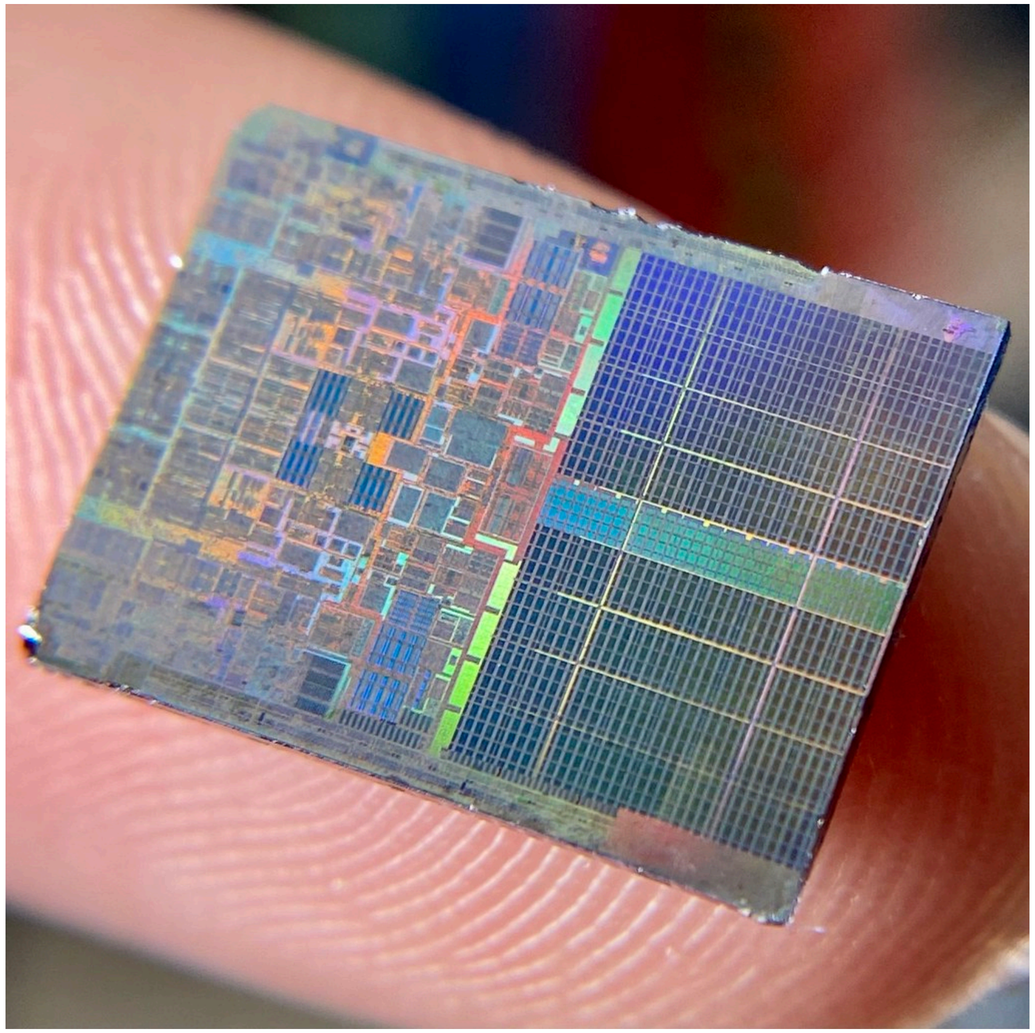
Transactions & Schedules

Remy Wang, 4/29/25

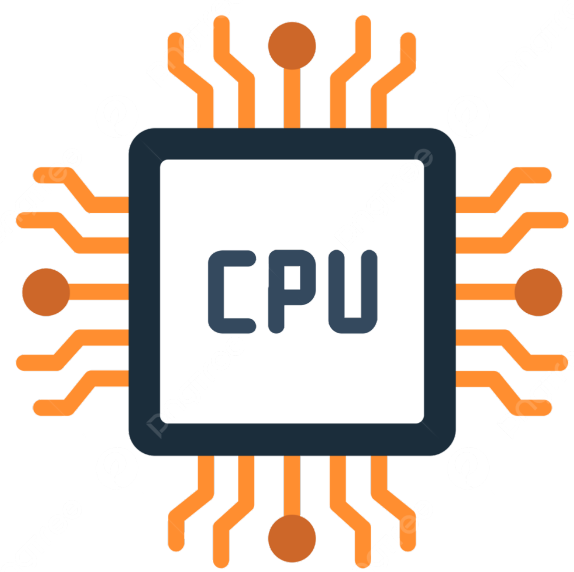
into the hard drive: https://youtu.be/f07mLQwt-AI

magnets! https://youtu.be/f3BNHhfTsvk

real drive: https://youtube.com/shorts/0i1Ynk2WVGw



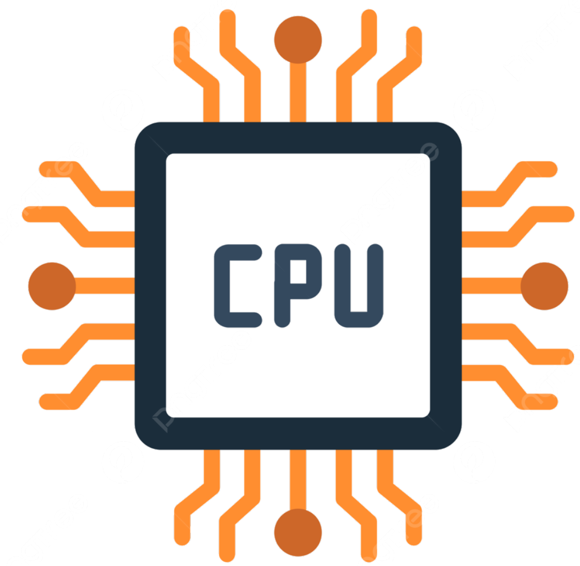
input/output (**IO**)



slow

fast

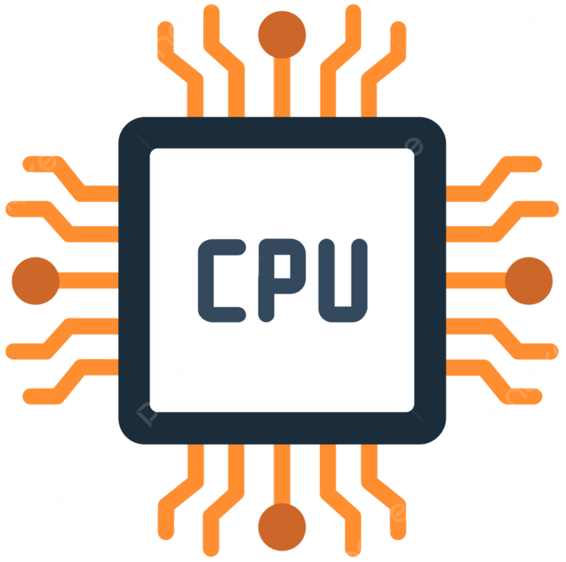
input/output (**IO**)



What do these have to do with transactions?

concurrency != parallelism

keep both busy



T1	T2
x = READ(A)	
x = f(x)	
WRITE(A, x)	
	y = READ(B)
	y = g(y)
	WRITE(B, y)

time ↓

time

T1	T2
x = READ(A)	y = READ(B)
x = f(x)	y = g(y)
WRITE(A, x)	WRITE(B, y)

T1	T2
x = READ(A)	
x = f(x)	y = READ(B)
WRITE(A, x)	y = g(y)
	WRITE(B, y)

time ↓

schedule

ordering of actions s.t.:

1. TXNs don't interfere
2. improve concurrency

schedule

strict ordering of actions s.t.:

1. TXNs don't interfere
2. improve concurrency

time ↓

T1	T2
~~x =~~ READ(A)	
~~x = f(x)~~	
WRITE(A~~, x~~)	
	~~y =~~ READ(B)
	~~y = g(y)~~
	WRITE(B~~, y~~)

time

T1	T2
READ(A)	
WRITE(A)	
	READ(B)
	WRITE(B)

T1	T2
READ(A)	
WRITE(A)	
	READ(B)
	WRITE(B)

time

T1	T2
	READ(B)
	WRITE(B)
READ(A)	
WRITE(A)	

time

serial schedule

1 TXN at a time

2 serial schedules can differ!

