Journal of Engineering and Applied Sciences 13 (5): 1228-1234, 2018
ISSN: 1816-949X
© Medwell Journals, 2018

# Parallel Processing for Data Mining and Data Analysis Applications

[1]Mohammad H. Moattar, [2]Maziyar Taharozi,
[2]Samira Arabi Yazdi and [2]Sodabeh Salehi Rekavandi
[1]Department of Computer Engineering, Mashhad Branch, Islamic Azad University, Mashhad, Iran
[2]Department of Computer Science, Islamic Azad University, Ferdows, Iran

**Abstract:** This study emphasize on how parallelism can be applied in data analysis. In recent decades where the large amount of data is produced by machines: software logs, cameras, microphones, RFIDs, etc. Creation speed rate of these data will increase exponentially with Moore's Law. Saving or storing such amount of data is inexpensive and using some parallel processing methods, the data can also be investigated and mined effectively. So, this study intends to debate about parallel programming procedures used in data analysis and data mining. The key motive for this parallelism is to make analysis more rapidly. This is generally attained by using multiple processors or multiple computers, execution dissimilar aspects of data analysis or mining, performing the tasks alongside and later consolidating the data into a single report.

**Key words:** Data mining, parallel, procedure, map, sharding, reduce

## INTRODUCTION

Data analysis encloses examining, cleaning, make over, showing data with the determination of distinguishing useful information, telling deductions and assistant decision making. It also includes the procedure of scientifically connecting statistical and/or rational ways to describe, demonstration and assess data. Data analysis has several facets and approaches. One of the prevalent procedures is data mining.

### Data mining

**What is data mining?:** Data mining or data detection is the computer-assisted procedure of mining through and considering enormous sets of data and then extracting the connotation of the data. It is precise data exploration procedure which emphases on modeling and knowledge discovery for predictive as an alternative of purely descriptive determinations. So, data mining is also sporadically mentioned as Knowledge Discovery in Databases (KDD).

Traditional procedures of data investigation, based mostly on a person dealing directly with the data do not scale to voluminous data sets. While, database knowledge has provided us with the basic tackles for the proficient storage and lookup of large data sets, the problem of how to help humans evaluate and comprehend large bodies of data leftovers a problematic and unsolved problem. The initial field of data mining assurances to offer new methods and smart utensils to encounter the challenge.

Data mining tackles predict deeds and future trends, let businesses to make proactive, knowledge-driven decisions. It can solve business problems that traditionally were too time-consuming to resolve. They search databases for concealed patterns, finding predictive information that experts may miss because it lies outside their expectations.

**Literature review:** Apiletti *et al.* (2017) introduced PaMPa-HD, a map reduce-based frequent closed itemset mining algorithm for high dimensional datasets. An encient solution has been proposed to parallelize and speed up the mining process. Furthermore, different strategies have been proposed to easily configure the algorithm parameter. The experimental results, performed on real-life high-dimensional use cases, show the effciency of the proposed approach in terms of execution time, load balancing and robustness to memory issues. Simoes *et al.* (2015) compares the performance of two models of load balancing (Proportional and Autotuned algorithms) of the JPPF platform in the processing of data mining from a database with osteoporosis and osteopenia. When performing the analysis of execution times it was observed that the proportional algorithm performed better in all cases. Thabtah *et al.* (2015) overcomed the problem of mining large training data sets by proposing a new learning method that repeatedly transformd data between line and item spaces to quickly discover frequent ruleitems, generate rules, subsequently rank and prune rules. This new learning method had been implemente in a parallel Map Reduce (MR) algorithm

**Corresponding Author:** Mohammad H. Moattar, Department of Computer Engineering, Mashhad Branch, Islamic Azad University, Mashhad, Iran

called MRMCAR which could been considered the first parallel AC algorithm in the literature. The new learning method could been utilised in the different steps within any AC or association rule mining algorithms which scales well if contrasted with current horizontal or vertical methods. The results revealed that MRMCAR is superior to both current AC mining algorithms and rule based classification algorithms in improving the classification performance with respect to accuracy, Yan *et al*. (2017) proposed a parallel mining algorithm of the constrained frequent pattern, called PACFP using the map reduce programming model. First, key steps in the algorithm such as mapping transaction in datasets to frequent item support count, constructing the constrained frequent pattern tree, generating the constrained frequent pattern and aggregating frequent patterns are implemented by three pairs of map and reduce functions. Second, migration of data recording is achieved by applying a data grouping strategy based on frequent item support and load balance is effectively solved while generating the constrained frequent pattern. In the end, their experimental results validate availability, scalability and expandability of the algorithm using celestial spectrum datasets. Xiao (2010) presented survey on the parallelization of well-known data mining techniques covering classification, link analysis, clustering and sequential learning which are the most important topics in data mining research and development. Basic terminology related to data mining and parallel computing was introduced. With each algorithm, he provide a description of the algorithm, review and discuss current research on parallel implementations of the algorithm.

