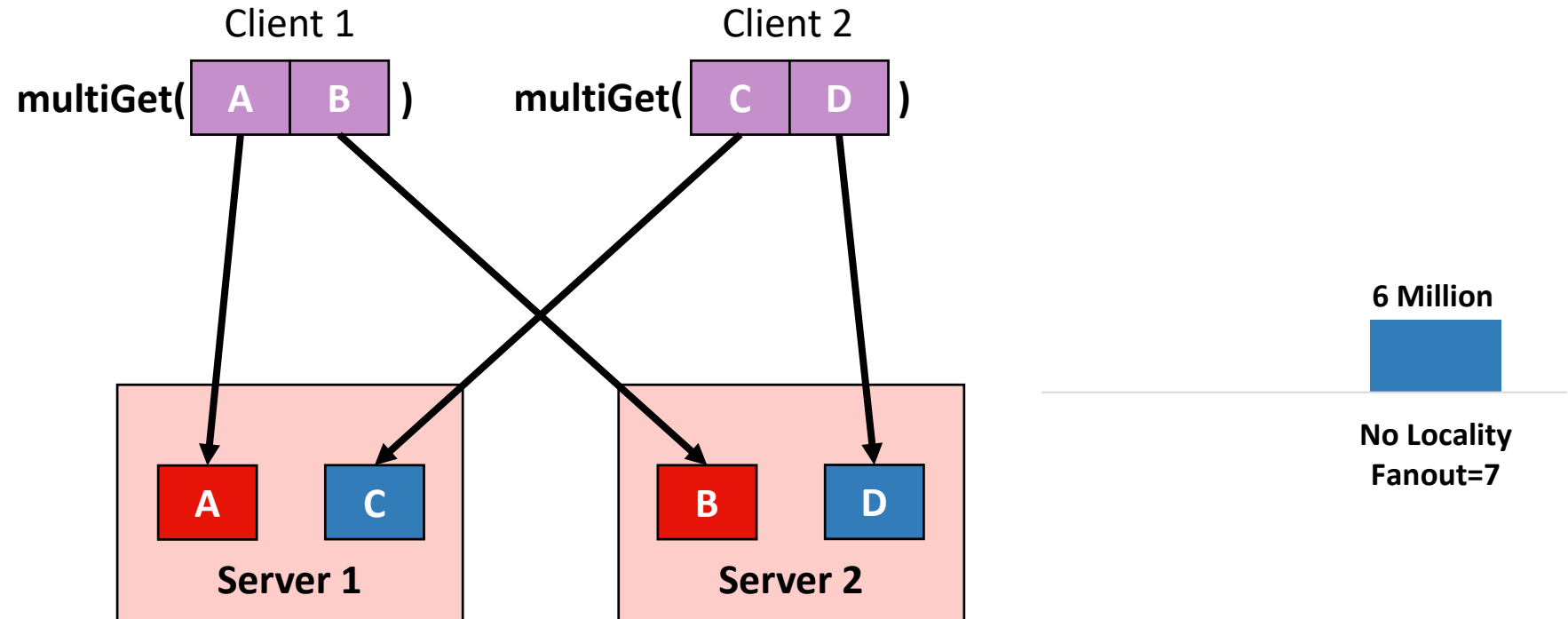# Rocksteady: Fast Migration for Low-Latency In-memory Storage

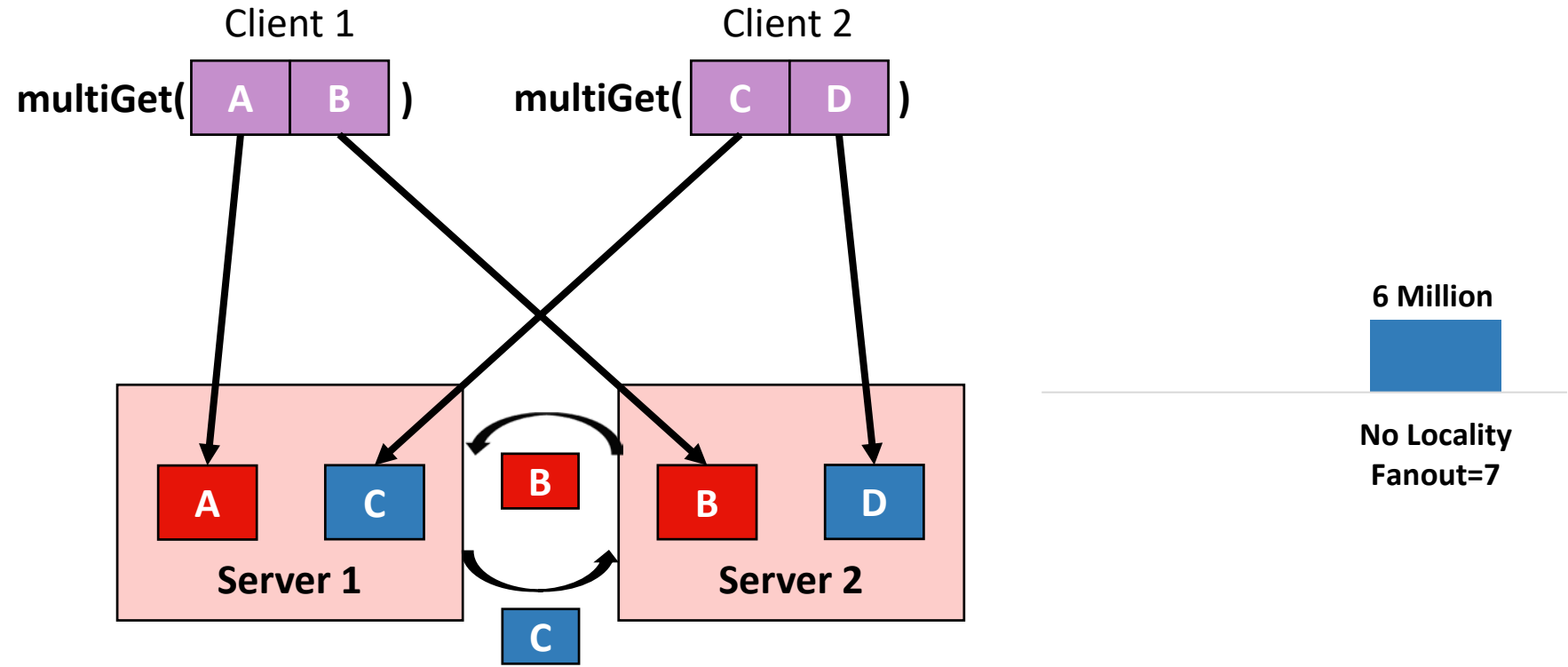**Chinmay Kulkarni**, Aniraj Kesavan, Tian Zhang, Robert Ricci, Ryan Stutsman

Introduction

- Distributed low-latency in-memory key-value stores are emerging
  - Predictable response times ~**10 μs median, ~60 μs 99.9th-tile**

- **Problem:** Must migrate data between servers
  - Minimize performance impact of migration → **go slow?**
  - Quickly respond to hot spots, skew shifts, load spikes → **go fast?**

- **Solution: Fast** data migration with **low impact**
  - Early ownership transfer of data, leverage workload skew
  - Low priority, parallel and adaptive migration

- **Result:** Migration protocol for RAMCloud in-memory key-value store
  - Migrates **256 GB in 6 minutes,** 99.9th-tile latency less than **250 μs**
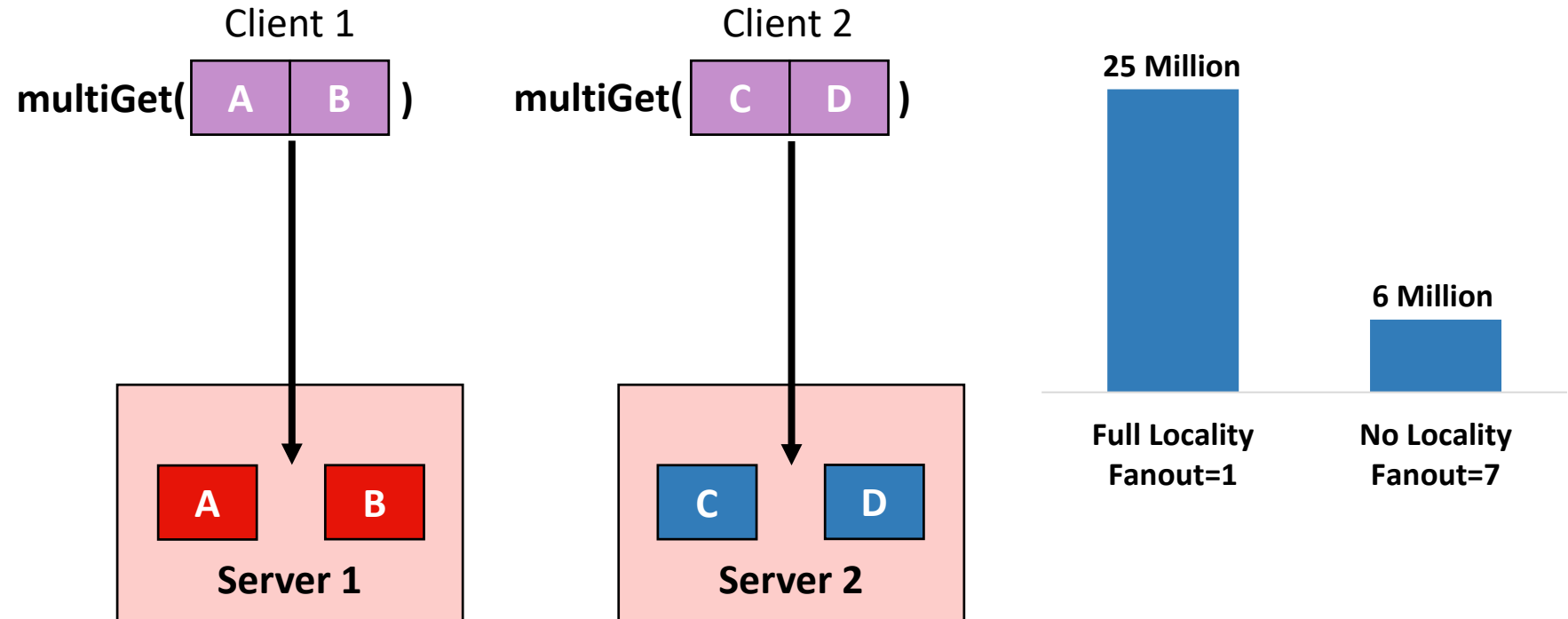 - Median latency recovers from **40 μs to 20 μs in 14 s**