	T1	T2	
A = 10			
	x = READ(A)		
x = 2x			
	WRITE(A, x)		
			A = ?
		y = READ(A)	
			y = y+2
		WRITE(A, y)	
			A = ?

time

time		T1	T2	
			y = READ(A)	A = 10
			WRITE(A, y)	y = y+2
		x = READ(A)		A = ?
	x = 2x	WRITE(A, x)		
	A = ?			

serial schedule

1 TXN at a time

2 serial schedules can differ!

not our problem though 🤷‍♂️

serial schedule

1 TXN at a time **slow**

2 serial schedules can differ!

not our problem though 🤷‍♂️

T1	T2
x = READ(A)	
WRITE(A, x)	
	y = READ(B)
	WRITE(B, y)

T1	T2
x = READ(A)	
	y = READ(B)
WRITE(A, x)	
	WRITE(B, y)

T1	T2
x = READ(A)	
x = 2x	
WRITE(A, x)	
	y = READ(B)
	y = y + 2
	WRITE(B, y)

T1	T2
x = READ(A)	
x = 2x	y = READ(B)
WRITE(A, x)	y = y+2
	WRITE(B, y)

serial schedule

1 TXN at a time **slow**

interleaved TXNs improve concurrency

serial schedule

1 TXN at a time **slow**

interleaved TXNs improve concurrency

but how?

same result!

T1	T2
x = READ(A)	
x = 2x	
WRITE(A, x)	
	y = READ(B)
	y = y + 2
	WRITE(B, y)

T1	T2
x = READ(A)	
x = 2x	y = READ(B)
WRITE(A, x)	y = y+2
	WRITE(B, y)

T1	T2
x = R(A)	
x = 2x	
W(A, x)	
	y = R(A)
	y = y + 2
	W(A, y)

T1	T2
	y = R(A)
	y = y + 2
	W(A, y)
x = R(A)	
x = 2x	
W(A, x)	

T1	T2
x = R(A)	
x = 2x	y = R(A)
W(A, x)	y = y+2
	W(A, y)

serializable schedule

equivalent to *some* serial schedule

how to check?

conflict

2 actions *conflict*

if they affect each other

T1	T2
R(A)	
	R(B)
W(A)	
	W(B)

T1	T2
R(A)	
	R(A)
W(A)	
	W(A)

conflict

$R_1(x), W_2(x)$

$W_1(x), R_2(x)$

$W_1(x), W_2(x)$

conflict-equivalent

2 schedules conflict-equivalent

if one "swaps" into the other

T1	T2
R(A)	
W(A)	
	R(B)
	W(B)

T1	T2
R(A)	
	R(B)
W(A)	
	W(B)

T1	T2
R(A)	
W(A)	
	R(B)
	W(B)

T1	T2
R(A)	
	R(B)
W(A)	
	W(B)

