A Combined Horizontal Parallel Apriori Algorithm and Adaptive Frequent Pattern Growth Algorithm for Big Data Mining

M. Sornalakshmi, S. Balamurali, M. Venkatesulu

Abstract*: Due to the massive data size and complexness, big data mining using a sole computer is a problematic task. With the rapid increase in the database size, parallel and distributed computing systems can yield better benefits in the data mining applications. Parallelization of the Association Rule Mining (ARM) algorithms is a significant task in the data mining application for effectively mining the frequent itemsets from the large-size databases. These mining algorithms allocate the database in a horizontal manner or increase the number of processors to decrease the overall time necessary for mining the frequent itemsets. In this paper, a combined Horizontal Parallel-Apriori (HP-Apriori) and Adaptive Frequent Pattern (FP) Growth algorithm is proposed to divide the database both horizontally and vertically into four sub-processes, for parallel processing of all four tasks. The Horizontal Parallel-Apriori algorithm increases the speed of the mining process using an index file. Adaptive Binomial Distribution (ABD) is applied to the Frequent Pattern Growth Algorithm to find the minimum support for mining the optimal frequent itemsets. Experimental analysis established that the combined algorithm outperforms in terms of minimizing the overall execution time and increasing the computational speed in high scalability.

Keywords: Apriori Algorithm, Big Data Mining, Frequent Pattern Growth Algorithm, Parallel and Distributed Processing

I. INTRODUCTION

The swift advancement of the data storage capability and data collection capability resulted in the prompt growth in the size of the database in various fields[1]. Big data processing involves huge data that needs additional real-time study[2]. Big data analytics using the machine learning techniques and data mining approaches has attracted the attention of the researchers. Due to the huge data size and high complexity, big data mining is extremely difficult with the present data mining techniques. Hence, this makes it as a great challenge for finding the valuable information from the big data due to the huge volume, dissimilar and independent sources with the distributed control to discover the intricate relationship between data [1].ARM discovers uniformities in the transaction database that results in the implementation of better tactical business decisions [2, 3]. The two stages of ARM are mining of the frequent itemsets and extraction of the association rules.

The Apriori algorithm is one among the mainly prevailing data mining algorithms used to extract useful information from the database [4]. For huge datasets, the existing ARM algorithms could not deliver effective mining results within a stipulated time. The Apriori algorithm applies a breadth-first search concept to calculate the support count of the frequent itemsets. Hence, the Apriori algorithm is less effective than other mining algorithms. During application of the ARM to the big data, the difficulty of mining the frequent itemsets is higher than the extraction of the association rules. This is due to the reason that scanning of the whole database is required for computing the support count of each frequent itemset. The confidence of the strong association rules is calculated simply. Thus, mining of the frequent itemset requires more time than the ARM step [5]. The parallel prototype is considered as an effective way to handle this issue. Maximum big-data based mining algorithms are based on the parallelism concept. The data is separated into the fragments and the mining operation is performed in parallel. Parallel-based mining algorithms partition the data in a horizontal or vertical manner, and perform data mining task on each subset. At last, the results are combined with their corresponding support count.

Apriori algorithm and FPGrowth algorithm are the extensively employed algorithms for mining the frequent itemsets. However, these algorithms suffer from an issue in defining the minimum support threshold (min_sup) for mining the frequent itemsets. If the min_sup is set too low, several frequent itemsets will be generated. This may cause these algorithms to be inefficient and consume more memory space. If the min_sup is set high, less frequent itemsets are mined. This work solves this issue by using the ABD to find appropriate min_sup adaptively. This facilitates in the mining of the optimal frequent itemsets.

HP-Apriori algorithm mines the frequent itemsets from the big data in a parallel manner. The dataset is partitioned in both horizontal and vertical ways into four subsets. The itemset mining process is furthermore divided into the sub-tasks: candidate itemset creation, support count computation and association rule combination. A node generates each candidate itemset and combines the association rules. Hence, the time needed for calculating the support count is highly greater than other tasks. The algorithm generates an index file to improve the support count computational speed.

