In-Network Leaderless Replication for Distributed Data Stores

VLDB 2022, Sydney, Australia, September 2022

Gyuyeong Kim* and Wonjun Lee

Network and Security Research Lab.

School of Cybersecurity

Korea University

NetLab
Network and Security Research Lab.

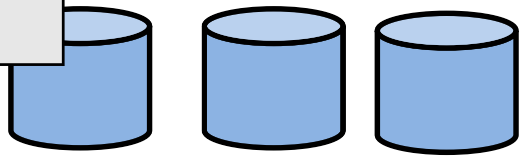
KOREA UNIVERSITY

*Currently at Sungshin Women's University

Distributed Data Stores

- Backbone for modern online services
- NoSQL key-value databases (e.g., Redis, Memcached)

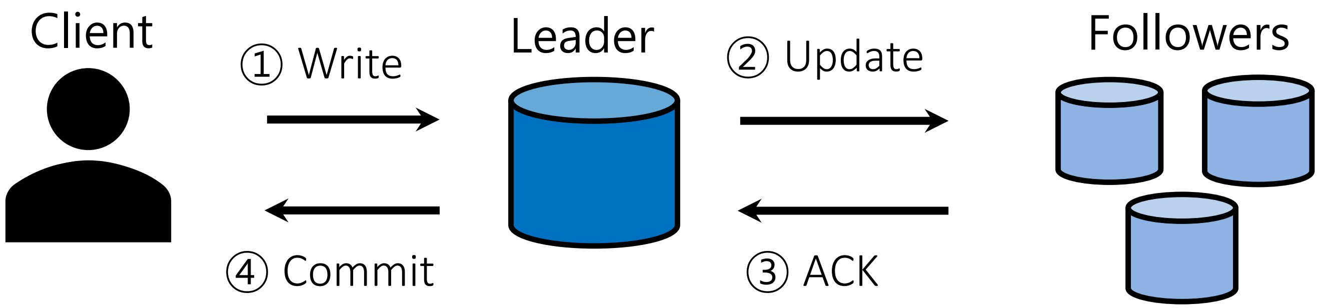
Key	Value
Key1	Value1
Key2	Value2
Key3	Value3



data stores

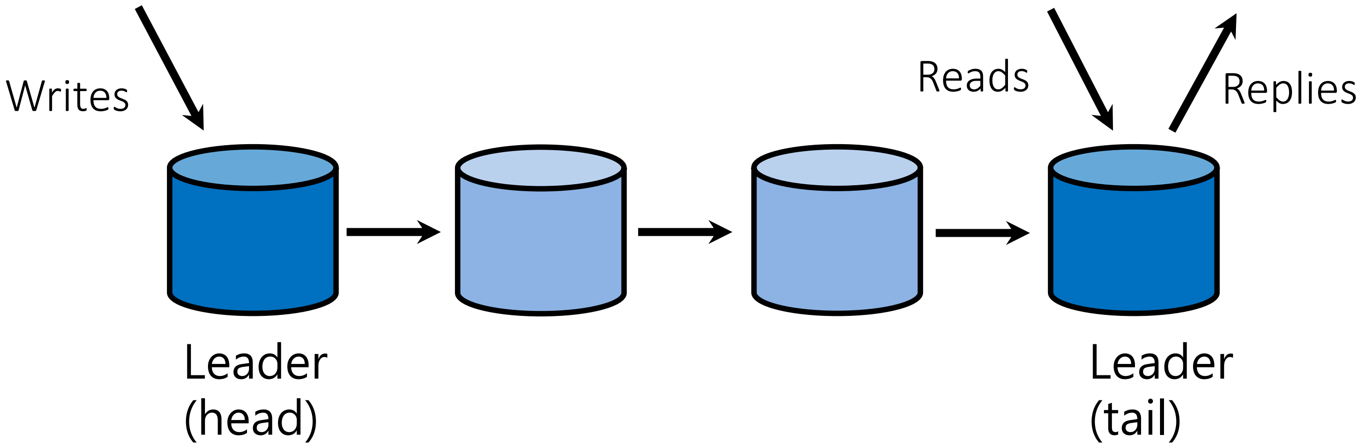
Data Replication 101

- A common technique to mask failures
- Leader-follower structure



Leader-based Replication

- 😃 Easy to ensure strong consistency
- 😟 Leader becomes the performance bottleneck
- 😟 Downtime due to leader election and membership changes



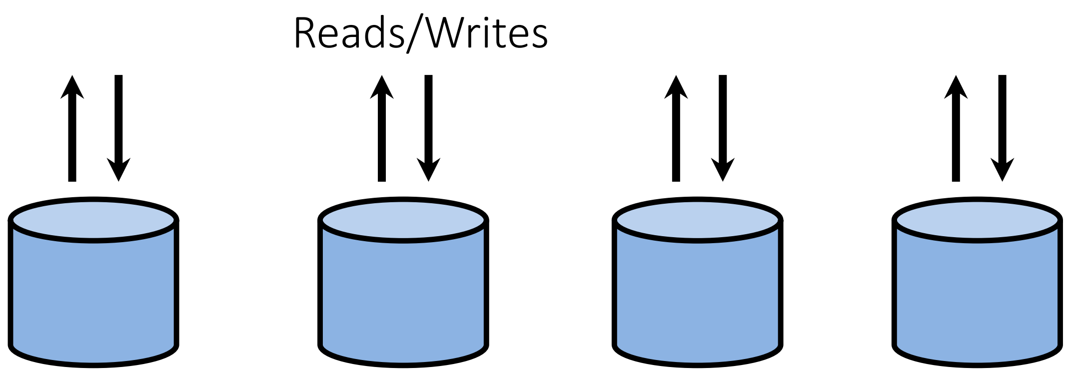
Only head/tail leader handle requests

Chain Replication (CR)@OSDI'04

Leaderless Replication - Pros

😃Scalable read performance by local reads

😃No downtime for leader election



Every replica can serve reads/writes

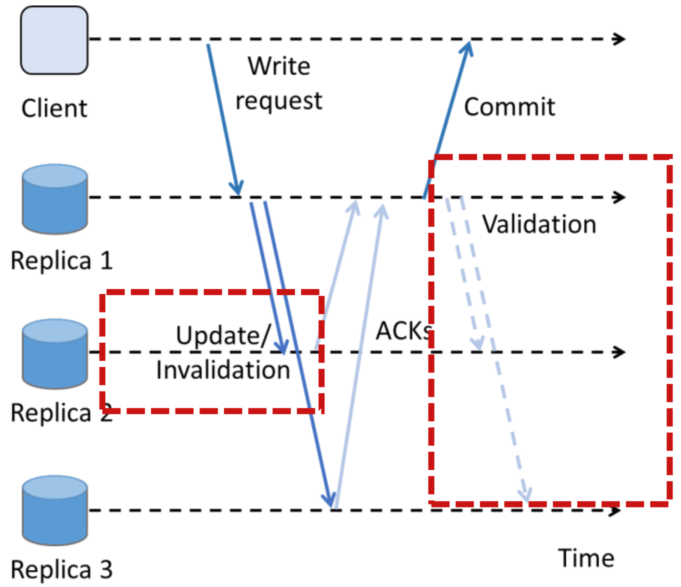
Hermes@ASPLOS'20

Leaderless Replication - Cons

😟 Extra coordination to ensure strong consistency

- Read-write conflicts: read access to an inconsistent object
- Inter-write conflicts: concurrent writes for the same object

