FREQUENT ITEMSET MINING

TEAM SOMETHING – ARJUN GUPTA, VRISHABH LAKHANI
• **Apriori** is an algorithm for frequent item set mining and association rule learning over transactional databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database. The frequent item sets determined by Apriori can be used to determine association rules which highlight general trends in the database: this has applications in domains such as market basket analysis.


APRIORI-BASED FREQUENT ITEMSET MINING ALGORITHMS ON MAPREDUCE

• Scalability of mining algorithm is prone to errors, thus it becomes necessary to use a MapReduce based framework in order to make an efficient reliable parallel Frequent Itemset Mining algorithm.

• Converting a serial Apriori-like mining algorithms into a distributed algorithm on the MapReduce framework is not difficult but the mining performance is unsatisfactory. Multiple-pass on the algorithm necessitates multiple map-reduce phases, thus a good scheduling scheme is needed such that it reduces the number of scheduling invocations, maximize the utilization of each node during a single MapReduce phase and perform smart Load Balancing.

• The paper compares three possible approaches to schedule the nodes in the cluster so that we can perform MapReduce task efficiently for each phase.
APRIORI-BASED FREQUENT ITEMSET MINING ALGORITHMS ON MAPREDUCE

• SPC algorithm, where it iterates over the dataset once in every phase of the MapReduce.
• FPC algorithm, where it iterates over the dataset k times when the maximum length of the frequent itemsets is k.
• In order to avoid algorithms from suffering from overloading candidates for a mapper if the number of candidates after merging is too large, we use DPC, which dynamically schedules the nodes for each phase.
• The paper concludes DPC can be used to stricke a balance between reducing the number of map-reduce phases and increasing the number of prunes candidates. Thus is ourperforms the other two methods and can be used for efficient Frequent Itemset Mining with Mapreduce
Map (key, value = itemset in transaction t) :
Input: a database partition D and $L_{k-1}$ (k > 2)
1. read $L_{k-1}$ from DistributedCache ;
2. Candidate threshold $ct = \alpha \cdot |L_{k-1}|$;
3. $C_{sel} = C_k = \text{apriori-gen}(L_{k-1})$;
4. for (counter = 0; |$C_{sel}$| $\leq ct$; counter++)
5. $C_{k+1} + \text{counter} = \text{apriori-gen}(C_{k+counter})$;
6. $C_{sel} = C_{sel} \cup \text{apriori-gen}(C_{k+1+\text{counter}})$;
7. end
8. construct a prefix-tree for $C_{sel}$ :
9. foreach transaction $t_i \in D$, do
10. $C_i = \text{subset}(C_{sel}, t_i)$;
11. foreach candidate $c \in C_i$ do
12. output <c, 1> ;
13. end
14. end

Reduce (key=itemset, value=count) :
1. foreach key y do /* Initial y.count = 0 */
2. foreach value v in y's value list do
3. y.count += v ;
4. end
5. if y.count $\geq$ minimum support count
6. output <y, y.count> ; /* collected in $L_k \cdots L_{k+\text{counter}}$ */
7. end
8. end

Figure 8. Phase-k (k>2) of DPC algorithms.

Algorithm DPC
1. Phase-1: find $L_1$ /* Figure 1 */
2. Phase-2: find $L_2$ /* Figure 2 */
3. for (k = 3; $L_{k-1} \neq \phi$) /* each phase k */
4. Map function /* Figure 8 */
5. Reduce function /* Figure 8 */
6. k += (counter+1) ;
7. end

Figure 9. Algorithm DPC.
Problem:

- Generally, it takes long to find the association rules between datasets when a database contains a large number of transactions.
- By applying parallel-distributed data mining techniques, the mining process can be effectively speeded up.
- With parallel-distributed data mining the calculation is done in a distributed environment but most of the time, irregular and imbalanced computation loads are allocated between processors and thus the overall performance is degraded.
A WEIGHTED LOAD-BALANCING PARALLEL APRIORI ALGORITHM FOR ASSOCIATION RULE MINING

• Solution:
  • In this paper the Weighted Distributed Parallel Apriori algorithm (WDPA) is presented as a solution for this problem. In the proposed method, a database has only to be scanned once because metadata are stored in TID tables. This approach also takes the TID count into consideration. Therefore, WDPA improves load-balancing as well as reduces idle time of processors.
By only calculating the Lattice number and ignoring the length of the itemset TID, an uneven distribution of workload occurs. Therefore, this algorithm also takes TID length into consideration and regards it as a weight value, which makes the distribution of itemsets more accurate and more even.

\[
\text{Cnt\_Lattice}(I_i) = [(\text{len}(\text{freq}_{k-1}) - 1) - i]
\]

\[
\text{TotalCnt\_Lattice} = \sum_{i=0}^{\text{len}(\text{freq}_{k-1})-1} \text{Cnt\_Lattice}(I_i)
\]

\[
\text{Value\_WeightTid}(I_i) = \sum_{j=i+1}^{\text{len}(\text{freq}_{k-1})-1} \text{len}(I_{i_{\text{TID}}}) \times \text{len}(I_{j_{\text{TID}}})
\]

\[
\text{TotalValue\_WeightTid} = \sum_{i=0}^{\text{len}(\text{freq}_{k-1})-1} \sum_{j=i+1}^{\text{len}(\text{freq}_{k-1})-1} \text{len}(I_{i_{\text{TID}}}) \times \text{len}(I_{j_{\text{TID}}})
\]
Figure 4. Distributing frequent 1-itemsets on $P_1$
EFFICIENT MINING USING ENHANCED APRIORI WITH HASH TREE AND FUZZY

• Their design tool allows experimenting with the concepts of fuzzy modification of association rules.
• They then analyzed the crisp boundary problem in the algorithm and how it can be overcome by a modified association Apriori hash tree fuzzy algorithm.
• Goal is to increase efficiency of overall system by compromising with the accuracy.
EFFICIENT MINING USING ENHANCED
APRIORI WITH HASH TREE AND FUZZY

• Ways to improve efficiency
  • Prune without checking all \( k-1 \) itemsets
  • Join without looping over entire set \( L_{k-1} \)
  • Reduce the number of transactions
  • Reduce the number of candidates
  • Reduce the number of subsets to be considered per transaction
THANK YOU