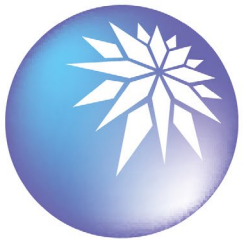
Network-Accelerated Multiget Coordination for Distributed Key-Value Stores

Jiyoon Bang and Gyuyeong Kim

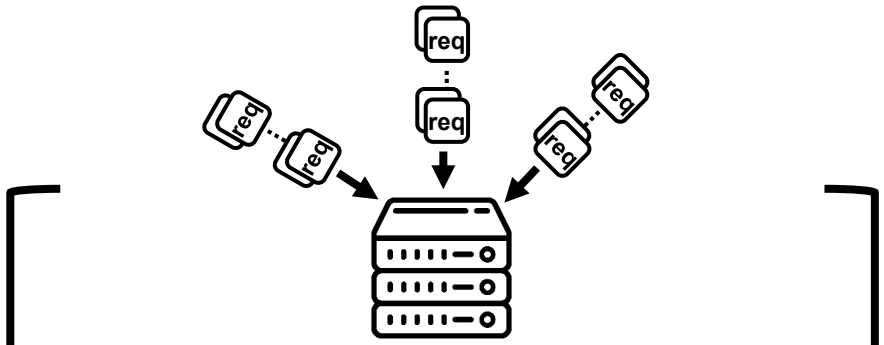
IEEE CCGrid 2025

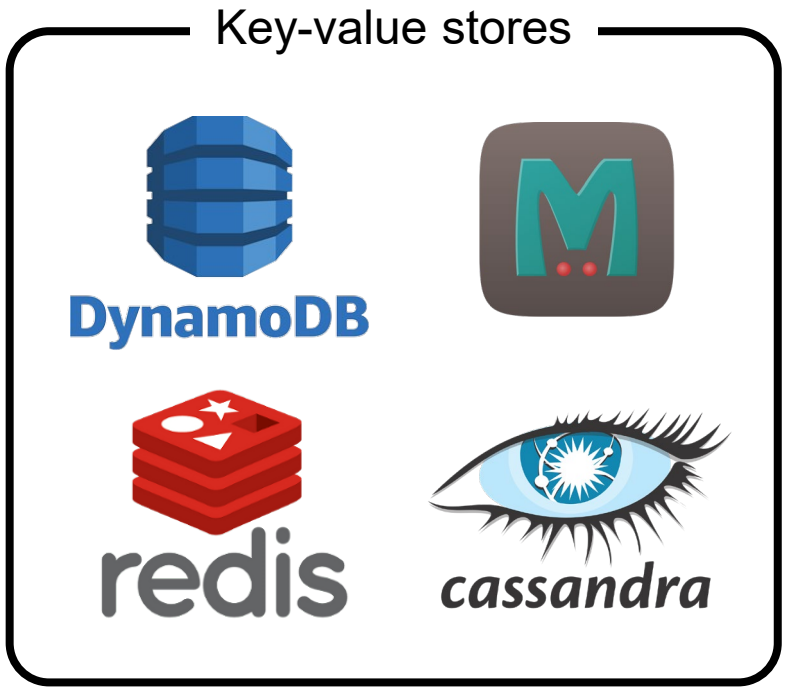


성신여자대학교
SUNGSHIN WOMEN'S UNIVERSITY

Key-value stores

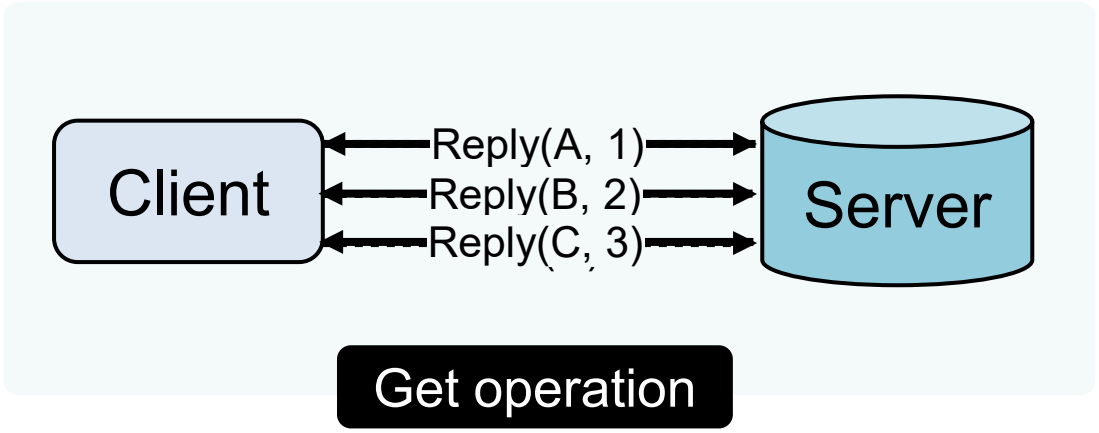
- Key-value stores have been fundamental building blocks for modern online services
  - To access key-value pairs quickly

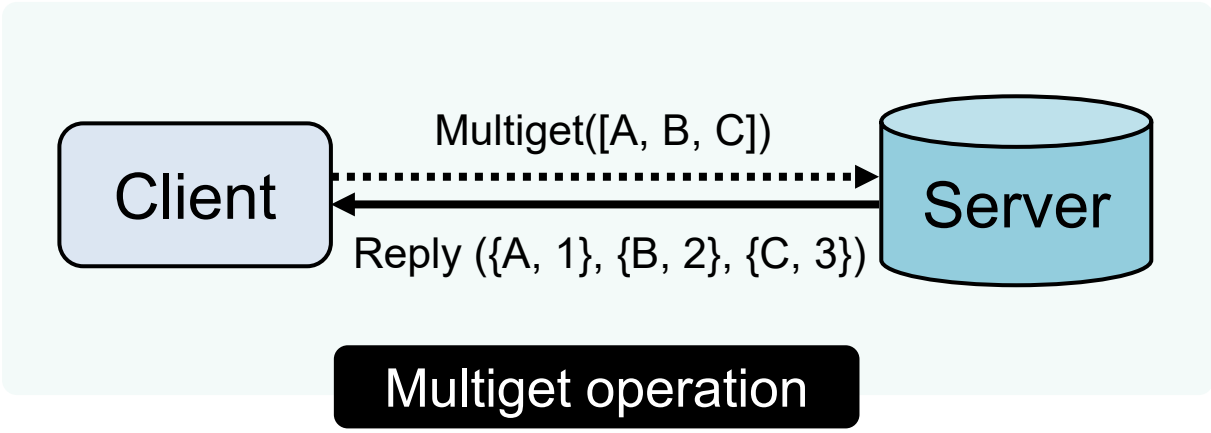




Multiget operations in key-value stores

- Many key-value stores support a **multiget operation**
  - Batches get operations for multiple keys to get the values within a single request
  - Meta leverages the multiget to improve the performance of Memcached*



