Consistency and Linearizability

CS 4740: Cloud Computing

Fall 2024

Lecture 9

Yue Cheng

University of Virginia

Some material taken/derived from:

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

Fault tolerance vs. Consistency

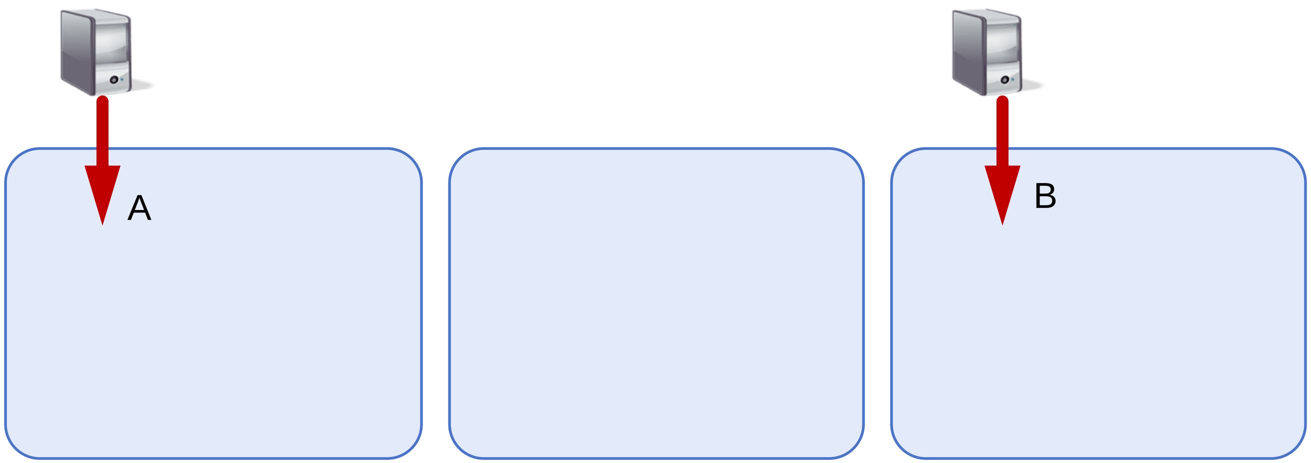
Fault-tolerance / durability:

- Don't lose operations

Consistency:

- Ordering between (visible) operations

Correct consistency model?



- Let's say A and B send an op.
- All readers see A → B ?
- All readers see B → A ?
- Some see A → B and others B → A ?

Consistency models

Strong consistency — **Eventual consistency**

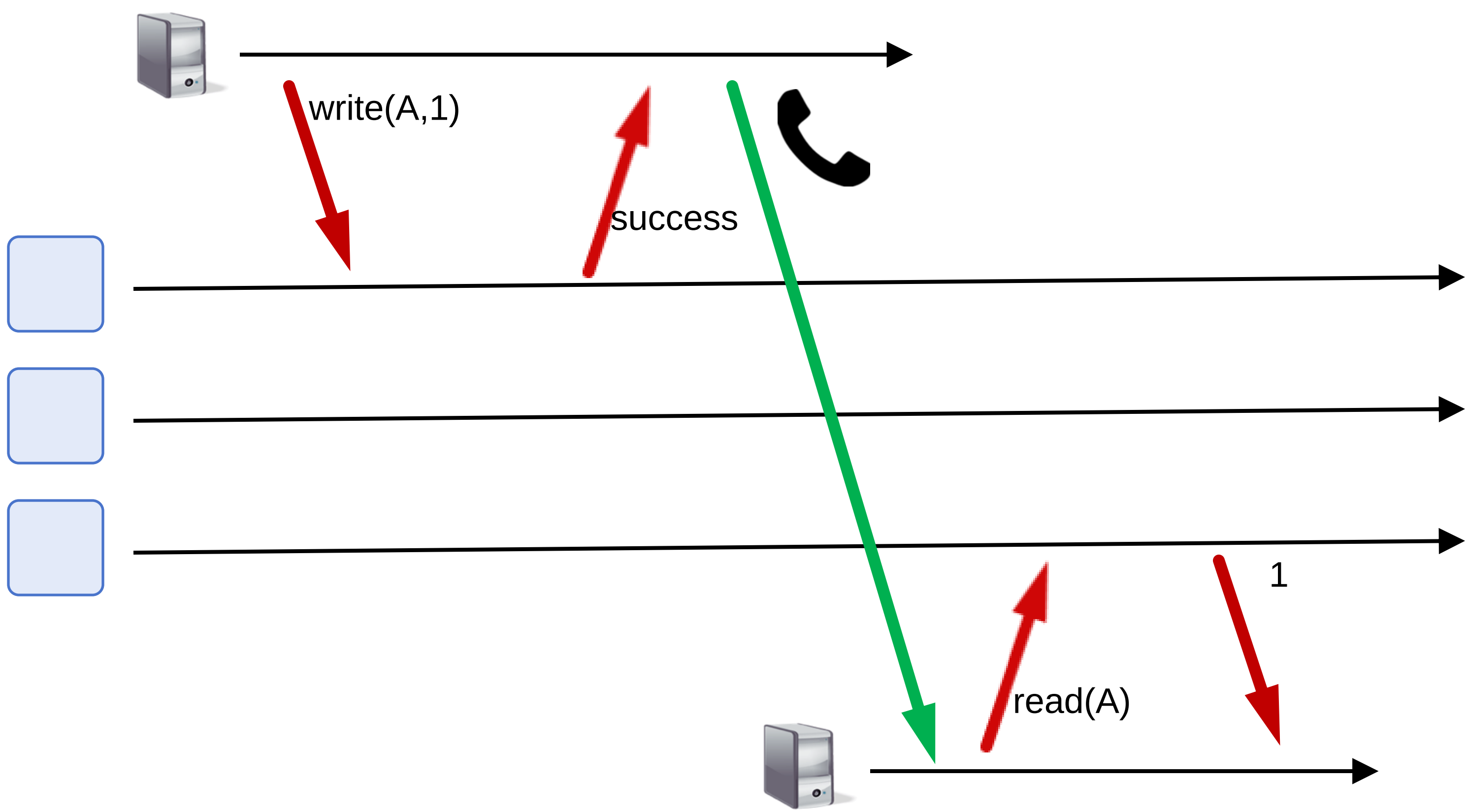
Paxos / Raft — Amazon Dynamo

Strong consistency

- Provide behavior of a single copy of object:
  - Read should return the most recent write
  - Subsequent reads should return same value, until next write

