APPLYING PARALLEL ASSOCIATION RULE MINING TO HETEROGENEOUS ENVIRONMENT

P.Asha[1]
[1]Research Scholar,Computer Science and Engineering Department, Sathyabama University,
Chennai,Tamilnadu,India.
ashapandian225@gmail.com

Dr.T.Jebarajan[2]
[2]Principal, Kings College of Engineering,
Chennai, Tamilnadu,India.
drtjebarajan@gmail.com

Abstract

The work aims to discover frequent patterns by generating the candidates and frame the association rules after which filter out only the efficient rules based on various Rule Interestingness measures. As all these require heavy computation, application of complete parallelization to every individual phase would yield better performance. The paper illustrates the system behavior in a heterogeneous environment with both shared memory and distributed memory parallelization while efficiently mining the data.

Keywords: Sequential, Parallel, Data Mining, Grid Computing, Interestingness Measures.

1. Introduction

Association rule mining, the major technique of data mining, involves finding frequent itemsets with minimum support and generating association rules with maximum confidence. There exist various algorithms to find the frequent pattern and their association rules.

Hilage and Kulkarni have provided a good review on various data mining techniques such as association rules, rule induction technique, apriori algorithm, decision tree and neural network. It focuses on how data mining techniques are used for different application areas for finding out meaningful pattern from the database.[7]

Rakesh Agrawal, Tomasz Imielinski and Arun Swami, have presented an efficient algorithm that generates all significant association rules between items in the database. The algorithm incorporates buffer management and novel estimation and pruning techniques. The paper also present results of applying this algorithm to sales data obtained from a large retailing company, which shows the effectiveness of the algorithm.[2]

Zaki proposed a framework based on finding frequent itemsets as well as reduce redundancy caused by traditional algorithms without loss of information.[10]

The Prutax algorithm, proposed by Hipp .J, Myka .A, Wirth .R and Guntzer combines several frequent itemsets mining optimizations, as a way to discover generalized frequent itemsets faster. It was still locked in the traditional framework of "finding frequent itemsets first". However, it did not take into consideration rules that could learn in depth in hierarchies, and further redundancy issues related to such rules.[6]

The task of finding all frequent item sets for large datasets requires lot of computation which can be minimized by exploiting parallelism to the sequential algorithms.[4] However, applying parallelism technique that suits systems of different configuration and functionality will be a challenging task. The proposed model aims at the process parallelization while mining and extracting frequent patterns in a heterogeneous environment. Detailed analysis of various ARM algorithms and comparative study has been made with respect to sequential and parallel mining and the performance evaluation implies that the parallelized Apriori serves best.

2. Association Rule Mining

2.1 Parallelized Apriori Algorithm

2.1.1 Apriori algorithm

* Given L_{k-1}, the set of all frequent (k-1)-itemsets, generate a superset of all the set k-itemsets.

* If an itemset X has minimum support, so do all subsets of X.

```
* Candidate generation:
    Join $L_{k-1}$ with $L_{k-1}$:
    Insert into $C_k$;
    Select p.item1, p.item2…., p.itemk-1, q.itemk-1
    From $L_{k-1}$ p, $L_{k-1}$ q;
    Where p.item1=q.item1, p.item2=q.item2,
    .....p.itemk-2 =q.itemk-2, p.itemk-1<q.itemk-1;
* In the prune step, delete all itemsets c € $C_k$,Where (k-1) subset of c is not in $L_{k-1}$. [1]
```

2.2.2 Parallel Algorithm

This algorithm uses a simple principle of allowing "redundant computations in parallel". The first pass is special. For all pass k>1, the algorithm works as follows:

* Each processor P_i generates the complete C_k, using the complete frequent itemset L_{k-1} created at the end of pass k-1.

* Processor P_i makes a pass over its data partition D_i and develops local support counts for candidates in C_k.

* Processor P_i exchanges local C_k counts with all other processors to develop global C_k counts. Processors are forced to synchronize in this step.

* Each processors P_i now computes L_k from C_k.

* Each processor P_i independently makes the decision to terminate.

2.2 Measures of Rule Interestingness

In the ARM, the emphasis is on the quality of each individual rule. [3, 5]

Let, N_{left} be instances matching left
N_{right} be instances matching right
N_{both} be instances matching both left and right
N_{total} be the total number of instances

Basically used measures are:

- Confidence
- Support
- Completeness

Additional Rule Interestingness Measures:

- Lift and Leverage

  These are used to reduce the number of rules to a manageable size or rank rules in order of importance.

a. Confidence

It is given by, N_{both} / $N_{left.}$ That is,the proportion of right hand sides predicted by the rule that are correctly predicted.

b. Support

It is given by, N_{both} / $N_{total.}$ That is, the proportion of training set correctly predicted by the rule.

c. Completeness

It is given by, N_{both} / $N_{right.}$ That is, the proportion of matching right hand sides that are correctly predicted by the rule.

d. Lift

Lift of a rule, L -> R measures how many more times the items in L and R occur together in transactions than would be expected if the item sets L and R were statistically independent. Support(R) is the proportion of transactions matched by R.

