Distributed Data Access Control Algorithm Using Mining Association Rules

Dr.N.Rajkumar[1], Dr.S.N.Sivanandam[2], J. Stanly Thomas[3]

[1]Professor and Head, Department of CSE, New Horizon College of Engineering, Bangalore, Karnataka, India
[2]Professor and Head, Department of CSE, PSG College of Technology, Coimbatore, Tamilnadu, India
[3]Research Scholar, Dept of Computer Applications, Periyar University, Salem, Tamilnadu, India

Abstract
In this paper, we present a new algorithm called Distributed data access control algorithm using association rules in large database respectively. The large amounts of data, the high scalability of distributed systems, and the easy partition and distribution of a centralized database, it is important to investigate efficient methods for distributed mining of generalized association rules [3,4]. This study discloses some interesting relationships between locally and globally large itemsets [6,7]. The proposed algorithm, which distributes a data with various databases and participate in a network using mining association rules. In this paper, efficient algorithm for mining generalized association rules in distributed database has been proposed and is based on FDM algorithm. Concepts of data mining with distribution law can improve the finite set. Every entry redirected into a FAK to simplify the next cycle of operation. Hit Count can also measure in a FAK by means of succeeded value. Time study divided into two phases according to a nature of data. Former one is frequently accessed data and the later one is ordinary data set. The FDM algorithm is combined with partition algorithm to give fast distributed data access control algorithm called DDACA.

Key words:
Data mining, Association rules, DDACA, FAK

1. Related Works

Apriori based distributed algorithms called fast distributed mining of association rules (FDM) developed by Cheung et.al.,[8]. The description of KSA [10] and FDM algorithm is given below.

Let DB be a database with D transactions. Assume that there are n sites S1, S2 ….Sn in a distributed system and the database DB is partitioned over the n sites in to {DB1, DB2 …DBn} respectively.

The FDM algorithm generates a small set of candidate sets and local pruning candidate sets in each site (si) and combines all the locally large item sets in each site to produce globally large itemsets of whole database DB. It suggests three optimizations; local pruning, global pruning, and count polling. Each site generates candidates using the globally large itemsets of site si from all sites and assigns a home site for each candidate. The home site then broadcasts the global supports to all other sites. Thus, FDM requires far less communication, and local pruning cuts it down even more.

Key semaphore algorithm [3] is quite delayed algorithm due to FAK (i.e.) it requests a master database only if a user request not physically available in a FAK. Semaphores such as “wait and signal” may create a long waiting state for chained user request.

2. Proposed Scheme

In the following sections, the problem of mining generalized association rules in the distributed environment is presented. The following section concentrates on the problem description followed by the novel and efficient algorithm for mining generalized association rules in the distributed environment.

3. Problem Description

Let DB_i ($1 \le I \le n$) be a partitioned database located in n sites S_1, $S_2 \ldots S_n$ with their sizes as DB_1, $DB_2 \ldots\ldots\ldots DB_n$ respectively. Let the size of DB and the partitions DB_i be D and D_i respectively. Let X, Let X.sup and X_i sup_i be the respective support counts of X in DB and DB_i.

It is called X.sup. The global support count and $X.sup_i$ is the local support count of X at site s_i. For a given minimum support S,X is globally large if $X.sup \ge S$ X D; accordingly, X is locally large at site s_i, if $X.sup_i \ge Sx\ D_i$. Let L denote all the globally large itemsets in DB and L_k is used to denote all globally large K-item sets in L. A special database termed as frequently accessed key (FAK) supports an instant data access without semaphore controls.

The major highlight of the DDACA is to distribute a user request to both the FAK as well as the master database, which has a free status.

User request is given to FAK for quick result in case of frequently accessed data. Every entry redirected in to FAK simplifies the next cycle of operation.

Manuscript received August 5, 2008
Manuscript revised August 20, 2008

4. Distributed Data Access Control Algorithm (DDACA)

In the following section, the proposed distributed algorithm DDACA is presented in a detailed manner. The pseudocode for the distributed data access control algorithm (DDACA) is given below:

Input :

(1) DB_i: The database partition at each site with equal size, say D_i.

(2) S: The minimum support threshold; both used at each site (I=1 …n).

