Parallel and distributed Data Mining Techniques: GPU Approach

Tagreed Alsulimani

Management Information System Department
College of Business, University of Jeddah
Jeddah, Kingdom of Saudi Arabia (KSA)

Abstract
Recent advancements in information technology have led to significant transformations in new processors such as Graphical Processing Units (GPUs) and multicore processors. These innovations are characterized by high-performance capabilities, marked by extensive parallelism and the integration of nonvolatile memory with hybrid storage hierarchies. Today, the task of uncovering relationships among various sets of items within vast and diverse datasets stands as a crucial challenge in information systems. Consequently, leveraging parallel architectures, including emerging processors, has emerged as a promising avenue for enhancing the performance of information systems. In this paper, we present a comprehensive survey of diverse techniques and solutions employed in the parallel and distributed data mining solutions. We delve into the rationale behind and the hurdles associated with the utilization of parallel processors such as multicore CPUs and GPUs, as well as distributed technologies.

Keywords

Information system; distributed; parallel ; GPU; data mining;

I. INTRODUCTION

Parallel data mining, and more generally data mining and knowledge discovery in large databases (KDD), is an active field which has received considerable attention in the research and development communities. Data mining has been recognized to be an important and challenging research area, governing the processes essential to scientific discoveries, predictive modeling, and decision-making for business applications, particularly when it comes to large data sets. These data sets may be in a variety of forms and tuples, and inherently distributed in nature, stretching from hundreds of gigabytes of data to many terabytes or more, and they are being collected primarily from computational and communications systems like the Web, large-scale scientific experiments, automated medical records, and electronic commerce. Such data sets present distinct technical challenges arising from their volume and distributed nature, but are of ever-increasing business, scientific and social importance [34].

Manuscript received December 5, 2024
Manuscript revised December 20, 2024
https://doi.org/10.22937/IJCSNS.2024.24.12.25

The main part of the distributed association rule mining algorithms is based on Apriori algorithm but these algorithms suffer from the drawbacks of the Apriori algorithm. A detailed study shows that, even with the best implementation, these algorithms still under-utilize a multicore system due to poor data locality and insufficient parallelism expression.

Finding frequent itemset is one of the most investigated fields in data mining. The Apriori algorithm is the most established algorithm for frequent itemset mining [1]. One of the most important problems in data mining is association rule mining. It requires very large computation and I/O traffic capacity. Let $I = \{i1, i2, ..., im\}$ be a set of items, and D be a transaction database, where each transaction T is a set of items with $T \subseteq I$ and it has a unique identifier (TID). An itemset with k items is called a k−itemset. The support of an itemset A, denoted as sup(A), is the number of transactions in a database D which contain A. An association rule is an expression $A \Rightarrow B$, where itemset $A, B \subset I$, and $A \cap B = \emptyset$. the confidence of the association rule, which is the fraction $sup(A \cup B)/sup(A)$, and the support of the association rule, which is equal to $sup(A \cup B)$. Several sequential and parallel algorithms for discovering frequent itemset have been proposed in the literature. The most popular algorithm among them is the Apriori algorithm of Agrawal and Srikant [3]. Eclat [15] and Partition [10] consists in representing the dataset vertically by giving to each item its tidset, i.e. the set of transactions containing this item.

In this paper, we analyze the state-of-the-art data mining and knowledge discovery techniques that utilize parallel computing resources to solve data-intensive and computationally complex problems. Popular parallel computing paradigms like message passing, data parallelism, shared memory multiprocessing, and hybrid models are surveyed in the context of data mining techniques like parallel association rule mining, parallel classification, fast parallel clustering techniques and their hybrid models. We also analyze the limitations of parallelism in the context of data mining and discuss future and important research directions. We provide guidelines and suggestions on how to implement the applicable data mining techniques.

II. Data Mining : General notions

With massive amounts of data continuously being collected and stored, many industries are becoming interested in the extraction of useful information from large databases. Association mining is one of the most important data mining tasks. Association rule mining (ARM) finds interesting correlation relationships among a large set of data items. A typical example of association rules mining is market basket analysis. This process analyses customer buying habits by finding associations between the different items that customers place in their ”shopping baskets”. Such information may be used to plan marketing or advertising strategies, as well as catalogue design. Each basket represents a different transaction in the transactional database, as-sociated to this transaction the items bought by a specific customer. An example of an association rule over such a database could be that ”80% of the customers that bought bread and milk, also bought eggs”.