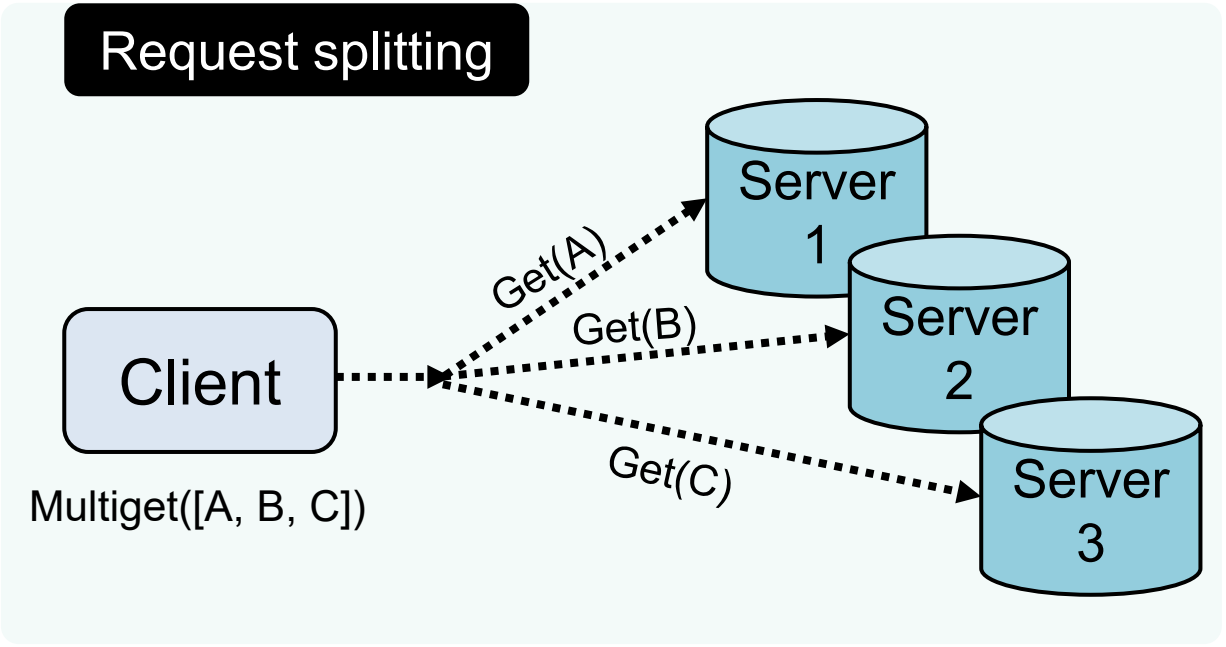


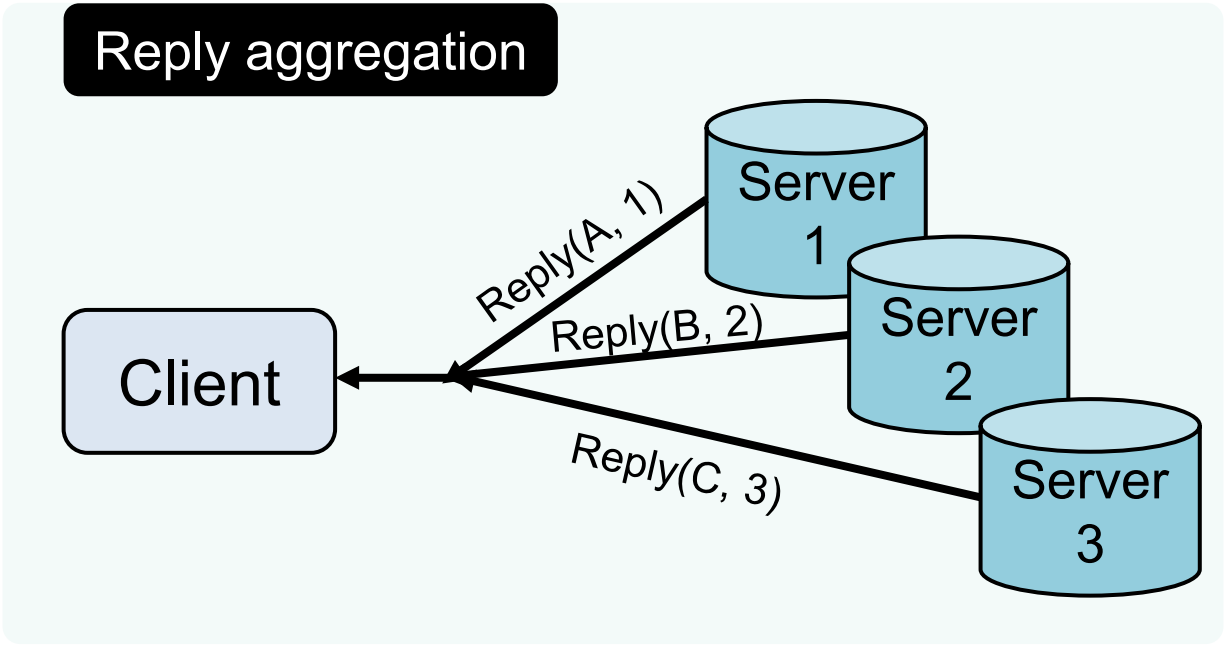
Can **reduce client-side overhead** and **latency**

*Rajesh Nishtala et al., “Scaling Memcache at Facebook,” in *Proc. of USENIX NSDI,* 2013.

Multiget coordination overhead

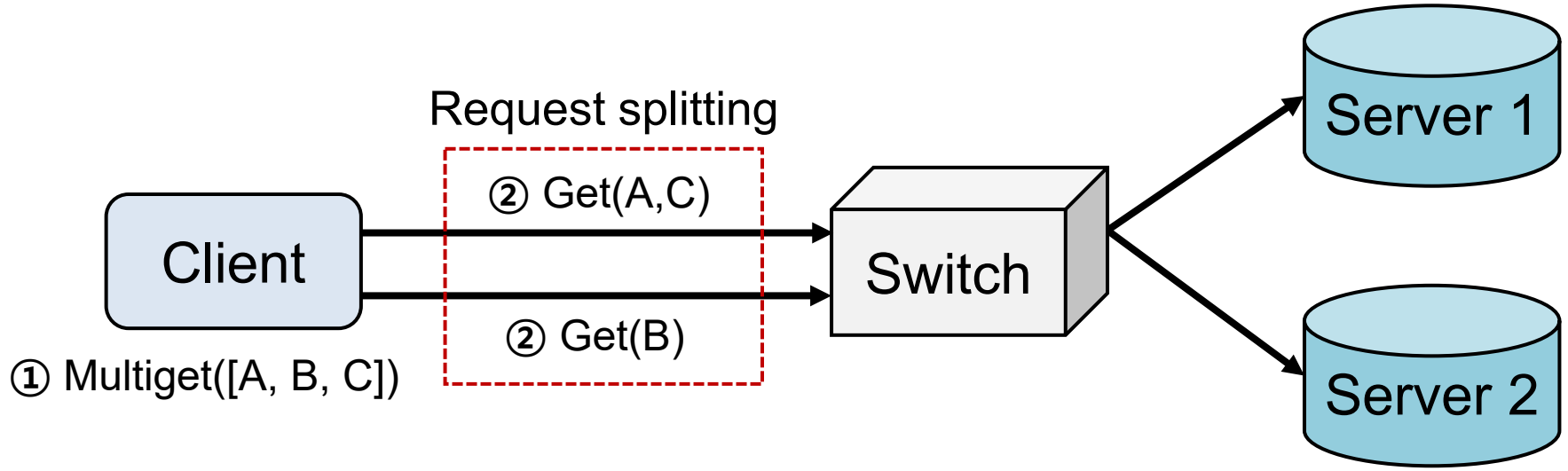
- The requested keys are distributed over multiple storage servers
- The multiget operation requires extra coordination
  - Split a single request into multiple sub-requests
  - Aggregate sub-replies into a single reply





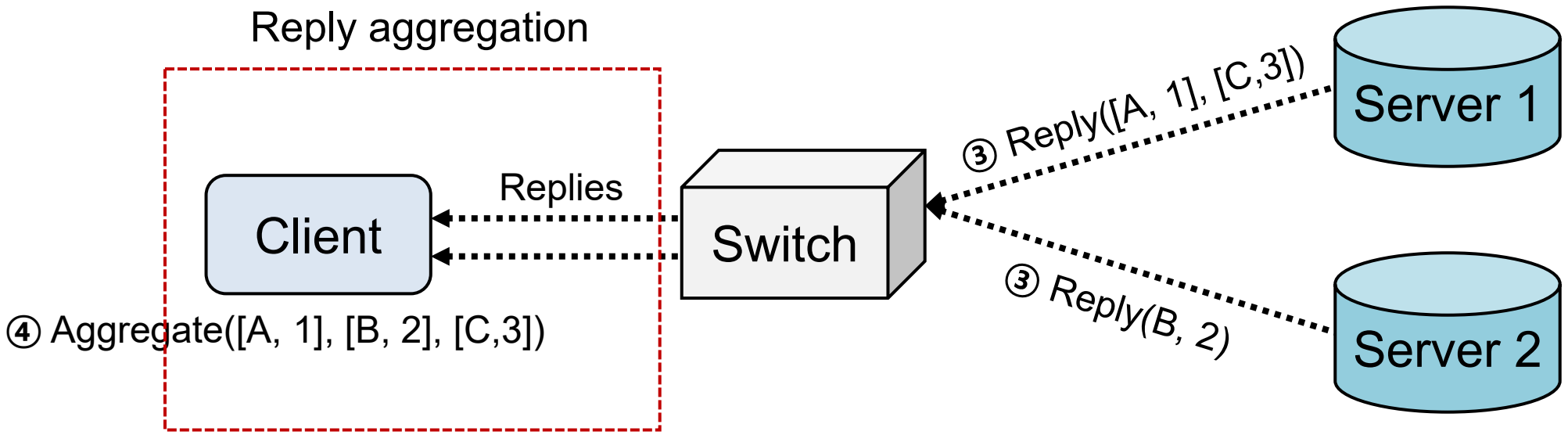
Existing architectures: Client-based

- The client application is responsible for both request splitting and reply aggregation
  - e.g., Redis, Memcached, and RocksDB



Existing architectures: Client-based

- The client application is responsible for both request splitting and reply aggregation
  - e.g., Redis, Memcached, and RocksDB