😟 Still requires coordination to propagate membership changes



Write coordination with Hermes@ASPLOS'20

Leader-based vs. Leaderless

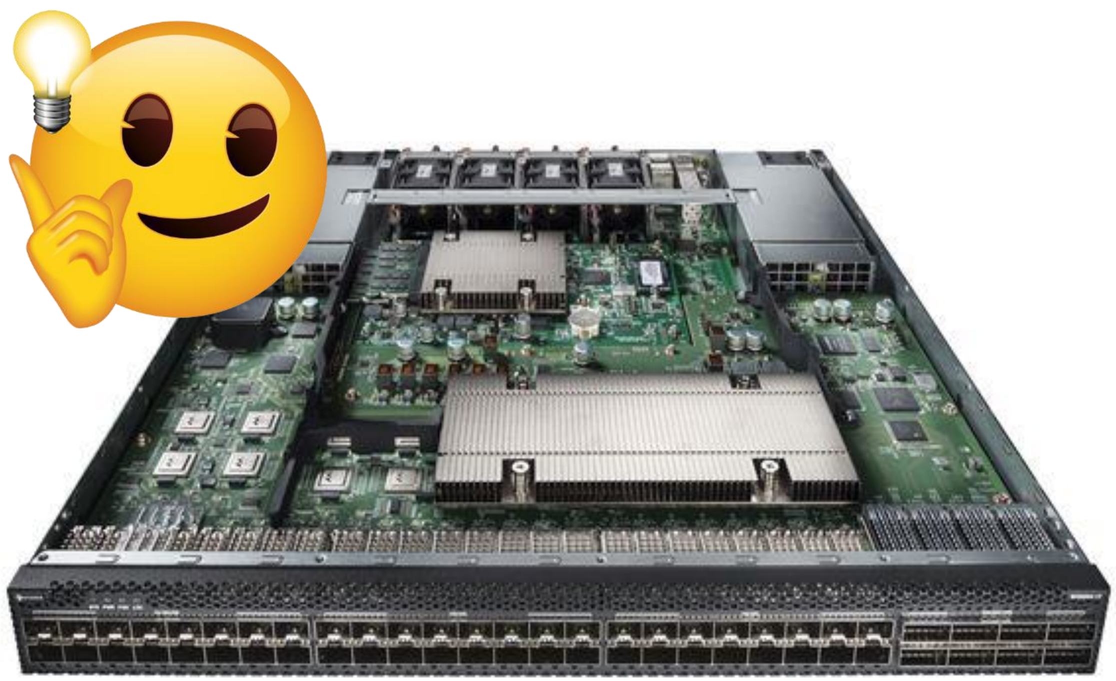
	Leader-based protocol	Leaderless protocol
Strong consistency	O	O
[High perf.] Read scalability	X	O
[High perf.] No inter-replica coordination for writes	X	X
[Fault tolerance] No leader election	X	O
[Fault tolerance] No coordination for membership changes	X	X

How to achieve high performance, strong consistency, and fault tolerance simultaneously?

A Case for In-Network Replication

- Let’s move the entire replication functions into **the network!**

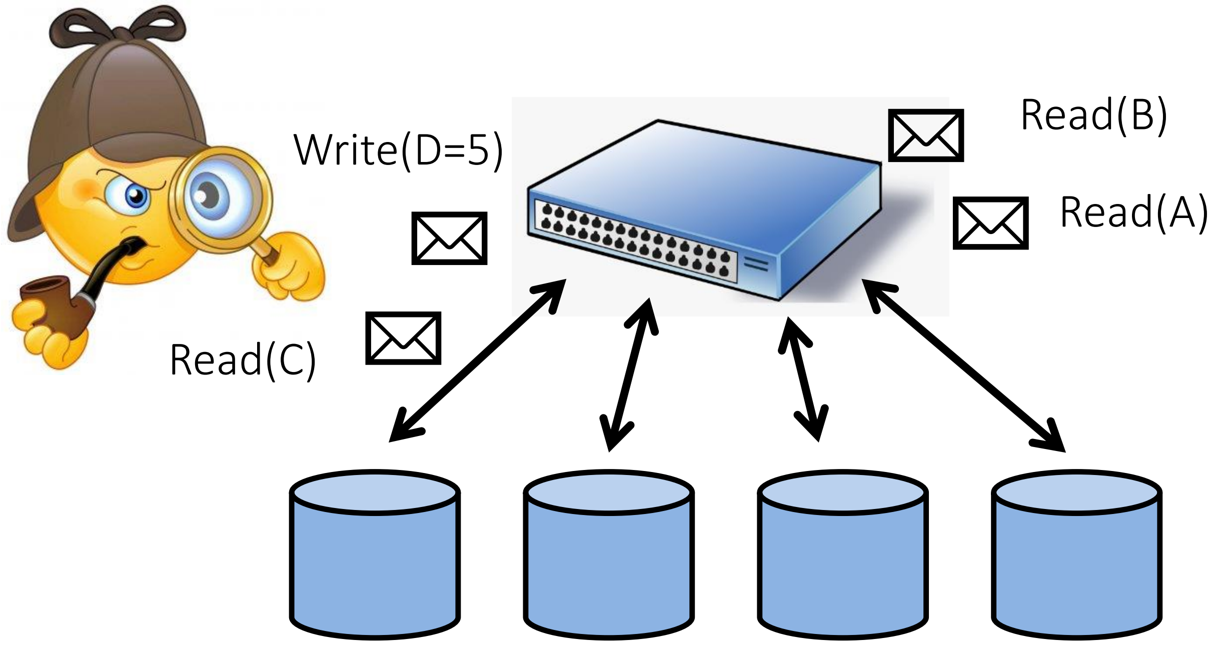
- Emerging programmable switches
  - High performance
  - High Flexibility



Programmable switch with Intel Tofino ASIC

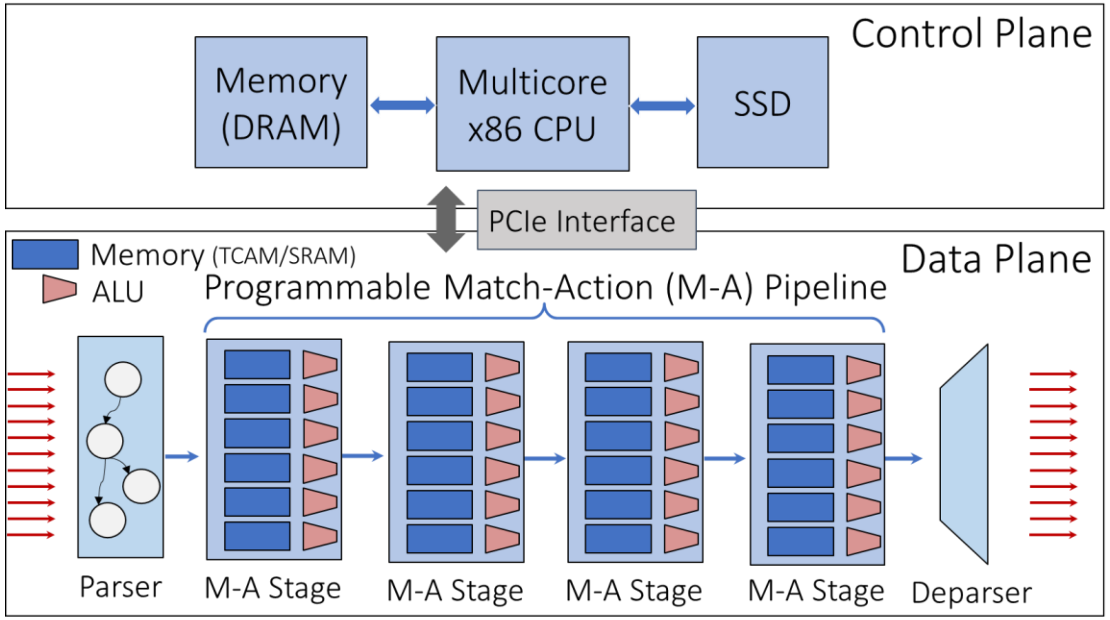
Why In-Network Replication?

- **Global view:** every message passes through the ToR switch
- **Centralized point:** the coordination overhead is due to distributed object/server state management



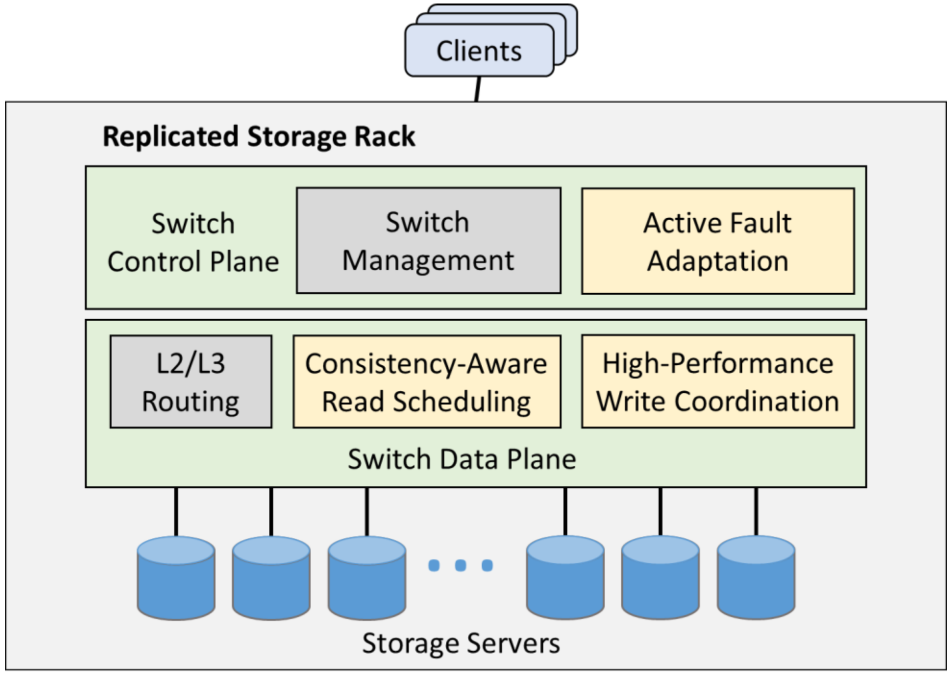
Programmable Switch Architecture

- Switch ASICs like Intel Tofino allow us to program the data plane in P4
  - Programmable parser to **identify replication messages**
  - Stateful memory to **maintain object and server states**
  - Programmable packet processing logic to **perform replication functions**