The association rule mining is a two-step process:

1. the first step consists of finding all frequent itemset that occur at least as frequently as the fixed minimum support. The search space for enumerating all frequent itemset is extremely large. For example, with m attributes there are, in the worst case, O(mk) potential sequences of length at most *k*;
2. the second step consists of generating strong implication rules (which satisfy the minimum confidence constraints) from these frequent item- sets.

The second step is the easiest of the two. The overall performance of mining association rules is determined by the first step. Because of that, we concentrate our attention in this paper on the frequent sets counting problem (FSC), which is the most time-consuming phase of the association mining process.

Nevertheless, given the search complexity, sequential algorithms cannot provide scalability, in terms of the data size and the performance for large databases, we must rely on parallel multiprocessor systems to fill this role.

Several parallel algorithms have been developed for association mining for both distributed memory, shared memory and recently hierarchical systems. In this paper, we examine the issue of mining association rules among items in large databases transactions using the algorithm Apriori [2].

III. Parallel and distributed data mining : A review

Data mining, which in recent years has emerged as one of the central problems in large databases, has the following desirable goals: finding patterns, associations, and relationships in databases, often interpreted as "knowledge". The volume of data in real-world applications is often so high that it cannot be handled with traditional data management technology. Furthermore, data processing needs are very diverse, ranging from simple queries to report generation and decision support. For example, answering complex queries or complex analyses are essential components of many database applications used at that time. With the advances in modern database technology, such applications are common. However, sacrificing system performance in exchange for ease of use can have significant implications for the timeliness of the results. Given the above motivation, it is easy to appreciate the significance of parallel data mining. In this section, we briefly touch upon the parallel computing models and identify the issues in parallelizing data mining algorithms. We will assume familiarity with the general classes of data mining algorithms; the interested reader is referred to the appropriate survey.

Parallel computing has been attracting significant research interest for various application problems, and for good reasons. It is a natural way of solving spatially distributed, inherently parallelizable problems. Algorithms in areas like fluid dynamics and nuclear physics have successfully exploited the underlying parallelism in these applications. As the computer hardware field moves towards a system architecture made of multiple processors (called multiple instruction multiple data or MIMD), it is becoming increasingly necessary to appreciate the issues involved in developing parallel algorithms for application areas not traditionally considered "scientific computing". In fact, high-performance workstations and large-scale parallel machines are now being used to address many nontraditional applications. These areas are as diverse as medicine, finance, business and commerce, computer-aided design, image processing, and computer graphics [35].

Most of the existing algorithms, use local heuristics to handle the computational complexity. The computational complexity of these algorithms are fast enough for application domains where N is relatively small. However, in the data mining domain where millions of records and a large number of attributes are involved, the execution time of these algorithms can become prohibitive, particularly in interactive applications. Parallel algorithms have been suggested by many groups developing data mining algorithms.

In general, there are two main categories of approaches that offer scalability to data mining algorithms. The first category comprises techniques that find algorithms or problem instance representations which allow parallel execution of independent units of computation [32]. These techniques tend to be more generally applicable to a wide range of computational problems, as they offer only limited forms of non-independent communication or synchronization. The second category of techniques, which prove to be more powerful in terms of yielding greater scalability and efficiency, decompose the instance representation of the data mining problem itself. Decomposition-based approaches allow non-independent communication and synchronization, and hence, when applied to data mining problems, tend to have the potential to surpass the performance that can be achieved with techniques based on executions with independent units of computation [33].

