Efficient frequent pattern mining algorithm based on node sets in cloud computing environment

Efficient frequent pattern mining algorithm based on node sets in cloud computing environment

Vinay Kumar Billa V N [1], Lakshmanna K[2], Rajesh K[2], Praveen Kumar Reddy M[2], Nagaraja G[2] and Sudheer K[2]

[1]Deputy manager, Business intelligent unit, Axis bank, Bangalore, India
[2]School of Information Technology and Engineering, VIT University, Vellore-632014, Tamil Nadu, India.

E-mail: lakshman.kuruva@vit.ac.in

AbstractThe ultimate goal of Data Mining is to determine the hidden information which is useful in making decisions using the large databases collected by an organization. This Data Mining involves many tasks that are to be performed during the process. Mining frequent itemsets is the one of the most important tasks in case of transactional databases. These transactional databases contain the data in very large scale where the mining of these databases involves the consumption of physical memory and time in proportion to the size of the database. A frequent pattern mining algorithm is said to be efficient only if it consumes less memory and time to mine the frequent itemsets from the given large database. Having these points in mind in this thesis we proposed a system which mines frequent itemsets in an optimized way in terms of memory and time by using cloud computing as an important factor to make the process parallel and the application is provided as a service. A complete framework which uses a proven efficient algorithm called FIN algorithm. FIN algorithm works on Nodesets and POC (pre-order coding) tree. In order to evaluate the performance of the system we conduct the experiments to compare the efficiency of the same algorithm applied in a standalone manner and in cloud computing environment on a real time data set which is traffic accidents data set. The results show that the memory consumption and execution time taken for the process in the proposed system is much lesser than those of standalone system.

1. Introduction

Data Mining includes lots of tasks and techniques which are essential for critical decision making out of large amount of information. To get the final result of the process of data mining one should go through a collection of these tasks and techniques. One of these techniques is Frequent Pattern Mining. As its name says this technique finds the most frequent itemsets that are present in a database. The databases on which Frequent Pattern Mining technique can be used are called transactional databases. These databases contain transactions in millions and each of these transactions contains a different combination of items. These items can be anything depends upon the context of the transaction. The main theme of Frequent Pattern Mining is to discover the hidden patterns in the given transactional databases. The results produced are not the ultimate result of the data mining process. These results can be the input of another task to get the desired output. Having a very huge set of applications Frequent Pattern Mining stays at the first in the list of techniques used to find the frequent items. The results that are given by the process are useful to form Association rules in order to make decisions in fields like Customer transaction analysis, web mining, Software bug analysis, Chemical and Biological Analysis etc.

The databases that we deal with are very large in this case and it is obvious when we mine frequent patterns from the database in a standalone method it consumes lot of memory and execution time. It makes the process slow and costly. In order to make the process fast and minimize the high memory usage parallel processing is introduced in which the process runs in more than one computer and the resources needed for the process are shared among the nodes that are participating in that parallel processing. There are some algorithms based on parallel processing methodology like PFP-growth algorithm and others were proposed. These algorithms brought a huge difference in efficiency when compared to standalone algorithms.

In 2014 Zhi-Hong Deng and Sheng-Long Lv proposed an algorithm that uses a tree data structure called POC tree and mine frequent itemsets using a different concept called Nodesets. This algorithm makes a huge margin in efficiency in terms of memory and execution time when compared to its previously proposed algorithms. This algorithm is named as FIN algorithm. FIN algorithm is an extended and improved version of the algorithm which uses the Node-lists and N-lists data structures. On the other hand FIN algorithm uses a novel data structure called Nodeset for mining the frequent itemsets. FIN algorithm directly discovers frequent itemsets in a search tree called set- enumeration tree. In this thesis FIN algorithm is used to discover frequent itemsets but in a different way to make the algorithm even better and efficient.