conflict-serializable

a schedule is conflict-serializable

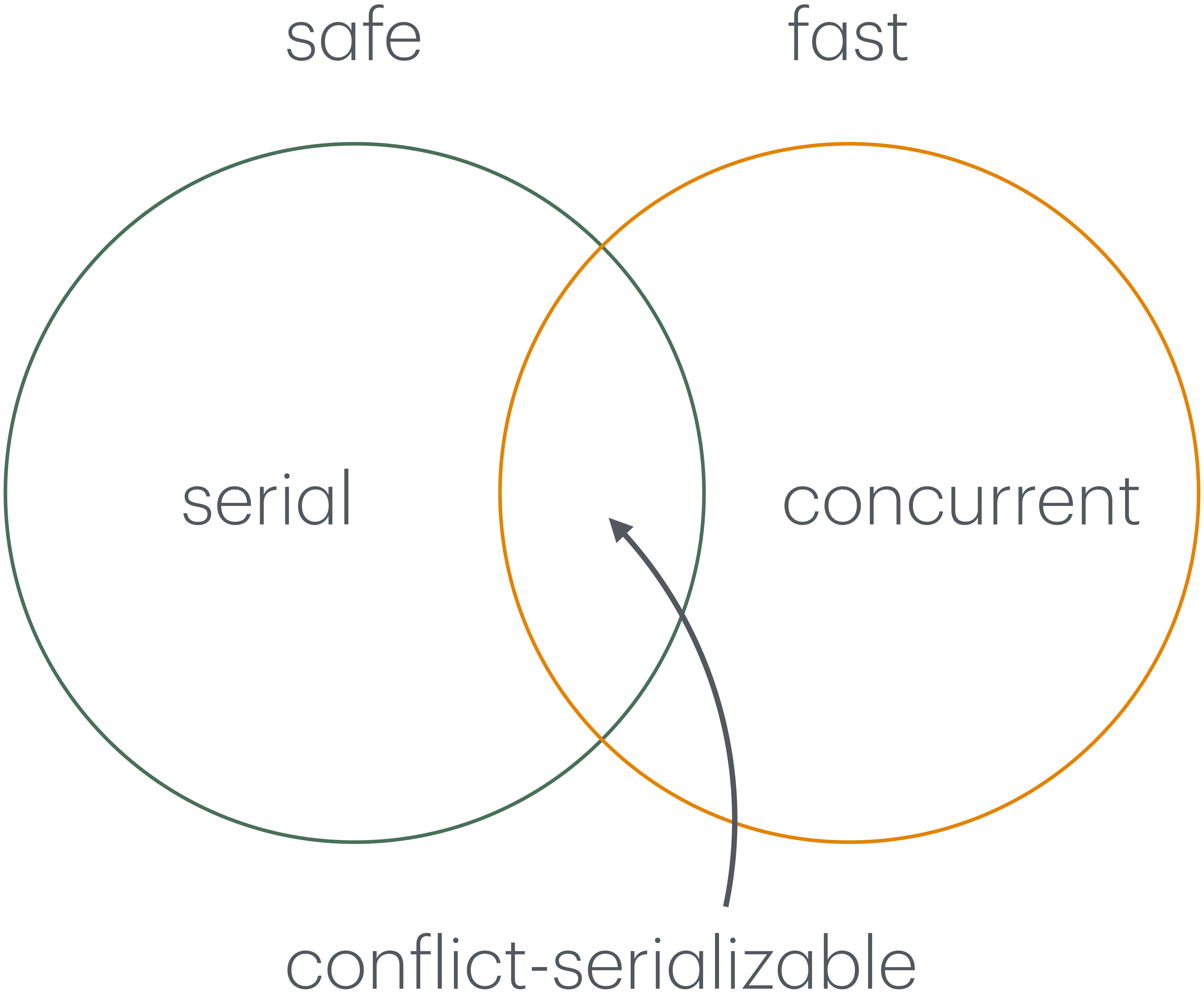
if conflict-equivalent to a serial one

