Data-Intensive Computing Systems

Concurrency Control (II)

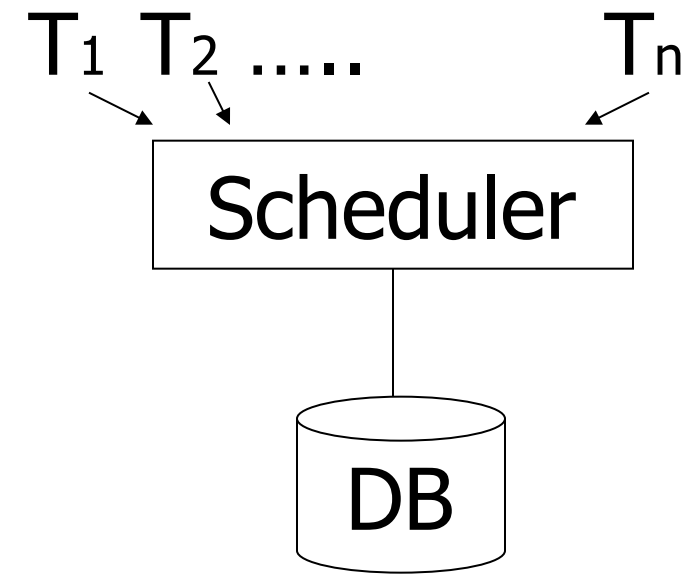
Shivnath Babu

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

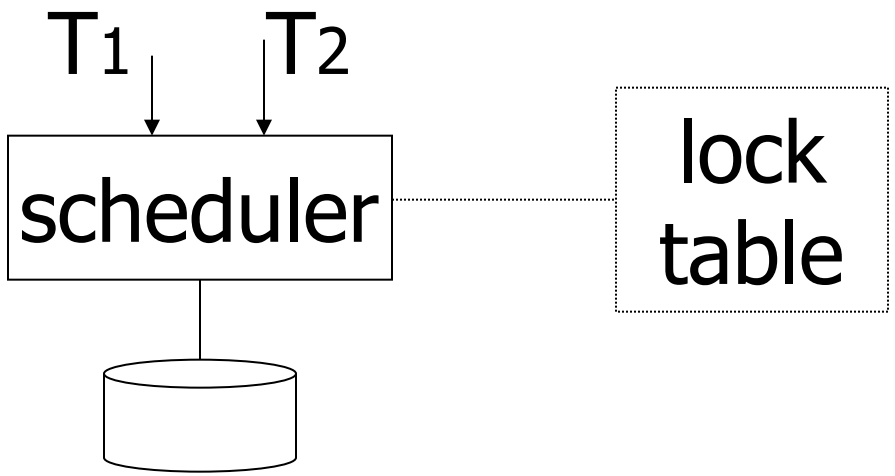


A locking protocol

Two new actions:

lock (exclusive): $l_i(A)$

unlock: $u_i(A)$

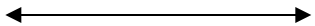


Rule #1: Well-formed transactions

T_i: ... $l_i(A)$ ... $p_i(A)$ ... $u_i(A)$ ...

Rule #2 Legal scheduler

$S = \ldots\ldots\ldots l_i(A) \ldots\ldots\ldots\ldots u_i(A) \ldots\ldots\ldots$



no $l_j(A)$

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?

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

Schedule F

T1	T2
l_1(A);Read(A)	
A←A+100;Write(A);u_1(A)	
	l_2(A);Read(A)
	A←Ax2;Write(A);u_2(A)
	l_2(B);Read(B)
	B←Bx2;Write(B);u_2(B)
l_1(B);Read(B)	
B←B+100;Write(B);u_1(B)	

Schedule F

T1	T2	A	B
		25	25
l_1(A);Read(A)			
A←A+100;Write(A);u_1(A)		125	
	l_2(A);Read(A)		
	A←Ax2;Write(A);u_2(A)	250	
	l_2(B);Read(B)		
	B←Bx2;Write(B);u_2(B)		50
l_1(B);Read(B)			
B←B+100;Write(B);u_1(B)			150
		250	150

