Hardening *Cassandra* Against Byzantine Failures

Roy Friedman and Roni Licher
Technion - Israel Institute of Technology

OPODIS 2017

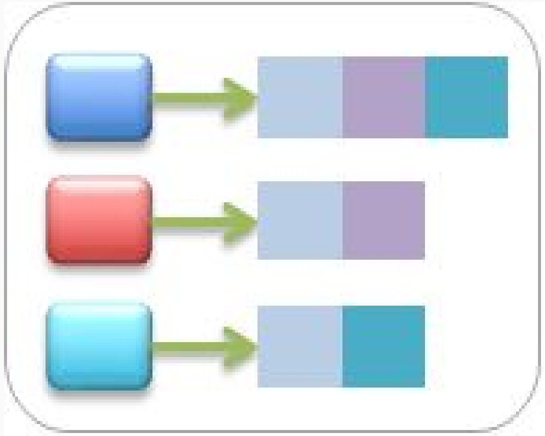
Overview

In this research we:

- Analyze the presence of byzantine failures in Cassandra
- Suggest solutions to prevent them
- Iterate to improve common case performance
- Benchmark implementation

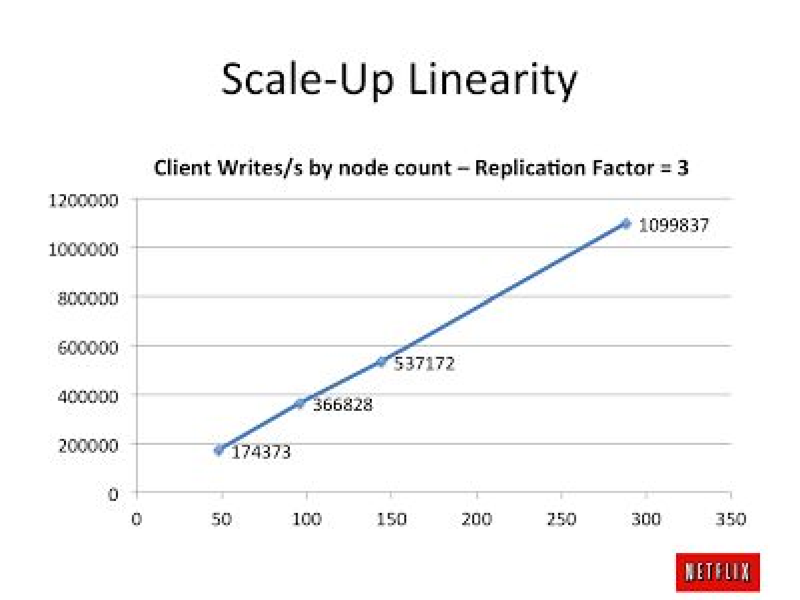
Cassandra

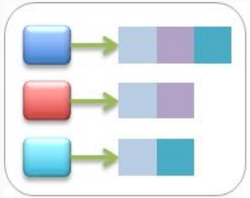
- Distributed Database
- Open Source
- Column Families



Cassandra

- Distributed Database
- Open Source
- Column Families
- Tunable Consistency
- Very Scalable

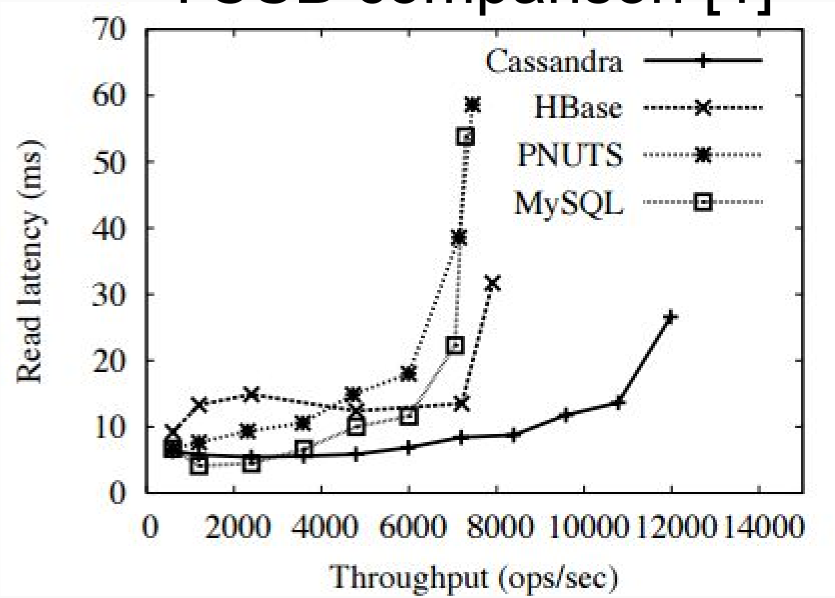


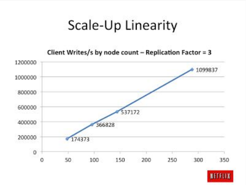


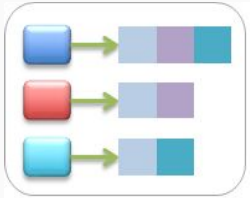
Cassandra

- Distributed Database
- Open Source
- Column Families
- Tunable Consistency
- Very Scalable
- Great performance

YCSB comparison [1]



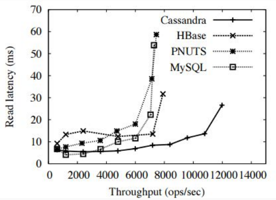


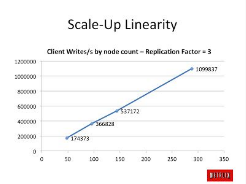


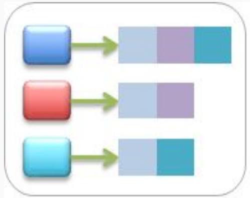
[1] Cooper, Brian F., et al. "Benchmarking cloud serving systems with YCSB." *Proceedings of the 1st ACM symposium on Cloud computing*. ACM, 2010.

Cassandra

- Distributed Database
- Open Source
- Column Families
- Tunable Consistency
- Very Scalable
- Great performance
- Highly adopted:

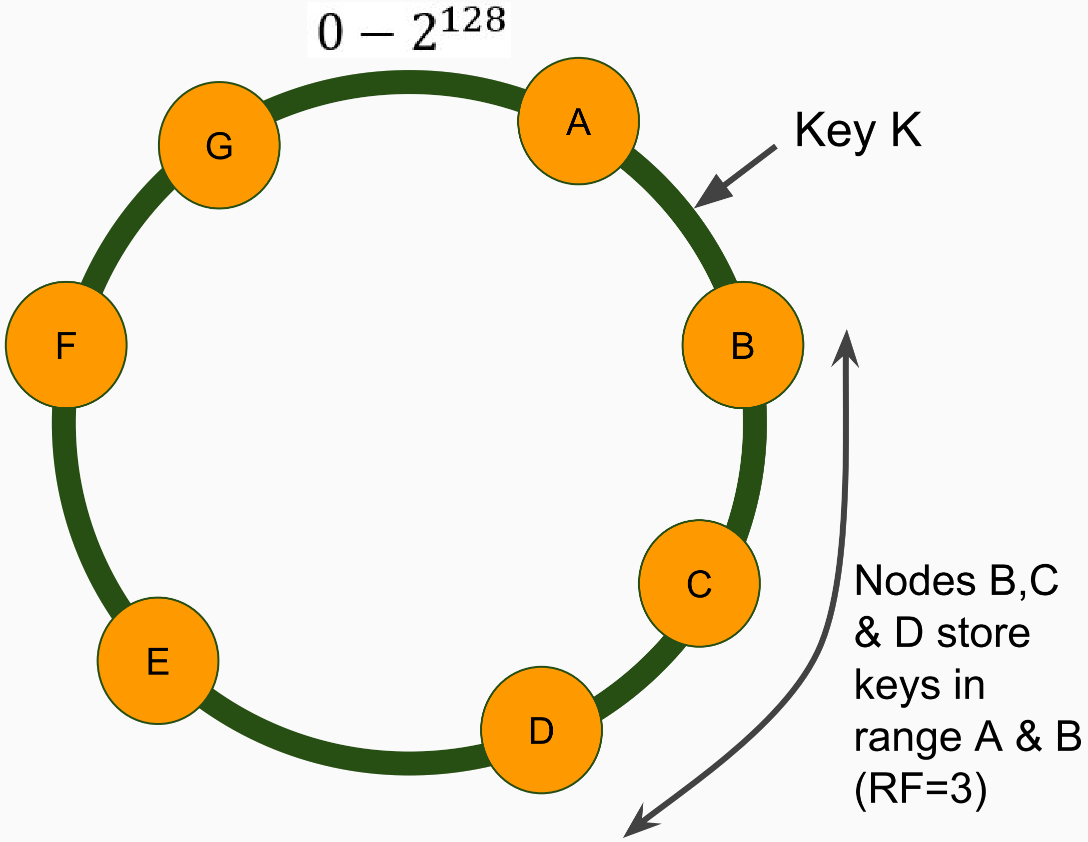




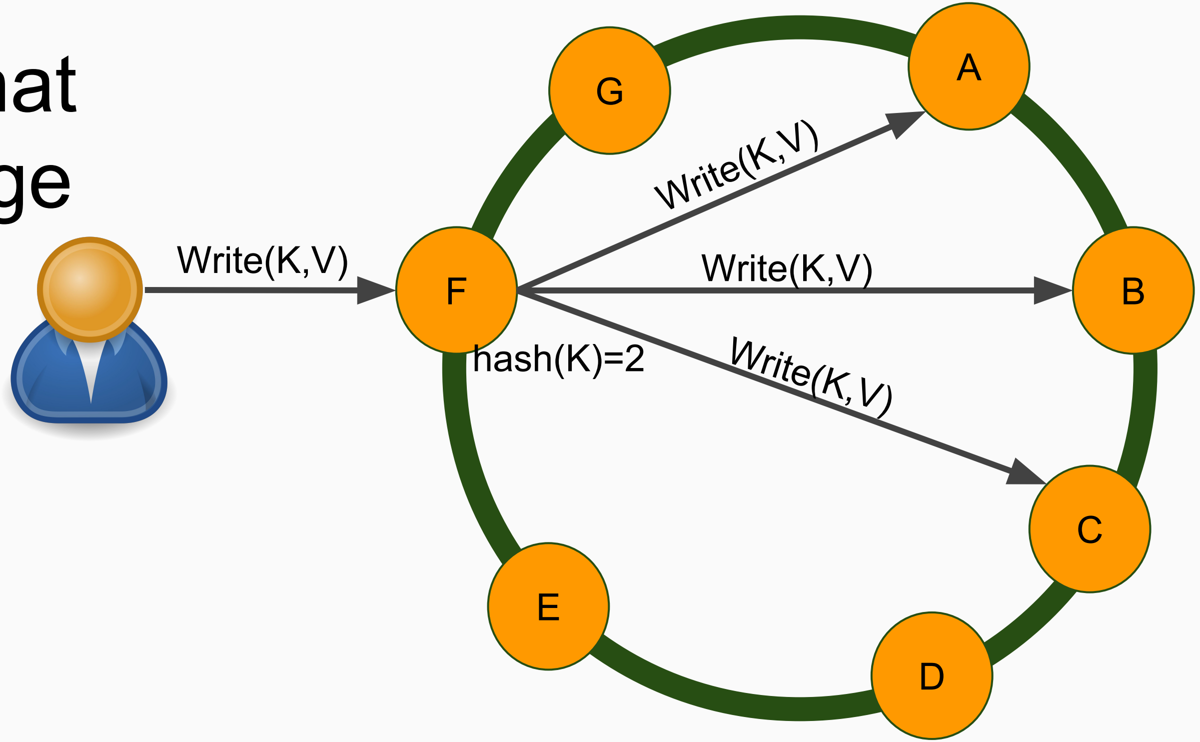


Cassandra - The Ring

- Distributed Hash Table
- Replication
- Full membership view (gossip based)

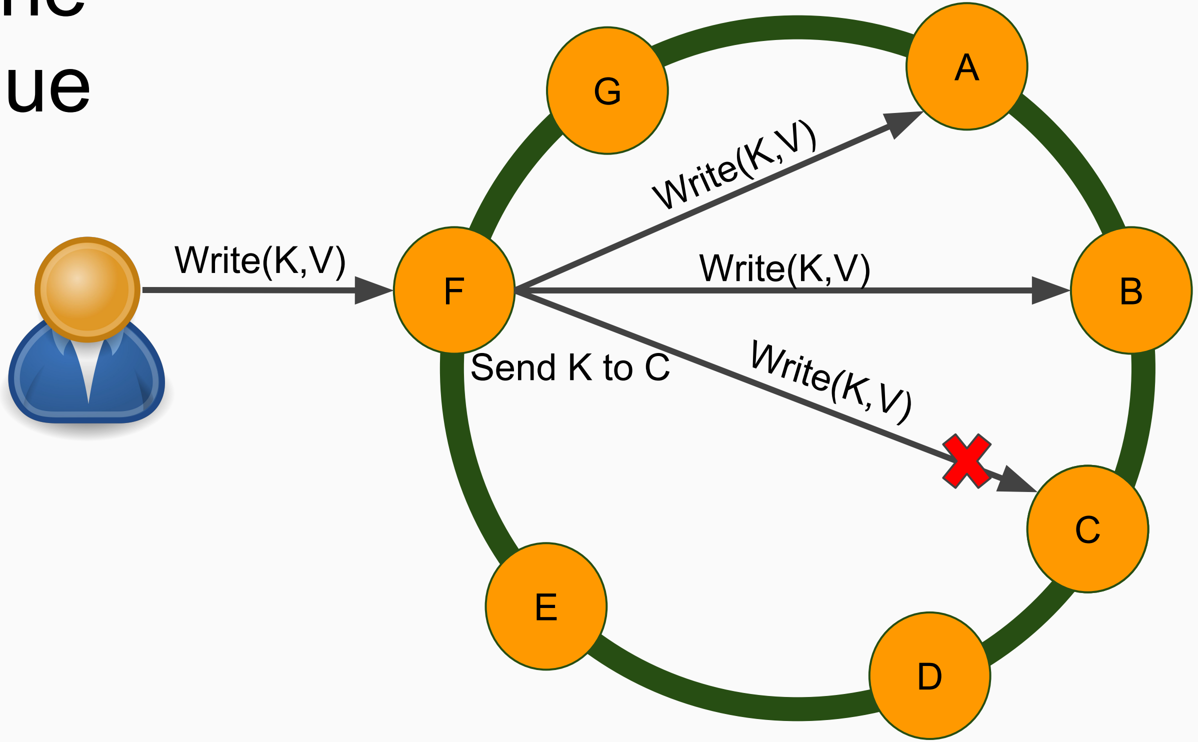


Client decides the number of nodes that have to acknowledge the operation



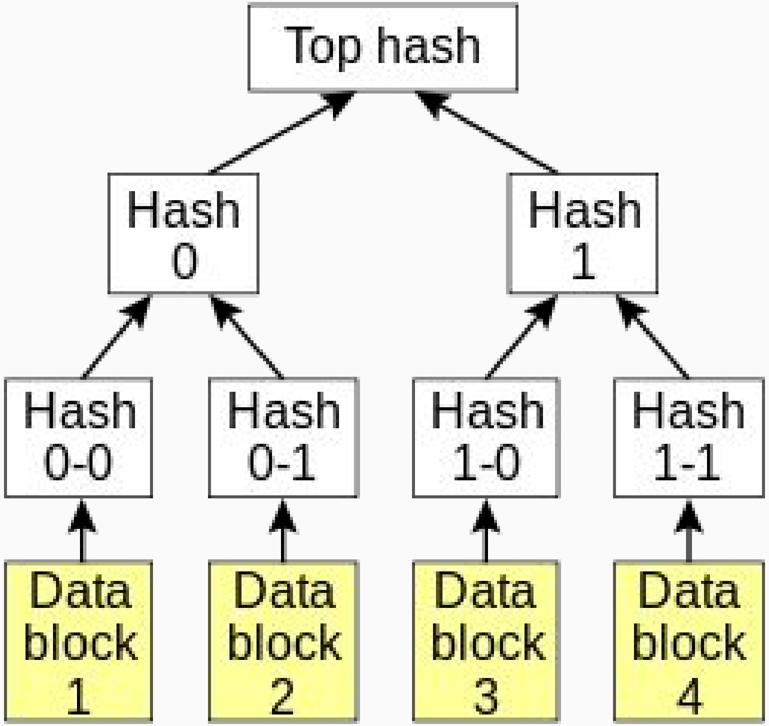
Cassandra - Handling Failures - Hinted Handoff

On a node failure, the proxy saves the value



Cassandra - Handling Failures - Anti-Entropy

- If a node is unresponsive for long enough, the saved hint might get deleted
- Nodes can exchange Merkle Trees and sync (expensive)
- A value can be updated during a Read-Repair