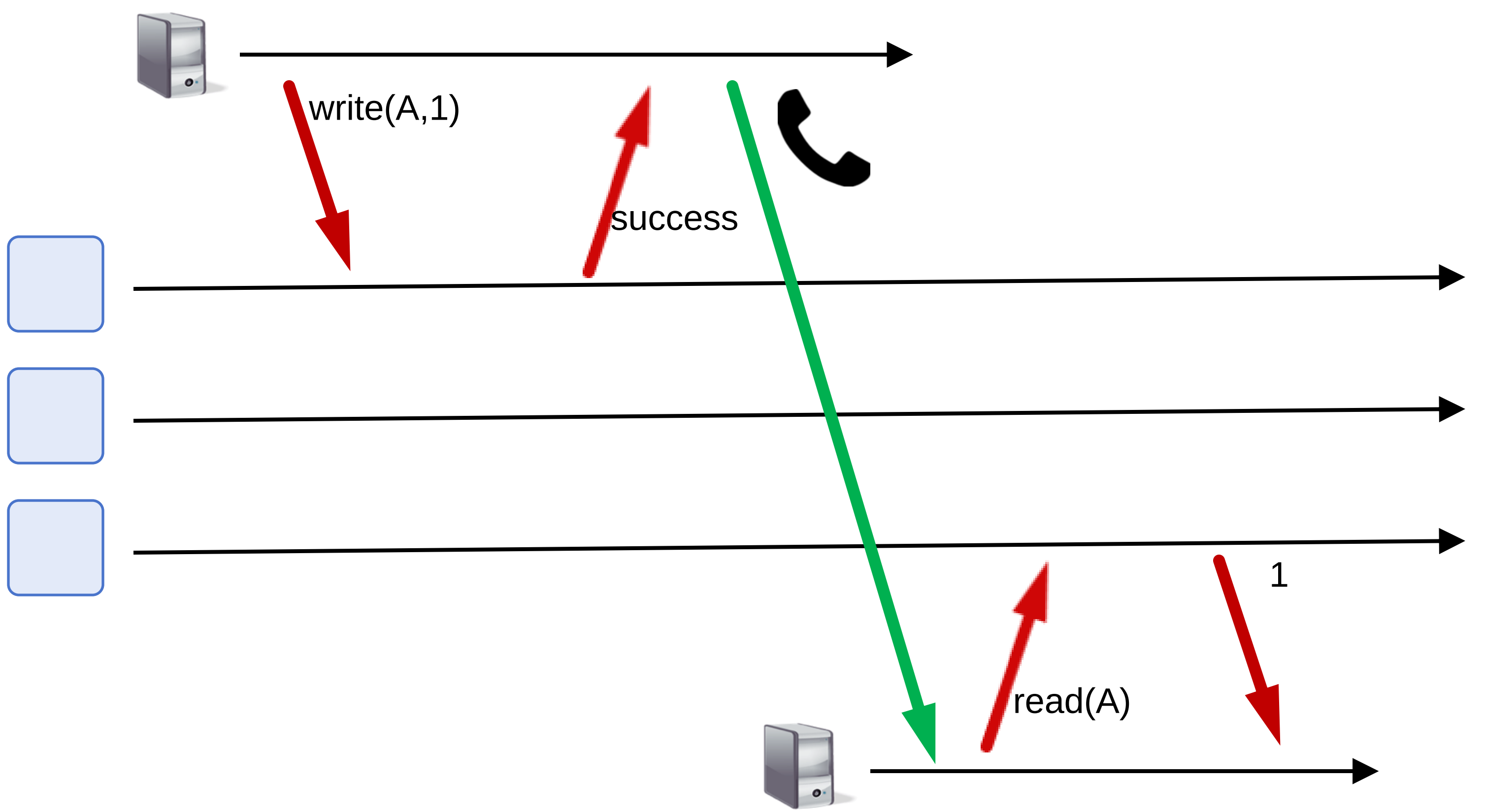
- Telephone intuition:
  1. Alice updates Facebook post
  2. Alice calls Bob on phone: “Check my Facebook post!”
  3. Bob read’s Alice’s wall, sees her post

Strong consistency?



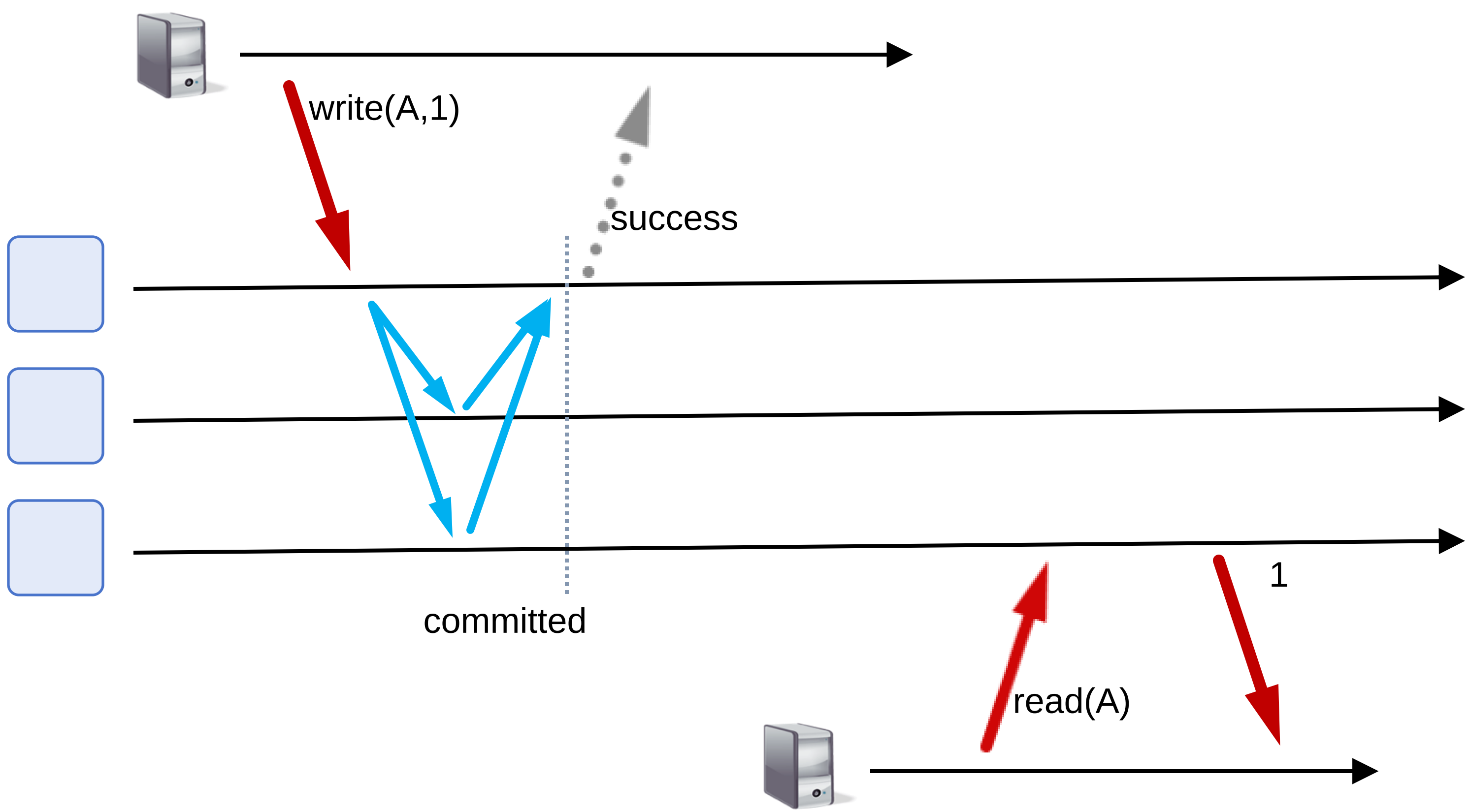
Phone call: Ensures *happens-before* relationship, even through “out-of-band” communication

