ISSN: 2277-9655
Impact Factor: 1.852

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

One Time Mining by Multi-Core Preprocessing on Generalized Dataset

Aviral Kumar Singh[*1], S. R. Tondon[2], Tarun Dhar Diwan[3]
aviralkumarsingh@live.com

Abstract

One of the important problems in data mining is discovering association rules from databases of transactions where each transaction consists of a set of items. Many industries are interested in developing the association rules from their databases due to continuous retrieval and storage of huge amount of data. The discovery of interesting association relationship among business transaction records in many business decision making process such as catalog decision, cross-marketing, and loss-leader analysis. The enormity and high dimensionality of datasets typically available as input to problem of association rule discovery, and the time consuming operation in this discovery process is the computation of the frequency of interesting subset of items (called candidates) in the database of transactions. Hence, it is has become vital to develop a method that will make speedup the preprocessing computation. In this paper, We have proposed An Integrated approach of Parallel Computing and ARM for mining Association Rules in Generalized data set that is fundamentally different from all the previous algorithms in that multi-core preprocessing is done and by avoiding recurring scan of dataset number of passes required is reduced. The response time is calculated on space delimited text dataset.

Keywords: Data Mining, Association Rule Mining (ARM), Association rules, Apriori algorithm, Frequent pattern.

Introduction

The rapid development of computer technology, especially increased capacities and decreased costs of storage media, has led businesses to store huge amounts of external and internal information in large databases at low cost. Mining useful information and helpful knowledge from these large databases has thus evolved into an important research area [3, 2, 1].

Association rule mining (ARM) [18] has become one of the core data mining tasks and has attracted tremendous interest among data mining researchers. ARM is an undirected or unsupervised data mining technique which works on variable length data, and produces clear and understandable results. Association Rule Mining (ARM) algorithms [17] are defined into two categories; namely, algorithms respectively with candidate generation and algorithms without candidate generation. In the first category, those algorithms which are similar to Apriori algorithm for candidate generation are considered. Eclat may also be considered in the first category [8]. In the second category, the FP-Growth algorithm is the best–known algorithm.

The main drawback of earlier algorithms is the repeated scans over large database. This may be a cause of decrement in CPU performance, memory and increment in I/O overheads. The performance and efficiency of ARM algorithms mainly depend on three factors; namely candidate sets generated, data structure used and details of implementations [8]. In this paper we have proposed an Algorithm which uses these three factors. Suppose if there are 104 frequent 1 itemsets, Apriori algorithm may produce 107 candidate 2 itemsets, count them and judge their frequency [11]. Besides, it may produce as many as 2100 (about 1030) candidate itemsets in order to find the frequent itemset which includes 100 items.What's more, it may scan the database many times to check a lager candidate through matching mode.

Transactional database is considered as a two dimension array which works on generalized value dataset. The main difference between proposed algorithm and other algorithms is that instead of using transactional array in its natural form, our algorithm uses transpose of array i.e. rows and columns of array are interchanged and transposition is done using parallel matrix transpose algorithm (Mesh Transpose) [20]. The parallel architecture that lends itself most naturally to matrix operations is the mesh. Indeed, an n x n mesh of processors can be regarded as a matrix and is therefore perfectly fitted to accommodate an n x n data matrix, one element per processor. This is precisely the approach we shall use to compute the transpose of an n x n matrix initially stored in an n x n mesh of processors. We find that the time taken for matrix transpose decreases with an increase in the number of processors. We also observe that the speedup is very high for small as well as very large size of matrix when we increase the number of processors. The idea of our algorithm is quite simple.

ISSN: 2277-9655
Impact Factor: 1.852

Since the diagonal elements are not affected during the transposition, that is, element aii of A equals element aii of AT, the data in the diagonal processors will stay stationary.

The advantage of using transposed array is to calculate support count for particular item. There is no need to repeatedly scan array. Only by finding the row sum of the array will give the required support count for particular item, which ultimately results in increased efficiency of the algorithm.