Assumptions for Byzantine Hardening

- Fewer than ⅓ of the nodes are Byzantine:

$$\boxed{N = 3f+1}$$

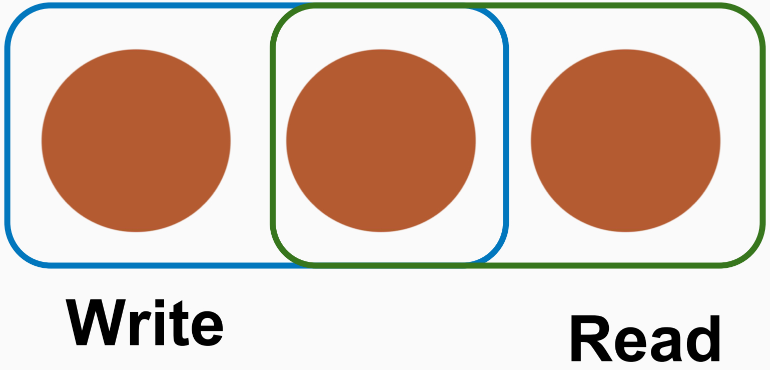
- Fully connected network
- Public Key Infrastructure and SSL
- Loosely synchronized clocks (not perfect)

Replication

- On writes, waiting for all nodes is not possible
- Quorums:
  - All read sets have to intersect with all write sets
  - In Cassandra majority is used:

N=2f+1 **W=f+1** **R=f+1**

f=1



Byzantine Replication?

f=1

Read 2 **Read 6**

Byzantine Replication

Using Byzantine quorums:

- Writes and reads intersect in at least one ***correct*** node

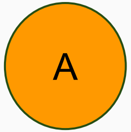
N=3f+1 W=2f+1 R=2f+1

Digital Signatures

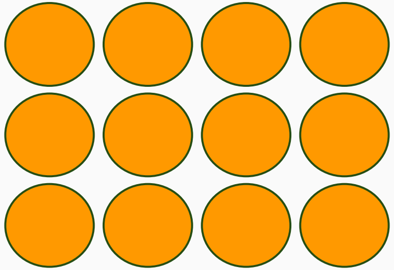
- A proof for the origin of the data
- Requires a shared key

Public Key Signatures:

Sign with a private key



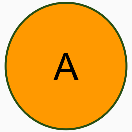
Verify with a public key



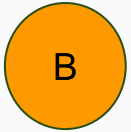
Slow

Symmetric Key Signatures:

Sign with a private key



Verify with a private key

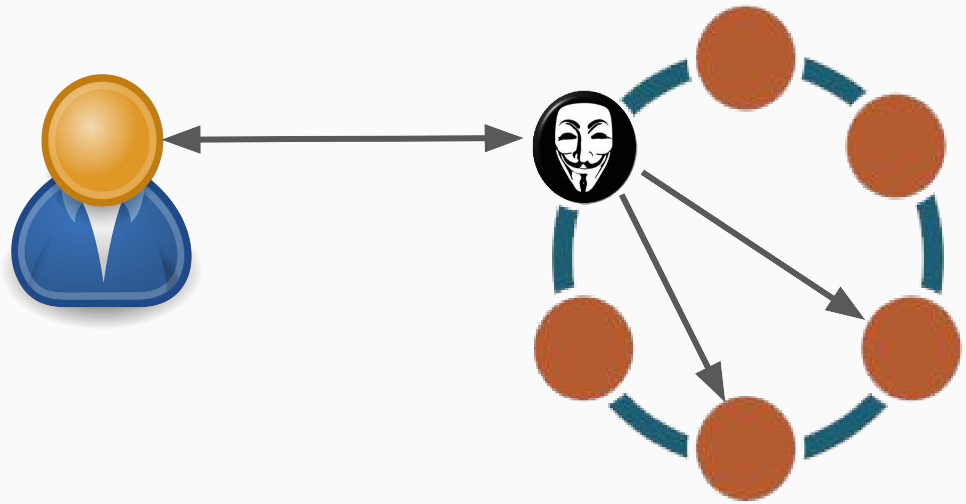


B cannot prove to a third party that he got a message from A

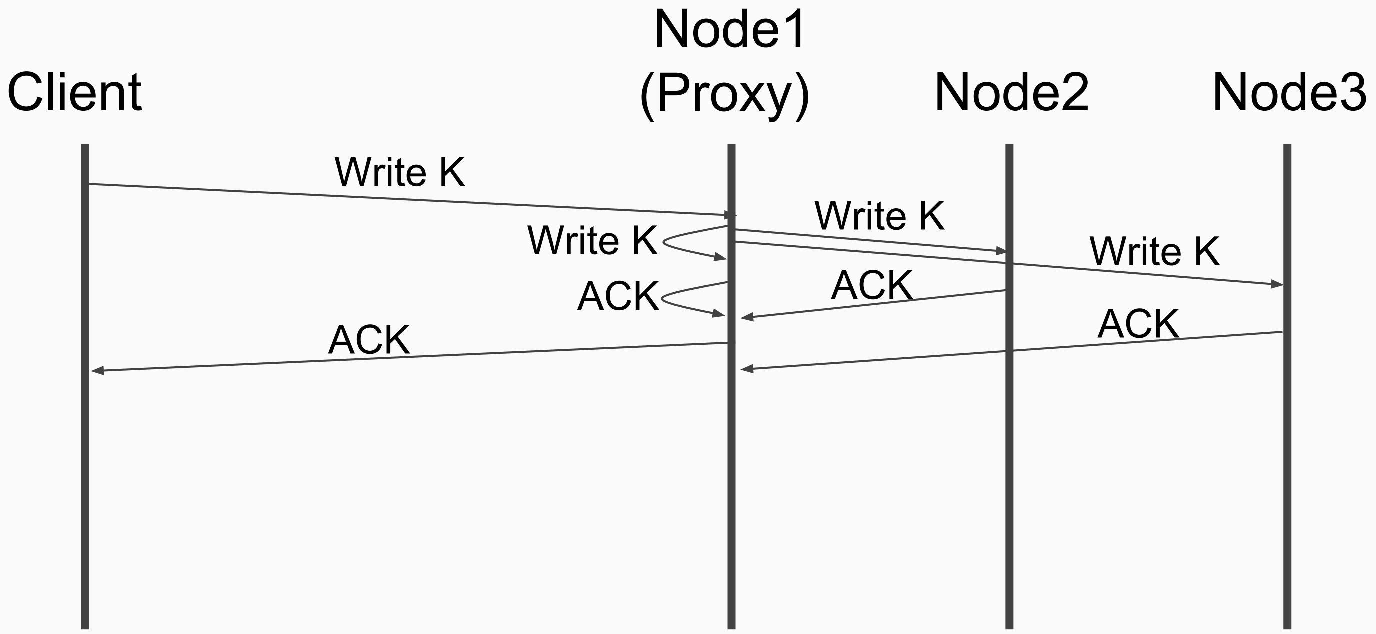
Fast

The Proxy - Plain Cassandra

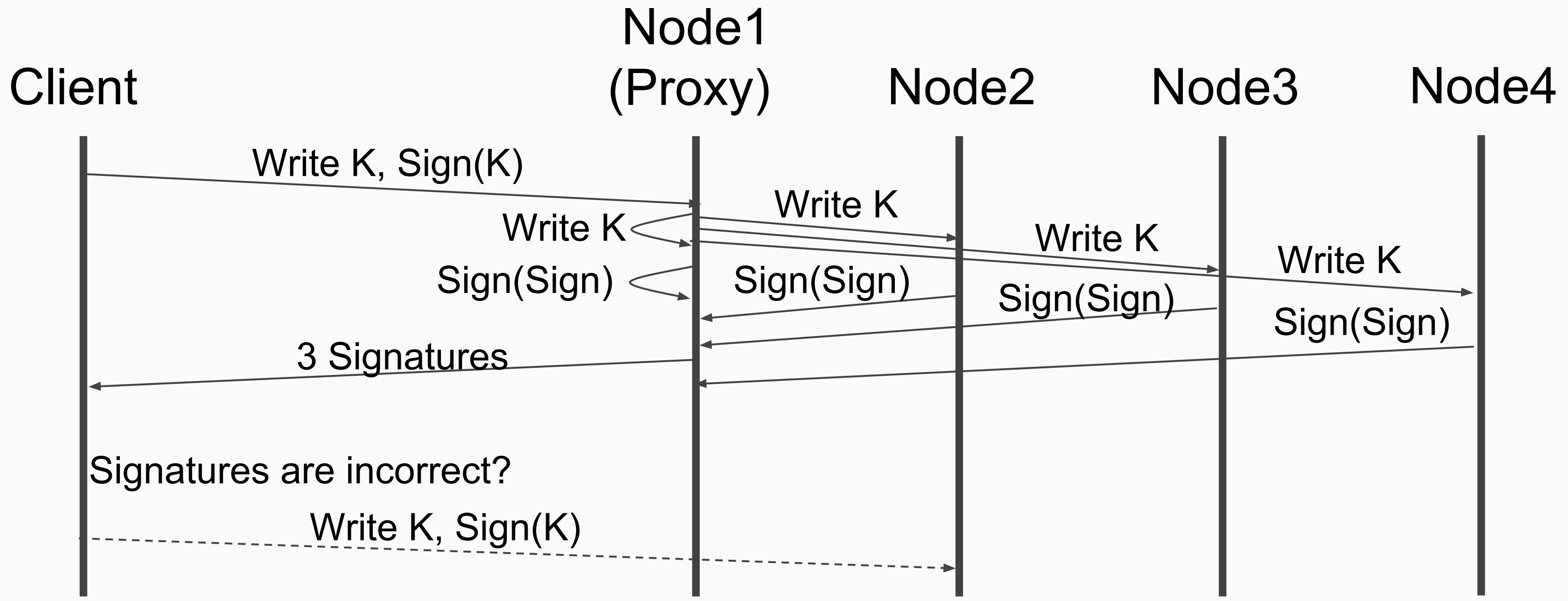
The proxy cannot be trusted



Write Algorithm - Plain Cassandra



Write Algorithm - Hardened Cassandra (Option 1)