Why Migrate Data?

multiGet(A B **)**   **multiGet(** C D **)**

Client 1   Client 2

6 Million

**No Locality
Fanout=7**

A   C
Server 1

B   D
Server 2

Poor spatial locality → High multiGet() fan-out → More RPCs

Migrate To Improve Spatial Locality

Spatial Locality Improves Throughput

Client 1

multiGet([A | B])

Client 2

multiGet([C | D])

25 Million

6 Million

**Full Locality
Fanout=1**

**No Locality
Fanout=7**

A	B
Server 1
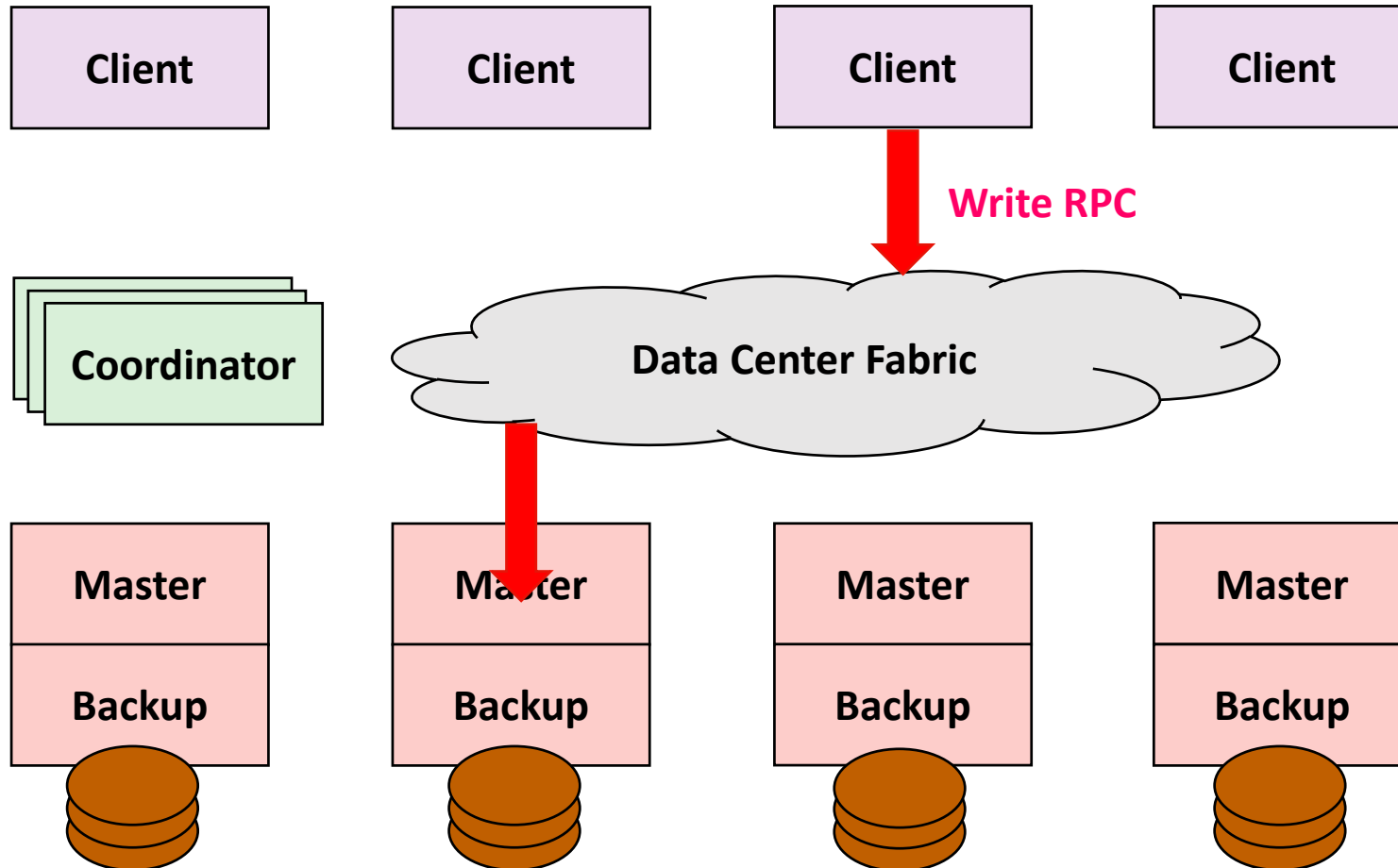
C	D
Server 2

Better spatial locality → Fewer RPCs → Higher throughput

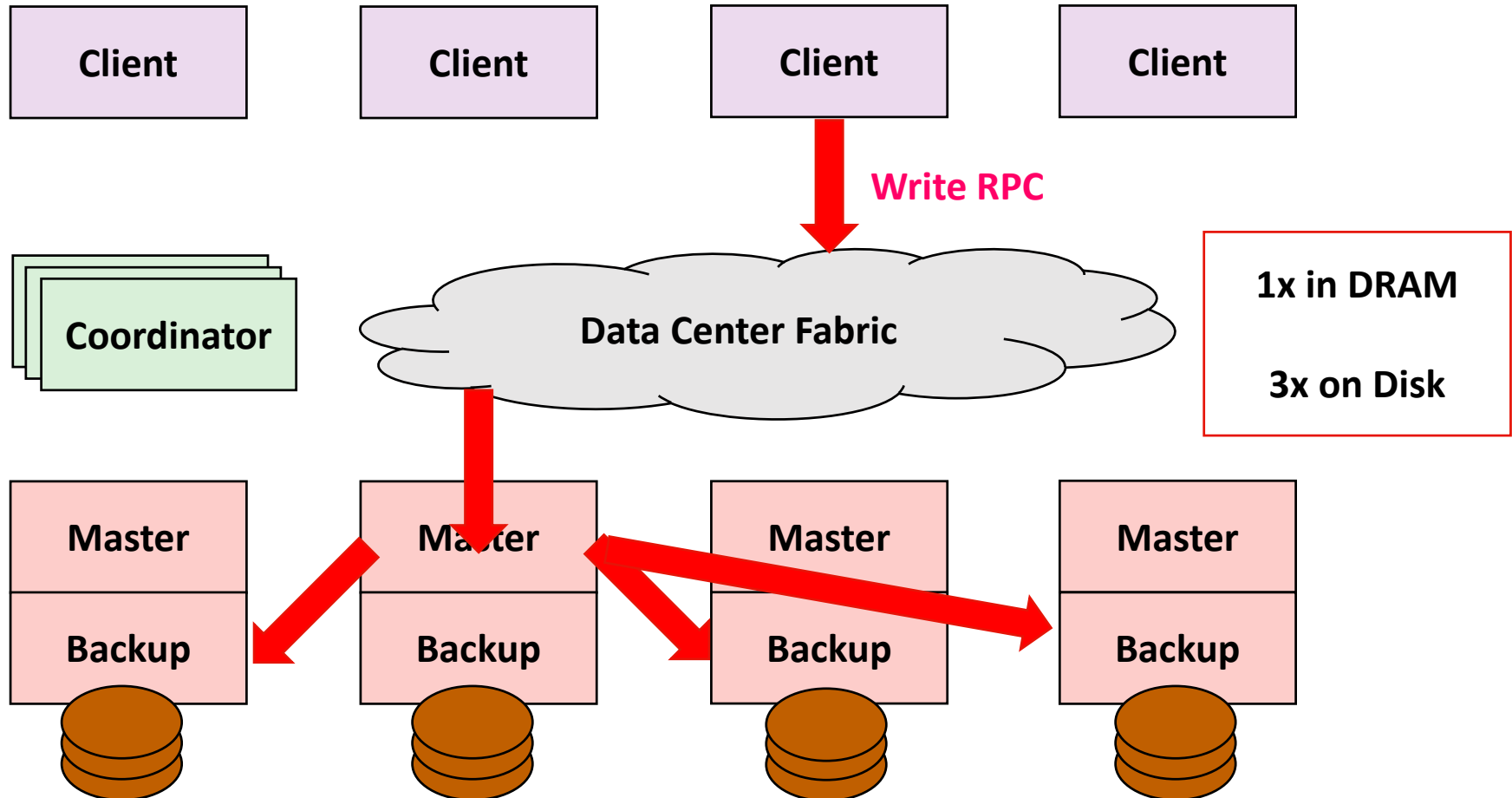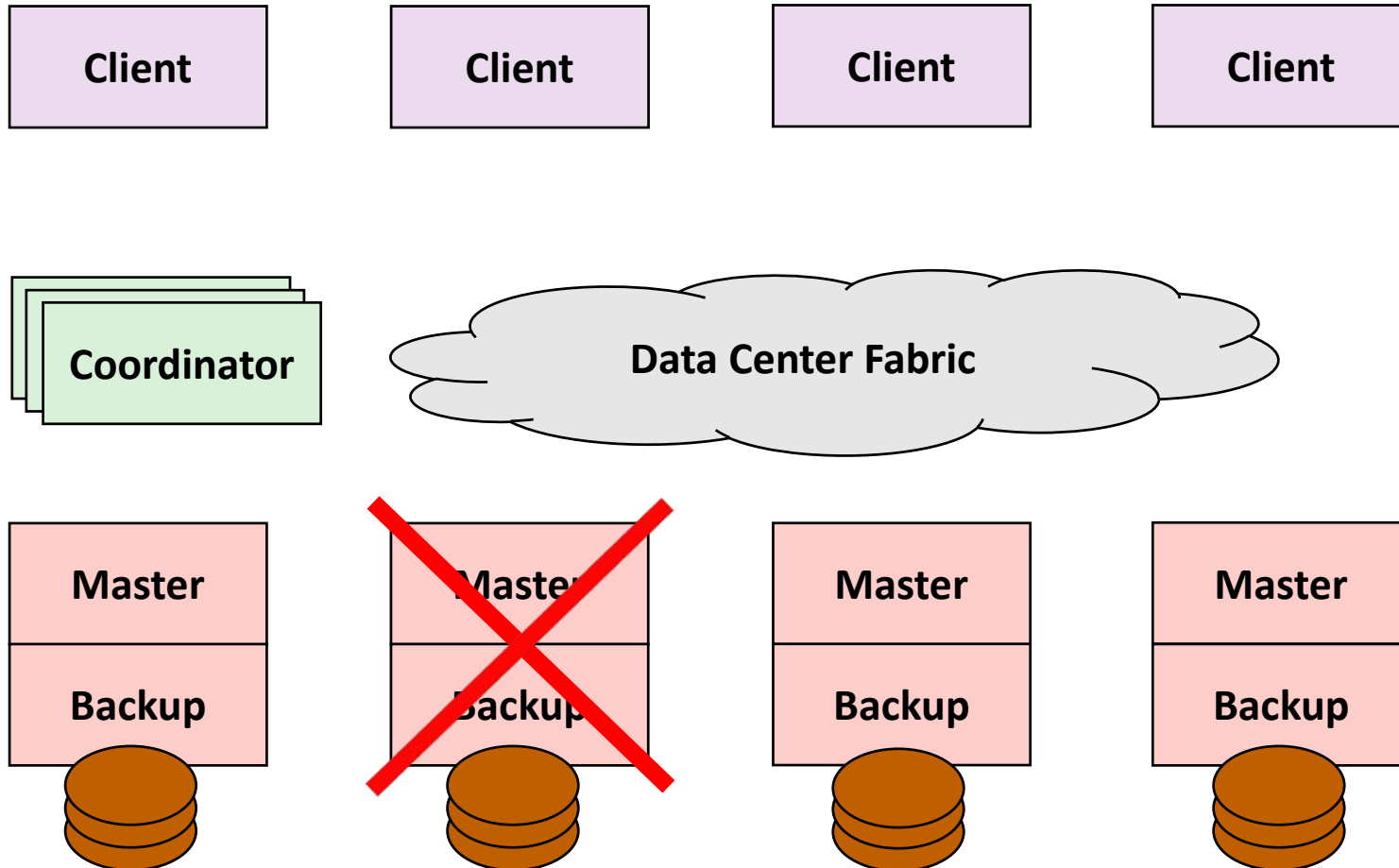**Benefits multiGet(), range scans**

The RAMCloud Key-Value Store
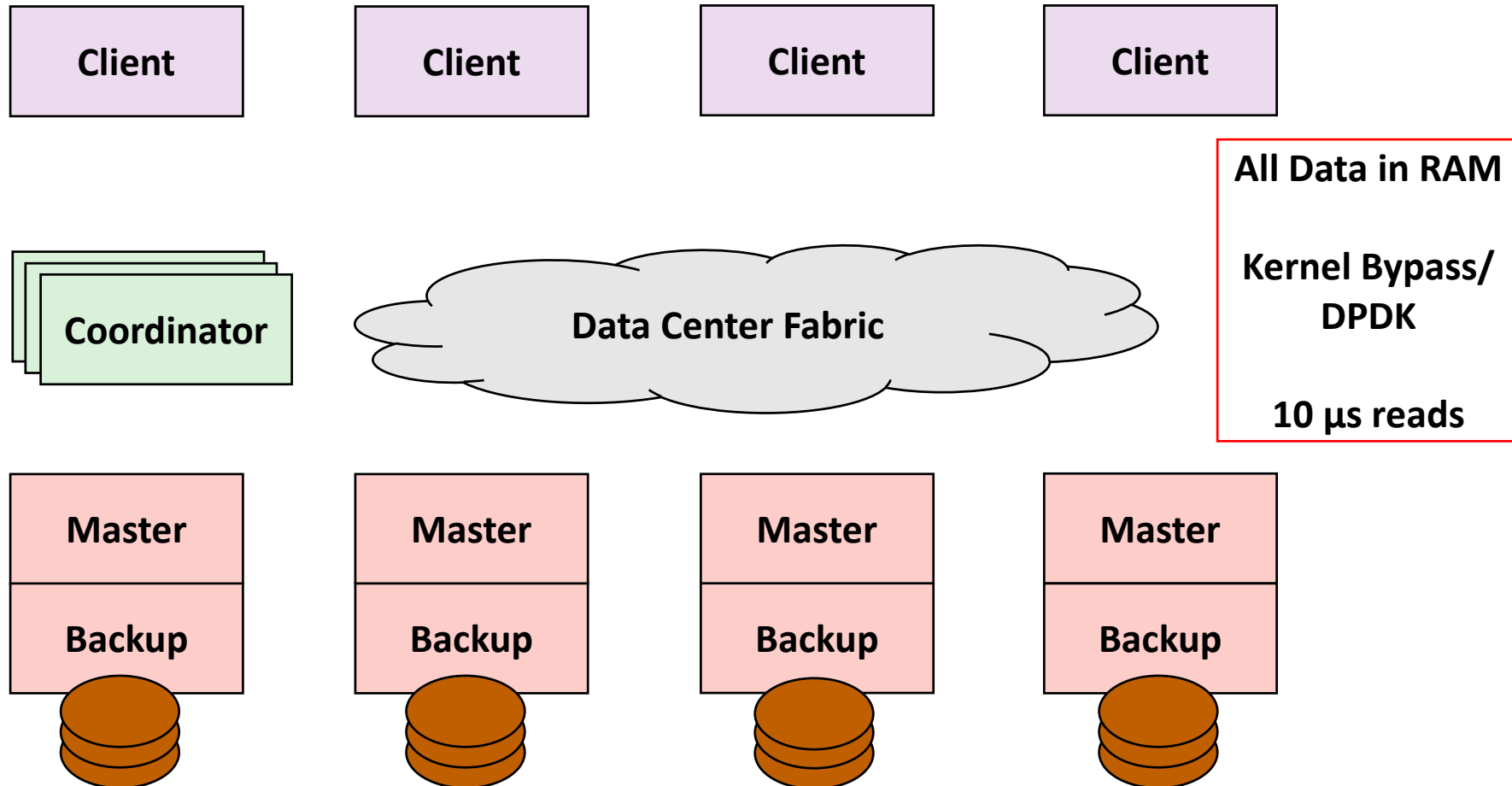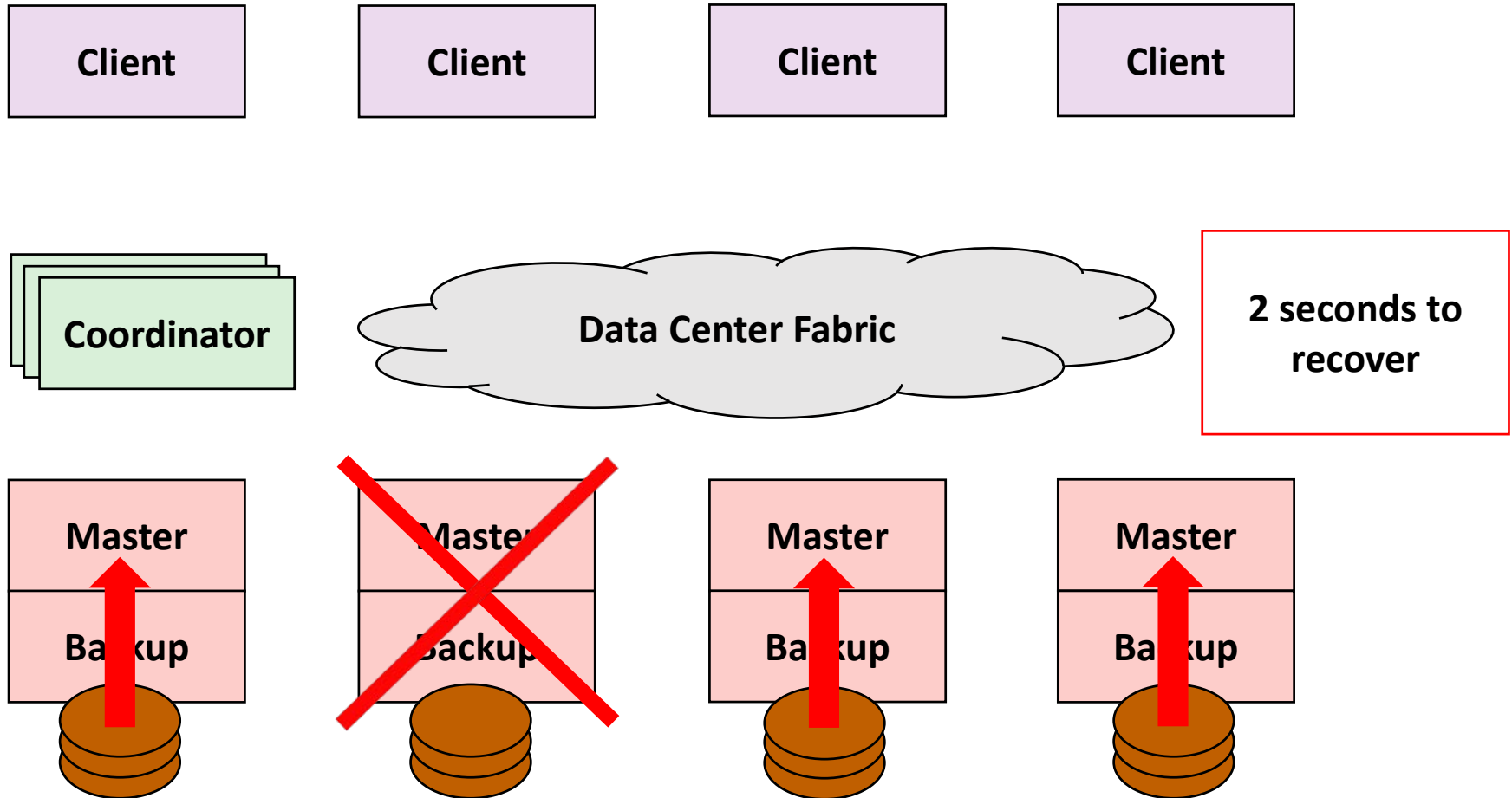
The RAMCloud Key-Value Store