The paper is systematized in the subsequent way: Section II presents a brief overview of

Revised Manuscript Received on December 16, 2019.
* Correspondence Author
M.Sornalakshmi, Department of Computer Applications, Kalasalingam Academy of Research and Education, sorna.jesus@gmail.com,
S. Balamurali*, Department of Computer Applications, Kalasalingam Academy of Research and Education, sbmurali@rediffmail.com*
M. Venkatesulu, Department of Computer Applications, Kalasalingam Academy of Research and Education, venkatesulum2000@gmail.com

the existing works relevant to the mining of the frequent itemsets from big data. Section III describes the horizontal parallel Apriori algorithm and Frequent Pattern growth algorithm. Section IV presents the comparative analysis of the proposed combined algorithm and existing techniques. The merits of the work are determined in Section V.

II. EXISTING MINING ALGORITHMS

In the parallel data mining process, the huge data is separated into smaller portions. Parallelism approach focuses on partitioning the huge data into smaller portions, such that each portion is processed individually using a single processor. Hence, the number of support count calculations can be done simultaneously and parallelly[1]. In [6], the partition algorithm emphases on the efficient partitioning of the dataset on a huge database. A level-wise algorithm is used for generating local frequent itemsets and collecting their global support counts during a second pass. Due to the availability of minimum information in the minimum portions, this algorithm yields many False Positives (FPs).

Park et al. [7]formulated Parallel Data Mining (PDM) algorithm to identify the global set of large frequent itemsets and minimize the required amount of inter nodal data exchange. The large frequent itemsets are calculated by replacing their support count. Agrawal and Shafer [8]suggested parallel data mining algorithms for mining the association rules. In the Count Distribution algorithm, the local candidate itemsets are created from the large itemsets found at the previous iteration[9]. The Candidate Distribution algorithm divides the candidate itemsets, and replicates the itemsets in a selective way despite of partitioning and switching the database transactions. The Data Distribution algorithm divides the entire database and assigns the candidate itemsets to various processors. But, this Data Distribution algorithm suffered from large communication overhead due to scanning of the entire database in all iterations.

Mueller [10] proposed an algorithm incorporating both Transaction Identifier (TID)-lists and partitioning approach. The prefix tree is used to avoid counting of the needless candidates. Vu and Alaghb and [11] proposed a new parallel method for finding the frequent patterns. Two mining schemes are combined and the optimal appropriate mining scheme is applied to each subset to acclimate to the characteristics of the data and execute faster on the sparse and dense databases. The novel lock-free scheme of this method reduces the synchronization requirements and improves the data individuality to yield high scalability.

III. PROPOSED WORK

The Apriori algorithm includes the following stages

- Candidate itemset creation at the 'n'th level:

❖ The Apriori Algorithm associates the frequent itemsets of size (n-1) with the 1-itemsets for creating 'n' number of candidate itemsets with the frequency measure.

- Support value computation at the 'n'th level:

❖ The frequencies of all candidate itemsets are calculated and the frequent itemsets among them are determined.

These stages are repeated continuously till not any new frequent itemset is mined at the level 'n'. Initially, the Apriori algorithm performs database scanning for counting the support value of the 1-itemsets. The frequent 1-itemsets are selected and supersets are created, while passing through the transaction database again. The candidate itemsets are generated by associating the frequent (n-1) itemsets with the frequent 1-itemsets. Subsequently, the support count of the novel candidate itemsets are calculated by scanning the dataset and removing the itemsets having lowest support threshold than the minimum support threshold. This involves several passes through the whole dataset. Finally, a list of frequent n-itemsets is generated as the output.