Signatures and verifications (using only PKI):

Client:
Sign: 1
Verify: 2f+1

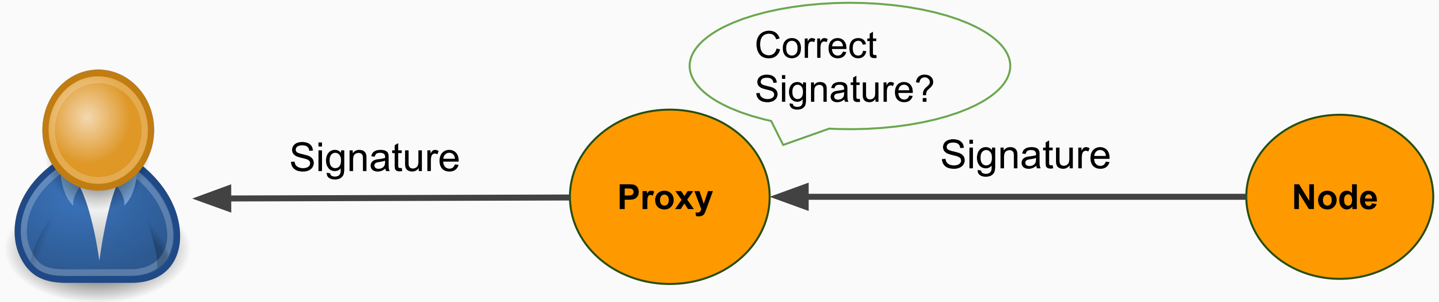
Proxy:
Verify: 2f+1

Nodes:
Sign: 3f+1
Verify: 3f+1

Total:
Sign: 3f+2
Verify: 7f+1

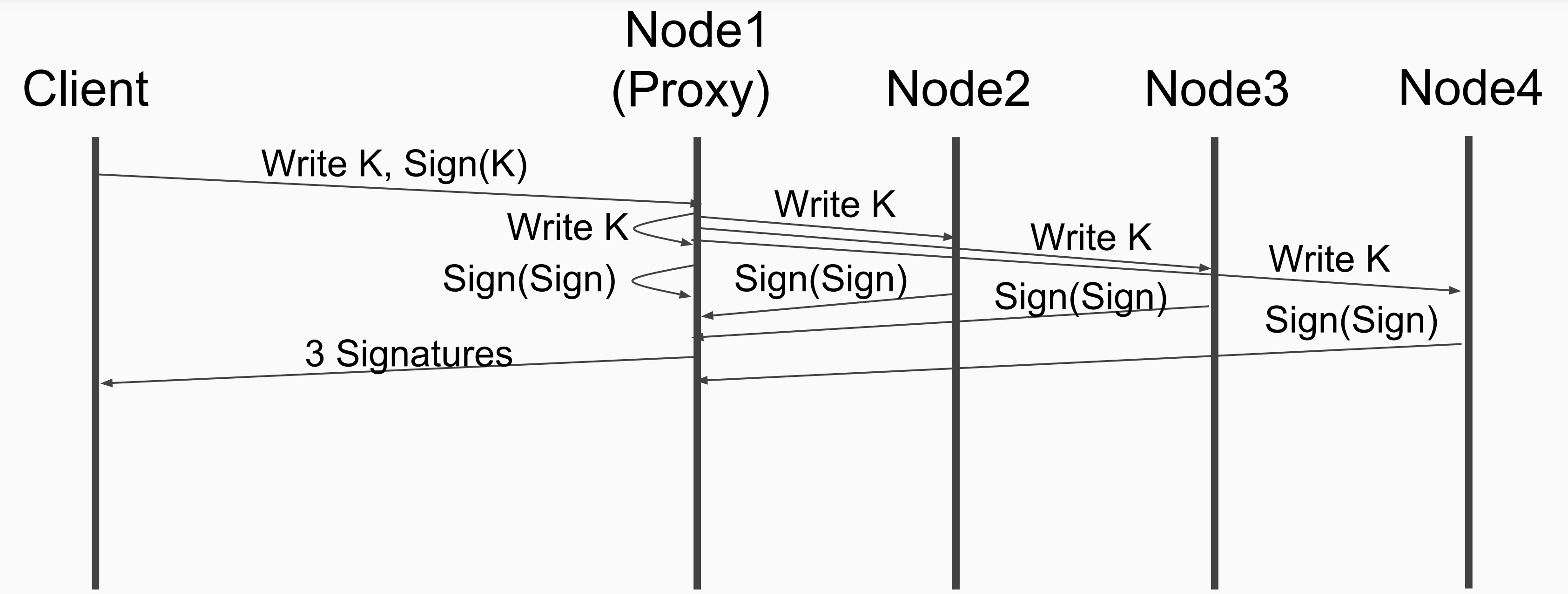
Write Algorithm - Hardened Cassandra (Optimization 1)

- Should the proxy verify the nodes signatures?



- No, if client isn't happy, contact the proxy again...

Write Algorithm - Hardened Cassandra (Optimization 1)



Signatures and verifications (using only PKI):

Client:
Sign: 1
Verify: 2f+1

Proxy:

