2009 International Conference on Computer Engineering and Applications
IPCSIT vol.2 (2011) © (2011) IACSIT Press, Singapore

OPT-DIC- An Efficient Algorithm for Distributed Association Rule Mining

Preeti Paranjape [1], Umesh Deshpande[2]
[1]Shri Ramdeobaba Kamla Nehru Engineering College(SRKNEC) Nagpur,
[2] Dept. of Electronics and Computer Science, Visvesvarayya National Institute of Technology (VNIT)
Nagpur, Maharashtra, India 440 010

Abstract. We present a distributed algorithm based on Dynamic Itemset Counting (DIC) for generation of frequent itemsets. The nature of DIC represents a paradigm shift from Apriori-based algorithms in the number of passes of the database hence reducing the total time taken to obtain the frequent itemsets. Our distributed algorithm, called the Optimistic Messaging DIC (OPT-DIC) gives much better results than Count Distribution (CD). We exploit the advantage of DIC - that of starting the counting of an itemset as early as possible. Hence, OPT-DIC shows remarkable improvement in the amount of time taken because of reduction in the number of passes of the database and comparatively lesser number of candidates generated.

Keywords: Association Rule Mining, Distributed ARM, Dynamic Itemset Counting, Optimistic Messaging DIC

1. Introduction

ARM has been used extensively for the classical problem of market basket analysis where it is required to find the buying habits of customers. Determining what products customers are likely to buy together can be very useful for planning and marketing. Association rules are used to show the relationships between these data items. Many centralized algorithms exist for Association Rule Mining(ARM). Most of the algorithms depend on the discovery of frequent itemsets for generation of association rules. Since the total number of itemsets is exponential in terms of the number of items, it is not possible to count the frequencies of these sets by reading the database in just one pass. Different algorithms for the discovery of association rules aim at reducing the number of passes by generating candidate sets, which are likely to be frequent itemsets. They attempt to eliminate infrequent sets as early as possible. Dynamic Itemset Counting (DIC) is one such algorithm, which does not wait for a complete database pass to start counting the candidate itemsets. It therefore reduces the number of passes of the database and generates fewer number of candidate itemsets. With the presence of multinational companies at different geographical locations across the globe, the data they need for decision making is inherently distributed. It is necessary to analyse the data to allow company-wide activities such as planning, marketing and sales. Analyzing data locally is not enough. A straightforward solution is to transfer all data to a central site where data mining is done. However even when such a site is available, it may incur huge communication costs to transfer the local datasets because of their sizes. Sometimes the local data cannot be transferred because of the security or privacy of the datasets. Distributed Association Rule Mining (DARM) is an active field in which global association rules are formed for the distributed data. The performance affecting issues in a distributed environment are the disk I/O minimization, the time required for synchronization between the nodes and the message transmission over the network. Our distributed algorithm namely Optimistic Messaging DIC (OPT-DIC) focusses on disk I/O minimization by reducing the number of database passes and has almost no issues of synchronization between the nodes. It generates far fewer candidate sets than Apriori-based, level-wise algorithms because the nodes start counting an itemset early and only if it is frequent at atleast one node. This also reduces to a very large extent the number of bytes tranmitted over the network. The rest of the paper is organized as follows. Section 2 discusses the existing centralized algorithms and DIC. Section 3 deals with the issues in Distributed Association Rule Mining and the work done in the field of Distributed Association Rule Mining.

[1] preetiparanjape@yahoo.com

Section 4 discusses the Optimistic Messaging DIC algorithm.Section 5 discusses the results obtained after comparing the Optimistic Messaging DIC algorithm with CD against various datasets.

