Zyzzyva: Speculative Byzantine Fault Tolerance

Ramakrishna Kotla, Lorenzo Alvisi, Mike Dahlin, Allen Clement, and Edmund Wong

Presented By: Michael Wilson & Eli Goldweber
Zyzzyva (ZIZ-uh-vuh)

<table>
<thead>
<tr>
<th></th>
<th>PBFT</th>
<th>Q/U</th>
<th>HQ</th>
<th>Zyzzyva</th>
<th>SMR Lower Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cost</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total replicas</td>
<td>3f+1</td>
<td>5f+1</td>
<td>3f+1</td>
<td>3f+1</td>
<td>3f+1</td>
</tr>
<tr>
<td>Replicas with application state</td>
<td>2f+1</td>
<td>5f+1</td>
<td>3f+1</td>
<td>2f+1</td>
<td>2f+1</td>
</tr>
<tr>
<td><strong>Throughput</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAC ops at bottleneck server</td>
<td>2 + (8f+1)/b</td>
<td>2 + 8f</td>
<td>4 + 4f</td>
<td>2 + 3f/b</td>
<td>2</td>
</tr>
<tr>
<td><strong>Latency</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Critical path-NW 1-way latencies</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>2/3</td>
</tr>
</tbody>
</table>

- Existing protocols focus on one area of optimization
- Zyzzyva: one protocol that matches or tops its competitors
- Uses speculation to increase speed
PBFT
PBFT

Primary

Replica 1

Replica 2

Replica 3

Pre-prepare | Prepare | Commit
Overview

- State Machine Replication for Byzantine Fault Tolerance
  - Checkpoint
  - Agreement
  - View Change
- Replicas can become temporarily inconsistent
- Clients detect inconsistency and force replicas to converge on a total ordering
- Trust => Verify

Output commit is at the Client not at the Replicas
Checkpoint and Replica State

- Garbage Collected History
- Committed History
- Speculative History
- Max CC
Agreement Case 1: Unanimous (3f+1)
Agreement Case 2: Majority Agree (2f+1 -- 3f)
Agreement Case 3: Less than 2f+1

Client
Primary
Replica 1
Replica 2
Replica 3

Speculation

2f
Agreement Case 3: Less than 2f+1

Option 1: $R_i$ matches Client Request to cached response

Speculation
Agreement Case 3: Less than $2f+1$

Option 2: $R_i$ Has not seen this request yet

$<CONFIRM - REQ, v, m, i> \sigma_i$

$<ORDER - REQ> \sigma_i$
Agreement Filling in Holes

Option 1: Primary is able to supply missing requests

\[ <\text{FILL}-\text{HOLE}, v, k, n, i> \sigma_i \]
Agreement Filling in Holes

Option 2: Replica times out waiting for primary to supply missing requests

< FILL - HOLE, v, k, n, i > σ_i
The Case of the Uncommitted Request

How do the replicas learn the outcome of the agreement protocol?
Safe and Live?

- No two requests complete with the same sequence number $n$
- $h_n$ is a prefix of $h_{n'}$ for $n < n'$ and completed requests $r$ and $r'$
- If the primary is correct when a correct client issues the request, then the request completes
- If a request from a correct client does not complete during the current view, then a view change occurs
View Change

< I - HATE - THE - PRIMARY, ν > σ_i

What can R_1, R_2 and R_3 do?
$< I - H A T E - T H E - P R I M A R Y , \nu > \sigma_i$
View Change

\[ \langle I - HATE - THE - PRIMARY, v > \sigma_i \rightarrow \langle VIEW - CHANGE, v + 1, CC, O, i > \sigma_i \]
View Change

\[<I - HATE - THE - PRIMARY, v >_{a_i} \rightarrow <VIEW - CHANGE, v + 1, CC, O, i >_{\sigma_i}\]

Client

Primary

Replica 1

Replica 2

Replica 3

New-View Timer
View Change

\[ \langle I - HATE - THE - PRIMARY, v \rangle_{\sigma_i} \rightarrow \langle \text{VIEW} - \text{CHANGE}, v + 1, CC, O, i \rangle_{\sigma_i} \rightarrow \langle \text{NEW} - \text{VIEW}, v + 1, P \rangle_{\sigma_p} \]
View Change

\[ \langle I - HATE - THE - PRIMARY, v > \sigma_i \rightarrow \langle VIEW - CHANGE, v + 1, CC, O, i > \sigma_i \rightarrow \langle NEW - VIEW, v + 1, P > \sigma_p \rightarrow \langle VIEW - CONFIRM, v + 1, n, h, i > \sigma_i \]
The Case of the Missing Phase

Client

Primary

Replica 1

Replica 2

Replica 3

3f+1

Pre-prepare
The Case of the Missing Phase

Client

Primary

Replica 1

Replica 2

Replica 3

Pre-prepare

Prepare

2f+1

2f+1
The Case of the Missing Phase

Client

Primary

Replica 1

Replica 2

Replica 3

2f+1

Pre-prepare

Prepare

Commit
Case of the Missing Phase: PBFT

We are stuck…. This breaks liveness
View Change

$< I - HATE - THE - PRIMARY , \nu > \sigma_i$
View Change

\[
\begin{align*}
&<l - HATE - THE - PRIMARY, v > \sigma_i \\
&\rightarrow <\text{VIEW-CHANGE}, v + 1, CC, O, i > \sigma_i \\
&\rightarrow <\text{NEW VIEW}, v + 1, P > \sigma_p \\
&\rightarrow <\text{VIEW-CONFIRM}, v + 1, n, h, i > \sigma_i
\end{align*}
\]
The Case of the Missing Phase

Client
Primary
Replica 1
Replica 2
Replica 3

Pre-prepare Prepare Commit

2f+1 2f+1

[Diagram showing the phases of communication between client, primary, and replicas]
Implementation Optimizations

- Replacing Signatures with MACs
- Separating Agreement from Execution
  - Client acts after getting appropriate matching responses and \( f+1 \) matches from execution replicas

\[
3f+1 \supseteq 2f+1 \supseteq f
\]
Zyzzyva5

- 5f+1 total replicas
  - 2f+1 execution replicas
  - 3f witness replicas
Evaluation
Batch size = 1

Bottleneck server cryptographic operations

Batch size = 10
Q & A
Questions

- What are some of the tradeoffs made by the speculation of Zyzzyva?
  - Are there situations that you may prefer PBFT over Zyzzyva?

- Could a faulty Client compromise the safety property?

- Does the overhead of Client participation make this reasonable to implement?
  - Clients need to ‘match’ responses