Resilient Cloud-based Replication with Low Latency

December 9, 2020

Michael Eischer and Tobias Distler

Friedrich-Alexander University Erlangen-Nürnberg (FAU)

Supported by grant no. DI 2097/1-2 ("REFIT")
Geo-Replicated State Machines Using PBFT

Tolerate up to $f$ Byzantine faults

Replicated key-value store

Clients and replicas distributed worldwide

---


Eischer et al. Resilient Cloud-based Replication with Low Latency
Geo-Replicated State Machines Using PBFT

Tolerate up to $f$ Byzantine faults

Replicated key-value store

Send request to leader

Clients and replicas distributed worldwide

---


Eischer et al. Resilient Cloud-based Replication with Low Latency
Geo-Replicated State Machines Using PBFT


Eischer et al. Resilient Cloud-based Replication with Low Latency
Geo-Replicated State Machines Using PBFT

Tolerate up to $f$ Byzantine faults

Leader runs Byzantine agreement via WAN links

Clients and replicas distributed worldwide

Replicated key-value store

Send request to leader

---


Eischer et al. Resilient Cloud-based Replication with Low Latency
Geo-Replicated State Machines Using PBFT

Replicated key-value store

Leader runs Byzantine agreement via WAN links

Clients and replicas distributed worldwide

Send request to leader

Tolerate up to $f$ Byzantine faults

---


Eischer et al. Resilient Cloud-based Replication with Low Latency
Geo-Replicated State Machines Using PBFT


Eischer et al. Resilient Cloud-based Replication with Low Latency
Latency depends on leader location
High latency due to many WAN communication steps

---

Hierarchical Geo-Replicated State Machines Using Steward


Eischer et al. Resilient Cloud-based Replication with Low Latency
Hierarchical Geo-Replicated State Machines Using Steward

Hierarchical Geo-Replicated State Machines Using Steward

Request to local cluster
Generate and distribute cluster-local decisions


Resilient Cloud-based Replication with Low Latency
Hierarchical Geo-Replicated State Machines Using Steward²


Eischer et al. Resilient Cloud-based Replication with Low Latency
Hierarchical Geo-Replicated State Machines Using Steward

Request to local cluster
Generate and distribute cluster-local decisions
Very complex protocol
Reply to client

Eischer et al. “Resilient Cloud-based Replication with Low Latency”

Hierarchical Geo-Replicated State Machines Using Steward

Request to local cluster
Generate and distribute cluster-local decisions
Very complex protocol
Reply to client

\[\text{Very complex protocol}\]

\[\text{Generate and distribute cluster-local decisions}\]

\[\text{Request to local cluster}\]

\[\text{Reply to client}\]

\[\text{Hierarchical Geo-Replicated State Machines Using Steward}\]

Hierarchical Geo-Replicated State Machines Using Steward

2

Generate and distribute cluster-local decisions

Need replica site near clients for best latency

Very complex protocol

Request to local cluster

Reply to client

\[ \eta \]

Challenges

Need for a replication protocol that provides

- **Efficiency**: No complex protocols over wide-area links
- **Modularity**: Allow integrating with different consensus protocols
- **Adaptability**: Add and remove new locations
Our Approach: SPIDER
SPIDER: Architecture

Agreement group

Execution groups
SPIDER: Architecture

Cloud region

Agreement group

Availability zones

Execution groups

Eischer et al. Resilient Cloud-based Replication with Low Latency
Resilient Cloud-based Replication with Low Latency

Eischer et al.

Cloud region

Agreement group

Availability zones

Inter-regional message channel

Execution groups
Inter-Regional Message Channel (IRMC) - Message Forwarding

Channel abstraction

Subchannel A

Subchannel B

Distributed queue

Multiple FIFO subchannels with position indices
Inter-Regional Message Channel (IRMC) - Message Forwarding

Channel abstraction

Subchannel A

Subchannel B

Distributed queue

Multiple FIFO subchannels with position indices

send(A, 17, M)  
$f + 1$ send calls

Sender S₁

Sender S₂

Sender S₃

Receiver R₁

Receiver R₂

Receiver R₃

Receiver R₄

Eischer et al.

