

Consistency



COS 316: Principles of Computer System Design

Lecture 12

Wyatt Lloyd

Why Do We Build Systems?

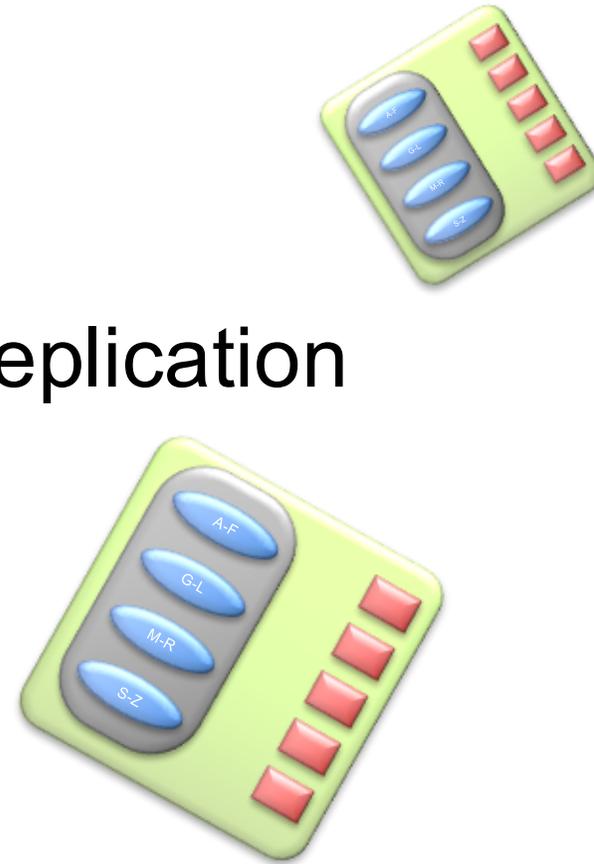
- ...
- Abstract away complexity

Distributed Systems are Highly Complex Internally

Sharding



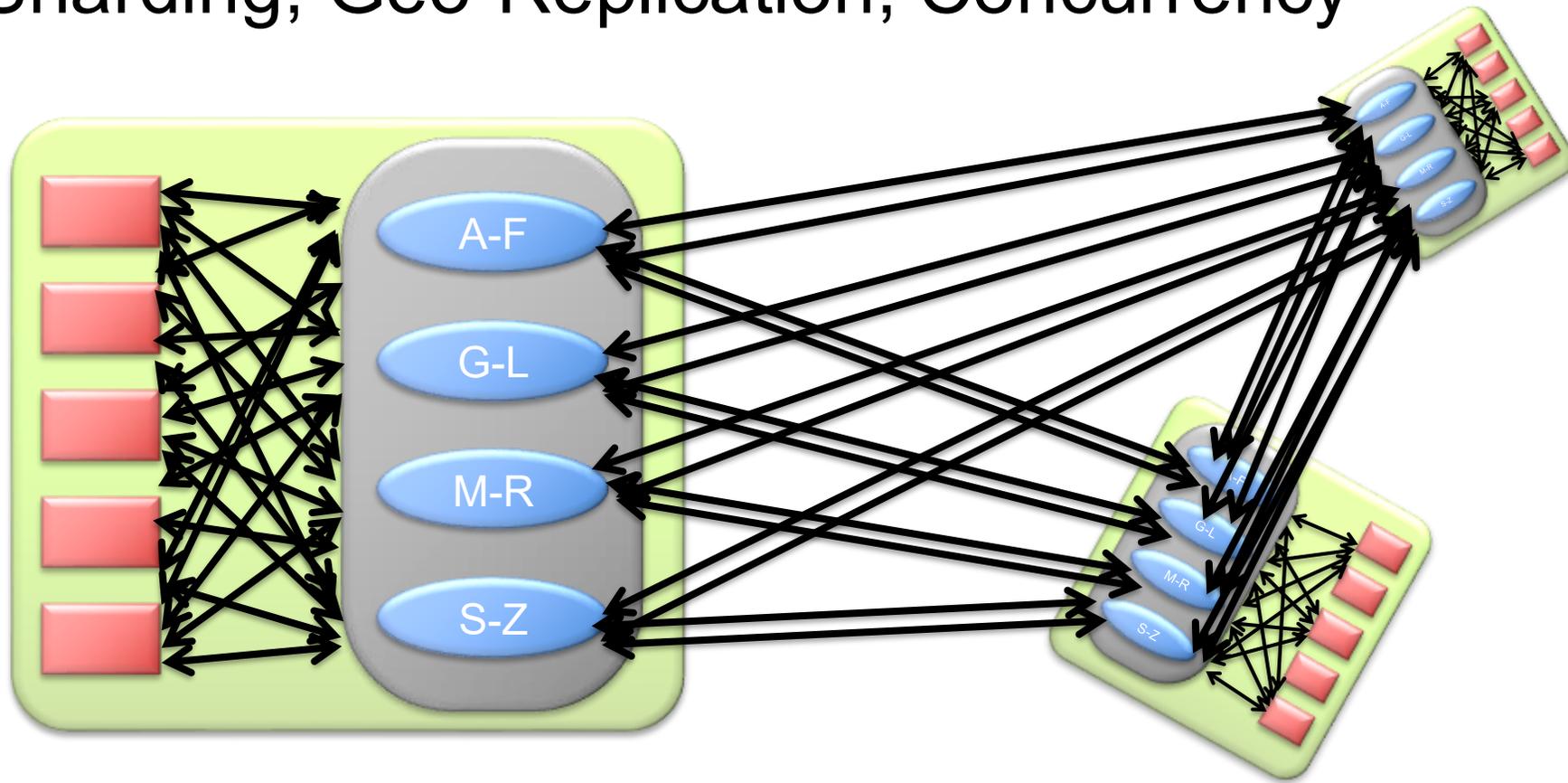
(Geo)-Replication