The RAMCloud Key-Value Store

Fault-tolerance & Recovery In RAMCloud

Fault-tolerance & Recovery In RAMCloud

| Client | Client | Client | Client |

Coordinator

Data Center Fabric

2 seconds to recover

Master
Backup

Master
**Backup**

Master
Backup

Master
Backup

Performance Goals For Migration

- **Maintain low access latency**
  - 10 µsec median latency → System extremely sensitive
  - Tail latency matters at scale → Even more sensitive

- **Migrate data fast**
  - Workloads dynamic → Respond quickly
  - Growing DRAM storage: 512 GB per server
    - Slow data migration → Entire day to scale cluster

Rocksteady Overview: Early Ownership Transfer
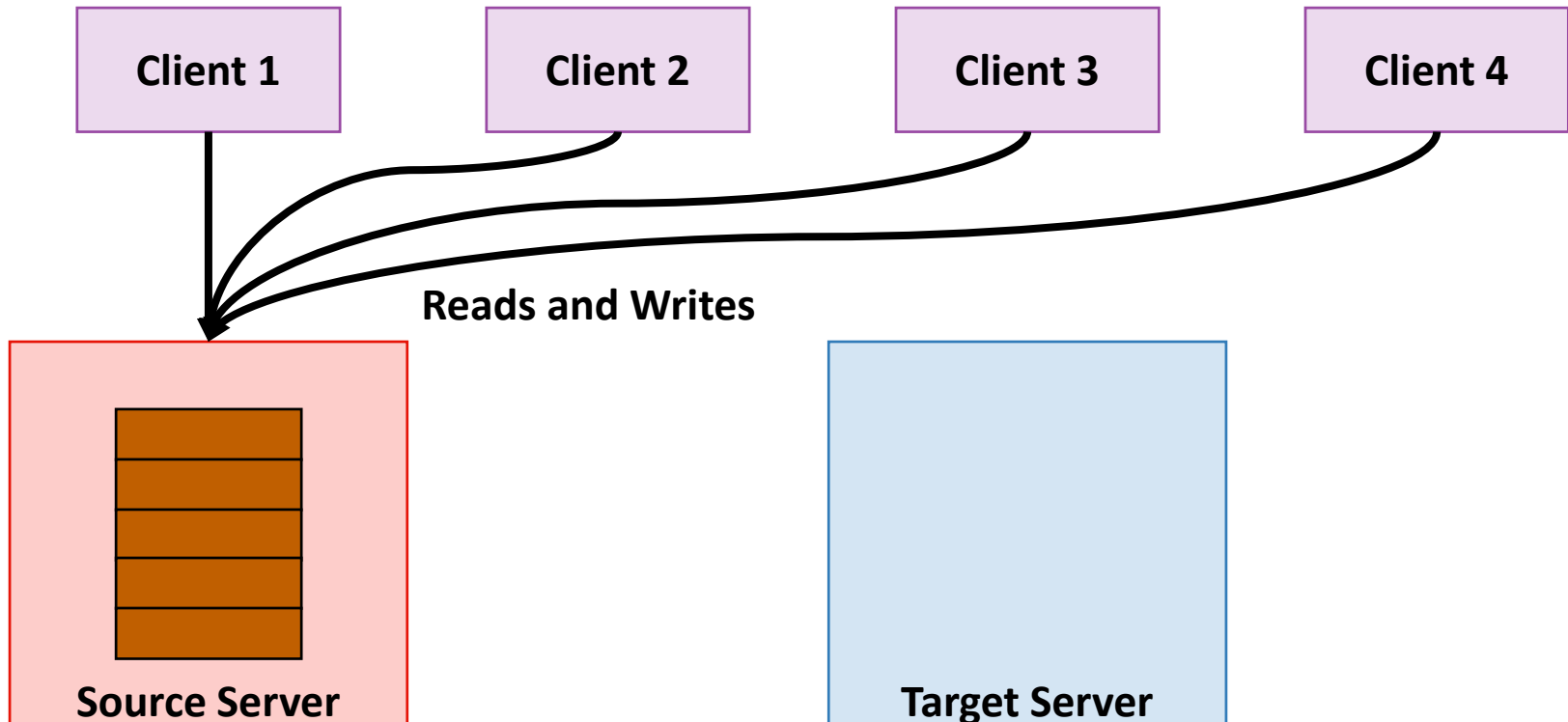
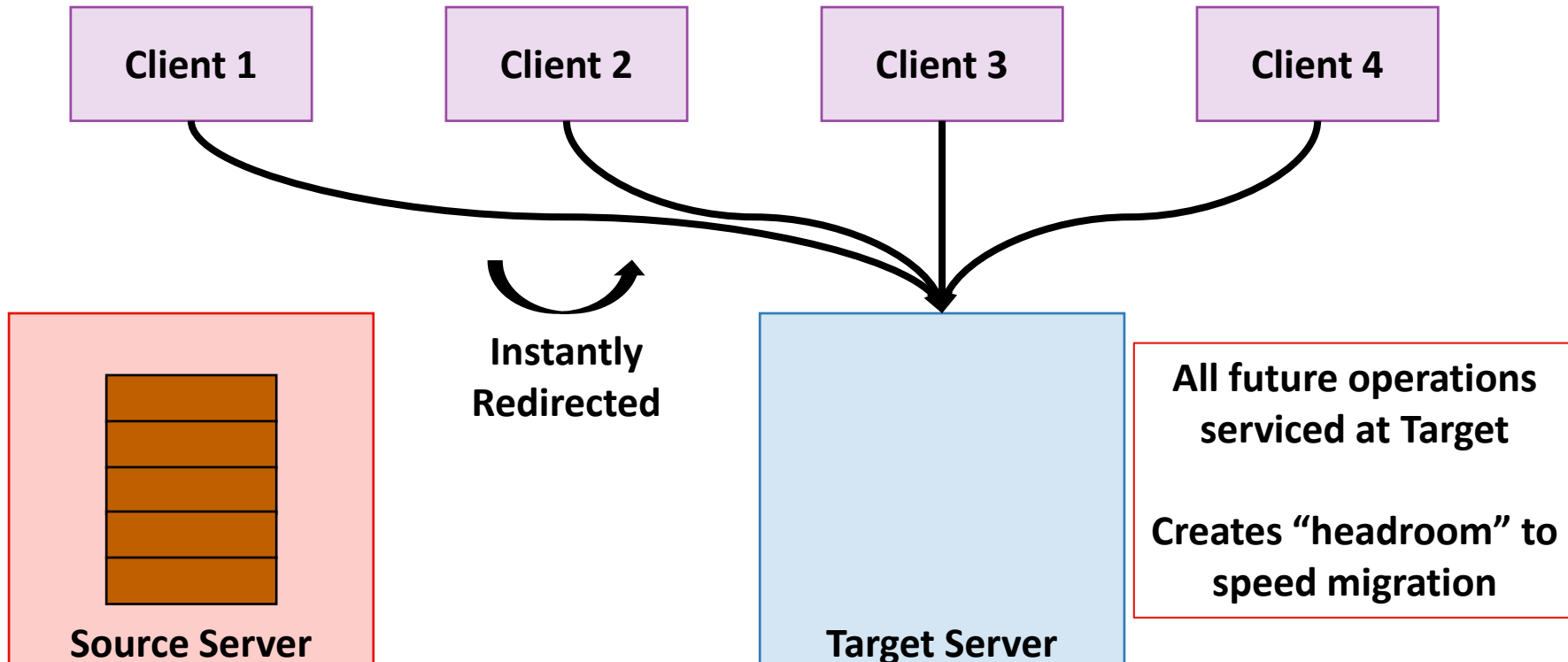**Problem:** Loaded source can bottleneck migration