Output:

L: The set of large itemset in DB at all sites.

```
1.  Module ans_set
2.  Key: variable;
3.  Begin
4.  ma1, ma2, macap, fak: array [1...capacity][1..capacity] of data;
5.   r_input, c_input: (1...capacity);
6.  large: integer;
7.  in: usrin;
8.  status: (b/f);  {b->busy,f->free}
9.  procedure state_watch (pdata: data);
10. begin
11. forall ma [r_input] to capacity
12. begin
13. {find free status among 'n' database}
14. if status is 'f' then
15. begin
16. ma[r_input].status:='b';
17. call ans_search(ma[r_input]);
18. end if;
19. 19.  end;
20. end; {state_watch}
21. procedure ans_search (pdata:data);
22. begin
23. forall r_input to capacity
24. begin
25. forall c_input to capacity
26. begin
27. if fak[r_input][c_input]=in : pdata.ma[r_input]=in then
28. begin
29. {data found in fak}
30. raise found
31. updatecount[r_input][c_input]= count[r_input][c_input] +1;
32. pdata.ma[r_input].status:='f';
33. quit;
34. end if;
35. end for;
36. end for;
37. end; {ans_search}
38. procedure hitcount( );
39. begin
40. large:=fak[1]; {basis of hit count}
41. forall r_input=2 to capacity
42. begin
43. if fak[r_input]>large then
44. begin
45. large:=fak[r_input];
46. {focus of large}
47. end if;
48. end;
49. end; {hitcount}
50. end; {key}
51. r_input:=1;
52. c_input:=1;
53. count[r_input][c_input]:=1;
54. end;
55.  {ans_set}
```

5. Description of Distributed Data Access Control Algorithm

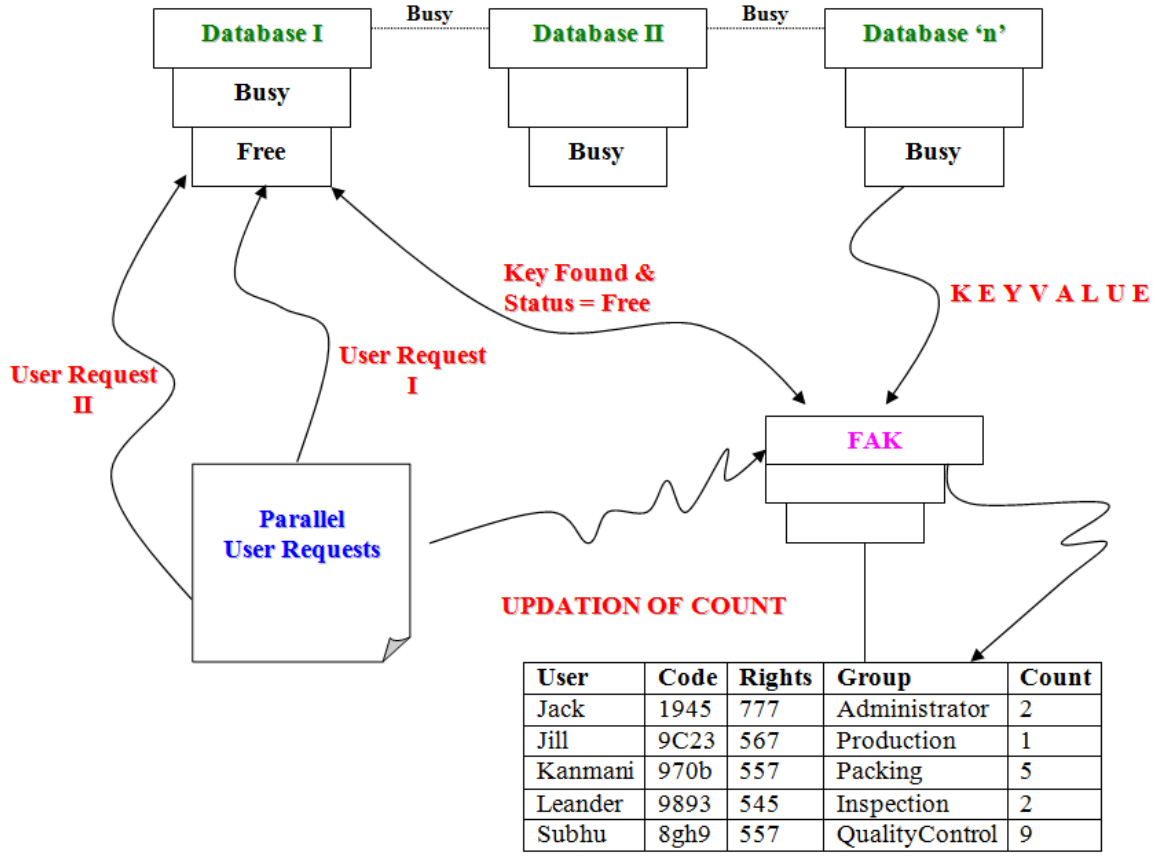
The step-by-step procedure for description of DDACA algorithm is given below:

