A Self-Configurable Geo-Replicated Cloud Storage System

Masoud Saeida Ardekani, Inria & UPMC
Douglas B. Terry,

Special thanks to:
Marcos K. Aguilera, Mahesh Balakrishnan, Ramakrishna Kotla

OSDI 2014
Scenario
Scenario

Key point: Configurations need to adapt.

The events, characters and firms depicted in this talk are fictitious. Any similarity to actual labs, living or dead, or to actual firms, is purely coincidental.
Configuration Service

- **Selects** new configuration to improve overall utility delivered to clients

- **Installs** new configuration *while* clients continue to read and write data
Storage System Model

**Configuration** =

1. Location of primary replicas
2. Location of secondary replicas
3. Synchronization period between primary and secondary replicas

Based on Pileus [SOSP 2013]
Consistency-based SLAs

• Applications declare acceptable consistency/latency pairs and utility
• E.g. shopping cart

<table>
<thead>
<tr>
<th>consistency</th>
<th>latency</th>
<th>utility</th>
</tr>
</thead>
<tbody>
<tr>
<td>strong</td>
<td>300 ms.</td>
<td>1.0</td>
</tr>
<tr>
<td>read my writes</td>
<td>300 ms.</td>
<td>0.5</td>
</tr>
<tr>
<td>eventual</td>
<td>300 ms.</td>
<td>0.1</td>
</tr>
</tbody>
</table>
Selecting a Configuration

- Clients
- SLAs
- Read/writes
- Latencies
- Config. Generator
- Constraints
- # of replicas
- Location
- Cost
- Configuration maximizes utility vs. cost
Installing a New Configuration

Current → Configuration Service → New

Actions:
- Move Primary
- Add Primary
- Downgrade Primary
- Add Secondary
- Remove Secondary
- Adjust Sync Period
Example: Move Primary

1. Set Reconfiguration-in-Progress (RiP) flag
2. Wait $\Delta$ seconds
3. Add new primary to write-only replicas
4. Clear RiP flag
5. Sync new primary from old
6. Set RiP flag
7. Wait $\Delta$ seconds
8. Write new configuration
9. Clear RiP flag
**Leasing Configurations**

- **Slow mode** = unsure of current configuration
- **Fast mode** = hold lease on configuration for $\Delta$ seconds
<table>
<thead>
<tr>
<th></th>
<th>Fast</th>
<th>Slow</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Read</strong></td>
<td>Read from best replica (ala Pileus)</td>
<td>Do speculative read then check configuration</td>
</tr>
<tr>
<td><strong>Write</strong></td>
<td>Write to all primaries</td>
<td>Lock configuration, then write</td>
</tr>
</tbody>
</table>
Implementation Notes

• Built on Azure Blob Store
• C.S. stores configuration blob
  – read periodically by clients
  – RiP flag stored as metadata
• Clients periodically write SLAs, read/write ratios, and latencies to blobs
  – read by C.S. during reconfiguration
• C.S. can run intermittently
Evaluation Setup

YCSB [SoCC’10]
Workload B (95% Read, 5% Write)
1000 Objects (1KB each)

<table>
<thead>
<tr>
<th>Consistency</th>
<th>Latency</th>
<th>Utility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strong</td>
<td>100 ms</td>
<td>1.0</td>
</tr>
<tr>
<td>RMW</td>
<td>100 ms</td>
<td>0.7</td>
</tr>
<tr>
<td>Eventual</td>
<td>250 ms</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Social Network SLA

Constraints
Replication factor ≤ 2
Single primary
Evaluation

No-Reconfiguration
Evaluation

- Make Europe Primary
- Move Secondary to U.S.
- Move Secondary to Asia
- Make U.S. Primary
- Make Asia Primary

The graph shows the utility over time for two configurations: No-Reconfiguration and Every 4 hour. The Utility values fluctuate between 0 and 1 throughout the day, with different peaks and troughs.
Improved Consistency

No-Reconfiguration

- Eventual: 34%
- Strong: 33%
- RMW: 33%

Every 4 hour

- Eventual: 17%
- RMW: 34%
- Strong: 49%
Conclusions

- Storage systems should adapt to changing client demands
- Utility/cost is a useful metric for selecting improved configurations
- Automatic reconfiguration can occur in parallel with running applications
- Substantial consistency gains are possible