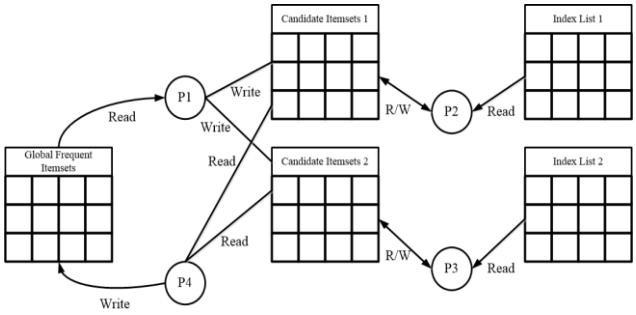

Figure 1 Relationship among the sub-process in the Horizontal Parallel Apriori Algorithm

The Horizontal Parallel Apriori Algorithm [12] involves four sub-processes for mining the frequent itemsets from the big data. The sub-processes are candidate generation phase, two support count calculation phases and frequent itemset determination phase. The support value calculation process consumes more time for scanning the entire big dataset, due to the high size of the transactions in the big dataset. Figure 1 depicts the relationship between the sub-processes in the Horizontal Parallel Apriori Algorithm.

During each iteration, the list of the candidate itemsets is regenerated and global frequent itemsets is updated. Both the index lists are created at the first step. The first phase generates two lists of candidate itemset by using the frequent (n-1) itemsets by inserting the candidate itemset one by one, so that a candidate itemset is introduced in the primary list and next candidate itemset is inserted into the secondary list. The second and third phases count the support of the itemsets 1 and 2 using the index lists 1 and 2, respectively. Lastly, the support counts are added together into the lists of the candidate itemset.

The fourth phase combines both the candidate itemsets 1 and 2 while determining the global support for each candidate itemset and inserting the combined itemsets into the list of global frequent itemsets. These steps are executed for n>1. For k=1, these steps are employed in different ways as described follows:

The database is separated in the horizontal direction into twofold partitions of similar data size. Each horizontal partition is vertically divided into twofold subsets. The big data is again separated into four subsets that are further allocated to the parallel processing. In the first step, each node separately creates the candidate 1-itemsets and index list for each subset. Then, phases 1 and 4 combine the local 1-itemset lists while the phases 2 and 3 merge the local index lists, such that a single index list file is created for every horizontal partition. Finally, the phases 2 and 3 calculate the support count of 1-itemsets. Phase 4 groups two 1-itemset lists to

define the global frequent 1-itemset list.

Figure 2 shows the flow chart of the FP growth Frequent Itemset algorithm. The first step is to input the transaction dataset. Secondly, the number of specific items should be calculated from the transaction dataset. Then, the total number of items and ABD of each frequent item are calculated. The optimal adaptive threshold is found out by using the ABD and applied to the Frequent Pattern growth algorithm for mining the optimal frequent itemsets [13].

HP Apriori and FP growth Frequent Itemset Algorithm

Input: 'Q'–Transaction database, 'S' - Support threshold and 'I' -Item prefix such that $I \subseteq J$.
Output: $F[I](Q,S)$ is a list of frequent itemsets
$C_1 = \{\{i\} | i \in I\};$
N=1;
If (N=1)
{
For each transaction $T_j \in partition$ do
For each candidate itemset$c_1 \neq C_N$ do
If $(c_1 \subseteq T)$ then
{
$S(c_1) = S(c_1) + 1$
$L1\{\ \} \leftarrow T_j;$ //Phase 1
$L2\{\ \} \leftarrow T_j;$ //Phase 2 and Phase 3
$L4\{\ \} \leftarrow T_j;$ //Phase 4
}
$H1\{\ \} \leftarrow c_1;$ //index list of Phase 1
$H2\{\ \} \leftarrow c_1;$ //index list of Phase 2 and Phase 3
$H4\{\ \} \leftarrow c_1;$ //index list of Phase 4
$H\_P1 \leftarrow Combination\ of\ H1\ and\ H2;$//Phase 1
$L\_P1 \leftarrow Combination\ of\ L1\ and\ L2;$ //Phase 2 and Phase 3
$H\_P2 \leftarrow Combination\ of\ H1\ and\ H2;$//Phase 4
N++
}
Else
{
While $(C_n \neq \emptyset)$
Generation of candidate n-itemsets; //Phase 2 and Phase 3
Support value calculation for H_P1; //Phase 2 and Phase 3
Combine $H\_P1\ and\ H\_P2$; // Phase 4
Identify frequent itemsets; //Phase 4
N++;
}

