Consistency

COS 316: Principles of Computer System Design

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:**

Control how much of this complexity is abstracted away
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

- Strongly Consistent Distributed System
- Weakly Consistent Distributed System
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

Mythical Single Machine

$P_A \rightarrow w(x=1) \rightarrow P_B$

$w(x=2)$
Real-Time Ordering Examples

Mythical Single Machine

$P_A \quad w(x=1)$

$P_B \quad w(x=2)$

$P_C \quad w(x=3)$
Linearizable?

\[ P_A \quad \vdash \quad w(x=1) \quad \]

\[ P_B \quad \vdash \quad w(x=2) \quad \]

\[ P_C \quad \vdash \quad w(x=3) \quad \]

\[ P_D \quad \vdash \quad r(x)=2 \quad \vdash \quad r(x)=3 \quad \]

\[ \checkmark \quad w_1, w_2, r_2, w_3, r_3 \]
Linearizable?

\[ \begin{align*}
P_A & \vdash w(x=1) \\
P_B & \vdash w(x=2) \\
P_C & \vdash w(x=3) \\
P_D & \vdash r(x)=2 \quad \vdash r(x)=3 \\
P_D & \vdash r(x)=1 \quad \vdash r(x)=2 \\
\end{align*} \]

\( w_1, r_1, w_2, r_2, w_3 \)
Linearizable?

\[ \begin{align*}
\mathcal{P}_A & \vdash w(x=1) \\
\mathcal{P}_B & \vdash w(x=2) \\
\mathcal{P}_C & \vdash w(x=3) \\
\mathcal{P}_D & \vdash r(x)=2 \\
\mathcal{P}_D & \vdash r(x)=1 \\
\mathcal{P}_D & \vdash r(x)=2
\end{align*} \]
Linearizable?

$P_A \models w(x=1)$

$P_B \models w(x=2)$

$P_C \models w(x=3)$

$P_D \models r(x)=2 \quad r(x)=3$

$P_D \models r(x)=1 \quad r(x)=2$

$P_D \models r(x)=2 \quad r(x)=2$

$P_D \models r(x)=1 \quad r(x)=3$

$w_1, r_1, w_2, w_3, r_3$
Linearizable?

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

- $P_D \vdash r(x)=2 \quad r(x)=3$
- $P_D \vdash r(x)=1 \quad r(x)=2$
- $P_D \vdash r(x)=2 \quad r(x)=2$
- $P_D \vdash r(x)=1 \quad r(x)=3$
- $P_D \vdash r(x)=2 \quad r(x)=1$
Linearizable?

\( P_A \vdash w(x=1) \)
\( P_B \vdash w(x=2) \)
\( P_C \vdash w(x=3) \)
\( P_D \vdash w(x=4) \vdash w(x=5) \)
\( P_E \vdash w(x=6) \)
\( P_F \vdash r(x)=2 \vdash r(x)=3 \vdash r(x)=6 \vdash r(x)=5 \)  

\( w_1, w_2, r_2, w_4, w_3, r_3, w_6, r_6, w_5, r_5 \)
\( \text{OR} \)
\( w_1, w_4, w_2, r_2, w_3, r_3, w_6, r_6, w_5, r_5 \)
\( \text{OR} \)
\( w_1, w_2, r_2, w_3, r_3, w_4, w_6, r_6, w_5, r_5 \)
Linearizable?

\[ P_A \quad \models \quad w(x=1) \quad \models \]
\[ P_B \quad \models \quad w(x=2) \quad \models \]
\[ P_C \quad \models \quad w(x=3) \quad \models \]
\[ P_D \quad \models \quad w(x=4) \quad \models \quad w(x=5) \quad \models \]
\[ P_E \quad \models \quad w(x=6) \quad \models \]
\[ P_G \quad \models \quad r(x)=2 \quad \models \quad r(x)=5 \quad \models \quad r(x)=6 \quad \models \quad r(x)=5 \quad \models \quad \mathbf{X} \]
Linearizable?

$P_A \vdash w(x=1)$

$P_B \vdash w(x=2)$

$P_C \vdash w(x=3)$

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

$P_E \vdash w(x=6)$

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

$w_1, w_4, r_4, w_2, r_2, w_3, r_3, w_5, w_6, r_6$
Linearizable?

\[ P_A \vdash w(x=1) \]
\[ P_B \vdash w(x=2) \]
\[ P_C \vdash r(x)=1 \]
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!!
   • 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

- Linearizability
- Causal+ Consistency
- Eventual Consistency

- CAP

- PRAM 1988 (Princeton)
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

Friends

↓ Then ↓

New Job!

↓ Then ↓

Add to Cart

↓ Then ↓

Error
404 – File not found

Employment retained

Purchase retained

Deletion retained
Causal+ Sufficient

Then

Then

Then

Then

Hide from Timeline

Proceed to checkout
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

Causal+ Consistency

Eventual Consistency

Behaves like a single machine

Everyone sees related operations in the same order

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 → Causal+ → 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?