Strong consistency?



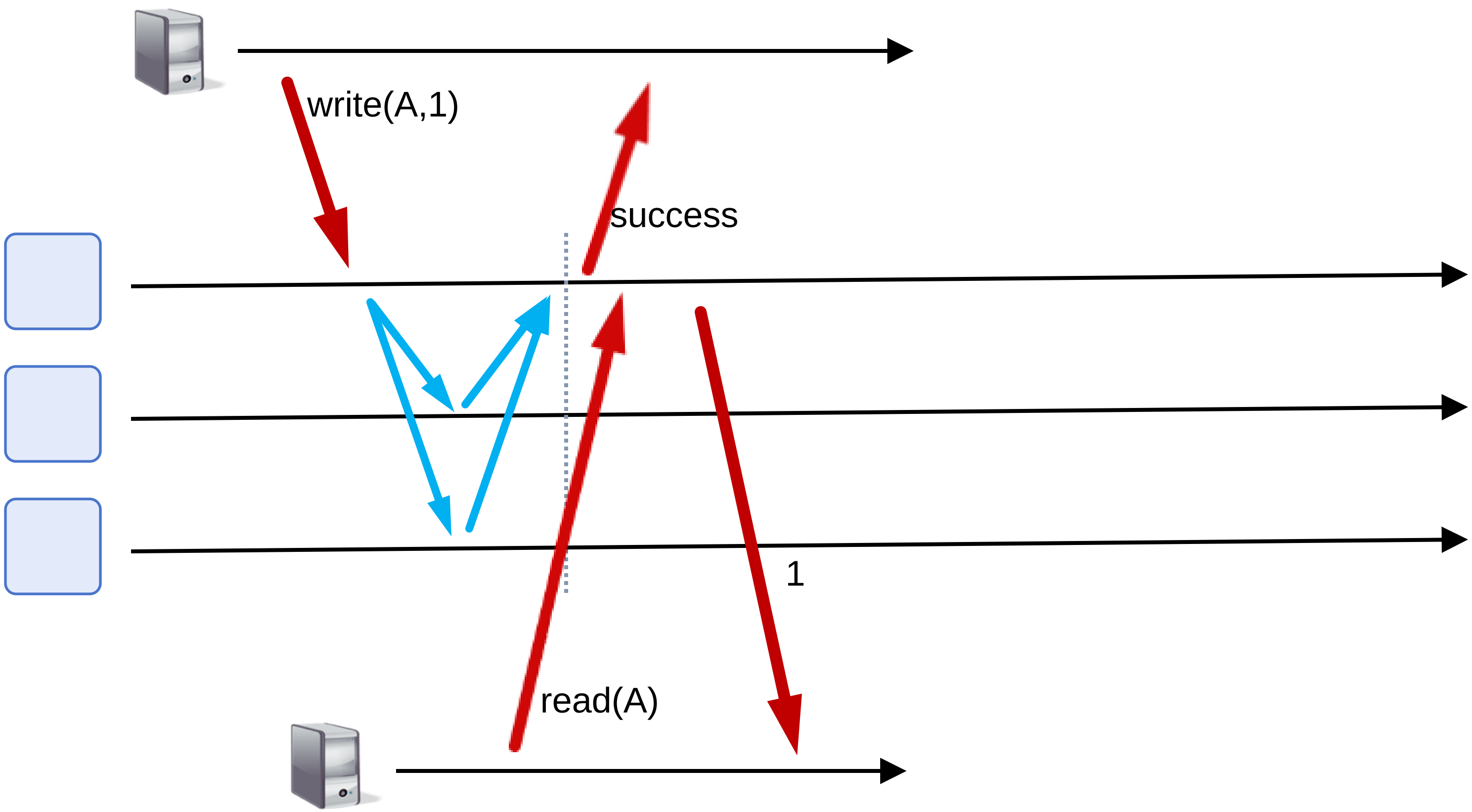
One cool trick: Delay responding to writes/ops until properly committed

Strong consistency? This is buggy!