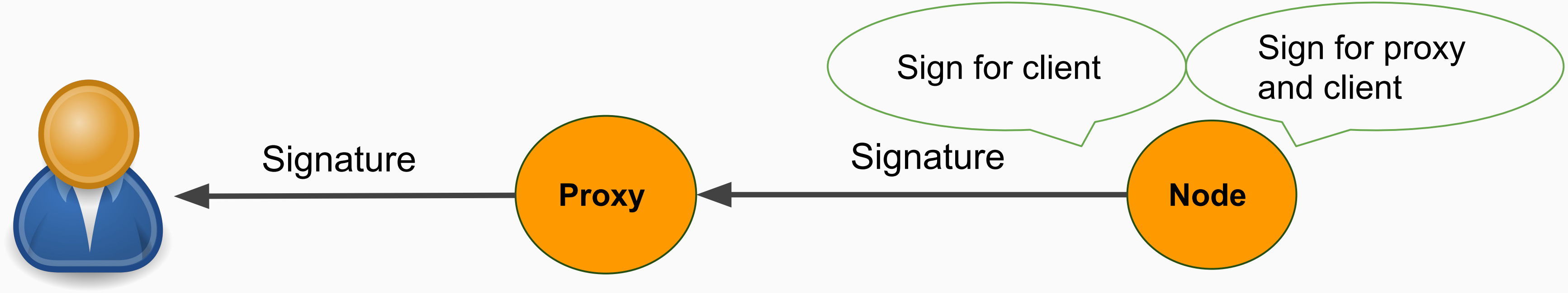
Verify: 2f+1

Nodes:
Sign: 3f+1
Verify: 3f+1

Total:
Sign: 3f+2
Verify: 5f+1

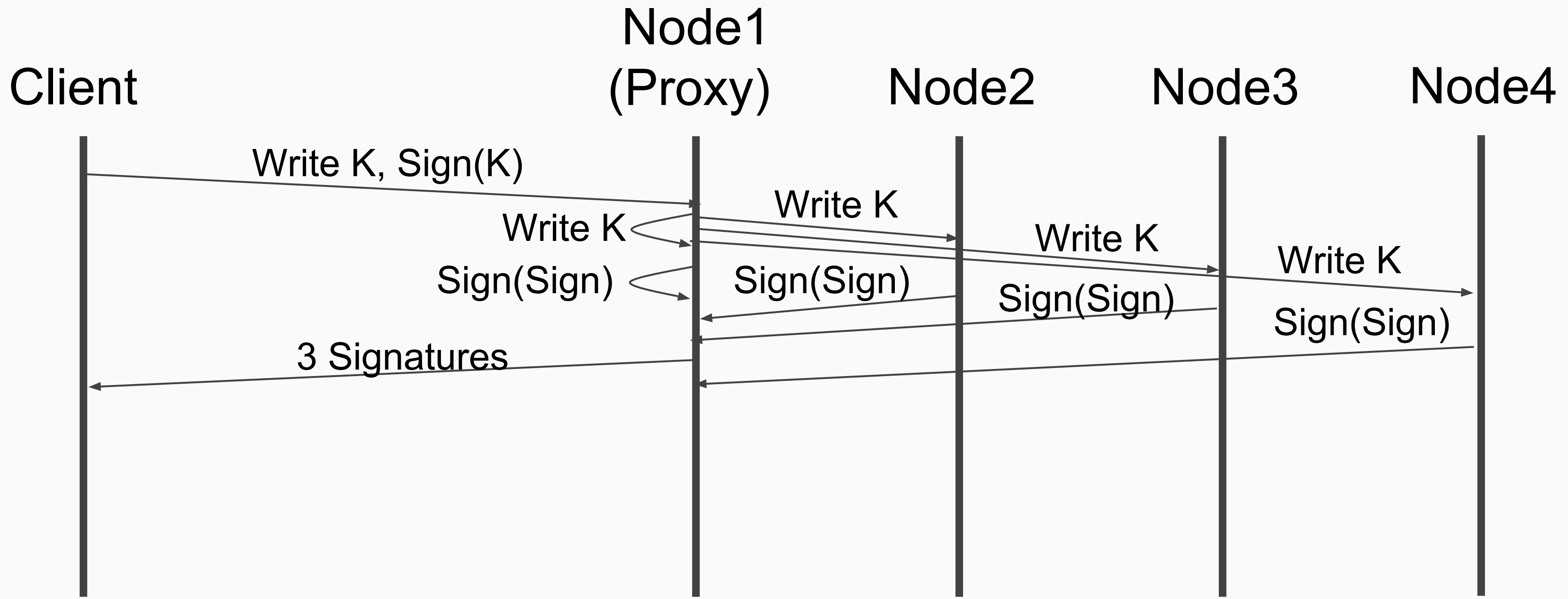
Write Algorithm - Hardened Cassandra (Optimization 2)

- Now, the nodes sign and only the client verify it…



- Switch to symmetric key signatures from nodes to client

Write Algorithm - Hardened Cassandra (Optimization 2)



Signatures and verifications:

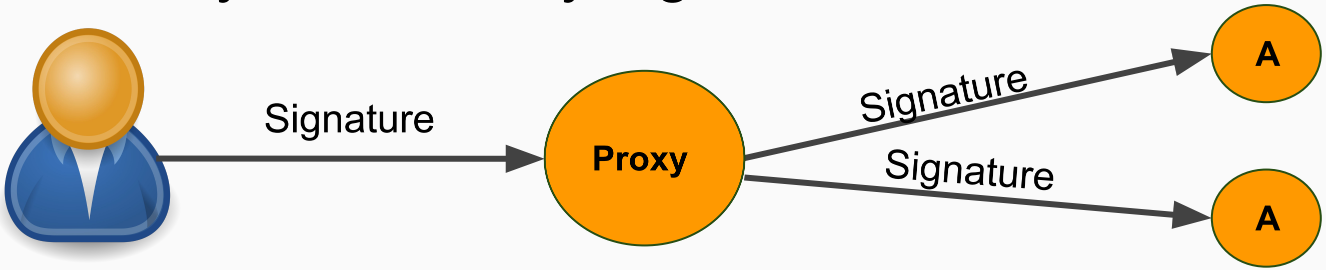
Client:
Sign: 1(p)
Verify: 2f+1(s)

Nodes:
Sign: 3f+1(s)
Verify: 3f+1(p)

Total:
Sign: 3f+1(s) & 1(p)
Verify: 2f+1(s) & 3f+1(p)

Write Algorithm - Hardened Cassandra (Optimization 3)

- Still, not fast enough
- Switch to symmetric key signatures from client to nodes?



- If new nodes join, how can they verify the signature?
- If a node misses a write, how can it trust his neighbours?
- How can the client know which nodes are responsible for each value?

Using only symmetric signatures is tricky...

Write Algorithm - Hardened Cassandra (Optimization 3)

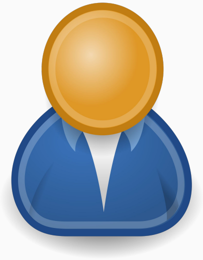
- Still, no
- Switch ... odes?

- If new ... ?
- If a noc ... ours?
- How ca ... ble for each va

y...

Write Algorithm - Hardened Cassandra (Optimization 3)

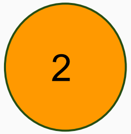
- A client signs the value with a public key signature
- Then, covers the value and signature with symmetric signatures, one for each node
- A node will verify only the symmetric signature and store the public signature



Value

Public_Sign(V)

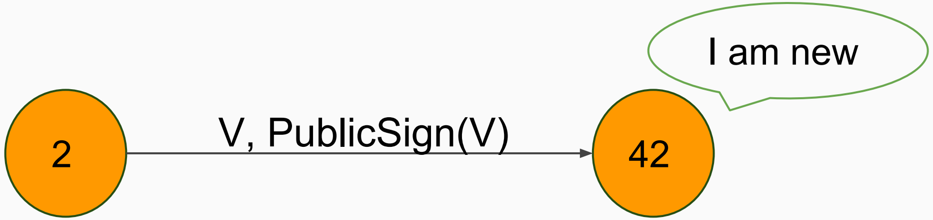
Symmetric_Sign_Node1(V, PS)
Symmetric_Sign_Node2(V, PS)
Symmetric_Sign_Node3(V, PS)
Symmetric_Sign_Node4(V, PS)



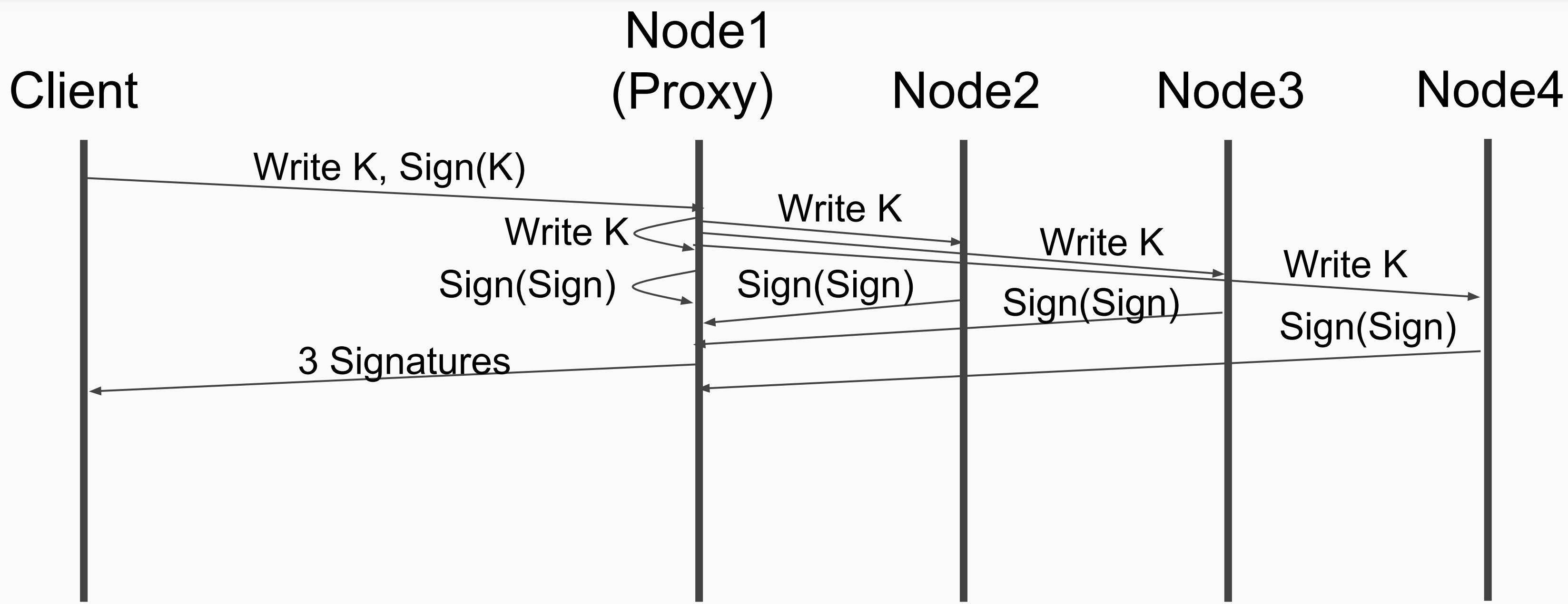
Verify:
Symmetric_Sign_Node2(V, PS)
Store:
Value, Public_Sign(V)

Write Algorithm - Hardened Cassandra (Optimization 3)

- Existing nodes will use only the symmetric signatures
- New nodes / outdated nodes will use the public key signature



Write Algorithm - Hardened Cassandra (Optimization 3)



Signatures and verifications:

Client:
Sign: 1(p) & 3f+1(s)
Verify: 2f+1(s)

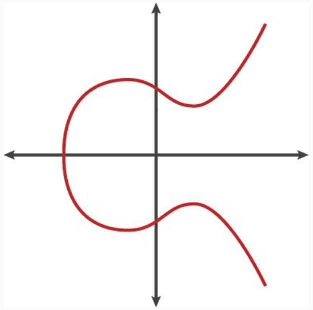
Nodes:
Sign: 3f+1(s)
Verify: 3f+1(s)

Total:
Sign: 6f+1(s) & 1(p)
Verify: 5f+2(s)

Write Algorithm - Hardened Cassandra (Optimization 4)

- Left with only one public key signature
- Can we do it fast?