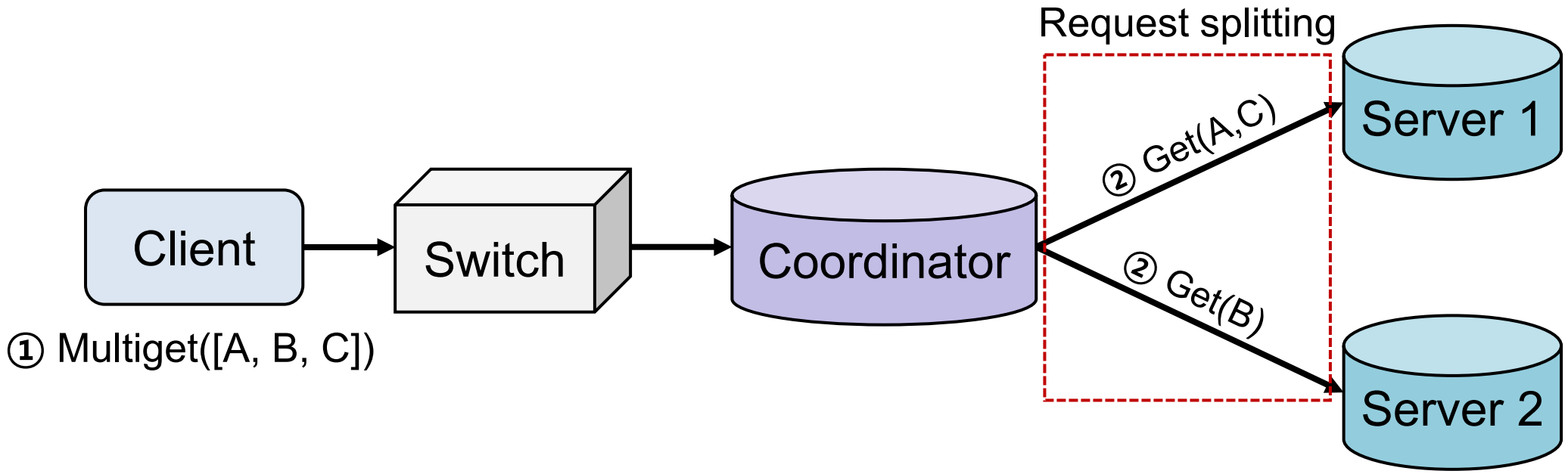


👍 High throughput

👎 Low latency

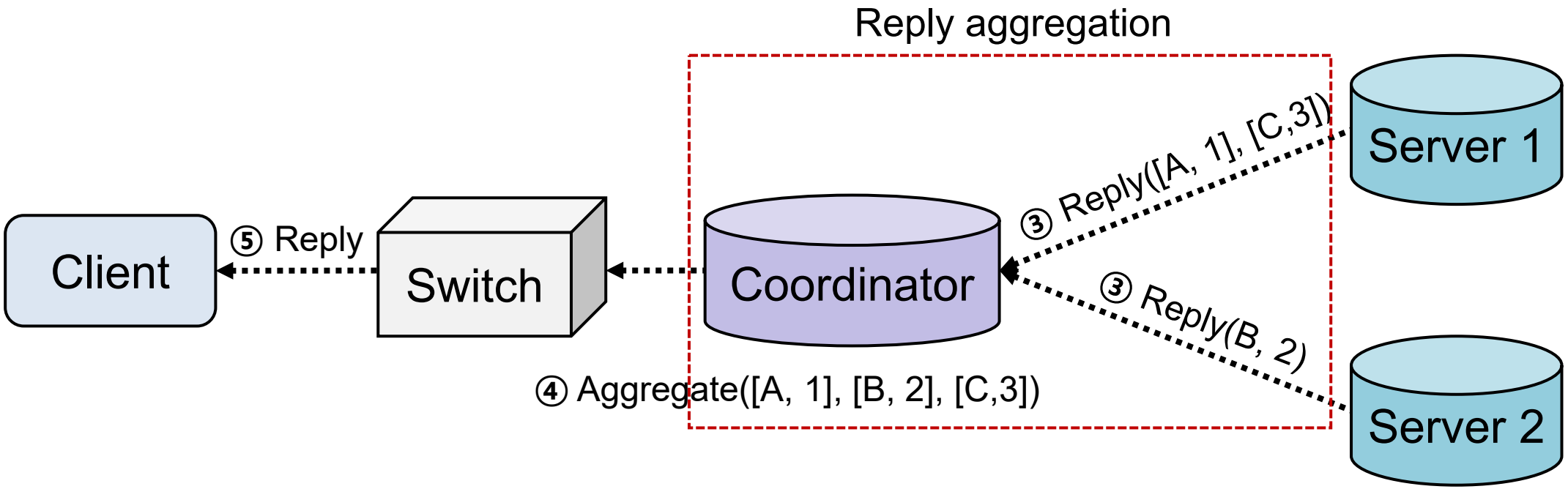
Existing architectures: Coordinator-based

- A dedicated coordinator node coordinate multiget operations
  - e.g., Cassandra, DynamoDB, and MongoDB



Existing architectures: Coordinator-based

- A dedicated coordinator node coordinate multiget operations
  - e.g., Cassandra, DynamoDB, and MongoDB



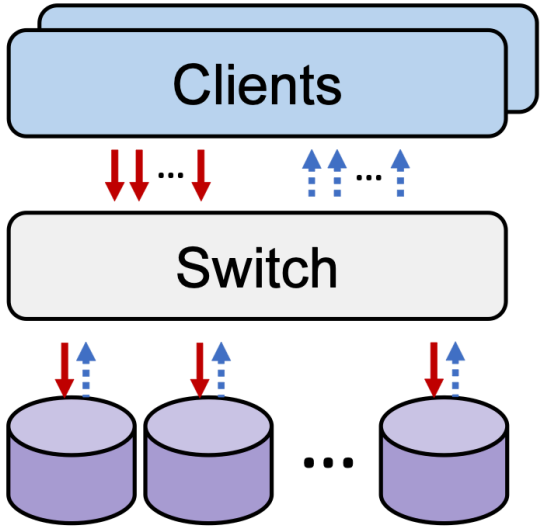
👍 Low client overhead

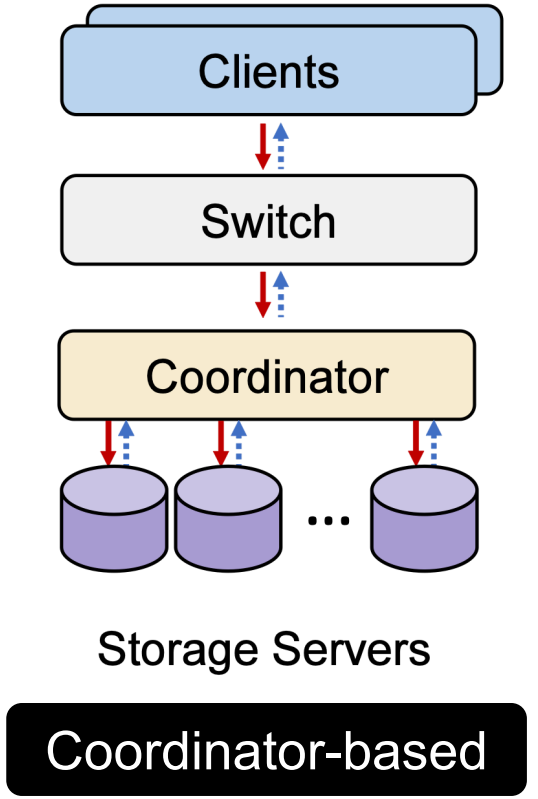
👎 Scalable

👎 High throughput

Existing architectures

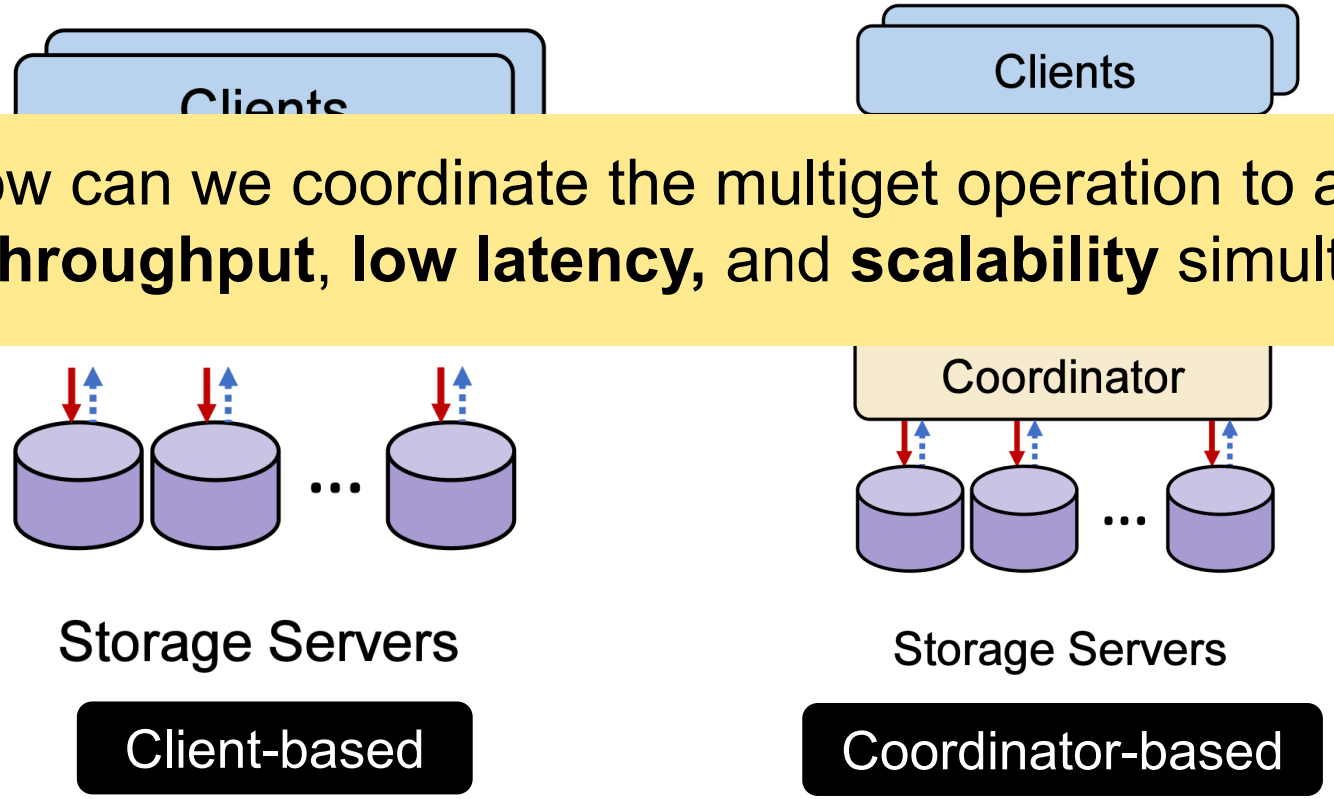
- Existing architectures cannot achieve **high throughput**, **low latency**, and **scalability** at the same time





