Zyzzyva: Speculative Byzantine Fault Tolerance

SOSP '07 Best paper award

-Ayush Goel, Matthew Furlong
Introduction/Motivation

- Why do we need Byzantine Fault Tolerance?

Increased reliability!
A Review of PBFT

\(<\text{PRE-PREPARE}\> (d(m), v, n, i)\)

\(<\text{PREPARE}\> (d(m), v, n, i)\)

\(<\text{COMMIT}\> (d(m), v, n, i)\)
Problems with BFT Protocols

- Why aren’t BFT protocols commonly used in practice?
- The tradeoff...
Zyzzyva Goals

- Maintain cost while increasing throughput and decreasing latency from previous BFT protocols

<table>
<thead>
<tr>
<th></th>
<th>PBFT</th>
<th>Q/U</th>
<th>HQ</th>
<th>Zyzzyva</th>
<th>State Machine Repl. Lower Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>3f+1</td>
<td>5f+1</td>
<td>3f+1</td>
<td>3f+1</td>
<td>3f+1 [31]</td>
</tr>
<tr>
<td>Replicas</td>
<td>2f+1</td>
<td>5f+1</td>
<td>3f+1</td>
<td>2f+1</td>
<td>2f+1</td>
</tr>
<tr>
<td>with application state</td>
<td>4f+1 [41]</td>
<td>3f+1</td>
<td>3f+1</td>
<td>2f+1</td>
<td>2f+1</td>
</tr>
<tr>
<td>Throughput</td>
<td>3f+1</td>
<td>5f+1</td>
<td>3f+1</td>
<td>3f+1</td>
<td>3f+1 [31]</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>2f+1</td>
</tr>
<tr>
<td>Latency</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>2/3 [2]</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 [2]</td>
</tr>
</tbody>
</table>
Zyzzyva Key Insights

- Speculatively execute requests
Zyzzyva Key Insights

- Speculatively execute requests
Zyzzyva Key Insights

- Speculatively execute requests
- The client has increased responsibility
Zyzzyva Key Insights

- Speculatively execute requests
- The client has increased responsibility
Zyzzyva - Agreement Protocol

- Maintained state by each replica
  - Ordered history

![Diagram showing ordered history over time](image-url)
Zyzzyva - Agreement Protocol

- Maintained state by each replica
  - Ordered history
  - Max Commit Certificate

![Diagram showing time and history with blocks A, B, C, ...]
Zyzzyva - Agreement Protocol

- Maintained state by each replica
  - Ordered history
  - Max Commit Certificate

![Diagram showing committed and speculative history with time progression]

**Committed History**
- A
- B
- C
- D
- E
- F

**Speculative History**

Time
Zyzzyva - Agreement Protocol

- Maintained state by each replica
  - Ordered history
  - Max Commit Certificate
  - Checkpoint
Two possible scenarios for each round:
- Ideal execution
- Faulty execution
Zyzzyva - Step 1

Client

REQUEST<operation, timestamp, client_id>

R0

R1

R2

R3
Zyzzyva - Step 2

Client

R0

ORDER_REQ<view, seq_numb, history, message_digest, ND>

R1

R2

R3
Zyzzyva - Step 3

```
<<SPEC_RESPONSE<view,
  seq_numb, history, client_id,
  timestamp>replica_id, app_reply,
  OR>
```
Zyzzyva - Step 4

Client

R0

R1

R2

R3

3f +1 matching responses
Zyzzyva - Step 4

And we’re done!
Zyzzyva - Step 4.a

Client

R0

R1

R2

R3

2f +1 matching responses

And we're done?
Client sets a timer after issuing a request
Zyzzyva - Step 4b-3

Client

R0

R1

R2

R3

2f +1 matching responses

<LOCAL-COMMIT, view, message_digest, history, replica_id, client_id>
Zyzzyva - Step 4b-4

Client

R0

R1

R2

R3

REPLY

2f +1 matching responses
Zyzzyva - No Response to Client
Zyzzyva - No Response to Client

Client

R0

R1

R2

R3

<CONFIRM-REQ, view, message, replica_id>

Client times out
Zyzzyva - No Response to Client

ORDER-REQ<view, seq_numb, history, message_d digest, ND>
Protocol

View Change

● Objective
  ○ Detect faulty primary
  ○ Ensure no change in history
  ○ Consistent order across views

● Algorithm
  ○ Stop accepting new messages
  ○ Send View change message
  ○ Send New View message

● Key Difference
  ○ The case of the missing phase
  ○ The case of the uncommitted Request
Protocol

View Change

- The case of the missing phase
  - Strengthen the condition under which a replica commits to a view change
  - Add a new phase to the view change protocol.
Protocol

View change

• The case of the uncommitted request
  ○ Weaken the condition under which a request appears in the history included in the new-view message.
Protocol

View Change

Client

R0

R1

R2

R3

3f +1 matching responses
Implementation Optimizations

- Replacing signatures with MACs
- Separate Agreement from execution
- Request batching
- Caching out of order requests
- Read only Optimization
Evaluation

Throughput

![Throughput Graph](image-url)
Evaluation

Latency
Evaluation

Fault Scalability
Evaluation

Fault Scalability
Conclusion

- Leverages fast agreement based on fast paxos
- Speculative execution of requests by replicas
- Replicas execute request without having to confirm that requests with lower sequence number have completed
- Throughput and latency approach the theoretical lower bounds for any BFT protocol.
Discussion

- Any future in this line of work?
- Block chains
Discussion

- S3
  - Rely on eventually consistent replicas of eventually consistent metadata

- Netflix (S3mpr)
- MongoDB
  - 42% write data loss (reverting writes, failovers)

- Riak (open source adaptation of dynamo)
  - Custom conflict resolution systems. Last write wins.
  - 72% data loss
  - With data partition 92% data loss

- Cassandra
  - 28% data loss (doesn’t use vector clocks)