Concurrent access by many client

Distributed Systems are Highly Complex Internally

Sharding, Geo-Replication, Concurrency



Distributed Systems are Highly Complex Internally

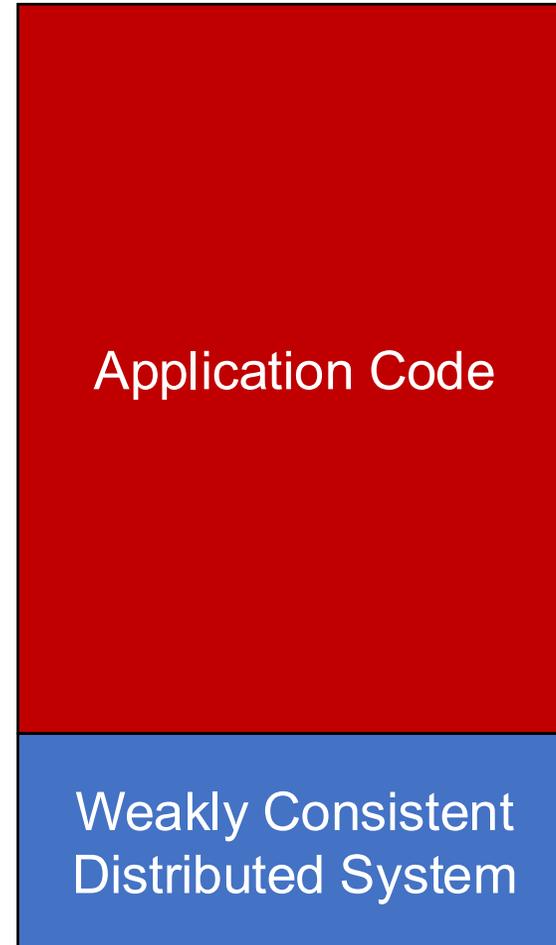
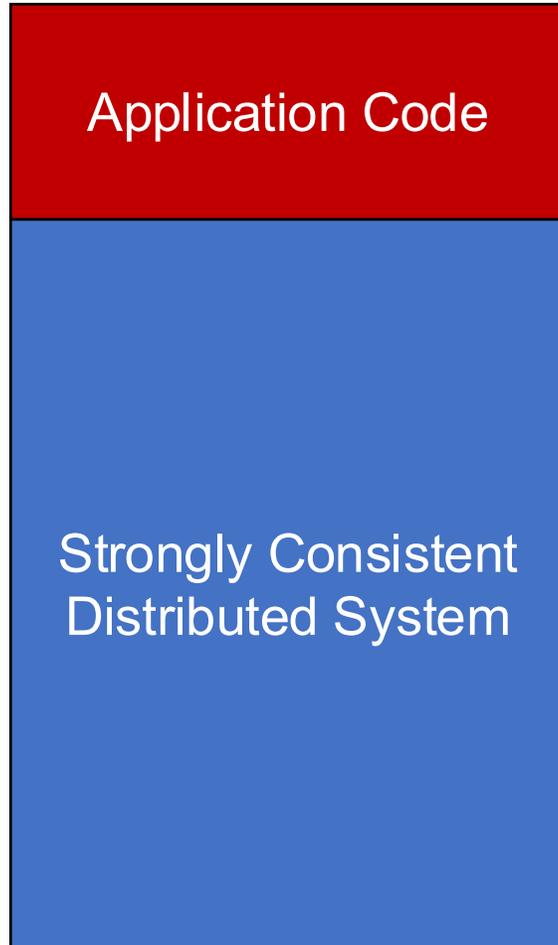
Sharding, Geo-Replication, Concurrency



