Data-Intensive Computing Systems

Concurrency Control (II)

Shivnath Babu
How to enforce serializable schedules?

Option 1: run system, recording P(S); at end of day, check for P(S) cycles and declare if execution was good
How to enforce serializable schedules?

Option 2: prevent P(S) cycles from occurring

\[ T_1 \quad T_2 \quad \ldots \quad T_n \]

Scheduler

DB
A locking protocol

Two new actions:

lock (exclusive):  \text{l}_i (A)
unlock:  \text{u}_i (A)
Rule #1: Well-formed transactions

Ti: ... li(A) ... pi(A) ... ui(A) ...
Rule #2 Legal scheduler

\[ S = \ldots \text{li}(A) \ldots \text{ui}(A) \ldots \]

no \text{li}(A)
Exercise:

- What schedules are legal?
- What transactions are well-formed?

S1 = l_1(A)l_1(B)r_1(A)w_1(B)l_2(B)u_1(A)u_1(B)
   r_2(B)w_2(B)u_2(B)l_3(B)r_3(B)u_3(B)

S2 = l_1(A)r_1(A)w_1(B)u_1(A)u_1(B)
   l_2(B)r_2(B)w_2(B)l_3(B)r_3(B)u_3(B)

S3 = l_1(A)r_1(A)u_1(A)l_1(B)w_1(B)u_1(B)
   l_2(B)r_2(B)w_2(B)u_2(B)l_3(B)r_3(B)u_3(B)
Exercise:

- What schedules are legal?
  - What transactions are well-formed?

\[ S_1 = l_1(A)l_1(B)r_1(A)w_1(B)l_2(B)u_1(A)u_1(B) \]
\[ r_2(B)w_2(B)u_2(B)l_3(B)r_3(B)u_3(B) \]

\[ S_2 = l_1(A)r_1(A)w_1(B)u_1(A)u_1(B) \]
\[ l_2(B)r_2(B)w_2(B)l_3(B)r_3(B)u_3(B) \]

\[ S_3 = l_1(A)r_1(A)u_1(A)l_1(B)w_1(B)u_1(B) \]
\[ l_2(B)r_2(B)w_2(B)u_2(B)l_3(B)r_3(B)u_3(B) \]
## Schedule F

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( l_1(A) ; \text{Read}(A) )</td>
<td>( l_2(A) ; \text{Read}(A) )</td>
</tr>
<tr>
<td>( A \leftarrow A + 100 ; \text{Write}(A) ; u_1(A) )</td>
<td>( A \leftarrow A \times 2 ; \text{Write}(A) ; u_2(A) )</td>
</tr>
<tr>
<td>( l_1(B) ; \text{Read}(B) )</td>
<td>( l_2(B) ; \text{Read}(B) )</td>
</tr>
<tr>
<td>( B \leftarrow B + 100 ; \text{Write}(B) ; u_1(B) )</td>
<td>( B \leftarrow B \times 2 ; \text{Write}(B) ; u_2(B) )</td>
</tr>
</tbody>
</table>
## Schedule F

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>l₁(A);Read(A)</td>
<td>l₂(A);Read(A)</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>A ← A + 100; Write(A); u₁(A)</td>
<td>A ← A x 2; Write(A); u₂(A)</td>
<td>125</td>
<td></td>
</tr>
<tr>
<td>l₁(B);Read(B)</td>
<td>l₂(B);Read(B)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B ← B + 100; Write(B); u₁(B)</td>
<td>B ← B x 2; Write(B); u₂(B)</td>
<td></td>
<td>50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>250</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>150</td>
</tr>
<tr>
<td></td>
<td></td>
<td>250</td>
<td>150</td>
</tr>
</tbody>
</table>
Rule #3 Two phase locking (2PL) for transactions

\[ T_i = \ldots \text{li}(A) \ldots \text{ui}(A) \ldots \]

- no unlocks
- no locks
# locks held by Ti

![Graph showing the growing and shrinking phases of locks over time.](image-url)
## Schedule G

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l_1(A); \text{Read}(A)$</td>
<td></td>
</tr>
<tr>
<td>$A \leftarrow A + 100; \text{Write}(A)$</td>
<td></td>
</tr>
<tr>
<td>$l_1(B); \ u_1(A)$</td>
<td></td>
</tr>
</tbody>
</table>

