Efficient Mining of Association Rules in Distributed Databases

David W.Cheung, Vincent T.Ng,
Ada W.Fu, Yongjian Fu

Presented by Yuhong Guo

Outline

- Motivation
- Problem Definition
- DMA
- Performance study
- Discussion
- Conclusion

Motivation

For association rule mining:

- Many algorithms in Sequential Environment: Apriori, DHP, Partition
- In Parallel Environment:
  --PDM (generalization of DHP)
  --CD (parallelization of Apriori)

Motivation

- Problem of PDM and CD:
  --Higher number of candidate sets
  --O(n*n) messages for support count exchange for each candidate set.
- DMA (Distributed Mining of Association Rules)
  --The data skewness property of distributed databases.
  --A much smaller number of candidate sets
  --O(n) messages for support count exchange for each candidate set.

Problem Definition

- An association rule is of the form : $X \Rightarrow Y$, where $X \subset I$, $Y \subset I$ and $X \cap Y = \varnothing$.
- The rule $X \Rightarrow Y$ holds in the transaction database DB with confidence c if c% of transactions in DB that contain X also contain Y.
- The rule $X \Rightarrow Y$ has support s in DB if s% of transactions in DB contains $X \cup Y$.

Problem Definition

- s – minsup , c -- minconf
  DB is a partitioned database
  $\{DB^1, DB^2, ..., DB^n\}$
  located at sites $S^1, S^2, ..., S^n$
- An itemset X is globally large if its support count X.sup$\geq s \times |DB|$. Then X is a large itemset.
- IF $X.\sup^i \geq s \times |DB^i|$, X is locally large at S^i

Problem Definition

- If X is both locally large at site S^i and globally large, we say X is heavy at site S^i .
- Problem of mining association rules in a distributed databases DB can be reduced to finding of all globally large itemsets.

Lemmas

DMA

- Lemma1: If an itemset X is locally large at a site S^i, then all its subsets are also locally large at site S^i.
- Lemma2: If an itemset X is globally large, then there exists a site S^i such that X and all its subsets are locally large at site S^i.

Lemmas

- Lemma3: If an itemset X is globally large,then there exists a site S^i such that X is heavy at site S^i. (=lemm2)
- Lemma4: If an itemset X is heavy at a siteS^i, then all its subsets are also heavy at site S^i.
- Lemma5: If X∈ L_k, then there exists a site S^i, such that X and all its size(k-1) subsets are heavy at site S^i.

(Lemm5=Lemma3+Lemma4)

Notation Table

D	The number of transactions in database DB
s	The support threshold $minsup$
L_k	The set of globally large k-itemsets
CA_k	The set of candidate sets generated from L_{k-1}
$X.sup$	The global support count of an itemset X
D^i	The number of transactions in the partition DB^i
HL_k^i	The set of heavy k-itemsets at site S^i
CH_k^i	The set of candidate sets generated from HL_{k-1}^i
LL_k^i	The set of locally large k-itemsets in CH_k^i
$X.sup^i$	The local support count of an itemset X at site S^i

Theorem 1

- Theorem1: For k>1, the set of all large k-itemsets L_k is a subset of

$$CH_k = \bigcup_{i=1}^{n} CH_k^i,$$

$$where \qquad CH_k^i = Apriori\_gen(HL_{k-1}^i).$$

Hence CH_k is a set of candidate sets for the size-k large itemsets.

Theorem 1

$$CH_k = \bigcup_{i=1}^{n} CH_k^i = \bigcup_{i=1}^{n} Apriori\_gen(HL_{k-1}^i),$$

$$CA_k = Apriori\_gen(L_{k-1}) = Apriori\_gen(\bigcup_{i=1}^{n} HL_{k-1}^i).$$

- In general,$|CH_k| \le |CA_k|$. So by using theorem 1, the number of candidate sets can be reduced.