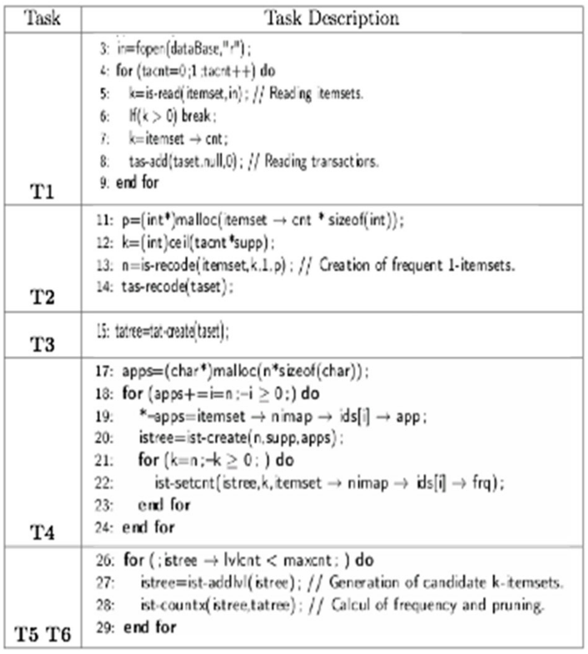
Many parallel algorithms for solving the FSC problem have been proposed to provide scalability for association rule mining algorithms. Most of them use Apriori algorithm as fundamental algorithm, because of its success on the sequential setting [1]. The reader could refer to the survey of Zaki on ARM algorithms and relative parallelization schemas [11][10]. Agrawal et al. proposed a broad taxonomy of the parallelization strategies that can be adopted for Apriori [1]. Two different parallelization of Apriori on distributed memory machine where presented in Agrawal et Al 1996.

Several strategies for parallel frequent itemset computation were proposed by Agrawal and Shafer [2], including the Count Distribution algorithm. This algorithm is a parallelization of Apriori. Each processor generates the partial support of all candidate itemset from its local database partition. At the end of each iteration, the global supports are generated by exchanging the partial supports among all the processors. Zaki et al. proposed the Common Candidate Partitioned Database (CCPD) and the Partition Candidate Common Database (PCCD) algorithms, which both are Apriori-like algorithms [14]. Souliou et al. proposed the PPS algorithm based on CD algorithm. The difference consists on that PPS makes use of partial support trees [13]. Another algorithm called Data-VP was presented by Coenen et al. [8]. PC clusters have become popular in parallel processing. They do not involve specialized interprocessor networks, so the latency of data communications is rather long.

Other advantage of the distributed approach is that it can make significant data processing during the data distribution phase.

- **Count distribution CD** In CD, the database is partitioned and distributed across n processors.
- **Data distribution DD** In DD The candidates are partitioned over all the processor in a round-robin fashion. The communication overhead of broadcasting the database partitions can be reduced by asynchronous communication. Experiments shows that algorithms based on count distribution outperforms the other algorithms. Data distribution is the worst approach, while candidate distribution obtained good performances but paid a high overhead due to the need of redistributing the dataset. In shared memory many algorithms are proposed by Zaki et Al [12].
- **Common Candidate Partition Data** CCPD algorithm, it is essentially the same as count distribution, but uses some optimization techniques to balance the candidate hash tree and to short-circuit the candidate search for fast support counting.
- **Eclat algorithm** was designed to overcome the shortcomings of the CD and CND algorithms. It utilizes the aggregate memory of the system by partitioning the candidates into disjoint sets using the equivalence class partitioning. Eclat uses the vertical database layout which clusters all relevant information in an itemset's tid-list. Each processor computes all the frequent itemset from one equivalence class before proceeding to the next.

The most parallel approaches work as following. The different steps to generate frequent itemset are presented on the following figure (see Fig 1) [37].