Consistency Models

- Contract between a (distributed) system and the applications that run on it
- A consistency model is a set of **guarantees** made by the distributed system

Stronger vs Weaker Consistency



Stronger vs Weaker Consistency

- Stronger consistency models
 - + Easier to write applications
 - System must hide many behaviors
- Fundamental tradeoffs between consistency & performance
 - (Discuss CAP, PRAM, SNOW in 418!)
- Weaker consistency models
 - Harder to write applications
 - Cannot (reasonably) write some applications
 - + System needs to hide few behaviors

Consistency Hierarchy

Linearizability

Behaves like a single machine



Causal+ Consistency

Everyone sees related operations in the same order



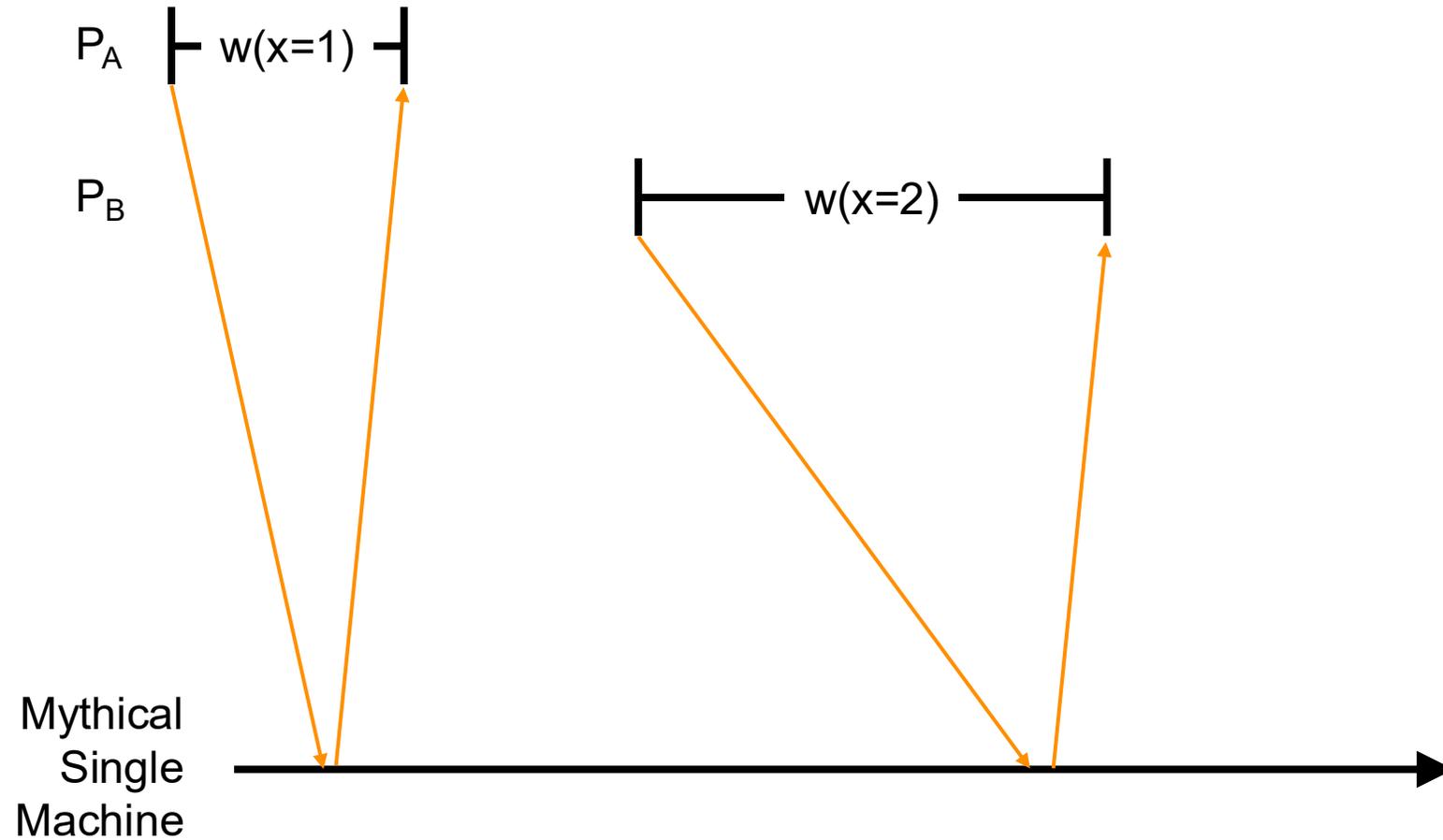
Eventual Consistency

Anything goes

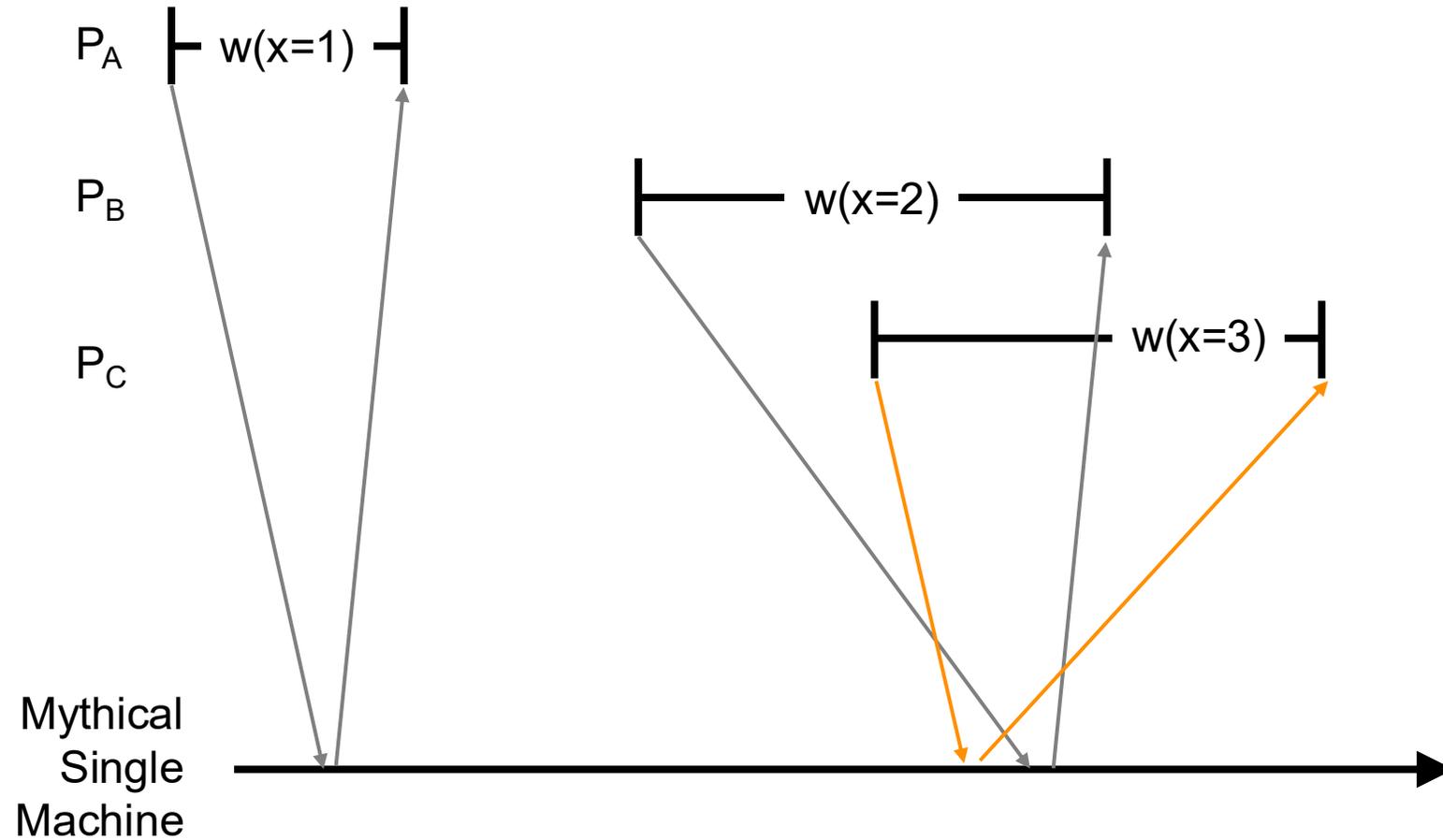
Linearizability == “Appears to be a Single Machine”

- External client submitting requests and getting responses from the system can't tell this is not a single machine!
