CAP Theorem & Causal Consistency

CS 475: Concurrent & Distributed Systems (Fall 2021)

Lecture 12

Yue Cheng

Some material taken/derived from:

- Princeton COS-418 materials created by Michael Freedman.
- MIT 6.824 by Robert Morris, Frans Kaashoek, and Nickolai Zeldovich.

Tradeoffs are fundamental?

2PC / Consensus | **Eventual consistency**

Seq. (Strong).

Paxos / Raft | **Dynamo**

Strong (L).

"CAP" conjecture for distributed systems

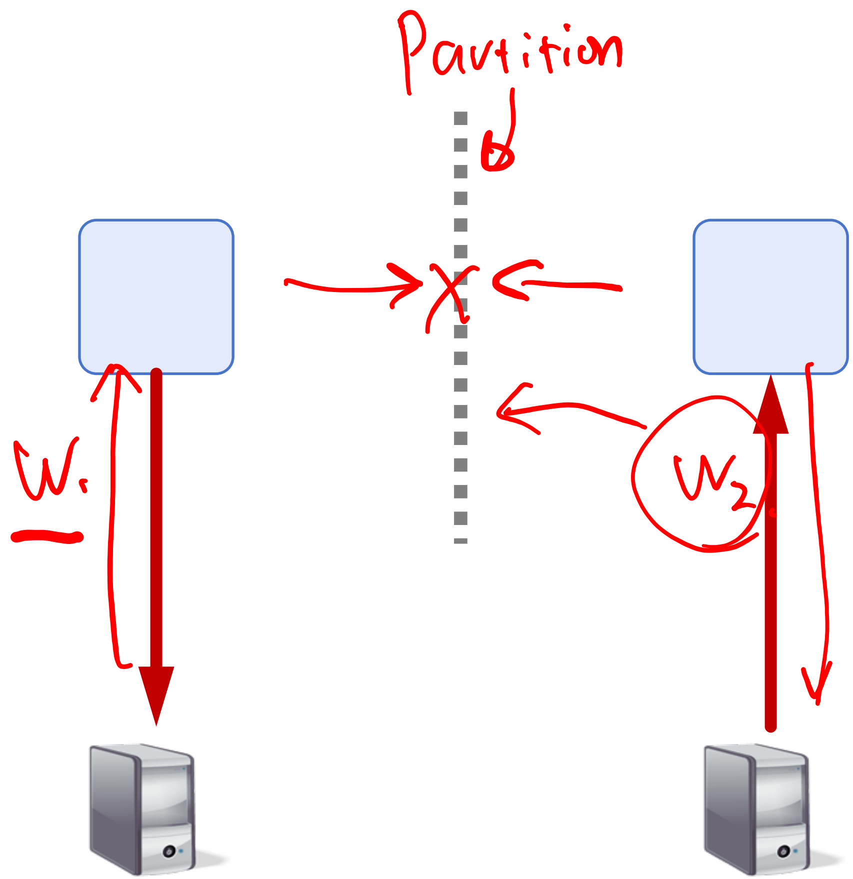
- From keynote lecture by Eric Brewer (2000)
  - History: Eric started Inktomi, early Internet search site based around "commodity" clusters of computers
  - Using CAP to justify "BASE" model: Basically Available, Soft-state services with Eventual consistency

"CAP" conjecture for distributed systems

- From keynote lecture by Eric Brewer (2000)
  - History: Eric started Inktomi, early Internet search site based around "commodity" clusters of computers
  - Using CAP to justify "BASE" model: Basically Available, Soft-state services with Eventual consistency
- Popular interpretation: 2-out-of-3
  - Consistency (Linearizability)
  - Availability
  - Partition Tolerance: Arbitrary crash/network failures