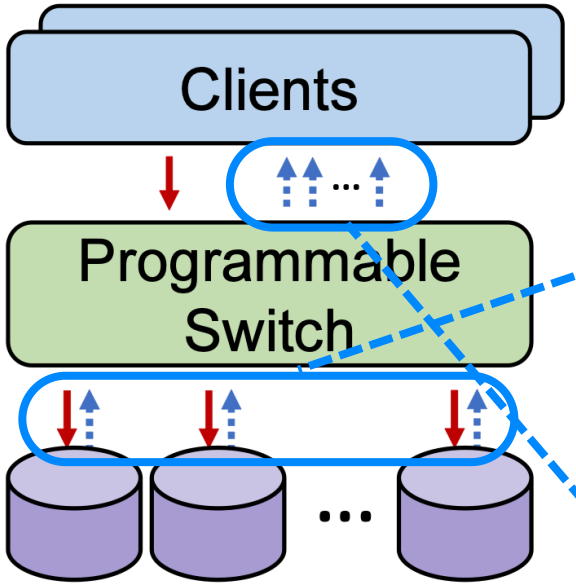
Existing architectures

- Existing architectures cannot achieve **high throughput**, **low latency**, and **scalability** at the same time



NetMC: Network-accelerated multiget coordination

- Dividing the work of multiget coordination between the **network switch** and the **client**
  - Each coordination function is **best performed at its vantage point**



Request splitting on programmable switch

- Well-suited for stateless and I/O-intensive request splitting
- Can process billions of packets per second
- Can customize the packet processing logic in the switch data plane

Reply aggregation on clients

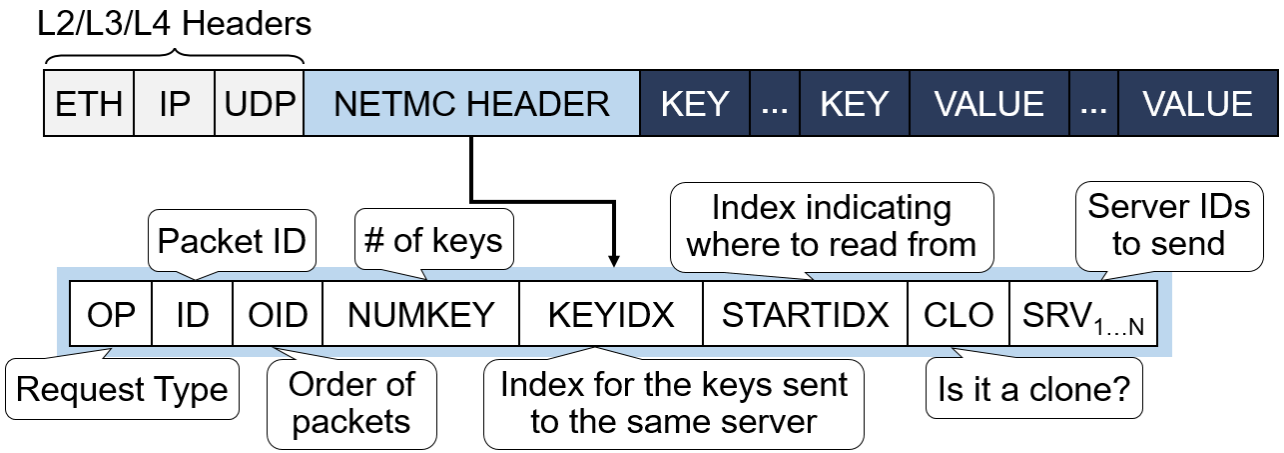
- Suitable for stateful operations and complex logic for variable-length data

Technical challenges

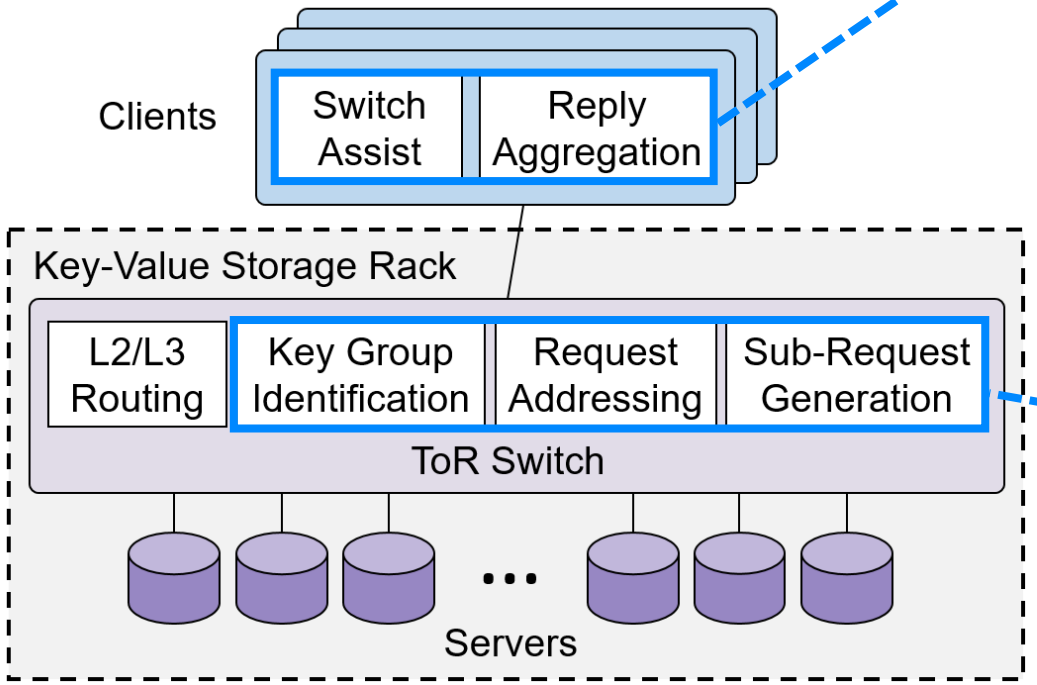
- How to realize switch-based request splitting?
  - Limited memory access
  - Limited computational capability
    - Not support loops and modulo operations for a random number

➡ Simplifying the switch mechanism as much as possible

- Through the **client-side assistance** that puts metadata as hints into the custom packet header



NetMC architecture



Clients

Switch assist
- Put extra metadata into the packet header

Reply aggregation
- Aggregate sub-replies from storage servers

Switch data plane modules

Key group identification
- Identify the current key group
- Check if there are more key groups to process

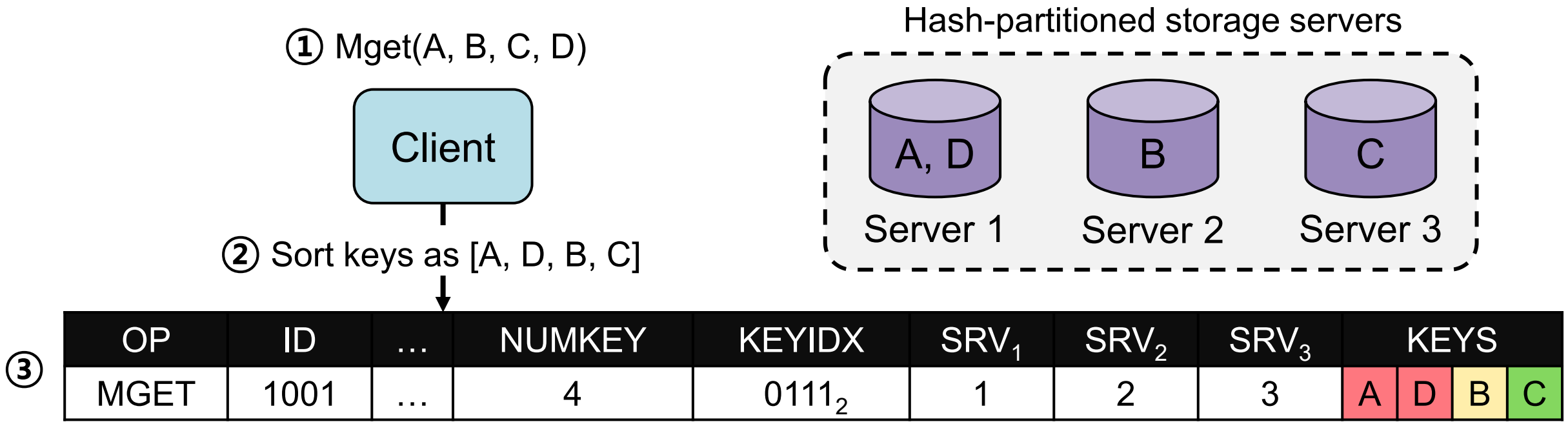
Packet addressing
- Assign the target storage server's IP for the current key group