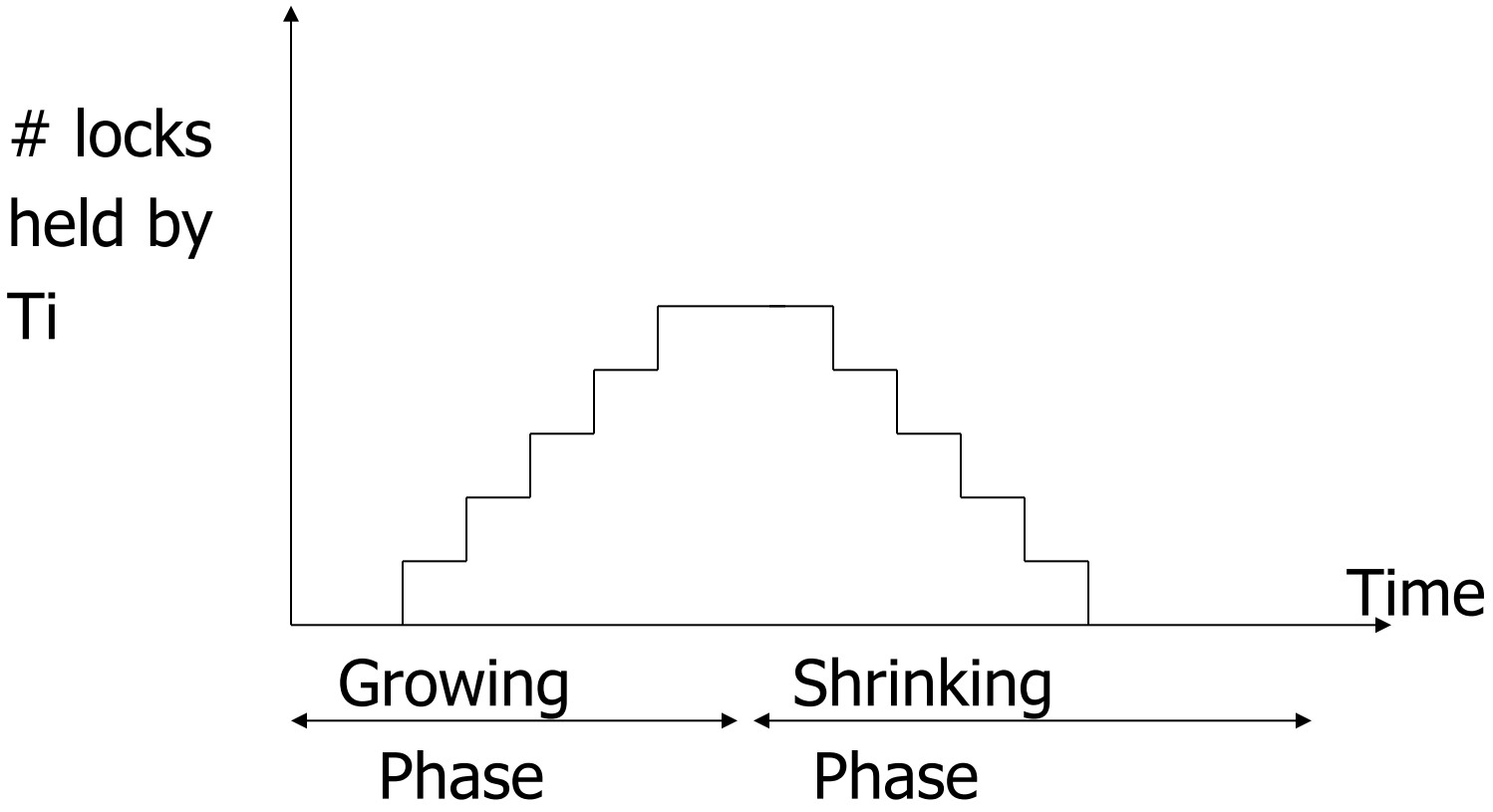
Rule #3 Two phase locking (2PL)

for transactions

T_i = ... $l_i(A)$ ..... $u_i(A)$ .....