1. Getting user input in deciphered format.
2. Monitor 'n' number of databases for free status.
3. Input Key is transferred to database, which has "free" status in a searching process.
4. Change the status of the database to "busy" state to protect.
5. It transfers simultaneously an Input key to "FAK" for searching.
6. If an input key is frequently used, then the entry may exists in FAK and FAK process before the master database. Because, FAK is comparatively smaller than master database.
7. If the searching process is successfully completed then update "Hit Count" value by 1 using the procedure ans_search.
8. After releasing of data from master database, change status to "free" for next key using the procedure state_watch.
9. Calculate the greater "Hit count" of a cycle using the procedure hit count.

6. Classics of Distributed Database System

Distributed database systems are capable of handling both local and global transactions. The system resolves all local database requests, access to data at other sites, and any requests it may receive from other sites. The system masks

differences in the various local systems by providing a common network wide view of the data.
Through appropriate translation mechanisms, requests expressed on the common view can be translated to the local system view being accessed. In addition to network and data distribution characteristics, the major issues in a Distributed Database Management System (DDBMS) are query processing (including transaction processing), concurrency control, and recovery.



User	Code	Rights	Group	Count
Jack	1945	777	Administrator	2
Jill	9C23	567	Production	1
Kanmani	970b	557	Packing	5
Leander	9893	545	Inspection	2
Subhu	8gh9	557	QualityControl	9

Fig 1: FAK hit count system

Even when a portion of a system (i.e., a local site) is down, the system remains available. With replicated data, the failure of one site allows access to the replicated copy of the data from another site[10]. The remaining sites continue to function. The greater accessibility enhances the reliability of the system. A query involving data from several sites can be subdivided into sub-queries and the parts evaluated in parallel.

Data distribution in DDBMs with redundant copies can be used to increase system availability and reliability [11]. If data can be obtained from a site other than the one that has failed, then availability improves, and as the system can still run, reliability does too.

7. FAK hit count system

Fig 1 shows FAK hit count system, which consists of different databases. Parallel user request getting in the format of deciphered entity. Control transfers to a database participate in a network according to a status it has.

Once database intakes a user request status code automatically changed to "Busy". And database locked for other user request. To overcome the delay problem in KSA, user request given to FAK database as well as master database simultaneously.

Probability of FAK is higher to win a searching process in case requested data is frequently accessed. On the other hand, if it is not a frequently accessed data then the probability of master database is high.

If FAK wins a searching process hit count value updated by 1. If master database wins then FAK receives a user request with key value from a master database and starts a hit count account with initial deposit of 1. Interrupt signal named "FOUND" passed from a database that wins a searching process to lost database. Simple check of greater hit count can raise the frequent user entry and its rights.

8. Generation of Synthetic Data and Real-life Dataset

The databases used in the experiments are synthetic data as well as real-life datasets. This is confirmed by the Table 3 and 5, which shows the relative execution time as we increase the number of input records from 100 sets to 100000 sets, for four different levels of datasets.

The entities consist of user_no, user_name, user_group, basic_pay, user_rights, user_count, DA, HRA, CCA, emp_category and marital_status. There were 100,000 sets in the data. Our experiments were performed on an IBM Sun Solaris system with firewall and 512 RDRAM, 80GB RAID swappable system.

9. Difference between KSA and DDACA System

The below Table 2 shows the difference between key semaphore algorithm (KSA) and distributed data acess control algorithm (DDACA).

Key semaphore algorithm	Distributed data access control algorithm
Semaphores such as "wait and signal" may create a long waiting state for chained user request.	User request distributed to different free databases only, therefore no chance for deadlock or long wait.
Higher priority node may dominate a signal always. (Long wait for lower priority systems)	Instead of priority system, group function used to intake set of inputs in parallel fashion.
Parallel input may disturb an entire algorithm	Well suited for parallel algorithm.
It takes more time to process a single input. Because every request can attack a master database only.	Strictly omit the delay factor in KSA by means distributed database with FAK system

Table 2: Difference between KSA and DDACA

Phase I

The different datasets and time study of KSA and DDACA based on the "Frequently Accessed Key (FAK)" system is shown in Table 3 and it can be transferred in to the bar chart representation of Fig 4. All the inputs are naturally exists in "FAK" subsystem.