- There is some **total order** over all operations
 - Processes all requests one by one
- Order preserves the **real-time ordering** between operations
 - If operation A **completes** before operation B **begins**, then A is ordered before B in real-time
 - If neither A nor B completes before the other begins, then there is no real-time order
 - (But there must be *some* total order)

Real-Time Ordering Examples



Real-Time Ordering Examples



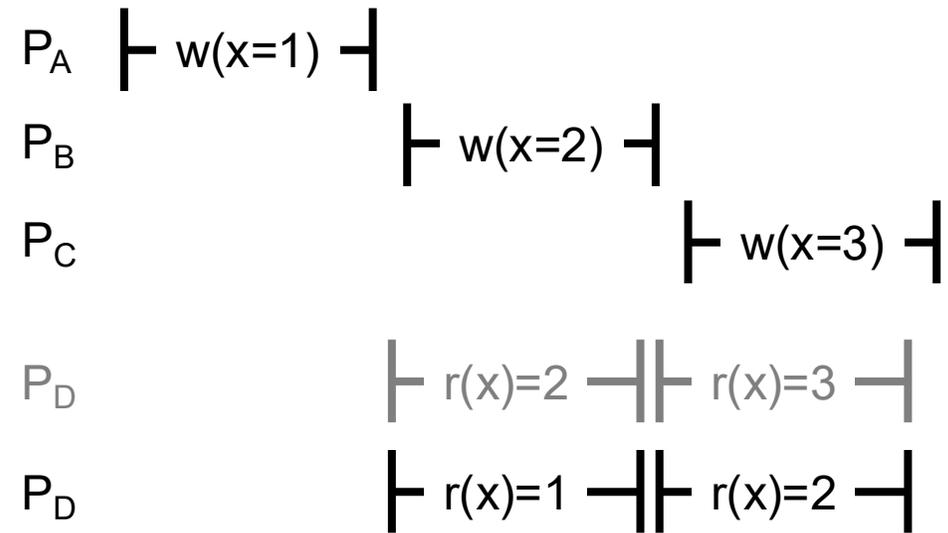
Linearizable?

