International Journal of Scientific Research and Management (IJSRM)
||Volume||5||Issue||06||Pages||5588-5590||2017||
Website: www.ijsrm.in ISSN (e): 2321-3418
Index Copernicus value (2015): 57.47 DOI: 10.18535/ijsrm/v5i6.34

Parallel State of the Art Algorithms for Frequent Itemset Mining – A Concise Descriptive Summary of Scalable Approaches

Nafisur Rahman[1]***, Samar Wazir***[2]
[1] Department of Computer Science and Engineering,
Faculty of Engineering Sciences and Technology,
Jamia Hamdard, New Delhi
nafiis@gmail.com
[2] Department of Computer Science and Engineering,
Faculty of Engineering Sciences and Technology,
Jamia Hamdard, New Delhi
samar.wazir786@gmail.com

Abstract: ***Frequent Item set Mining is a Data Mining task that has attracted the researchers' interests in a way that very few other tasks have done. This concept is generally used in Decision Support problems. Many serial and parallel algorithms have been developed for Frequent Item set Mining. In this paper, we have focused on the developments of parallel algorithms in this area so far. We start with an Apriori-based parallel algorithm that focuses on minimizing the communication overhead even if, in parallel, it requires redundant duplicate computations. Then we discuss an algorithm that utilizes the system memory in amore effective manner. Then we give an account of the algorithm that reduces the synchronization between the processors, segments the database, and integrates load balancing. Then we describe an algorithm that partitions the Candidate Item sets intelligently. Finally, we give an account of an algorithm that combines two famous algorithms to leverage their cumulative advantage. While keeping the original ideas intact, we have avoided the cumbersome notations to keep it easily intelligible.***

Keywords: Data Mining, Frequent Itemsets, Market Basket Analysis, Serial Algorithms, Parallel Algorithms.

1. Introduction

Data Mining [10], [12] is aimed at discovering valuable information, which is otherwise non-obvious, from an enormously large collection of data. According to the terminologies used in the field of *Data Mining*, *Frequent Itemsets* [1] refer to interesting patterns discovered from databases. An *Itemset* is a collection of one or more items. An *Itemset* with a support equal to or more than a minimum support threshold, it is called a *Frequent Itemset*. *Frequent Itemset Mining* has been very useful in Market Basket Analysis. It is extensively applied in various decision support problems intended to help different types of businesses. Several approaches have been employed, over the years, for *Frequent Itemset Mining* [11]. Some of the algorithms follow serial approach while others follow parallel approach [2], [8]. Many *Apriori*-based [3] parallel algorithms have been proposed. The *Count Distribution Algorithm* [4] allows redundant computations in parallel and idle processors to cut down the communication requirements. The *Data Distribution Algorithm* [4] exploits the aggregate system memory in a better way and ensures that the processors count mutually exclusive candidates thereby increasing the number of candidates counted with the increase in the number of processors. The *Candidate Distribution Algorithm* [4] partitions both the datasets and the *Candidate Itemsets* in such a way that ensures processors' independence. The Intelligent *Data Distribution Algorithm* [5], [6], [7] partitions the *Candidate Itemsets* intelligently so that hash tree can be efficiently built in a parallel fashion. The *Hybrid Distribution Algorithm* [5], [6], [7] combines the goodness of both *Count Distribution Algorithm* and the *Data Distribution Algorithm*.

2. Count Distribution

This algorithm aims to avoid communication by following the principle of redundant parallel computations. The algorithm makes a number of passes to find all the *Frequent Itemsets*. In the first pass, local *Candidate Itemsets* are generated by all the processors dynamically depending on the items present in the local datasets. In all subsequent passes, the processors generate the complete candidate *Itemset* based on the complete frequent *Itemset* discovered in the previous pass. Then the processors make a pass over their local partitioned data sets to develop the local support counts for the candidates in their respective *Candidate Itemsets*. The processors then exchange local *Candidate Itemsets*' counts with each other to develop global *Candidate Itemsets*' counts. The processors are then synchronized before they find the *Frequent Itemsets* from the *Candidate Itemsets*. The processors independently decide whether to terminate or to move on to the next pass.