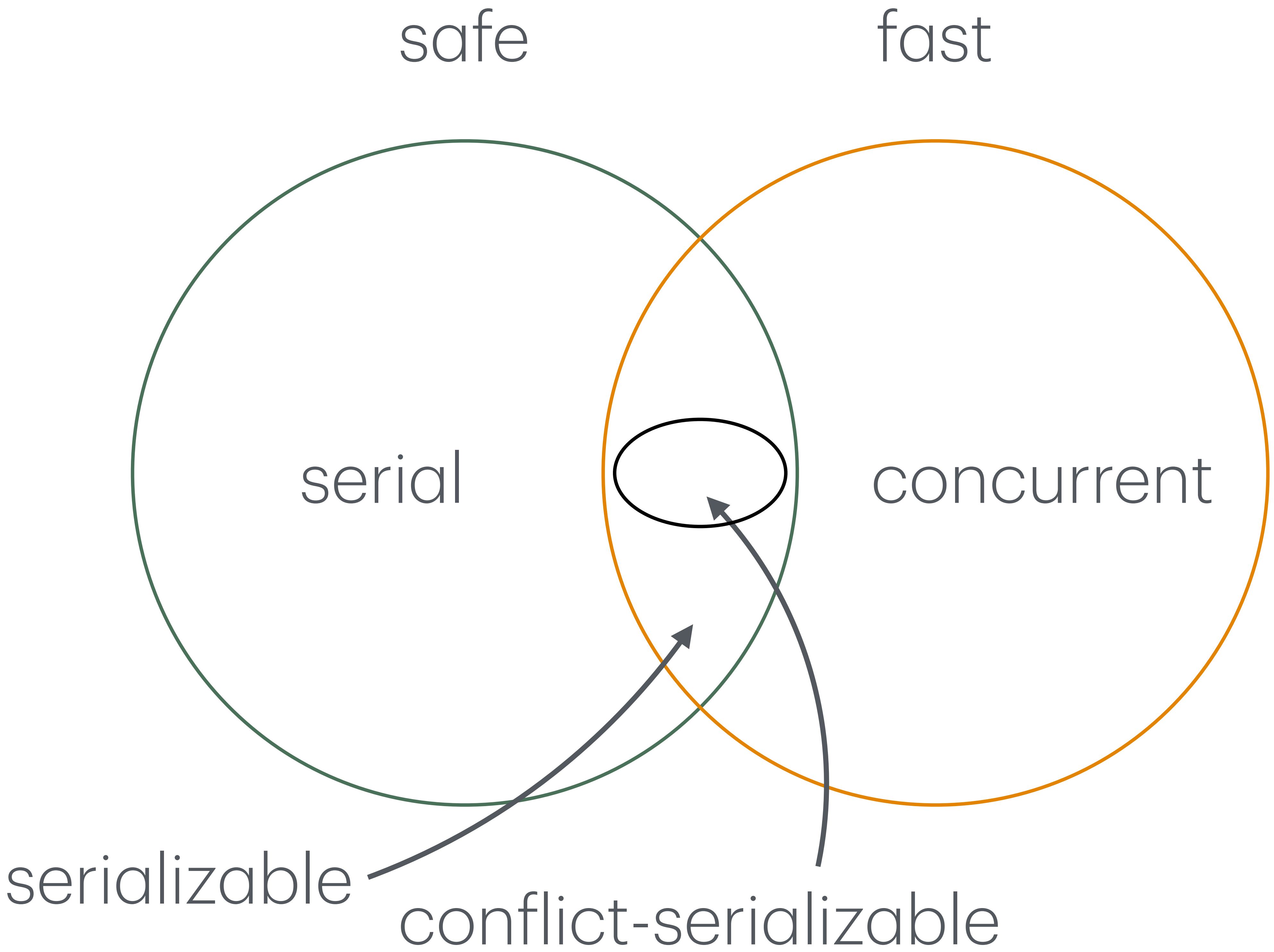
T1	T2
R(A)	
W(A)	
	R(B)
	W(B)

T1	T2
R(A)	
	R(B)
W(A)	
	W(B)

T1	T2
R(A)	
W(A)	
	R(A)
	W(A)

