Concurrency Control Algorithms and its Variants: A Survey

Neera Batra*, A.K Kapil**

* Deptt. of Computer Science & Engineering, M. M. Engineering College, Mullana-133203 Haryana, INDIA
** Institute of computer technology & Business Management,M. M. Engineering College, Mullana-133203 Haryana, INDIA
batraneera1@gmail.com, anil_kdk@yahoo.com

Abstract: This paper surveys many variants of concurrency control algorithms in database systems. We classify the different alternatives under locking, time-stamp, optimistic algorithms. Though the performance of different concurrency control algorithms have been explored extensively for database management systems but to the best of author's knowledge, the relative variants of different protocols used for concurrency control algorithms have not been reported yet.
Keywords: Distributed Database, Concurrency Control, Time Stamp Ordering, Optimistic Concurrency Control.

I. INTRODUCTION

A distributed database [1[[2][5] is a database that is under the control of a central database management system (DBMS) in which storage devices are not all attached to a common CPU. It may be stored in multiple computers located in the same physical location or may be dispersed over a network of interconnected computers.

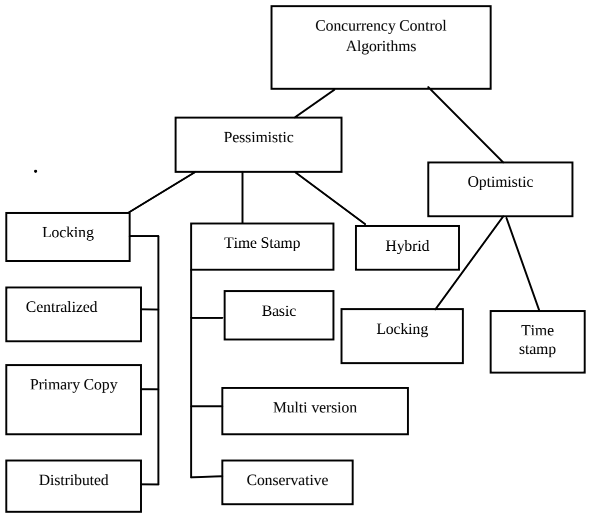

Figure 1. Classification of concurrency control

Concurrency control is an integral part of the database system. It is used to manage the concurrent execution of operations by different transactions on the same data item such that consistency is maintained. When multiple users access multiple database objects residing on multiple sites in a distributed database system, the problem of concurrency control arises. Figure 1 shows the classification of concurrency control.

The rest of the paper is organized as follows. Section II covers the literature review. Section III introduces the possible challenges in distributed database. Section IV presents the algorithms and its various variants. Section V is concerned with performance analysis of different variants and section VI is all about conclusion.

II. LITERATURE REVIEW

Concurrency control [1] in a distributed database has been the focus of research for many years. One research [4] uses divergence control lock model based on prudent order sharing and a check-out / check-in protocol supporting transactions disconnections. Mobile clients encounter wide variations in connectivity ranging from high-bandwidth, low latency communications through wired networks to total lack of connectivity. These attributes can be supported by using weak read and write operations which access only local and potentially inconsistent copies and perform updates using bounded inconsistency which is defined by allowing controlled variations among copies located at weakly connected sites [2]. Mobile transaction concurrency can be increased using semantic knowledge to relax the notion of absolute transaction atomicity by providing a high degree of inter-transaction parallelism [8]. Blocking problem can be reduced by attaching multiple back-up-sites to the coordinator site [7]. System performance can be improved by using the Data Caching. Caching stores desired data in the local storage of data processing node to improve data availability and data access time [11]. In mobile applications, synchronization architectures are required to provide the best speed, network traffic and storage. An approach for synchronization is deferred updates i.e. each client synchronizes with the server when an update is required. In other words, a view is updated only when a query is updated against it and the client is connected to the server. [6].

III. ALGORITHMS

A. Algorithms Based on Wait Mechanism

When two transactions conflict, one solution is to make one transaction wait until the other transaction has released the entities common to both. To implement this, the system can provide locks on the database entities. Transactions can get a lock on an entity from the system, keep it as long as the particular entity is being operated upon, and then give the lock back. If a transaction requests the system for a lock on an entity, and the lock has been given to some other transaction, the requesting transaction must wait.

