Timestamps for Serializability

Alternative to Locks for Concurrency Control

Timestamp Serializability Requirement

• Lock Version: Schedule is equivalent to some serial Schedule
• Timestamp Version: Schedule is equivalent to exactly one serial Schedule
• That schedule: TIMESTAMP ORDERED
TIMESTAMP ORDER

• Timestamp ordered Serial Schedule: transactions execute in the same order as they began in the database.
• Goal: Schedule that is serializable to Timestamp Ordered Serial Schedule.

Punish Out of Order Transactions

• Proper order for conflicting pairs:
  Older gets to data item first
  Younger gets to it next
• Out of order discovered when older turns up late.
• Older punished with death.
Example TO Schedules

Serial in the Timestamp Order

T1(10)
Read A
Read B
Write A

T2(20)
Read C
Read B
Write B
Read D

T3(30)
Read C
Write C
Write E

Equivalent: each pair in the same order

Another Example

Serial in the Timestamp Order

T1(10)
Read A
Read B
Write A

T2(20)
Read C
Read B
Write B
Read D

T3(30)
Read C
Write C
Write E

Obviously Serializable...

But NOT equivalent -- so invalid under this protocol
Protocol Method

• Identify Violations of TS order
• Kill Violator
• Restart Violator with new TS
• Mechanism: Each Data Item is Timestamped
  – INHERITED from TS of youngest transaction to access item
  – NOT FROM TIME OF ACCESS
  – item gets Read TS and Write TS

Proper Order: Write-Read

• Only concerned with conflicting Operations
• Proper: Older Writes value then Younger Reads it
• Violation: Younger Reads value, then Older Writes it
• Older should have written first
• Older dies when it tries to write
TS Order: Older Writes, Younger Reads

T4(40) T5(50)
Write A Read A

A
ReadTS 50
WriteTS 40

OK to Read: T5's Timestamp is > A's WriteTS

T4(40) T5(50)
Write A Read A

A
ReadTS 50
WriteTS

Violation: T4's Timestamp is < A's ReadTS. T4 should have been there already. T4 dies.

TSO: Older Reads before Younger Writes

T4(40) T5(50)
Read A Write A

A
ReadTS 40
WriteTS 50

OK to Write: T5's Timestamp is > A's ReadTS

T4(40) T5(50)
Read A Write A

A
ReadTS
WriteTS 50

Violation: T4's Timestamp is < A's WriteTS. T4 should have been there already. T4 dies.
TSO: Older Writes before Younger Writes

<table>
<thead>
<tr>
<th>T4(40)</th>
<th>T5(50)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write A</td>
<td>Write A</td>
</tr>
</tbody>
</table>

A
ReadTS
WriteTS 50

OK to Write: T5’s Timestamp is > A’s WriteTS

<table>
<thead>
<tr>
<th>T4(40)</th>
<th>T5(50)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write A</td>
<td>Write A</td>
</tr>
</tbody>
</table>

A
ReadTS
WriteTS 50

Violation: T4’s Timestamp is < A’s WriteTS. T4 should have been there already. T4 dies.

Summary

- Reading: you cannot read a younger transaction’s value
  - Transaction’s TS must be >= data item’s write TS
- Writing: you cannot write what a younger transaction has read
  - Transaction’s TS must be >= data item’s read TS
- Writing: you cannot overwrite a younger transaction’s value
  - Transaction’s TS must be >= data item’s write TS
Notes

• TSO: killed transaction restarted with new Timestamp
  – younger transactions favored by rules
• TSO protocol does not use locks so NO Deadlock
• TSO protocol guarantees serializable schedules
• TSO drawback: transactions killed unnecessarily