From the day when the frequent pattern mining came to the picture there are so many methods that are proposed related to the algorithms to mine frequent patterns, parallel and distributed mining process and the privacy preservation of the data which undergo the process of frequent pattern mining. But there is no specific architecture and framework to achieve the efficient way of mining patterns. The main idea of the thesis is to provide an entire framework which includes an efficient Frequent Pattern Mining algorithm. This framework allows user to execute the algorithm on a dataset in parallel processing paradigm. In this case we are using cloud computing environment which contains number of nodes each of which is responsible for the mining process. This proposed system provides an application as a service. By providing this service in a cloud computing environment the memory and execution time required to complete the entire process is divided among the number of hosts that are present and available in the cloud and the resultant memory consumption and execution time is very much reduced.

To evaluate the performance of the proposed system we need to do experiment on any dataset to get some statistics based on which the efficiency of the system is measured. Here in this case memory usage and execution time are taken as metrics of evaluation. The dataset used in this experiment is a Traffic Accidents data set collected by Karoline Guerts based on road accidents that are occurred in Belgium. This data set contains a huge set of items that can undergo the process of frequent itemsets discovery. We applied the algorithm in a standalone method which involves a single system containing the algorithm works on this data and a parallelized system contains number of hosts present in a cloud computing environment. These results are compared in order to deterrmine the efficiency of the system. The graphical notations of the results are presented in the following chapers.

2. Existing Methods

Most of the frequent itemset mining algorithms can be clustered into some groups such as join-based, TreeProjetion, DepthProject and FP-growth etc. Apriori algorithm was a basic algorithm which works on join-based method which is proposed by Agarwal and Srikanth 1994. The Apriori algorithm uses a level wise approach in which every frequent itemsets of length (n+1) is generated only after the generation of itemsets of length n . The main theme on which the algorithm works is that every subset of a frequent pattern is also frequent. There are some optimizations are also proposed by Agarwal and Srikanth in 1994 and named them as AprioriTid and AproiriHybrid. These methods use the transaction

IDs and each transaction is replaced by a previously determined transaction or null transaction. Let the set of n candidates in Cn that are contained in transaction T be denoted by R(T , Cn). This set R(T , Cn) is temporarily included in the transaction database Tn . The DHP algorithm, also known as the Direct Hashing and Pruning method, was proposed by Jang in 1995. This algorithm needed two more optimizations to make it even better in terms of performance. In each iteration of the candidate itemsets if prunes the items and after pruning it trims the transaction to increase the efficiency of the process of support counting.

After join-based algorithms the era of frequent pattern mining is moved to TreeProjection algorithms which are based on set-enumeration concepts. The main reason of using recursive projections is to use the counting work at a given level of the tree again and again. There are many algorithms which use strategies like depth-first, breadth-first and combined strategy of above stated two strategies.

Early times of twenty first century the mining frequent itemsets used the Vertical Mining Algorithms which projects the problem database vertically as an inverted list. The reason behind this is this structure enables the most efficient way of counting. In the list of these algorithms we can find Eclat algorithm proposed by Mohammed J. Zaki in 2000 at the top. Eclat uses the strategy named breadth-first search on lattice partitions, after partitioning the candidate set into disjoint groups. This algorithm is considered as a breadth-first algorithm. Other versions and depth-first search strategies of Eclat, after proper experiments are presented with results, are embeded in later work such as dEclat proposed by Mohammed J. Zaki and Karam Gouda in 2003. The dEclat work is also contains some additional optimizations such as diffsets to improve counting performance of the algorithm.

FP-growth algorithm proposed by Jia Wei Han in 2004, is a combination of compressed representtation of tree projection of database for more effective counting of support and suffix based pattern discovery. In this method the database is projected as a FP (Frequent Pattern)-tree then the frequent patterns are mined using the FP-tree. This is the most efficient method when compared with any of the previously discussed methods, because the memory resources and the time taken to execute the mining process are very less. This requires only two times scanning of the database where the Apriori-like algorithms need to scan the database number of times which is equal to the length of the longest transaction. After FP-growth algorithm H-mine algorithm was proposed by Jian Pei and Jiewei Han in 2007. It makes use of a efficient memory hyper-structure called H-Struct. The fundamental strategy on which the algorithm works is mine the database by partition it and the mining process goes on for each partition in the memory. Finally, the combined result of each partition is taken as the final result in discovering frequent pattern. H-Mine algorithm contains a module that can identify whether the database is sparse or dense, and it allows the user to select different data structures dynamically based on its identification.