Task	Task Description
T1	`3: in=fopen(dataBase,"r");` `4: for (tacnt=0;1;tacnt++) do` `5:   k=is-read(itemset,in); // Reading itemsets.` `6:   If(k > 0) break;` `7:   k=itemset → cnt;` `8:   tas-add(taset,null,0); // Reading transactions.` `9: end for`
T2	`11: p=(int*)malloc(itemset → cnt * sizeof(int));` `12: k=(int)ceil(tacnt*supp);` `13: n=is-recode(itemset,k,1,p); // Creation of frequent 1-itemsets.` `14: tas-recode(taset);`
T3	`15: tatree=tat-create(taset);`
T4	`17: apps=(char*)malloc(n*sizeof(char));` `18: for (apps+=i=n;--i ≥ 0;) do` `19:   *--apps=itemset → nimap → ids[i] → app;` `20:   istree=ist-create(n,supp,apps);` `21:   for (k=n;--k ≥ 0; ) do` `22:     ist-setcnt(istree,k,itemset → nimap → ids[i] → frq);` `23:   end for` `24: end for`
T5 T6	`26: for ( ; istree → lvlcnt < maxcnt; ) do` `27:   istree=ist-addlvl(istree); // Generation of candidate k-itemsets.` `28:   ist-countx(istree,tatree); // Calcul of frequency and pruning.` `29: end for`

Fig 1. The different tasks to extract frequent Itemset

The figure (see Fig 2) presents a different step to execute candidate generation in parallel. The database transactions is partitioned during the initial iteration among processors participating in the execution, and each processor calculates partial supports of its items from its local database partition. Each processor calculates partial supports of its 1-itemset candidates from its local database partition. At the end, the task does a sum reduction to obtain the global counts by exchanging local counts with all other processors. Once the global Fk has been determined, each processor builds the entire candidate Ck+1 in parallel, and the previous process is repeated until all frequent itemset are found. The goal of this approach is minimizing communication, because only the counts are exchanged between processors. To minimize the communications phase, each processor Pi can maintain an n support array where n is the number of items and each case contains the corresponding support. This structure doesn't suffer from any of the overheads because the support array is sorted in increasing order, then the global count support of a kth candidate by simply adding the corresponding columns in each processor. This is a vectors of bits accessed with high locality, and without using expensive comparisons and conditional branch instructions. A column summation can be done to calculate the global support.

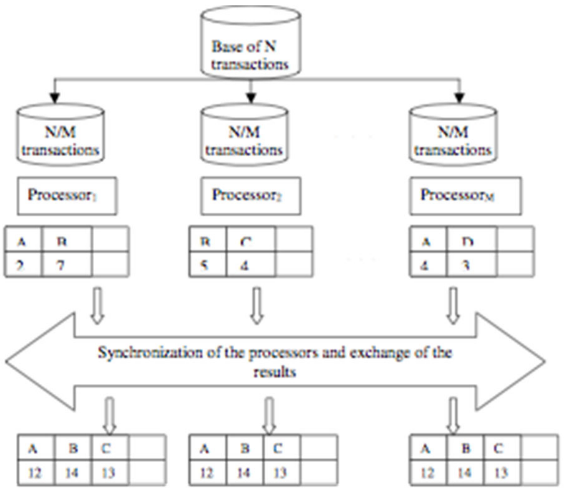


Fig. 2 A parallel candidate generation

The figure below (see Fig 3) presents the parallel steps to extract frequent itemset. This step consists to partition the 1-itemset candidates, so that each processor can prune the non-frequent 1-itemset. After that, each processor can locally sort his support array according to the frequency. A step of global sort is necessary. To do that, we adopted a master-slave approach, where we have one master and the rest of available processors are slaves. At the end of this step, the master processor sends the global support array to the others processors.

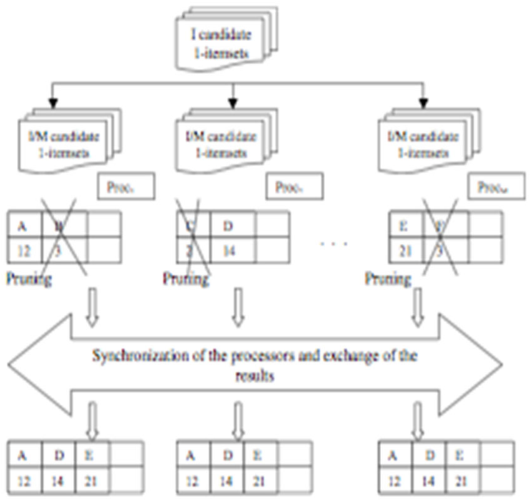


Fig. 3 A parallel frequent itemset generation

The Fig 4. presents the pruning step to eliminate non frequent itemset. In this task, we reduce the size of transactions database by pruning out the non-frequent 1-itemset. Hence, to do this step rapidly, we partition this database equitably between all processors and each one scans its local portion to skipped the non-frequent.

