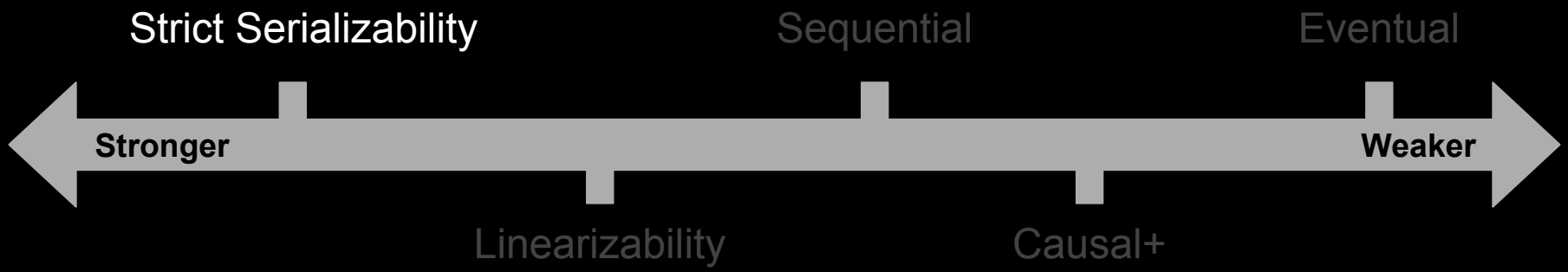
Consistency

11/16/2018

Consistency Models

Consistency Models



Strict Serializability

- Total order: There exists a legal total ordering of transactions.
  - Legal: In the total ordering, a read operation sees the latest write operation.
- Preserves real-time ordering: Any transaction *A* that completes before transaction *B* begins, occurs before *B* in the total order.
- Properties
  - Writes in a completed transaction appear to all future reads
  - Once a read sees a value, all future reads must also return the same value (until new write)

Pros: Easily reason about correctness of transactions

Cons: High read and write latencies

Strict Serializability Example

Strictly Serializable? **Yes**

P1: {W(x)b, W(y)b}

P2: {W(x)a}

P3: {R(x)a} {R(x)b}

P4: {R(x)b} {R(y)b}

Strictly Serializable? **No**

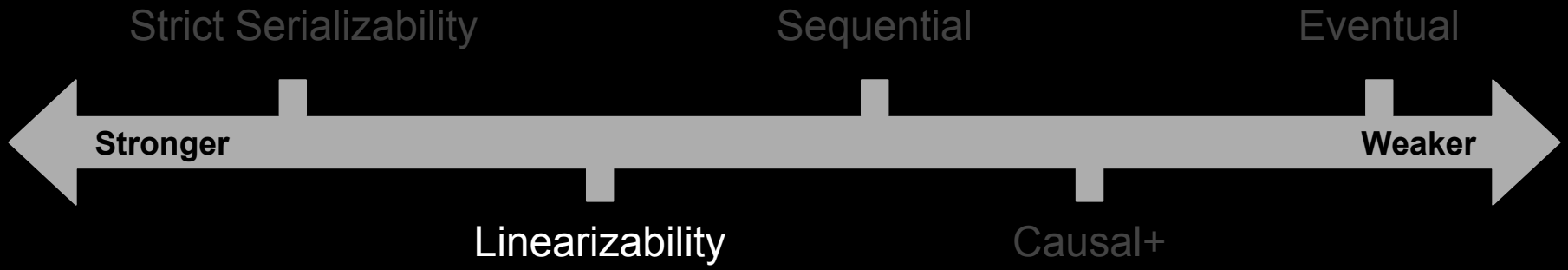
P1: {W(x)b, W(y)b}

P2: {W(x)a}

P3: {R(y)b} {R(x)a}

P4: {R(x)b} {R(y)b}

Consistency Models



Linearizability

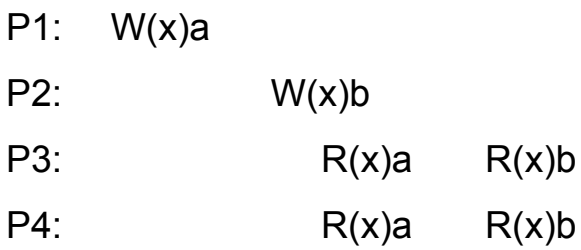
- Total order: There exists a legal total ordering of operations
  - Legal: In the total ordering, a read operation sees the latest write operation.
- Preserves real-time ordering: Any operation *A* that completes before operation *B* begins, occurs before *B* in the total order.
- Difference from *strict serializability*?
  - In Linearizability, clients only have consistency guarantees for operations, where strict serializability allows clients to use transactions.
- Properties
  - A completed write appears to all future reads
  - Once a read sees a value, all future reads must also return the same value (until new write)