B. Variations of Lock Algorithm

Writing in the backup site (using BC protocol):
In [7], one backup site is attached to each coordinator site. After receiving responses from all participants in the first phase, the coordinator communicates its decision only to its backup site in the backup phase. Afterwards, it sends final decision to participants. When blocking occurs due to the failure of the coordinator site, the participant sites consult coordinator's backup site and follow termination protocols. In this way, BC protocol achieves non-blocking property in most of the coordinator site failures. If both the coordinator and its backup site fail simultaneously, the participants wait until the recovery of either the coordinator site or the backup site. BC protocol has following merits. First, it eliminates the blocking of transactions in most of the coordinator failures. Second, it ensures consistency of the database in case of partitioning failures. And third, the performance of BC protocol is close to 2PC protocol.
Locking at Central Node, Execution at all Nodes (LCN Method): Instead of executing the transaction at the central node, we can only assign the locks at the central node and send the transaction back to node X. The transaction Ti is executed at node X. The values of the read set are read, and the values of the write set are obtained at node X. Node X sends the values of the write set and obtains acknowledgments from all other nodes. It then knows that transaction Ti has been completed. The node X sends a message to unlock entities referenced by Ti. The central node after receiving this message releases the locks and starts assigning locks to waiting transactions.
Global Two-phase Locking (G2PL): In G2PL, Instead of a transaction getting all locks in the beginning and releasing all locks in the end, the policy of two phase locking is employed. Each transaction obtains the necessary locks as they are needed and then releases locks on entities that are no longer needed. A transaction cannot get a lock after it has released any lock.
Divergence control lock model based on prudent ordered sharing and a Check-Out/Check-In protocol supporting disconnections (DC/POS-PAI-2PL) [4]: DC/POS-PAI-2PL, a two-phase lock strategy reduces blocking in concurrency control for mobile real-time transactions. This protocol is meant for the flat transaction model and based on the serializability. In this strategy, a Check-Out/Check-In protocol is developed which can support disconnected transactions to continue their executions after a fixed server has received a "Check Out" message from a mobile host, it creates an "image transaction" to replace the transaction in the mobile host to check out data.
S2PL: A Secure multilevel database based on secure 2PC protocol [5] in which two security levels: high and low are used. A primary concern in multilevel security is information leakage by a high security level transaction to a transaction executing at a low security level. Leakage can occur in two ways: directly through an overt operation such as reading a data item or indirectly through a covert channel. Direct leakage can be prevented by mandatory access control policies such as the Bell-La Padula (BL) model [5] but handling of covert channel needs modifications in conventional concurrency control schemes such as two-phase locking (2PL) and timestamp ordering (TO).
Primary Copy Locking (PCL): In this variant as in [3], instead of selecting a node as the central controller, a copy of each entity on any node is designated as the primary copy of the entity. A transaction must obtain the lock on the primary copy of all entities referenced by it. At any given time, the primary copy contains the most up-to -date value for that entity.
Weaker Consistency: In the proposed scheme, data located at strongly connected sites are grouped together to form clusters. Mutual consistency is required for copies located at the same cluster, while degrees of inconsistency are tolerated for copies at different clusters. The interface offered by the database management system is enhanced with operations providing weaker consistency guarantees. Such weak operations allow access to locally, i.e., in a cluster, available data. Weak reads access bounded

inconsistent copies and weak writes make conditional updates

C. Algorithms Based on Time-Stamp Mechanism

Timestamp is a mechanism in which the serialization order is selected a priori; the transaction execution is obliged to obey this order. In timestamp ordering, each transaction is assigned a unique timestamp by the scheduler or concurrency controller. Obviously, to achieve unique timestamps for transactions arriving at different nodes of a distributed system, all clocks at all nodes must be synchronized or else two identical timestamps must be Resolved.

Clock Synchronization by message passing (CSM Method): Lamport [8] has described an algorithm to synchronize distributed clocks via message passing. If a message arrives at a local node from a remote node with a higher timestamp, it is assumed that the local clock is slow or behind. The local clock is incremented to the timestamp of the recently received message. In this way all clocks are advanced until they are synchronized. When the operations of two transactions conflict, they are required to be processed in timestamp order. It is easy to prove that timestamp ordering (TSO) produces Serializable histories.

CSM Correctness Criteria: Thomas [9] has studied the correctness and implementation of [8] approach and described it. Essentially each node processes conflicting operations in timestamp about its direct predecessor only. Each read-write conflict relation and write-write conflict relation is resolved by timestamp order. Consequently all paths in the relation are in timestamp order and, since all transactions have unique timestamps, it follows that no cycles are possible in a graph representing transaction histories.