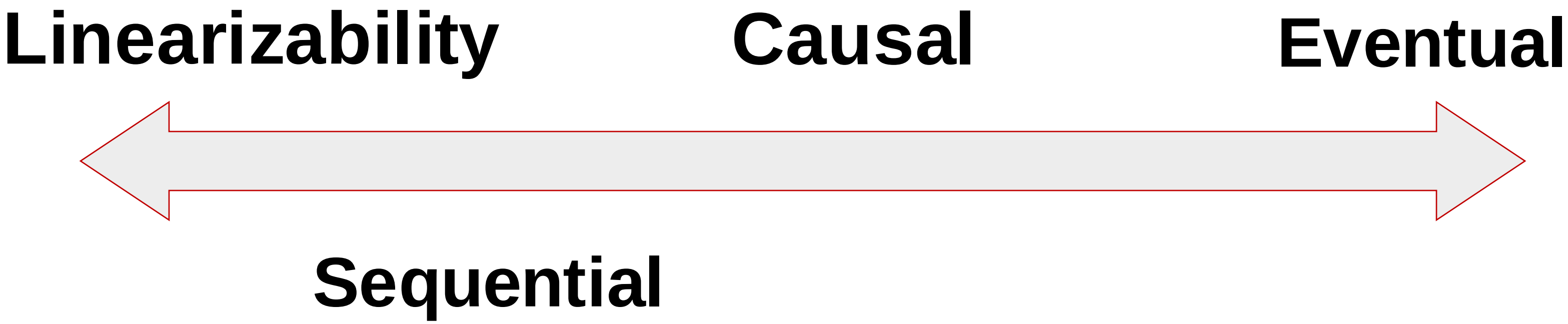
- Isn't sufficient to return value of third node: It doesn't know precisely when op is "globally" committed
- Instead: Need to actually *order* read operation

Strong consistency!

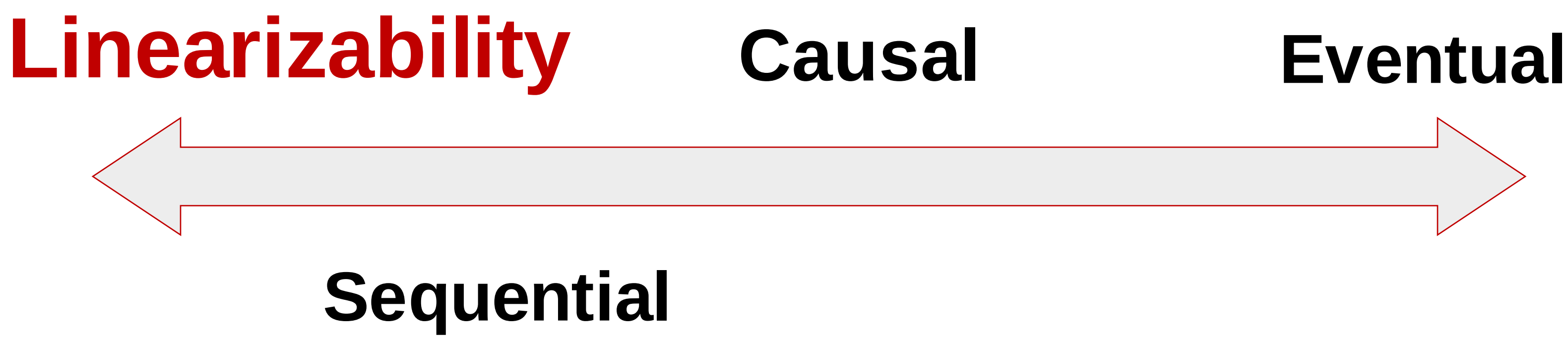


Order all operations via (1) leader, (2) consensus

Consistency models



Consistency models



Strong consistency = linearizability

- Linearizability (Herlihy and Wing 1991)
  1. All servers execute all ops in ***some*** identical sequential order
  2. Global ordering preserves each client's own local ordering
  3. Global ordering preserves **real-time guarantee**

Informally, linearizability specifies that each concurrent operation **appears** to occur **instantaneously** and **exactly once** at some point in time between its invocation and its completion.

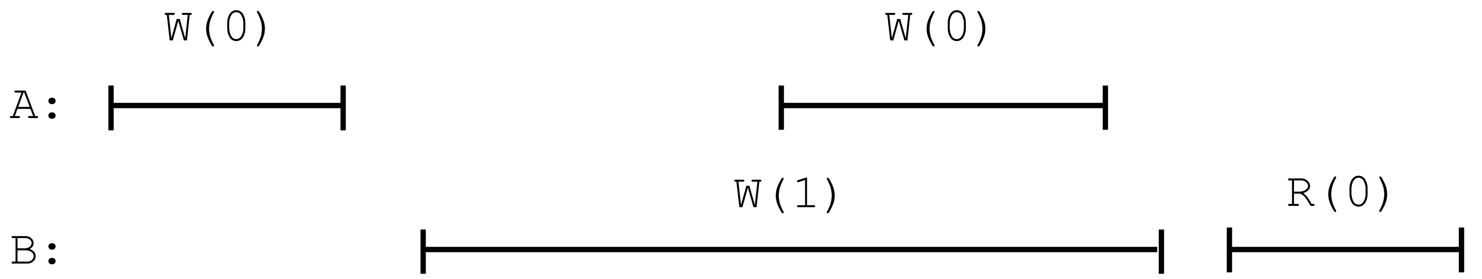
Strong consistency = linearizability

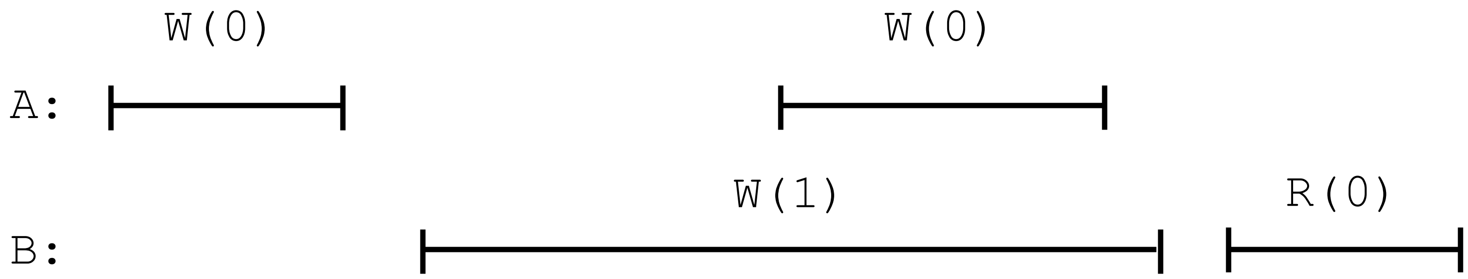
- Linearizability (Herlihy and Wing 1991)
  1. All servers execute all ops in *some* identical sequential order
  2. Global ordering preserves each client's own local ordering
  3. Global ordering preserves **real-time guarantee**
     - All ops receive global timestamp using a sync'd clock
     - If $ts_{op1}(x) < ts_{op2}(y)$, OP1(x) precedes OP2(y) in sequence