ECDSA (The Elliptic Curve Digital Signature Algorithm)



Fast Signing

Slow Verification

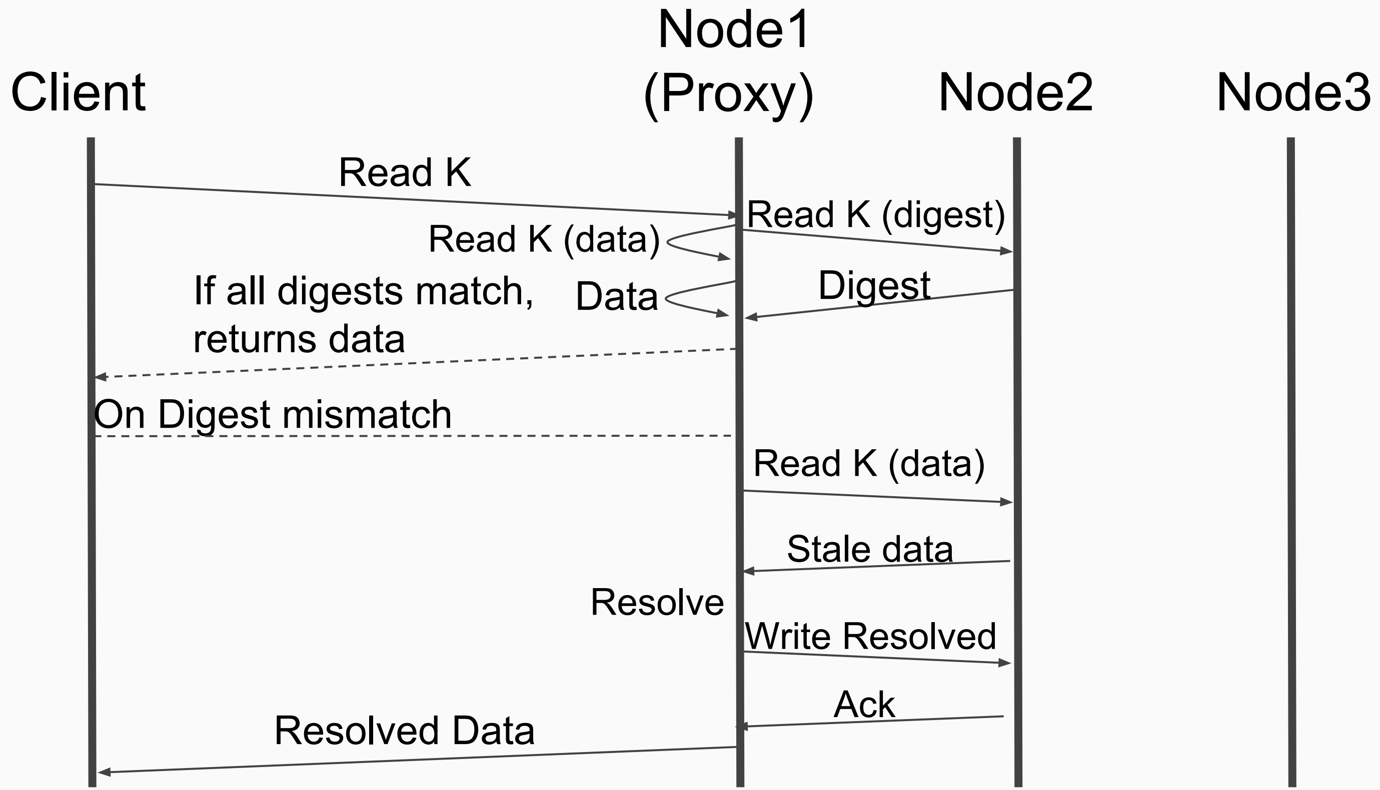
RSA (Rivest, Shamir, and Adleman)