CAP Theorem: Proof

P + A

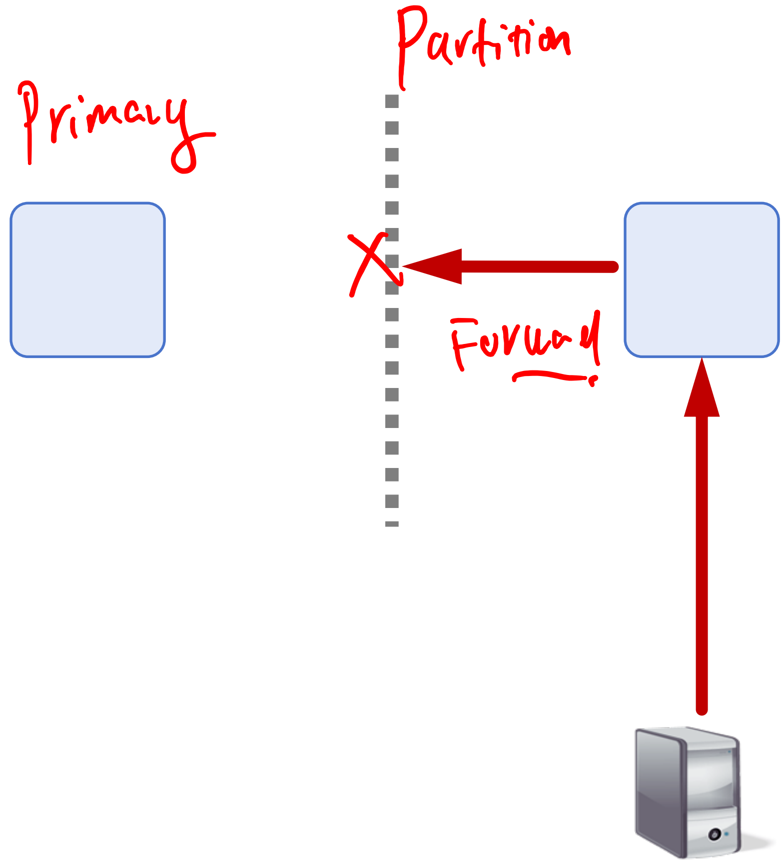


Not consistent

Gilbert, Seth, and Nancy Lynch. "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services." ACM SIGACT News 33.2 (2002): 51-59.

CAP Theorem: Proof

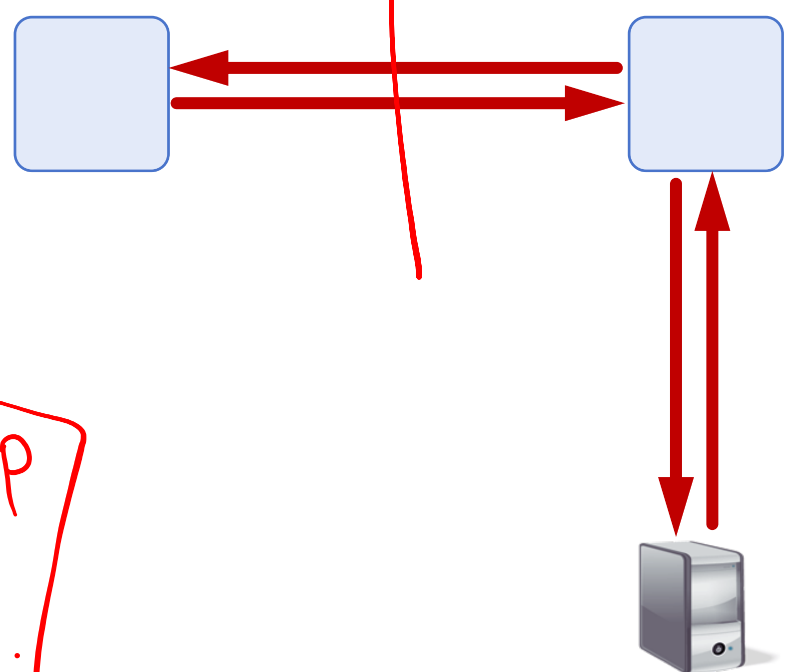
P. + C.



Not available

Gilbert, Seth, and Nancy Lynch. "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services." ACM SIGACT News 33.2 (2002): 51-59.