$P_A \quad \vdash w(x=1) \dashv$
 $P_B \quad \quad \quad \vdash w(x=2) \dashv$
 $P_C \quad \quad \quad \quad \quad \vdash w(x=3) \dashv$
 $P_D \quad \quad \quad \vdash r(x)=2 \dashv \dashv \vdash r(x)=3 \dashv$



w_1, w_2, r_2, w_3, r_3

Linearizable?



w_1, r_1, w_2, r_2, w_3

Linearizable?

$P_A \quad \vdash w(x=1) \dashv$

$P_B \quad \quad \quad \vdash w(x=2) \dashv$

$P_C \quad \quad \quad \quad \quad \vdash w(x=3) \dashv$

$P_D \quad \quad \quad \vdash r(x)=2 \dashv \parallel \vdash r(x)=3 \dashv$

$P_D \quad \quad \quad \vdash r(x)=1 \dashv \parallel \vdash r(x)=2 \dashv$

$P_D \quad \quad \quad \vdash r(x)=2 \dashv \parallel \vdash r(x)=2 \dashv$



W_1, W_2, r_2, r_2, W_3

Linearizable?

$P_A \quad \vdash w(x=1) \dashv$

$P_B \quad \quad \quad \vdash w(x=2) \dashv$

$P_C \quad \quad \quad \quad \quad \vdash w(x=3) \dashv$

$P_D \quad \quad \quad \vdash r(x)=2 \dashv \parallel \dashv r(x)=3 \dashv$

$P_D \quad \quad \quad \vdash r(x)=1 \dashv \parallel \dashv r(x)=2 \dashv$

$P_D \quad \quad \quad \vdash r(x)=2 \dashv \parallel \dashv r(x)=2 \dashv$

$P_D \quad \quad \quad \vdash r(x)=1 \dashv \parallel \dashv r(x)=3 \dashv$



w_1, r_1, w_2, w_3, r_3

Linearizable?

$P_A \vdash w(x=1) \dashv$

$P_B \vdash w(x=2) \dashv$

$P_C \vdash w(x=3) \dashv$

$P_D \vdash r(x)=2 \dashv \parallel \dashv r(x)=3 \dashv$



$P_D \vdash r(x)=1 \dashv \parallel \dashv r(x)=2 \dashv$



$P_D \vdash r(x)=2 \dashv \parallel \dashv r(x)=2 \dashv$



$P_D \vdash r(x)=1 \dashv \parallel \dashv r(x)=3 \dashv$



$P_D \vdash r(x)=2 \dashv \parallel \dashv r(x)=1 \dashv$



Linearizable?

$P_A \vdash w(x=1) \dashv$

$P_B \quad \vdash w(x=2) \dashv$

$P_C \quad \quad \quad \vdash w(x=3) \dashv$

$P_D \quad \quad \quad \vdash w(x=4) \dashv \quad \vdash w(x=5) \dashv$

$P_E \quad \quad \quad \quad \quad \quad \vdash w(x=6) \dashv$

$P_F \quad \quad \quad \vdash r(x)=2 \dashv \dashv \vdash r(x)=3 \dashv \dashv \vdash r(x)=6 \dashv \dashv \vdash r(x)=5 \dashv \quad \checkmark$

$w_1, w_2, r_2, w_4, w_3, r_3, w_6, r_6, w_5, r_5$

OR

$w_1, \mathbf{w}_4, w_2, r_2, w_3, r_3, w_6, r_6, w_5, r_5$

OR

$w_1, w_2, r_2, w_3, r_3, \mathbf{w}_4, w_6, r_6, w_5, r_5$

Linearizable?

