Computing Applications (2000).
[2] T.T. Rogers, J.L. McClelland, Pre´cis of semantic cognition: a parallel distributed processing approach, Behavioral and Brain Sciences 31 (2008) 689–749.
[3] J. Shalf, The new landscape of parallel computer architecture, SciDAC 2007, Journal of Physics: Conference Series 78 (2007) 012066.

Sujatha R. Upadhyaya was awarded a Ph.D. in Computer Science from IIT Madras–INDIA, in the year 2007. She worked with Infosys Technologies, Bangalore, India as a senior researcher until January 2011. Presently, she leads the research efforts at Xurmo Technologies, Bangalore. Her research interests span across ontologies, knowledge modeling, machine learning, data and text analytics and performance of algorithms. At the moment she's exploring the applications of large scale machine learning especially on parallel platforms and graphics hardware. She can be contacted at sujatha.upadhyaya@gmail.com