The remainder of this paper is organized as follows: Section 2 provides a brief review of the related work. In Section 3, we explain Frequent Itemset and Association Rule Mining through Apriori Algorithm. In Section 4, we introduce our approach of frequent itemset generation using parallel preprocessing. An illustration of the algorithm and experiment analysis is presented in section 5 and section 6 respectively. Finally, we concluded our work.

Related Work

One of the most well known and popular data mining techniques is the Association rules or frequent item sets mining algorithm. The algorithm was originally proposed by Agrawal et al. [4] [5] for market basket analysis. Because of its significant applicability, many revised algorithms have been introduced since then, and Association rule mining is still a widely researched area.

Agrawal et. al. presented an AIS algorithm in [4] which generates candidate item sets on-the-fly during each pass of the database scan. Large item sets from previous pass are checked if they are present in the current transaction. Thus new item sets are formed by extending existing item sets. This algorithm turns out to be ineffective because it generates too many candidate item sets. It requires more space and at the same time this algorithm requires too many passes over the whole database and also it generates rules with one consequent item.

Agrawal et. al. [5] developed various versions of Apriori algorithm such as Apriori, AprioriTid, and AprioriHybrid. Apriori and AprioriTid generate item sets using the large item sets found in the previous pass, without considering the transactions. AprioriTid improves Apriori by using the database at the first pass. Counting in subsequent passes is done using encodings created in the first pass, which is much smaller than the database. This leads to a dramatic performance improvement of three times faster than AIS.

Scalability is another important area of data mining because of its huge size. Hence, algorithms must be able to “scale up” to handle large amount of data. Eui-Hong et. al [16] tried to make data distribution and candidate distribution scalable by Intelligent Data Distribution (IDD) algorithm and Hybrid Distribution (HD) algorithm respectively. IDD addresses the issues of communication overhead and redundant computation by using aggregate memory to partition candidates and move data efficiently. HD improves over IDD by dynamically partitioning the candidate set to maintain good load balance. Different works are reported in the literature to modify the Apriori logic so as to improve the efficiency of generating rules. These methods even though focused on reducing time and space, in real time still needs improvement.

Frequent Item Set And Association Rule

The aim of Association rule mining is exploring relations and important rules in large datasets. A dataset is considered as a sequence of entries consisting of attribute values also known as items. A set of such item sets is called an item set. Frequent item sets are sets of pages which are visited frequently together in a single server session.

Let I ={ I1, I2, … , Im }be a set of items. Let D, the task-relevant data, be a set of database transactions where each transaction T is a set of items such that $T \subseteq I$. Each transaction is associated with an identifier, called TID. Let A be a set of items. A transaction T is said to contain A if and only if $A \subseteq T$. An association rule is an implication of the form $A \Rightarrow B$, where $A \subset I$, $B \subset I$, and $A \cap B = \varnothing$. The rule $A \Rightarrow B$ holds in the transaction set D with support s, where s is the percentage of transactions in D that contain $A \cup B$ (i.e., the union of sets A and B, or say, both A and B). This is taken to be the probability, $P(A \cup B)$. The rule $A \Rightarrow B$ has confidence c in the transaction set D, where c is the percentage of transactions in D containing A that also contain B. This is taken to be the conditional probability, P(B|A). That is,

$$support(A \Rightarrow B) = P(A \cup B) \quad \text{.....(2.1)}$$
$$confidence(A \Rightarrow B) = P(B|A) \quad \text{.....(2.2)}$$

A set of items is referred to as an itemset. An itemset that contains k items is a k-itemset. The set {bread, butter} is a 2-itemset. The occurrence frequency of an itemset is the number of transactions that contain the itemset, it is also known, as the frequency, or support count. If the relative support of an itemset I satisfies a pre specified minimum support threshold then I is a frequent itemset. The set of frequent k-itemsets is commonly denoted by Lk. From Equation (2.2), we have

$$confidence(A \Rightarrow B) = P(B,A) = \frac{support(A \cup B)}{support(A)} = \frac{suppor_{count(A \cup B)}}{support_{count(A)}}$$

