Timestamp-Ordering Protocol for Concurrent Transactions - A Performance Study

Sunil Dhondu Mone
Asst. Professor,
R. C. Patel Art, Com and Sci.
College,Shirpur, Dist.- Dhule.(M.S.)

ABSTRACT
When we have multiple transactions to execute at an instance, one can prefer to execute them concurrently, which result in improve throughput, improve resource utilization, reduce waiting time, and also help to reduce average response time. While transaction get executed concurrently, in get facilitate by concurrency schemes, numerous concurrency schemes has been proposed. The minimum criteria they are expected to fulfill are like serializability. To test performance of various concurrency schemes various criteria may be consider. In this paper I am considering time stamp ordering protocol and criteria like conflict serializability, view serializability, recoverability, and cascadeless, deadlock, level of concurrency support.

Keywords
Consistency. Serializability, conflict serializability, view serializability, recoverability, cascadeless,

1. INTRODUCTION
Transaction is a sequence of instructions. While transaction get execute it access as well as update database. While transaction update database it access and store data item in temporary storage area and perform updating in temporary area and update database later.

While we have multiple transactions to execute at an instance, one can prefer to execute all multiple transactions concurrently, Concurrent execution of transaction have many advantages; few notable advantages are as follows.

1) it improve throughput : A transaction consists of may steps. Some involve I/O operation, while some other involve CPU operation. CPU and I/O operations may perform parallel, ultimately multiple transaction may execute concurrently, in same amount of time multiple transaction get execute and help to improve throughput.(Abraham Silberrschatz, 2006)

2) Improve utilization: As multiple transaction get execute concurrently, and may utilize multiple resources parallel. Moreover compare to I/O and CPU speed, CPU speed may utilize properly.(Abraham Silberrschatz, 2006)

3) Reduce waiting time: Conversely if transaction get execute sequentially waiting time of transaction get execute at last will be high, which shall be reduce in concurrent execution of transactions.

4) Increase response time: As transactions get execute Concurrently, average response time of all concurrently executing transaction may get increase.(Abraham Silberrschatz, 2006)

5) Multiprogramming: As concurrent execution of transaction utilize resources in parallel offer multiprogramming concept.

While transactions get execute concurrently, they are expected to fulfill criteria serializability. Since serializability offers transaction isolation, Atomicity, as well as data consistency and data durability. It is well known as ACID property of transaction. Serializability may be conflict serializability, view serializability.

Numerous concurrency schemes have been proposed, In this paper I am considering only Timestamp-Ordering Protocol. We can consider criteria to test performance of concurrency schemes base on conflict serializability, view serializability, recoverability, cascadeless, level of concurrency offers.

2. PERFORMANCE TESTING CRITERIA

2.1 Serializability
Any concurrent schedule (which consist more than one transactions executing concurrently) is said to be serialize schedule if final effect on database is same as sequential execution of same transactions involve in schedule.

2.2 Conflict Serializability
Let us consider a schedule S in which there are two consecutive instructions, Ii and Ij, of transactions Ti and Tj, respectively (i<>j). If Ii and Ij refer to different data items, when we can swap Ii and Ij without affecting the result of schedule. We say that Ii and Ij conflict if they are operations by different transactions on same data item, and at least one of them is a write operation.

If a schedule S can be transformed into schedule S' by a series of swaps of non-conflicting instructions, we say that S and S' are conflict equivalent. The concept of conflict equivalence leads to the concept of conflict serializability. (Gehrke)

2.3 View Serializability
The schedules S and S' are said to be view equivalent if following conditions are met:

1. For any data item Q, if Ti reads the initial values of Q in S, then transaction Ti in schedule S', also read initial value of Q.
2. For each data item Q, in schedule S if transaction Ti executes read(Q) and Tj executes write(Q), then in S' must also Ti executes read(Q) and Tj executes write(Q).
3. For each data item Q if any transaction Ti perform final write operation in S, then in schedule S' also Ti perform final write.

We say that schedule S is view serializable if it is view equivalent to a serial schedule.

Every conflict-serializable schedule is also view serializable, but there are view-serializable schedule that are not conflict serializable, which content blind writes.(Abraham Silberrschatz, 2006)

2.4 Recoverable

Consider following schedule

T1	T2
Read(A)	
Write(A)	
	Read(A)
Read(B)	

In which commit operation of T2 appear immediately after read(A) this is example of non-recoverable schedule, which should not be allowed. Database system require that all schedule be recoverable. A recoverable schedule is a schedule in which, for each pair of transaction Ti and Tj such that Tj reads a data item previously written by Ti the commit operation of Ti appears before the commit operation of Tj. (Abraham Silberrschatz, 2006)

2.5 Cascadeless

Consider following schedule

T1	T2	T3
Read(A)		
Read(B)		
Write(A)		
	Read(A)	
	Write(A)	
		Read(A)

Suppose that, T1 fails, T1 must be rolled back. Since T2 is dependents on T1, T2 must rolled back. Since T3 dependents on T2, T3 must be roll back. This scenario, in which single transaction failure leads to a series of transaction rollbacks, is called cascading rollback. Cascading rollback is undesirable, since it leads to undoing significant amount of work. It is desirable to restrict schedule for cascading rollback. Such schedule is call cascadeless schedule.

Cascadeless schedule is one where, for each pair of transaction Ti and Tj such that Tj reads a data item previously written by Ti, the commit operation of Ti appears before read operation of Tj. Note that every cascadeless schedule is also recoverable. (Abraham Silberrschatz, 2006)

3. TIMESTAMP-BASED PROTOCOLS

These types of protocols provide another method of determining the serializability order. That is providing order among transaction in advanced. Method used to do so is use a timestamp-ordering scheme.

