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

Linearizability

- All replicas execute operations in some total order
- That total 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

\[
\begin{align*}
P_A & \quad \dashv \quad w(x=1) \\
P_B & \quad \dashv \quad w(x=2) \\
P_C & \quad \quad \dashv \quad w(x=3) \\
P_D & \quad \dashv \quad w(x=4) \quad \quad \dashv \quad w(x=5) \\
P_E & \quad \quad \dashv \quad w(x=6) 
\end{align*}
\]
Linearizability

• Single machine processes requests one by one in the order it receives them
  • Will receive requests ordered by real-time in that order
  • Will receive all requests in some order

• Atomic Multicast, Viewstamped Replication, Paxos, and RAFT provide Linearizability

• Single machine processing incoming requests one at a time also provide Linearizability 😊

Linearizability is Ideal?

• Hides the complexity of the underlying distributed system from applications!
  • Easier to write applications
  • Easier to write correct applications

• But, performance trade-offs

Stronger vs Weaker Consistency

• Stronger consistency models
  + Easier to write applications
    • More guarantees for the system to ensure
      (Results in performance tradeoffs)

• Weaker consistency models
  • Harder to write applications
  + Fewer guarantees for the system to ensure

Strictly Stronger Consistency

• A consistency model A is strictly stronger than B if it allows a strict subset of the behaviors of B
  • Guarantees are strictly stronger
Linearizability

- All replicas execute operations in some total order
- That total 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.

Sequential Consistency

- All replicas execute operations in some total order
- That total order preserves the process ordering between operations
  - If process P issues operation A before operation B, then A is ordered before B by the process order (i.e., preserves local ordering)
  - If operations A and B done by different processes then there is no process order between them. But there must be some total order.

Sequential Consistency ≈ “Appears to be a Single Machine”

- Single machine processes requests one by one in the order it receives them
  - Will receive requests ordered by process order in that order
  - Will receive all requests in some order

Linearizability is strictly stronger than Sequential Consistency

Linearizability: ∃ total order + real-time ordering
Sequential: ∃ total order + process ordering

where Process ordering ⊆ Real-time ordering
A week later, Alice receives an email informing her that her password change request successfully completed. The user account system hashes the plain text that Alice enters and notifies Alice her password change request. The user account system receives the hashed password and changes her password. The user account system is related to a company named Shopazon, which notifies their users of this public data breach. Shopazon stores users' credit card information, but requires users to enter passwords for an online shopping service. As an example of this insufficiency, the user account system may be perceived as correct by the application users. We consider the need for applications that require timeliness guarantees in order to ensure that the effects of operations are observed in a timely manner by all clients.

### Sequential But Not Linearizable

A
- w(alicePwd=0x5f4)
- r(alicePwd)=0x81d

B
- One week later

### Consistency Hierarchy

- Linearizability (e.g., RAFT)
  - Sequential Consistency
    - Causal+ Consistency (e.g., Bayou)
      - Eventual Consistency (e.g., Dynamo)

### Causal+ Consistency

- Partially orders all operations, does not totally order them
  - Does not look like a single machine

- Guarantees
  - For each process, \( \exists \) an order of all writes + that process’s reads
  - Order respects the happens-before (\( \rightarrow \)) ordering of operations
  - + replicas converge to the same state
    - Skip details, makes it stronger than eventual consistency

### Causal Consistency

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

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

**Concurrent: Ops not causally related**
Causal Consistency

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

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

Concurrent: Ops not causally related

<table>
<thead>
<tr>
<th>Operations</th>
<th>Concurrent?</th>
</tr>
</thead>
<tbody>
<tr>
<td>a, b</td>
<td>N</td>
</tr>
<tr>
<td>b, f</td>
<td>Y</td>
</tr>
<tr>
<td>c, f</td>
<td>Y</td>
</tr>
<tr>
<td>e, f</td>
<td>Y</td>
</tr>
<tr>
<td>e, g</td>
<td>N</td>
</tr>
<tr>
<td>a, c</td>
<td>Y</td>
</tr>
<tr>
<td>a, e</td>
<td>N</td>
</tr>
</tbody>
</table>

Causal+ But Not Sequential

Happens Before Order

P_a: w(x=1) → r(y)=0

P_b: w(y=1) → r(x)=0

Process Ordering

P_a: w(x=1) → r(y)=0

P_b: w(y=1) → r(x)=0

No Total Order

P_a: w(x=1) → r(y)=0

P_b: w(y=1) → r(x)=0
Eventual But Not Causal+

As long as $P_B$ eventually would see $r(x)=1$ this is fine

Consistency Hierarchy

- Linearizability: e.g., RAFT
- Sequential Consistency
- Causal Consistency: e.g., Bayou
- Eventual Consistency: e.g., Dynamo

Causal Consistency: Quiz

- Valid under causal consistency
- Why? $w(x=3)$ and $w(x=2)$ are concurrent
  - So all processes don’t (need to) see them in same order
- $P_C$ and $P_O$ read the values ‘1’ and ‘2’ in order as potentially causally related. No ‘causality’ for ‘3’. 
**Sequential Consistency: Quiz**

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

  - Invalid under sequential consistency
  - Why? $P_C$ and $P_D$ see 2 and 3 in different order
  - But fine for causal consistency: 2 and 3 are not causally related

**Causal Consistency: Quiz**

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

  - $x=2$ happens after $x=1$

**Causal Consistency: Quiz**

- $P_A \vdash w(x=a)$
- $P_B \vdash w(x=b)$
- $P_C \vdash r(x)=b \quad \vdash r(x)=a$
- $P_D \vdash r(x)=a \quad \vdash r(x)=b$

  - $P_B$ doesn’t read value of 1 before writing 2