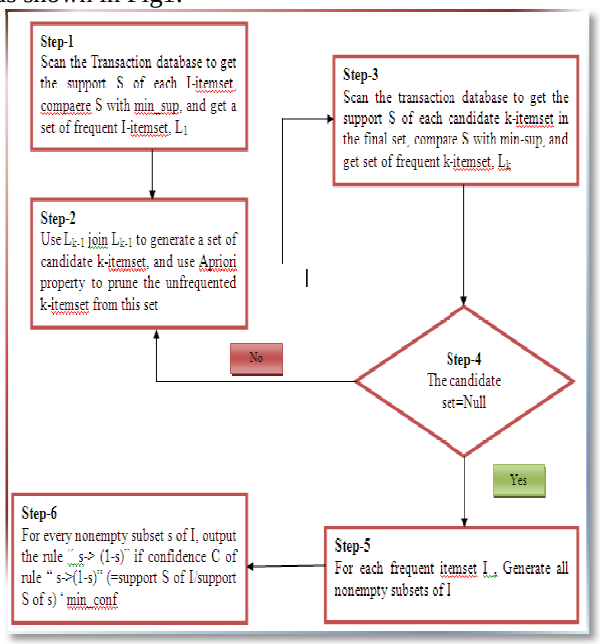
ISSN: 2277-9655
Impact Factor: 1.852

Let τ = I1, I2... Im be a set of binary attributes, called items. Let T be a database of transactions. Each transaction t is represented as a binary vector, with t[k] = 1 if t bought the item Ik, and t[k] = 0 otherwise. There is one tuple in the database for each transaction. Let X be a set of some items in τ. We say that a transaction t satisfies X if for all items Ik in X, t[k] = 1.

By an association rule, we mean an implication of the form X⇒Ij, where X is a set of some items in τ, and Ij is a single item in τ that is not present in X. The rule X ⇒ Ij is satisfied in the set of transactions T with the confidence factor 0≤ c ≤ 1 if at least c% of transactions in T that satisfy X also satisfy Ij. We will use the notation X ⇒Ij | c to specify that the rule X ⇒ Ij has a confidence factor of c.[3]

A. Apriori Algorithm

The Apriori algorithm is one of the most popular algorithms for mining frequent patterns and association rules [4]. It introduces a method to generate candidate itemsets Ck in the pass k of a transaction database using only frequent itemset Lk−1 in the previous pass. The idea rests on the fact that anysubset of a frequent itemset must be frequent as well. Hence, Ck can be generated by joining two itemsets in Lk−1 and pruning those that contain any subset that is not frequent as shown in Fig1.



: Apriori Algorithm

Our Approach

Association rule and frequent itemset mining has become now a widely research area and hence, faster and faster algorithms have been presented.The Association Rule Mining algorithms such as Apriori, FP-Growth requires repeated scans over the entire database. All the input/output overheads that are being generated during repeated scanning the entire database decrease the performance of CPU, memory and I/O overheads.

Much work has been carried out on improving the efficiency of the apriori algorithm by reducing the I/O time and minimizing the set of candidate itemsets. However, all these works suffer from problem of scans over the databse at least once. The efficiency of these algorithms can still be improved by reducing the time required for counting the supports of candidate itemsets. We aim to obtain an efficient algorithm which reduces the time needed to count the supports of candidate itemsets.

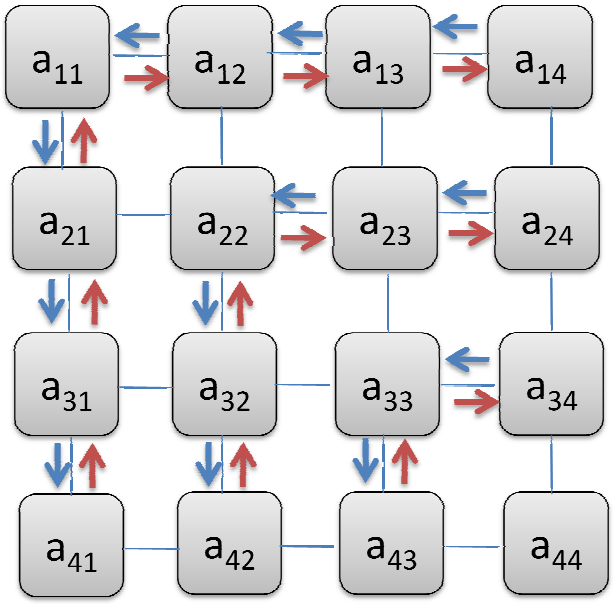
The process of finding large itemsets is divided into following parts.

