ADVANCED OPERATING SYSTEMS

MINU MARIA JOSEPH

Module 5 [7 Hours]

Database Systems: Requirements of a Database Operating System, Problem of Concurrency Control, Serializability, Basic Synchronization Primitives for Concurrency Control- Lock Based Algorithms-Static Locking, Two-Phase Locking (2PL), Time Stamp Based Algorithms- Basic Timestamp Ordering Algorithm, Thomas Write Rule (TWR), Multiversion Timestamp Ordering Algorithm, Conservative Timestamp Ordering Algorithm, Optimistic Algorithms.

Self-Study: Computer security and database security.

- A concurrency control algorithm controls the interleaving of conflicting actions of transactions so that the integrity of a database is maintained, i.e., their net effect is a serial execution.

Locks

- In lock based techniques, each data object has a lock associated with it
- A transaction can request, hold, or release the lock on a data object.
- When a transaction holds a lock, the transaction is said to have locked the corresponding data object.
- A transaction can lock a data object in two modes: exclusive and shared.
- If a transaction has locked a data object in exclusive mode, no other transaction can concurrently lock it in any mode.
- If a transaction has locked a data object in shared mode, other transactions can concurrently lock it but only in shared mode.
- Basically, by locking data objects, a transaction ensures that the locked data objects are inaccessible to other transactions, while temporarily in inconsistent states.

Timestamp

- A timestamp is a unique number that is assigned to a transaction or a data object and is chosen from a monotonically increasing sequence.
- Timestamps are commonly generated according to Lamport's scheme .
- Every site Si has a logical clock C_i, which takes integer values.
- When a transaction T is submitted at a site S_i,
- S_i increments C_i by one and then assigns a 2-tuple (C_i, i) to T.
- The 2-tuple is referred to as the timestamp of T and is denoted by TS(T).

Timestamp

- ➢Every message contains the current clock value of its sender site,
- ➢When a site S_j receives a message with clock value t, it sets C_j to max(t + 1, Cj).
- ➢For any two timestamps
- ✓$ts_1 = (t_1 , i_1)$ and
- ✓ $ts_2 = (t_2 , i_2)$,
- ✓$ts_1 < ts_2$, if either $(t_1 < t_2)$, or $(t_1 = t_2$ and $i_1 < i_2)$

Timestamp

- ➢Timestamps have two properties:
- ✓ uniqueness
- • Timestamps generated by different sites differ in their site id part and
- •Timestamps generated by the same site differ in their clock value part and

Timestamp

✓Monotonicity

•a site generates timestamps in increasing order.

➢Timestamps allow us to place a total ordering on the transactions of a distributed database system by simply ordering the transactions by their timestamps.

➢In concurrency control algorithms for distributed database systems, whenever two concurrent transactions conflict, all sites must agree on a common order of serialization.

➢This can be achieved by

✓ Assigning timestamps to transactions

✓Then having every site serialize conflicting transactions by their timestamps.

Lock Based Algorithms

- In lock based concurrency control algorithms, a transaction must lock a data object before accessing it ,
- In a locking environment, a transaction T is a sequence {a1(d1), a2(d2), ... ,an(dn)} of n actions,
  - ✓ a_i is the operation performed in the i^{th} action
  - ✓ the d_i is the data object acted upon in i^{th} action.
- In addition to read and write, lock and unlock are also permissible actions in locking algorithms.

Lock Based Algorithms

- . A transaction can lock a data object d_i with a "lock(d_i)" action and can relinquish the lock on d_i by an "unlock(d_i)" action.
- A log that results from an execution where a transaction attempting to lock an already locked data object waits, is referred to as a legal log.

Lock Based Algorithms

➢ . A transaction is well-formed if it

✓ Locks a data object before accessing it,

✓ Does not lock a data object more than once, and

✓ Unlocks all the locked data objects before it completes.

Static Locking

- In static locking, a transaction acquires locks on all the data objects it needs before executing any action on the data objects.
- Static locking requires a transaction to predeclare all the data objects it needs for execution.
- A transaction unlocks all the locked data objects only after it has executed all of its actions.
- Static locking is conceptually very simple.

Static Locking

- However, it seriously limits concurrency because any two transactions that have a conflict must execute serially.
- This may significantly limit the performance of the underlying database system.
- Another drawback of static locking is that it requires a priori knowledge of the data objects to be accessed by transactions.
- This may be impractical in applications where the next data object to be locked depends upon the value of another data object.