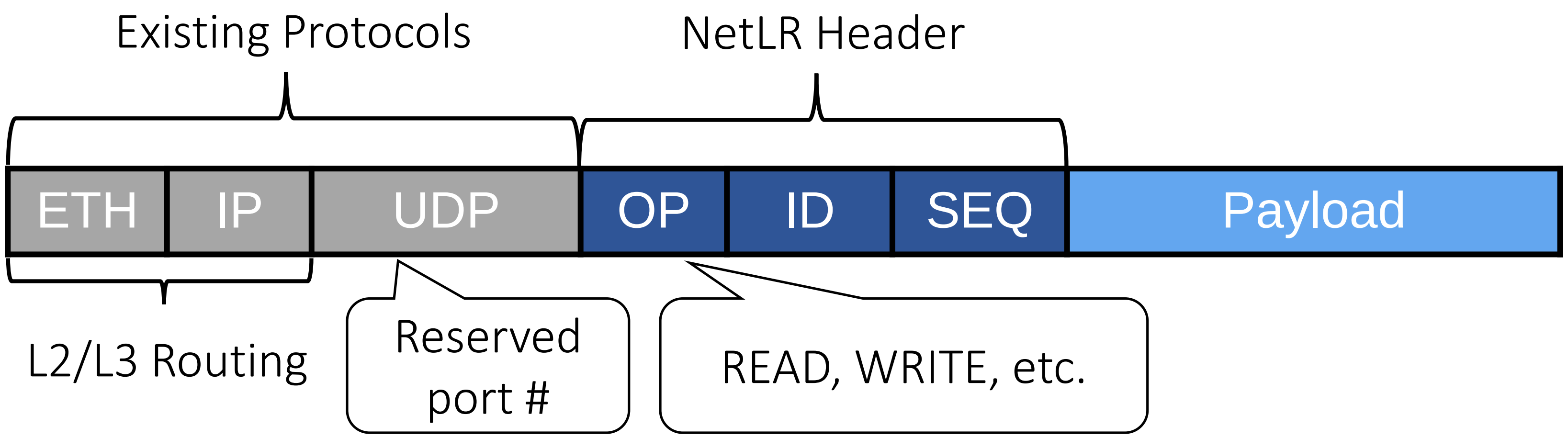


NetLR: In-network Replication Coordinator

- NetLR directly performs data replication in the network
  - No inter-replica coordination for strongly-consistent writes and membership changes
- Consistency-aware read scheduling
  - Forwards requests to consistent replicas only
  - Maintains inconsistent object list
- High-performance write coordination
  - Clones write requests
  - Aggregates replies and commits the write
- Active fault adaptation
  - Centralized membership management
  - Maintains the liveness state of servers
  - Periodic polling to the port status

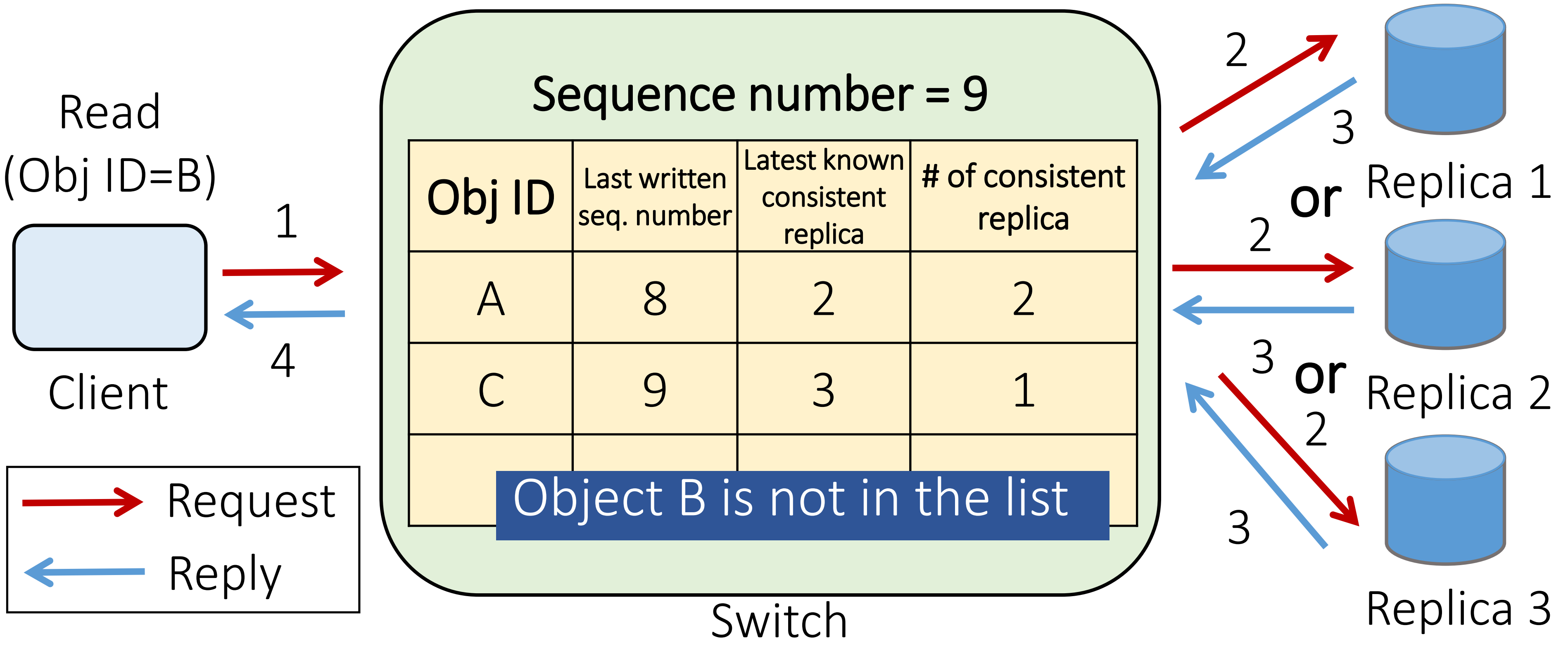


Packet Format



- OP: operation type
- ID: object ID (key)
- SEQ: request sequence number

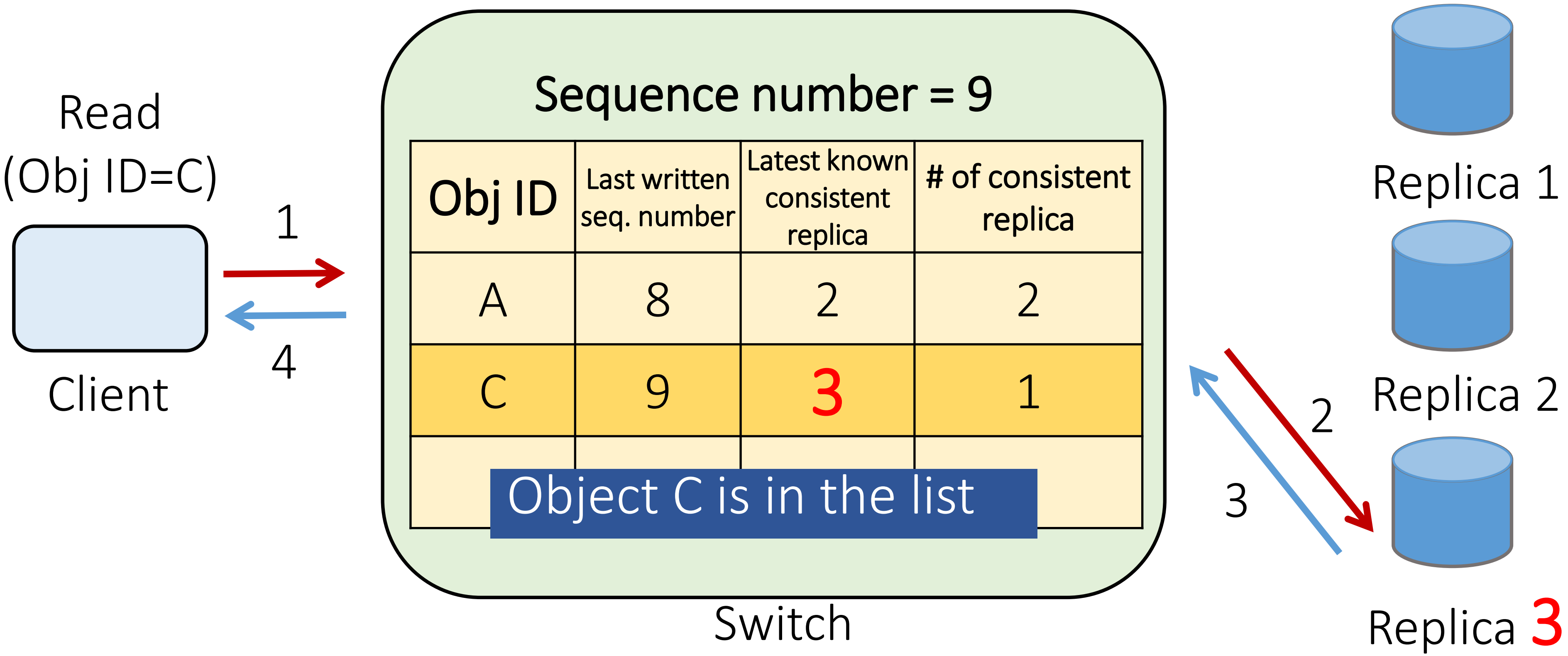
Read Processing for Consistent Objects



Obj ID	Last written seq. number	Latest known consistent replica	# of consistent replica
A	8	2	2
C	9	3	1