2. Association Rule Mining

Let $I=i_1, i_2, ...i_m$ be a set of items. Let D be a database of transactions, where each transaction T consists of a set of items I. The support of an itemset X is the number of transactions in which the itemset occurs as a subset. An itemset is frequent or large if its support is more than some user defined minimum support threshold 'δ'. Thus support is the number of transactions in the database that contain the itemset X. An association rule is an implication of the form X =>Y. The rule X => Y holds in the transaction set D with confidence 'c' if c% of transactions in D that contain X also contain Y. The rule X => Y has support s in the transaction set D if s% of transactions in D contain X U Y. The problem of mining association rules is to generate all association rules that have a certain user-defined minimum support and confidence. Several centralized algorithms exist for Association Rule Mining. One of the first algorithms is Apriori [1], [2] on which most of the parallel algorithms are based. Apriori is an iterative, level-wise algorithm which uses a bottom-up search starting with the counting of frequent 1- itemsets. It generates these itemsets after a complete scan of the database. It then uses a self-join to find the 2-itemsets from the frequent 1- itmsets. It then scans the database to find the frequent 2-itemsets and continues this process till the maximal itemsets are generated. The number of passes is equal to the size of the maximal n-itemset. It uses the large itemset property that is any subset of a large itemset must be large. The large itemsets are also said to be downward closed because if an itemset satisfies the minimum support requirements so will its subsets. Hence, if we know that an itemset is small then we need not generate its supersets as candidates because they will also be small. The performance of Apriori directly depends on the length of the longest frequent itemset. A remarkable breakthrough in sequential algorithms was achieved by the Dynamic Itemset Counting (DIC) [3] algorithm which represents a shift in the method in which frequent itemsets are generated. Since Dynamic Itemset Counting (DIC) forms the basis of our distributed algorithm, we would discuss this algorithm in detail.

2.1. Dynamic Itemset Counting (DIC)

Dynamic Itemset Counting (DIC) [3] is an algorithm which reduces the number of passes made over the data while keeping the number of itemsets which are counted in any pass relatively low. In the first M transactions the algorithm starts counting the 1-itemsets. After M transactions for a given minimum support threshold, if any of the itemsets exceeds the minimum support threshold in those M transactions, then we start counting the 2- itemsets before waiting for a complete scan of the database. In this way, DIC starts counting the 1-itemsets and then quickly adds counters for the 2,3,4,...k-itemsets. We will define this M as a checkpoint. DIC uses checkpoints M transactions apart. DIC counts the frequent itemsets and the minimal small itemsets. Minimal small itemsets are those itemsets which form the boundary between the frequent itemsets and the infrequent ones. Their subsets are frequent itemsets. For every itemset, the counting stops from the same point from where it started i.e after one complete database pass. Thus an itemset can be considered for counting at the next checkpoint instead of waiting until the end of the previous pass. If the data is fairly homogeneous and for small values of M, DIC takes very few passes. If the data is non-homogeneous or it is very correlated, it may not be realized that an itemset is actually large until it has been counted in most of the database.This effect can be reduced considerably with randomizing the order of the transactions. The most important issue in the performance of any ARM algorithm is the type of data structure used to keep track of the many itemsets generated. Particularly the data structure should support the addition of new itemsets, the incrementation of counters of those itemsets and maintaining the itemset states as those that are being counted or active and those which have been counted over the entire database. When itemsets become large the counting of the supersets should be started. The incrementation of the counters has to be done efficiently otherwise the performance of the entire algorithm may degrade. The data structure used in DIC is a trie in which each itemset is sorted by its items.Every itemset that has to be counted or has been counted has a node associated with it. Every node stores the last item in the itemset it represents, a counter, a marker as to where in the file its counting was started, its state and its branches. The branches point to the supersets of the itemsets. These operations are performed at every checkpoint.

3. Distributed Association Rule Mining

In centralized data mining the main concern for the efficiency of a data mining algorithm is its I/O and CPU time. The I/O time is the number of diskreads or the number of passes of the database made by the algorithm.In a distributed environment the communication cost is added which is determined by the network

[1] preetiparanjape@yahoo.com

bandwidth and the number of messages that are sent across the network. Many distributed Association Rule Mining algorithms are a distributed version of Apriori or a modified version of Apriori. Count Distribution, Data Distribution, Candidate Distribution are a few of them [4]. The Count Distribution (CD) algorithm focuses on minimizing communication. In the first pass, each processor dynamically generates its local candidate itemset depending on the items actually present in its local data partition. Hence candidates counted by different processors may not be identical. Each processor exchanges local counts to develop global C_1. C_k is a set of candidate k-itemsets or potentially frequent itemsets. Synchronization takes place at this step. L_k is now generated from C_k. L_k is a set of frequent k- itemsets or itemsets with minimum support. Each processor then decides to terminate or continue to the next pass and as all processors have the same L_k, this decision will be identical. Thus every processor scans its local data asynchronously in parallel and synchronizes at the end of each pass to develop global counts. Experimentation involving Count Distribution, Data Distribution and Candidate Distribution [4] has shown that Count Distribution (CD) outperforms the other two. Hence we have compared our algorithm against CD.