- Parallel Data Preprocessing
- Generating candidate sets.

A. Parallel Data Preprocessing

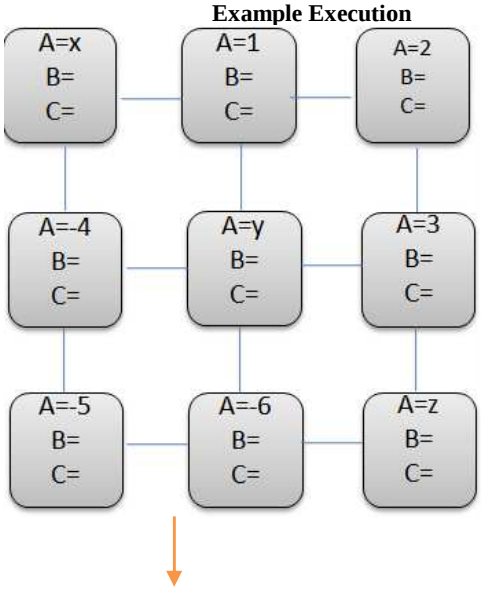
The idea of our algorithm is quite simple. Since the diagonal elements are not affected during the transposition, that is, element aii of A equals element aii of A^T, the data in the diagonal processors will stay stationary. An n x n mesh of processors can be regarded as a matrix and is therefore perfectly fitted to accommodate an n x n data matrix, one element per processor. A(i,j) is used to store aij initially and aji when the algorithm terminates. B(i,j) is used to store data received from P(i,j + 1) or P(i - 1,j), that is, from its right or top neighbors. C(i,j) is used to store data received from P(i,j - 1) or P(i + 1,j), that is, from its left or bottom neighbors.