Can forward to any replica since **all the replicas have the newest data**

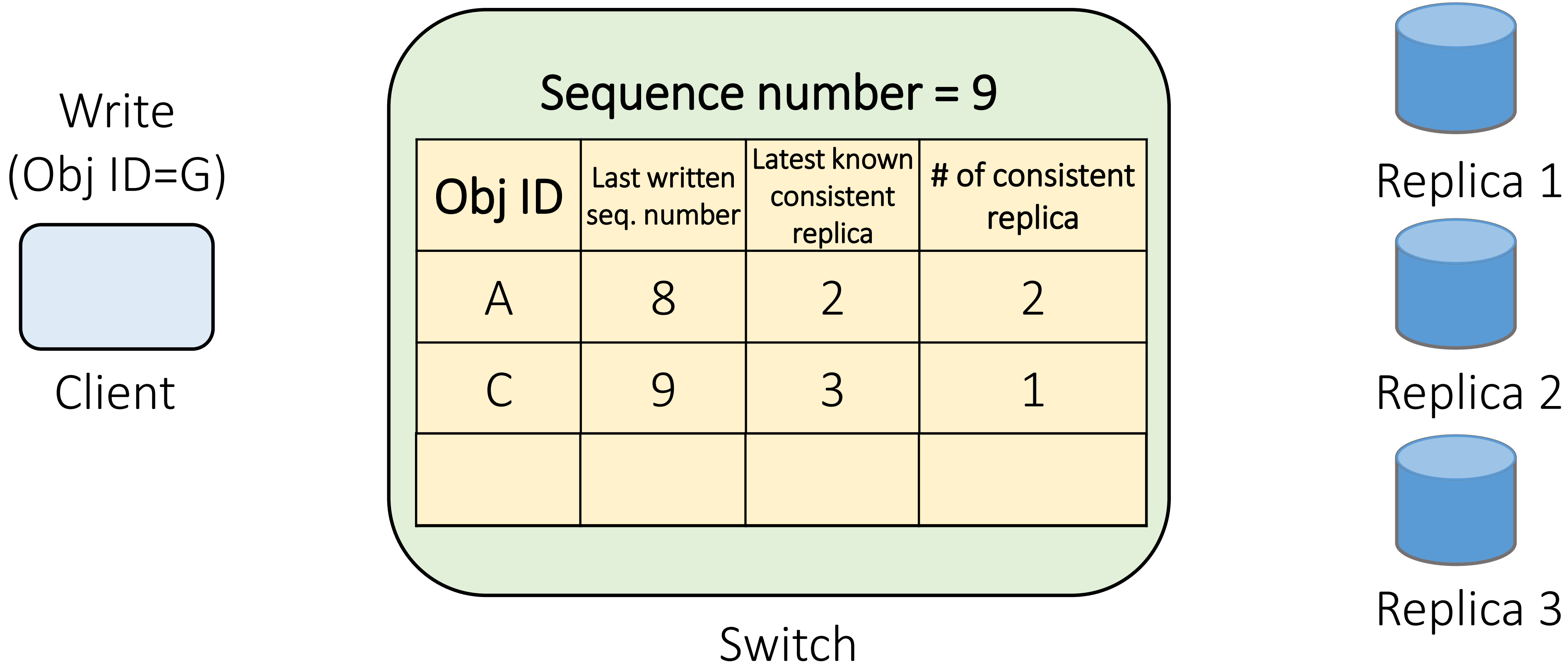
Read Processing for Inconsistent Objects



Obj ID	Last written seq. number	Latest known consistent replica	# of consistent replica
A	8	2	2
C	9	3	1

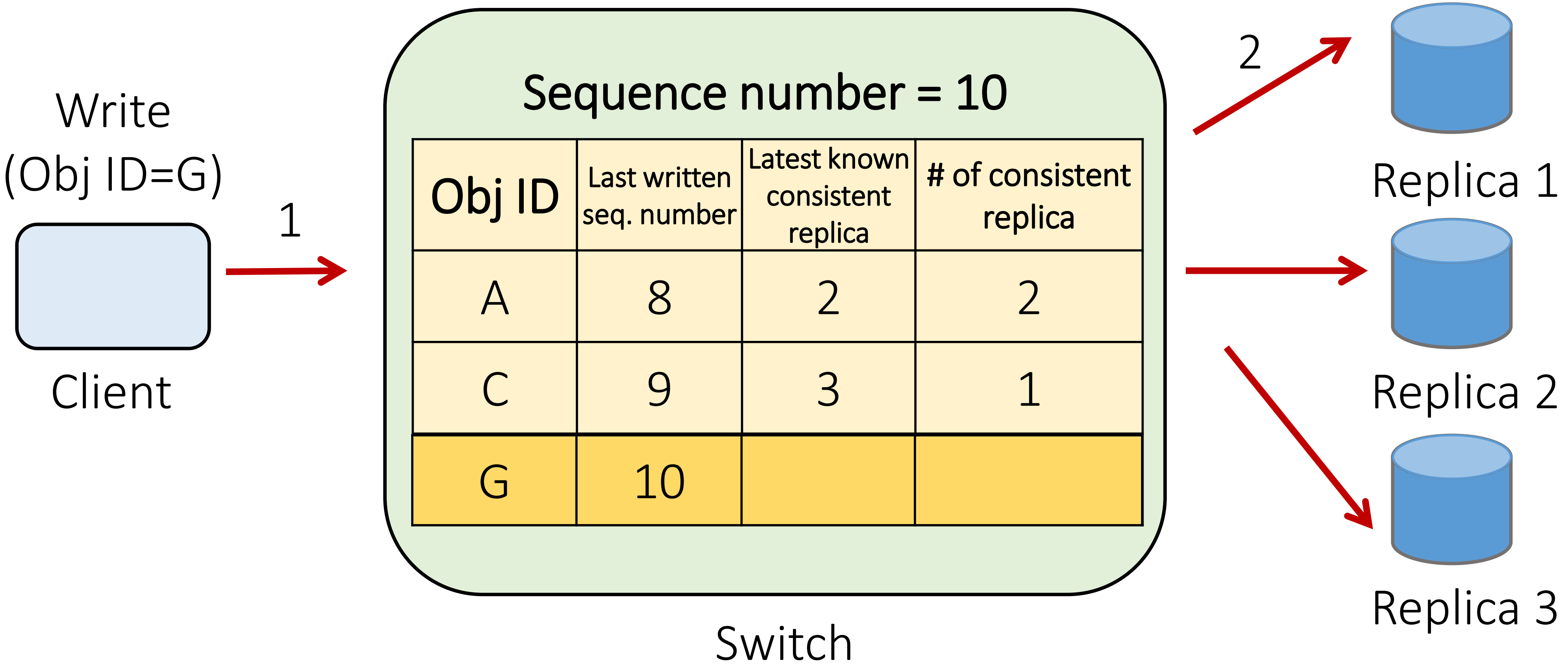
Can forward to the latest known consistent replica only