4. The Distributed Version of DIC

We present a distributed algorithm based on DIC namely Optimistic Messaging DIC (OPT-DIC). OPT-DIC runs DIC at each node. At every checkpoint in DIC it reads M transactions and performs all operations of incrementation of the counters and adding supersets of items which become frequent. In the OPT-DIC algorithm we also send and recieve messages at this checkpoint. At each node at every checkpoint, messages in the incoming queue are checked for counts of itemsets which have become potentially frequent at other sites. If counting for those itemsets has not begun at that node, it begins counting for that itemset. With this step we would like to mention that OPT-DIC starts counting only those itemsets which have become locally frequent at atleast one node. This reduces the number of candidates who may ultimately not contribute to the frequent itemset generation. The node sends messages at checkpoints regarding the counts of itemsets which can be potentially frequent without waiting for complete counting of that itemset. This initiates early counting of that itemset at other sites. The main advantage of this algorithm is that it does not wait to synchronize with the other sites. It reports the potential candidate itemsets as soon as they turn potentially frequent at the next checkpoint. This leads to a significant reduction in the number of passes of the database as compared to CD.

A. Optimistic Messaging DIC (OPT-DIC)

In OPT-DIC every site needs to maintain certain information with respect to every other site. To maintain this information, the messages used in OPT-DIC are:

1. T_i - number of transactions present at node N_i
2. $C_{candidate}$ - candidate itemset at node N_i
3. Cinter - candidate itemset with its intermediate local count at node N_i
4. C_{final}^{i} - candidate itemset with its final local count at node N_i
5. Fi - Final message at node Ni, indicating completion of counting.

Initially each node broadcasts message Ti, sending its number of transactions to every node. DIC is run locally at every node. Each node initiates counting of an itemset I at the (n-1) other nodes, if that itemset looks potentially frequent at its site. It does so by broadcasting a message C_i candidate at the next checkpoint. After recieving the counts, it can locally decide whether I is globally frequent or not. Each node N_i maintains information associated with each itemset I to indicate whether it is locally frequent or infrequent and globally frequent or infrequent or unknown. The messages to be broadcast are maintained in a message queue Q_{out} and broadcast at the next checkpoint. All incoming messages are kept in an incoming message queue Q_{in}. When a node Ni counts the itemset I over the entire database and if I is locally or globally frequent, a message C_i final with the count of I is generated at N_i. If no itemsets are to be counted locally N_i broadcasts F_i and waits for incoming messages. If (n-1) F_j messages are recieved by N_i, then all nodes have completed counting.

[1] preetiparanjape@yahoo.com

```
For each node N_i, we perform the following
begin
  Run DIC locally
  At each checkpoint at N_i
     I. For any message m in Q_in do
     begin
           1. If m is C^j_final
                 update count of I for j
           2. If m is C^j_inter
                 a) Start counting of I,
                 if not yet started
                 b) If I is globally frequent
                 OR globally infrequent
                       Mark I
     end

     II. If counting of I complete do
     begin
           If ( I is locally infrequent
           AND globally frequent)
           OR
           (I is locally frequent
           AND globally infrequent)
                 Add C^d_final for I in Q_out
     end

     III. During counting of I do
     begin
           If I becomes a local candidate
           OR I turns locally frequent
           AND I not marked globally frequent
                 Add C^d_inter OR C^d_candidate;
                 to Q_out
     end

     IV. If no itemsets are to be counted
           Broadcast Fi.
     Ni waits for incoming messages
     If (n-1) F_j messages received
        Counting at all nodes is complete.

end
```

C. The Algorithm

The OPT-DIC algorithm retains the basic essence of DIC in the distributed version in terms of communicating the itemsets at the next checkpoint when they become likely candidates locally. This helps the other sites in starting the counting for those itemsets, if counting for them has not already been started,at the next possible checkpoint. In the worst case the communication complexity is such that there is a message broadcast at every checkpoint. To reduce the number of messages the interval between checkpoints (the value of M can be increased but many a times this adversely affects the performance as there is a delay in conveying the candidate itemsets which are locally frequent. The disadvantage of CD and many Apriori-based algorithms is that the number of bytes transmitted increase rapidly with the number of nodes. This is also because a lot of globally infrequent but locally frequent candidate itemsets are broadcast between nodes. This factor is reduced to a great extent in OPT-DIC as a node starts counting an itemset only if it is frequent at atleast one site. Without communicating with the other nodes CD proceeds with its entire database pass and then broadcasts the itemsets generated. This may contain many itemsets which may not contribute towards the global frequent itemset generation.