Solution: Instantly shift ownership and all load to target

Rocksteady Overview: Early Ownership Transfer

Problem: Loaded source can bottleneck migration

Solution: Instantly shift ownership and all load to target

Client 1   Client 2   Client 3   Client 4

Instantly Redirected

Source Server

Target Server

All future operations serviced at Target

Creates "headroom" to speed migration

Rocksteady Overview: Leverage Skew

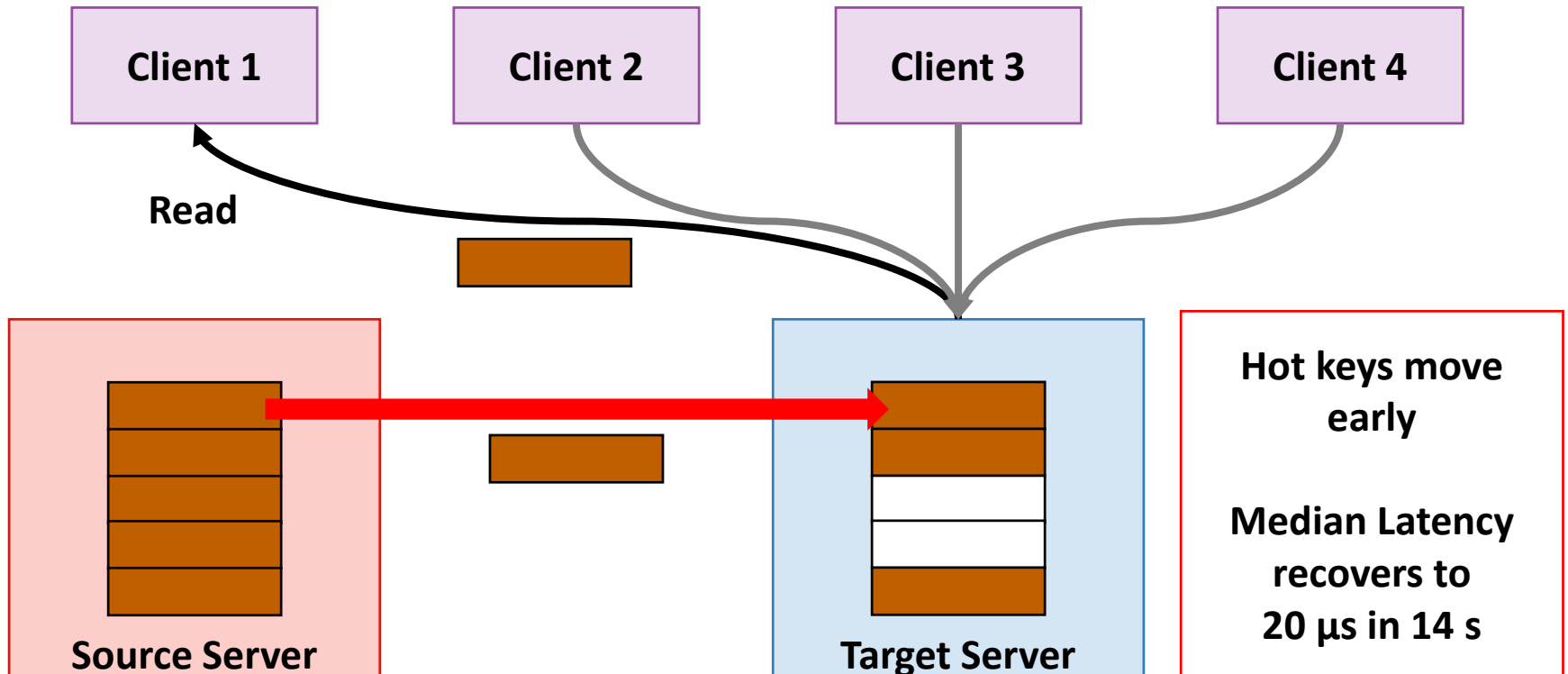**Problem:** Data has not arrived at source yet

Solution: On demand migration of unavailable data
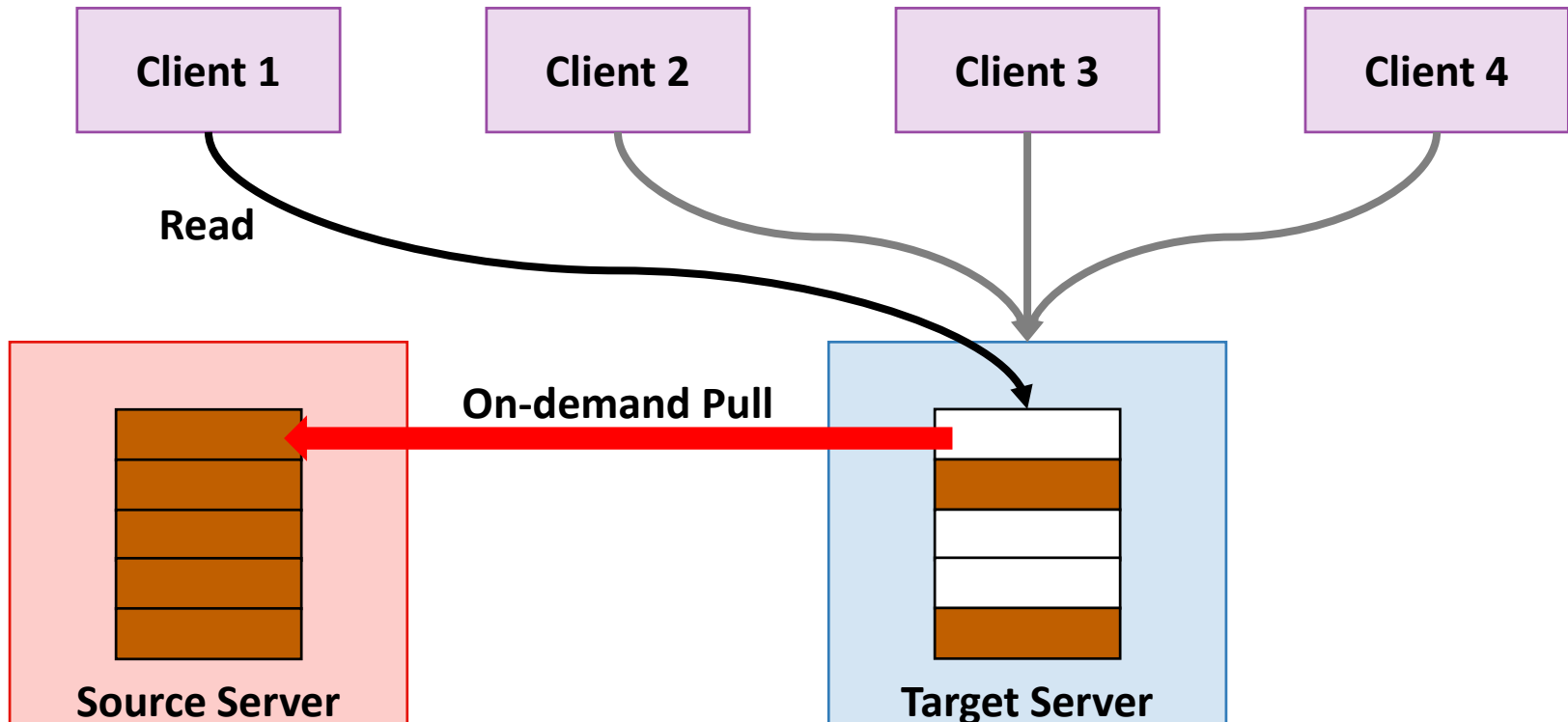
Rocksteady Overview: Leverage Skew