OSN Method: OSN [10] increases the level of concurrent execution of transactions, called ordering by serialization numbers. The OSN method works in the certifier mode and uses time interval technique in conjunction with short-term locks to provide serializability. The scheduler is distributed and the standard transaction execution policy is assumed. However, the write operations are copied into the database only when the transaction commits. The amount of concurrency provided by the OSN method is demonstrated by log classification. A performance analysis realized through simulation showed that the OSN method performs better than the basic timestamp ordering

Deferred Sync: In mobile databases , Deferred Sync [6] converts relational data into XML tree structure and then makes use of deferred views in order to minimize bandwidth and storage space for the client by using a conflict detection and resolution algorithm based on priority selection to synchronize the data. Using a view, the client can retrieve only the data relevant to them instead of transferring the entire database. In order to resolve conflicts, a combination of priority selection and latest timestamp ordering is used. Deferred sync makes use of the view in XML to coordinate the synchronization between devices.

Timestamp Ordering with Transaction Classes:
This mechanism was used in the development of a prototype distributed database management system called SDD-1, developed by the Computer Corporation of America [1]. SDD-1, The system employs a series of four synchronization protocols which vary in cost and provide varying levels of synchronization control. Different transactions must be executed using different protocols depending upon the fact which protocol is meant for which transaction class .The decision as to which transactions must use each protocol, is based on a formal mathematical analysis of the ways in which transactions in a distributed database system can interfere with each other.

D. Algorithms Based on Rollback Mechanisms

Concurrency control schemes are typically categorized as follows: pessimistic control and optimistic control. The pessimistic concurrency control is vulnerable to deadlock.
In optimistic approach, the idea is to validate a transaction against a set of previously committed transactions. If the validation fails, the read set of the transaction is updated and the transaction repeats its computation and again tries for validation. The validation phase will use conflicts among the read sets and the write sets along with certain timestamp information.

Optimistic concurrency control (OCC Method): Optimistic concurrency control with cache coherency control- Tae-Young Cheo [11] combined cache coherency control and optimistic concurrency control rather than they operate in separate. According how many caches retain its activity, concurrency controls are classified as direct validation scheme and reduced validation scheme. Direct validation scheme preserves activity of cache, and it invalidates transaction which tries to accesses invalidated objects again using GSO algorithm. Reduced validation scheme uses 'reduction of I/O frequency' of cache, and reduces the number of access for an object to one or two using GMS, GMM and LMM algorithms. In the case that the access number is one, the transaction is valid. In case the access number is two, the object must not be modified between the accesses.

E. Performance Analysis

Based on all the algorithms surveyed in this paper, here is the classification of all variants under three main approaches for concurrency control: Locking, Time Stamp and Optimistic.

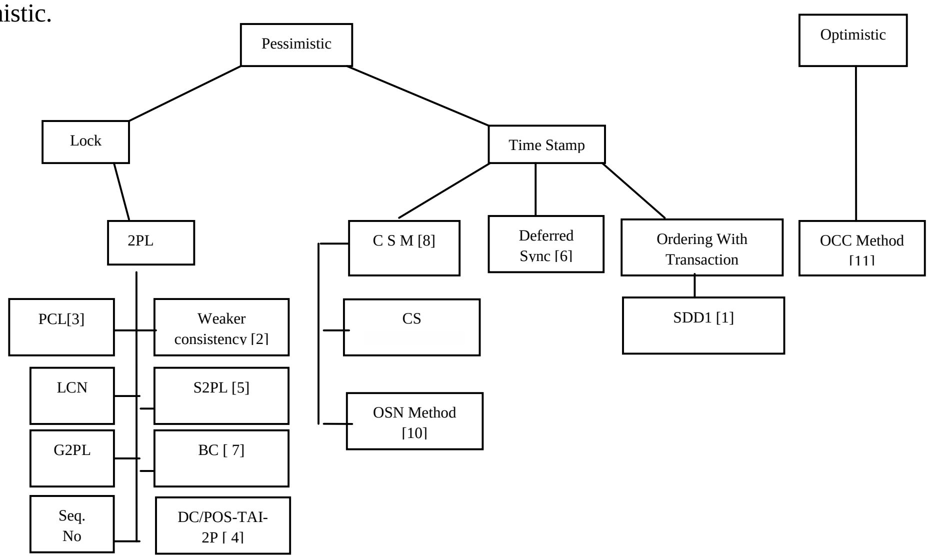

Figure 2. (Different variants of 2PL, Time Stamp and Optimistic concurrency control **approaches)**