A novel data structure Node-list came into the picture to make the mining process even efficient. This is proposed by Deng and Wang in 2010. The data in this algorithm was represented with the help of a tree structure called PPC-tree. In its subsequent year Deng in 2012 one more data structure which is also based on the PPC-tree, called N-list. These two data structures are used to store the information about the frequent itemsets that are discovered so far during the process. Usually, the nodes in a N-list or Node-list are sorted by the ascending order of the pre-order of nodes. Their main difference is that Node-lists are built by descendant nodes while N-lists are built by ancestor nodes. These two algorithms based on Node-list and N-list have been proven efficient but the problem with these structures is that they need to encode each node of a PPC-tree with pre-order and post-order code which takes much memory and time for execution. To optimize this Zhi-Hong Deng and Sheng-Long Lv in 2014, proposed an algorithm which uses the Nodesets instead of using Node-list and N-lists. This algorithm is called FIN algorithm. This optimizes the memory usage by half of the memory used in above algorithms because it uses only the pre-order (or post-order) of each node.

3. Proposed System

A clear and sophisticated architecture is proposed in this paper, which allows the user who wants to discover frequent itemsets by uploading the transactional dataset file as an input through a web application front end. This dataset undergoes some pre-execution events like gathering cloud nodes information and splitting of datasets into sub-datasets etc, to make it compatible to apply the desired algorithm at each of the nodes. At this stage the divided data is transferred to each node for further execution. After proper execution of algorithm on the data it produces an intermediate result at every single server. These intermediate results will later be processed to get final result. This final result is presented to the user in a very understandable way to make use of the process more effectively.
In this research a complete framework is proposed based on cloud computing environment which supports the most efficient algorithm in mining of frequent pattern mining and parallelizing the algorithm. This framework is embedded in a web application to give the user who wants to mine patterns from large databases, an easy way to access and mine patterns. The proposed system will have the following architecture as shown in figure 1.

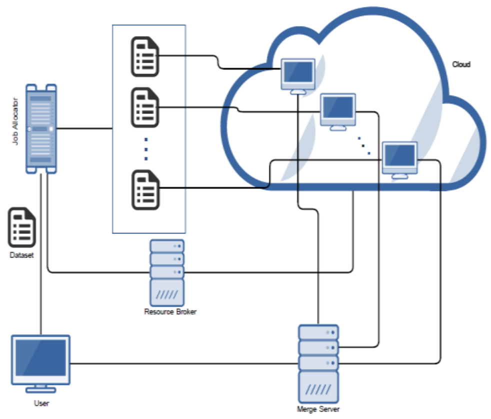

Figure 1. Proposed architecture

This proposed system architecture contains many components which are specialized in their own tasks to make the entire system powerful. Each of these components has its own responsibilities and cooperate and communicate with other components by passing some data necessary information across various components of the Architecture. Components that are present in the architecture can be named as

User terminal
Resource Broker (RB)
Job Allocator (JA)
Cloud nodes
Merge Server (MS)

User terminal is one of the five components of the system which enables the interaction between user and the system. User terminal provides a rich user interface for the user to interact with the entire process of mining the frequent patterns. Being a web application this user terminal can be any personal computer, server, mobile phone, tablet which works on any of the existing operating systems such as windows, android, IOS and so on. User can open this application from any of the browsers from any of the above said devices and upload the dataset that he/she wants to mine frequent patterns. This application is designed in such a way that the users need not to go through the algorithm and there is no need of having any knowledge about the high level components of the process. The minimum support value which plays an important role while making decisions can also be given by the user through this terminal. This user interface is developed using HTML, CSS, JavaScript and JSP. These technologies make the interface attractive and easy to use.

To maintain all the information about each and every node in the cloud we have a Resource Broker (RB) which supplies the relevant information about the resources that are available for the use of mining process. According to the number of resources that are active and can be used by the user, the dataset will be divided to send to each of the node existed in the cloud. This information about cloud nodes will be used by the Job Allocator. Whenever a node in the cloud is not available then the Resource Broker will update its information and project that node as unavailable. On the other hand if a new node is added to the cloud then it includes its information and RB will make that node available for the process.