5. Experimentation And Analysis

We have compared OPT-DIC and CD using a Discrete Event based Simulator. We have tested Optimistic Messaging DIC and CD on the mushroom dataset [5], the retail dataset and two synthetic datasets T10I4D100K and T40I10D100K generated from [6]. The performance metrics we have considered are the total time taken, number of passes, number of diskreads, number of bytes and the number of messages. The number of diskreads take into account the number of passes so we have not elaborated on the number of passes. We have experimented on various values of support threshold ‘δ’ for the various datasets.We have packetized the mesages generated at the end of each pass in CD and at a checkpoint in OPT-DIC with a MTU of 1500 bytes and a header of 20 bytes. To calculate the disk access time we have considered the seek time (3 msec), disk latency time (2 msec) and disk transfer rate (1000 Mbps). To calculate time for transmission across the network, latency time (15 msec) and bandwidth (1 Mbps) have been considered.

A. Results for Equipartitions

We have tested the results of the algorithm by equipartitions and on gaussian partitioning, i.e. to check the results on variable paritions as well as for similar partitions.

(1) Total time taken : Due to the huge size of the database and the disk and network latency, time taken is heavily dependent upon the maximum number of passes and the number of messages transmitted. In case of CD, sites synchronize at the end of every pass. This makes the sites with minimum records wait for the sites with maximum records to send their counts.In case of OPT-DIC, none of the sites try to synchronize. From Figure 1 we observe that the time required for OPT-DIC for T10I4D100K is around 52% less than that required for CD and this reduction in time is almost constant for an increase in the number of nodes. The time required for OPT-DIC for T40I10D100K is around 32% less than that required for CD and this reduction in time is almost constant for an increase in the number of nodes. We have observed that the rate

[1] preetiparanjape@yahoo.com

of reduction in time required for OPT-DIC as compared to CD for Retail is almost constant between 60 to 70%. We observe that the rate of reduction in time required for OPT-DIC as compared to CD for Mushroom increases with the increase in number of nodes.

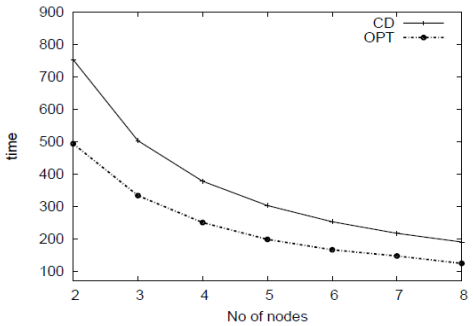

Fig. 1: Time taken for T10I4D100K for δ = 1% and M=100

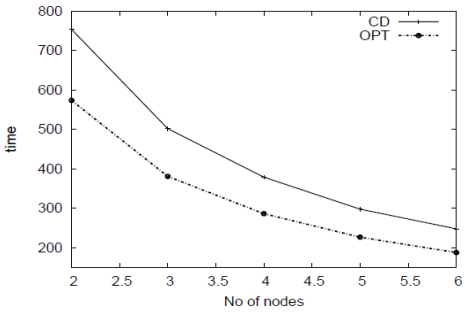

Fig. 2: Time taken for T404D100K for δ = 4% and M=100

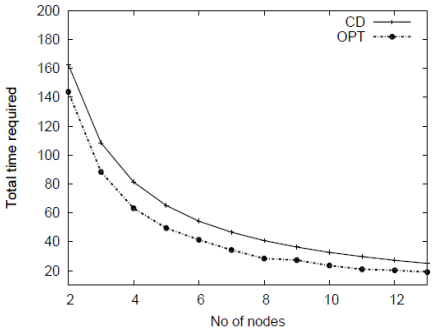

Fig. 3: Time taken for mushroom for δ = 50% and M=100

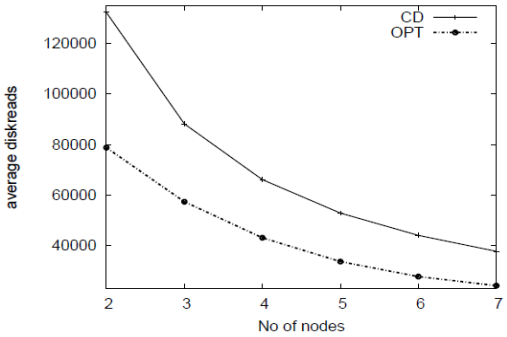

Fig. 4: Diskreads required for 'retail' for δ = 10% and M=100

(2) Number of diskreads : The number of passes in OPT-DIC are much less than those in CD. Local counts of all itemsets is always maintained. As a result if n globally frequent itemsets are not locally frequent at the same site, they would never be used to generate a candidate at the next level. Not only does this reduce the number of candidates but it can save upto one extra pass. All these factors reduce the number of diskreads.