$F[I] \leftarrow \{\ \}$
for all $i \in J$occurring in Q **do**
$F[I] \leftarrow F[I] \cup \{I \cup \{i\}\}$
Create Q_i
$Q_i \leftarrow \{\ \}$
$H \leftarrow \{\ \}$
for $j \in J$occurring in Q so that $j > i$**do**
if support $(I \cup \{i,j\}) \geq T$**then**
$H \leftarrow H \cup \{j\}$
for all $(tid, A) \in Q\ with\ I \in A$**do**
$Q_i \leftarrow Q_i \cup \{(tid, A \cap H)\}$
Depth-first recursion
Calculate $F[I \cup \{i\}](Q_i, S)$
$F[I] \leftarrow F[I] \cup F[I \cup \{i\}]$

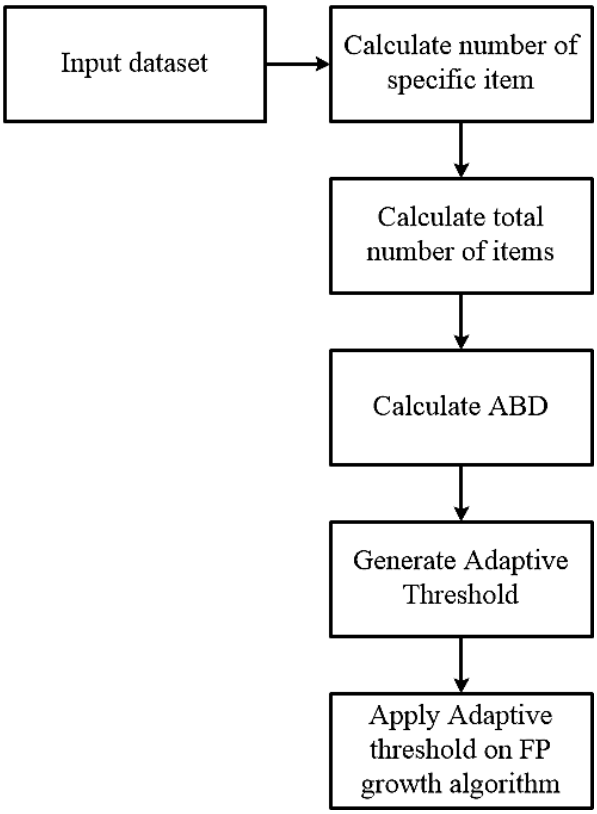

Figure 2 Flow chart of the FP growth algorithm

IV. PERFORMANCE ANALYSIS

The performance of the combined algorithm is evaluated using the Chess and Mushroom datasets from UCI repository [14]. The adaptive threshold value for the Chess dataset is 0.65 and Mushroom dataset is 0.75. Then, 0.65 and 0.75 are set as min_supfor the Chess and Mushroom dataset, respectively. If the min_sup<0.65 and 0.75, then a large number offrequent itemsets will be created and the execution time of the algorithm is increased. If the min_sup>0.65 and 0.75, the execution time is reduced significantly but some effective frequent itemsets are lost. The confidence level is fixed to be 70% better to create effective strong rules. The confidence rate needs to be fixed high for more accurate result.