Table 1. Variants and their respective properties

Sr. No.	Protocol	Eliminating blocking	Consistency assured	Load Balancing	Increased Efficiency	Reduced Overhead	Flexibility	Security	Deadlock prevention	Lesser comm. cost
1.	BC [7]	✓	✓						✓	
2.	LCN		✓	✓		✓			✓	
3.	Seq. no.				✓	✓				✓
4.	G2PL		✓				✓			
5.	DC/POS-PAI-2PL [4]	✓	✓		✓					
6.	S2PL [5]		✓					✓		
7.	PCL[3]		✓		✓				✓	
8.	Weaker consistency [2]	✓					✓		✓	
9.	CSM [8]	✓	✓		✓				✓	
10.	CSM Correctness[9]	✓	✓						✓	
11.	OSN Method[10]		✓		✓					✓
12.	Deferred Sync[6]	✓		✓	✓			✓		✓
13.	SDD-1[1]			✓	✓				✓	
14.	OCC Method[11]	✓	✓		✓	✓				✓

Here is a table with all the variants and their properties which uniquely identify them and differentiates them from all the other algorithms. As each application has different environment under which it runs, different execution requirements, speed and security requirements, depending upon the requirement of the application, the most suitable variant can be chosen for concurrency control. It also varies from application to application how much efficiency, load balancing, deadlock prevention and frequency of disconnections need to be monitored.

IV. CONCLUSION

In this paper, comparative study among different variants of lock algorithm, time stamp ordering and optimistic concurrency control algorithms which have been implemented over last 20 years in distributed, mobile databases have been done based on various chosen parameters like reduced blocking, consistency, load balancing, efficiency, security etc. as mentioned in table 1. Depending upon application's requirement and resources available for a system, suitability of a particular variant to be used can be decided for a specific environment.

REFERENCES

[1] Philip A. Bernstein, James B. Rothnie, JR., Nathan GoodMan, And Christos A. PapaDimitriou," The Concurrency Control Mechanism of SDD-1: A System for Distributed Databases", In IEEE Transactions On Software Engineering, VOL. SE-4, , No. 3, pp 1-15, MAY 1978.

[2] Evaggelia Pitoura and Bharat Bhargava, "Data Consistency in Intermittently Connected Distributed Systems" In IEEE Transactions on Knowledge and Data Engineering, VOL.11, NO. 6, pp.896-915, November/December 1999.

[3] Bharat Bhargava, "Concurrency Control In Database Systems", In IEEE Transactions on Knowledge And Data Engineering, VOL. 11, No. 1, pp. 03-16, January/February 1999.

[4] Guo Quiong Liao, YunSheng Liu, LiNa Wang, ChuJi Peng, " Concurrency control of Real-time Transactions with Disconnections in Mobile Computing Environment", Proc. Of the 2003 International Conference on computer Networks and Mobile Computing (ICCNMC"03) February, 2003.

[5] Navdeep Kaur, Rajwinder Singh, A.K. sarje and Manoj Misra, "Performance Evaluation of Secure Concurrency Control Algorithm for Multilevel Secure Distributed Database" , IEEE, pp. 186-191, April 2004.

[6] Kevin miller, Chris Gee, Ryan Inaba, Tansel Ozyer, Anthony Lo, Reda Alhajj "Synchronization of Mobile XML Databases by Utilizing Deffered Views", IEEE , pp. 186-192, 2004.

[7] P. Krishna Reddy and Masaru Kitsuregawa, "Reducing the blocking in two-phase commit with backup sites: University of Tokyo, Japan, pp. 01-09, August, 2002.

[8] L. Lamport, "Time Clocks and the Ordering of Events in a Distributed System," Comm. ACM, vol. 21, no. 2, 1979.

[9] R.H. Thomas, "A Majority Consensus Approach to Concurrency Control for Multiple Copy Systems," Trans. Database Systems, ACM, vol. 4, no. 2, pp. 180-209, 1979.

[10] Ugur Halici and Asuman Dogac, "Concurrency Control in Distributed Databases through Time Intervals and Short-Term Locks" IEEE Transactions on Software Engineering VOL. 15. NO. 8. pp. 1-10, august 1989.

[11] Tae-Young Choe, "Optimistic Concurrency Control based on Cache Coherency in Distributed Database Systems" Kumoh National Institute of Technology 1, YangHo Dong, Gumi, Korea, IJCSNS (International Journal of Computer Science and Network Security), VOL.8 No.11, pp. 1-7 November 2008.