Lift values greater than 1 implies interesting. They indicate that transactions containing L tend to contain R more often than transactions that do not contain L.

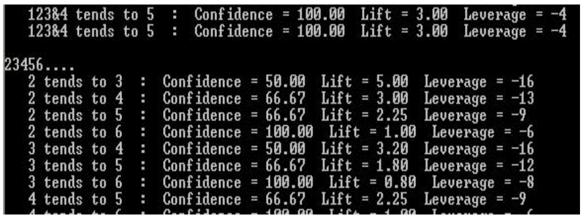
Lift (L-> R) = count (L U R) / count(L) * support(R)

e. Leverage

Leverage of a rule, L -> R measures the difference between the support for LUR(items in L and R occurring together in the database) and the support that would be expected if L and R were independent.

Leverage (L-> R) = support (L U R) - support(L) * support(R)

The value of leverage of a rule is always less that its support. The number of rules satisfying the support>=minsup and confidence>= minconf constraints can be reduced by setting a leverage>=0.0001 corresponding to an improvement in support of one occurrence per 10000 transactions in a database.[9]



```
   123&4 tends to 5  :  Confidence = 100.00  Lift = 3.00  Leverage = -4
   123&4 tends to 5  :  Confidence = 100.00  Lift = 3.00  Leverage = -4

23456....
   2 tends to 3  :  Confidence = 50.00  Lift = 5.00  Leverage = -16
   2 tends to 4  :  Confidence = 66.67  Lift = 3.00  Leverage = -13
   2 tends to 5  :  Confidence = 66.67  Lift = 2.25  Leverage = -9
   2 tends to 6  :  Confidence = 100.00  Lift = 1.00  Leverage = -6
   3 tends to 4  :  Confidence = 50.00  Lift = 3.20  Leverage = -16
   3 tends to 5  :  Confidence = 66.67  Lift = 1.80  Leverage = -12
   3 tends to 6  :  Confidence = 100.00  Lift = 0.80  Leverage = -8
   4 tends to 5  :  Confidence = 66.67  Lift = 2.25  Leverage = -9
```

Fig. 1. Association Rules

3. Proposed System Architecture

The system architecture includes the input database as the transactional dataset, distributed environment and the entire set of module that can be processed under the distributed environment.

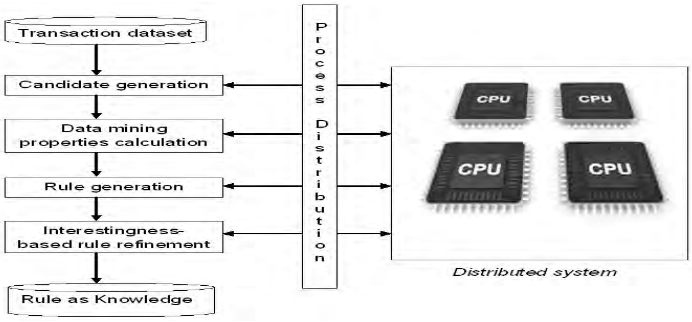


Fig. 2. Architectural diagram

The work flow diagram provides a detailed execution procedure of the proposed system. Every individual data mining step is parallelized to improve the performance.

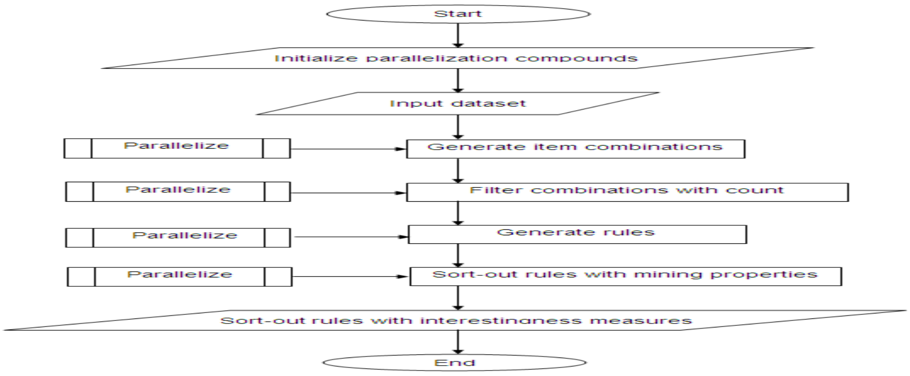

Start
Initialize parallelization compounds
Input dataset
Parallelize
Generate item combinations
Parallelize
Filter combinations with count
Parallelize
Generate rules
Parallelize
Sort-out rules with mining properties
Sort-out rules with interestingness measures
End