Pros: Easy to reason about correctness

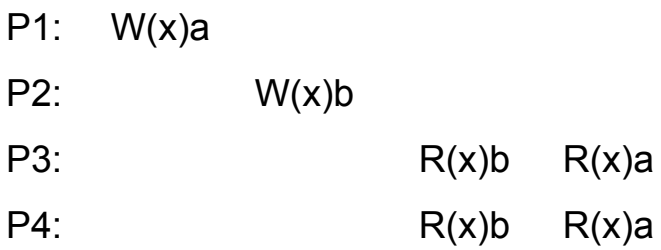
Cons: High read and write latencies

Linearizability Example

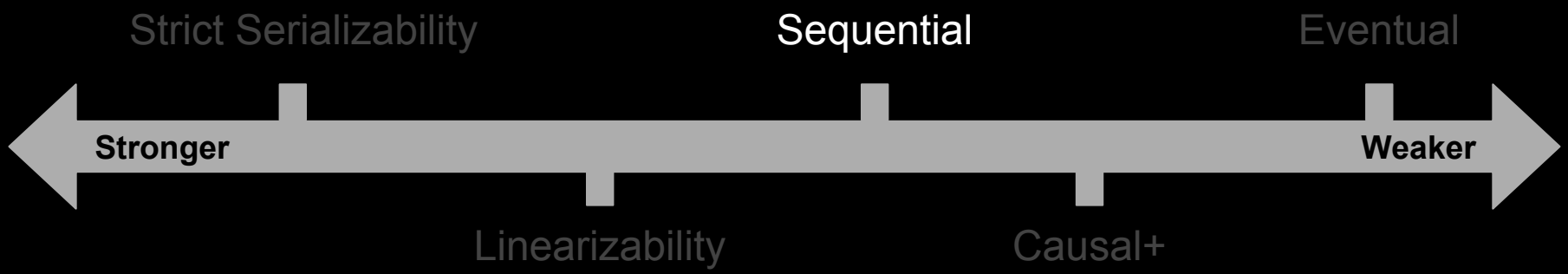
Linearizable? **Yes**



Linearizable? **No**



Consistency Models



Sequential Consistency

- Total order: There exists a legal total ordering of operations.
  - Legal: In the total ordering, a read operation sees the latest write operation.
- Preserves process ordering: All of a process' operations appear in that order in the total order.
- Difference from *linearizability*?
  - Sequence of ops across processes not determined by real-time

Pros: Can allow more orderings than linearizability

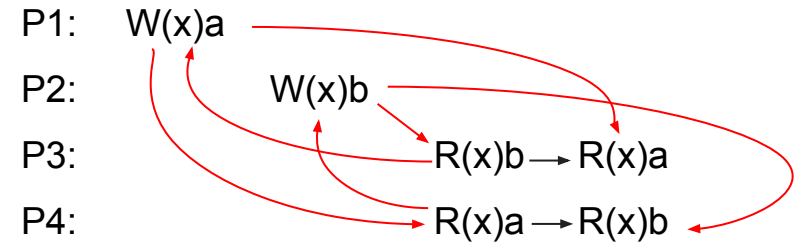
Cons: Many possible sequential executions