Sub-request generation
- Generate sub-requests

Key group: a subset of keys belonging to the same target server

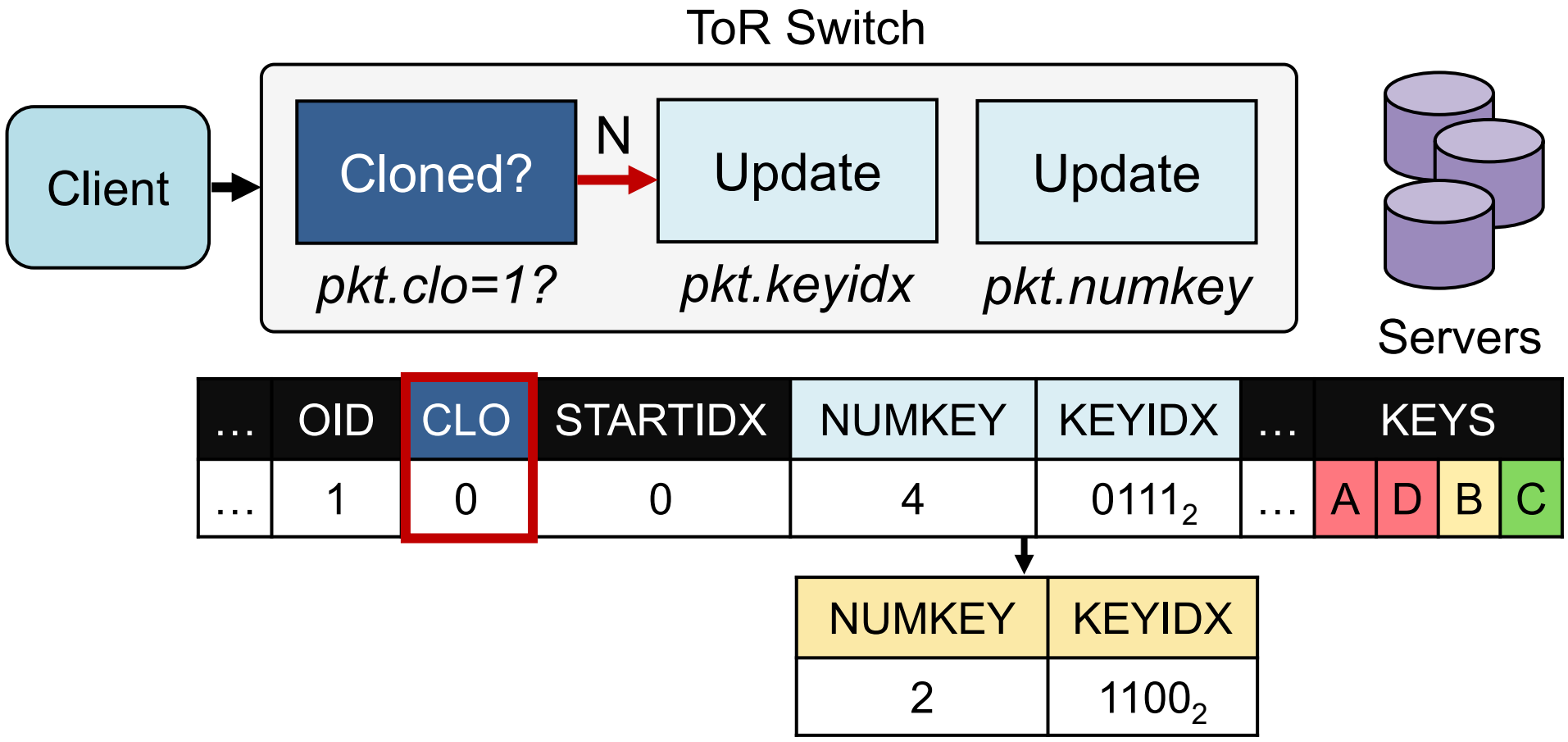
Multiget request generation at clients

- The client generates a multiget request
  - Retrieves the values of keys {A,B,C,D} that are distributed across 3 storage servers



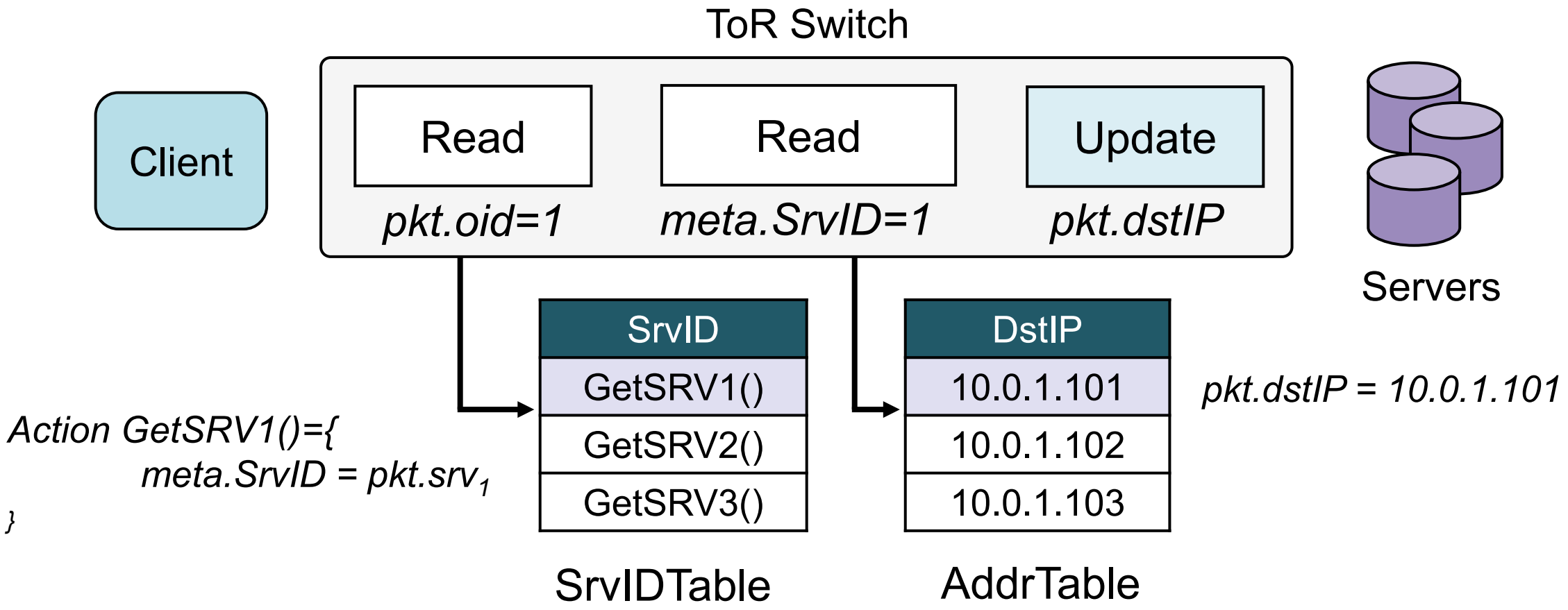
Request splitting at switch data plane

- Step 1: check if it is the cloned packet
- Step 2: identify the current key group and update the metadata