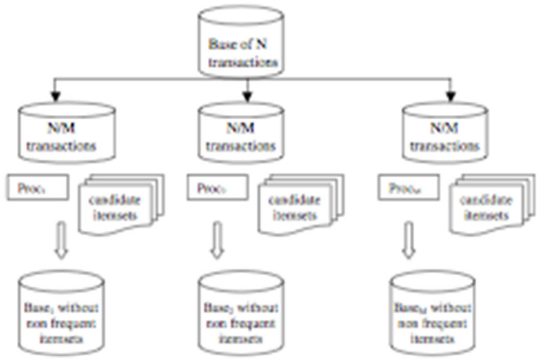


Fig. 4 A parallel non frequent itemset pruning

To achieve an equal distribution of the candidate itemset, we use a partitioning algorithm that is based on bin-packing [6] such that the sum of numbers of candidate itemset are roughly equal. This gives about the same size hash tree in all the processors and thus provides a correct load balancing among processors.

For example, suppose that our tree is composed by four itemset {A, B, C, D} and we have two processors {P1,P2}, the result of this distribution is :

- P1 is charged to do generate 2-itemset by simply joining all pairs of items A et D with her successors, hence, we obtain {{AB },{AC },{AD }}.
- P2 generate all combinations based on the items B et C and we obtain {{BC},{BD},{CD}}.

Note that the equal assignment of candidates to the processors does not guarantee the perfect load balance among processors. This is because the cost of traversal of the hash tree are determined not only by the size of the tree, but also by the presence of items in the transactions.

For more improving the cache performance multi-core CPUs and GPUs processors are used. Ghoting and al. [17] proposed a cache conscious FP-tree (CC-tree), a reorganization of the original FP-tree by allocating the nodes in sequential memory space and a tiling strategy for temporal locality. This structure yields a better cache performance than FP-growth, but it still experiences cache misses when traversing the CC-tree due to his tree structure. Razs presented nonordfp [18] which implements FP-growth without rebuilding the projected FP-tree recursively to improve the cache performance. Li et al. proposed an FP-growth implementation [14] based on two techniques: a cache conscious FP-array and a lock free parallelism enhancement to improve data locality performance and makes use of the benefits from hardware and software pre-fetching.

For a performance improvement of this algorithm, most researchers [20, 28, 29, 14, 30, 31] have been made for parallelizing FP-growth. Pramudiono and Kitsuregawa [20] reported results for parallel FP-growth algorithm on shared nothing cluster environment. In [20], Li et al. proposed a Parallel FP-growth that shard a large-scale mining task into independent parallel tasks. Osmar et al. presented a MLFPT (Multiple Local FP-tree) approach [22] that consists of two main stages: the first stage is the construction of a parallel FP-tree for each processor and the second stage for mining these data structures much like the FP-growth algorithm.

In [20], Manaskasemsak et al. presented a parallel version of FI-growth algorithm [25] that parallelizes the association rule mining process by employing a data parallelism technique on a PC cluster.

Researchers have also studied FIM algorithms on new-generation graphics processing units (GPUs), regarded as massively multi-threaded many-core processors. Different from multi-core CPUs, the cores on the GPU are virtualized, and GPU threads are executed in SIMD (Single Instruction, Multitple Data) and are managed by the hardware. Such a design simplifies GPU programming and improves program scalability and portability. Nevertheless, it makes the implementation of algorithms with complex control flows a challenging task on the GPU, even though the GPU has an order of magnitude higher computation capability as well as memory bandwidth than a multi-core CPU. Taking advantage of the massive computation power and the high memory bandwidth of the GPU, there have been some studies that focus on studying the GPU acceleration for FIM algorithms.