**Why parallel data mining?** Bulks of data are report in both scientific and marketable arenas. A significant commercial curiosity is exist in increasing and refining data mining applications in order to extract valuable information in the form of patterns. Computer systems have been keeping an excessive deal of information including credit card trades, allegiance reward systems record our shopping habits, CCTV cameras, GPS systems embedded in our smartphones and navigation systems record our movement and locations. Large and intricate databases are reported in the commercial zone for specimen Amazon's two largest databases combine 42 TB of data and AT&T's largest database includes 312 TB. Facebook depository has received daily rate of about 600 TB. The depository has seen a 3x progress in the amount of data stored in the last year. In the year 2020, it has been predicted that, the size of our digital cosmos will be 44 times as big as it was in the year 2009. Progresses in storage technology make it possible to store all these data bulks at a very low cost, therefore, the universal trial in data mining is the scalability of data mining ways to these large data volumes.

Data mining approaches that purpose at extracting patterns and information from this large quantity of data are gaining approval in many applications. Data mining procedures are developed to evaluate these bulks of data automatically and deliver users with the intrinsic knowledge in these data without any manual efforts. As the data size has been growing from gigabytes to terabytes or even greater, consecutive data mining procedures may be unsuccessful in delivering the results in sensible amount of time. Also, the partial memory in a single processor can bounds the data that can be held, so causing in large slowing down of the process. Though, using parallel milieu, the processes can very well exploit vast collective memory and processing power of parallel processors. So, the questions associated to execution time and memory necessities faced in sequential processing can be well addressed.

Though, demanding to attain good routine and scalability to large size of data is a very tricky duty. Primarily, it is vital to hire a good data association plan and decomposition strategy, so that, the workload is consistently partitioned among all procedures with negligible data requirement. Second, decreasing synchronization and communication overhead is vital in order for parallel procedures to balance well.

**Parallel data mining proceduers and frameworks:** Performing data mining jobs concurrently named as parallel data mining. The parallelization of a data mining duty often monitors a data parallel way as the computational workload of data mining works is usually directly reliant on the size of data that wants to be administered. The data is divided into smaller subsets, in data parallelization and dispersed to multiple processors on which the data mining jobs are executed instantaneously. Menandas and Joshi (2014) concern with the fast development of networking, data storage and the capacity of data collection, that big data are now rapidly expanding in all science and engineering domain presented a Parallel Processing Technique (PPT) that characterized the features of big data revolution, reduced complexity and proposed a big data processing model from the data mining perspective. This model involves data accessing and computing, data privacy and domain knowledge and big data mining algorithms and also the big data revolution.

**Multiprocessor computer designs:** A multiprocessor design is essential for executing data parallel procedures.

The key impression in data parallelism is to split the workload and allocate it to several processing units. There are two appropriate multiprocessor designs, tightly-coupled designs and loosely-coupled designs, hybrids between the two designs are conceivable.

A loosely-coupled design includes multiple standalone computers. Each computer has one processing unit and its local private memory. This design needs data provision and gathering mechanisms and a communication network. Execution of this design is also, often denoted to as 'Massively Parallel Processors' (MPP).

A tightly-coupled design contains of multiple processors that share a communal memory using a shared bus system. No data provision is requisite as the processors do not have a private memory. Maneuver of this design is also often denoted to as 'Shared memory Multiprocessor machines'.

**Benefits of loosely-coupled architecture:**

- We have no bottleneck as the processors do not segment system bus
- This architecture is tougher to hardware failures as the processors are hosted on diverse computers over the network
- A loosely-coupled scheme can be upgraded regularly by simply substituting long-standing workstations with newer ones

**Weakness of loosely-coupled planning:** It needs communication and collaboration between its computing nodes which introduces an additional overhead for the application.