Job Allocator (JA) plays an important role in allocating the work to the nodes present in the cloud. It gets the information about available resources in the cloud, from resource broker and divides the dataset into number of sub-datasets depends upon the number of available resources. The number of divided databases depends upon the number of active cloud nodes that are present in the cloud to get this task done. This uses a specific algorithm to divide the dataset into parts. An algorithm named Equal Working Set (EWS) algorithm is used for dividing the dataset. EWS algorithm divides the dataset in equal size with all sub-datasets containing the equal number of transactions. It ensures that the workload provided to all the nodes in the cloud is equal and the resources required for the process like memory and execution time will be same for all the nodes.

Cloud Nodes are the virtual devices where actually the process is executed. These nodes contain the algorithm to discover frequent patterns in the dataset. Here FIN algorithm is implemented and deployed in each of the nodes in cloud. Whenever a cloud node gets the sub-dataset from Job Allocator then this data undergoes the mining process by applying the algorithm implemented in that node. After successful completion of the process intermediate result is produced. This intermediate result is forwarded to Merge Server for further processing to get final result.

A Merge Server (MS) merges all the intermediate results that are produced by the execution of the tasks in each of the cloud nodes. Merge Server produces the final result by gathering the intermediate results and passes it to the user terminal for presentation and further analysis of the result. The final result is displayed in such a way that the user can easily understand and analyze the results. This makes the process of decision making simple and flexible.

3.1. Methodology

A simple and user friendly interface is needed for any system to utilize its services by the users. So here an attractive and easy to use web application is designed for the user to upload his dataset to be mined, to interact with the process of mining and to get the final results in very understandable manner. This page gives the user the ability to upload his dataset.

Resource Broker takes the responsibility of getting and providing the resources information and their availability from cloud and to the user. This Resource Broker when gets the user's requirement of resources, divides the dataset into sub-datasets and distributes those datasets to the available nodes for processing.

The implemented FIN algorithm is deployed in every node that is present in the cloud so that those nodes behave like hosts for the process of pattern mining. In this study we simulate the cloud part in java so that the cloud contains some nodes. These nodes contains the code for the above implemented algorithm. The Resource Broker after dividing the dataset into sub-datasets it distrubutes the data to the nodes. The distribution process of datasets is shown for the knowledge of which node contains which part of the dataset, in the following page.

The processed data in each node produces some intermediate result and these intermediate results are collected from each node of the cloud and merge them into a final result to provide the user the clear idea on the results that are obtained. This final result is presented to the customer in such a way that an average user with a little knowledge about Data Mining and Pattern Mining can understand and use them in an appropriate way to make decisions.

On the other hand in order to evaluate the performance of the proposed system there is a necessity of maintaining the statistics about the entire process of mining. In this statistics the performance of each host is determined by calculating the time taken to run the algorithm and memory used at each of the host. These statistics will later be compared with stand alone model of the proposed algorithm and then the efficiency of the proposed system will be known.

4. Results and Discussions

The algorithm used in this thesis is FIN algorithm. This algorithm was proposed in 2014 and it produced some optimized results when compared to previously proposed algorithms. At the time of proposal this algorithm worked in standalone method. To maximize the benefits and minimize the memory consumption and execution time a new way of execution of this algorithm is proposed in this paper. This system proposes the parallel execution of the above said algorithm in a cloud computing environment. To evaluate the proposed system we need to compare the results of the current system with standalone system. We can say that the current system is efficient if we get more optimized results.

For evaluation of the system we use the same dataset for both the systems. Here we are using Traffic Accidents dataset created by Karoline Guerts based on the accidents that occur during the period 1991-2000 in Belgium. First we apply the algorithm on above stated dataset and observe the results. The following table shows the statistics of the mining process in standalone system. The table contains some credentials which are useful to measure the performance like itemsets obtained, time of execution and memory needed for different values of minimum support.