GPGPU for FIM algorithms was for the first time addressed in [26], where Luo et al. presented two GPU-based implementations of the well-known Apriori algorithm, that take advantages of the GPU's massively multi-threaded SIMD (Single instruction, multiple data) architecture. Both implementations employ a bitmap data structure to exploit the GPU's SIMD parallelism. One implementation runs entirely on the GPU, since the other employs both the CPU and the GPU for processing. Another Apriori based FIM algorithm for GPU is presented in [27], GPApriori, which includes a set of fine-grained parallel data structures and algorithms design to achieve promising degree of speed up on modern GPU.

Different approaches are proposed in [28] by Teodoro and al. based on the Tree Projection algorithm described in [27]. Nonetheless, Tree Projection is not a state-of-the-art algorithm for FIM, as it is outperformed by FP-growth [15,31]. In [30], Orlando and Silvestri proposed gpuDCI, a parallel algorithm inspired by DCI [16] that exploits GPUs to efficiently mine frequent itemset.

Fig. 5 shows an example of multi-threading execution under 4 cores.

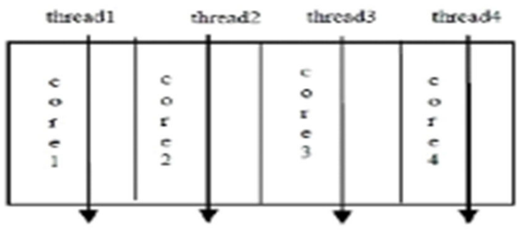


Fig. 5 A multi-threading core execution

IV. SYNTHESIS

The performance of the parallel data mining algorithms is based on two principles factors such the data communication cost and performance of time. The data communication cost can be reduced by using a good partitioning solution like intelligent partitioning Algorithm (K-means for example) to avoid a large overhead. The processing time is based on the size of database, the number of scanned database and the size of used memory.

Many parallel data mining systems use partitioning methods to distribute the data among the processors in a parallel computer. Some methods work better with cube-type operations, while some methods are more appropriate for workload-intensive applications. The choice of method depends on the data characteristic, the application characteristics, and the architecture of the parallel system. The overall performance of a parallel system depends on scheduling methods within the parallel algorithm that aim at balancing workloads and minimizing communication between processes. In addition, the performance of a system can also be influenced by the data representation and data transfer costs between the disks, memory, and processors. The remaining of the chapter discusses important research in parallel data mining architectures, partitioning methods, and data placement methods, and discussions are concluded with a summary of research problems, future research directions, and general conclusions to the work in this direction.

Parallel data mining methods focus on partitioning the data and employing multiple processors to finish the partitioning process. This allows parallel data mining methods to perform some level of in-memory computations with ease by working on data subsets in parallel across different processors. The number of records in each processor needs to be carefully chosen to minimize the cost of distributing data to every processor and to minimize the cost of reducing the problem results. The compression and distribution of data often balance the task allocation and data input process in parallel data mining systems. Scalability, speedup, and efficiency are metrics in parallel computing that need to be investigated carefully. They provide information on how a parallel system performs as its size and data volume are increased, and they serve as indicators of the partitioning methods that depend on system and data characteristics [36].

V. CONCLUSION

This paper provides a comprehensive survey of parallel data mining approaches, categorizing them based on the machine architecture, data splitting methods, processing roles, and algorithms employed. By presenting various strategies and methodologies, this survey aims to shed light on the state of the art in parallel data mining and provide valuable insights for researchers and practitioners in the field.

The One way to anticipate future needs is to look at emerging technologies and trends in technology usage in general. Several researchers have studied technology and tool usage, when searching for patterns in technology adoption. Some of the new exciting places in data mining include: Microsoft's new eScience Studio toolset, useful for researchers in many areas, Hudmonja (at the University of Modena and Reggio Emilia), which is useful for easy parallel data mining web-site clustering, Gandalf, running on the Gage heterogeneous/monocore/MIMD machine, which offers superior performance for parallel decision tree learning, the excellent cloud tools by the Apache Hadoop project, SKMT-Master by the University of Karlsruhe, which provides a great tool for testing clustering algorithms on countless sets of clustering partitions, the R and SAGA interface to create workflows, the grid middleware in general and especially the graphics specification in OpenMP.

