Consistency

11/16/2018
Consistency Models

- Strict Serializability
- Linearizability
- Sequential
- Causal+
- Eventual

Stronger → Weaker
Consistency Models

- Strict Serializability
- Linearizability
- Sequential
- Causal+
- Eventual

- Stronger
- Weaker
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

<table>
<thead>
<tr>
<th>Strictly Serializable?</th>
<th>Yes</th>
<th>Strictly Serializable?</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1:</td>
<td>{W(x)b, W(y)b}</td>
<td>P1:</td>
<td>{W(x)b, W(y)b}</td>
</tr>
<tr>
<td>P2: {W(x)a}</td>
<td></td>
<td>P2: {W(x)a}</td>
<td></td>
</tr>
<tr>
<td>P3:</td>
<td>{R(x)a}</td>
<td>P3:</td>
<td>{R(y)b}</td>
</tr>
<tr>
<td></td>
<td>{R(x)b}</td>
<td></td>
<td>{R(x)a}</td>
</tr>
<tr>
<td>P4:</td>
<td>{R(x)b}</td>
<td>P4:</td>
<td>{R(y)b}</td>
</tr>
<tr>
<td></td>
<td>{R(y)b}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Consistency Models

- Strict Serializability
- Linearizability
- Sequential
- Causal+
- Eventual

Stronger → Linearizability → Sequential → Causal+ → Eventual → Weaker
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, whereas 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

<table>
<thead>
<tr>
<th>Linearizable?</th>
<th>Yes</th>
<th>Linearizable?</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1:</td>
<td>W(x)a</td>
<td>P1:</td>
<td>W(x)a</td>
</tr>
<tr>
<td>P2:</td>
<td>W(x)b</td>
<td>P2:</td>
<td>W(x)b</td>
</tr>
<tr>
<td>P3:</td>
<td>R(x)a</td>
<td>P3:</td>
<td>R(x)b</td>
</tr>
<tr>
<td>P4:</td>
<td>R(x)a        R(x)b</td>
<td>P4:</td>
<td>R(x)b        R(x)a</td>
</tr>
</tbody>
</table>
Consistency Models

- Strict Serializability
- Linearizability
- Sequential
- Causal+
- Eventual
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

<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
</tr>
</thead>
<tbody>
<tr>
<td>W(x)a</td>
<td>W(x)b</td>
<td>R(x)b</td>
<td>R(x)a</td>
</tr>
</tbody>
</table>

**Sequentially Consistent?**  No

<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
</tr>
</thead>
<tbody>
<tr>
<td>W(x)a</td>
<td>W(x)b</td>
<td>R(x)b</td>
<td>R(x)a</td>
</tr>
</tbody>
</table>

Diagrams illustrating the operations and their order for sequential consistency.
Consistency Models

Strict Serializability → Linearizability → Causal+ → Eventual

Stronger → Sequential → Weaker
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
<table>
<thead>
<tr>
<th>Ops</th>
<th>Concurrent</th>
</tr>
</thead>
<tbody>
<tr>
<td>a,b</td>
<td>No</td>
</tr>
<tr>
<td>a,e</td>
<td>Yes</td>
</tr>
<tr>
<td>a,g</td>
<td>No</td>
</tr>
<tr>
<td>c,e</td>
<td>Yes</td>
</tr>
<tr>
<td>c,d</td>
<td>No</td>
</tr>
<tr>
<td>d,g</td>
<td>No</td>
</tr>
<tr>
<td>d,f</td>
<td>No</td>
</tr>
<tr>
<td>e,g</td>
<td>No</td>
</tr>
<tr>
<td>a,d</td>
<td>No</td>
</tr>
</tbody>
</table>
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

- Strict Serializability
- Linearizability
- Sequential
- Causal
- Eventual

Stronger on the left, Weaker on the right.
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