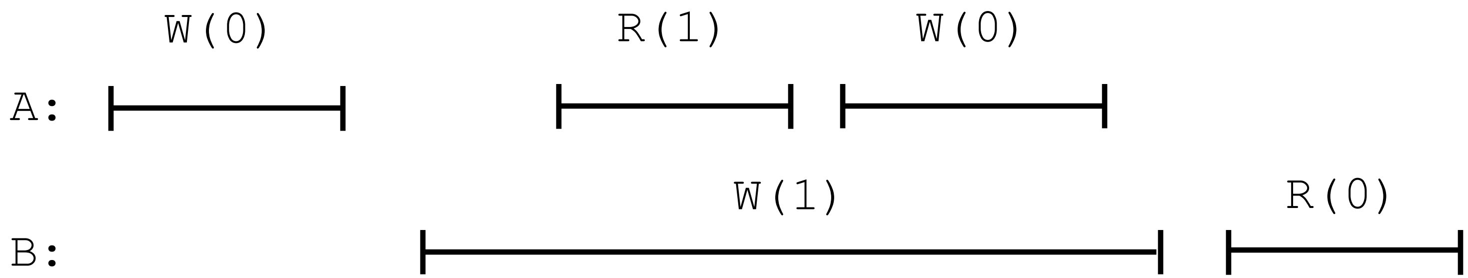
Strong consistency = linearizability

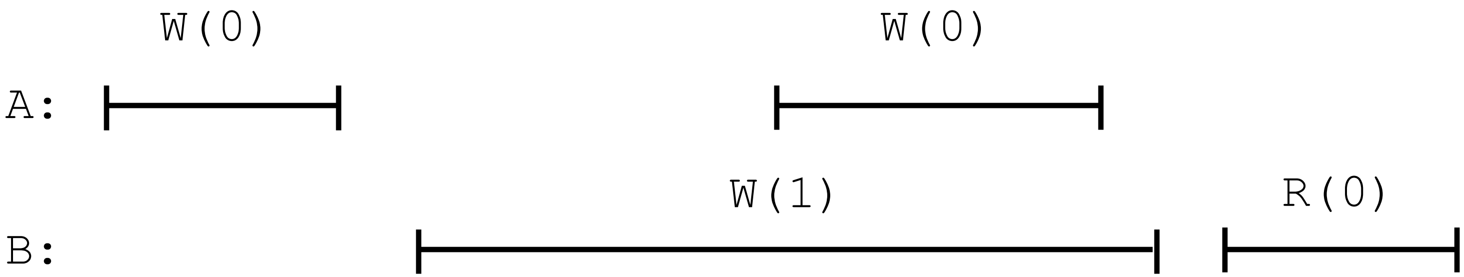
- Linearizability (Herlihy and Wing 1991)
  1. All servers execute all ops in *some* identical sequential order
  2. Global ordering preserves each client's own local ordering
  3. Global ordering preserves **real-time guarantee**
     - All ops receive global timestamp using a sync'd clock
     - If $ts_{op1}(x) < ts_{op2}(y)$, OP1(x) precedes OP2(y) in sequence
- Once write completes, all later reads (by wall-clock start time) should return value of that write or value of later write.
- Once read returns particular value, all later reads should return that value or value of later write.

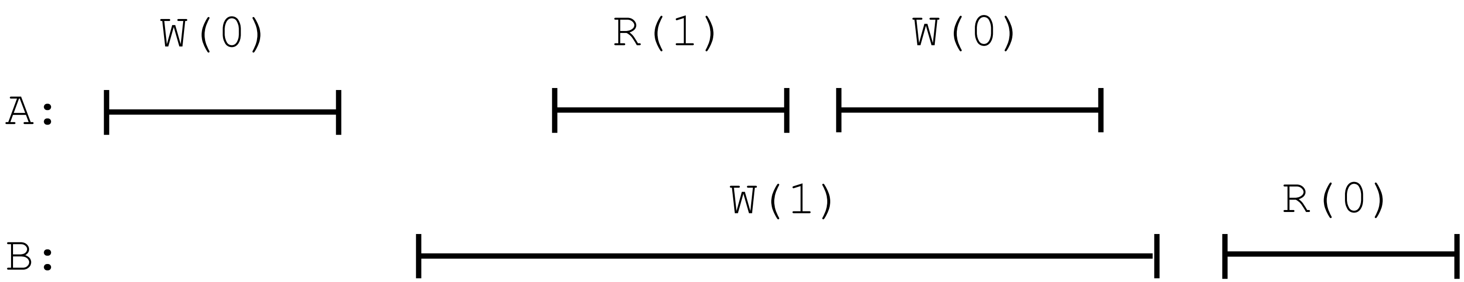
Real-time ordering examples

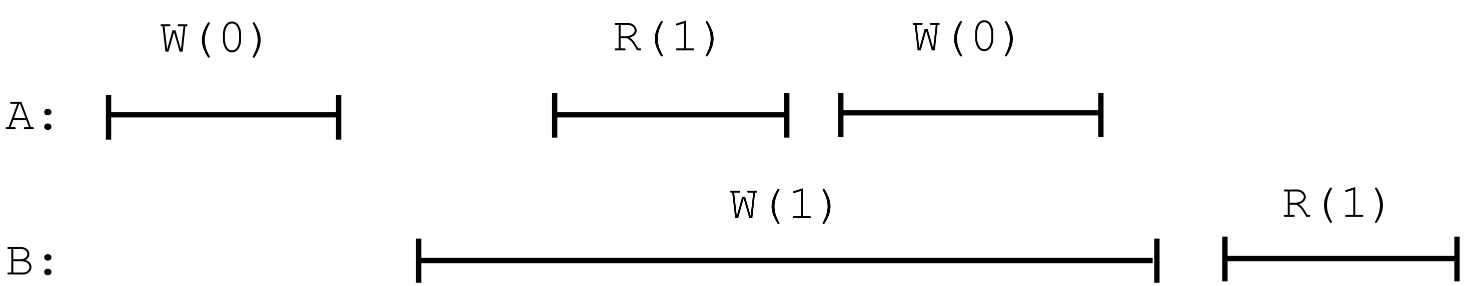

W(0)
W(0)
A:
W(1)
R(0)
B:


W(0)
W(0)
A:
W(1)
R(0)
B:

