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
  - Might be slow

• 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
  + Can be faster!
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

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

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

Mythical Single Machine
Linearizable?

\begin{align*}
P_A & \vdash w(x=1) & w_1, w_2, r_2, w_3, r_3 \\
P_B & \vdash w(x=2) & \\
P_C & \vdash w(x=3) & \\
P_D & \vdash r(x)=2 & \vdash r(x)=3 & \\
\end{align*}

\checkmark
Linearizable?

\[ P_A \vdash w(x=1) \]
\[ P_B \vdash w(x=2) \]
\[ P_C \vdash w(x=3) \]

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

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

\[ P_A \models w(x=1) \]
\[ P_B \models w(x=2) \]
\[ P_C \models w(x=3) \]
\[ P_D \models r(x)=2 \]
\[ P_D \models r(x)=3 \]
\[ P_D \models r(x)=2 \]
\[ P_D \models r(x)=2 \]

\[ w_1, w_2, r_2, r_2, w_3 \]
Linearizable?

$P_A \vdash w(x=1)$

$P_B \vdash w(x=2)$

$P_C \vdash w(x=3)$

$P_D \vdash r(x)=2$

$P_D \vdash r(x)=1$

$P_D \vdash r(x)=2$

$P_D \vdash r(x)=1$

$P_D \vdash r(x)=3$

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

\[ P_A \vdash w(x=1) \]
\[ P_B \vdash w(x=2) \]
\[ P_C \vdash w(x=3) \]
\[ P_D \vdash r(x)=2 \]
\[ P_D \vdash r(x)=3 \]
\[ P_D \vdash r(x)=2 \]
\[ P_D \vdash r(x)=2 \]
\[ P_D \vdash r(x)=1 \]
\[ P_D \vdash r(x)=3 \]
\[ P_D \vdash r(x)=2 \]
\[ P_D \vdash r(x)=1 \]

✔ ✔ ✔ ✔ ✔ ✔ ✔ ✔ ✔
Linearizable?

\[ \begin{align*}
&\text{P}_A \vdash w(x=1) \downarrow \\
&\text{P}_B \quad \vdash w(x=2) \downarrow \\
&\text{P}_C \quad \vdash w(x=3) \downarrow \\
&\text{P}_D \quad \vdash w(x=4) \downarrow \quad \vdash w(x=5) \downarrow \\
&\text{P}_E \quad \vdash w(x=6) \downarrow \\
&\text{P}_F \quad \vdash r(x)=2 \quad \vdash r(x)=3 \quad \vdash r(x)=6 \quad \vdash r(x)=5 \quad \checkmark
\end{align*} \]

\[ w_1, w_2, r_2, w_4, w_3, r_3, w_6, r_6, w_5, r_5 \]

OR

\[ w_1, w_4, w_2, r_2, w_3, r_3, w_6, r_6, w_5, r_5 \]

OR

\[ w_1, w_2, r_2, w_3, r_3, w_4, w_6, r_6, w_5, r_5 \]
Linearizable?

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

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

\[
\begin{align*}
w_1, w_4, r_4, w_2, r_2, w_3, r_3, w_5, w_6, r_6
\end{align*}
\]
Linearizable?

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


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!!
   - Covered extensively in 418

3. …
Consistency Hierarchy

- Linearizability
  - Causal+ Consistency
    - Eventual Consistency

Behaves like a single machine
Everyone sees related operations in the same order
Anything goes
Causal+ Consistency Informally

1. Potential causality: event \( a \) could have a causal effect on event \( b \).

2. 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

Photo Added

Purchase retained

Deletion retained
Causal+ Not Sufficient
(Need Linearizability)

• Need a total order of operations
  • e.g., Alice’s bank account $\geq 0$

• Need a real-time ordering of operations
  • e.g., Alice changes her password, Eve 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 □ 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?