$\text{delayed}$

$l_2(A); \text{Read}(A)$

$A \leftarrow Ax2; \text{Write}(A)$

$\text{l}_2(B)$
## Schedule G

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l_1(A); \text{Read}(A)$</td>
<td>$l_2(A); \text{Read}(A)$</td>
</tr>
<tr>
<td>$A \leftarrow A + 100; \text{Write}(A)$</td>
<td>$A \leftarrow A \times 2; \text{Write}(A)$</td>
</tr>
<tr>
<td>$l_1(B); u_1(A)$</td>
<td>delayed $l_2(B)$</td>
</tr>
<tr>
<td>Read(B); $B \leftarrow B + 100$</td>
<td></td>
</tr>
<tr>
<td>Write(B); $u_1(B)$</td>
<td></td>
</tr>
</tbody>
</table>
### Schedule G

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l_1(A); \text{Read}(A)$</td>
<td>$l_2(A); \text{Read}(A)$</td>
</tr>
<tr>
<td>$A \leftarrow A + 100; \text{Write}(A)$</td>
<td>$A \leftarrow Ax2; \text{Write}(A)$</td>
</tr>
<tr>
<td>$l_1(B); u_1(A)$</td>
<td>$l_2(B)$</td>
</tr>
<tr>
<td>$\text{Read}(B); B \leftarrow B + 100$</td>
<td>$l_2(B); u_2(A); \text{Read}(B)$</td>
</tr>
<tr>
<td>$\text{Write}(B); u_1(B)$</td>
<td>$B \leftarrow Bx2; \text{Write}(B); u_2(B)$</td>
</tr>
</tbody>
</table>
# Schedule H  \( (T_2\text{ reversed}) \)

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( l_1(A) ; \text{Read(}A\text{)} )</td>
<td>( l_2(B) ; \text{Read(}B\text{)} )</td>
</tr>
<tr>
<td>( A \leftarrow A + 100 ; \text{Write(}A\text{)} )</td>
<td>( B \leftarrow B \times 2 ; \text{Write(}B\text{)} )</td>
</tr>
<tr>
<td>( l_1(B) ) delayed</td>
<td>( l_2(A) ) delayed</td>
</tr>
</tbody>
</table>
• Assume deadlocked transactions are rolled back
  – They have no effect
  – They do not appear in schedule

E.g., Schedule H =

This space intentionally left blank!
Next step:

Show that rules $#1,2,3 \Rightarrow$ conflict-serializable schedules
Conflict rules for $l_i(A)$, $u_i(A)$:

- $l_i(A)$, $l_j(A)$ conflict
- $l_i(A)$, $u_j(A)$ conflict

Note: no conflict $< u_i(A), u_j(A)>$, $< l_i(A), r_j(A)>$, ...
Theorem  Rules #1,2,3  ⇒  conflict (2PL) serializable schedule

To help in proof:

Definition  Shrink(Ti) = SH(Ti) = first unlock action of Ti
Lemma

Ti → Tj in S ⇒ SH(Ti) <_S_ SH(Tj)

Proof of lemma:

Ti → Tj means that

S = ... p_i(A) ... q_j(A) ...; p,q conflict

By rules 1,2:

S = ... p_i(A) ... u_i(A) ... l_j(A) ... q_j(A) ...

By rule 3:

SH(Ti)    SH(Tj)

So,

SH(Ti) <_S_ SH(Tj)
Theorem  Rules #1,2,3  \Rightarrow  conflict
     (2PL)  serializable
     schedule

Proof:
(1) Assume P(S) has cycle
     \[ T_1 \rightarrow T_2 \rightarrow \ldots T_n \rightarrow T_1 \]
(2) By lemma: \( SH(T_1) < SH(T_2) < \ldots < SH(T_1) \)
(3) Impossible, so P(S) acyclic
(4) \Rightarrow S is conflict serializable
Beyond this simple 2PL protocol, it is all a matter of improving performance and allowing more concurrency. 

- Shared locks
- Multiple granularity
- Inserts, deletes, and phantoms
- Other types of C.C. mechanisms