**Benefit of tightly-coupled planning:** It is generally more proficient at processing data as it prevents data repetition and does not need to transfer information between processing units.

**Weakness of tightly-coupled architecture:**

- There is a bottleneck with the quantity of processors, the system can support in the framework of data mining applications
- Upgrading the system needs it to be substituted entirely

**Map reduce model for parallel data mining:** Google's map reduce is a programming exemplary for processing large quantities of data in a parallel and distributed style. It suggestions resources to simplify the growth of parallel data mining approaches offering load balancing and fault tolerance. Data mining applications parallelized using map reduce exploit the Google File System (GFS) which deals a means of storage data in a distributed mode and unnecessarily over a network of commodity workstations. Google's map reduce is exclusive software. Though, Hadoop compromises an open source execution of Google's map reduce model based on its Hadoop Distributed File System (HDFS) which is Hadoop's implementation of GFS. Tsai *et al*. (2016) compared the performance differences between the distributed and map reduce methodologies over large scale datasets in terms of mining accuracy and efficiency. The experiments was based on four large scale datasets which was used for the data classification problems. The results showed that the classification performances of the map reduce based procedure are very stable no matter how many computer nodes are used, better than the baseline single machine and distributed procedures except for the class imbalance dataset. In addition, the map reduce procedure requires the least computational cost to process these big datasets.

A Google's map reduce work has three phases: map, shuffle and decrease. It is obligatory that each phase be accomplished in order before starting the next phase. A data flow plan for map reduce is given as:

**Map:** Map reduce divided an application into smaller portions named mappers. Each mapper can be treated by any of the workstations in the nodes in the cluster. The map duty is implemented based on necessities of the application. The output of map determination is key-value pairs that aid as inputs to shuffle phase. For instance, let's study a database of dogs encompassing their license id, strain and name:

- 14867 Charlie Muffy
- 88350 Toby Ddotty
- 72205 Buster Pancakes
- 95712 Toby Esther
- 87865 Buster Lassie
- 85093 Charlie Albert
- 20798 Charlie Muffy
- 12334 Buster Lassie

Map reduce tasks that find out the most general name for the breed has the output of the map function as:

**General names:**

- Charlie, Muffy
- Toby, Dotty
- Buster, Pancakes
- Toby, Esther
- Buster, Lassie
- Charlie, Albert
- Charlie, Muffy
- Buster, Lassie

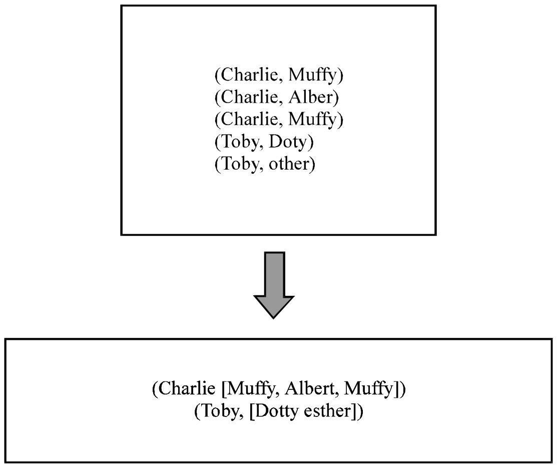

Fig. 1: Shuffle phase clusters

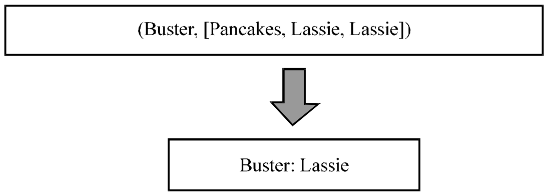

Fig. 2: Reduce phase uses reduce instruction

Great reliability of the application is delivered by the framework's competence to recover a miscarried mapper. Intermediate fallouts made by these mappers are then joint by one or more reducer nodes.

**Shuffle:** The shuffle phase clusters all the names collected for each breed and then outputs a single list comprising all the names (Fig. 1).

**Reduce:** A reduce phase uses reduce() instruction that has to be applied by the programmer. The reduce() instruction takes the shuffled intermediate fallouts to output the outcomes. In above instance, the reduce() instruction will take the input (Fig. 2).

