Global Journal of Computer Science and Technology
Volume 11 Issue 23 Version 1.0 December 2011
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

An Analytical Review of Orientation Based Concurrency Control Algorithm

By Sumit Kumar, Ms. Ritu Devi
M M University, India

Abstract - There is an ever-increasing demand for higher throughputs in transaction processing systems leading to higher degrees of transaction concurrency.Concurrency control in Database management systems ensures that database transactions are performed concurrently without violating the data integrity of the database. Thus concurrency control is an essential element for correctness in any system where two database transactions or more, executed with time overlap, can access the same data. There are problems like Deadlock,Livelock and prevention of these problems is vital in concurrency control of distributed database systems.Many techniques have been proposed for managing concurrent execution of transactions in database systems.A new method for concurrency control in distributed DBMS's,is discussed which will improve system performance by reducing the chances of deadlock and livelock and reducing restart ratio.

Keywords : *concurrency,deadlock,timestamp,lock etc.*

GJCST Classification : *D.4.1*

Strictly as per the compliance and regulations of:

An Analytical Review of Orientation Based Concurrency Control Algorithm

Sumit Kumar [α], Ms. Ritu Devi [Ω]

Abstract - There is an ever-increasing demand for higher throughputs in transaction processing systems leading to higher degrees of transaction concurrency.Concurrency control in Database management systems ensures that database transactions are performed concurrently without violating the data integrity of the database. Thus concurrency control is an essential element for correctness in any system where two database transactions or more, executed with time overlap, can access the same data. There are problems like Deadlock,Livelock and prevention of these problems is vital in concurrency control of distributed database systems.Many techniques have been proposed for managing concurrent execution of transactions in database systems.A new method for concurrency control in distributed DBMS's,is discussed which will improve system performance by reducing the chances of deadlock and livelock and reducing restart ratio.

Keywords : concurrency,deadlock,timestamp,lock etc.

I. Introduction

Concurrency control is the activity of coordinating concurrent accesses to a database in a multiuser database management system (DBMS). Concurrency control permits users to access a database in a multiprogrammed fashion while preserving the illusion that each user is executing alone ona dedicated system[2]. The main technical difficulty in attaining this goal is to prevent database updates performed by one user from interfering with database retrievals and updates performed by another. The concurrency control problem is exacerbated in a distributed DBMS (DDBMS) because (1) users may access data stored in many different computers in a distributed system, and (2) a concurrency control mechanism at one computer cannot instantaneously know about interactions at other computers.

II. Background of Concurrency Control Methods

Many methods for concurrency control exist[1] [4][5][6][8][9][10].The major methods, which have each many variants,are:

1. Locking - Locking is a mechanism commonly used to solve the problem of synchronizing access to shared data[6].Controlling access to data by locks assigned to the data.Several types of locks are used in concurrency control such as Binary(1 or 0) locks,Shared/Exclusive locks. each data item has a lock associated with it. Before a transaction T, may access a data item, the scheduler first examines the associated lock. If no transaction holds the lock, then the scheduler obtains the lock on behalf of T,. If another transaction T, does hold the lock, then T, has to wait until T2 gives up the lock. That is, the scheduler will not give T, the lock until T releases it. The scheduler thereby ensures that only one transaction can hold the lock at a time, so only one transaction can access the data item at a time.When a lock is set, other transactions that need to set a conflicting lock are blocked until the lock is released, usually when the transaction is completed. The more transactions that are running concurrently, the greater the probability that transactions will be blocked, leading to reduced throughput and increased response times.One variation of basic locking protocol that ensure serializability is two phase locking protocol[10].This protocol requires that each transaction issue lock and unlock requests in two phases:

1. Growing phase – A transaction may obtain locks,but may not release lock.
2. Shrinking phase – A transaction may release locks,but may not obtain any new locks.

A method called optimistic method with dummy locks is also there for concurrency control in distributed databases. The advantage of using dummy locks is that although they are long-term locks, they do not block the execution of transactions in any way[]

2. Serialization graph checking (also called Serializability, or Conflict, or Precedence graph checking) – Although two phase locking ensure serializability,they may lead to a deadlock.Deadlock occurs when each transaction T in a set of two or more transaction is waiting for some item that is loked by some other transaction T1 in the set. There are otherways one could enforce serializability as well.Deadlock can be precisely detected by constructing a directed graph called wait-for-graph.The nodes of WFG are labelled with active transaction names.In a WFG there exist an edge from Ti to Tj iff transaction Ti is waiting for transaction Tj to release some lock.Is there exist a cycle in WFG,it means deadlock has occure and broken by aborting a transaction.The transaction chosen for abort is called the victim. While such a scheme is possible, it is hardly practical.