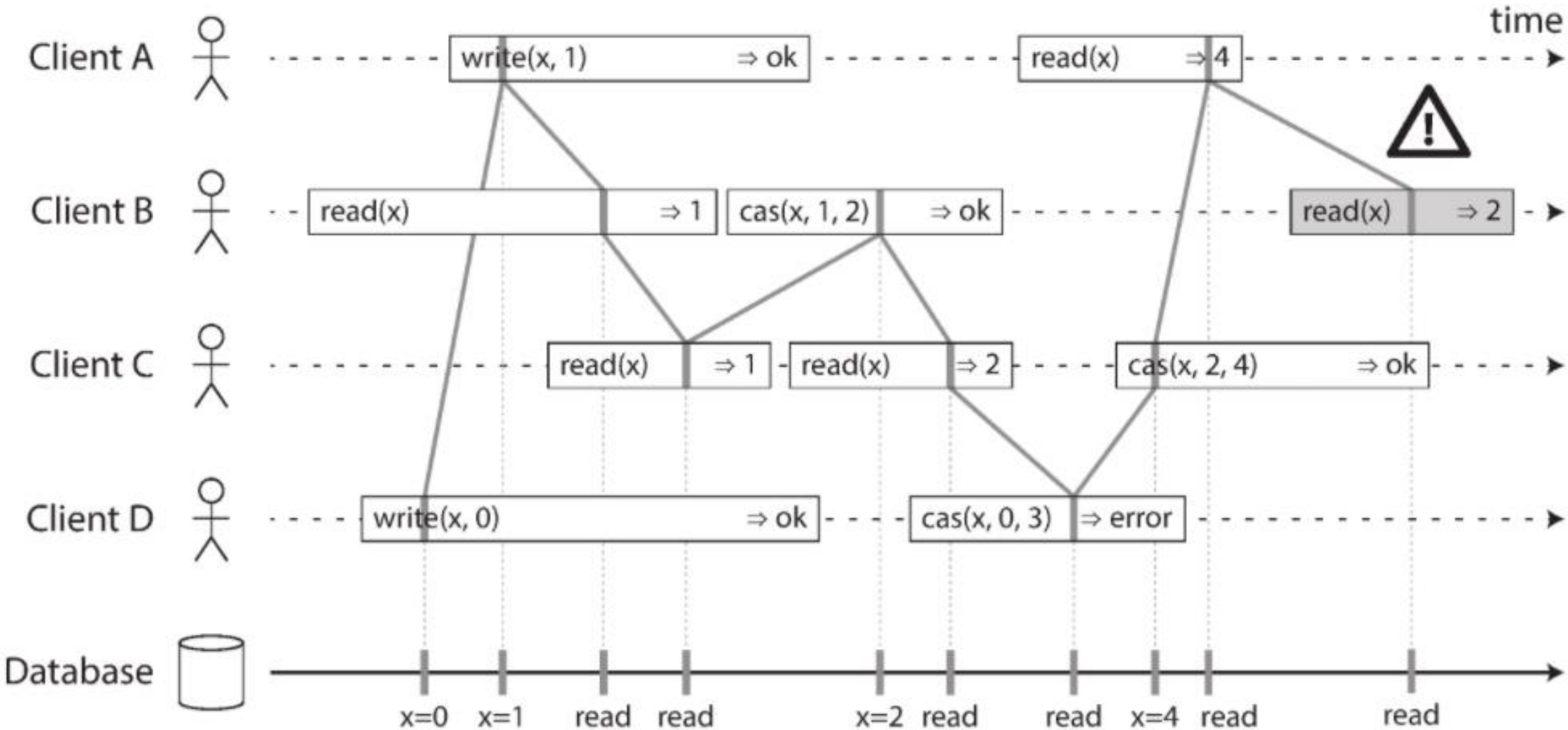

W(0)
R(1)
W(0)
A:
W(1)
R(0)
B:


W(0)
W(0)
A:
W(1)
R(0)
B:


W(0)
R(1)
W(0)
A:
W(1)
R(0)
B:

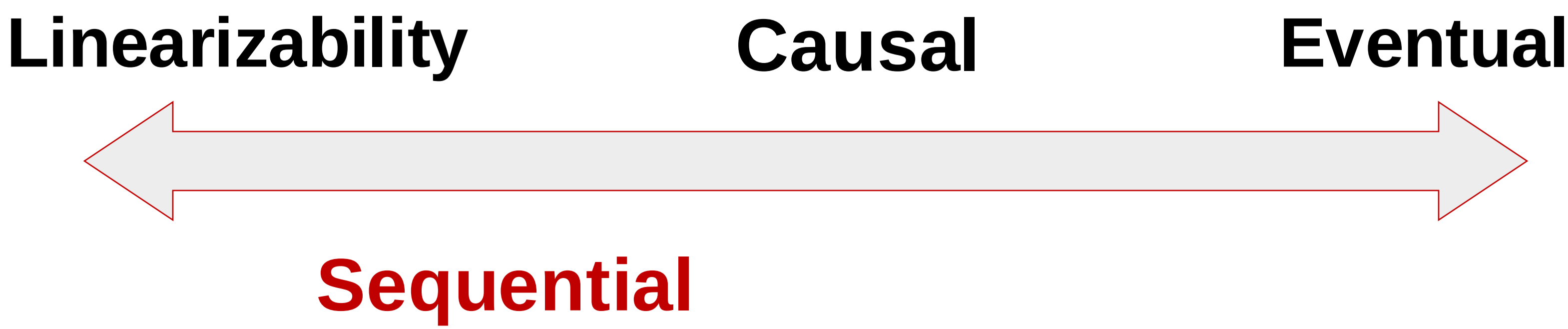

W(0)
R(1)
W(0)
A:
W(1)
R(1)
B:

Real-time ordering examples



*: https://www.oreilly.com/library/view/designing-data-intensive-applications/9781491903063/ (Page 328)