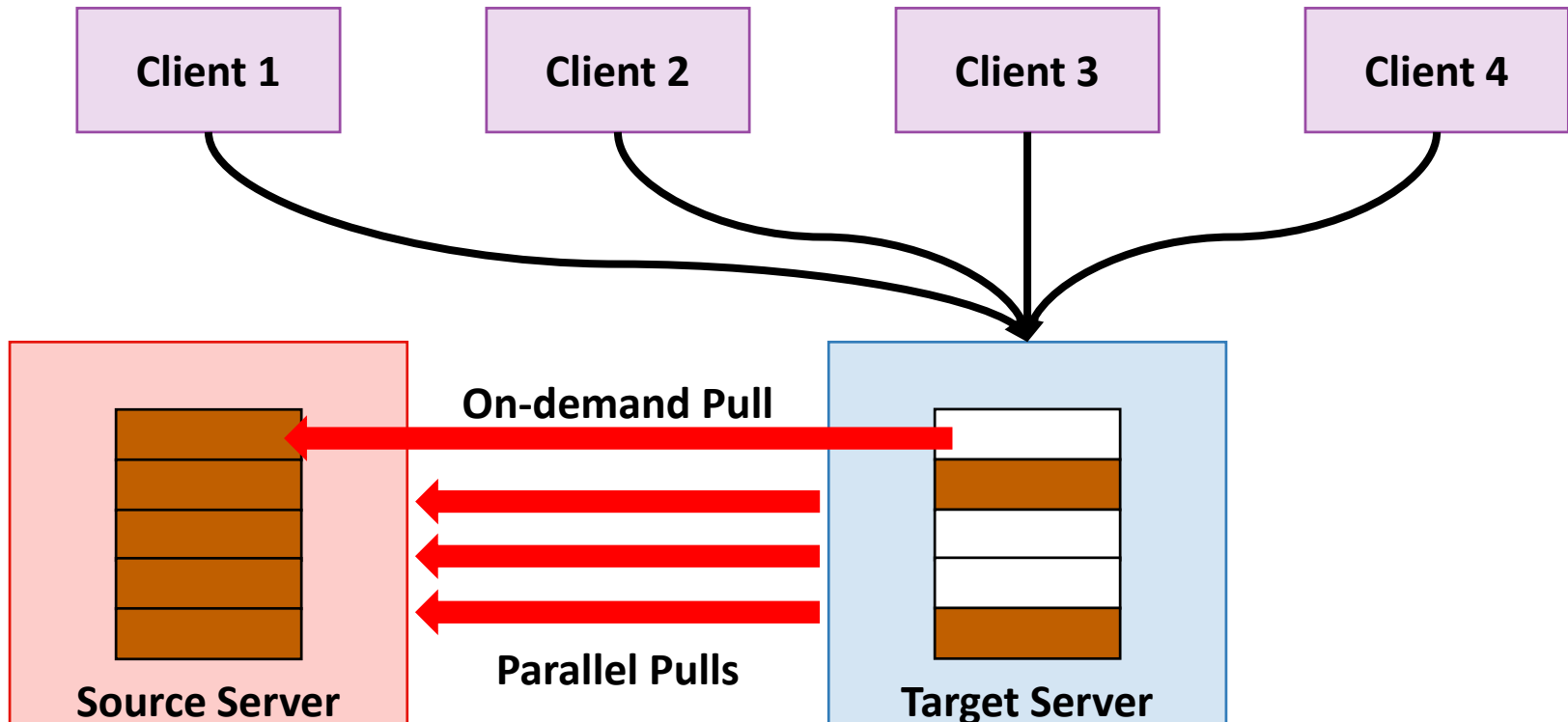**Problem:** Data has not arrived at source yet

Solution: On demand migration of unavailable data

Rocksteady Overview: Adaptive and Parallel

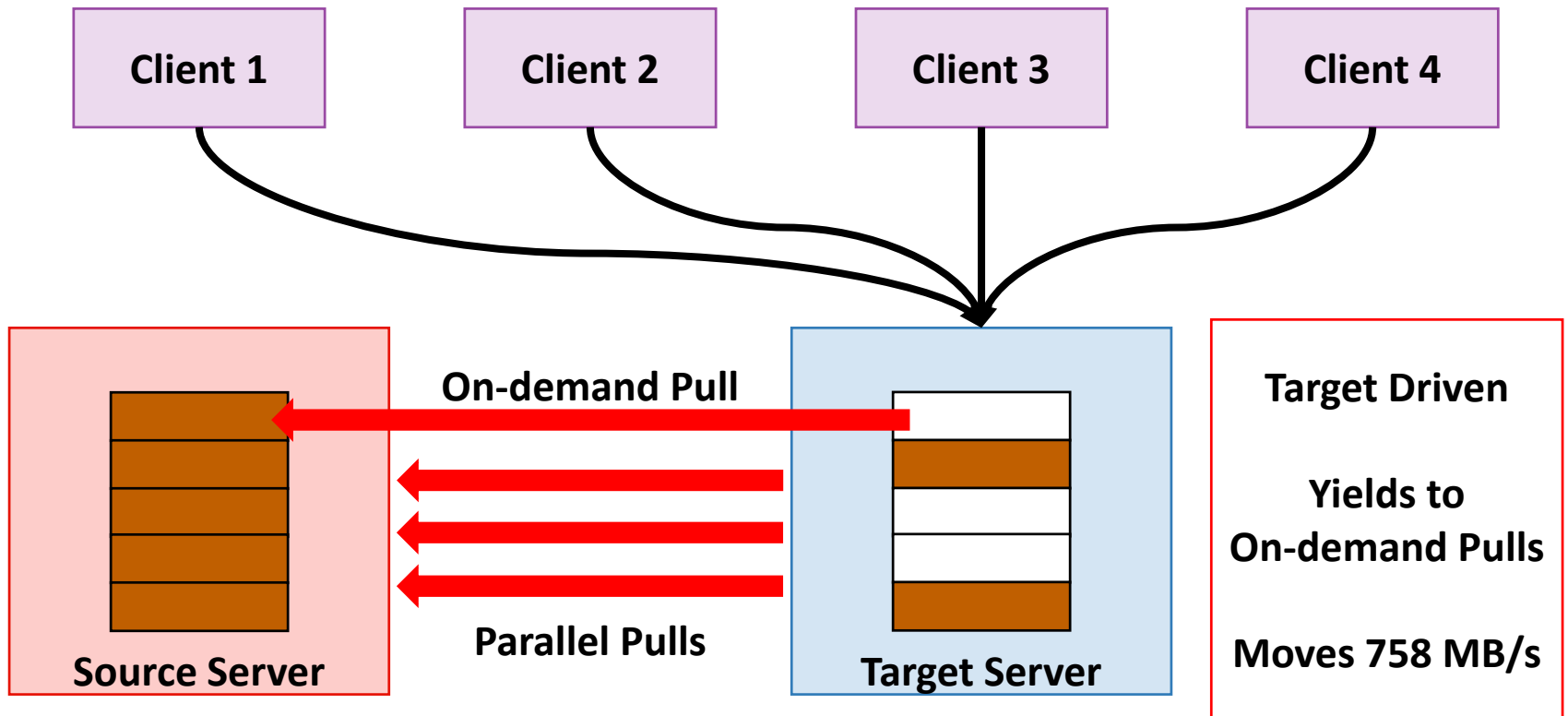**Problem: Old single-threaded protocol limited to 130 MB/s**