T1	T2
R(A)	
	R(A)
W(A)	
	W(A)


safe
fast
serial
concurrent
conflict-serializable



safe
fast
serial
concurrent
serializable
conflict-serializable

conflict-serializable

a schedule is conflict-serializable

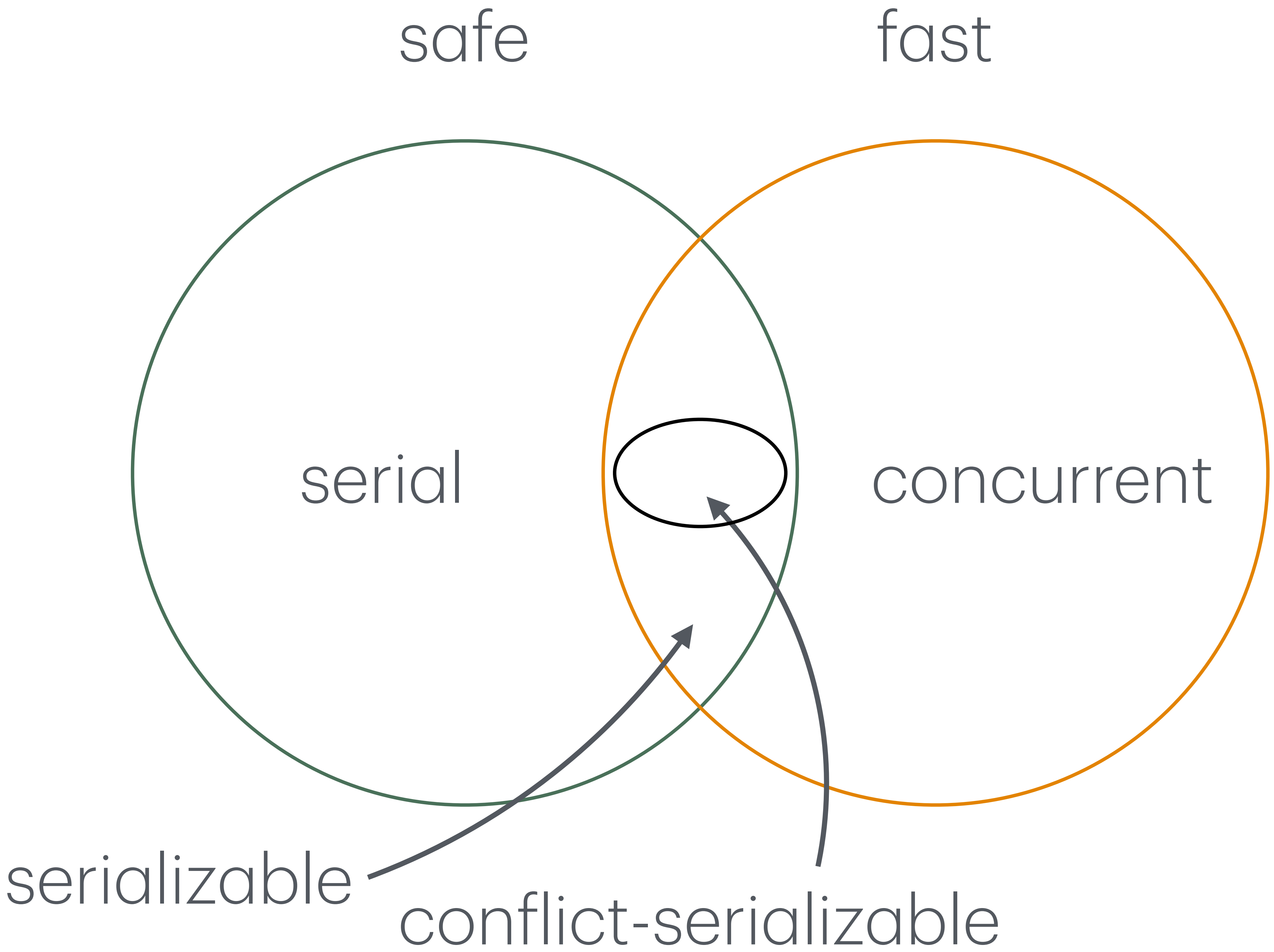
if conflict-equivalent to a serial one

serializable

a schedule is serializable

if equivalent to a serial one

T1	T2	...	**TJ**
x = R(A)		...	
	y = R(A)	...	
	W(A, y)	...	
W(A, x)		...	
			W(A, 0)


safe
fast
serial
concurrent
serializable
conflict-serializable

check conflict-serializable?