**Sharding:** Sharding splits the input of a phase into multiple data sets (shards) that are administered in parallel. At any point in the order, all the shards in a phase must be proficient before executing the shards for next phase. The input data is divided into shards and allocated to the Mapper class. The output of the map function is then organized by shuffle phase that shards its input and allots the shards to diminish phase. The amount of shards used in each phase can be altered.

**Reduce:** A reduce phase uses reduce() instruction that has to be applied by the programmer. The reduce() instruction takes the shuffled intermediate fallouts to output the outcomes. In above instance, the reduce() instruction will take the input:

**Sharding:** Sharding splits the input of a phase into multiple data sets (shards) that are administered in parallel. At any point in the order, all the shards in a phase must be accomplished before executing the shards for next phase. The input data is divided into shards and allocated to the Mapper class. The output of the map function is then organized by shuffle phase that shards its input and allocates the shards to reduce phase. The quantity of shards used in each phase can be transformed.

Map reduce's significance in the data mining communal has been established in countless tasks. For instance, Google stated having exploited map reduce in at least 900 tasks. Though, this was in 2008 and it is very credible that there are many more tasks by now.

## MATERIALS AND METHODS

**Parallelism in decision tree induction procedures:** Ordering formed by decision tree in the method of a tree edifice. It divides a dataset into slighter subsets and at the same time advances related decision tree incrementally. A tree with decision nodes and leaf nodes is the last outcome. A decision node has two or further twigs based on the decision. Leaf node signifies a classification. The root decision node signifies the best forecaster. An instance of decision tree is presented in the following:

There are two key methods of parallelizing decision tree classifiers: the synchronous tree construction method and the partitioned tree construction method.

**Synchronous tree construction method:** In the synchronous tree construction, the teaching dataset is dispersed between N processors. So, each processor holds an exact copy of the tree in its memory during tree induction. The processors enlarge the identical tree node by gathering data of their local data and then sharing these data by communicating with further processors. Lastly each processor will do the same tree node increase independently on their copy of the tree.

**Benefit:** There is no communication of planning data.

**Weakness:**

- The communication of data rises as the tree cultivates
- The workload between the processors is varying during the tree induction and may cause workload inequities

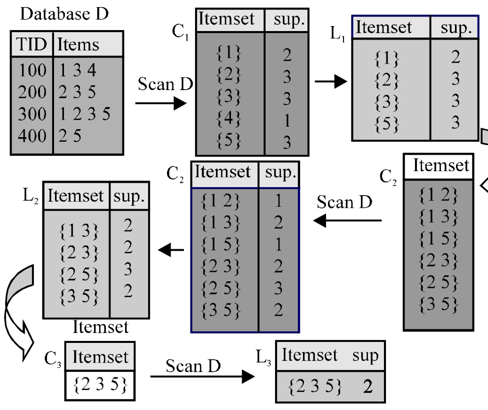

Fig. 3: An instance of Apriori-based procedure

**Partitioned tree construction method:** In partitioned tree construction, altered processors work on diverse fragments of the tree and the teaching data. In these circumstances, the root node is allotted to a single processor. The child nodes are then disseminated to varied processor. Each processor enlarges the sub-tree of its child node exclusively. This is repeatedly recursively until all processors have been assigned to diverse sub-trees (Fig. 3).

**Benefit:** Each processor works self-sufficiently no communication is needed.

**Weakness:** A single processor has the whole workload primarily.

**Hybrid method:** A hybrid scheme in this, takes the profits of both the methods. The hybrid method originally twitches with synchronous tree construction method. When the communication overhead augmentations considerably, then the process shifts to partitioned tree construction, so, working independently.

**Parallel Apriori-based procedure:** Apriori procedure is the most broad and beneficial procedure of association rule mining of data mining. Association instructions are “if-then instructions” with two measures which quantify the support and confidence of the rule for a given data set. Having their source in market basked analysis, association instructions are now one of the most general tackles in data mining.