Solution: Pipelined and parallel at source and target

Rocksteady Overview: Adaptive and Parallel

**Problem:** Old single-threaded protocol limited to 130 MB/s

Solution: Pipelined and parallel at source and target

| Client 1 | Client 2 | Client 3 | Client 4 |

On-demand Pull

Parallel Pulls

Source Server

Target Server

Target Driven

Yields to On-demand Pulls

Moves 758 MB/s

Rocksteady Overview: Eliminate Sync Replication

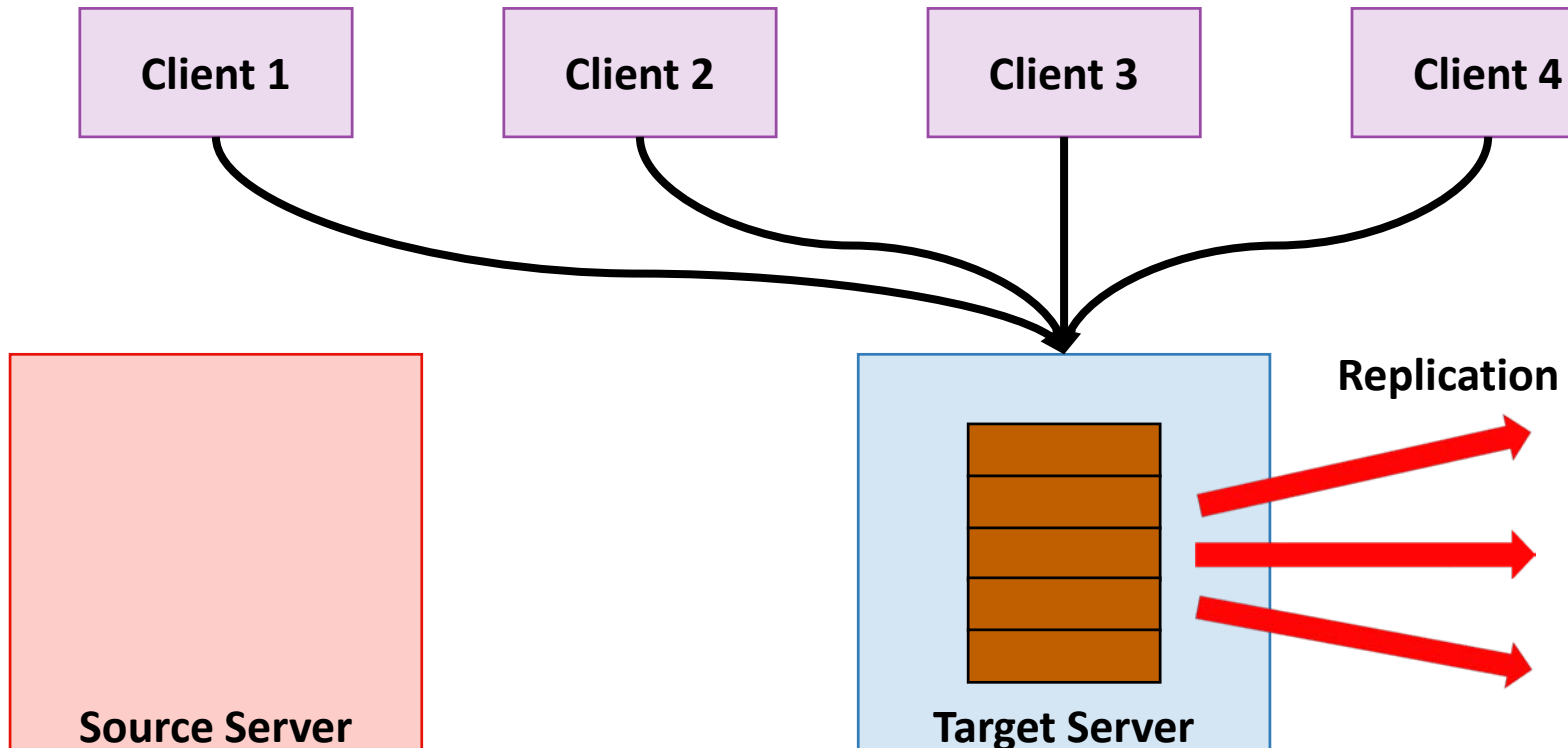**Problem:** Synchronous replication bottleneck at target

Solution: Safely defer replication until after migration
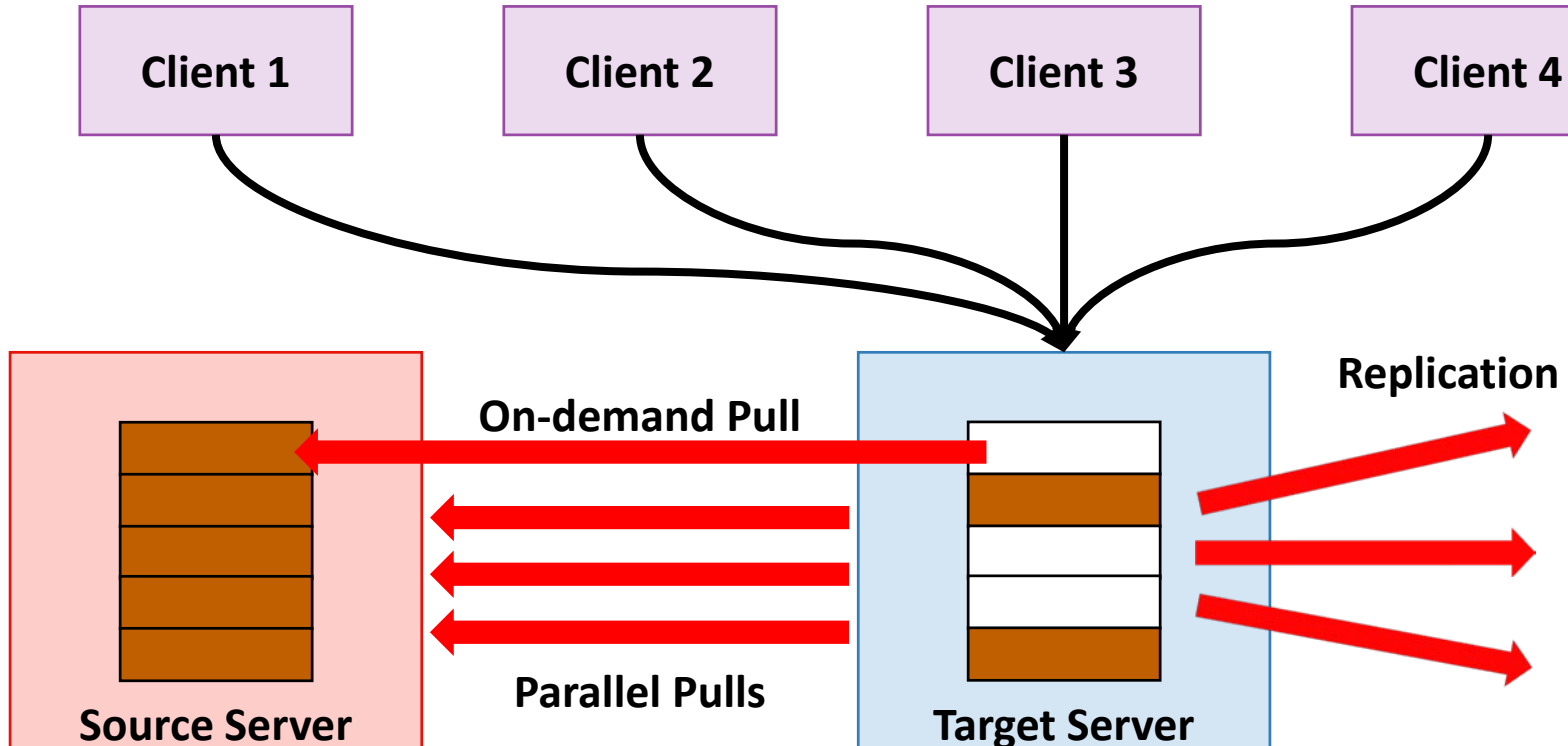
Rocksteady Overview: Eliminate Sync Replication

Problem: Synchronous replication bottleneck at target

Solution: Safely defer replication until after migration

Rocksteady: Putting it all together

- **Instantaneous ownership transfer**
  - Immediate load reduction at overloaded source
  - Creates "headroom" for migration work

- **Leverage skew to rapidly migrate hot data**
  - Target comes up to speed with little data movement

- **Adaptive parallel, pipelined at source and target**
  - All cores avoid stalls, but yield to client-facing operations

- **Safely defer replication at target**
  - Eliminates replication bottleneck and contention

Rocksteady

- **Instantaneous ownership transfer**

- Leverage skew to rapidly migrate hot data

- Adaptive parallel, pipelined at source and target

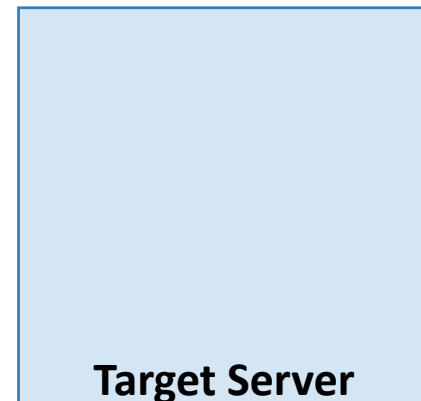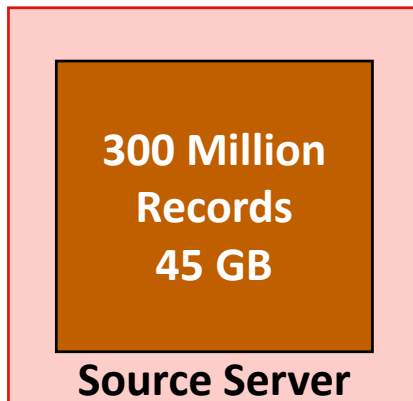- Safely defer synchronous replication at target