The comparative time study shows DDACA follows stronger FAK system then the KSA.

Data Sets	KSA	DDACA
100 sets	2.38 ms	0.56 ms
1000 sets	2.7 ms	1.21 ms
10000 sets	3.51 ms	1.35 ms
100000 sets	5.72 ms	1.37 ms

Table 3: Phase I. Time study based on FAK

Phase II

The different datasets and time study based on the Distributed Database system is shown in Table 5 and it can be transferred in to the bar chart representation of Figure 6. In this analysis, we found lot of deviation between KSA and DDACA. Because KSA is quite delayed algorithm due to FAK (i.e.) it requests a master database only if a user request is not physically available in a FAK.

Master database only intakes if it is not in a busy state. Till busy state is cleared user request must wait in a queue. This algorithm checks only the local database for busy state. However, in our approach it checks all the databases participate in a network.

Data Sets	KSA	DDACA
100 sets	5.20 ms	1.76 ms
1000 sets	5.31 ms	1.92 ms
10000 sets	6.24 ms	2.16 ms
100000 sets	7.12 ms	2.22 ms

Table 5: Phase II.Time study based on distribute database

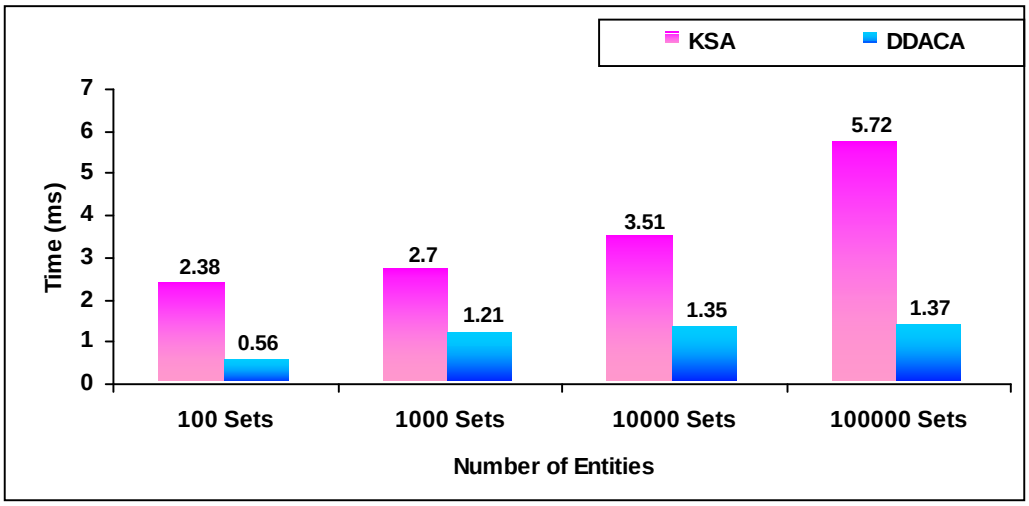


Fig 4: Time study based on FAK

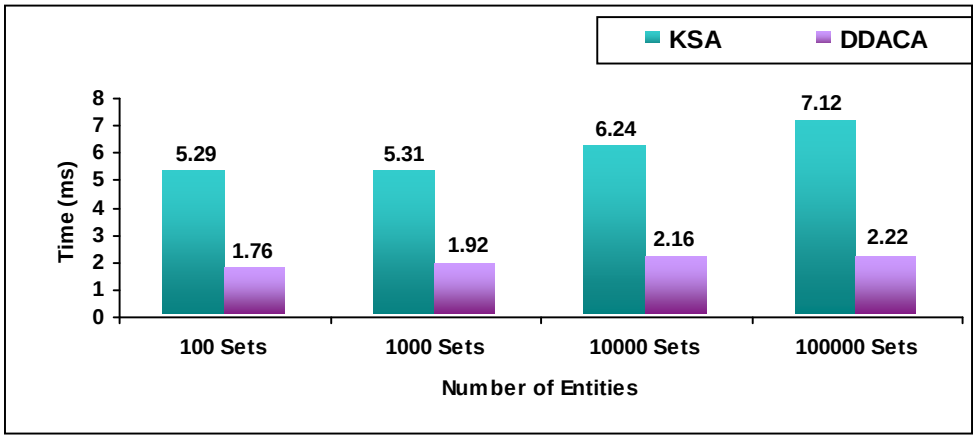


Fig 6: Time study based on distributed database

10. Performance Evaluation

The performance study has been divided into two phases to compute DDACA with KSA. As in phase I, and Phase II the experimental results show that the inputs are naturally exists in the 'FAK' Subsystem. The program was the only major job running on the machine throughout all the experiments to achieve a fair environment for comparison. The execution time of DDACA follows stronger FAK system than the KSA. The performance of DDACA and KSA in a large database is also compared. In KSA, checks only the local database for busy state. But in our approach, it checks all the databases participate in a network.