$$C = M^e \bmod n$$

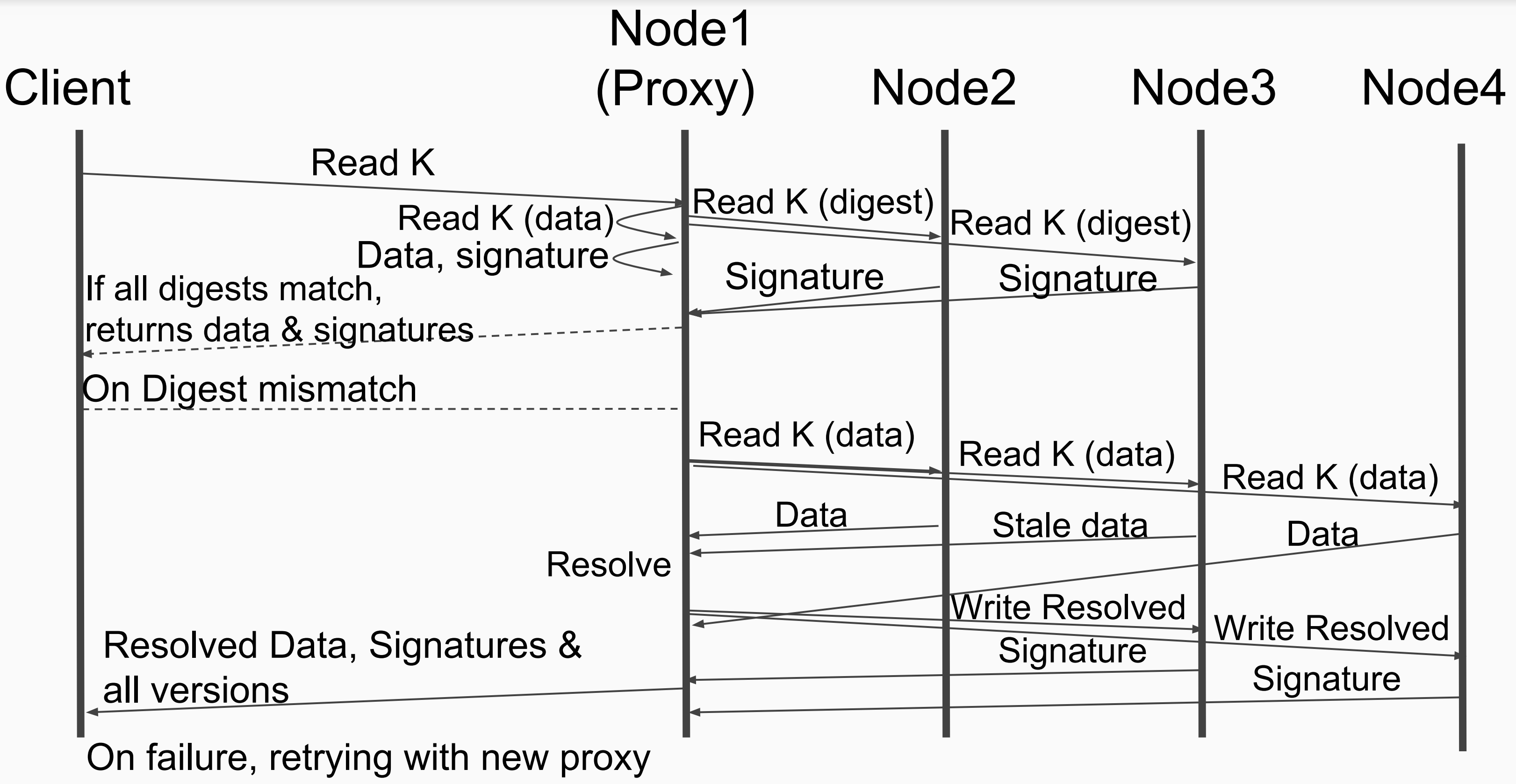
Slow Signing

Fast Verification

Read Algorithm - Plain Cassandra



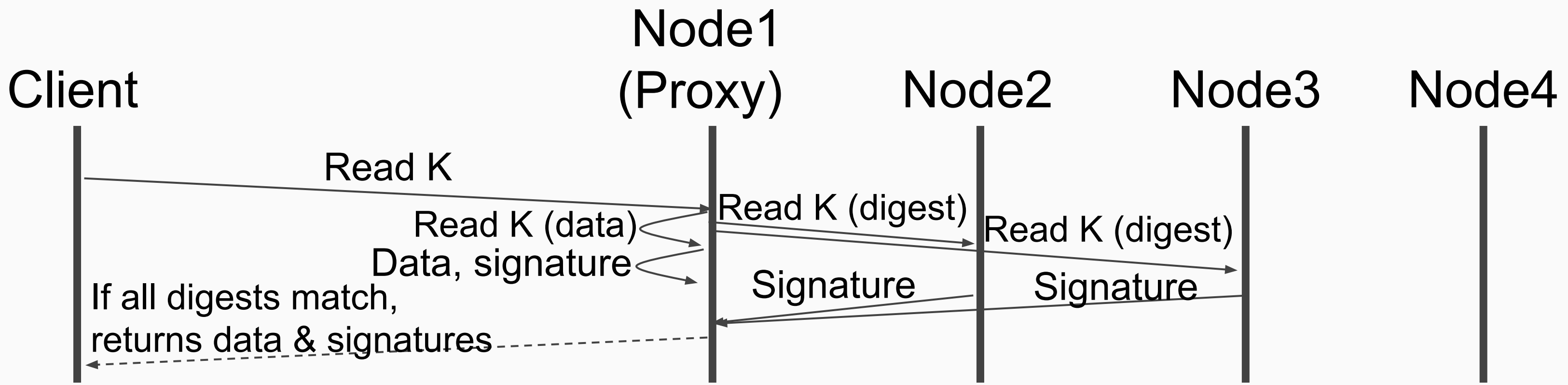
Read Algorithm - Hardened Cassandra



Read Algorithm - Hardened Cassandra (Optimizations)

Same as in the write path:

- Proxy does not verify, client contacts it again if necessary
- Symmetric signatures from nodes to client



Signatures and verifications:

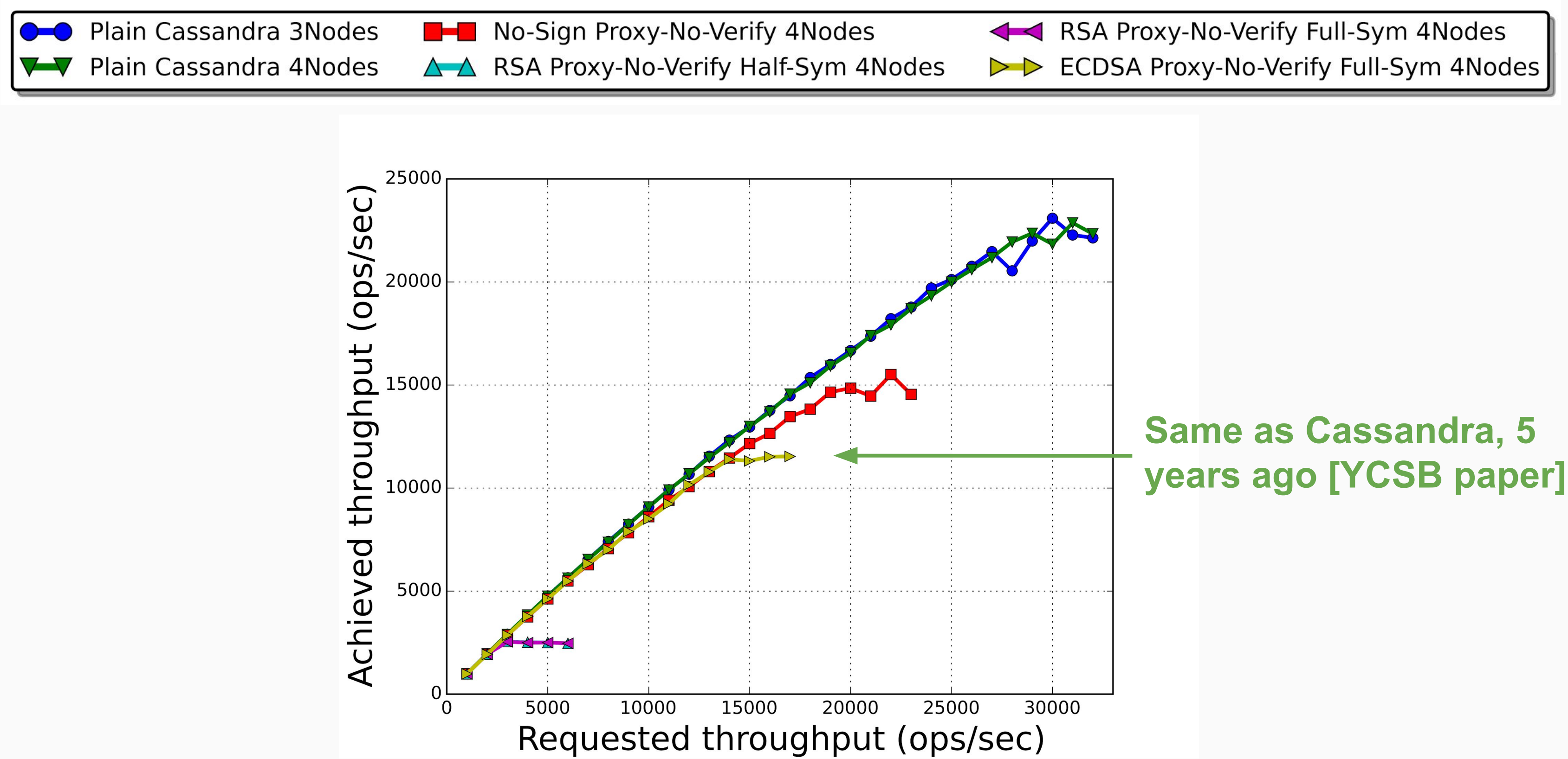
Client:
Verify: 2f+1(s)

What about verifying the data signature?

Nodes:
Sign: 2f+1(s)

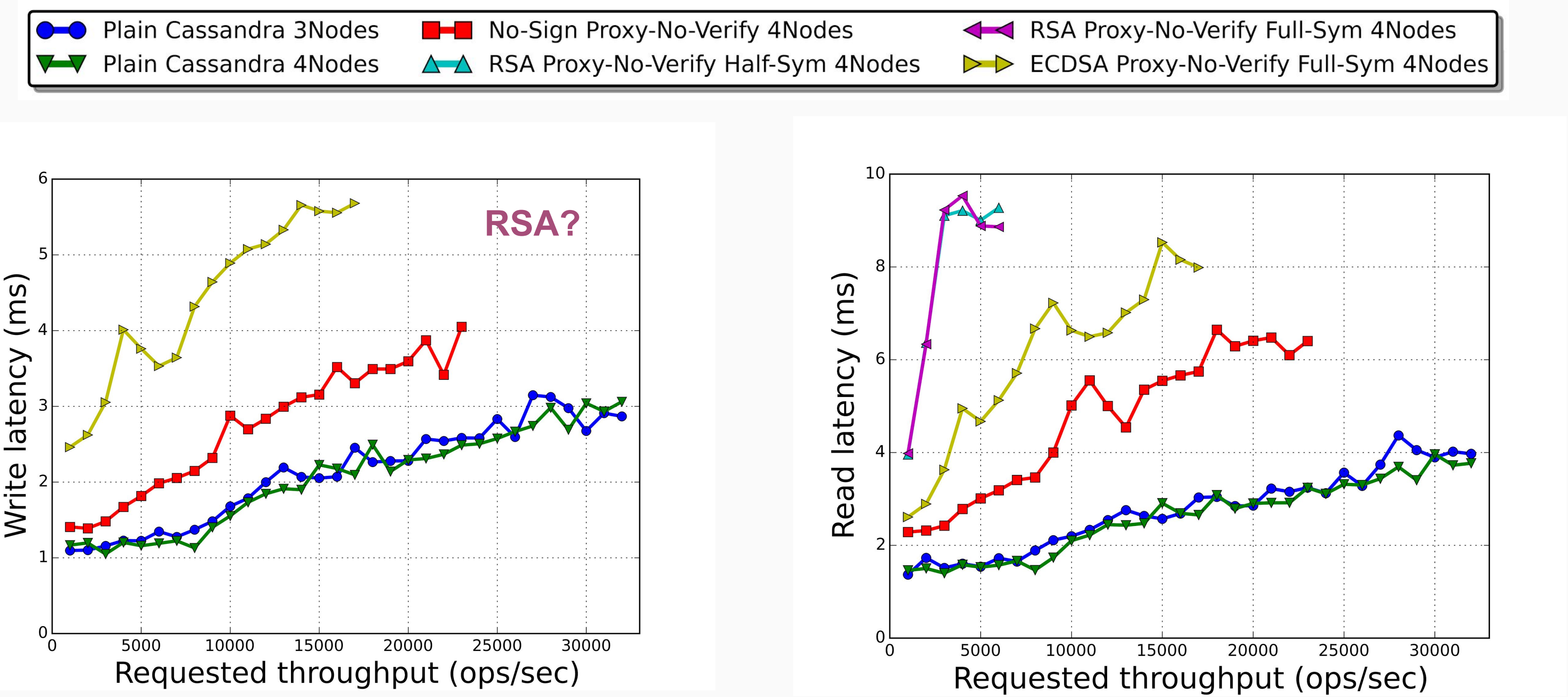
Total:
Sign: 2f+1(s)
Verify: 2f+1(s)

