Distributed Databases

CS347

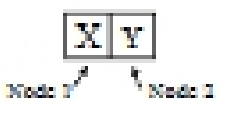
Lecture 15

June 4, 2001

Topics for the day

- Concurrency Control
  - Schedules and Serializability
  - Locking
  - Timestamp control
- Reliability
  - Failure models
  - Two-phase commit protocol

Example



constraint: X=Y

	T_1		T_2
1	a ← X	5	c ← X
2	X ← a+100	6	X ← 2c
3	b ← Y	7	d ← Y
4	Y ← b+100	8	Y ← 2d

Possible Schedule

(node X)		(node Y)	
1 (T_1)	a ← X		
2 (T_1)	X ← a+100		
5 (T_2)	c ← X	3 (T_1)	b ← Y
6 (T_2)	X ← 2c	4 (T_1)	Y ← b+100
		7 (T_2)	d ← Y
		8 (T_2)	Y ← 2d

If X=Y=0 initially, X=Y=200 at end

Precedence: intra-transaction ↓
inter-transaction ↓

Definition of a Schedule

Let T= $\{T_1, T_2, \ldots, T_N\}$ be a set of transactions.
A schedule S over T is a partial order with ordering relation $<_S$ where:

1. $S = \cup T_i$
2. $<_S \supseteq \cup <_i$
3. for any two conflicting operations $p,q \in S$, either $p <_S q$ or $q <_S p$

Note: In centralized systems, we assumed S was a total order and so condition (3) was unnecessary.

Example

(T_1) $r_1[X] \rightarrow w_1[X]$

(T_2) $r_2[X] \rightarrow w_2[Y] \rightarrow w_2[X]$

(T_3) $r_3[X] \rightarrow w_3[X] \rightarrow w_3[Y] \rightarrow w_3[Z]$

S: $r_3[Y] \rightarrow w_3[X] \rightarrow w_3[Y] \rightarrow w_3[Z]$

$r_2[X] \rightarrow w_2[Y] \rightarrow w_2[X]$

$r_1[X] \rightarrow w_1[X]$

Precedence Graph

- Precedence graph P(S) for schedule S is a directed graph where
  - Nodes = $\{T_i \mid T_i \text{ occurs in } S\}$
  - Edges = $\{T_i \rightarrow T_j \mid \exists\, p \in T_i, q \in T_j$ such that p, q conflict and $p <_S q\}$

$r_3[X] \rightarrow w_3[X]$

S: $r_1[X] \rightarrow w_1[X] \rightarrow w_1[Y]$

$r_2[X] \rightarrow w_2[Y]$

P(S): $T_2 \rightarrow T_1 \rightarrow T_3$

Serializability

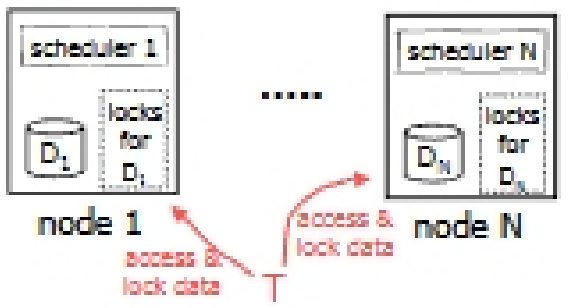
Theorem: A schedule S is serializable iff P(S) is acyclic.

Enforcing Serializability

- Locking
- Timestamp control

Distributed Locking

- Each lock manager maintains locks for local database elements.
- A transaction interacts with multiple lock managers.



Locking Rules

- Well-formed/consistent transactions
  - Each transaction gets and releases locks appropriately
- Legal schedulers
  - Schedulers enforce lock semantics
- Two-phase locking
  - In every transaction, all lock requests precede all unlock requests.

These rules guarantee serializable schedules

Locking replicated elements

- Example:
  - Element X replicated as X_1 and X_2 on sites 1 and 2
  - T obtains read lock on X_1; U obtains write lock on X_2
  - Possible for X_1 and X_2 values to diverge
  - Possible that schedule may be unserializable
- How do we get global lock on logical element X from local locks on one or more copies of X?

Primary-Copy Locking

- For each element X, designate specific copy X_i as primary copy
- Local-lock(X_i) ⇒ Global-lock(X)

Synthesizing Global Locks

- Element X with n copies X_1 .... X_n
- Choose "s" and "x" such that
  - $2x > n$
  - $s + x > n$
- Shared-lock(s copies) ⇒ Global-shared-lock(X)
- Exclusive-lock(x copies) ⇒ Global-exclusive-lock(X)