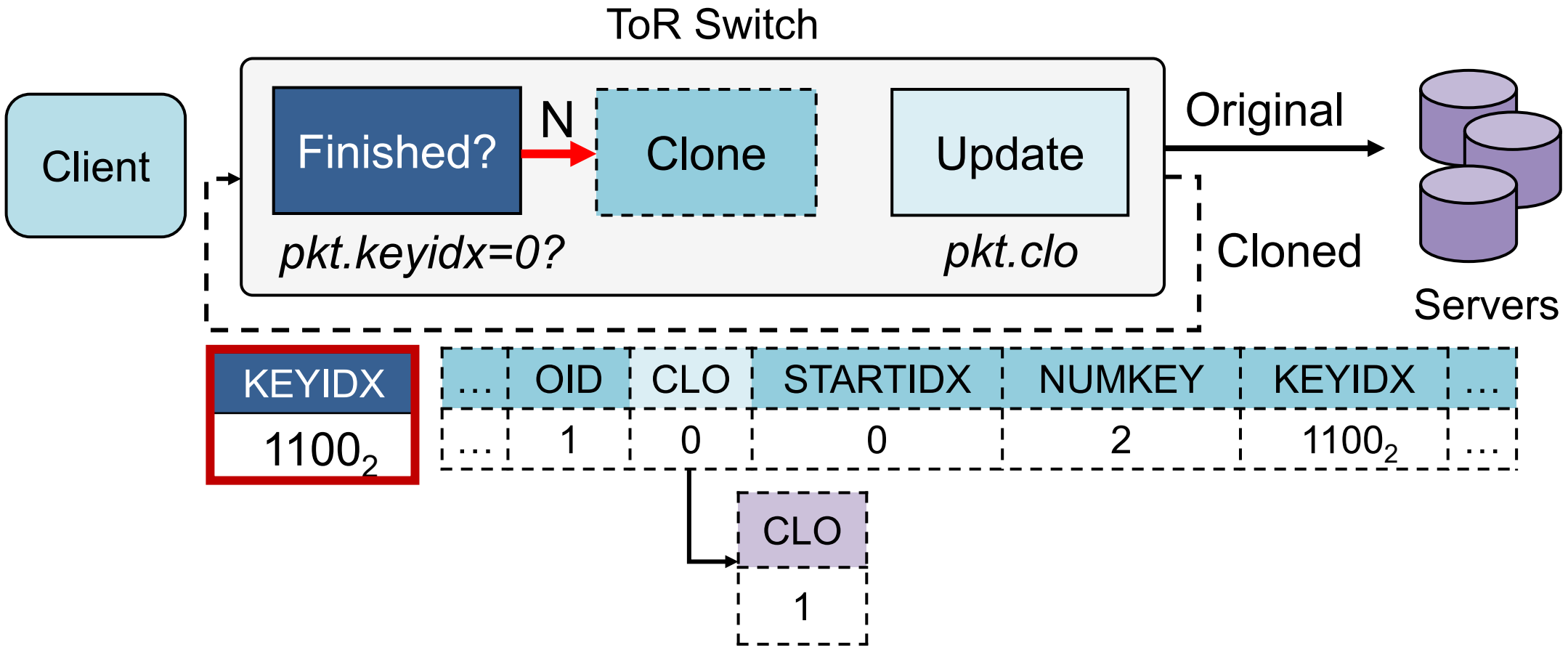
Request splitting at switch data plane

- Step 3: assign the target server's IP for the current key group



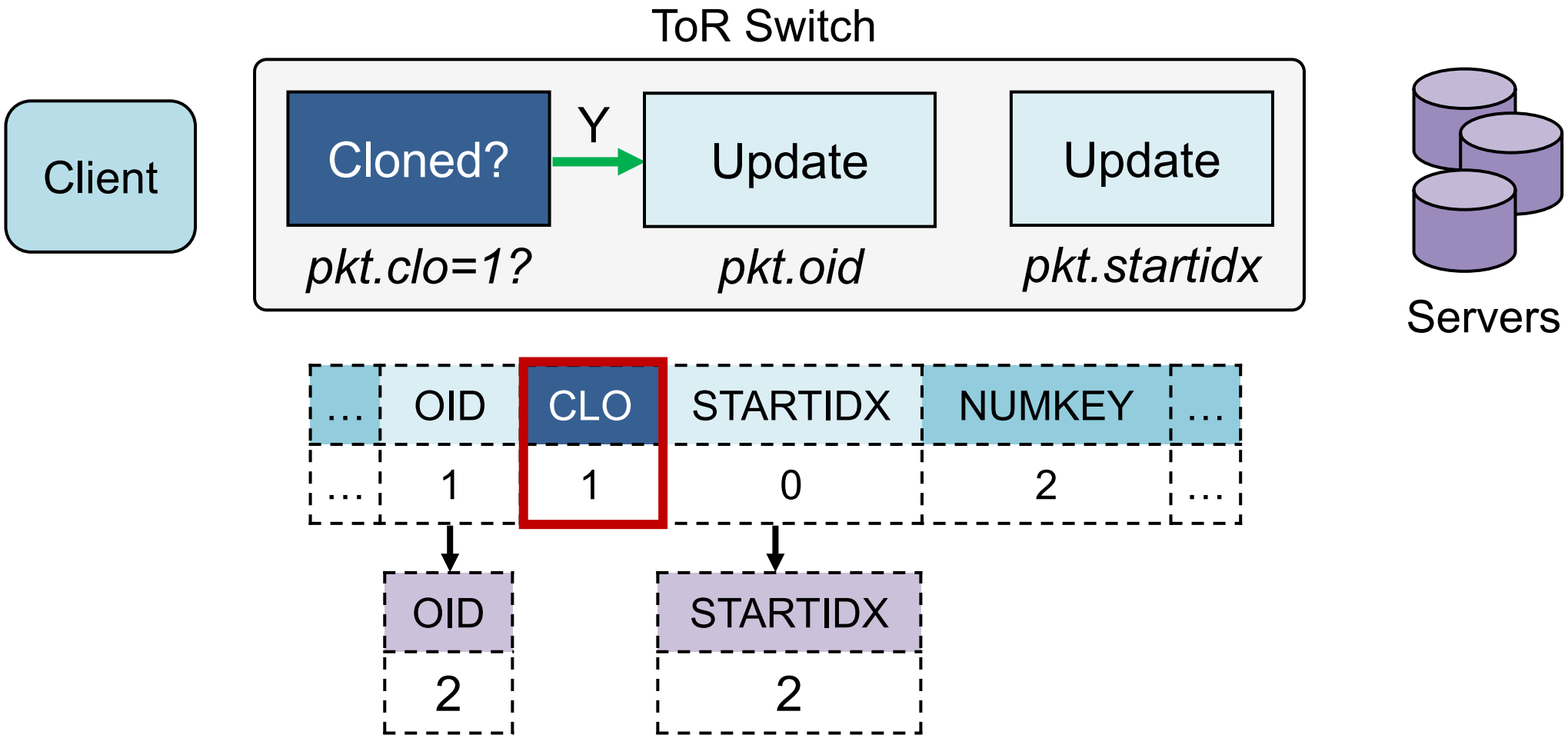
Request splitting at switch data plane

- Step 4: check if there are more key groups to process
- Step 5: generate a sub-request



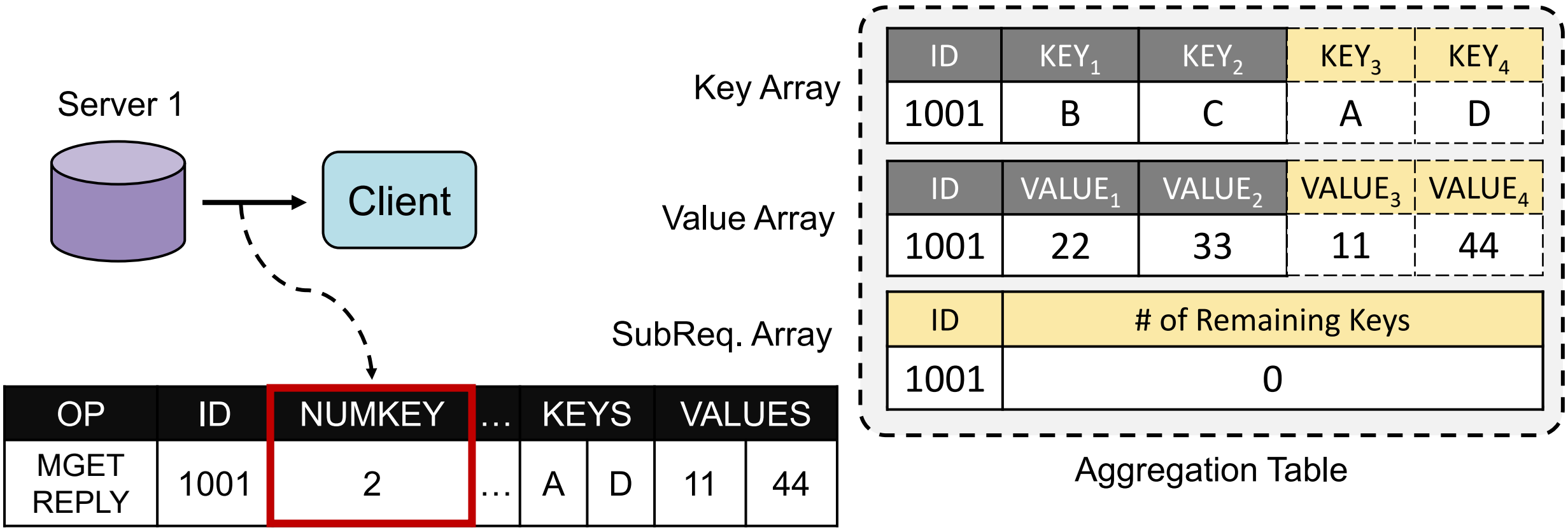
Request splitting at switch data plane

- Step 6: update the metadata of the cloned packet for the next processing



Reply aggregation at clients

- The client appends the contained key-value pairs to the aggregation table
  - The reply is committed when the number of remaining keys becomes zero



Evaluation

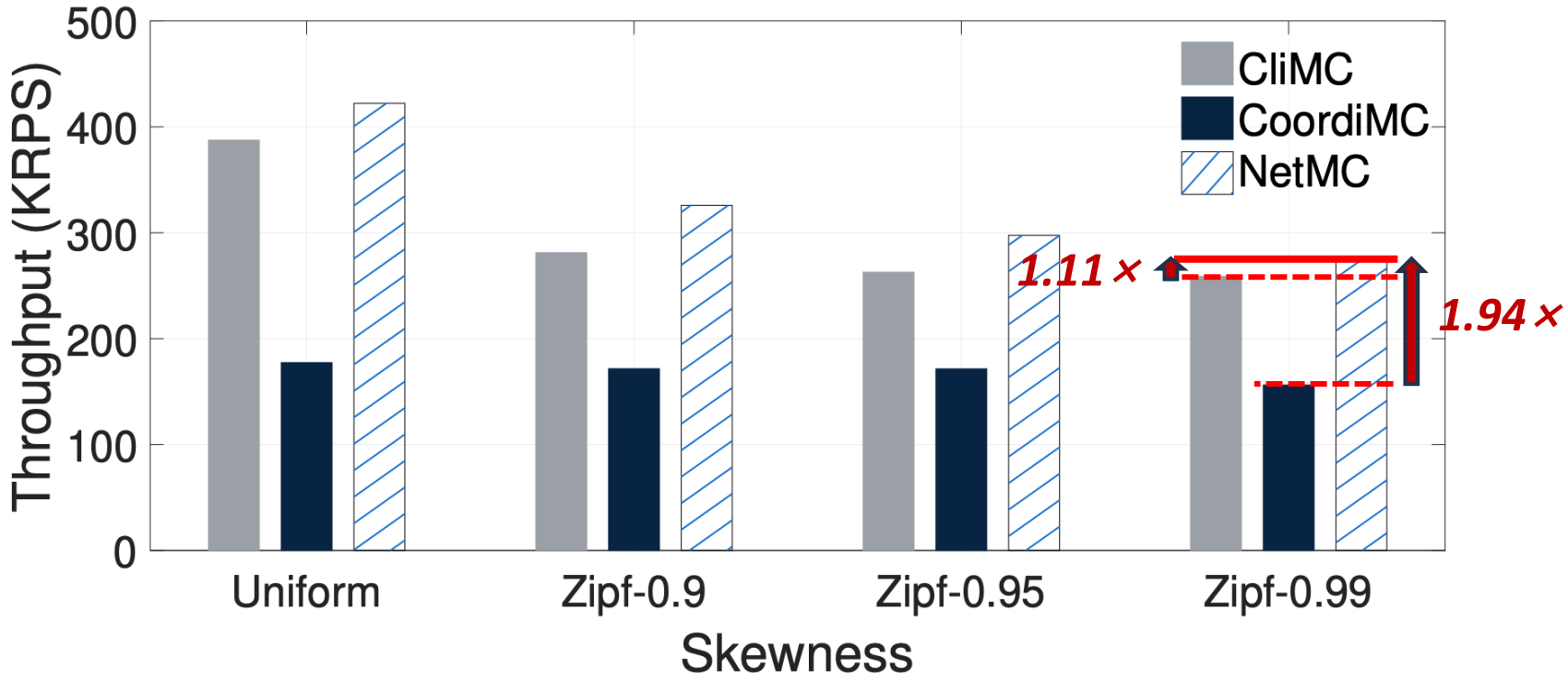
- Testbed
  - 6.5Tbps Intel Tofino switch
  - 8 servers with NVIDIA ConnectX-5 100G NIC

- Baseline
  - CliMC (Client-based architecture)
  - CoordiMC (Coordinator-based architecture)