CAP Theorem: Proof

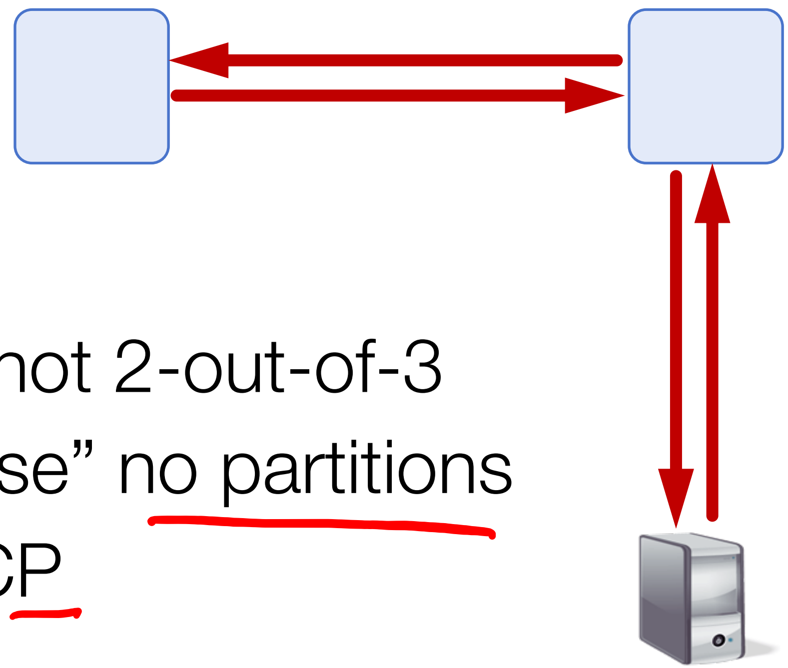


Not partition tolerant

Gilbert, Seth, and Nancy Lynch. "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services." ACM SIGACT News 33.2 (2002): 51-59.

CAP Theorem: AP or CP

PNUT



Not partition tolerant

Criticism: It's not 2-out-of-3

- Can't "choose" no partitions
- So: AP or CP

More tradeoffs L vs. C

- **L:** Low-latency: Speak to fewer than quorum of nodes?
  - 2PC: write N, read 1
  - Raft: write ⌊N/2⌋ + 1, read ⌊N/2⌋ + 1
  - General: |W| + |R| > N

More tradeoffs L vs. C

- **L:** Low-latency: Speak to fewer than quorum of nodes?
  - 2PC: write N, read 1
  - Raft: write $\lfloor N/2 \rfloor + 1$, read $\lfloor N/2 \rfloor + 1$
  - General: $|W| + |R| > N$

- L and C are fundamentally at odds
  - "C" = linearizability, sequential, serializability (more later)

PACELC

- If there is a partition (P):
  - How does system tradeoff A and C?
- Else (no partition)
  - How does system tradeoff L and C?

PACELC

Expressiveness.

- If there is a partition (P):
  - How does system tradeoff A and C?
- Else (no partition)
  - How does system tradeoff L and C?

- Is there a useful system that switches?
  - Dynamo: PA/EL
  - “ACID” DBs: PC/EC

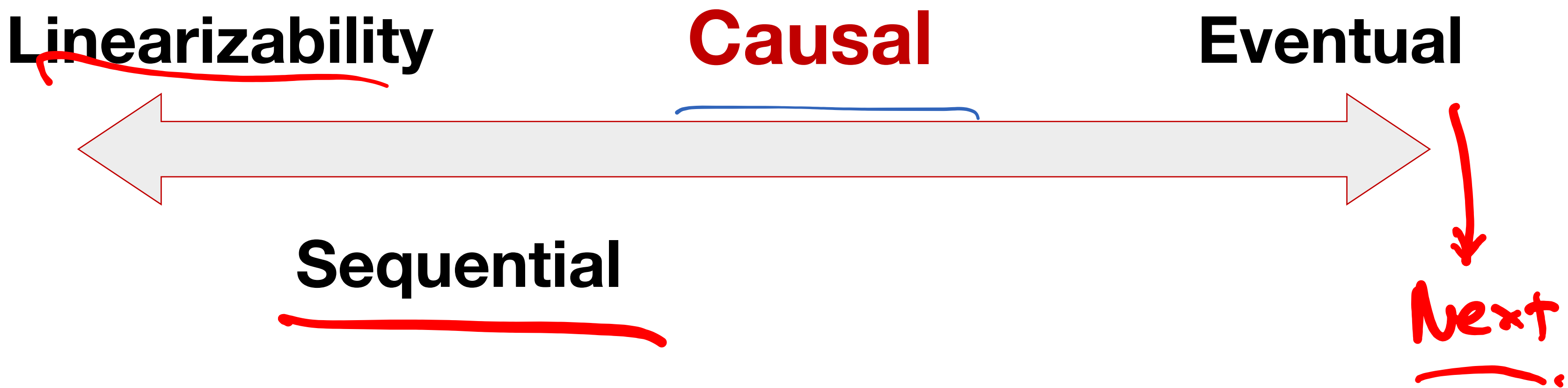
PACELC

- If there is a partition (P):
  - How does system tradeoff A and C?
- Else (no partition)
  - How does system tradeoff L and C?