Sequential Consistency Example

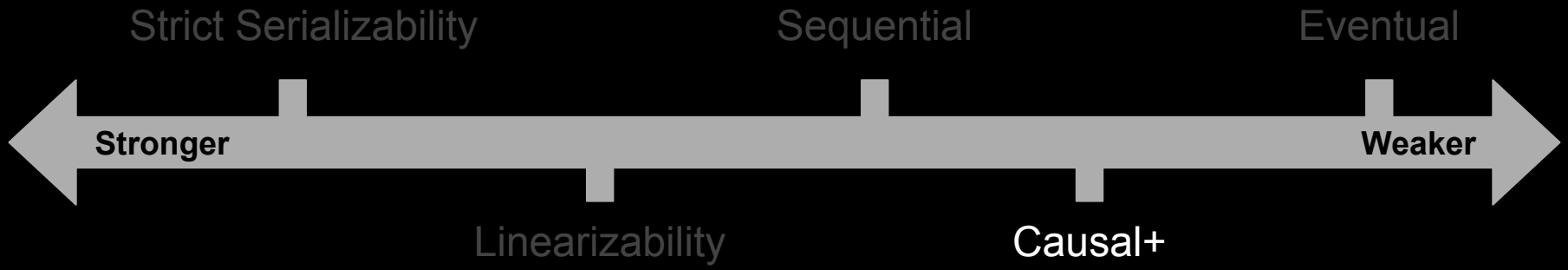
Sequentially Consistent? Yes

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

Sequentially Consistent? No



Consistency Models



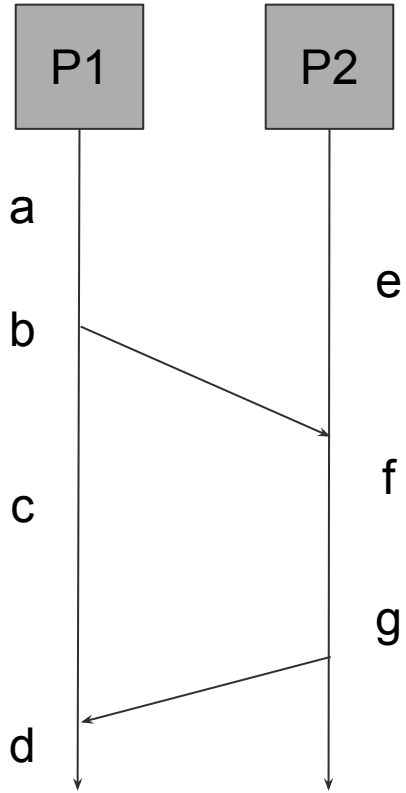
Causal+ Consistency

- Partial order: Order causally related ops the same way across all processes
- +: Replicas eventually converge
- Difference from *sequential consistency*?
  - Only causally related ops need to be ordered: no total order
  - Concurrent ops may be ordered differently across different processes

Pros: Preserves causality while improving efficiency

Cons: Need to reason about concurrency

Ops	Concurrent
a,b	**No**
a,e	**Yes**
a,g	**No**
c,e	**Yes**
c,d	**No**
d,g	**No**
d,f	**No**
e,g	**No**
a,d	**No**



Causal+ Consistency Example

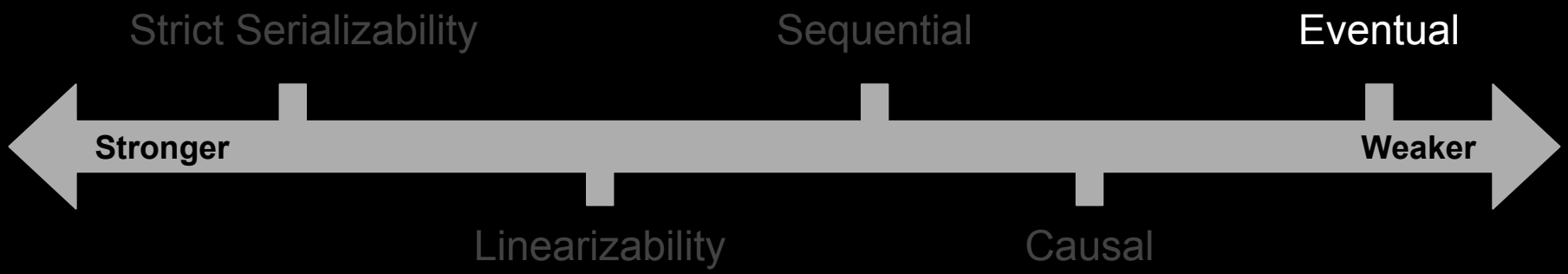
Causally+ Consistent? Yes

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

Causally+ Consistent? No

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

Consistency Models



Eventual Consistency

- Eventual convergence: If no more writes, all replicas *eventually* agree
- Difference from *causal consistency*?
  - Does not preserve causal relationships
  - Is the “+” in causal+
- Frequently used with application conflict resolution, anti-entropy

Pros: Super duper highly available

Cons: No safety guarantees, need conflict resolution

In a nutshell...

Strict Serializability: Total order + real time guarantees over *transactions*

Linearizability: Total order + real time guarantees over *operations*

Sequential consistency: Total order + process order

Causal+ consistency: Causally ordered + replicas eventually converge

Eventual consistency: Eventually everyone should agree on state