References

[1] R. Agrawal and J. Shafer, Parallel mining of association rules, IEEE Transaction On Knowledge and Data Engineering 8 (1996) 962–969.

[2] R. Agrawal and R. Skirant, Fast algorithms for mining association rules in large databases, Proceedings of the 20th Int́l Conference of Very Large Databases (VLDB'94), (June 1994), pp. 478–499.

[3] A. J. Bernstein, Program analysis for parallel processing, Proceedings of IEEE Trans. on Electronic Computers, (October 1966), pp. 757–762.

[4] C. Borgelt, Efficient Implementations of Apriori and Eclat. Workshop of Frequent ItemSet Mining Implementations., 2003).

[5] T. Fosdick and al., An Introduction to High Performance Scientific Computing (MIT Press, 1996).

[6] E. Han, G. Karypis and V. Kumar, Scalable parallel data mining for association rules, Proceedings ACM Conference of Management of Data, (ACM Press, New York, 1997), pp. 277–288.

[7] T. M. S. http://www.mhpcc.edu/training/workshop/mpi/main.html [8] G. of DataBases: Site http://www.almaden.ibm.com/cs/quest [9] G. of dense DataBases: Site http://fimi.cs.helsinki.fi/data.

[8] Y. Slimani, K. Arour and M. Jemni, Informatique répartie : Chapter Découverte parallèle de régles associatives (Hermes, Lavoisier, March 2005).

[9] M. Zaki., Parallel and distributed association mining: a survey., IEEE Concurrency 7 (4) (1999) 14–25.

[10] M. Zaki, M. Ogihara, S. Parthasarathy and W. Li, Parallel data mining for association rules, IEEE Transactions Knowledge and Data Engineering (August 1996) 962–969.

[11] M. Zaki., Parallel and distributed association mining: a survey., IEEE Concurrency 7 (4) (1999) 14–25.

[12] M. Zaki, M. Ogihara, S. Parthasarathy and W. Li, Parallel data mining for association rules, IEEE Transactions Knowledge and Data Engineering (August 1996) 962–969.

[13] Coenen F., Leng P., and Ahmed S. T-trees, vertical partitioning and distributed association rule mining. Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM'03), pages 513–516, 2003.

[14] E. Li and L. Liu, "Optimization of frequent itemset mining on multiple-core processor," in Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria, September 23-27, 2007, 2007, pp. 1275–1285.

[15] P. J. Han Jiawei and Y. Yiwen, "Mining frequent patterns without candidate generation," in Proceedings of the 2000 ACM SIGMOD international conference on Management of data, New York, NY, USA, 2000, pp. 1–12.

[16] S. Brin, R. Motwani, and C. Silverstein, "Beyond market baskets: Generalizing association rules to correlations," SIGMOD Rec., vol. 26, no. 2, pp. 265–276, Jun. 1997. [Online]. Available: http://doi.acm.org/10.1145/253262. 253327

[17] A. Ghoting, G. Buehrer, S. Parthasarathy, D. Kim, A. D. Nguyen, Y.-K. Chen, and P. Dubey, "Cache-conscious frequent pattern mining on a modern processor." in VLDB, K. Böhm, C. S. Jensen, L. M. Haas, M. L. Kersten, P.-A. Larson, and B. C. Ooi, Eds. ACM, 2005, pp. 577–588.

[18] B. Rácz, "nonordfp: An fp-growth variation without rebuilding the fp-tree." in FIMI, ser. CEUR Workshop Proceedings, R. J. B. Jr., B. Goethals, and M. J. Zaki, Eds., vol. 126. CEUR-WS.org, 2004.

[19] I. Pramudiono and M. Kitsuregawa, "Parallel fp-growth on pc cluster," in Proceedings of the 7th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, ser. PAKDD'03. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 467–473. [Online]. Available: http://dl.acm.org/citation.cfm?id=1760894.1760956

[20] null Bundit Manaskasemsak, null Nunnapus Benja-

[21] mas, A. Rungsawang, null Athasit Surarerks, and null Putchong Uthayopas, "Parallel association rule mining based on fi-growth algorithm," vol. 1. Los Alamitos, CA, USA: IEEE Computer Society, 2007, pp. 1–8.