Then the same experiment is done using the same data set on the proposed parallelized system for frequent pattern mining and came up with some results. But here we have more than one host in which the algorithm is executed. So we need to consider each and every host's statistics in order to measure the efficiency. The same metrics that are considered in above experiment are noted in the present case also. We final get a table as follows which shows individual statistics of each node in the cloud.

The comparison on both the systems gives us a graph presented below. In this graph where the line belongs to standalone system having high values for execution time, is pretty much far from the cluster of lines show the statistics of the hosts participated in proposed parallelized system. These lines have almost equal values of execution time for corresponding minimum support values.

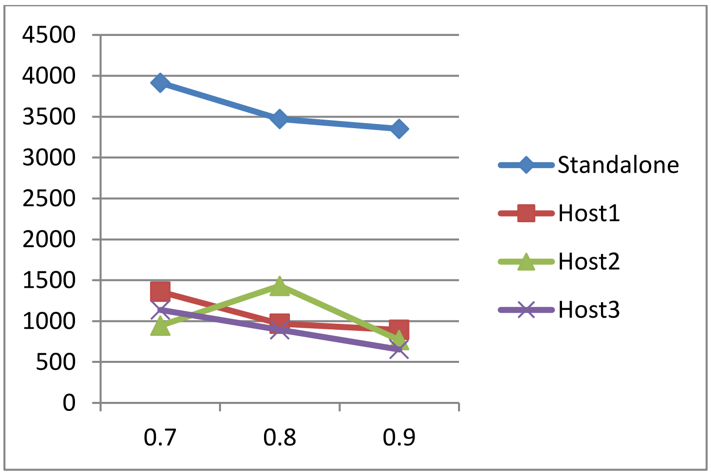

Figure 2. Graph comparing the execution times of both systems

As a part of evaluation of the system there is one more factor which plays an important role. It shows the memory used in standalone process is much more than that of each host in the parallelized system.

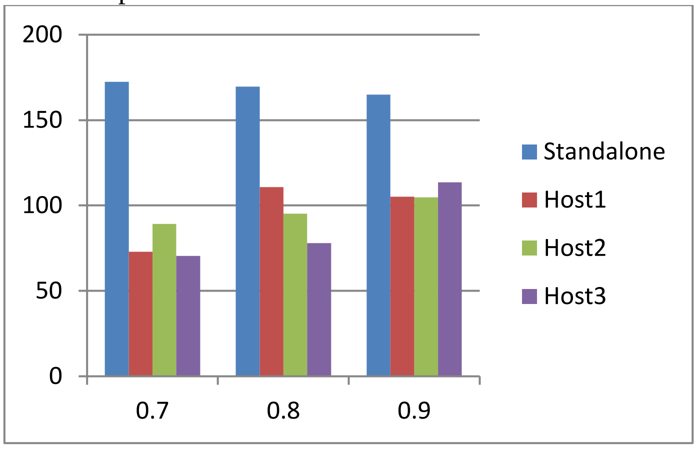

Figure 3. Bar diagram comparing the memory usage of both the systems

5. Conclusion

The experimental results show that the system which is proposed is much more efficient than the existed standalone system in terms of execution time and the memory utilization for the execution of the algorithm. Hence the proposed system manages to reduce the resources that are required and makes the process fast. By using the system in cloud computing environment we can say that the overall workload will be equally distributed and individual workload of a host in the cloud is reduced which makes the process light and faster. The entire process of parallelizing an algorithm says that any efficient algorithm can be made more efficient when it is given proper resources by implementing it in a current powerful yet recent technology called cloud. This cloud provides the required resources with high availability and reliability. By using this powerful technology this idea of parallelizing is boosted and made real with less effort and high accuracy.

Future Work: As we discussed so far this paper is mainly focused on making an algorithm more efficient. But when we consider the problem in a commercial manner we can make a tool which allows the user to get the task of discovering frequent itemsets without much knowledge on the field of data mining. Based on the utilization of resources such as memory of cloud nodes and processing time the user can be charged according to the cost of the resources.