Performance - YCSB - Workload A - 50/50 Read/Writes - Achieved Throughput

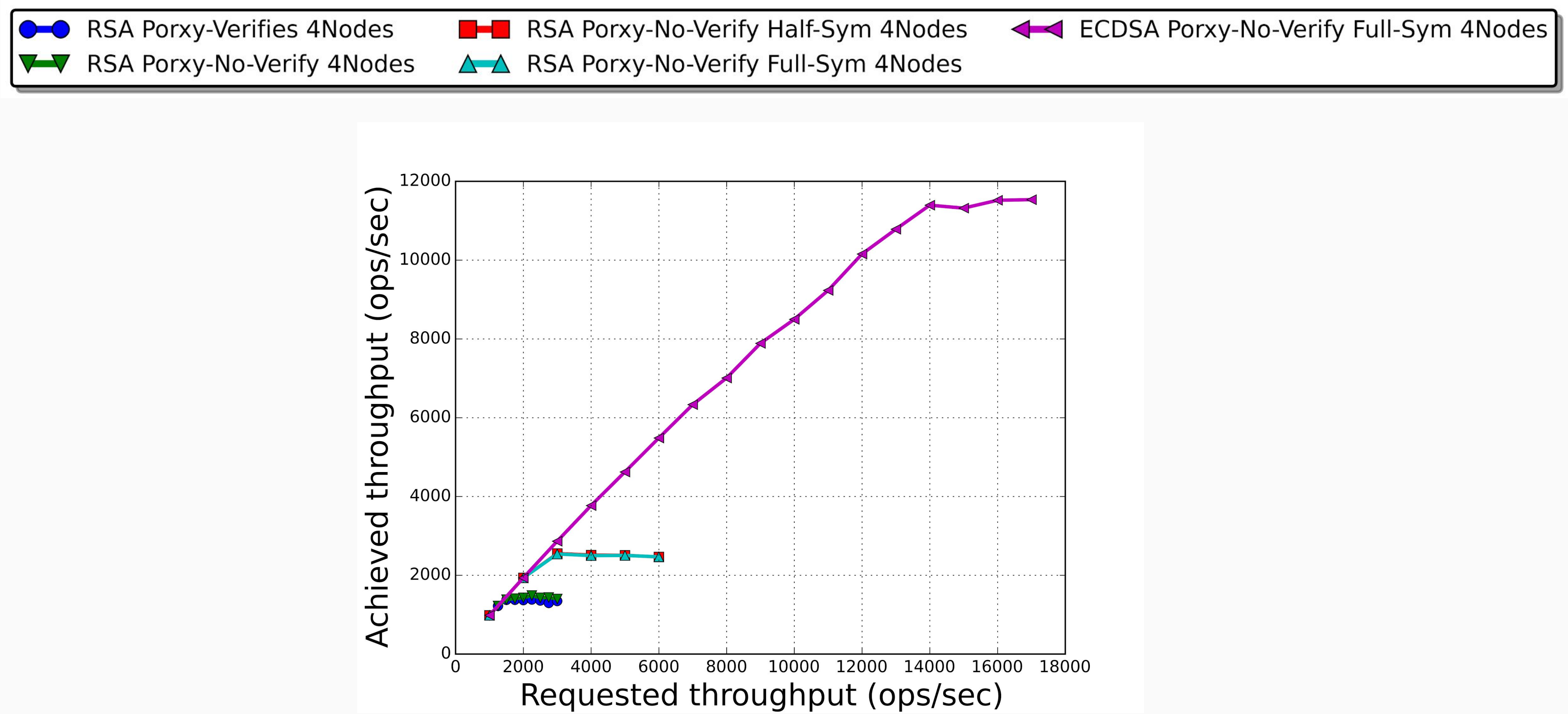


Same as Cassandra, 5 years ago [YCSB paper]

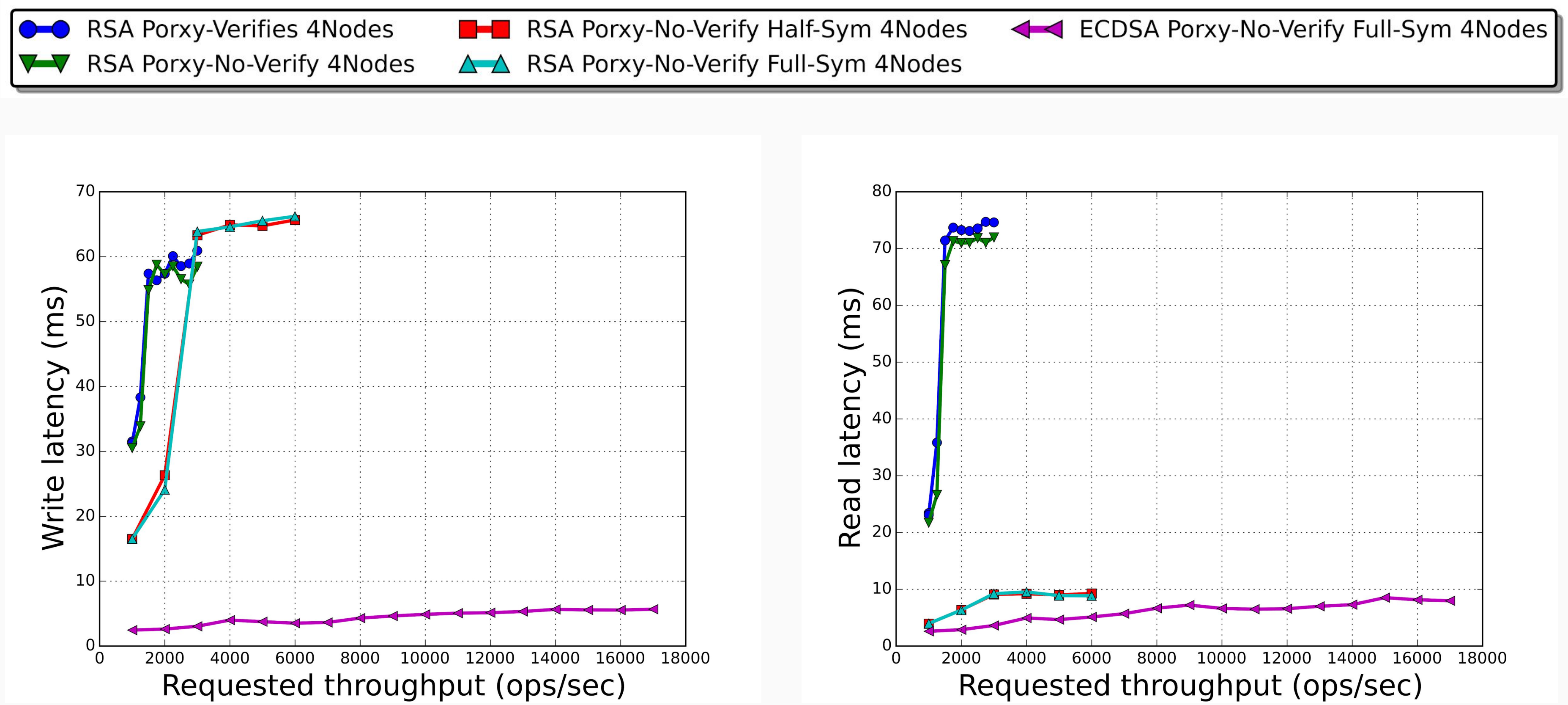
Performance - YCSB - Workload A - 50/50 Read/Writes - Latency



Performance - YCSB - Workload A - 50/50 Read/Writes - Achieved Throughput - More



Performance - YCSB - Workload A - 50/50 Read/Writes - Latency - More



Other issues in the Paper

- Byzantine clients
- Deleting values
- Column families
- Membership

Future Work

- Improve performance
  - Introduce real batching
- Support more functionalities
  - Lightweight transactions
  - Multi data-center operations

Questions?

Thank You