Write Processing



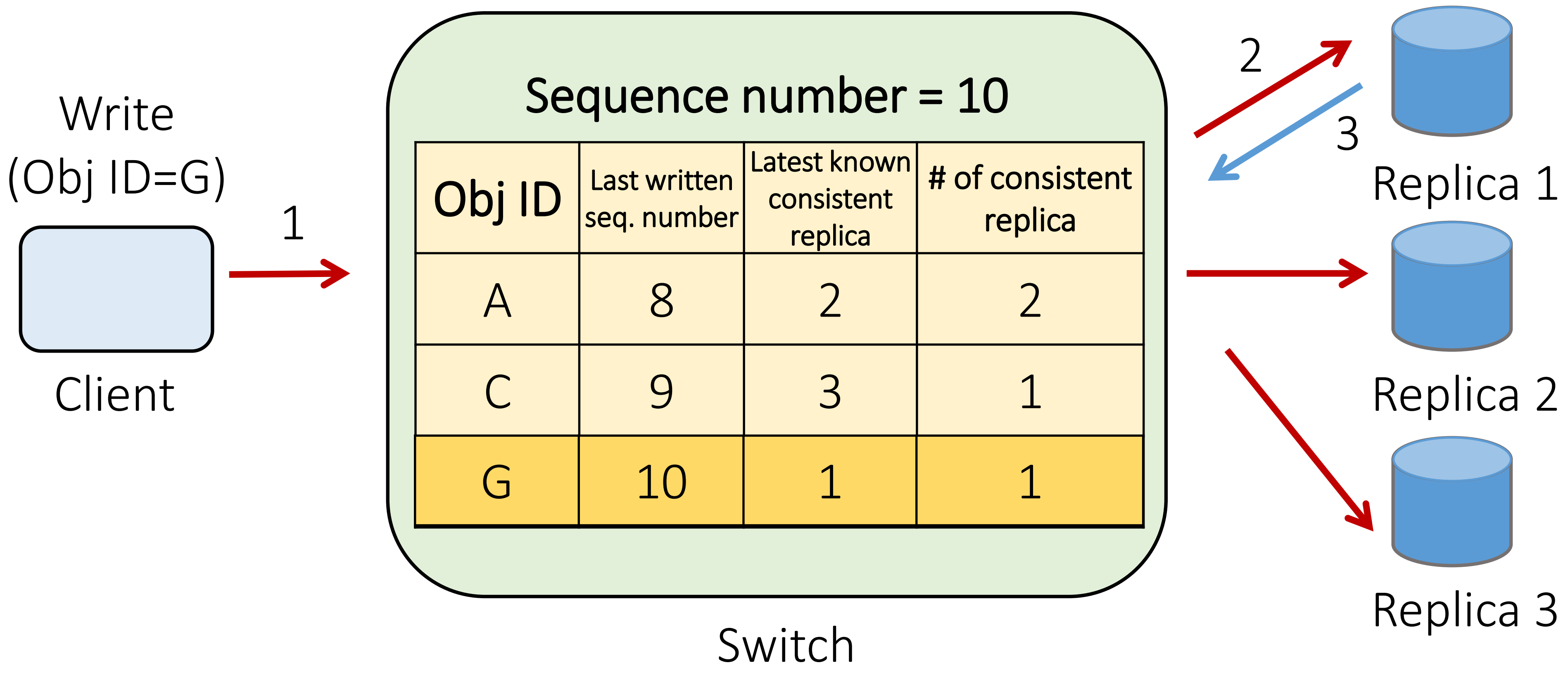
Obj ID	Last written seq. number	Latest known consistent replica	# of consistent replica
A	8	2	2
C	9	3	1

Write Processing



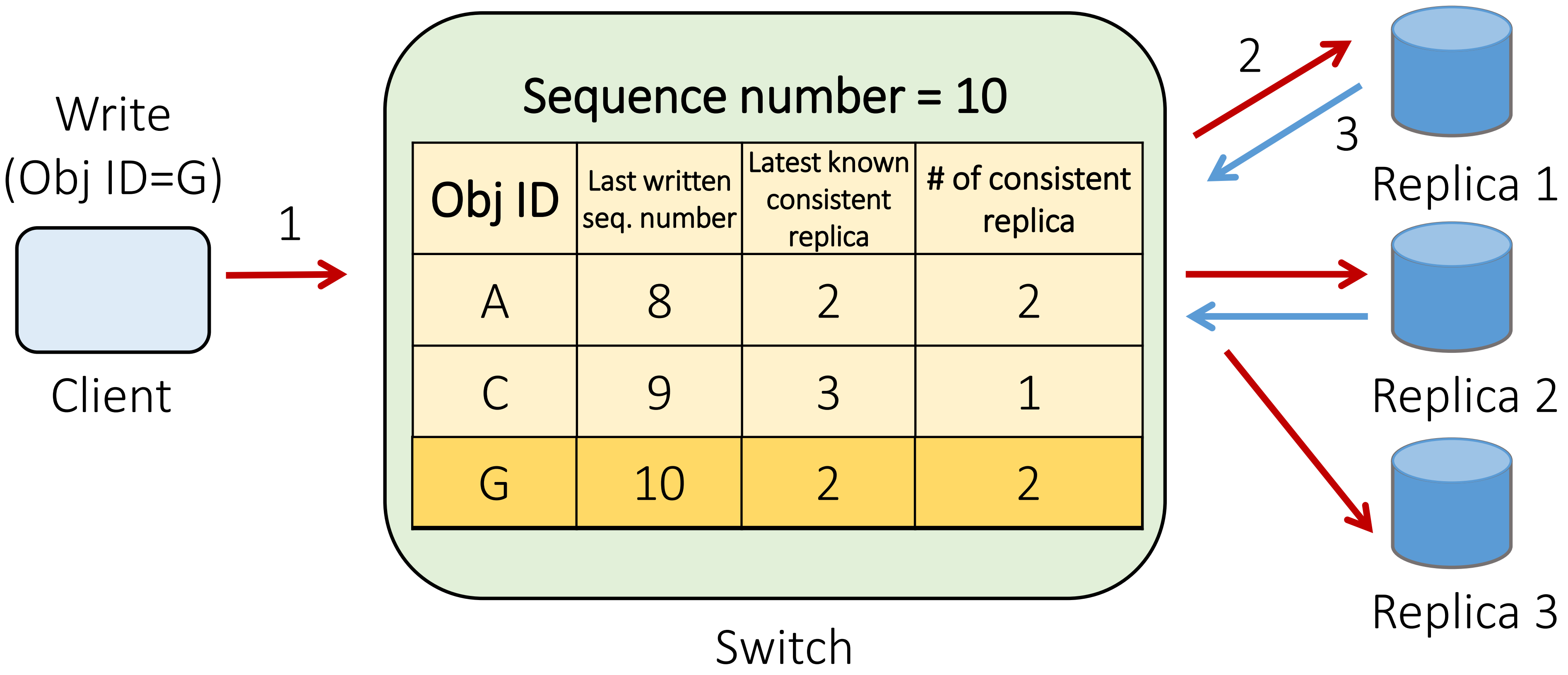
Obj ID	Last written seq. number	Latest known consistent replica	# of consistent replica
A	8	2	2
C	9	3	1
G	10		

Write Processing



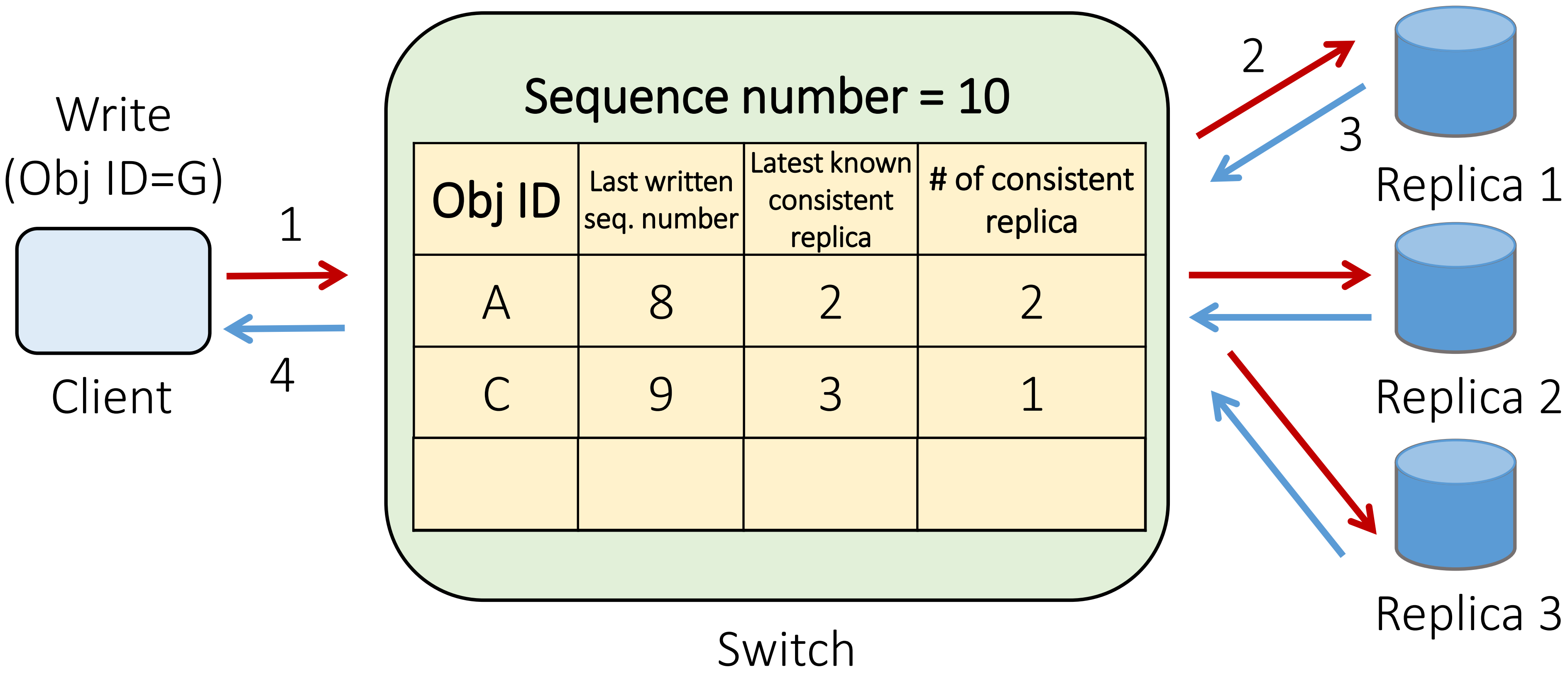
Obj ID	Last written seq. number	Latest known consistent replica	# of consistent replica
A	8	2	2
C	9	3	1
G	10	1	1