Let us consider a scenario where a medium scale computerized business which contains a large amount of transactional data. When the company is in the position that it could not afford a special team of analysts to make use of the data of the company to make decisions to develop the business, then the user can make use of the commercial tool proposed here by logging in into the web application and uploads his database file and carries the process of mining out. At last after completion of the process the user is charged for utilization of the resources. These results produces in this process are stored in the database and can be used to analyze at any point of time. This idea can make the system useful and effective in commercial aspects also.

References

[1] Deng Z H and Lv S L 2014 Fast mining frequent itemsets using Nodesets *Expert Systems with Applications* **41** 4505–4512

[2] Lin K W and Lo Y C 2013 Efficient algorithms for frequent pattern mining in many-task computing environment *Knowledge-based Systems* **49** 10-21
[3] Wikipedia website related to associative rule learning https://en.wikipedia.org/wiki/Association_rule_learning
[4] Agrawal R and Srikant R 1998 Fast algorithm for mining association rules In VLDB **94** 487–499
[5] QiuY, LanY J and XieQ S 2004 An improved algorithm of mining from FP-tree *Proceedings of the Third International Conference on Machine Learning and Cybernetics* 26-29
[6] Aggerwal C C and Han J 2014 A book on Frequent Pattern Mining *Springer International Publishing Switzerland*
[7] Agrawal R and Srikant R 1994 Fast Algorithms for Mining Association Rules in Large Databases VLDB Conference 487–499
[8] https://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Frequent_Pattern_Mining/The_FP-Growth_Algorithm, Data Mining Algorithms In R/Frequent Pattern Mining/The FP-Growth Algorithm
[9] Deng Z Hand Wang Z 2010 A New Fast Vertical Method for Mining Frequent Patterns *International Journal of Computational Intelligence Systems* **3**(6) 733-744
[10] Deng Z H, Wang Z and J Jiang 2012 A New Algorithm for Fast Mining Frequent Itemsets Using N-Lists *Science China Information Sciences* **55**(9) 2008 - 2030
[11] Han J, Pei J and Yin Y 2000 Mining frequent itemsets without candidate generation In SIGMOD'00 1–12
[12] Grahne Gand Zhu J 2005 Fast algorithms for frequent itemset mining using P-trees *IEEE TKDE Journal* **17**(10) 1347–1362
[13] Agarwal R, Aggarwal C C and Prasad V V V 1999 ATree Projection Algorithm for Generation of Frequent Item sets *Journal of Parallel and Distributed Computing* **61**(3) 350–371
[14] Agrawal R and Srikant R 1994 Fast Algorithms for Mining Association Rules in Large Databases *VLDB Conference* 487–499
[15] HanJ, PeiJ and YinY 2000 Mining Frequent Patterns without Candidate Generation *ACM SIGMOD Conference*
[16] LiuG, Lu HandYuJ X 2003 AFOPT:An Efficient Implementation of Pattern Growth Approach FIMI Workshop
[17] Azkural E and Aykanat C 2004 A Space Optimization for FP-Growth FIMI workshop
[18] Pei J, Han J, Lu H, Nishio S, Tang S and Yang D 2001 H-mine: Hyper-structure mining of frequent patterns in large databases ICDM Conference
[19] Lakshmanna K and Khare N 2016 Constraint-Based Measures for DNA Sequence Mining using Group Search Optimization Algorithm *International Journal of Intelligent Engineering & systems* 9.3 91-100
[20] Lakshmanna K, Rajesh K, Thippa Reddy G, Nagaraja G and Subramanian D V 2016 An Enhanced Algorithm For Frequent Pattern Mining From Biological Sequences *International Journal Of Pharmacy & Technology* 8.2 12776-12784
[21] Lakshmanna K and Khare N 2016 FDSMO: Frequent DNA Sequence Mining Using FBSB and Optimization *International Journal of Intelligent Engineering & systems* 9.4 157-166
[22] Lakshmanna K and Khare N 2016 Mining DNA Sequence Patterns with Constraints Using Hybridization of Fireflyand Group Search Optimization *Journal of Intelligent systems* DOI 10.1515/jisys-2016-0111