Fig. 3. Working procedure of proposed system

Even though lots of parallel ARM algorithms have been developed so far, most of the algorithms are designed for homogeneous system with static load balancing which is far from reality. Algorithm for heterogeneous system with dynamic load balancing is required to develop with high performance, a model of which is implemented in the proposed system. Also certain rule interestingness measures were used to address the problem of rule redundancy.

The rules generated can grow unwieldy with the increase in transaction if support and confidence thresholds are small and of which many may be redundant. So, there is a great need of the algorithm with the constraints discussed above. With all these functionalities to be carried out, the system that processes the functions should be highly configured and possibly distributive. The proposed system exploits the ARM procedure under distributed environment and a comparative study has been made that witnessed valuable efficiency difference of distributed and parallel ARM over sequential ARM.

4. Performance Evaluation

4.1 Serial Execution Vs Parallel Execution

The performance improves in a better way when executed parallel and would be more effective with the increase in number of cores.

Table 1. Execution time of serial and parallel ARM algorithm

Threshold	Serial Execution(sec)	Parallel Execution(sec)
.5	0.097	0.085
.10	0.080	0.060
.15	0.065	0.045
.20	0.040	0.025
.25	0.020	0.009

Real dataset from Yahoo Financial Data Set Repository is chosen for this performance study. The dataset is available in biz.yahoo.com\p. The corresponding result is shown in the above table.

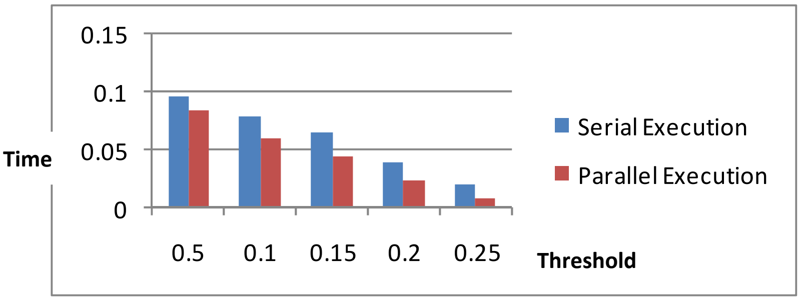


Fig. 4. Graph representing execution time in serial and parallel algorithm

The above graph concludes that the process time decreases as the range of threshold increases. However, irrespective to the threshold, the time difference between sequential and parallel execution varies within a constant range.

4.2 Scalability Evaluation

- Serial run-time is indicated using T_s
- Parallel run-time using T_p

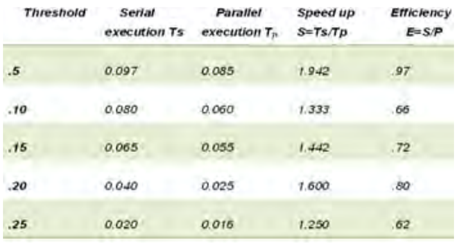
4.2.1 Speed up

Speed-up, S, is defined as the ratio of the serial run-time of the best sequential algorithm for solving a problem to the time taken by the parallel algorithm to solve the same problem on p processors. $S=T_s/T_p$.[8]

4.2.2 Efficiency

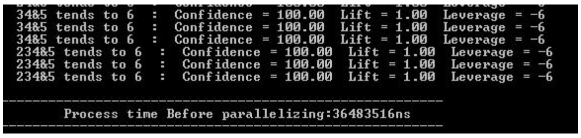
Efficiency E, is a measure of the fraction of time for which a processor usefully employed. It is defined as the ratio of speed-up to the number of processors. E=S/P, Where P is the no of processors involved. [8]

Table 2. Efficiency comparison in serial and parallel ARM algorithm



Threshold	Serial execution Ts	Parallel execution T_p	Speed up S=Ts/Tp	Efficiency E=S/P
.5	0.097	0.085	1.942	.97
.10	0.080	0.060	1.333	.66
.15	0.065	0.055	1.442	.72
.20	0.040	0.025	1.600	.80
.25	0.020	0.016	1.250	.62