But, in our approach master database takes an input set simultaneously with FAK, therefore no delay in this approach. On the other hand, in KSA after failure message received from FAK it requests for searching process to master database.

11. Conclusion

In this paper, we introduced the distributed data access control algorithm for mining association rules. It is possible to mine globally valid results from distributed data without revealing information that compromises the individual sources. The major highlight of this algorithm is to distribute a user request to both the frequently accessed key as well as the master database which has a free station, the algorithm has been implemented on an experimental basis and its performance study shows that DDACA is superior over key semaphore algorithm.

References

[1] Jiawei Han, Micheline Kamber, "Data mining Concepts and Techniques", Harcourt India Private Limited ISBN: 81-7867-023-2.

[2] R.Epstein and M.R.Stonebraker, "Analysis of Distributed Database Processing Strategies", Proceedings of the International Conference on VLDB, 1988.

[3] R.Bagrodia,"ProcessSynchronization:Designand Performance Evaluation of Distributed Algorithms",IEEE Trans.on Software Engineering, Sep-1989.

[4] Jiawei Han, Yandong CAI and Nick Cercone,"Data Driven Discovery of Quantitative Rules in Relational Database", IEEE transactions on Knowledge and data Engineering, Vol.5, No.1, pp.29-40, February 1993.

[5] Heikki Mannila, Hannu Toivonen, and A.Inkeri Verkamo,"Efficient algorithms for discovering association rules", In KDD-94: AAAI Workshop on Knowledge Discovery in Databases, pages 181-192, Seattle, Washington, July 1994.

[6] Rakesh Agrawal and Ramakrishnan Srikant," Fast Algorithms for mining Association Rules", In Proc. of the 20th Int'l Conference on very large databases, Santiago, Chile, pp487-499, September 1994.

[7] A.Savasere, E.Omoecinski, and S.Navathe,"An efficient algorithm for mining association rules in large databases", In Proc. of Proc. of the VLDB Conference, Zurich, Switzerland, September 1995.

[8] Y. Fu and J. Han, V. Ng, A. Fu, and Y. Fu, "A Fast Distributed Algorithm for Mining Association Rules," Proc.of Int'l Conf. Parallel and Distributed Information Systems, PP. 31-44, Dec 1996.

[9] R. Agrawal and J.C. Shafer, "Parallel Mining of Association Rules: Design, Implementation, and Experience,"IEEE Trans. Knowledge and Data Engg, vol.8, pp. 962-969, 1996.

[10] Lamport. L., R.Shostak and M.Pease,"Time, Clocks, and the Ordering of events in a distributed System" Communications of the ACM, vol.21, no. 7(July): pp.558-565.1997.

[11] R.Bayardo,"Efficient Mining Long Patterns from Database", Proc.ACM Special Interest Group on Management of Data (SIGMOD), June 1998.

Dr.N Rajkumar obtained his Bachelor's degree in Computer Science and Engineering from Madurai Kamaraj University in 1991 and His Masters in Engg .the same stream in the 1995 from Jadavpur University. Kolkata. He has completed his Masters in Business Administration from IGNOU in the year 2003. His doctorate is in the field of Data Mining, which he completed in 2005 from PSG College of Technology, Coimbatore. He is currently the Head of the Department of Computer Science and Engineering at New Horizon College of Engineeering, Bangalore, Karnataka. He has served in the field of education for over 17 years at various Technical Institutions. He has been instrumental in the conduct of 30 Short- Term Courses and has also attended 20 courses conducted by other Institutions and Organizations. He has authored 2 books for the benefit of the Student Community in Networking and Computer Servicing. He has published as many as 30 papers, 3 in International Journals, 17 International Conferences and others at the National level in his area of expertise namely Data Mining, Networking and Parallel Computing respectively. He has guided 100-Project scholar's to-date.