Author [α] : Computer Science & Engg. Department, M M University, India. E-mail : sumit0709@gmail.com
Author [Ω] : Computer Science & Engg. Department, M M University, India. E-mail : ritudevi@rediffmail.com

3. Timestamp ordering (TO) – In an alternative approach to locking is use of timestamps[4][10]. Ordered timestamps are assigned to transactions, and controlling or checking access to data by timestamp order. The general idea is to give each transaction a "timestamp" which indicates when the transaction began (serial number or system time). To generate timestamp values,transaction manager can use system clock value i.e TS(T) is equal to value of clock when T has entered the system.Alternatively,the transaction manager can use a counter that is incremented after a new timestamp has been assigned.To implement this scheme,the timestamp ordering algorithm associates with each data item X two timestamp values:

A. write_TS(X) – the maximum timestamp value of a transaction that successfully executed write_item(X).
B. read_TS(X) – the maximum timestamp value of a transaction that successfully executed read_item(X).

1. When T tries to write(X)
   - if Read_TS(X) > TS(T) or Write_TS(S) > TS(T)
     Intuition: X has been read or written by a "later" transaction
   - Abort T
     else
   - Execute and set write-TS(X) = TS(T)
2. When T tries to read(X)
   - if Write_TS(X) > TS(S)
     X was written by a "later" transaction
   - Abort T
     else
   - Execute and update read-TS(X)

III. Rules for a Database Transaction

A database transaction is a unit of work, typically encapsulating a number of operations over a database (e.g., reading a database object, writing, acquiring lock, etc.).Every database transaction obeys the following rules:

- Atomicity - Either the effects of all or none of its operations remain ("all or nothing") when a transaction is completed (committed or aborted respectively). In other words, to the outside world a committed transaction appears (by its effects on the database) to be indivisible, atomic, and an aborted transaction does not leave effects on the database at all, as if never existed.
- Consistency - Every transaction must leave the database in a consistent (correct) state.A transaction must transform a database from one consistent state to another consistent state. Thus since a database can be normally changed only by transactions, all the database's states are consistent. An aborted transaction does not change the database state it has started from, as if it never existed (atomicity above).
- Isolation - Transactions cannot interfere with each other.Moreover, usually (depending on concurrency control method) the effects of an incomplete transaction are not even visible to another transaction. Providing isolation is the main goal of concurrency control.
- Durability - Effects of successful (committed) transactions must persist through crashes (typically by recording the transaction's effects and its commit event in a non-volatile memory).

IV. Requirements for Database Transcation

Every database transaction should fullfill following requirements:

- Safety Property: The safety property states that at any point of time, only one transaction can access the data.
- Liveness Property: This property states the absence of deadlock and starvation. Two or more transactions should not endlessly wait for a particular object which will never arrive. In addition, a transaction must not wait indefinitely to access an object while other transactions are repeatedly acquiring the same.
- Fairness: Fairness property states that each transaction should get chance to access an object. In concurrency control algorithms, the fairness property generally means the requests are executed in the order of their arrival (time is determined by a logical clock) in the system.

V. Need for Concurrency Control

If transactions are executed serially, i.e. sequentially with no overlap in time, no transaction concurrency control required.However if concurrent transactions with interleaving operations are allowed in an uncontrolled manner, some unexpected, undesirable result may occur. Here are some typical examples:

1. The lost update problem: when a transaction writes a new value of a data-item on top of a first value written by a first concurrent transaction, and the first value is lost to other transactions running concurrently which need to read the first value.
2. The dirty read problem: when Transactions read a value written by a transaction that has been later aborted. This value disappears from the database upon abort, and should not have been read by any transaction ("dirty read"). The reading transactions end with incorrect results.
3. The incorrect summary problem: While one transaction takes a summary over the values of all the instances of a repeated data-item, a second transaction updates some instances of that data-

item. The resulting summary does not reflect a correct result for any precedence order between the two transactions (if one is executed before the other), but rather some random result, depending on the timing of the updates, and whether certain update results have been included in the summary or not.

VI. Review of Timestamp and Orientation Based Currency Control Algorithm

In the concept of timestamp ordering[4][7], transaction timestamp TS(T) is a unique identifier assigned to each transaction based on the order in which transaction are started.Hence if transaction T_1 starts before transaction T_2 then TS(T1)<TS(T2).There are two method for preventing deadlock using the concept of timestamp ordering:

a. Wait-die: suppose that transaction T1 wants to lock an item X but is not able to do so because X is locked by some other transaction T2 with a conflicting lock.Now if TS(T1)<TS(T2).Then T1 is allowed to wait,otherwise abort T1 and restart it later with the same timestamp.
b. Wound-wait: if TS(T1)<TS(T2) then abort T2 and restart it later with the same time stamp;otherwise T1 is allowed to wait.