3.1 Timestamps

With each transaction T in the system, we associate a unique fixed timestamp, denoted by TS(Ti). This timestamp is assigned by the database system before the transaction Ti starts execution. If a transaction Ti has been assigned timestamp TS(Ti), and a new transaction Tj enters the system, then TS(Ti) < TS(Tj). There are two simple methods for implementing this scheme:

1. Use the value of the *system clock* as the timestamp; that is, a transaction's time- stamp is equal to the value of the clock when the transaction enters the system.

2. Use a logical counter that is incremented after a new timestamp has been assigned; that is, a transaction's timestamp is equal to the value of the counter when the transaction enters the system.

The timestamps of the transactions determine the serializability order. Thus, if TS(Ti) < *TS(Tj),* then the system must ensure that the produced schedule is equivalent to a serial schedule in which transaction Ti appears before transaction *Tj.*

To implement this scheme, we associate with each data item Q two timestamp values:

- W-timestamp(Q) denotes the largest timestamp of any transaction that executed write(Q) successfully.
- R-timestamp(Q) denotes the largest timestamp of any transaction that executed read(Q) successfully.

These timestamps are updated whenever a new read(Q) or write(Q) instruction is executed.(Abraham Silberrschatz, 2006)

3.2 The Timestamp-Ordering Protocol

The timestamp – ordering protocol ensures that any conflicting read and write operations are executed in timestamp order. This protocol operates as follows.

1. Suppose that transaction Ti issue read(Q)

   a. If TS(Ti) < W-timestamp(Q), then Ti need to read a value of Q that was already overwritten. Hence, the read operation is rejected, and Ti is rolled back.

   b. If TS(Ti) >= W-timestamp(Q), then the read operation is executed, and R-timestamp(Q) is set to the maximum of R-timestamp(Q) and$TS(T_i)$.

2. Suppose that transaction Ti issues Write (Q).

   a. If TS (Ti) < R-timestamp (Q), then the value ofQ thatT_i is producing was needed previously, and the system assumed that value would never be produced. Hence, the system rejects the write operation and rolls Ti back.

   b. If TS(Ti) < W-timestamp(Q), thenTi is attempting to write an obsolete value of Q. Hence, the system rejects this write operation and rolls Ti back.

   c. Otherwise, the system executes the write operation and sets

W-time- stamp(Q) to TS(Ti).

If a transaction Ti is rolled back by the concurrency-control scheme as result of issuance of either a read or writes

operation, the system assigns it a new timestamp and restarts it.(Abraham Silberrschatz, 2006)

To illustrate this protocol, we consider transactions T1 and T2. Transaction T1 displays the contents of accounts A and B.

T1: Read(B)
Read(A)
Display(A+B)

Transaction T2 transfers Rs50 from account B to account A, and display contain of both.

T2: Read (B)
B= B – 50
Write (B)
Read (A)
A= A+ 50
Write (A)
Display (A+B)

Let us assume that timestamp will be assign immediately before its first instruction. Consider following schedule in which TS(T1) < TS(T2) and schedule is possible under timestamp protocol.

T1	T2
read(B)	read(B)
	b=b – 50
	write(B)
read(A)	
	read(A)
display(A + B)	
	A = A + 50
	write(A)
	display(A+ B)

(Abraham Silberrschatz, 2006)

4. ANALYSIS

4.1 Advantages

1) The timestamp ordering protocol ensures conflict serializability. This is because conflicting operations are processed in timestamp order.

2) The timestamp based protocol ensures freedom from deadlock, since no transaction ever waits.(Gehrke)

4.2 Disadvantages

1) There is possibility of starvation of long transactions. This may cause due to sequence of conflicting short transactions causes repeated restarting of the long transaction. One can avoid this by finding transaction, which is restarts repeatedly, and conflicting transaction need to be temporarily blocked enable the transaction to finish.

2) The timestamp based protocol may produce schedule which is not recoverable. But one can make timestamp based schedule recoverable using one of the following several ways.

   a) Recoverability and cascadelessness can be ensured by performing all write together at the end of transaction. And write must be atomic, that is while one transaction writing a data item, no one transaction allows to read/access.

   b) Recoverability and cascadelessness can also be guaranteed by using limited form of locking, as well as reads of uncommitted items are postponed until the transaction that updated the item commits.

   c) Recoverability only can ensure by tracking uncommitted write, and allowing a transaction Ti to commit only after the commit of any transaction that wrote a value, read by Ti. That is committing dependencies.(Abraham Silberrschatz, 2006)

5. CONCLUSION

- Serializability is a minimum criteria for concurrent transactions scheme.
-Cascadeless and deadlock is inversely proportional to concurrency.

- To get more concurrency, schedule need to allow for deadlock, and cascading rollback up to certain extend,

- Time stamp protocol ensures no deadlock, since data items are locked prior to execution of transaction.

6. REFERENCES

[1] Abraham Silberrschatz, H. F. (2006). Database System Concepts Fifth Edition. NY 10020: McGrawHill.

[2] Akil Kumar, M. S. (n.d.). Performance Evaluation of Operating System Transaction Manager. University of California, Berkeley, Ca 94720.

[3] Date. (n.d.). An Introduction to Database System, 7th Edition. Pearson Education.

[4] Gehrke, R. R. (n.d.). Database Management System, Third Edition. McGraw Hill.

[5] Ramez Elmasri, S. B. (2007). Fundamental of Database System, Fifth Edition. Pear Publication.

[6] RL, L. S. (1996). A multigranularity locking model of concurrency control in OODB. IEEE, Transaction on Knowlede and Data Engg. 8(1), 144-156.

[7] Stonebraker, M. J. (n.d.). The performance of Concurrency Control Algorithms fo Database Management Systems. Singapore: Proceddings of Tenth International Conference on Vary large Databse.