Figure3 illustrates the comparative analysis of the execution time of the chess dataset. Figure 4 presents the comparative analysis of the execution time of the mushroom dataset. From the graphs, it is observed that the performance of the proposed combined algorithm is better than the FP-growth algorithm and Apriori algorithm. In Table I, the execution time analysis of the proposed combined algorithm, Apriori and FP growth algorithms for the Chess dataset is shown. Table II depicts the execution time analysis of the proposed combined algorithm, Apriori and FP-growth algorithm for the Mushroom dataset is presented.

Table I Execution time analysis of the proposed algorithm, Apriori and FP growth algorithm for Chess dataset

Support	Time (sec)		
	Apriori	Frequent Pattern Growth	Proposed Algorithm
0.6	165	95	70
0.65	72	42	32
0.7	35	27	24
0.75	25	19	15
0.8	20	16	8
0.85	9	7	2
0.9	4	2	0.5

Table II Execution time analysis of the proposed combined algorithm, Apriori and FP growth algorithm for Mushroom dataset

Support	Time (sec)		
	Apriori	Frequent Pattern Growth	Proposed Algorithm
0.6	139	82	65
0.65	50	38	30
0.7	28	22	15
0.75	20	17	9
0.8	13	9	5
0.85	4	2	1
0.9	2	1	0.5

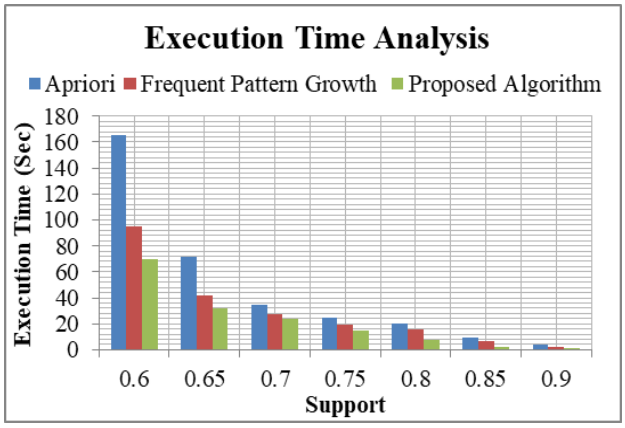

Figure 3 Execution time analysis of the proposed algorithm, Apriori and FPgrowth algorithm for Chess dataset

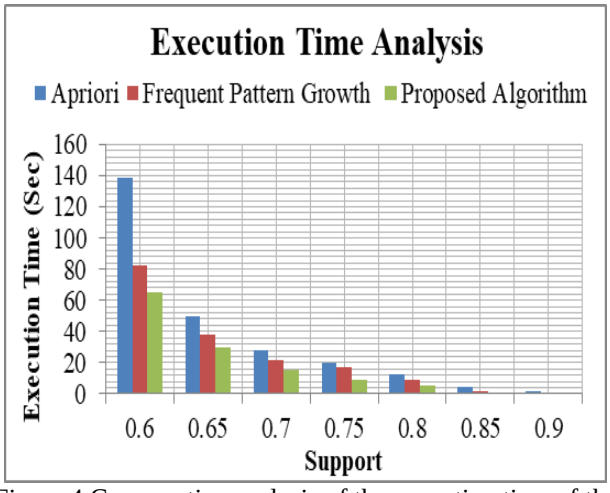

Figure 4 Comparative analysis of the execution time of the proposed algorithm, Apriori and FP growth algorithm for Mushroom dataset

V. CONCLUSION

The traditional ARM algorithms could not be able to perform the extraction of information from big data. Parallel ARM in the big data reduces the overall execution time required for the mining process. This work includes parallel Apriori algorithm that mines frequent itemsets using four sub-tasks in the parallel manner. The HP-Apriori algorithm performs partitioning of the data in a horizontal manner and applies two processes to each horizontal partition for computing the support count of the local candidate itemsets. Thus, the execution time is minimized significantly. The adaptive threshold is generated automatically and applied on the Apriori and FP growth algorithms for the effective mining of the frequent itemset. Hence, the difficulties in determining the minimum support threshold are reduced. This resulted in the effective reduction of the execution time. The future scope of the work is to apply different clustering techniques for mining the associations for a large database.