Consistency models



Weaker: Sequential consistency

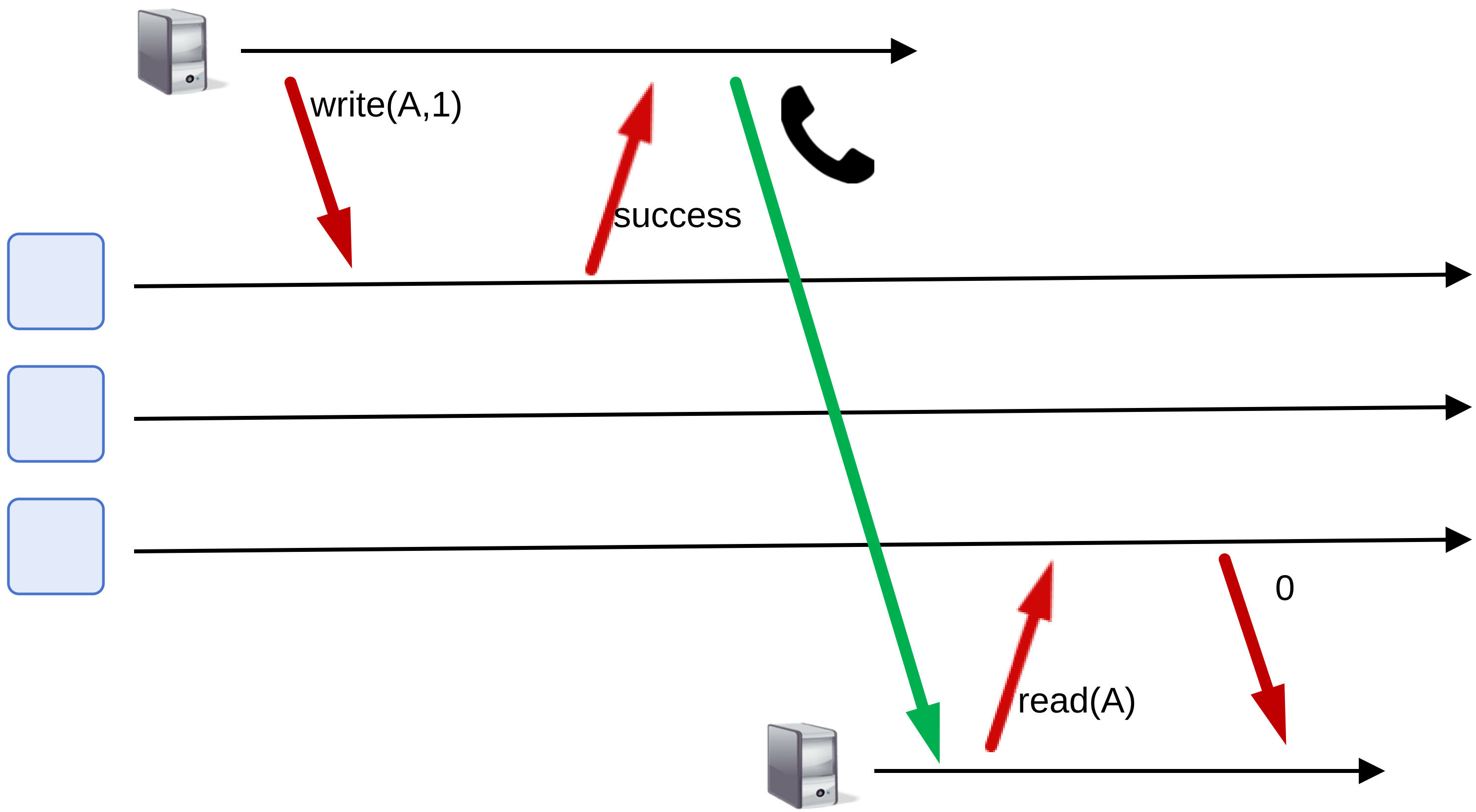
- Sequential = Linearizability – real-time ordering
  1. All servers execute all ops in *some* identical sequential order
  2. Global ordering preserves each client's own local ordering

Weaker: Sequential consistency

- Sequential = Linearizability – real-time ordering
    1. All servers execute all ops in *some* identical sequential order
    2. Global ordering preserves each client's own local ordering

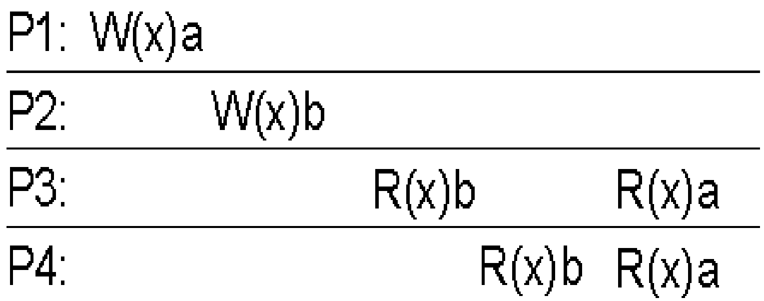
- With concurrent ops, "reordering" of ops (w.r.t. real-time ordering) acceptable, but all servers must see same order
    - e.g., linearizability cares about time
      sequential consistency cares about program order

Sequential consistency

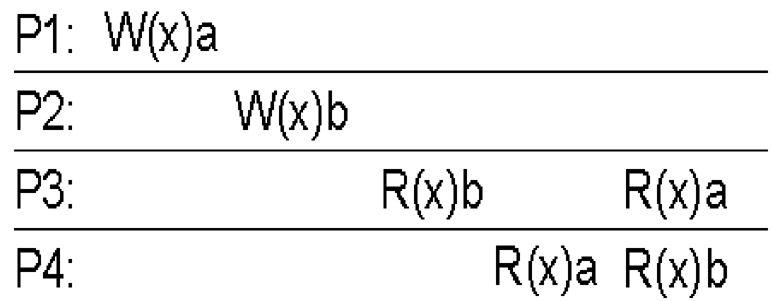


In example, system orders read(A) before write(A,1)

Valid sequential consistency?

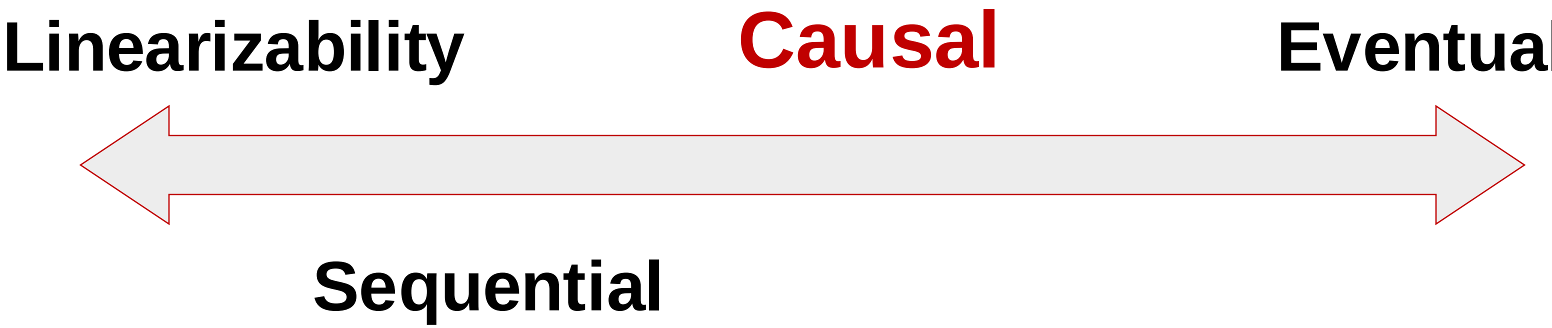


(a)



(b)

Consistency models



Recall use of logical clocks (lec 8?)