- Is there a useful system that switches?
  - Dynamo: PA/EL
  - "ACID" DBs: PC/EC

http://dbmsmusings.blogspot.com/2010/04/problems-with-cap-and-yahoos-little.html

Consistency models



Recall use of logical clocks (lec N?)

5
6.

- Lamport clocks: $C(a) < C(z)$ Conclusion: None
- Vector clocks: $V(a) < V(z)$ Conclusion: $a \rightarrow \ldots \rightarrow z$

Recall use of logical clocks (lec N?)

- Lamport clocks: C(a) < C(z) Conclusion: None
- Vector clocks: V(a) < V(z) Conclusion: a → … → z

BBS

- Distributed bulletin board application
  - Each post gets sent to all other users
  - Consistency goal: No user to see reply before the corresponding original message post
  - Conclusion: Deliver message only **after** all messages that **causally precede** it have been delivered

Causal Consistency

Causal Consistency

1. Writes that are *potentially* causally related must be seen by all machines in same order.

Causal Consistency

1. Writes that are *potentially* causally related must be seen by all machines in same order.

2. Concurrent writes may be seen in a different order on different machines.

Causal Consistency

1. Writes that are *potentially* causally related must be seen by all machines in same order.

2. Concurrent writes may be seen in a different order on different machines.

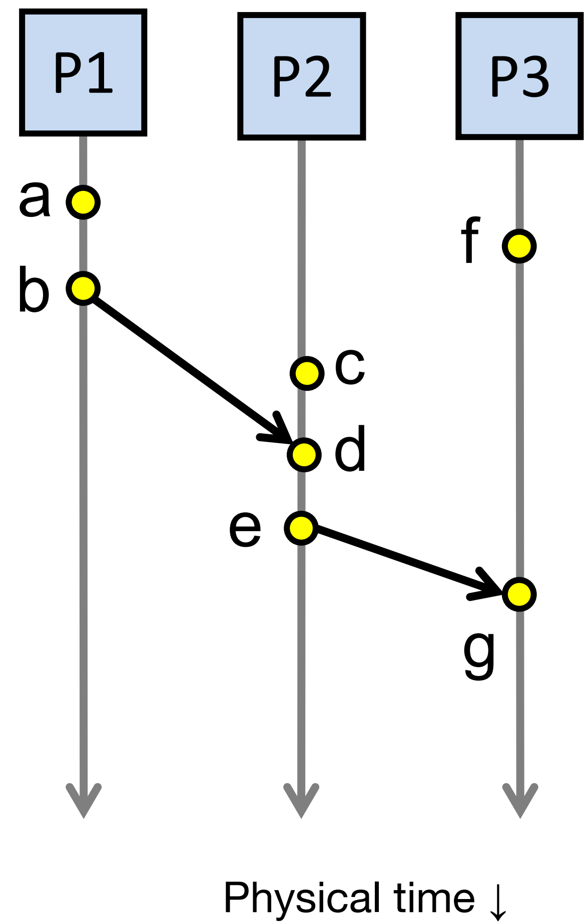
- Concurrent: Ops not causally related

Causal Consistency

1. Writes that are *potentially* causally related must be seen by all machines in same order.

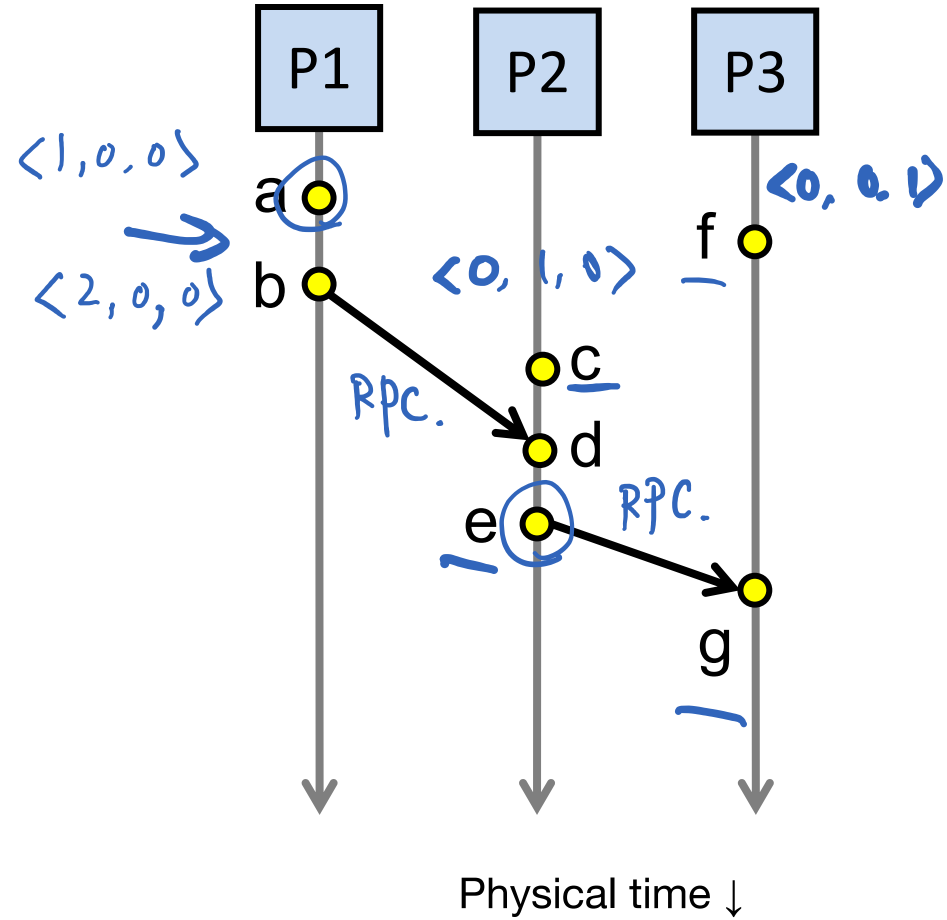
2. Concurrent writes may be seen in a different order on different machines.

- Concurrent: Ops not causally related



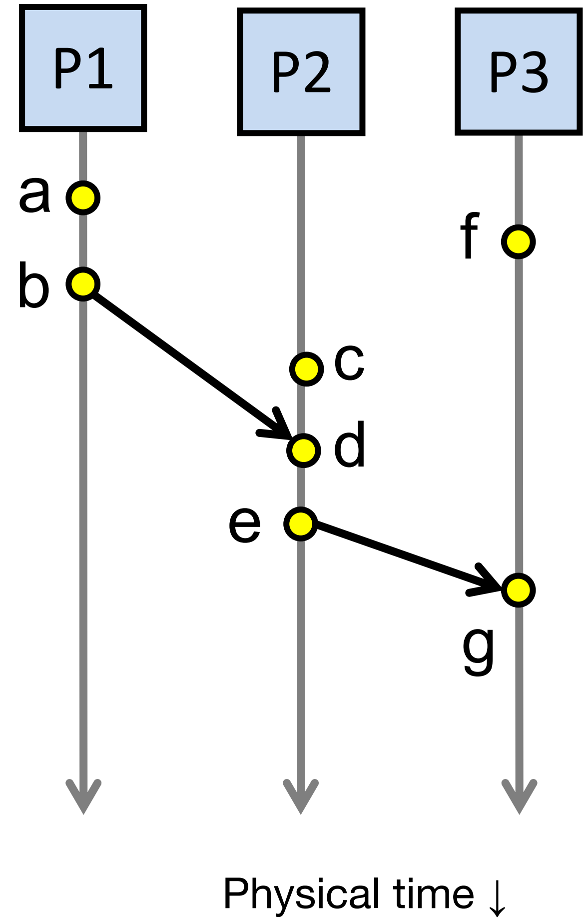
Causal Consistency

Operations	Concurrent?
a, b	N
b, f	Y
c, f	Y
e, f	Y
e, g	N
a, c	Y
a, e	N



Causal Consistency

Operations	Concurrent?
a, b	**N**
b, f	**Y**
c, f	**Y**
e, f	**Y**
e, g	**N**
a, c	**Y**
a, e	**N**



Causal Consistency: Quiz

P1:	W(x)a			W(x)c		
P2:		R(x)a	W(x)b			
P3:		R(x)a			R(x)c	R(x)b
P4:		R(x)a			R(x)b	R(x)c

concurrent

Valid.

