CSE 486/586 Distributed Systems
Consistency --- 1

Steve Ko
Computer Sciences and Engineering
University at Buffalo

Recap
• Views?
  – Versioned membership
• View-synchronous group communication?
  – Providing group communication with a dynamic group
  – A way to design replicated state machines
  – “What happens in the view, stays in the view.”

Examples

Consistency

• Consider that this is a storage service that serves read/write requests.
• Need consistent updates to all copies of object

Consistency Question
• How do we ensure that multiple copies have the same object?
  • Let’s think about this in terms of read/write operations…
  • From the client’s perspective, when do you know if an object has a new value?
  • It depends on when writes become visible to reads.
  • There are several guarantees we can provide.
    – Linearizability
    – Sequential consistency
    – Causal consistency
    – …
  • We’ll see the first two; and later the third.

Linearizability
• What would be the strongest (and probably most natural) form of consistency?
  • Linearizability
    – A read operation returns the most recent write, regardless of the clients.
  • Think of a single system read/write. What happens for a write followed by a read?
  • Why does this mean in a distributed setting?
    – Multiple clients can interact with different servers. Servers maintain replicas.
    – Client C1 writes to server S1 at time t, client C2 reads from server S2 at time t+1. S2 should return what C1 wrote.
Linearizability: Deriving the Definition

• What's the first requirement in maintaining replicas?
  – It should act as a single copy.
  – I.e., if you say that your system provides linearizability then it should appear to your clients that your system only has single copies of objects.

• How (conceptually, not algorithmically)?
  – Hint with a single server with a single client as follows.
  – Given a set of operations from the client, there is a single order (program order) that explains what values were written and what values were read on a single copy.
  – Adapt that in a distributed setting?

• Single copy semantics
  – There should be a single interleaving of operations that explains the results of all clients' read/write operations as if all of them were done over a single copy.

Linearizability: Deriving the Definition

• Can you come up with a single interleaving?
  – C1: write A
  – C2: write B
  – C3: read B, read A
  – C4: read B, read A
  – One possibility: C2 (write B) -> C3 (read B) -> C4 (read B) -> C1 (write A) -> C3 (read A) -> C4 (read A)

• Can you come up with a single interleaving?
  – C1: write A
  – C2: write B
  – C3: read B, read A
  – C4: read A, read B

Real-time aspect
  – You always should read what is written right before you.
  – I.e., A write should be visible to the next read immediately.

Problem: read and write operations take time

Linearizability: Deriving the Definition

• Linearizability
  • Single-copy semantics
  • A read operation returns the most recent write, regardless of the clients.

• Real-time aspect
  – You always should read what is written right before you.
  – I.e., A write should be visible to the next read immediately.

• Problem: read and write operations take time

Linearizability Subtleties

• Clear-cut (black—write & red—read)

• Not-so-clear-cut (parallel)
  – Case 1:
  – Case 2:
  – Case 3:

Linearizability Subtleties

• An operation takes time to finish.
  – E.g., a read op R starts at X ms and finishes at Y ms.

• A value written by a write operation becomes (physically) visible at some point during the operation.
  – E.g., a write op W starts at X ms and finishes at Y ms. At Z ms (X < Z < Y), the value gets actually written and becomes visible.

• What's a reasonable thing to do with this?
  – If W finishes at X, R starts at Y, and X < Y, then R should read what W wrote.
  – If R overlaps with W, then it can read either the previous value or the value written by W.
Linearizability Subtleties

- Definite guarantee
- Relaxed guarantee when overlap
- Case 1
- Case 2
- Case 3

Linearizability

- Let’s say you’re an oracle.
- Let your clients make requests (concurrent read/write).
- Let your system (with replicas) execute the requests.
- Write down the real-time execution of operations of your system. Two things to write down:
  - At what points in time each operation starts and ends.
  - Real-time precedence among operations: if A ends then B starts in real time, then A precedes B. (Caution: this is not a total order.)
- See if you can come up with an ordering of operations that meets three conditions:
  - All operations in the ordering appear one at a time as if each operation happened atomically.
  - The ordering gives the correct result as if it was done over a single copy.
  - The ordering preserves the real-time precedence of operations (i.e., the ordering written down from the above).

Linearizability Examples

- Example 1
  
  \[
  \begin{align*}
  \text{a.write}(x) \\
  \text{a.read()} & \rightarrow x \\
  \text{a.read()} & \rightarrow x
  \end{align*}
  \]

- Example 2
  
  \[
  \begin{align*}
  \text{a.write}(x) \\
  \text{a.read()} & \rightarrow 0 \\
  \text{a.read()} & \rightarrow x
  \end{align*}
  \]

Linearizability

- Let the sequence of read and update operations that client i performs in some execution be \( o_1, o_2, \ldots \).
- “Program order” for the client
- (Textbook definition) A replicated shared object service is linearizable if for any execution (real), there is some interleaving of operations (virtual) issued by all clients that:
  - meets the specification of a single correct copy of objects
  - is consistent with the real times at which each operation occurred during the execution
- Main goal: any client will see (at any point of time) a copy of the object that is correct and consistent
- The strongest form of consistency

Linearizability

- Example 3
  
  \[
  \begin{align*}
  \text{a.write}(x) \\
  \text{a.read()} & \rightarrow x \\
  \text{a.read()} & \rightarrow x \\
  \text{a.read()} & \rightarrow y \\
  \text{a.write}(y)
  \end{align*}
  \]

Chain Replication

- One technique to provide linearizability

\[
\begin{array}{ccc}
\text{N0} & \rightarrow \text{N1} & \rightarrow \text{N2} \\
\text{Head} & \rightarrow \text{Reads} & \rightarrow \text{Replies} \\
\end{array}
\]
**Summary**

- Linearizability
  - Single-copy semantics
  - Real-time aspect
- A read operation returns *the most recent* write, regardless of the clients.

**Acknowledgements**

- These slides contain material developed and copyrighted by Indranil Gupta (UIUC).