Example 1

Example 1:

$DB-3sites: DB^1, DB^2, DB^3, L_1=\{A,B,C,D,E,F,G\}.$

$HL_1^1=\{A,B,C\}, HL_1^2=\{B,C,D\}, HL_1^3=\{E,F,G\},$

$CH_2^1=\{AB,BC,AC\}, CH_2^2=\{BC,CD,BD\}, CH_2^3=\{EF,FG,EG\},$

$CH_2=\{AB,BC,AC,CD,BD,EF,FG,EG\}, 8-candidates.$

$CA_2=Apriori\_gen(L_1), 21-candidates.$

$|CH_2| \leq |CA_2|$

Local Pruning of candidate sets

- According to Lemma 2, 3 or 5:
  X$\in$ Lk $\Rightarrow$ $\exists$site S^i ,X is locally large at S^i,
  || I.e. X$\in LL_k^i$
  $\forall S^i$, X$\notin LL_k^i \Rightarrow$ X $\notin$ Lk
- Reduce locally candidate set by removing X which is not locally large.
  From CH_k^i to LL_k^i

Procedure for DMA to Compute HL_k^i (1-5)

1.Candidate Sets Generation

$CH_k^i = Apriori\_gen(HL_{k-1}^i)$

2.Local Partition Scanning

For each X$\in CH_k^i$, compute its local support count $X.\sup^i$.

Procedure for DMA to Compute HL_k^i (1-5)

3.Local Pruning $CH_k^i \xrightarrow{pruning} LL_k^i$

4.Support Count Exchange

For each X$\in LL_k^i$, broadcast to other sites to collect support counts , then

$LL_k^i \Rightarrow HL_k^i$.

5.Broadcast Mining Result HL_k^i

Example for Computing HL_k^i DMA

- Example 2:

The database has 150 transactions and 3 partitions, each has 50 transactions. Support threshold s=10%.

We have known HL_1^i :

$HL_1^1 = \{A, B, C\}, HL_1^2 = \{B, C, D\}, HL_1^3 = \{E, F, G\},$

$\Downarrow (1) candidate\_generation$

$CH_2^1 = \{AB, BC, AC\}, CH_2^2 = \{BC, CD, BD\}, CH_2^3 = \{EF, FG, EG\}$

Example for Computing HL_k^i DMA

(2)*local–partition–scanning*
==========>

S^1		S^2		S^3	
CH_2^1	$X.sup^1$	CH_2^2	$X.sup^2$	CH_2^3	$X.sup^3$
AB	5	BC	10	EF	8
BC	10	CD	8	FG	3
AC	2	BD	4	EG	3

(3)*local – pruning*
========== >

S^1		S^2		S^3	
LL_2^1	$X.sup^1$	LL_2^2	$X.sup^2$	LL_2^3	$X.sup^3$
AB	5	BC	10	EF	8
BC	10	CD	8	~~FG~~	~~3~~
~~AC~~	~~2~~	~~BD~~	~~4~~	~~EG~~	~~3~~

Example for Computing HL_k^i DMA

(4)support–count–exchange
==========>

locally large candidate sets	request broadcast from sites	$X.sup^1$	$X.sup^2$	$X.sup^3$	$X.sup$
AB	S^1	5	4	4	13
BC	S^1, S^2	10	10	2	22
CD	S^2	4	8	4	16
EF	S^3	4	3	8	15

(5)broadcast–heavy–2–itemsets
==========>

$HL_2^1 = \{BC\}, HL_2^2 = \{BC, CD\}, HL_2^3 = \{EF\}$

Message Optimization for Finding Large itemsets DMA

- **Problem:** For the support count exchange, in worst case, it'll require O(n*n) messages for each candidate set (big overlap), though the possibility is small.
- How to make sure this O(n*n) to be O(n)?

  Solution: For each candidate set, determine a polling site by hash function or whatever.

Message Optimization for Finding Large Itemsets

DMA

- Replace the procedure steps 4,5 with:

1. Candidates sent to Polling Sites.

$$S^i \xrightarrow{Send\ -LL_k^{i,p}} polling\ -site : S^p$$

2. Polling Site send Polling Requests.

$$S^p \xrightarrow{Send\ -LL_k^{i,p}} all\ -sites : S^j (j \neq i)$$

3. Remote Site reply Polling Requests (scan again?No).

4. Polling Site Compute Heavy Itemsets, Broadcast them together with their global support counts.

Optimizing Partition Scanning for Count Exchanges

DMA

- **Problem**: A site S^i needs to do local partition scan for:

(1) getting the local counts of all the candidate sets CH_k^i generated at site S^i.