Resilient Cloud-based Replication with Low Latency
Inter-Regional Message Channel (IRMC) - Message Forwarding

Channel abstraction

Subchannel A

Subchannel B

Sender S₁

Sender S₂

Sender S₃

Receiver R₁

Receiver R₂

Receiver R₃

Receiver R₄

Distributed queue

Multiple FIFO subchannels with position indices

send(A, 17, M) $f + 1$ send calls

Faulty message

Wait for receive(A, 18)

Flow-control window

Limited capacity

Can request window move

Eischer et al. Resilient Cloud-based Replication with Low Latency
Inter-Regional Message Channel (IRMC) - Message Forwarding

Sender S₁

Sender S₂

Sender S₃

Receiver R₁

Receiver R₂

Receiver R₃

Receiver R₄

Distributed queue

Multiple FIFO subchannels with position indices

Channel abstraction

Subchannel A

Subchannel B

send(A, 17, M)

receive(A, 17) -> M

f + 1 send calls

Flow-control

window

Limited capacity

fr + 1 highest

Can request

window move
**Inter-Regional Message Channel (IRMC) - Message Forwarding**

Channel abstraction

Subchannel A

10 11 12 13 14 15 16 17 18 19

Subchannel B

85 86 87 88 89 90 91 92 ... 94

**send(A, 17, M)**

*f + 1 send calls*

**receive(A, 17) -> M**

**Faulty message**

**Distributed queue**

**Multiple FIFO subchannels with position indices**

Eischer et al.  
Resilient Cloud-based Replication with Low Latency
Inter-Regional Message Channel (IRMC) - Flow Control

Channel abstraction

Subchannel A

Sender $S_1$
A: $[14, 18], ...$

Sender $S_2$
A: $[14, 18], ...$

Sender $S_3$
A: $[14, 18], ...$

$fr + 1$ highest

Limited capacity

Flow-control window

Limited capacity

Receiver $R_1$

Receiver $R_2$

Receiver $R_3$

Receiver $R_4$

Can request
window move

Send $A, 17, M$

Faulty
message

Receive $A, 17$ -> $M$

Wait for
receive $A, 18$

Send calls

Eischer et al.
Resilient Cloud-based Replication with Low Latency

6
<table>
<thead>
<tr>
<th>C</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SPIDER - Request Processing

Preprocessing

Eischer et al. Resilient Cloud-based Replication with Low Latency
SPIDER - Request Processing

Preprocessing

Weakly consistent results

Eischer et al.
Resilient Cloud-based Replication with Low Latency
SPIDER - Request Processing

Preprocessing

Use client-specific subchannel

Request channel

Agreement
SPIDER - Request Processing

Preprocessing

Use client-specific subchannel

Request channel

Use agreement as black-box

Agreement

Only short-distance communication

Eischer et al. Resilient Cloud-based Replication with Low Latency
SPIDER - Request Processing

Preprocessing
Use client-specific subchannel

Request channel

Use agreement as black-box

Commit channel

Execution

Agreement

Only short-distance communication

Eischer et al. Resilient Cloud-based Replication with Low Latency
SPIDER - Request Processing

Preprocessing

Use client-specific subchannel

Use agreement as black-box

Agreement

Only short-distance communication

Request channel

Commit channel

Execution

Strongly consistent results

C

E

E

E

A

A

A

A

C′
SPIDER - Request Processing

Preprocessing

Use client-specific subchannel

Request channel

Agreement

Use agreement as black-box

Strongly consistent results

Commit channel

Only short-distance communication

Execution

Commit channel

Execution

Use client-specific subchannel

Preprocessing

Request channel

Agreement

Use agreement as black-box

Strongly consistent results

Commit channel

Execution

Commit channel

Execution

Use client-specific subchannel

Preprocessing

Request channel

Agreement

Use agreement as black-box

Strongly consistent results

Commit channel

Execution

Commit channel

Execution

Use client-specific subchannel
SPIDER - Garbage Collection

Preprocessing → Request channel → Agreement → Execution

- Create checkpoints in regular intervals

Eischer et al. Resilient Cloud-based Replication with Low Latency
Preprocessing

Request channel

Create checkpoints in regular intervals

Execution

Coordinate via channel flow-control

Agreement
Resilient Cloud-based Replication with Low Latency
Resilient Cloud-based Replication with Low Latency