Write Processing



Obj ID	Last written seq. number	Latest known consistent replica	# of consistent replica
A	8	2	2
C	9	3	1
G	10	2	2

Write Processing



Obj ID	Last written seq. number	Latest known consistent replica	# of consistent replica
A	8	2	2
C	9	3	1

Data Plane Implementation

- 5 pipeline M-A stages and 5.68% of switch memory usage
- Multiple register arrays for inconsistent object list
  - Uses hash for indexing to minimize memory usage
  - Objects exist temporarily only during write coordination

Obj ID	Last written seq. num.	Latest known consistent replica	# of consistent replicas
A	2	1	3
C	6	0	1
	5	3	4

Insertion & Search

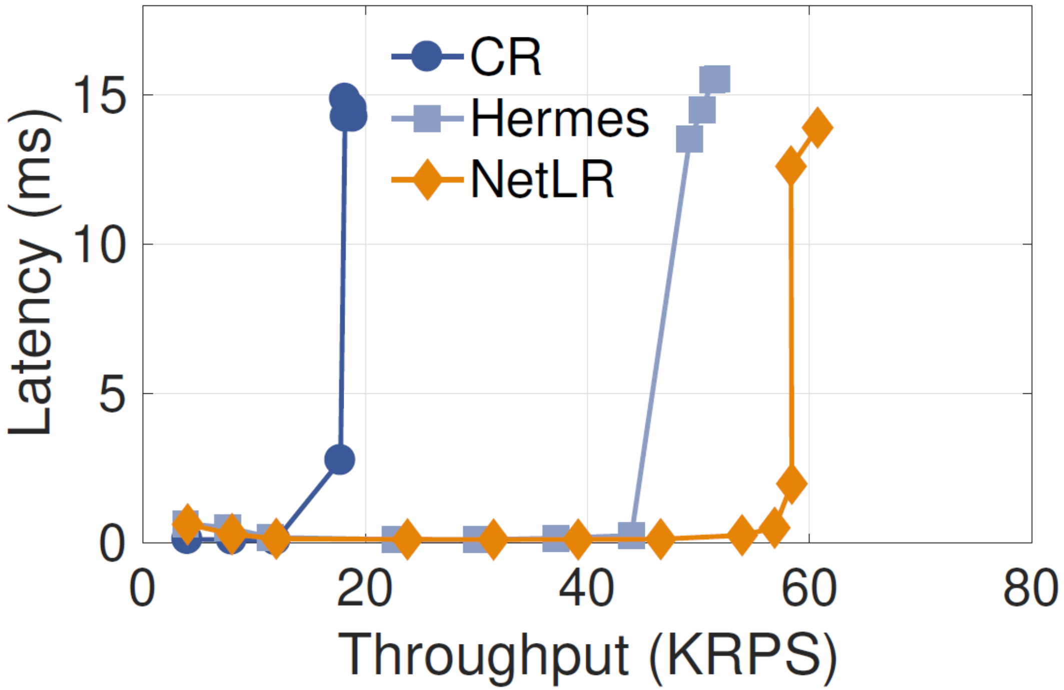
$h(\mathrm{C})$

$h(\mathrm{E})$ Deletion

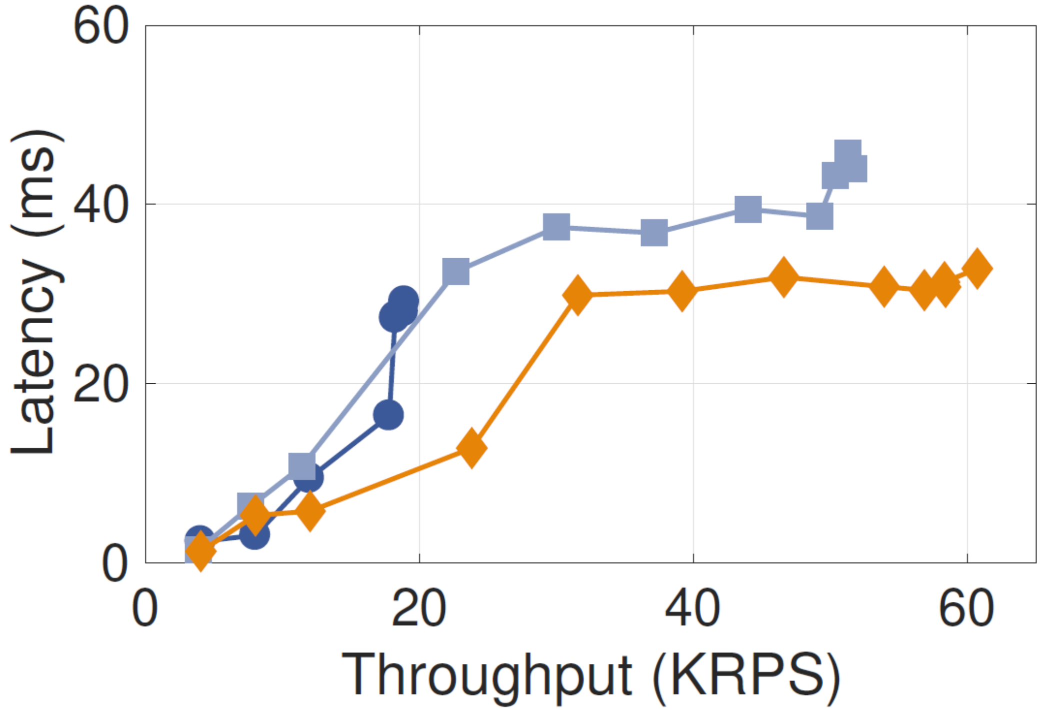
Evaluation

- Testbed setup
  - Edgecore Wedge100BF-32X switch with 3.2 Tbps Intel Tofino ASIC
  - 7 commodity servers with a 6-core CPU and 40GbE NIC
    - 6 of the servers are storage servers
    - One server acts as two clients with a dual-port NIC
- Comparison
  - CR@OSDI’04 (Represents leader-based protocol)
  - Hermes@ASPLOS’20 (Represents leaderless protocol)
  - Harmonia@VLDB’20 (in-network read-write conflict detection)
- Default workload
  - Two clients and four replicas
  - Read-heavy workload with 95:5 read:write ratio
  - 1M objects