3. Data Distribution

While the independent and asynchronous operation of the processors is ensured by the *Count Distribution Algorithm*, it does not fare very well on memory utilization front. The *Data Distribution Algorithm* is aimed at better memory utilization of the total system memory with the increase in the number of processors. This algorithm relies on each processor counting mutually exclusive candidates.

Like Count Distribution, *Data Distribution Algorithm* too makes a number of passes. The process of local candidate *Itemset* generation in the first pass is same as that in the *Count Distribution Algorithm*. In all subsequent passes, the processors generate the complete candidate *Itemset* based on the complete frequent *Itemset* discovered in the previous pass and retain only one nth of the *Candidate Itemsets* creating a subset, that will be counted, of the *Candidate Itemsets*. The *Itemsets* to be retained is determined by the processor id and it does not require communicating with other processors. Then the processors develop the local support counts for the candidates in their respective *Candidate Itemsets* using both the local datasets and the datasets received from other processors. The processors then compute the *Frequent Itemsets* using the local subsets of the *Frequent Itemsets*. The processors then exchange these local *Frequent Itemsets* to find the complete global frequent *Itemset*.

4. Candidate Distribution

Both the algorithms discussed above require consulting all the processors and gathering all the information before proceeding to the next pass. The *Candidate Distribution Algorithm* aims to overcome these dependencies. This is achieved by both the datasets and the *Candidate Itemsets* in a way that ensures independent functioning of all the processors. This algorithm divides the *Frequent Itemsets*, in some heuristically determined pass, between the processors in such a way that each processor can generate a unique candidate *Itemset* that does not depend on other processors. Repartitioning of data ensures that the processors can independently count the candidates in the *Candidate Itemsets*. Based on the quality of partitioning, part of the database may require replication on several processors. The processors proceed independently after candidate distribution. Counts or tuples don't need to be communicated. However, the processors do depend on each other for local candidate *Itemset* pruning. But this is done asynchronously where the processors are not required to wait for the complete pruning information to arrive from all other processors. Pruning is performed using the available information that has arrived. The pruning information that arrives late can be used in the subsequent passes.

5. Intelligent Data Distribution

The previous algorithms suffered from some serious problems like speed issues on realistic parallel computers connected with a sparse network, causing processors to stay idle on architectures without support for asynchronous communication, and no decrease in communication required with a decrease in computation. The Intelligent *Data Distribution Algorithm* addresses these issues by making the processors send locally stored portions of the database to all other processors by a ring based all to all broadcast [9]. It overcomes the contention problem of the Data Distribution and other algorithms. In this algorithm, a logical ring is formed by the processors who determine the neighbors on their left and right. Each processor has a send buffer and a receive buffer. In the beginning, send buffer is filled with a block of local data. Asynchronous send operation is initiated by the processors to the neighbors on their right with send buffer and asynchronous receive operation to the neighbors on left with receive buffer. The transactions in the send buffer are processed by the processors and the assigned count of the candidates is collected while the asynchronous operations are taking place. After this, the processors wait for the asynchronous operations to complete. Then the send buffer and the receive buffer switch their roles and the operation continues. In contrast to the Data Distribution and other algorithms, where all processors communicate with each other to send and receive data, this algorithm is confined to only point-to-point communication between the neighbors. This gets rid of the communication contention. The redundant work problem of the previous algorithms is solved by intelligent partitioning of the *Candidate Itemsets*.