REFERENCES

[1] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *IEEE transactions on knowledge and data engineering*, vol. 26, pp. 97-107, 2013.
[2] Y. Chen, F. Li, and J. Fan, "Mining association rules in big data with NGEP," *Cluster Computing*, vol. 18, pp. 577-585, 2015.
[3] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," in *Acm sigmod record*, 1993, pp. 207-216.
[4] Y. Le Bras, P. Lenca, and S. Lallich, "Optimonotone measures for optimal rule discovery," *Computational Intelligence*, vol. 28, pp. 475-504, 2012.
[5] M.-H. Nadimi-Shahraki, N. Mustapha, M. N. B. Sulaiman, and A. B. Mamat, "Efficient Candidacy Reduction For Frequent Pattern Mining," *International Journal of Computer Science and Information Security*, vol. 6, 2009.
[6] A. Savasere, E. R. Omiecinski, and S. B. Navathe, "An efficient algorithm for mining association rules in large databases," Georgia Institute of Technology1995.
[7] J. Park, M. Chen, and P. Yu, "Efficient parallel data mining for association rules [C] International Conference on Information and Knowledge Management," ed: ACM, 1995.
[8] R. Agrawal and J. C. Shafer, "Parallel mining of association rules," *IEEE Transactions on Knowledge & Data Engineering*, pp. 962-969, 1996.
[9] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *ACM sigmod record*, 2000, pp. 1-12.
[10] A. Mueller, "Fast sequential and parallel algorithms for association rule mining: A comparison," 1998.
[11] L. Vu and G. Alaghband, "Novel parallel method for association rule mining on multi-core shared memory systems," *Parallel Computing*, vol. 40, pp. 768-785, 2014.
[12] M.-H. Nadimi-Shahraki and M. Mansouri, "Hp-Apriori: Horizontal parallel-apriori algorithm for frequent itemset mining from big data," in *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*(2017, pp. 286-290.
[13] M. M. Hasan and S. Z. Mishu, "An Adaptive Method for Mining Frequent Itemsets Based on Apriori And Frequent Pattern growth Algorithm," in *2018 International Conference on Computer, Communication, Chemical, Material and Electronic Engineering (IC4ME2)*, 2018, pp. 1-4.
[14] U. M. L. R. C. f. M. L. a. I. Systems. Available: https://archive.ics.uci.edu/ml/index.php

AUTHORS PROFILE

M.Sornalakshmi received her M. Sc ., M.Phil. Degree in ,Computer Science at Madurai Kamaraj University , Tamilnadu, India and now Doing Research from Kalasalingam Academy of Research and Education., Tamilnadu, India Her areas of interest are Data Mining , Algorithms, Big Data.

S. Balamurali is a Professor of Statistics and Director of Computer Applications at the Kalasalingam Academy of Research and Education. He received his undergraduate, postgraduate and doctoral degrees in Statistics from Bharathiar University, India. His research interests include applied statistics, data mining, network security and bioinformatics.

M. Venkatesulu received the postgraduate degree in Mathematics from Sri Venkateswara University, Tirupati, India, in 1975, and the Ph.D. degree in mathematics from Indian Institute of Technology, Kanpur, India, in 1979.

He worked as a faculty member at Shri Sathya Sai University, Prashanthinilayam, India between 1983 and 2003. He also worked as a consultant for Satyam Computers, Hyderabad, India, for a short period. He was a Visiting Professor at University of Missouri, Kansas City, between August 2006 and May 2007. Currently he is working as a Senior Professor and Head of the Department of Information Technology at Kalasalingam University, Krishnankovil, Srivilliputtur, Tamil Nadu, and India. His area of interest includes differential equation, Image Processing, Cryptography, Bioinformatics, Big Data Analytics and Distributed Computing