- Workloads
  - 5% write ratio
  - Production workload distribution* with zipf-0.99

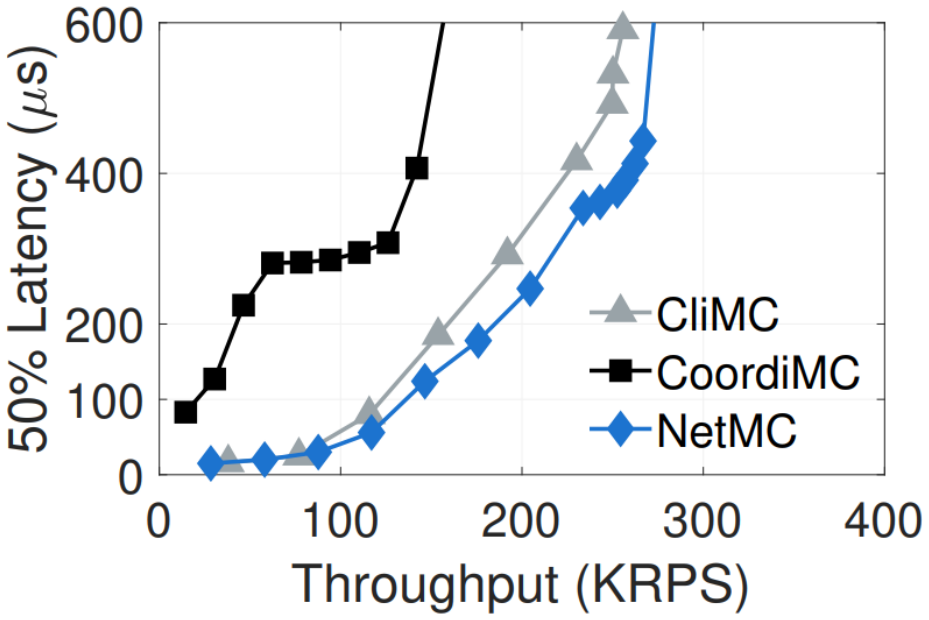
*Waleed Reda, Marco Canini, Lalith Suresh, Dejan Kostić, and Sean Braithwaite, “Rein: Taming Tail Latency in Key-Value Stores via Multiget Scheduling,” in *Proc. of ACM EuroSys*, 2017.

Throughput vs. skewness

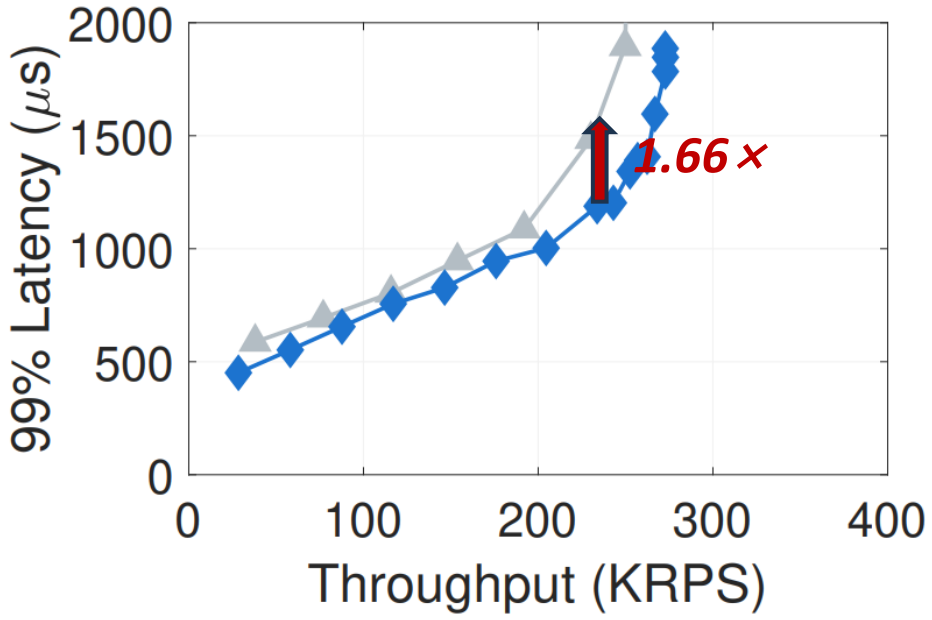


NetMC is resilient to skewness in key distributions

Latency vs. throughput



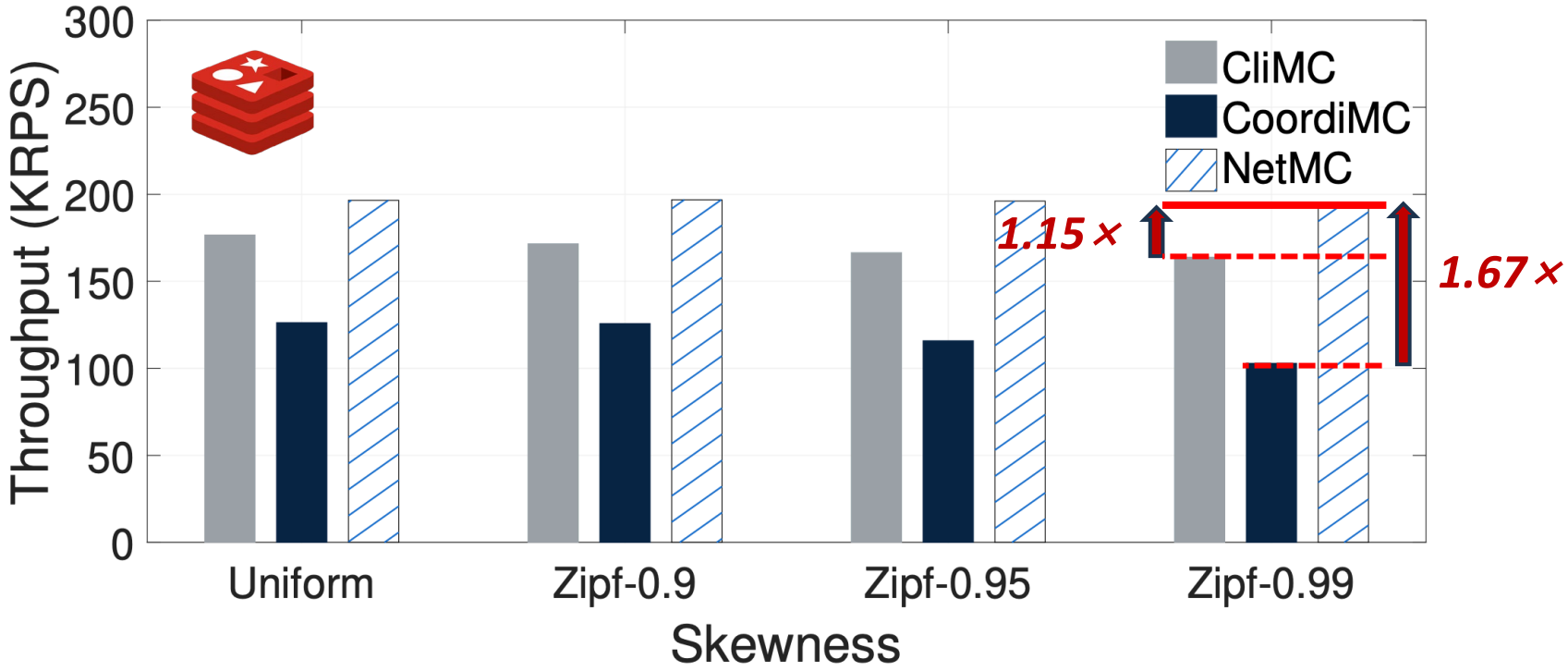
Median latency



99th percentile latency

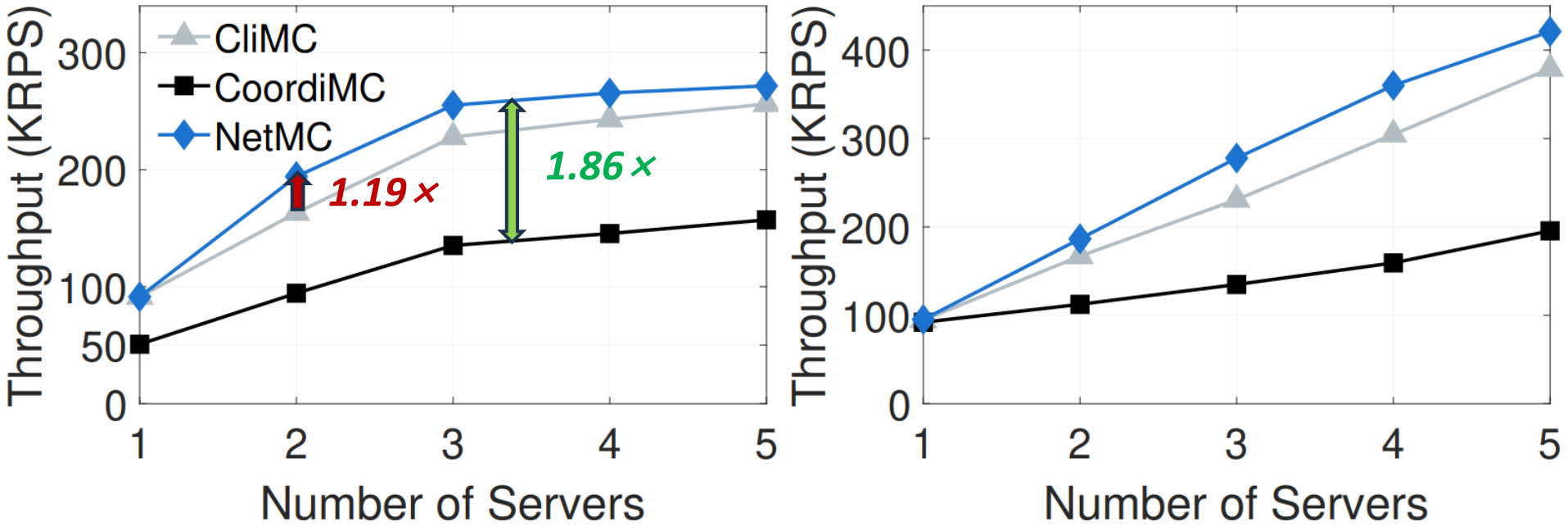
NetMC consistently achieves the lowest latency across most throughput levels