(3) Number of Bytes transmitted:

The number of bytes transmitted by CD in the first pass is quite high as it sends the counts and names of all the frequent 1-itemsets. In subsequent passes, only counts of all candidates are transmitted. In the case of OPT-DIC the count and name of each itemset with cardinality more than 1 is transmitted. But the number of candidate itemsets generated in CD are far more than those in OPT-DIC. This is because after synchronizing CD has only the global counts of itemsets. As a result even if n globally frequent itemsets are not locally frequent at the same site they may be used by CD to generate a candidate at the next level even though the candidate may never turn locally/globally frequent anywhere. Hence the number of bytes transmitted in CD are more than those in OPT-DIC.

(4) Number of messages transmitted: By messages here, we mean the number of packets transmitted over the network. In case of OPT-DIC, if some itemsets have turned potentially frequent, messages are generated at that checkpoint.In the worst case, a message is broadcast at every checkpoint. Hence the number of messages is higher in OPT-DIC compared to CD. But though the number of messages is higher, OPT-DIC fares much better than CD in the time taken.

[1] preetiparanjape@yahoo.com

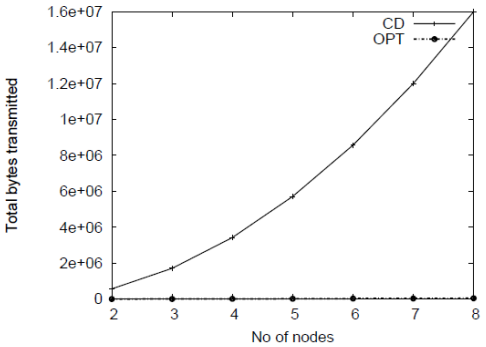

Fig. 5: Bytes generated for T10I4D100K for δ = 1% and M=100

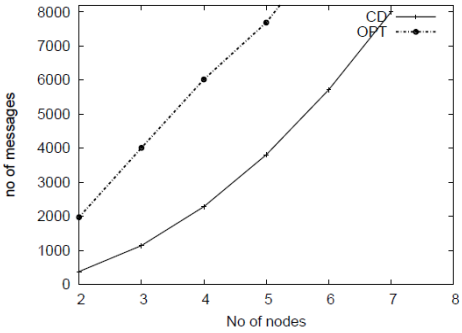

Fig. 6: Messages generated for T10I4D100K for δ = 1% and M=100

B. Results for Gaussian Partitioning

We have partitioned the database using the normal distribution which represents the actual distribution of data at various sites in an actual distributed setup. We observe that with gaussian partitioning the performance gain in terms of reduction in the time required in OPT-DIC is much higher than that with equipartitioning. The total bytes and messages transmitted in gaussian partitioning is also less than those in equipartitioning.

ACKNOWLEDGEMENT

We are thankful to Shreyas Belle and Rahul Suresh for their valuable contribution in the implementation of OPT-DIC.

REFERENCES

[1] R.Agrawal, T.Imilienski, and A.Swami, "Mining association rules between sets of items in large databases," Proc. of the ACM SIGMOD int'l Conf. on Management of Data, vol. May, no. 1993, pp. 207–216.

[2] R.Agrawal and R.Srikant, "Fast algorithms for mining association rules," In Proceedings of the 20th VLDB Conference, Santiago, Chile,1993, 1996.

[3] S.Brin, R.Motwani, J.Ullman, and S. Tsur, "Dynamic itemset counting and implication rules for market basket data," SIGMOD Record, vol. 6, no. 2, pp. 255–264, June 1997.

[4] R.Agrawal and J.Schafer, "Parallel mining of association rules," IEEE Transactions on Knowledge and Data Engineering, vol. 8, no. 6, pp. 962–969, 1996.

[5] Uci machine learning repository. [Online]. Available: http://archive.ics.uci.edu/beta/datasets/Mushroom

[6] Synthetic data generation code for association and sequential patterns. [Online]. Available: IBM Quest website at http://www.almaden.ibm.com/cs/quest/

[1] preetiparanjape@yahoo.com