Tolerating Slowdowns in Replicated State Machines using Copilots

Khiem Ngo, Siddhartha Sen, Wyatt Lloyd
Princeton University, Microsoft Research
Replicated State Machine (RSM)

- Fault tolerant group of replicas that acts like a single machine that does not fail

- RSMs are everywhere!
  - Distributed database, cloud storage, coordination services, …
Fault Tolerance for High Availability

Client

RSM

RSM

Replica
edcba

Replica
edcba

Replica
cb
Replicas can Slowdown instead of Fail

• Many causes:
  • Misconfigurations
  • Partial hardware failures
  • Garbage collection events
  • ...

• Effect: Replica takes longer than usual to send responses
RSMs tolerate failures, not slowdowns
Slowdowns Hurt Availability
We need slowdown tolerance!
Slowdowns Take Different Forms

• Duration
  • Transient slowdowns: not handled in general
  • Long-term slowdowns: eventually detected, but need to tolerate between onset and end of reconfiguration

• Severity
  • 10ms additional delay or 80ms?

• Scope
  • All processing paths or a subset?
Defining Slowdown Tolerance

• “slow” replica = responses to messages take more than threshold time $t$ over normal response time

• An RSM is s-slowdown-tolerant if it is not slow despite s slow replicas
  • Replacing the s slowest replicas with normal replicas should not change performance much

• This work’s focus: 1-slowdown-tolerance
No existing consensus protocol is 1-slowdown-tolerant
Multi-Paxos is Not 1-Slowdown-Tolerant
Multi-Paxos is Not 1-Slowdown-Tolerant

order  execute  reply
Multi-Paxos is Not 1-Slowdown-Tolerant

Vulnerable to slowdowns when processing path goes through a single replica.
Copilot: First 1-Slowdown-Tolerant Protocol
Ordering: Use Two Logs
Ordering: Combine Logs with Dependencies
Ordering: Dependency Cycles

Break cycles deterministically
Ordering: A Tricky Case

Possible ordering: a, a’, b, c’, c
Ordering: A Tricky Case

Possible ordering:
a, a', b, c', c
Possible ordering:
a, a', b, c', d'

RSM

Pilot

Replica
Ordering: Same on All Replicas

Compatibility check:
Only accept dependency if it cannot lead to multiple orders

Break cycles deterministically
Copilot Protocol

- Fast path
- Regular path*
- Order
- Exe
- Reply

Client

Pilot

Replica

Copilot
Copilot Protocol

- fast path
- regular path*

client

pilot

replica

copilot
Copilot Protocol: Dependencies?

RSM

Pilot

Copilot

Replica

Solution: **fast takeover** the slow pilot’s ordering work!
Copilot Protocol: Summary

• Proactive redundancy: two pilots process all commands

• Use dependencies to combine ordering from two pilots
  • Compatibility check ensures same order on all replicas
  • Cycles broken by priority
  • Fast takeover to avoiding waiting on slow pilot
Optimizations

• Ping-Pong Batching
  • Improve Copilot’s performance when both pilots are fast
  • Pilots propose compatible orderings and commit on fast path

• Null Dependency Elimination
  • Improve Copilot’s performance when one pilot is slow
  • Allow a fast pilot to safely avoid waiting on commits from a continually slow pilot and thus avoid fast takeover
Evaluation

- Tolerate slowdowns that are transient, have varying manifestations, have varying severity?
- How does Copilot perform without slow replicas?

- 5-replica RSM, moderate load
- Replicas and clients in the same datacenter

- Baselines:
  - EPaxos
  - Multi-Paxos
  - Fast-View-Change (10 ms view-change timeout)
Transient Slowdowns

Not slowdown-tolerant

Command Latency (ms)

Time (Second)

0 2 4 6 8 10 12

0 10 20 30 40 50 60 70 80 90 100

Multi-Paxos

FVC

Copilot

25 ms

12.5 ms
Gradual Slowdown

![Graph showing command latency CDF for Copilot and FVC with 40 ms as a reference point.](image)

- **Copilot** Tolerates Gradual Slowdown
- **40 ms** latency threshold
Performance Without Slow Replicas

- Comparable latency at low/moderate load
- 8% lower throughput than Multi-Paxos
Conclusion

• Slowdowns hurt availability, need s-slowdown-tolerant RSMs

• Copilot: first 1-slowdown-tolerant protocol
  • Slowdown tolerance: proactive redundancy and fast takeovers
  • Optimizations: ping-pong batching and null dependency elimination

• Copilot’s performance without slow replicas is competitive

• Copilot is the only protocol that tolerates any one slowdown