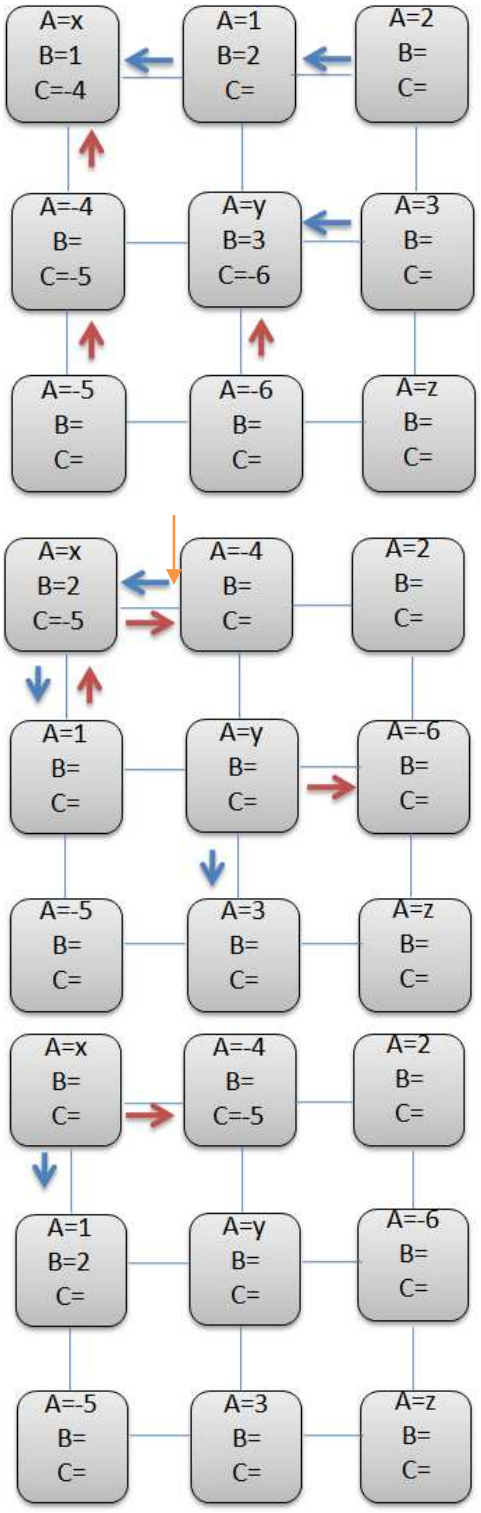
ISSN: 2277-9655
Impact Factor: 1.852



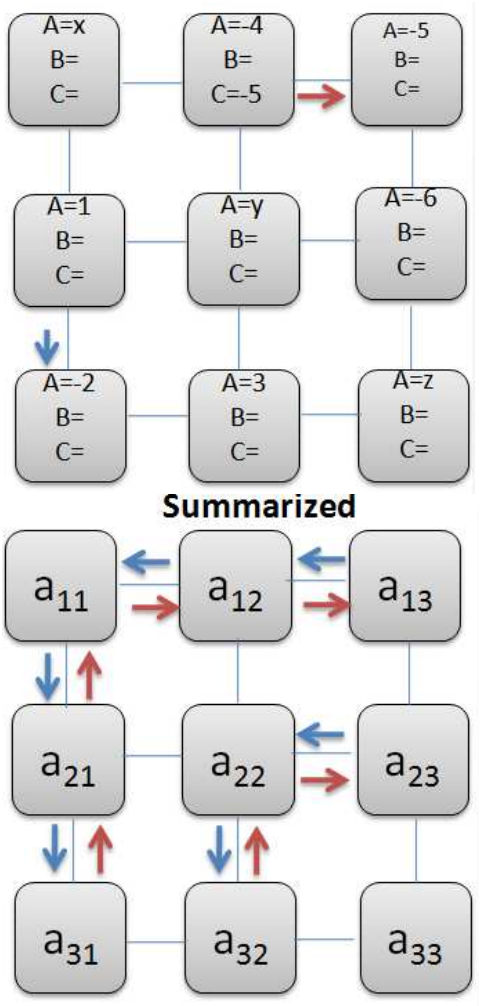
Matrix to be transposed, stored in mesh of processors.

Example Execution



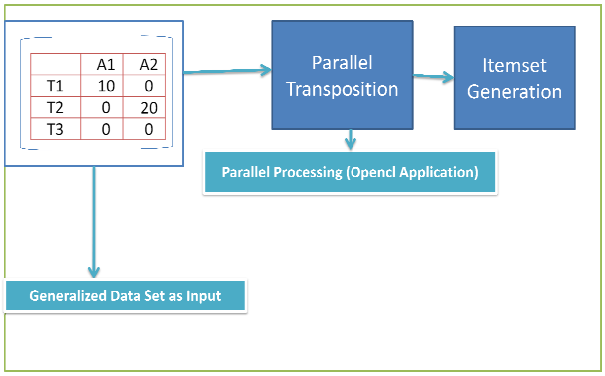


ISSN: 2277-9655
Impact Factor: 1.852



B. Candidate set Generation

We propose a new algorithm in which Transactional database is considered as a two dimension array which works on generalized value dataset. The main difference between proposed algorithm and other algorithms is that instead of using transactional array in its natural form, our algorithm uses transpose of array i.e. rows and columns of array are interchanged and transposition is achieved using parallel matrix transpose algorithm.



