RIFL: Implementing Linearizability at Large Scale and Low Latency

Jiaxin Wang
Motivation

• Consistency
  • Important issue in large-scale storage systems

• Linearizability
  • Strongest form of consistency for concurrent systems
Linearizability

• A **concurrent** execution of transactions is equivalent to one that executes the transactions serially in **some sequential order**

• The sequential order must preserve the **real-time** constraints of non-overlapping operations
Linearizability

1) Non-linearizable

2) Linearizable

Reference: Manos Kapritsos, EECS 591 Distributed System, Lecture 9
Problem

• Few large-scale storage systems implement linearizability today

• What violates linearizability?
  • “at-least-once semantics” (Re-execution of operations)
Solution - RIFL

- Reusable Infrastructure for Linearizability

- Ensure “exactly-once semantics” in large-scale systems
  - How? Reuse result generated by earlier execution for retries

- Reconfiguration tolerance
- Scalability
- Low latency
RIFL Architecture

• Assumption: Remote Procedure Call (RPC) mechanism

• RIFL stores RPC results

• For each retry
  • Do not re-execute
  • Return the stored result
RIFL Architecture Key Points

• RPC identification
  • unique, managed by a lease mechanism

• Completion record durability
  • RPC id and execution results included

• Retry rendezvous
  • where to find the completion record

• Garbage collection
  • when to delete a completion record
## RPC Identification

<table>
<thead>
<tr>
<th><strong>Client ID</strong></th>
<th><strong>Sequence Number</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>64-bit integer</td>
<td>64-bit integer</td>
</tr>
</tbody>
</table>

- **Client ID**:
  - Unique system-wide
  - Assigned by the client
  - Monotonically increasing
Completion Record

- **Client ID**: 64-bit integer
- **Sequence Number**: 64-bit integer
- **RPC Result**
- **Object Identifier**

- Retries use the same RPC ID
- For retries, return the result without re-execution
- Ensure migration durability
Retry Rendezvous

- In a large-scale system, retry operations may not be sent to the same server as the original request
- Data migration often occurs, maybe due to server crashes
- Distributed operations involve multiple servers

- Solution: associate each operation with a particular object
  - Completion record always stored in the same server as the object
Garbage Collection

• Completion record eventually needs to be deleted

• How does the server know it can be safely deleted?
  • Client acknowledges they have received the result
  • Server detects client crash (by a lease mechanism)
Design Details

• **RequestTracker**
  • Manage sequence number
  • Run on client machines

• **LeaseManager**
  • Manage client leases to detect client crashes
  • Run on both clients and servers

• **ResultTracker**
  • Manage completion records
  • Run on server machines
Normal RPC

Client

- LeaseManager: ClientID = 1
- RequestTracker: SeqID = 0

Server

- W(0)
- ResultTracker:
  - RPC status = NEW
  - ClientID = 1, SeqID = 0
  - ObjectID = 3
  - W(0) Success

Val: 2
Normal RPC

Client

- LeaseManager: ClientID = 1
- RequestTracker: SeqID = 0

Server

- W(0)
- ResultTracker: RPC status = NEW
- Val: 0
- ClientID = 1, SeqID = 0
- ObjectID = 3
- W(0) Success
- SeqID = 0, W(0) Success
- SeqID = 0, ACK
RPC Retry

Client

LeaseManager: ClientID = 1
RequestTracker: SeqID = 0

Server

ResultTracker: RPC status = FINISHED

ClientID = 1, SeqID = 0
ObjectID = 3
W(0) Success

Val: 2

W(0) Success

SeqID = 0, ACK

SeqID = 0, W(0) Success
RAMCloud Evaluation: Latency

![Graph showing latency distribution for RAMCloud evaluations. The graph compares linearizable writes (red line) and original writes (blue line). The y-axis represents the fraction of writes, and the x-axis represents latency in microseconds. The graph shows a significant reduction in latency for linearizable writes compared to original writes.](image)
RAMCloud Evaluation: Throughput

![Graph showing throughput vs client count for RAMCloud with linearizable and original writes. The graph depicts a linear relationship between client count and throughput, with a notable increase at a client count of 5, followed by stabilization.]
RAMCloud Evaluation: Scalability
RIFL ensures "exactly-once semantics" to guarantee linearizability in large-scale systems with low latency.
Q&A