If the slightest confidence is 50%, then the only two instructions made from this 2 itemset that have confidence higher than 50% are:

- Shoes⇒Pants Support = 50%, Confidence = 66%
- Pants⇒Shoes Support = 50%, Confidence = 100%

Table 1: Two main terms accordance

| Transaction ID’s | Items bought |
|---|---|
| 1 | Shoes, blouse, pants |
| 2 | Shoes, pants |
| 3 | Shoes, jeans |
| 4 | Blouse, swearshirt |

Table 2: Apriori procedures

| Frequent itemset | Support (%) |
|---|---|
| Shoes | 75 |
| Blouse | 50 |
| Pants | 50 |
| Shoes, pants | 50 |

**Instances of learning association rules:** Wal-Mart measured their data and found that on Friday afternoon young American males who buy diapers also tend to buy beer. So, Wal-Mart sited beer adjacent to diapers and the beer-sales went up. This is renowned because no one would have predicted such outcome and that’s the power of data mining. Amazon uses association mining to commend you the substances based on the current item you are browsing/buying. Google auto-complete searches regularly associated words that user type after that specific word.

Two main terms accordance with Apriori procedure were exist: support and confidence. The support is defined as the part of transactions in the data set which embraces the itemset. The confidence is separate as a conditional probability confidence (X = >Y). Below is an instance of working of Apriori procedure (Table 1 and 2).

**Parallelism in Apriori procedure:** The most parallel ARM procedures are based on parallelization of Apriori that iteratively makes and tests applicant itemsets from length 1 to k until no more regular itemsets are exist. Count distribution technique is one of the approaches that can be used to parallelize Apriori. The count distribution tactic follows a data-parallel plan and statically partitions the database into horizontal partitions that are self-reliantly scanned for the local counts of all candidate itemsets on each route. At the end of iteration, the local counts will be summed up across all processes into the total counts, so that, common itemsets can be institute.

The stages for the count distribution procedure are general as follows for distributed-memory multiprocessors:

- Phase 1: split the database uniformly into horizontal partitions and allot them among all routes
- Phase 2: each procedure scans its local database partition to collect the local count of each entry
- Phase 3: all procedures review the local counts and then interchange it with other process to get the whole counts of all matters and find frequent 1 itemsets

- Phase 4: set level k = 2
- Phase 5: from the mined regular itemsets, routes make candidate k-itemsets
- Phase 6: each route scans its local database divider to collect the local count of each applicant k-itemset;
- Phase 7: all routes summarize the local counts and then interchange it with other process to get the total counts of all items and find frequent k-itemsets
- Phase 8: repeat phase 5-8 with k = k+1. Stop when no extra frequent itemsets are bring into being

The count distribution procedure will scan the database 4 times to count the incidences of candidate 1-4 itemsets, respectively. Though, the workload is allocated among three processes and each of the process scans only the allocated partition to catch the local count. The total count is then established by summing up the local counts acquired from each of the local process.

## RESULTS AND DISCUSSION

**Pattern-growth technique:** Mallick *et al*. (2014) presented paper about pattern-growth that monetary and compactness constraints in addition to frequency and length are included in the sequential mining process for discovering pertinent sequential patterns from sequential databases. Also, a CFML-PrefixSpan algorithm is proposed by integrating these constraints with the original PrefixSpan algorithm which allows discovering all CFML sequential patterns from the sequential database. The proposed CFML-PrefixSpan algorithm has been validated on synthetic sequential databases. The experimental results ensure that the efficacy of the sequential pattern mining process is further enhanced in view of the fact that the purchasing cost, time duration and length are integrated with the sequential pattern ining process.

The pattern-growth performance originates frequent itemsets directly from the database with the use of original Frequent-Pattern Tree (FP-tree) structure where the repetitive transactions are compacted. The pattern-growth technique arranges the transaction itemset in frequency-ordered prefix tree such that they share common prefix part and re-occurrences of items/itemsets are automatically counted. Then the FP-tree is navigated to mine all frequent patterns. When all the frequent patterns are mined from the database, competent pattern bases are mined using a split and conquer plan. Conditional pattern base for each thing basically consists of the set of other items that habitually occur together with the item. The conditional pattern base is then converted to a conditional FP-tree which can be administered using a recursive FP-growth procedure.