Algorithm
// Transpose the transactional database
1. Transpose(Data Set)
2. Read the database to count the support of C1 to determine L1 using sum of rows.
3. L_1= Frequent 1- itemsets and k:= 2
4. While (k-1 ≠ NULL set) do
Begin
C_k: = Call Gen_candidate_itemsets (L_k-1)
Call Prune (C_k)
for all itemsets i □ I do
Calculate the support values using dot-multiplication of array;
L_k := All candidates in Ck with a minimum support;
k:=k+1
End
5. End of step-4
End Procedure

An Illustration

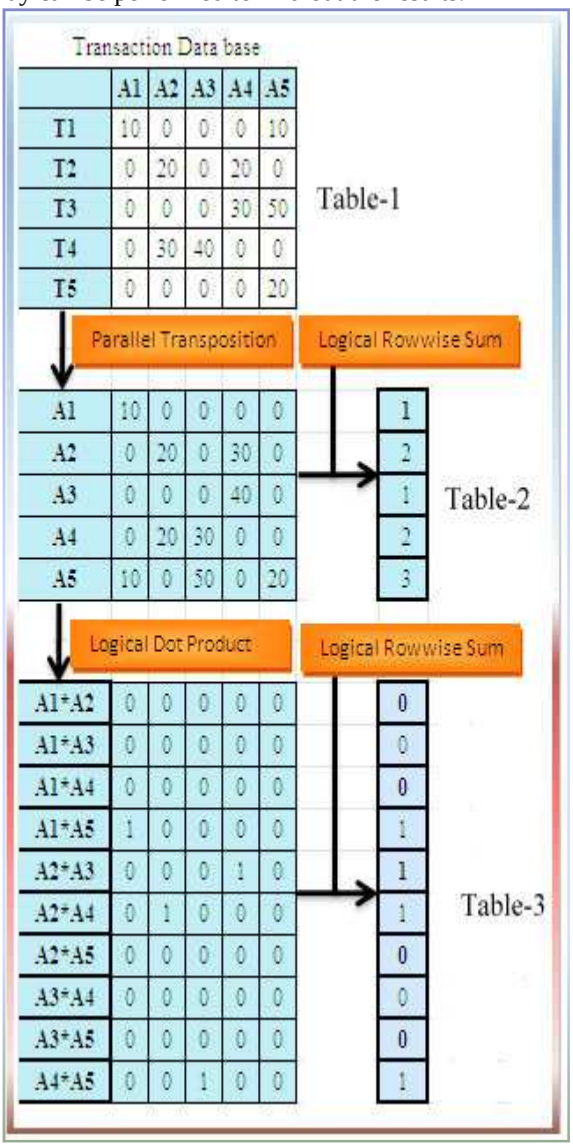
Suppose we have a transactional database in which the user transactions from T1 to T5 and items from A1 to A5 are stored in the form of generalized values, which is shown in Table-1(Fig 4)

Consider the transpose of transactional database as shown in Table-1 is stored in Table-2 by applying Parallel Transposition that can be used in our proposed algorithm. Assume the user specified minimum support is 40%, and then the steps for generating all frequent item sets in proposed algorithm will be repeated until NULL set is reached. In our algorithm, transactional dataset will be used in the transposed form. Therefore, candidate set and frequent itemset generation process will be changed as compared to Apriori algorithm.

Then the candidate 2-itemset will be generated by performing dot-multiplication of rows of array, as array consist of generalized values, the resultant cell will be produce in the form of 1. If the corresponding cells of the respective rows have 1, otherwise 0 will be in the resultant cell. In this approach, we will receive a new array consisting of candidate 2-itemsets to get the higher

ISSN: 2277-9655
Impact Factor: 1.852

order of itemsets. The above process between rows of array can be performed to find out the results.



An Illustration of our approach

Experimental Evaluations

The performance comparison of our mining algorithm with classical frequent pattern-mining algorithm Apriori is shown in Fig 7 . All the experiments are performed on 1.50Ghz Pentium-iv desktop machine with 256 MB main memory, running on Windows-XP operating system. The program for Apriori and our proposed algorithm were developed in Java JDK1.5 environment.