Eischer et al.
SPIDER - Adding a New Region

Preprocessing

Agreement

Request channel

Commit channel

New region

Execution

Setup channels

Retrieve current checkpoint

Eischer et al. Resilient Cloud-based Replication with Low Latency
SPIDER - Adding a New Region
SPIDER - Adding a New Region

Preprocessing

Agreement

Request channel

Commit channel

Execution

Retrieve current checkpoint

Setup channels

Process client request

Execute requests

New region

Eischer et al. Resilient Cloud-based Replication with Low Latency
Evaluation
Replicas in 4 AWS EC2 regions: Virginia, Oregon, Ireland, Tokyo

50 clients per region

**BFT**: PBFT with 1 replica per region

**HFT**: Steward with 4 replicas as cluster in each region

**SPIDER**: 4 agreement replicas in Virginia, 3 replicas per execution group per region

Response time [ms]

Leader in V  Leader in O  Leader in I  Leader in T

Leader site in V  Leader site in O  Leader site in I  Leader site in T

Client location

Latency depends on client location

Short roundtrip times to Ohio and Ireland

In agreement group region

Best latency for clients in Ohio

SPIDER only has to wait for single WAN roundtrip to agreement

Latency varies with leader location

Stable response times
Replicas in 4 AWS EC2 regions: Virginia, Oregon, Ireland, Tokyo

50 clients per region

BFT: PBFT with 1 replica per region

HFT: Steward with 4 replicas as cluster in each region

SPIDER: 4 agreement replicas in Virginia, 3 replicas per execution group per region
Replicas in 4 AWS EC2 regions: Virginia, Oregon, Ireland, Tokyo

50 clients per region

**BFT**: PBFT with 1 replica per region

**HFT**: Steward with 4 replicas as cluster in each region

**SPIDER**: 4 agreement replicas in Virginia, 3 replicas per execution group per region
Replicas in 4 AWS EC2 regions: Virginia, Oregon, Ireland, Tokyo

50 clients per region

**BFT**: PBFT with 1 replica per region

**HFT**: Steward with 4 replicas as cluster in each region

**SPIDER**: 4 agreement replicas in Virginia, 3 replicas per execution group per region

---

BFT

<table>
<thead>
<tr>
<th>Region</th>
<th>Response Time [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>177</td>
</tr>
<tr>
<td>O</td>
<td>132</td>
</tr>
<tr>
<td>I</td>
<td>89</td>
</tr>
<tr>
<td>T</td>
<td>20</td>
</tr>
</tbody>
</table>

HFT

Latency varies with leader location

SPIDER

Latency depends on client location: Short roundtrip times to Ohio and Ireland.

In agreement group region, the best latency for clients in Ohio is: 177ms, 132ms, 89ms.

SPIDER only has to wait for single WAN roundtrip to agreement.

Latency varies with leader location.

Stable response times.
- Replicas in 4 AWS EC2 regions: Virginia, Oregon, Ireland, Tokyo

- 50 clients per region

- BFT: PBFT with 1 replica per region

- HFT: Steward with 4 replicas as cluster in each region

- SPIDER: 4 agreement replicas in Virginia, 3 replicas per execution group per region
Summary
Summary

Problem
- Performance depends on leader location
- Either high latency or high complexity
- Best replica locations depend on client locations

SPIDER
- Efficient: IRMCs forward group decisions
- Modular: Decoupled agreement and execution groups
- Adaptable: Add or remove execution groups at runtime

More details in the paper
- Different possible Inter-Regional Message Channel (IRMC) implementations
- Handling malicious clients and other attacks

Eischer et al. Resilient Cloud-based Replication with Low Latency