Database Replication with 1-copy-Snapshot-Isolation

Yi Lin
5th CanDB, Toronto, Oct 17, 2006
Database Replication: What, Why?

- **What?**
  - Replicating data into several databases.

- **Why?**
  - Fast response time:
    - e.g., Montreal users only needs to visit the database in Montreal
  - Scalability:
    - e.g., With more databases, more users’ read requests can be handled.
  - Fault tolerance:
    - e.g. If Montreal database crashes, Montreal users can visit databases in Toronto.
Data Replication: challenge

Problem: If $T_1$ and $T_2$ both commit locally and then propagate the change to the other site, $x$ might not be consistent in two sites at the end.
serializability v.s. snapshot isolation

- **Serializability**
  - Highest *transaction isolation level*
  - The result is the same as executing them serially.
  - Conflict: read/write and write/write

- **Snapshot Isolation (SI):**
  - Read from a snapshot of the committed data as of the time the transaction starts.
  - Conflict: only write/write
  - First committer wins: of 2 concurrent conflicting write transactions, one commits, the other aborts
  - Similarity with optimistic concurrency control
  - Very popular (Oracle, PostgreSQL, SQL Server 2005)
Our new correctness criteria: 1-copy-SI

- **1-copy-SI**: The replicated system behaves as one database providing SI.
  - Each DB provides SI.
  - All read operations read from snapshots producible in one DB
Previous replication protocols

- **Lazy Primary** approaches: e.g., [Ganymed, Middleware04]
  - Restriction: Updates must always be performed on primary copy. Hence, Need to mark read-only transactions in advance
  - Not good in WANs: A update transaction with n operations needs n round trip times to primary copy which may be remote.
  - Single point of failure

- Protocols with middleware-based centralized scheduler (e.g., [Conflict Aware, USENIX03]) have similar problems with Lazy Primary

- Protocols with total order multicast,
  - Restriction: Some need to know all operations in advance
  - Some are table-based locking [Middle-R, ICDCS02]
  - Nearly all only look at 1-copy-serializability [Conflict Aware, Middle-R, State Machine, GlobData, ]
Finally, both sites commit T1 and abort T2. Data item X is consistent in both sites.
SEQ protocol without total order

Site A (sequencer) 

- $r_1(x)$
- $W_1(x)$

$T_1$ validation succeeds 
Commit $T_1$

Site B

- $r_2(x)$
- $W_2(x)$

$T_2$’s Writeset

$T_1$’s Writeset

$T_2$’s Writeset

apply $T_1$’s writeset

Abort $T_2$

Finally, both sites commit $T_1$ and abort $T_2$. Data item $X$ is consistent in both sites
1-copy-SI: problem and solution

• Problem:
The there may exist two contradicting snapshots in two different sites for transactions to read.

• Solution:
Only one of such snapshots is allowed to be read.
Experiment in LANs

5 sites, 100% update transactions,
Experiments in WANs

4 sites, 50% update rate, update in non-sequencer site
Experiments in WANs

4 sites, 50% update rate, reads in non-sequencer site