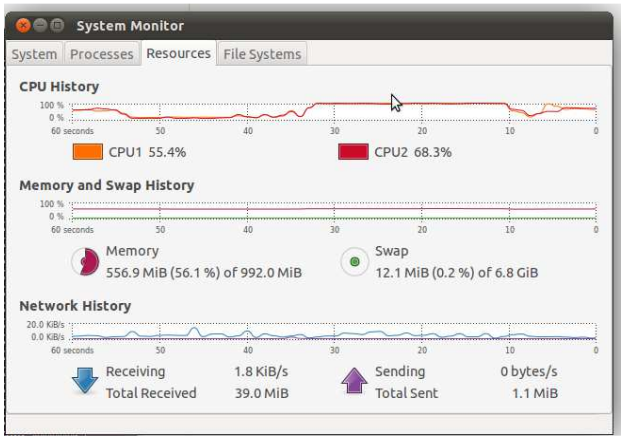


Figure 5: CPU performance of Sequential Execution

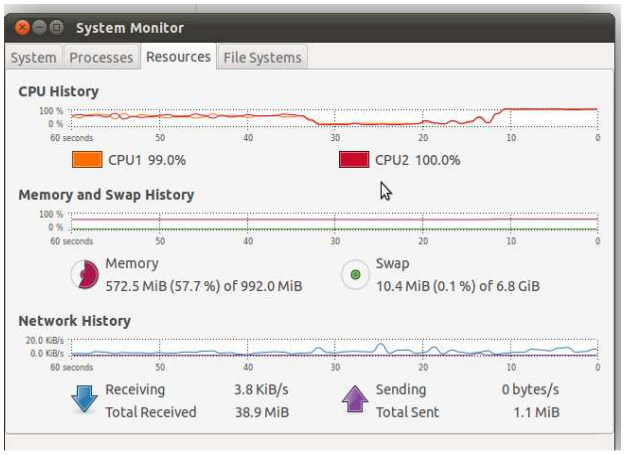


Figure 6: CPU performance of Parallel Execution

Figure 7 compares the time taken by the apriori algorithm and our algorithm for different support values. For lower support values algorithm takes more time because it generates too many candidate itemsets. These candidate itemsets are then tested for minimum support. It is obvious form the figure that as the support value increases time taken by the algorithm decreases. In our algorithm, only by finding the row sum of the array will give the required support count for particular item, which ultimately results in increased efficiency of the algorithm.

ISSN: 2277-9655
Impact Factor: 1.852

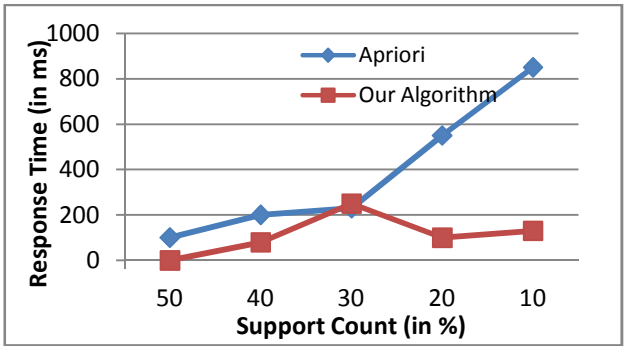


Figure 7: Performance analysis of algorithms.

Conclusions

ARM algorithms are important to discover frequent itemsets and patterns from large databases. In this project, we have designed An Algorithm for generation of frequent itemsets similar to Apriori algorithm. The proposed algorithm can improve the efficiency of Apriori algorithm and it is observed to be very fast. Our algorithm is not only efficient but also very fast for finding association rules in large databases. The proposed algorithm drastically reduces the I/O overhead associated with Apriori algorithm and retrieval of support of an itemset is quicker as compared to Apriori algorithm. This algorithm may be useful for many real-life database mining scenarios where the data is stored in generalized form. Our algorithm uses parallel transposition of generalized 2D data set, so the data preprocessing goes faster. This algorithm cannot be used with multimedia dataset.