$P_A \vdash w(x=1) \dashv$

$P_B \quad \vdash w(x=2) \dashv$

$P_C \quad \quad \quad \vdash w(x=3) \dashv$

$P_D \quad \quad \quad \vdash w(x=4) \dashv \quad \vdash w(x=5) \dashv$

$P_E \quad \quad \quad \quad \quad \quad \vdash w(x=6) \dashv$

$P_G \quad \quad \quad \vdash r(x)=2 \dashv \dashv \vdash r(x)=5 \dashv \dashv \vdash r(x)=6 \dashv \dashv \vdash r(x)=5 \dashv \dashv \quad \times$

Linearizable?

$P_A \vdash w(x=1) \dashv$

$P_B \quad \vdash w(x=2) \dashv$

$P_C \quad \quad \quad \vdash w(x=3) \dashv$

$P_D \quad \quad \quad \vdash w(x=4) \dashv \quad \vdash w(x=5) \dashv$

$P_E \quad \quad \quad \quad \quad \quad \vdash w(x=6) \dashv$

$P_H \quad \quad \quad \vdash r(x)=4 \dashv \dashv \vdash r(x)=2 \dashv \dashv \vdash r(x)=3 \dashv \dashv \vdash r(x)=6 \dashv \quad \checkmark$

$W_1, W_4, r_4, W_2, r_2, W_3, r_3, W_5, W_6, r_6$

Linearizable?

$P_A \vdash w(x=1) \dashv$

$P_B \quad \quad \quad \vdash w(x=2) \dashv$

$P_C \quad \quad \quad \quad \quad \quad \vdash r(x)=1 \dashv \quad \quad \quad \mathbf{X}$

Linearizability == “Appears to be a Single Machine”

- There is some **total order** over all operations
 - Processes all requests one by one
- Order preserves the **real-time ordering** between operations
 - If operation A **completes** before operation B **begins**, then A is ordered before B in real-time
 - If neither A nor B completes before the other begins, then there is no real-time order
 - (But there must be *some* total order)

How to Provide Linearizability?

1. Use a single machine 😊
2. Use “state-machine replication” on top of a consensus protocol like Paxos
 - Distributed system appears to be single machine that does not fail!!
 - Discussed in next class
 - Covered extensively in 418
3. ...