- 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 8?)

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

- 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.

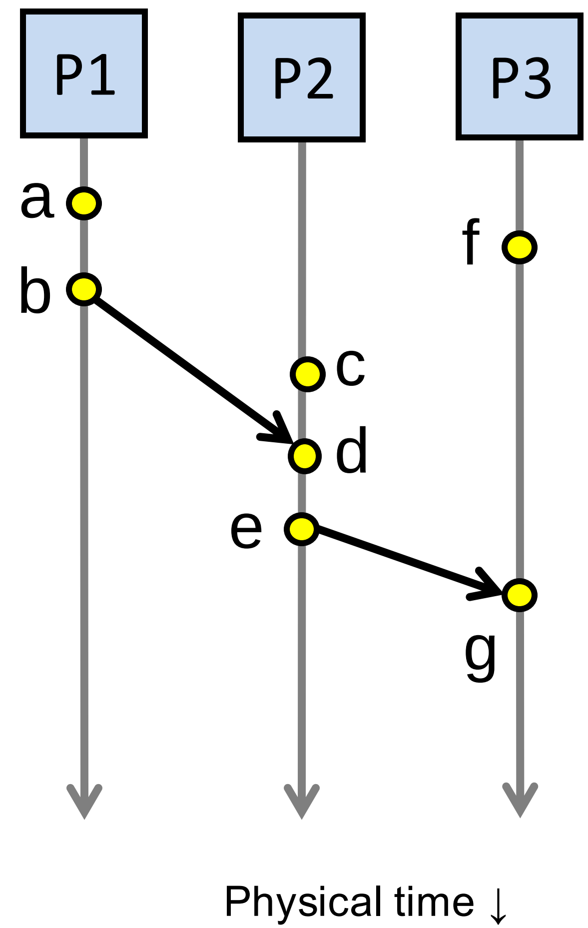
Concurrent: Ops not causally related

Causal consistency

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

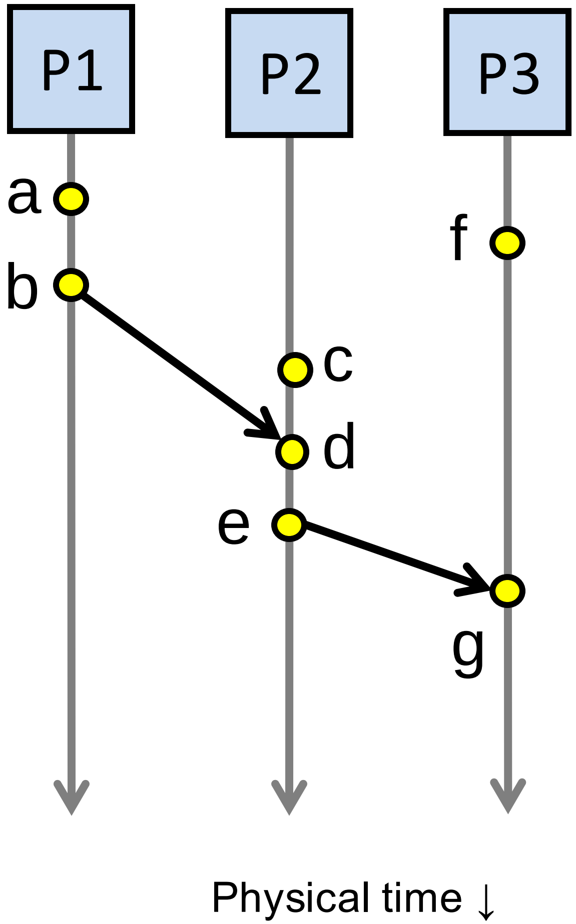
2. Concurrent writes may be seen in a different order on different machines.

Concurrent: Ops not causally related



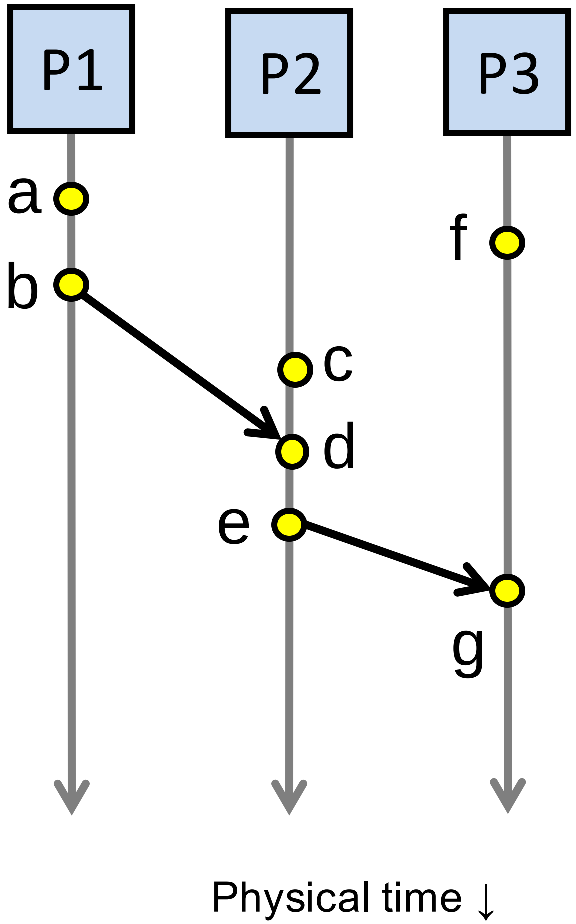
Causal consistency

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



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**



Consistency models

Strong consistency **Eventual consistency**

Paxos / Raft **Amazon Dynamo**

Next class