Stronger Semantics for Low-Latency Geo-Replicated Storage

Wyatt Lloyd*, Michael J. Freedman*, Michael Kaminsky†, and David G. Andersen‡

* Princeton University, † Intel Lab, ‡ Carnegie Mellon University

Presented by Mina Tahmasbi Arashloo
Motivation
and Problem Statement

• It is **provably impossible** to have the strongest forms of consistency and low latency in geo-replicated setting

• Key-value data model is **too simple**

• Can we build a system that provides **low latency**
  • with **stronger consistency** than eventual
  • for a **richer data model** ?
Contributions

• A scalable geo-replicated data store with:
  • low latency
  • causal consistency
  • support for column-family data model
  • read-only transactions
  • write-only transactions
Background: Web Service Architecture

* The figure is adopted from Wyatt Lloyd’s slides for his NSDI’13 presentation.
Background: Column-Family Data Model

- Pioneered by Google’s BigTable
- A “map of maps of maps” of named columns

<table>
<thead>
<tr>
<th>User Data</th>
<th>Associations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Friends</td>
</tr>
<tr>
<td></td>
<td>ID</td>
</tr>
<tr>
<td>Alice</td>
<td>1337</td>
</tr>
<tr>
<td>Bob</td>
<td>2664</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Background: Causal Consistency

![Diagram showing causal consistency with nodes w1, w2, w3, w4, w5, w6, w7, and w8 labeled for Alice, Bob, and Carol. The nodes and arrows represent dependencies and logical time. There is a table listing operations (Op) and their dependencies.]

<table>
<thead>
<tr>
<th>Op</th>
<th>Dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>w1</td>
<td>-</td>
</tr>
<tr>
<td>w3</td>
<td>w1</td>
</tr>
<tr>
<td>w5</td>
<td>-</td>
</tr>
<tr>
<td>w6</td>
<td>w3, w1</td>
</tr>
<tr>
<td>w8</td>
<td>w6, w5, w3, w1</td>
</tr>
</tbody>
</table>
Eiger - Assumptions

• Each data center should have:
  • Partitioned key-space across *logical* servers
  • Linearizability
  • Logical servers that are available unless the whole data center fails
Client Library

• Mediates access to the servers
  • Create sub-requests based on how the keys are partitioned
• Tracks causality and attaches dependencies to writes:
Basic Operations

• Logical time
  • based on Lamport clocks
  • provide global timestamps:
    • stored with the data
• Read Operations
  • return the data and timestamp
  • timestamp used for tracking dependencies
Basic Operations

• Local writes
  • updates the value
  • records timestamp (with the server id)

• Replication
  • The remote server discards if it has a newer version (based on timestamp)
  • Handles writes conflicts!
  • last writer wins
Read-Only Transactions

• First round:
  • receive *earliest valid time* (EVT) and *latest valid time* (LVT) from each server
  • If minimum LVT >= maximum EVT, there is a time where all the values are valid (*effective time*)
Read-Only Transactions

• Second round:
  • Ask the server for the location value at the effective time
Write-Only Transactions

- **Two-phase commit** with positive cohorts and indirection (2PC-PCI)
- **The client library**
  - chooses one key as the coordinator
  - sends sub-requests to corresponding servers with the coordinator key
- **Each server**
  - writes the value with “pending” status
  - sends a “YESVOTE” to coordinator
- **Coordinator**
  - timestamps the transaction
  - sends “COMMIT” to participants
Write-Only Transactions

- Each transaction sub-request is replicated
- Each remote server
  - sends a “NOTIFY” to the remote coordinator
- The remote coordinator
  - checks dependencies
  - sends “PREPARE” messages
  - the rest continues similar to the local datacenter
Failure

• Depends on the underlying building blocks assumptions for logical server’s failure

• Transient datacenter failure : no ill effects
  • requires other datacenters to redirect the client to the original datacenter for configured period

• Long datacenter failures : causality loss
  • move to a new datacenter with empty context

• Permanent datacenter failure : data loss
# Evaluation - Low Latency

<table>
<thead>
<tr>
<th>Latency (ms)</th>
<th>50%</th>
<th>90%</th>
<th>95%</th>
<th>99%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Reads</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cassandra-Eventual</td>
<td>0.38</td>
<td>0.56</td>
<td>0.61</td>
<td>1.13</td>
</tr>
<tr>
<td>Eiger 1 Round</td>
<td>0.47</td>
<td>0.67</td>
<td>0.70</td>
<td>1.27</td>
</tr>
<tr>
<td>Eiger 2 Round</td>
<td>0.68</td>
<td>0.94</td>
<td>1.04</td>
<td>1.85</td>
</tr>
<tr>
<td>Eiger Indirected</td>
<td>0.78</td>
<td>1.11</td>
<td>1.18</td>
<td>2.28</td>
</tr>
<tr>
<td>Cassandra-Strong-A</td>
<td>85.21</td>
<td>85.72</td>
<td>85.96</td>
<td>86.77</td>
</tr>
<tr>
<td>Cassandra-Strong-B</td>
<td>21.89</td>
<td>22.28</td>
<td>22.39</td>
<td>22.92</td>
</tr>
<tr>
<td><strong>Writes</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cassandra-Eventual</td>
<td>0.42</td>
<td>0.63</td>
<td>0.91</td>
<td>1.67</td>
</tr>
<tr>
<td>Cassandra-Strong-A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eiger Normal</td>
<td>0.45</td>
<td>0.67</td>
<td>0.75</td>
<td>1.92</td>
</tr>
<tr>
<td>Eiger Normal (2)</td>
<td>0.51</td>
<td>0.79</td>
<td>1.38</td>
<td>4.05</td>
</tr>
<tr>
<td>Eiger Transaction (2)</td>
<td>0.73</td>
<td>2.28</td>
<td>2.94</td>
<td>4.39</td>
</tr>
<tr>
<td>Cassandra-Strong-B</td>
<td>21.65</td>
<td>21.85</td>
<td>21.93</td>
<td>22.29</td>
</tr>
</tbody>
</table>
Evaluation - Scalability

**Graph:**
- **Y-axis:** Normalized Throughput (log)
- **X-axis:** Servers/Cluster (log)
- **Legend:**
  - 72% avg. increases
  - 96% avg. increases
- **Bar Chart:**
  - Servers/Cluster: 1, 2, 4, 8, 16, 32, 64, 128
  - Corresponding Throughput values: 1, 2, 4, 8, 16, 32, 64, 128

- **Trend Lines:**
  - 72% avg. increases
  - 96% avg. increases
Related Work

• Bayou
  • Requires single-machine replicas (datacenters)
• COPS
  • Also causal consistency, low latency, and read-only transactions
  • Eiger has richer data model, more powerful abstractions, and has less dependency overhead
Strengths

• Has low latency despite being geo-replicated
• Provides stronger consistency guarantees than previous work with negligible overhead
• Offers fast and non-blocking read-only and write-only transactions
• Scales almost linearly with #servers/datacenter
• Solid evaluation and comparison to previous work
Weaknesses

- Limited transactions
  - Read-only
  - Write-only
- Limited to causal consistency

* The figure is taken from Wyatt Lloyd’s PhD thesis.
Questions?
Comments?

• References
  
  