6. Hybrid Distribution

The Intelligent *Data Distribution Algorithm* utilizes the memory by partitioning the *Candidate Itemsets* among all the processors. With the increase in the number of processors, the number of candidates assigned to them decreases. Therefore, with fewer candidates per processor, balancing the work gets difficult. Also, the hash tree gets smaller due to fewer candidates resulting in the amount of computation being less than the communication involved. This reduces the efficiency and poses serious problems for the systems which cannot perform asynchronous communication. The *Hybrid Distribution Algorithm* combines the *Count Distribution Algorithm* and the *Data Distribution Algorithm* to overcome these problems. This

DOI: 10.18535/ijsrm/v5i6.34

algorithm partitions the transactions and the *Candidate Itemsets* among the processors of equal size groups. Within these groups, the Intelligent *Data Distribution Algorithm* is used to compute the counts. The *Hybrid Distribution Algorithm*, while retaining the goodness of the Intelligent *Data Distribution Algorithm*, provides good load balance and cuts down data movement.

7. Conclusions

In this paper, we have covered the essence of the popular parallel algorithms developed so far in the area of *Frequent Itemset Mining*. Developing parallel algorithms has proved to be a very challenging yet exciting task. Going by the interests shown by the researchers, advancements in the field over time, and ever increasing demands of speed and resource utilization, the future of the research in this area promises exciting results.

References

[1] V. Kumar and M. Joshi, "Tutorial on High Performance Data Mining", International Conference on High Performance Computing (HiPC-98), Dec. 1998.
[2] M. Chen, J. Han and P. S. Yu, "Data Mining: An overview from a Database Perspective", IEEE Transactions on Knowledge and Data Engineering, vol. 8, no. 6. Dec. 1996, pp. 866-883.
[3] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules", Proc. of the 20th VLDB Conference, Santiago, Chile, 1994, pp. 487-499.
[4] R. Agrawal and J. C. Shafer, "Parallel Mining of Association Rules", IEEE Transactions on Knowledge and Data Engineering, vol. 8, no. 6. Dec. 1996, pp. 962-969.
[5] E. Han, G. Karypis and V. Kumar, "Scalable Parallel Data Mining for Association Rules", Proc. 1997 ACM-SIGMOD Int. Conf. on Management of Data, Tucson, Arizona, 1997.
[6] E. Han, G. Karypis and V. Kumar, "Scalable Parallel Data Mining for Association Rules", IEEE Transactions on Knowledge and Data Engineering, vol. 12, no. 3, May/June 2000, pp. 337-352.
[7] M. V. Joshi, E. Han, G. Karypis and V. Kumar, "Efficient Parallel Algorithms for Mining Associations", URL: http://www.cs.umn.edu/~kumar.
[8] M. J. Zaki, "Parallel and Distributed Association Mining: A Survey", IEEE Concurrency, Oct. - Dec. 1999, pp. 14-25.
[9] Vipin Kumar, Ananth Grama, Anshul Gupta, and George Karypis, "Introduction to parallel computing: Algorithm Design and Analysis", Benjamin Cummings/Addison Wesley, Redwood City, 1994.
[10] Han J. and Kamber M. (2006) "Data Mining Concepts and Techniques", Second edition, Morgan Kaufmann Publishers.
[11] Agarwal C. C. and Han J. (2014) "Frequent Pattern Mining", Springer, 2014.
[12] Alex Berson and Stephen J. Smith, "Data Warehousing, Data Mining, & OLAP", TMH Edition-2004.

Author Profile

Nafisur Rahman received the B.Tech and M.Tech degrees in Computer Science and Engineering in 2012 and 2014, respectively. Since 2015, he has been teaching various courses, as an Assistant Professor, in the Department of Computer Science and Engineering (School of Engineering Sciences and Technology), Jamia Hamdard, New Delhi

Samar Wazir received the B.Tech in Information Technology and M.Tech in Computer Science in 2008 and 2012, respectively. Since 2014, he has been teaching various courses, as an Assistant Professor, in the Department of Computer Science and Engineering (School of Engineering Sciences and Technology), Jamia Hamdard, New Delhi