CSE 486/586 Distributed Systems
Consistency --- 2

Steve Ko
Computer Sciences and Engineering
University at Buffalo

Linearizability vs. Sequential Consistency

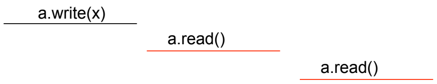
- Both care about giving an illusion of a single copy.
  - From the outside observer, the system should (almost) behave as if there's only a single copy.
- Linearizability cares about time.
- Sequential consistency cares about program order.
- We need to look deeper into both concepts to understand the difference.

Linearizability

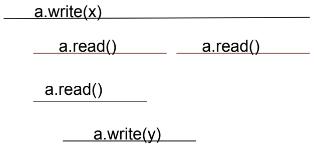
- Linearizability
  - Should provide the behavior of a single copy
  - A read operation returns the most recent write, regardless of the clients.
  - All subsequent read ops should return the same result until the next write, regardless of the clients.
- "The most recent" & "all subsequent"
  - Determined by time.
- Complication
  - In the presence of concurrency, read/write operations overlap.

Linearizability Complications

- Non-overlapping ops: time-based clear-cut ordering



- Overlapping ops: not clear-cut with time

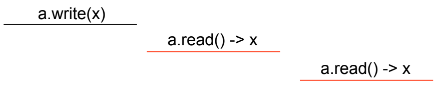


Linearizability Complications

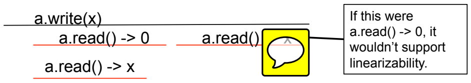
- Non-overlapping ops: time-based clear-cut ordering
  - Global time determines "most recent write" & "subsequent reads"
- Overlapping ops: not clear-cut with time
  - The system needs to provide an ordering of ops.
  - The ordering should give an illusion that it has a single copy.
- I.e., some ordering of operations where:
  - A read returns the result of the most recent write.
  - Once the result of the write becomes visible, all subsequent reads return the same result until the next write becomes visible.

Linearizability Examples

- Example 1



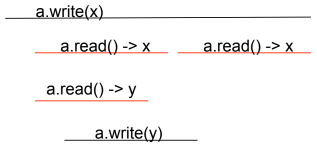
- Example 2



If this were a.read() -> 0, it wouldn't support linearizability.

Linearizability Examples

- Example 3

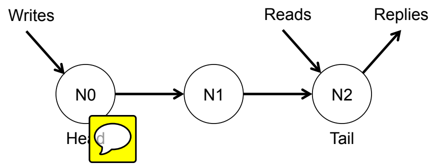


CSE 486/586 Administrivia

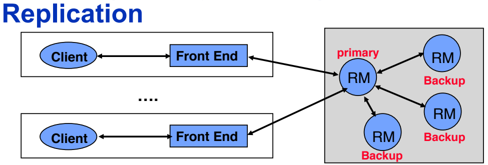
- PA3 deadline: 3/29 (Friday)
- Anonymous feedback form still available.
- Please come talk to me!

Chain Replication

- One technique to provide linearizability



Passive (Primary-Backup) Replication



- Request Communication: the request is issued to the primary RM and carries a unique request id.
- Coordination: Primary takes requests atomically, in order, checks id (resends response if not new id.)
- Execution: Primary executes & stores the response
- Agreement: If update, primary sends updated state/result, req-id and response to all backup RMs (1-phase commit enough).
- Response: primary sends result to the front end

Linearizability vs. Sequential Consistency

- Both care about giving an illusion of a single copy.
  - From the outside observer, the system should (almost) behave as if there's only a single copy.
- Linearizability cares about time.
- Sequential consistency cares about program order.

Sequential Consistency

- Sequential consistency
  - Should provide the behavior of a single copy
  - A read operation returns the most recent write, regardless of the clients.
  - All subsequent read ops should return the same result until the next write, regardless of the clients.
- "The most recent" & "all subsequent"
  - Ops within the same client: Determined by time (program order)
  - Ops across clients: Not determined by time, i.e., we can re-order them.
  - I.e., we just need to preserve the program order

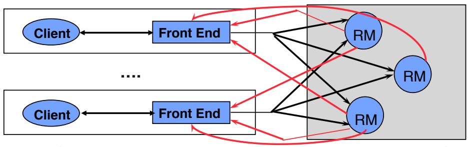
Sequential Consistency

- To the outside observer, the system needs to provide a global ordering of operations where:
  - It works like a single copy.
  - The ordering of ops coming from the same client is preserved.
- Linearizability vs. sequential consistency
  - With sequential consistency, the system has freedom as to how to interleave operations coming from different clients, as long as the ordering from each client is preserved.
  - With linearizability, the interleaving across all clients is pretty much determined already based on time.

Sequential Consistency Examples

- Example 1
  - P1: a.write(A)
  - P2: a.write(B)
  - P3: a.read()->B a.read()->A
  - P4: a.read()->B a.read()->A
- Example 2
  - P1: a.write(A)
  - P2: a.write(B)
  - P3: a.read()->B a.read()->A
  - P4: a.read()->A a.read()->B

Active Replication



- Request Communication: The request contains a unique identifier and is multicast to all by a reliable totally-ordered multicast.
- Coordination: Group communication ensures that requests are delivered to each RM in the same order (but may be at different physical times!).
- Execution: Each replica executes the request. (Correct replicas return same result since they are running the same program, i.e., they are replicated protocols or replicated state machines)
- Agreement: No agreement phase is needed, because of multicast delivery semantics of requests
- Response: Each replica sends response directly to FE

Summary

- Linearizability
  - The ordering of operations is determined by time.
  - Primary-backup can provide linearizability.
  - Chain replication can also provide linearizability.
- Sequential consistency
  - The ordering of operations preserves the program order of each client.
  - Active replication can provide sequential consistency.

Acknowledgements

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