The database is separated into equal partitions of transactions for accomplishing parallelism in pattern-growth way. These partitions can then be owed to various processes. Each of the process will be generating a FP-tree using its local database of transactions. After that FP-trees can used to originate conditional pattern roots and transform them into conditional FP-trees for all common objects. Then, each of the FP-trees can be treated individually, parallelism can be very well attained in this method. The FP-growth pattern can be parallelized using following below phases:

- Phase 1: wholly the processes are allocated identical partitions of transaction database
- Phase 2: wholly processes scan their local transaction database in parallel and mine all frequent objects
- Phase 3: each process constructs a local FP-tree from its local database partition with reverence to the total frequent entities. Items are arranged by frequencies in downward order within each scanned transaction
- Phase 4: each process makes local conditional pattern bases for all common items from local FP-tree
- Phase 5: assign common items to processes
- Phase 6: all its local conditional pattern bases are collected and transformed into the conditional FP-tree on the designated process for each common item
- Phase 7: each process recursively navigates each of the allocated conditional FP-trees to mine frequent itemsets in the incidence of the given item

Two phases for reaching parallelism are existing, in FP-growth way. In initial phase, the technique continues through phase 1-3 making local FP-trees. In the phase 2, 4-7 is performed in which all the common itemsets are mined by means of conditional pattern bases and conditional FP-trees. Process of mining twitches with a bottom-up traversal of the local FP-trees to make the conditional pattern bases starting from their corresponding things in the header tables. Individual conditional patterns entry base is a list of objects that go before a definite item on a route of a FP-tree up to the root with the count of each thing set to be that of the measured item node along that path.

As each local FP-tree will derive only part of the conditional pattern base for each shared thing, building the total conditional FP-tree requirements the gathering of local conditional pattern bases from all processes. Then a demand to the recursive FP-growth technique on each conditional FP-tree will produce all the conditional patterns on the elected process individually. Since, the multiple FP-trees can be made accessible to all processes if on a shared-memory system, the inexact pattern base and conditional FP-tree can be made on the fly for the elected process to mine the conditional patterns for one shared item after another. This can mostly lessen memory usage by not making all conditional pattern bases and conditional FP-trees for all common items at one time.

When all the transaction information has been built into the FP-trees in parallel FP-growth, the databases no time-consuming need scanning. Consequently, scanning the original database only twice, condensed the disk input-output. The chief communication/synchronization overhead lies in the altercation of local conditional pattern bases through all processes. Then the repetitive patterns are previously merged, the whole size of the conditional pattern bases is normally much smaller than the original database, causing in relatively low communication/ synchronization charge.

## CONCLUSION

Revolution in the capacities of data, we face with the fail of the single processor systems to bring the outcomes in reasonable expanse of time. Therefore, the commercial decision-makers and experts are relying on parallel data mining ways to extract the brief, significant information from the data in appropriate expanse of time. The ways and machines detailed above have to led to great prosperity in the arenas of research. Though, it still leftovers an active research region, regardless of excessive success in ascending data mining techniques where many unsolved demands need to be addressed and diverse inventive methods has to be explored.

## REFERENCES

Apiletti, D., E. Baralis, T. Cerquitelli, P. Garza and F. Pulvirenti *et al*., 2017. A parallel mapreduce algorithm to efficiently support itemset mining on high dimensional data. Big Data Res., 10: 53-69.

Mallick, B., D. Garg and P.S. Grover, 2014. Constraint-based sequential pattern mining: A pattern growth algorithm incorporating compactness, length and monetary. Int. Arab J. Inf. Technol., 11: 33-42.

Menandas, J.J. and J.J. Joshi, 2014. Data mining with parallel processing technique for complexity reduction and characterization of big data. Glob. J. Adv. Res., 1: 69-80.

Simoes, P.W., R. Venson, E. Communello, R.A. Casagrande and E. Bigaton *et al*., 2015. Distributed Parallel Computing in Data Analysis of Osteoporosis. In: MEDINFO 2015: EHealth-enabled Health, Sarkar I.N., ?A. Georgiou, ?P. Mazzoncini and A. Marques (Eds.). IMIA and IOS Press, Amsterdam, Netherlands, ISBN: 978-1-61499-563-0, pp: 1082-1083.

Thabtah, F., S. Hammoud and H. Abdel-Jaber, 2015. Parallel associative classification data mining frameworks based mapreduce. Parallel Process. Lett., Vol. 25,

Tsai, C.F., W.C. Lin and S.W. Ke, 2016. Big data mining with parallel computing: A comparison of distributed and MapReduce methodologies. J. Syst. Software, 122: 83-92.

Xiao, H., 2010. Towards parallel and distributed computing in large-scale data mining: A survey. MSc Thesis, Technical University of Munich, Munich, Germany.

Yan, X., J. Zhang, Y. Xun and X. Qin, 2017. A parallel algorithm for mining constrained frequent patterns using mapreduce. Soft Comput., 21: 2237-2249.