In wait-die protocol, only the requester with smaller timestamp can wait for the holder with larger timestamp and in the wound-wait protocol, only the requester with larger timestamp can wait for the holder transaction with smaller timestamp. The constraints of these protocols are so strong that only one-way waiting is allowed.Algorithm based on orientation will try to make the condition somehow weaker. This algorithm allows both side waiting i.e the older waits for the younger (as wait-die protocol) and younger waits for the older (as wound-wait protocol).In the reviewedalgorithm, a new term is introduced which is called as orientation of a transaction.It uses combination of time stamp and orientation to decide which transaction will wait and which transaction will be wounded when conflict exists among transactions. An orientation of a transaction T, denoted as Ot(T),can have three values:neutral, forward, and backward. Following are the orientation determination rules for the system:

*Rule 1:*The initial orientation of a transaction is 'n'.

Rule 2: When Tr requests for Th, if TS(Th)> TS(Tr) and Tr can waitfor Th, then Ot(Tr):= Ot(Th):= 'f'. We call this kind of waiting as forwardwaiting.

Rule 3: When Tr requests for Th, if TS(Th)< TS(Tr) and Tr can waitfor Th, then Ot(Tr).'= Ot(Th):= 'b'. We call this kind of waiting as backwardwaiting.

Rule 4: When Tr requests for Th, but Tr is not allowed to wait for Th,then one of them may be rolled back and restarted (the rolled-backtransaction is always the younger). The time stamp of restarted transactiondoes not change but its orientation is changed to 'n'.This algorithm based on orientation minimizes no. of restarts than other standard algorithm.

VII. Conclusion and Future Work

Standard wait die and wound wait only logically keep forward or backward orientation WFG, respectively, in its protocol. But in the algorithm based on orientation it keeps both backward and forward orientation WFG in the protocol. More importantly, it is not necessary to physically maintain any WFG in the system.The new algorithm is deadlock free and livelock free.This algorithm will require much fewer restarts than standard wait-die or wound-wait protocol and thus will achieve high throughout and efficiency of distributed database system.There is still a issue to research as future work,after finding a transaction conflicting withanother transaction how much time should have to wait torestart the aborted transaction. If it is restarted very soon thereremains probability to conflict again. On the other hand if thetransaction is restarted after some period of time the abortedtransaction, especially if is it a real time one, may fail to meetits deadline.

References Références Referencias

1. K. P. Eswaran, J. N. Gray, R. A. Lorie, and I. L. Traiger, "The notions of consistence and predicate locks in a database system,"Commun. ACM, vol. 19, no. 11, pp. 624-633, Nov. 1976.
2. A. Bernstein and Nathan Goodman,"Concurrency Control in Distributed Database Systems", *Computing Surveys, Vol. 13, No 2, June 1981.*
3. Ryan, R. R. Spiller, H, "The C programming language and a C compiler", IBM Systems Journal Vol: 24 Issue: 1,No. 37 – 48,April 1985.
4. Victor 0. K. LI, "Performance Models of Timestamp-Ordering Concurrency Control Algorithms in Distributed Databases" IEEE Trans. on Computers, Vol. C-36, No. 9, September 1987.
5. J.F.Pons and J.F. Vilarema, "A Dynamic and Integrated Concurrency Control for Distributed Databases"IEEE Journal on Selected Areas in Comm. Vol. 7, No. 3, April 1989.
6. Ugur Halici & Asuman Dogac," An Optimistic Locking Technique For Concurrency Control in Distributed Databases", IEEE Trans.on Software Engineering, vol. 17, no. 7. july 1991.
7. Subir Varma,"Performance Evaluation of the TimestampOrdering Algorithm in a Distributed Database"IEEE Trans. on Parallel and Distributed Systems, vol. 4, no. 6, June 1993.
8. F. Bukhari and Sylvia L. Osborn,"Two Fully Distributed Concurrency Control Algorithms" IEEE Trans. on Knowledge and Data Engineering, vol. 5, no. 5, October 1993.
9. Alexander Thomasian,"Distributed Optimistic Concurrency Control Methods for High-Performance

Transaction Processing" IEEE Trans. on Knowledge and Data Engineering, Vol. 10, No. 1, January/February 1998.

10. Philip .Alexander Thomasian,"Concurrency Control: Methods, Performance, and Analysis" ACM Computing Surveys (CSUR) Survey Volume 30 Issue 1, March 1998. .