Consistency Hierarchy

Linearizability

Behaves like a single machine



Causal+ Consistency

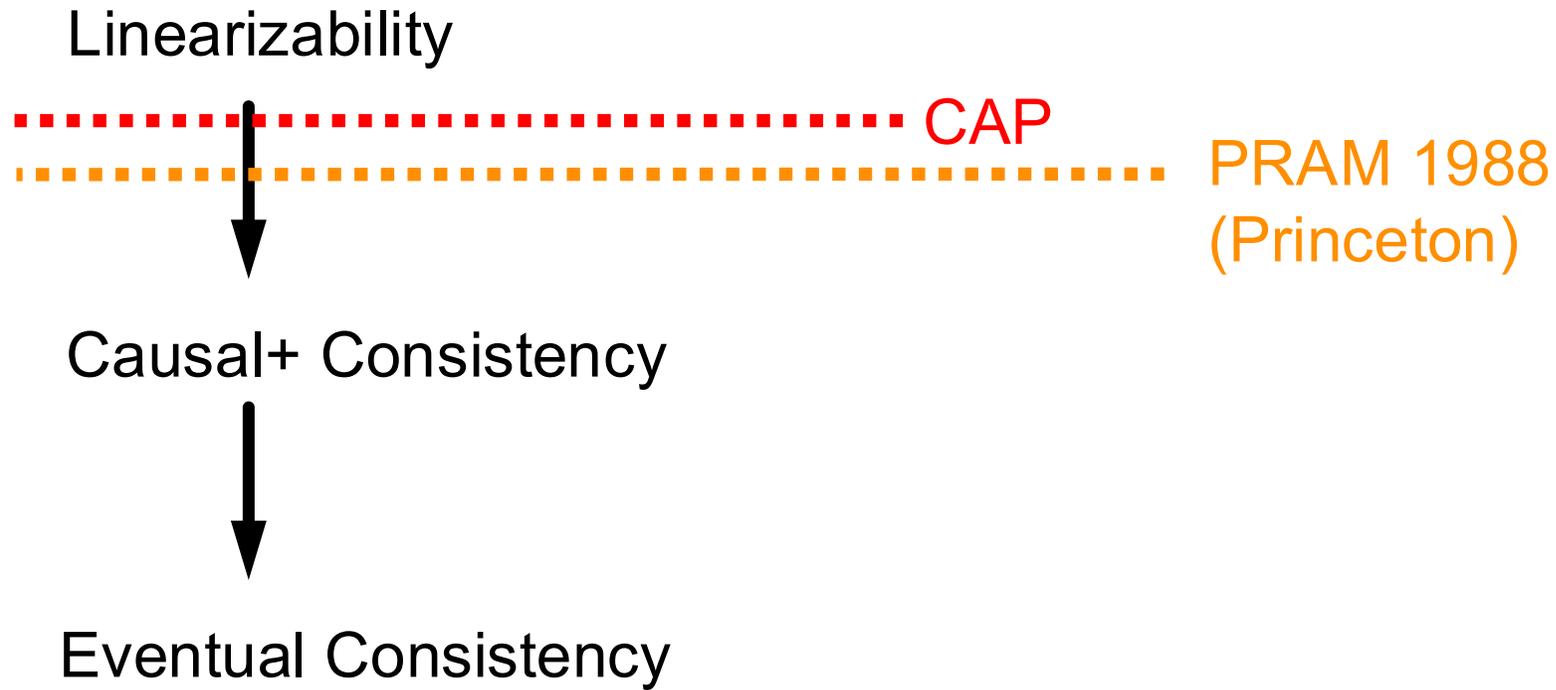
Everyone sees related operations in the same order



Eventual Consistency

Anything goes

Consistency Hierarchy



Causal+ Consistency Informally

1. Writes that are **potentially** causally related must be seen by everyone in the same order.
 2. Concurrent writes may be seen in a different order by different entities.
 - Concurrent: Writes not causally related
- Potential causality: event *a* **could** have a causal effect on event *b*.
 - Think: is there a path of information from *a* to *b*?
 - *a* and *b* done by the same entity (e.g., me)
 - *a* is a write and *b* is a read of that write
 - + transitivity

Causal+ Sufficient



↓ Then ↓



**Employment
retained**



↓ Then ↓



**Purchas
e
retained**



↓ Then ↓



**Deletion
retained**

Causal+ Sufficient



↓ Then ↓



Wvatt likes
My Little Pony.

↓ Then ↓



• **I am a new customer.**
(You'll create a password later)

Sign in using our secure server ▶

↓ Then ↓



Causal+ Not Sufficient

(Need Linearizability)

- Need a total order of operations
 - e.g., Alice's bank account ≥ 0
- Need a real-time ordering of operations
 - e.g., Alice changes her password, Bob cannot login with old password

Consistency Hierarchy

Linearizability

Behaves like a single machine



Causal+ Consistency

Everyone sees related operations in the same order



Eventual Consistency

Anything goes

Eventual Consistency

- Anything goes for now...
 - (If updates stop, eventually all copies of the data are the same)
- But, eventually consistent systems often try to provide consistency and often do
 - e.g., Facebook's TAO system provided linearizable results 99.9994% of the time [Lu et al. SOSP '15]
- “Good enough” sometimes
 - e.g., 99 vs 100 likes

Consistency Model Summary

- Consistency model specifies strength of abstraction
 - Linearizability \rightarrow Causal+ \rightarrow Eventual
 - Stronger hides more, but has worse performance
- When building an application, what do you need?
 - Select system(s) with necessary consistency
 - Always safe to pick stronger
- When building a system, what are your guarantees?
 - Must design system such that they always hold
 - Must confront fundamental tradeoffs with performance
 - What is more important?