Basic Timestamp ordering algorithm

- In the basic timestamp ordering algorithm (BTO), the scheduler at each DM keeps track of the largest timestamp of any read and write processed thus far for each data object.
- Let us denote these timestamps by R-ts(object) and W-ts(object), respectively.
- Let read(x, TS) and write(x, v, TS) denote a read and a write request with timestamp TS on a data object x.
- A read(x, TS) request is handled in the following manner:
- If TS < W-ts(x), then the read request is rejected and the corresponding transaction is aborted, otherwise it is executed and R-ts(x) is set to max{R-ts(x), TS}.

- A write(x, v, TS) request is handled in the following manner: If TS < R-ts(x) or TS < W-ts(x), then the write request is rejected, otherwise it is executed and W-ts(x) is set to TS.
- If a transaction is aborted, it is restarted with a new timestamp.
- This method of restart can result in a cyclic restart where a transaction can repeatedly restart and abort without ever completing.
- This algorithm has storage overhead for maintaining timestamps

Thomas Write Rule (TWR)

- The Thomas write rule (TWR) is suitable only for the execution of write actions
- For a write(x, v, TS), if TS < W-ts(x), then TWR says that instead of rejecting the write,

  simply ignore it.
- This is sufficient to enforce synchronization among writes because the effect of ignoring an obsolete write request is the same as executing all writes in their timestamp order.
- However, an additional mechanism is needed for synchronization between reads and writes because TWR takes care of only write-write synchronization.
- Note that TWR is an improvement over the BTO algorithm because it reduces the number of transaction aborts.

1. Whenever a Transaction *T* issues a **W_item(X)** operation, check the following conditions:

- If ***R_TS(X) > TS(T)*** or if ***W_TS(X) > TS(T)***, then abort and rollback T and reject the operation. **else,**
- Execute W_item(X) operation of T and set W_TS(X) to TS(T).

2. Whenever a Transaction *T* issues a **R_item(X)** operation, check the following conditions:

- If ***W_TS(X) > TS(T)***, then abort and reject T and reject the operation, else

If W_TS(X) <= TS(T), then execute the R_item(X) operation of T and set R_TS(X) to the larger of TS(T) and current R_TS(X).

T1(10)	T2(20)	T3(30)
R1(A)		
W1(A)		
	R2(A)	
		W3(A)
W1(A)		

Multiversion Timestamp ordering algorithm

- In the multiversion timestamp ordering (MTO) algorithm,
- A set of R-ts's and < W-ts, value > pairs (called versions) is kept for each data object at the respective DM's.
- The R-ts's of a data object keep track of the timestamps of all the executed read operations, and the versions keep track of the timestamp and the value of all the executed write operations.
- Read and write actions are executed in the following manner:

• A read(x, TS) request is executed by reading the version of x with the largest timestamp less than TS and adding TS to the x's set of R-ts's. A read request is never rejected.

- A write(x, v, TS) request is executed in the following way:
  - If there exists a R-ts(x) in the interval from TS to the smallest W-ts(x) that is larger than TS, then the write is rejected,
  - otherwise it is accepted and a new version of x is created with time- stamp TS.

Conservative timestamp ordering algorithm

- The conservative timestamp ordering algorithm (CTO) altogether eliminates aborts and restarts of transactions by executing the requests in strict timestamp order at all DM's.
- A scheduler processes a request when it is sure that there is no other request with a smaller (older) timestamp in the system.
- Each scheduler maintains two queues—a R-queue and a W-queue—per TM.
- These queues, respectively, hold read and write requests.
- A TM sends requests to schedulers in timestamp order and the communication medium is order preserving.
- A scheduler puts a new read or write request in the corresponding queue in timestamp order.

- ➢This algorithm executes read and write actions in the following way:

- ✓ A read(x, TS) request is executed in the following way.

If every W-queue is nonempty and the first write on each W-queue has a timestamp greater than TS, then the read is executed, otherwise the read(x, TS) request is buffered in the R-queue.

- ✓A write(x, v, TS) request with timestamp TS is executed in the following manner.

- ✓If all R-queues and all W-queues are nonempty and the first read on each R-queue has a timestamp greater than TS and the first write on each W-queue has a timestamp greater than TS. then the write is executed, otherwise the write(x, v, TS) request is buffered in the W-queue.

- ✓ When any read or write request is buffered or executed, buffered requests are tested to see if any of them can be executed.

- ✓That is, if any of the requests in R-queue or W-queue satisfies condition 1 or 2.