Throughput vs. Latency



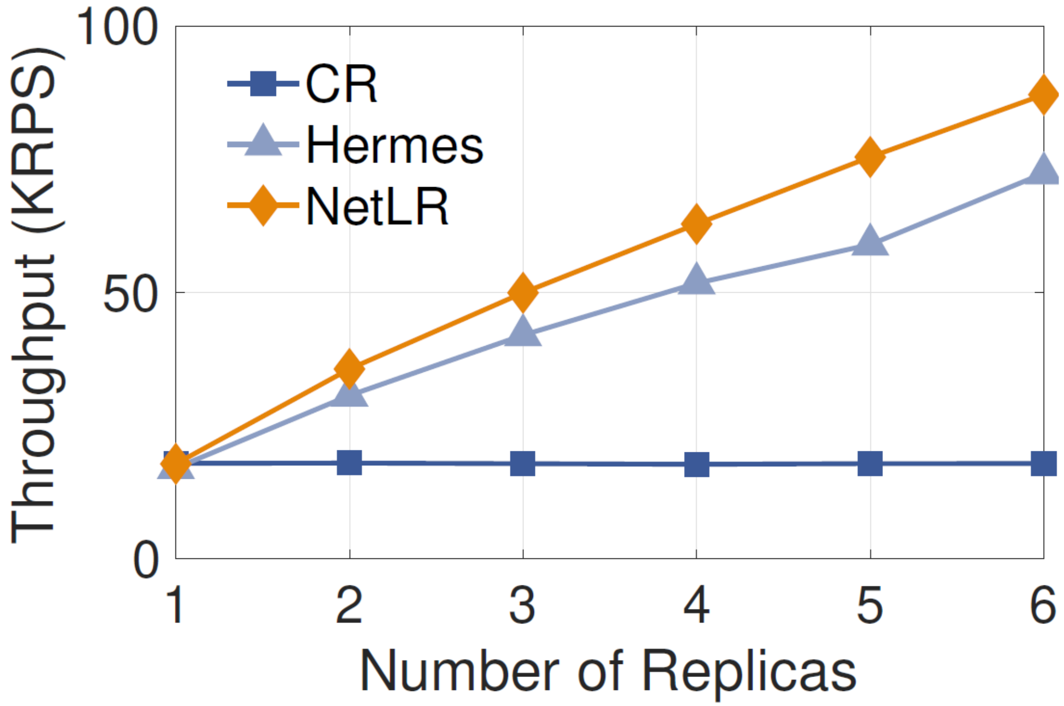
Median



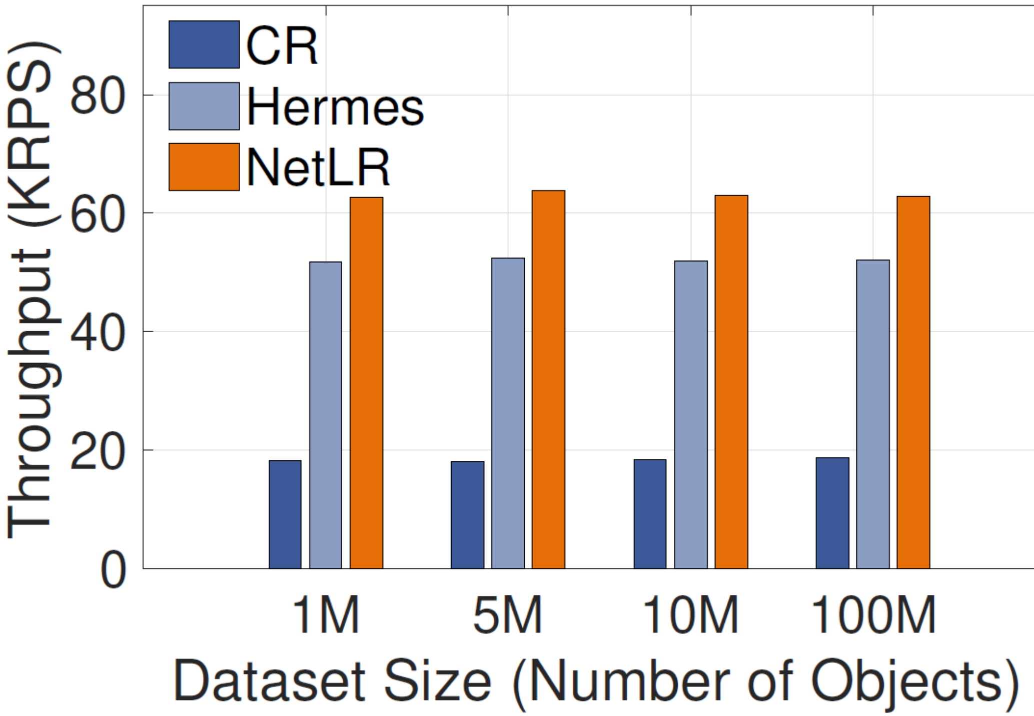
99th Percentile

NetLR improves throughput by up to 3.21x and 1.17x compared with CR and Hermes

Scalability



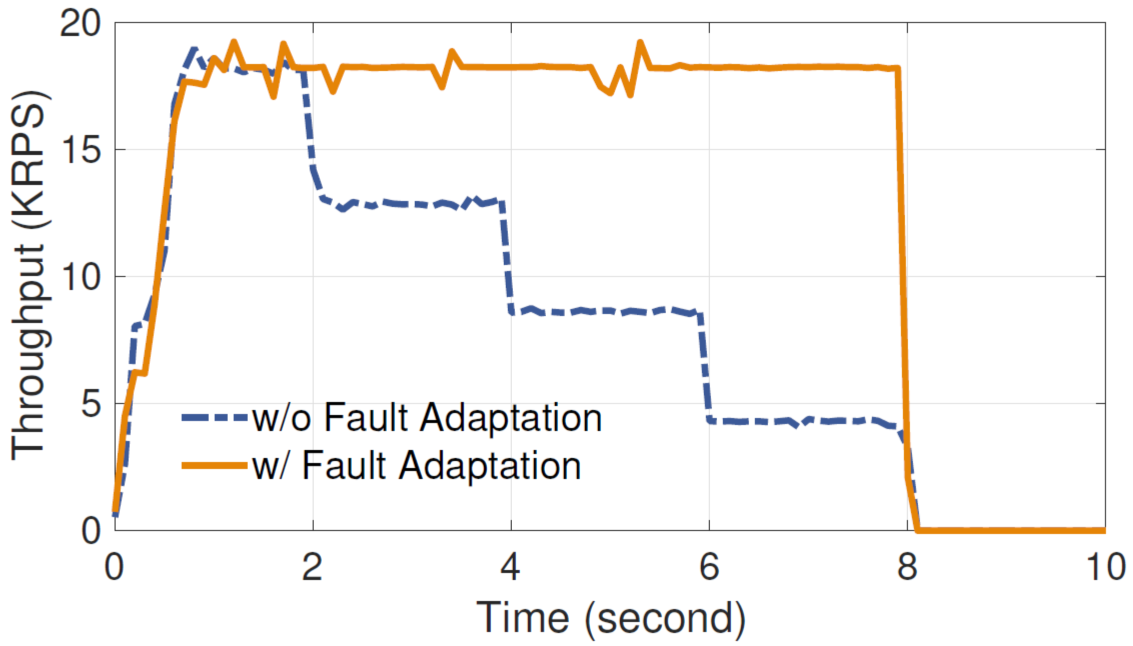
Impact of number of replicas



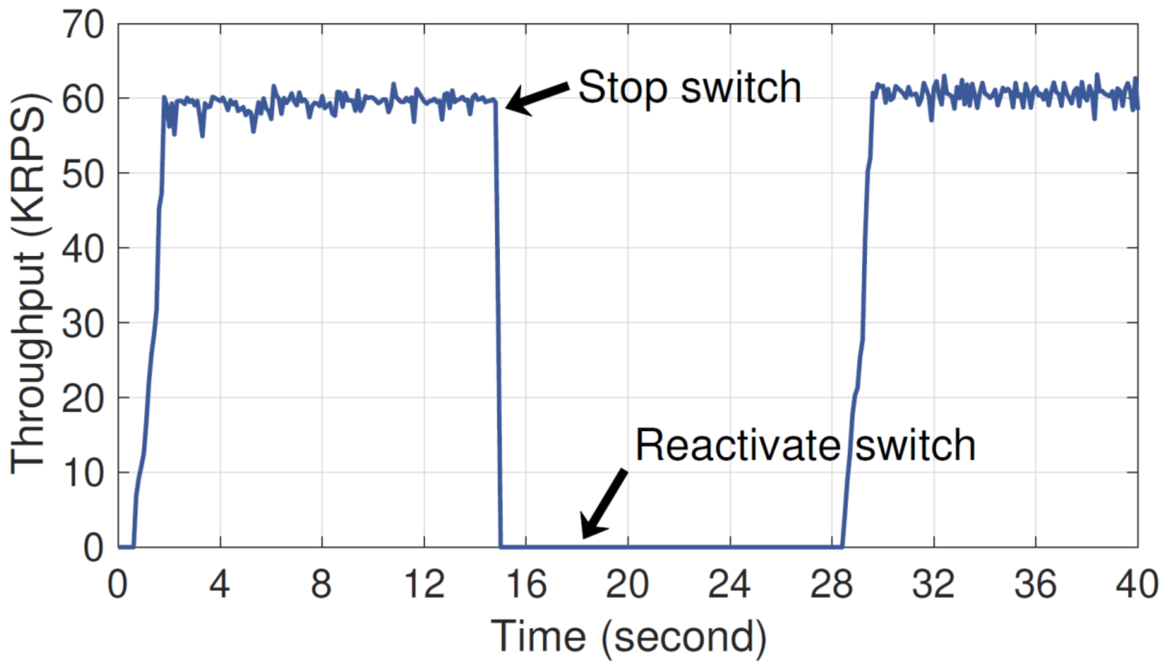
Impact of dataset size

NetLR provides near-linear scalability and is robust to the dataset size

Performance under Failures



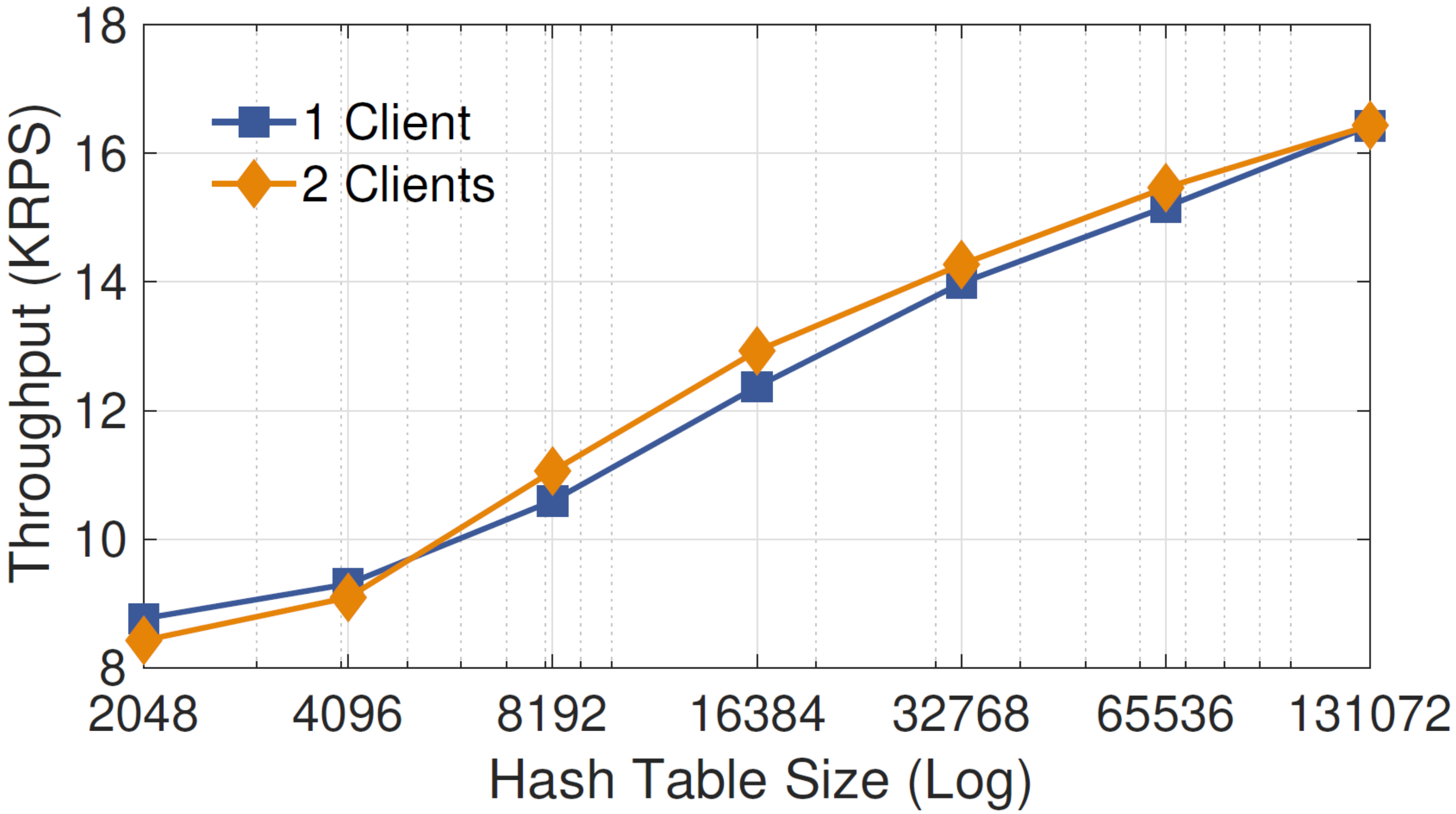
Server failures



Switch failures

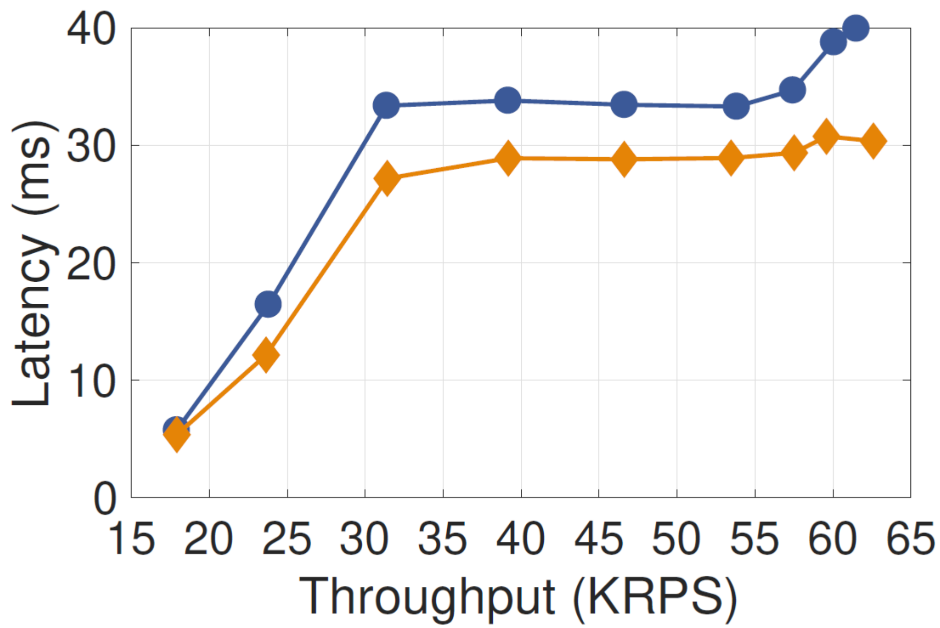
NetLR is robust to server and switch failures

Impact of switch memory size