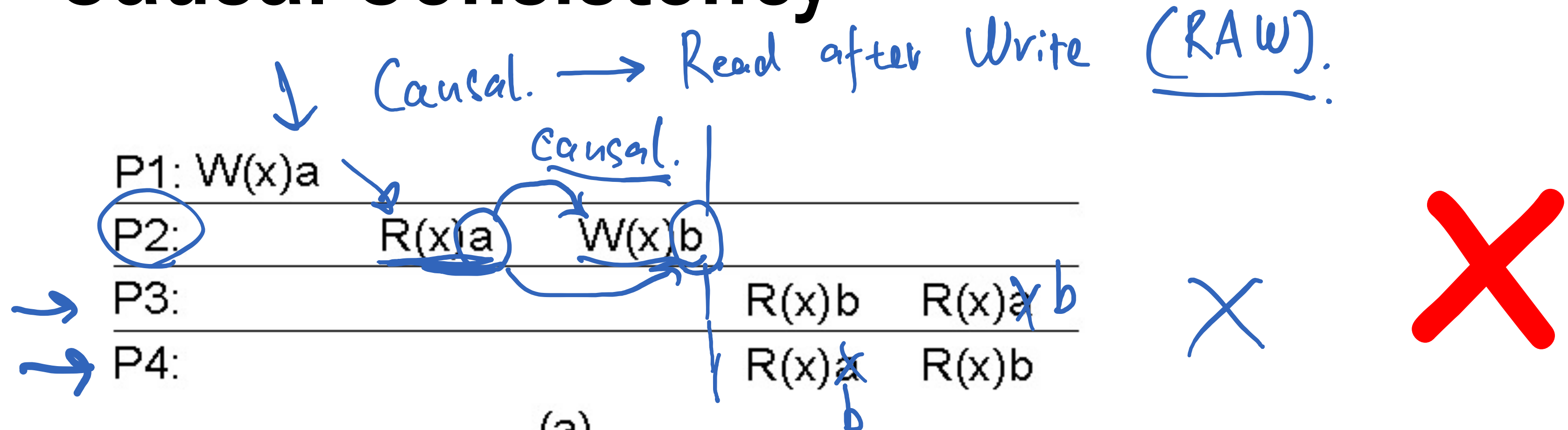
- Valid under causal consistency
- **Why?** *W(x)b* and *W(x)c* are concurrent
  - So all processes don't (need to) see them in same order
- P3 and P4 read the values 'a' and 'b' in order as potentially causally related. No 'causality' for 'c'.

Sequential Consistency: Quiz

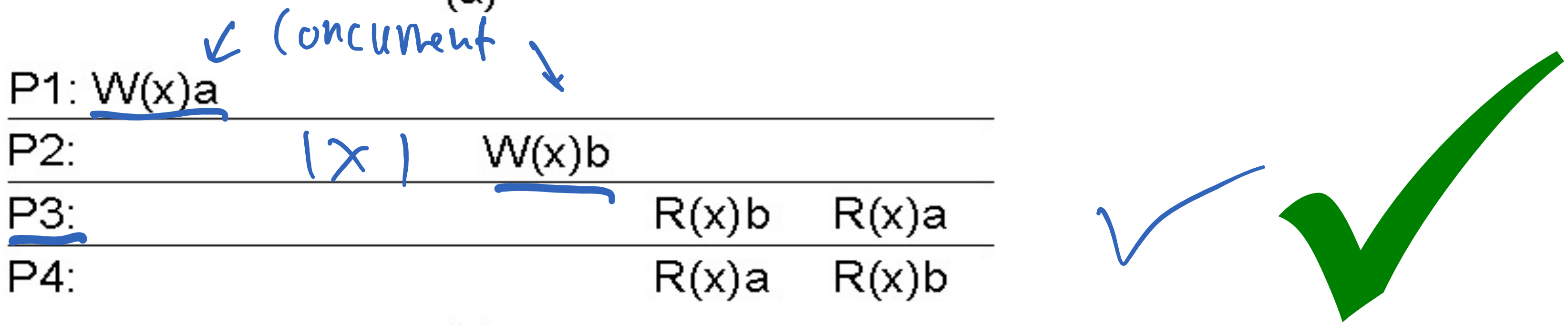
P1:	W(x)a			W(x)c		
P2:		R(x)a	W(x)b			
P3:		R(x)a			R(x)c	R(x)b
P4:		R(x)a			R(x)b	R(x)c

- Invalid under sequential consistency
- **Why?** P3 and P4 see b and c in different order
- But fine for causal consistency
  - B and C are not causually dependent
  - Write after write has no dep's, write after read does

Causal Consistency



(a)



(b)

A: Violation: W(x)b is potentially dep on W(x)a

B: Correct. P2 doesn't read value of a before W