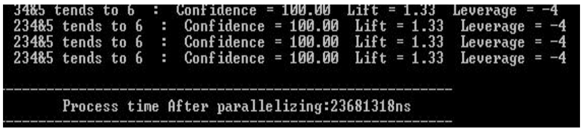
The performance improves in a better way when executed parallel and would be more effective with the increase in number of cores.



```
 34&5 tends to 6  :  Confidence = 100.00  Lift = 1.00  Leverage = -6
 34&5 tends to 6  :  Confidence = 100.00  Lift = 1.00  Leverage = -6
 34&5 tends to 6  :  Confidence = 100.00  Lift = 1.00  Leverage = -6
 234&5 tends to 6  :  Confidence = 100.00  Lift = 1.00  Leverage = -6
 234&5 tends to 6  :  Confidence = 100.00  Lift = 1.00  Leverage = -6
 234&5 tends to 6  :  Confidence = 100.00  Lift = 1.00  Leverage = -6

-------------------------------------------------------------
       Process time Before parallelizing:36483516ns
-------------------------------------------------------------
```

Fig. 3. Execution time -- Before parallelization



```
 34&5 tends to 6  :  Confidence = 100.00  Lift = 1.33  Leverage = -4
 234&5 tends to 6  :  Confidence = 100.00  Lift = 1.33  Leverage = -4
 234&5 tends to 6  :  Confidence = 100.00  Lift = 1.33  Leverage = -4
 234&5 tends to 6  :  Confidence = 100.00  Lift = 1.33  Leverage = -4

-------------------------------------------------------------
       Process time After parallelizing:23681318ns
-------------------------------------------------------------
```

Fig. 4. Execution time -- After parallelization

The Run time difference, before and after Parallelization is 12802198ns. Hence the Parallel Association rule mining shows a better performance than the sequential one.

5. Conclusion

From the above discussion, it is clear that the Apriori is the simplest sequential ARM algorithm developed with many drawbacks and to overcome that various parallel algorithms were developed. There were challenges associated with these parallel algorithms too; like how efficiently memory can be utilized, how communication and synchronization can be minimized in different i.e. Shared Memory system and Distributed Memory system architectures, how work load can be balanced among various processes, and about, how to achieve efficient task decomposition, data layout, data decompositions etc. To exploit parallelism one should focus on these major challenges.

As of now, the Parallel Association Rule mining process is implemented and tested under heterogeneous environment. In order to increase the performance further, the entire module should be processed under much wider distributed environment of highly configured resources.

Along with process distribution implemented to our module, including data distribution would increase the efficiency thereby providing an effective mining system.

References

[1] Zaki .M, "Generating Non-redundant Association Rules," In 6th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, August 2009, pp. 233-251.

[2] Rakesh Agrawal, Tomasz Imielinski and Arun Swami, "Mining association rules between sets of items in large databases," IEEE Trans. Data mining, vol. 22, No.10, June 2010, pp. 2-5.

[3] P.Asha, Dr.T.Jebarajan, "Analyzing the Sequential And Parallel ARM Algorithms And Its Impact In Grid Computing Environments" in the International Journal of Advanced Computing, Recent Science Publications, PP.1109 – 1114, Vol. 36, No. 1, ISSN: 20151-0845 ,January 2013. (Impact Factor : 2.31) - http://recentscience.org/ijca-international-journal-of-advanced-computing/

[4] Usama Fayyad, Gregory Piatetsky Shapiro,Smyth Padhraic, and Ramasamy Uthurasamy,"Advances in Knowledge Discovery and Data Mining," AAAI Press/ The MIT Press, pp. 74-87, 1996.

[5] P.Asha, Dr.T.Jebarajan, "Mining Interesting Association Rules With A Heterogeneous Environment" in the International Joint Conferences on CNC and CSEE 2013 ,Conference Proceedings published by Springer, ISSN: 1867-8211, PP. 222-228, Feb 22-23,2013.

[6] Hipp .J, Myka .A, Wirth .R and Guntzer .U, "A new algorithm for faster mining of generalized association rules". In Proc.2ndPKKD, 2007, pp. 2-6.

[7] Hilage and Kulkarni, " Review of literature on data mining," IJRRAS, pp. 46-52, Jan 1996.

[8] Text Book on "Parallel Computing - Theory and Practice", by Michael J. Quinn, Tata McGraw Hill Edition.

[9] Bing Liu,Wynne Hsu, Shu Chen, Yiming Ma, " Analyzing the subjective interestingness of association rules," Intelligent Systems and their Applications, IEEE , Vol. 15,No. 5,Sep 2000,pp. 47-55.

[10] Zaki, M. J., Parallel and distributed association mining: A survey. IEEE Concurrency, Special Issue on Parallel Mechanisms for Data Mining, 7(4):14--25, December 1999.