References

[1] *C.-Y. Wang, T.-P. Hong and S.–S. Tseng. "Maintenance of discovered sequential patterns for record deletion". Intell. Data Anal. pp. 399-410, February 2002.*

[2] *M.S. Chen, J.Han and P.S. Yu. "Data Mining : An overview from a database perspective", IEE Transactions on Knowledge and Data Engineering 1996.*

[3] *R.Agrawal, T. Imielinksi and A. Swami, "Database Mining: a performance perspective", IEE Transactions on knowledge and Data Engineering, 1993.*

[4] *Agrawal, R., Imielinski, T., and Swami, A. N. "Mining Association Rules Between Sets of Items in Large Databases". Proceedings of the ACM SIGMOD, International Conference on Management of Data, pp.207- 216, 1993.*

[5] *Agrawal. R., and Srikant. R., "Fast Algorithms for Mining Association Rules", Proceedings of 20th International Conference of Very Large Data Bases. pp.487-499,1994.*

[6] *Jong Park, S., Ming-Syan, Chen, and Yu, P. S. "Using a Hash-Based Method with transaction Trimming for Mining Association Rules". IEEE Transactions on Knowledge and Data Engineering, 9(5), pp.813-825,1997.*

[7] *M.H.Margahny and A.A.Mitwaly, "Fast Algorithm for Mining Association Rules" in the conference proceedings of AIML, CICC, pp(36-40) Cairo, Egypt, 19-21 December 2005.*

[8] *Y.Fu., "Discovery of multiple-level rules from large databases", 1996.*

[9] *F.Bodon, "A Fast Apriori Implementation", in the Proc.1st IEEE ICDM Workshop on Frequentc Itemset Mining Implementations (FIMI2003, Melbourne,FL).CEUR Workshop Proceedings 90, A acheme, Germany 2003.*

[10] *Akhilesh Tiwari, Rajendra K. Gupta, and Dev Prakash Agrawal, "Cluster Based Partition Approach for Mining Frequent Itemsets" in the International Journal of Computer Science and Network Security(IJCSNS), VOL.9 No.6,pp(191-199) June 2009.*

[11] *JiaWei Han Micheline Kamber."Data Mining:Concepts and Techniques"[M].Translated by Ming FAN, XaoFeng MENG etc. mechanical industrial publisher,BeiJing,2001,150-158.*

[12] *M.J. Zaki. "Scalable algorithms for association mining". IEEE Transactions on Knowledge and Data Engineering, 12 : 372 –390, 2000.*

[13] *Jochen Hipp, Ulrich G¨untzer, Gholamreza Nakhaeizadeh. "Algorithms for Association Rule Mining – A General Survey and Comparison".ACM SIGKDD, July 2000, Vol-2, Issue 1, page 58-64.*

[14] *Sotiris Kotsiantis, Dimitris Kanellopoulos. "Association Rules Mining: A Recent Overview". GESTS International Transactions on Computer Science and Engineering, Vol.32 (1), 2006, pp. 71-82.*

[15] *S. Brin, R. Motwani, J. D. Ullman, AND S. Tsur, "Dynamic itemset counting and implication rules for market basket data", SIGMOD Record 26(2), pp. 255–276, 1997.Kim Man Lui, Keith C.C. Chan, and John Teofil Nosek "The Effect of Pairs in Program Design Tasks" IEEE transactions on software engineering, VOL. 34, NO. 2, march/april 2008.*

[16] *Eui-Hong Han, George Karypis, and Kumar, V. Scalable "Parallel Data Mining for Association Rules". IEEE Transaction on Knowledge and Data Engineering, 12(3), pp.728-737, 2000.*

ISSN: 2277-9655
Impact Factor: 1.852

[17]Sanjeev Kumar Sharma & Ugrasen Suman "A Performance Based Transposition Algorithm for Frequent Itemsets Generation" International Journal of Data Engineering (IJDE), Volume (2) : Issue (2) : 2011

[18]Dr (Mrs).Sujni Paul "An Optimized Distributed Association Rule Mining Algorithm In Parallel and Distributed Data Mining With Xml Data For Improved Response Time".International Journal Of Computer Science And Information Technology, Volume 2, Number 2, April 2010

[19]Manoj Bahel, Chhaya Dule "Analysis of frequent item set generation process in Apriori & RCS (Reduced Candidate Set) Algorithm" National Conference on Information and Communication Technology, Banglore April 2010

[20]Sedukhin, S.G.; Zekri, A.S.; Myiazaki, T."Orbital Algorithms and Unified Array Processor for Computing 2D Separable Transforms" Parallel Processing Workshops (ICPPW), 2010 39th International Conference Page(s): 127 – 134