use the *precedence graph*

nodes: TXNs

edges: conflicts (between TXNs)

$r_2(A)$; $r_1(B)$; $w_2(A)$; $r_3(A)$; $w_1(B)$; $w_3(A)$; $r_2(B)$; $w_2(B)$

1 2 3

theorem

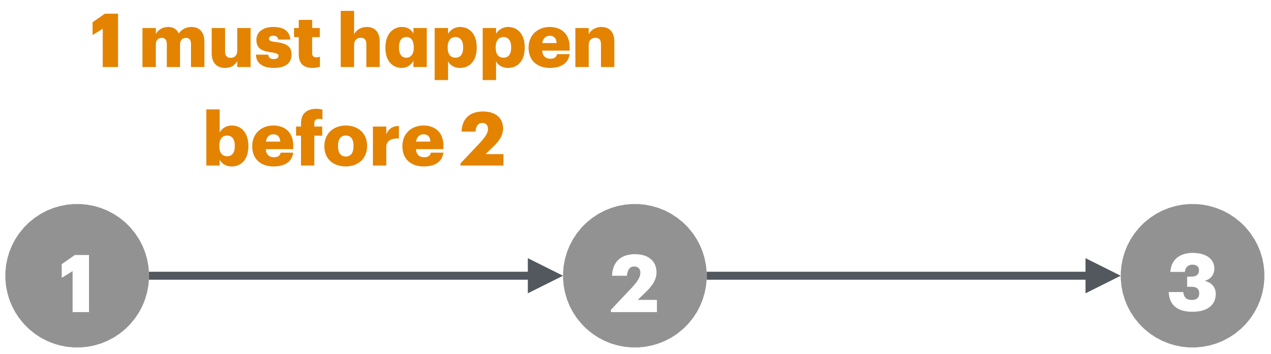
a schedule is conflict-serializable

iff the precedence graph has no cycle

nodes: TXNs

edges: conflicts (between TXNs)

$r_2(A); r_1(B); w_2(A); r_3(A); w_1(B); w_3(A); r_2(B); w_2(B)$



nodes: TXNs

edges: conflicts (between TXNs)

$r_2(A)$; $r_1(B)$; $w_2(A)$; $r_2(B)$; $r_3(A)$; $w_1(B)$; $w_3(A)$; $w_2(B)$

to ensure serializability...

use locks!

enforce *serial* schedule?

enforce *serial* schedule?

each TXN lock entire DB

SQLite does this!

T1	T2
L	
R(A)	
W(A)	
U	
	L
	R(B)
	W(B)
	U

enforce *serial* schedule?

each TXN lock entire DB

SQLite does this!

but uses read/write lock to be fast

SQLite locks

read lock upon SELECT

upgrade to write lock upon INSERT

read locks are shared, write exclusive

T1	T2
RL	
R(A)	RL
	R(B)
	U
WL	
W(A)	
U	

T1	T2
R(A)	
	R(A)
	W(A)
W(A)	

T1	T2
RL	
R(A)	RL
	R(B)
	WL
~~WL~~	W(B)
W(A)	
U	

one lock per DB "item"

item = row, entry, page, etc.

improve concurrency

T1	T2
L	
R(A)	
W(A)	
U	
	L
	R(B)
	W(B)
	U

T1	T2
L(A)	
R(A)	L(B)
W(A)	R(B)
U(A)	W(B)
	U(B)

T1	T2
L(A), R(A)	
W(A), U(A)	
	L(A), R(A)
	W(A), U(A)
	L(B), R(B)
	W(B), U(B)
L(B), R(B)	
W(B), U(B)	

T1	T2
R(A)	
W(A)	
	R(A)
	W(A)
	R(B)
	W(B)
R(B)	
W(B)	