no unlocks

no locks


locks
held by
Ti
Time
Growing
Phase
Shrinking
Phase

Schedule G

T1	T2
l_1(A);Read(A)	
A←A+100;Write(A)	
l_1(B); u_1(A)	
	l_2(A);Read(A)
	A← Ax2;Write(A); l_2(B)

delayed

Schedule G

T1	T2
l_1(A);Read(A)	
A←A+100;Write(A)	
l_1(B); u_1(A)	
	l_2(A);Read(A)
	A←Ax2;Write(A); l_2(B)
Read(B);B ← B+100	
Write(B); u_1(B)	

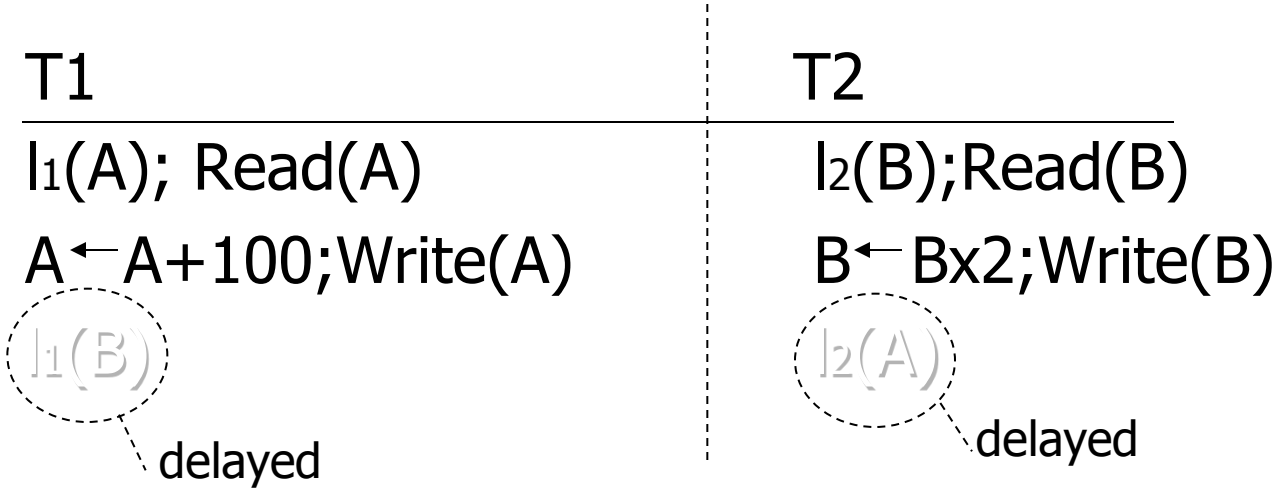
delayed

Schedule G

T1	T2
l_1(A);Read(A)	
A←A+100;Write(A)	
l_1(B); u_1(A)	
	l_2(A);Read(A)
	A←Ax2;Write(A); l_2(B)
Read(B);B ← B+100	
Write(B); u_1(B)	
	l_2(B); u_2(A);Read(B)
	B ← Bx2;Write(B);u_2(B);

delayed

Schedule H (T2 reversed)



- Assume deadlocked transactions are rolled back
  - They have no effect
  - They do not appear in schedule

E.g., Schedule H = 

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 $\Rightarrow$ conflict
(2PL) serializable
schedule

To help in proof:

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

Lemma

$Ti \rightarrow Tj$ in $S \Rightarrow SH(Ti) <_S SH(Tj)$

Proof of lemma:

$Ti \rightarrow Tj$ means that

$S = \ldots p_i(A) \ldots q_j(A) \ldots$; p,q conflict

By rules 1,2:

$S = \ldots p_i(A) \ldots u_i(A) \ldots l_j(A) \ldots q_j(A) \ldots$

By rule 3: SH(Ti) SH(Tj)

So, $SH(Ti) <_S SH(Tj)$

Theorem Rules #1,2,3 (2PL) $\Longrightarrow$ conflict 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