(2) replying the polling requests for $LL_k^{j,p}$.

Can we do all this in one scan?

Sure we can.

Optimizing Partition Scanning for Count Exchanges

DMA

- **Solution**: Do it in one scan at step 2 "Local Partition Scanning".
- How can we know those $LL_k^{j,p}$?

  A fact: The heavy sets for candidate set generation are available to all the sites at the end of each iteration.

  So each site only needs to scan its partition once to find local support counts of all the itemsets in $CH_k = \bigcup_{i=1}^{n} Apriori\_gen(HL_{k-1}^i)$

  Exception: The first iteration—take all the size-1 itemsets.

Message size optimization

DMA

- Since every site has the same set of candidate sets CH_k , so in a polling request for an itemset, we can send their position in ordered list of the itemsets in CH_k , instead of the itemset names

DMA Algorithm

DMA

- At each site S^i, iterates the following steps, starting from k=1.
  1. Generate candidate sets
  2. Local partition scanning
  3. Local Pruning
  4. Send Candidates to Polling Sites
  5. Receive Candidates as a Polling Sites
  6. Send Polling Requests as a Polling Site to Collect Support Counts.
  7. As a Polling Site, Compute Global Support Counts and Broadcast the Heavy Itemsets

Performance Study of DMA

- Synthetic databases generated with certain skewness.
- Compare DMA with CD.
- Two experiments:
  - ♦ performance comparison with different thresholds and database sizes.
  - ♦ performance comparison with different number of sites.

Experiment 1-different thresholds and database sizes

Setting:

- ♦ 3 sites.
- ♦ sizes of the databases range from 100K to 900K transactions.
- ♦ support threshold ranges from 0.75% to 2%.

Experiment 1-different thresholds and database sizes

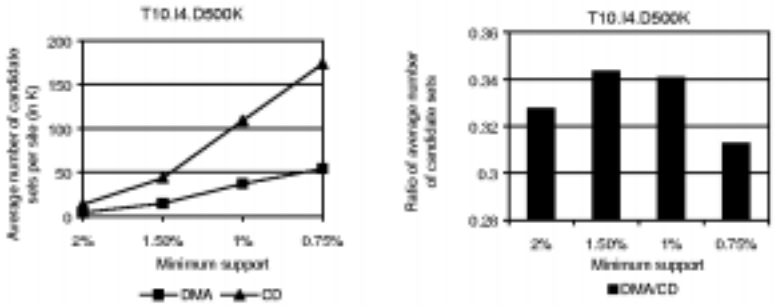

Figure 1: Candidate Sets Reduction.

Experiment 1-different thresholds and database sizes

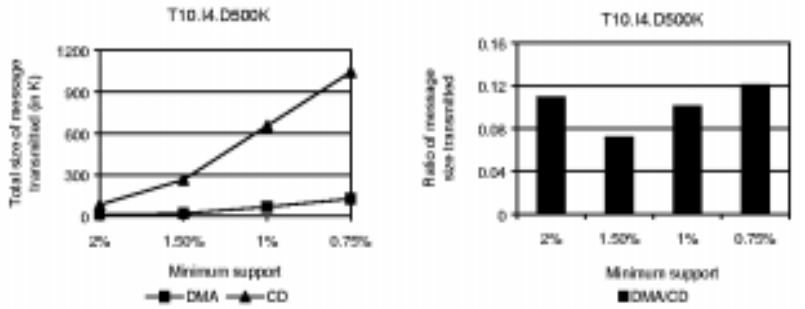

Figure 2: Message Size Reduction.