[22] H. Li, Y. Wang, D. Zhang, M. Zhang, and E. Y. Chang, "Parallel fp-growth for query recommendation," in Proceedings of the 2008 ACM Conference on Recommender Systems, ser. RecSys '08. New York, NY, USA: ACM, 2008, pp. 107–114.

[23] M. E.-H. P. L. Osmar R. Zaiane, "Fast parallel association rules mining without candidate generation," in Proceedings of the 2001 IEEE International Conference on Data Mining, 2001.

[24] A. K. Asif Javed, "Frequent pattern mining on message passing multiprocessor systems," in Distributed and Parallel Databases, vol. 16, no. 3, 2004, pp. 321–334.

[25] K. Amphawan and A. Surarerks, "An approach of frequent item tree for association generation," Proceeding of Artificial Intelligence and Soft Computing, 2005.

[26] W. Fang, M. Lu, X. Xiao, B. He, and Q. Luo, "Frequent itemset mining on graphics processors," in Proceedings of the Fifth International Workshop on Data Management on New Hardware, DaMoN 2009, Providence, Rhode Island, USA, June 28, 2009, 2009, pp. 34–42.

[27] F. Zhang, Y. Zhang, and J. D. Bakos, "Gpapriori: Gpu- accelerated frequent itemset mining," in Proceedings of the 2011 IEEE International Conference on Cluster Computing (CLUSTER), Austin, TX, USA, September 26- 30, 2011.

[28] G. Teodoro, N. Mariano, W. M. Jr., and R. Ferreira, "Tree projection-based frequent itemset mining on multicore cpus and gpus," in 22st International Symposium on Computer Architecture and High Performance Computing, SBAC-PAD 2010, Petropolis, Brazil, October 27-30, 2010.

[29] R. C. Agarwal, C. C. Aggarwal, and V. V. V. Prasad, "A tree projection algorithm for generation of frequent item sets," J. Parallel Distrib. Comput., vol. 61, no. 3, pp. 350–371, 2001.

[30] C. Silvestri and S. Orlando, "gpudci: Exploiting gpus in frequent itemset mining," in Proceedings of the 20th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, PDP 2012, Munich, Germany, February 15-17, 2012.

[31] Khedija Arour, Amani Belkahla: Frequent Pattern-growth Algorithm on Multi-core CPU and GPU Processors. CIT 22(3): 159-169 (2014)

[32] Agapito, G., Guzzi, P. H., & Cannataro, M. (2021). Parallel and distributed association rule mining in life science: A novel parallel algorithm to mine genomics data. Information Sciences.Plotnikova, V., Dumas, M., & Milani, F. (2020). Adaptations of data mining methodologies: a systematic literature review. Plotnikova V, Dumas M, Milani F. Adaptations of data mining methodologies: a systematic literature review. PeerJ Comput Sci. 2020 May 25;6.

[33] Bisong, E., Tran, E., & Baysal, O.. Built to Last or Built Too Fast? Evaluating Prediction Models for Build Times. IEEE/ACM 14th International Conference on Mining Software Repositories, Bisong, Ekaba and Tran, Eric and Baysal, Olga, 2017.

[34] Kholod, I., Shorov, A., & Gorlatch, S.. Efficient Distribution and Processing of Data for Parallelizing Data Mining in Mobile Clouds. J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl., 2020, 11(1), 2-17.

[35] Agapito, G., Guzzi, P. H., & Cannataro, M.. Parallel and distributed association rule mining in life science: A novel parallel algorithm to mine genomics data. Information Sciences 2021.

[36] Kumar, S., & Mohbey, K. K.. A review on big data based parallel and distributed approaches of pattern mining. Journal of King Saud University-Computer and Information Sciences, 2022. 34(5), 1639-1662.

[37] Khedija AROUR and Amani BELKAHLA. Frequent Pattern-growth Algorithm on Multi-core CPU and GPU Processors. J. Comput. Inf. Technol. 22(3): 159-169 2014.

Tagreed Alsulimani is an Associate Professor And the Supervisor of the Department of Information System Management at Business College of University of Jeddah