Recap: Linearizability

- **Linearizability**
  - Should provide the behavior of a single copy
  - A read operation returns the most recent write, regardless of the clients.
  - "The most recent": determined by time.
- **Complication**
  - In the presence of concurrency, read/write operations overlap.

Recap: Linearizability Complications

- Non-overlapping ops: time-based clear-cut ordering
  ```
  a.write(x)
  a.read()
  a.read()
  ```
- Overlapping ops: not clear-cut with time
  ```
  a.write(x)
  a.read()  a.read()
  a.read()
  a.write(y)
  ```

Linearizability Examples

- **Example 1**
  ```
  a.write(x)
  a.read() -> x
  a.read() -> x
  ```
- **Example 2**
  ```
  a.write(x)
  a.read() -> 0
  a.read() -> x
  a.read() -> x
  ```
  If this were a.read() -> 0, it wouldn’t support linearizability.

Example 3

```
  a.write(x)
  a.read() -> x
  a.read() -> x
  a.read() -> y
  ```

Chain Replication

- One technique to provide linearizability

```
N0  N1  N2
Head  Reads  Replies
N0  N1  N2
Tail
```
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

CSE 486/586, Spring 2014

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.
  - Steve writes on his facebook wall at 11am.
  - Atri writes on his facebook wall at 11:05am.
  - Everyone will see the posts in that order.
- Sequential consistency cares about program order.
  - Steve writes on his facebook wall at 11am.
  - Atri writes on his facebook wall at 11:05am.
  - It's not necessarily that the posts will be ordered that way (though everyone will see the same order).

CSE 486/586, Spring 2014

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.

CSE 486/586, Spring 2014

CSE 486/586 Administrivia

- PA3 deadline: 4/11 (Friday)

CSE 486/586, Spring 2014

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

CSE 486/586, Spring 2014
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

Two More Consistency Models

- **Even more relaxed**
  - We don’t even care about providing an illusion of a single copy.
- **Causal consistency**
  - We care about ordering causally related write operations correctly.
- **Eventual consistency (next lecture)**
  - As long as we can say all replicas converge to the same copy eventually, we’re fine.

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).