Experiment 1-different thresholds and database sizes

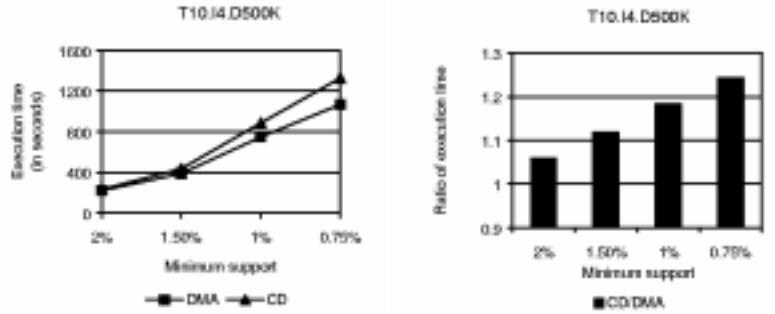

Figure 3: Execution Time Speed Up.

Experiment 1-different thresholds and database sizes

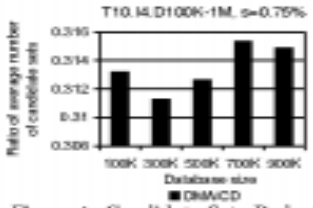

Figure 4: Candidate Sets Reduction.

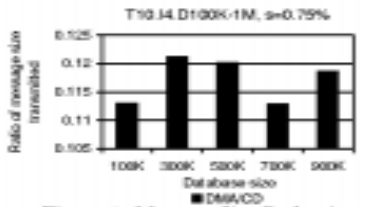

Figure 5: Message Size Reduction.

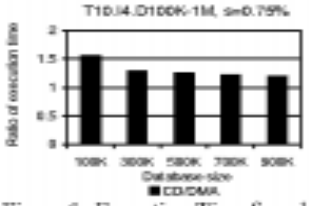

Figure 6: Execution Time Speed Up.

Experiment 2-different number of sites

Setting:

- size of database is 200K.
- support threshold is 3%.
- sites number are 3, 4, 5, 6 respectively.

Experiment 2-different number of sites

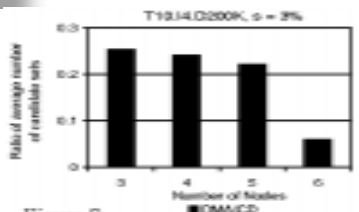

Figure 7: Candidate Sets Reduction (s = 3,4,5,6).

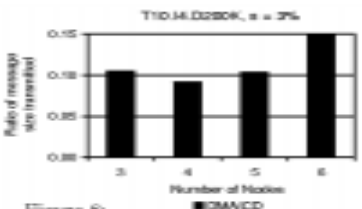

Figure 8: Message Size Reduction (s = 3,4,5,6).

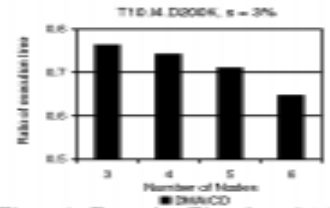

Figure 9: Execution Time (s = 3,4,5,6).

Discussion

- The efficiency of DMA is attributed to three techniques:
  - Candidate sets generation
  - Local Pruning (only local information considered)
  - Message Optimization O(n)
- Can we do more by using the global information got from the broadcast at the end of each iteration?

Global Pruning

- Based On: We can get the upper bound of the local support counts of X from the minimum local support counts of all the size (k-1) subset of X.

$$\max\sup^i(X) = \min\{Y.\sup^i \mid Y \subset X, and \mid Y \mid = k-1\}$$

The upper bound of global support counts: $X.\sup < \max\sup(X) = \sum_{i=1}^{n} \max\sup^i(X)$

Then if $\max\sup(X) < s \times D$, we can prune X.

Global Pruning Combined with Local Pruning

- Local Pruning followed by Global Pruning:

$$\max\sup(X) = X.\sup^i + \sum_{j=1, j\neq i}^{n} \max\sup^j(X)$$

- Global Pruning followed by Local Pruning
- Global Pruning at Polling Site:

$$\max\sup(X) = \sum_{i\in\Gamma} X.\sup^i + \sum_{j=1, j\notin\Gamma}^{n} \min(\max\sup^j(X), s \times D^j)$$

Conclusion

- Techniques:

  Candidate set generation,

  Local pruning,

  Message optimization,

  Optimizing partition scanning

- Performance:

  DMA has superior performance than CD in distributed databases.

- Future research:

  Extend DMA to the mining of multiple-level or generalized association rules in distributed databases.

Thank You!