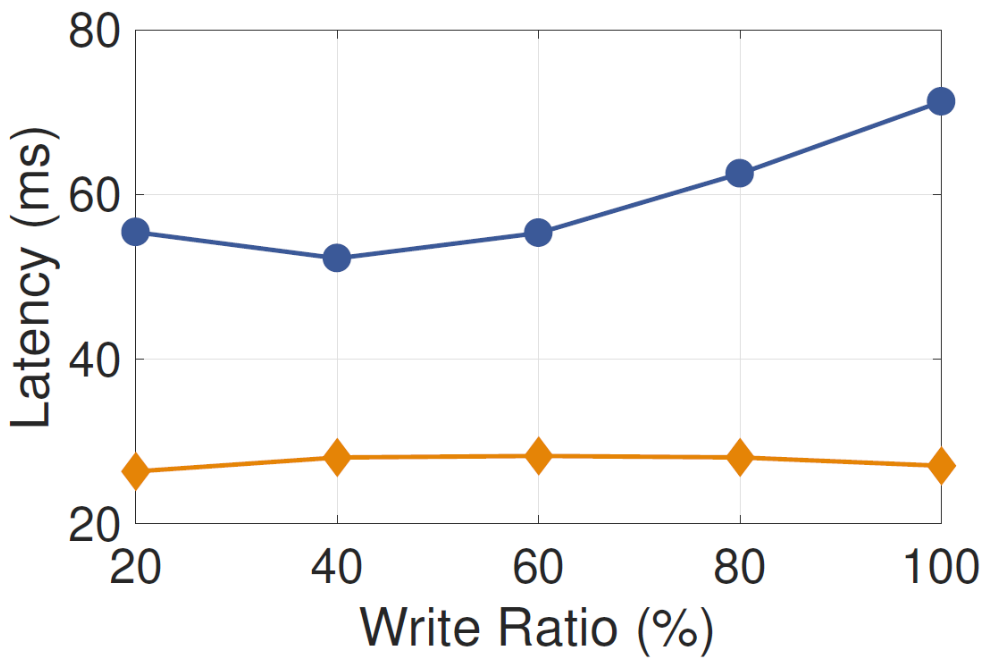


NetLR requires only 128K hash slots (≈5.68% of switch memory)
to achieve maximum write throughput

Comparison to Harmonia



Throughput vs. 99th percentile latency



Impact of write ratios on 99th percentile latency

NetLR has better tail latency than Harmonia by 2.03x on average

Conclusion

- NetLR is a new replicated data store architecture
  - High throughput and low latency
  - Strong consistency
  - Fault tolerance
- In-network leaderless replication
  - Leverages the flexibility and capability of programmable switches
- Emerging programmable switches have great potential to accelerate data stores

Thank you!