Application: Redis



NetMC achieves high throughput and low latency when integrated with Redis

Scalability

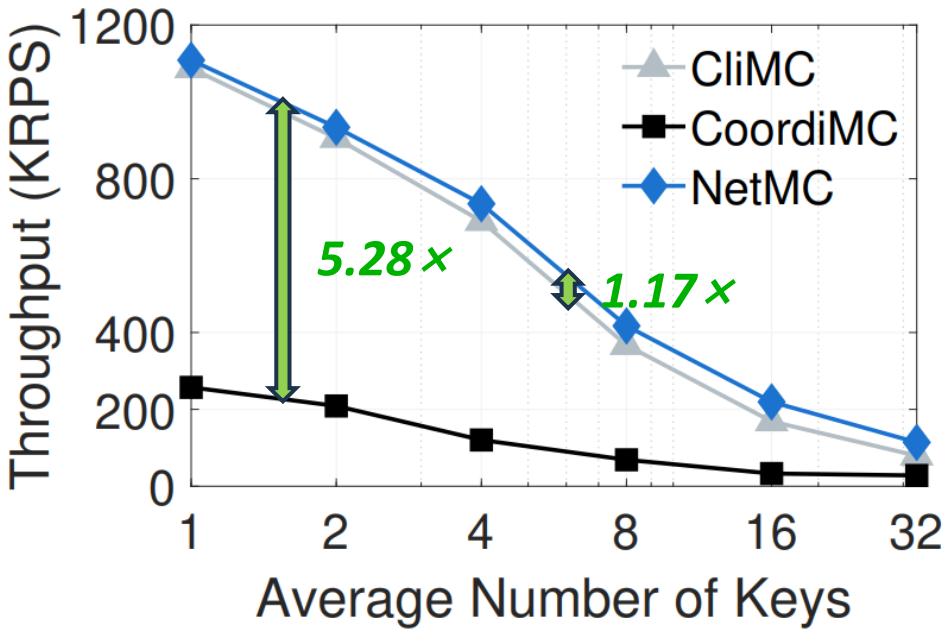


Skewed workload

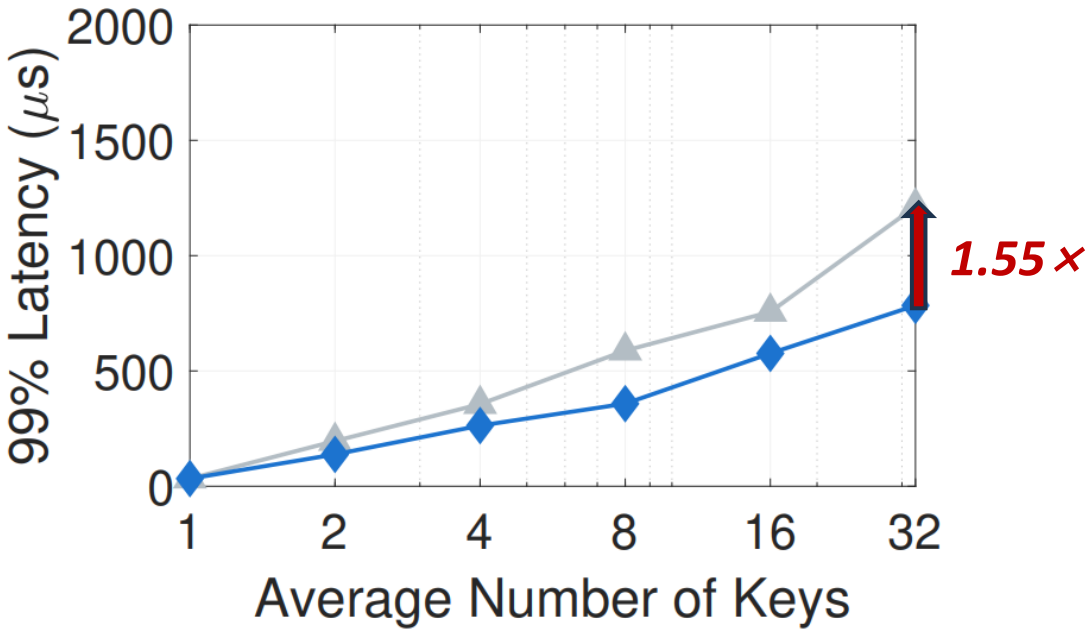
Uniform workload

NetMC provides the best throughput regardless of the number of servers

Impact of multiget size



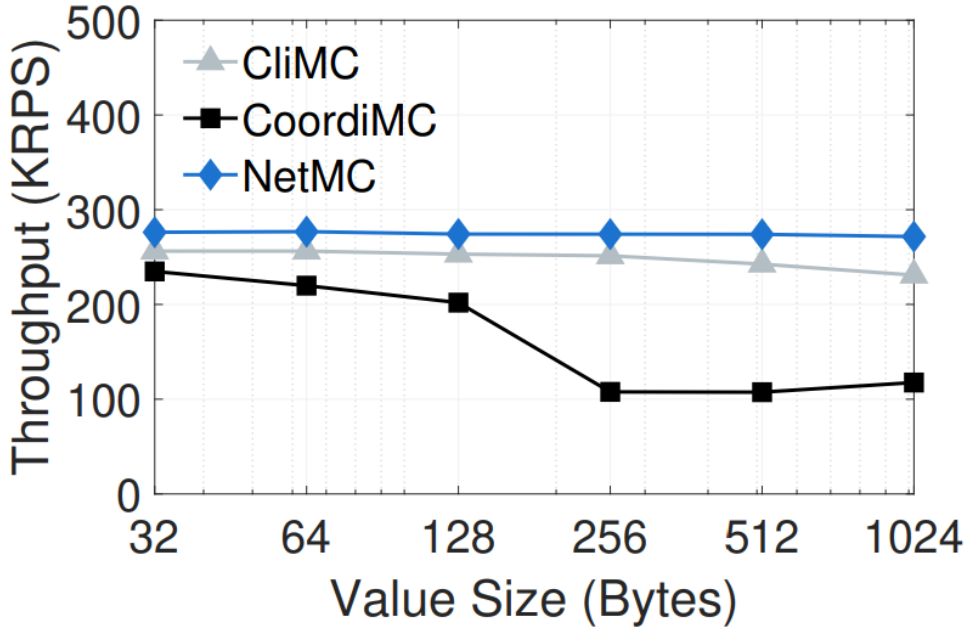
Throughput



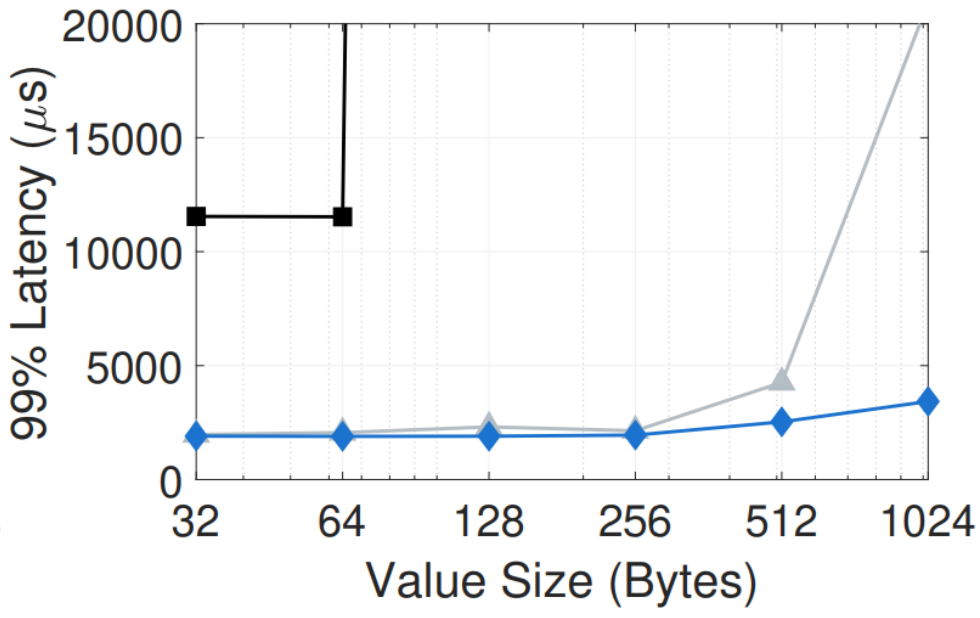
99th percentile latency

NetMC is robust to workload dynamics in terms of the multiget size

Impact of value size



Throughput



99th percentile latency

NetMC is robust to changes in value size

Conclusion

- We presented **NetMC**, a network-accelerated multiget coordination architecture
  - Providing **high throughput**, **low latency**, and **scalability** simultaneously

- NetMC offloads **stateless** and I/O-intensive request splitting to the programmable switch while handling **stateful** reply aggregation at the client
  - Our experimental results demonstrated that NetMC outperforms existing architectures

- We hope to fully leverage in-network computing capabilities to improve multiget operations and reduce coordination overhead in distributed key-value storage systems

Thank you!

Questions?