Evaluation Setup

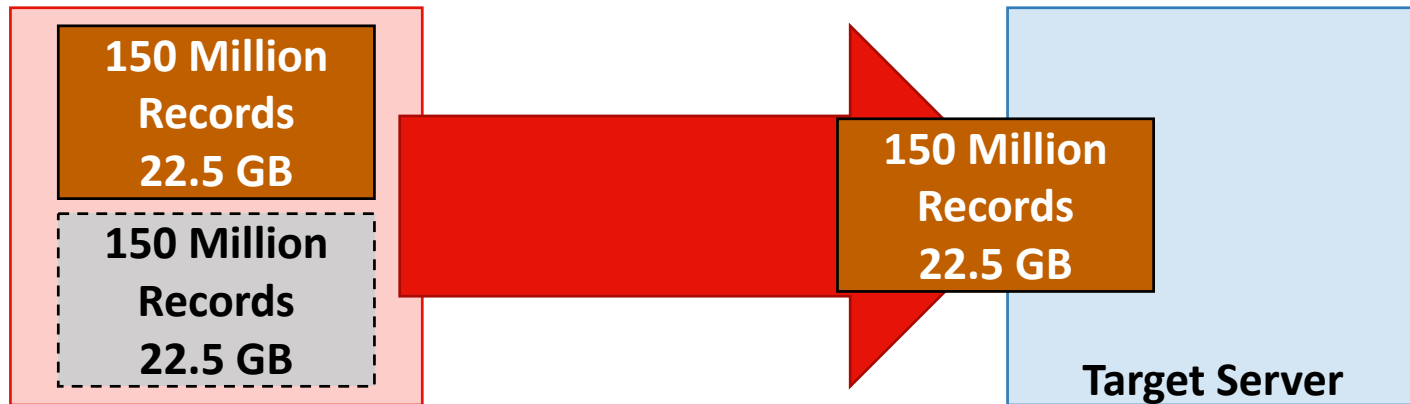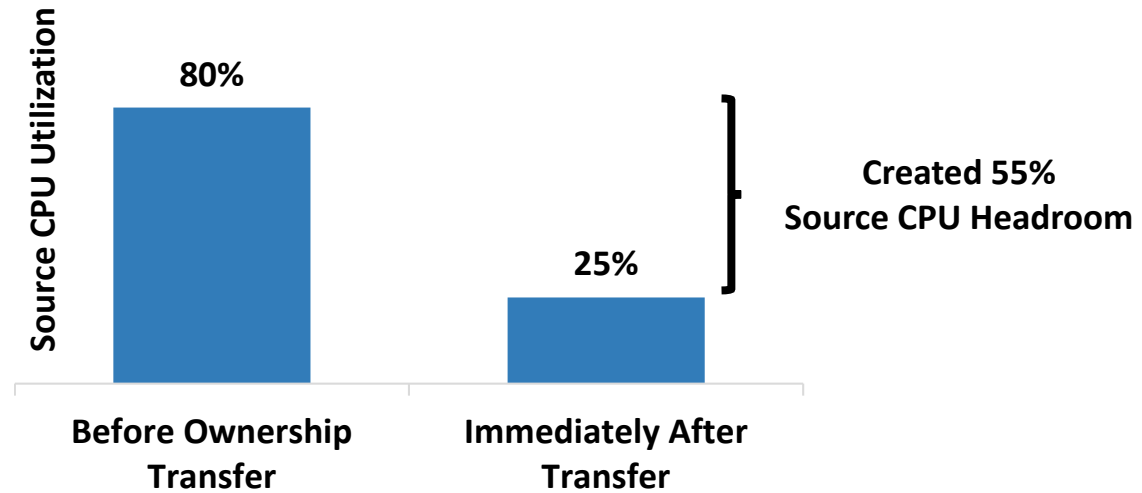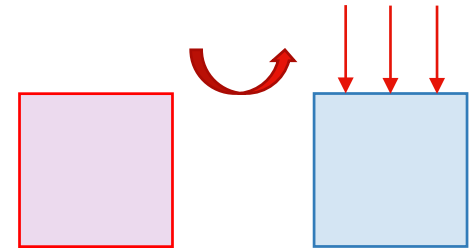| Client YCSB-B (95/5) Skew=0.99	Client YCSB-B (95/5) Skew=0.99	Client YCSB-B (95/5) Skew=0.99	Client YCSB-B (95/5) Skew=0.99

300 Million Records 45 GB

Source Server

Target Server

Evaluation Setup

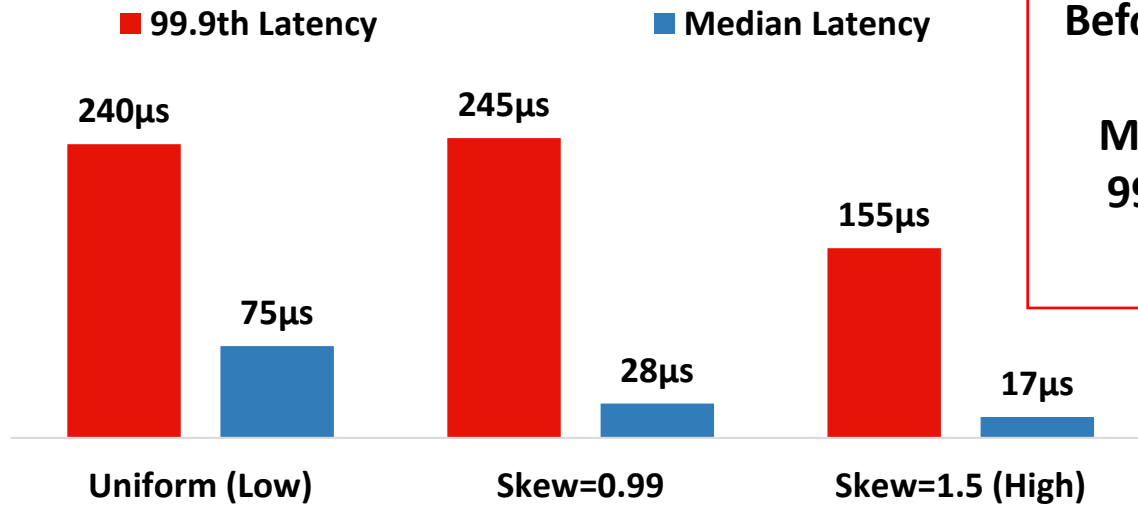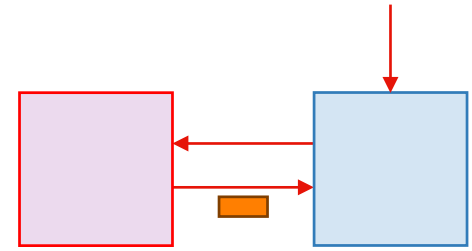# Instantaneous Ownership Transfer

Before migration: Source over-loaded, Target under-loaded

Ownership transfer creates Source headroom for migration

Rocksteady

- Instantaneous ownership transfer

- **Leverage skew to rapidly migrate hot data**

- Adaptive parallel, pipelined at source and target

- Safely defer synchronous replication at target

Leverage Skew To Move Hot Data


Before Migration:

Median=10 μs
99.9th = 60 μs

■ 99.9th Latency   ■ Median Latency

240μs   245μs

155μs

75μs

28μs

17μs

Uniform (Low)   Skew=0.99   Skew=1.5 (High)

After ownership transfer, hot keys pulled on-demand

More skew → Median restored faster (migrate fewer hot keys)
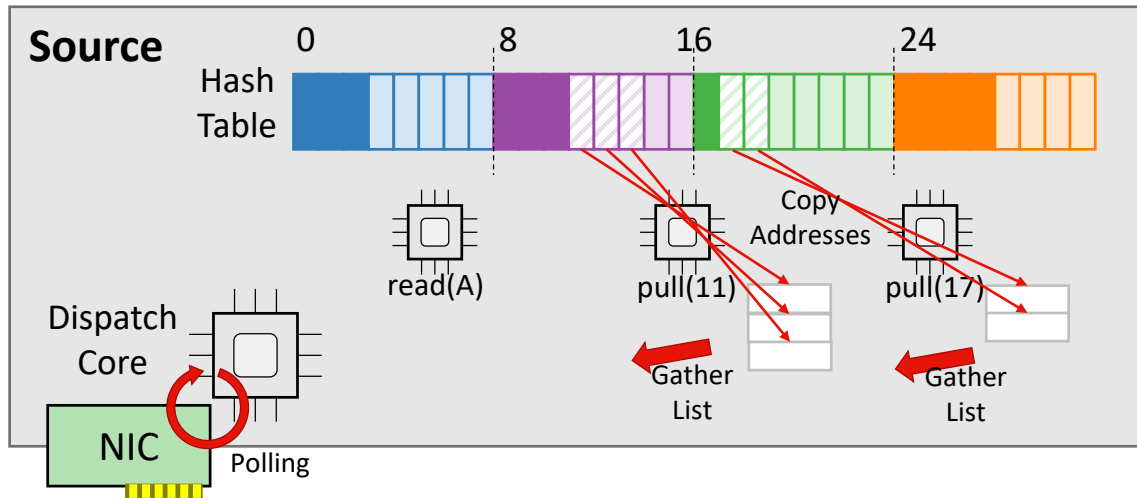
Rocksteady

- Instantaneous ownership transfer

- Leverage skew to rapidly migrate hot data

- **Adaptive parallel, pipelined at source and target**

- Safely defer synchronous replication at target

Parallel, Pipelined, & Adaptive Pulls
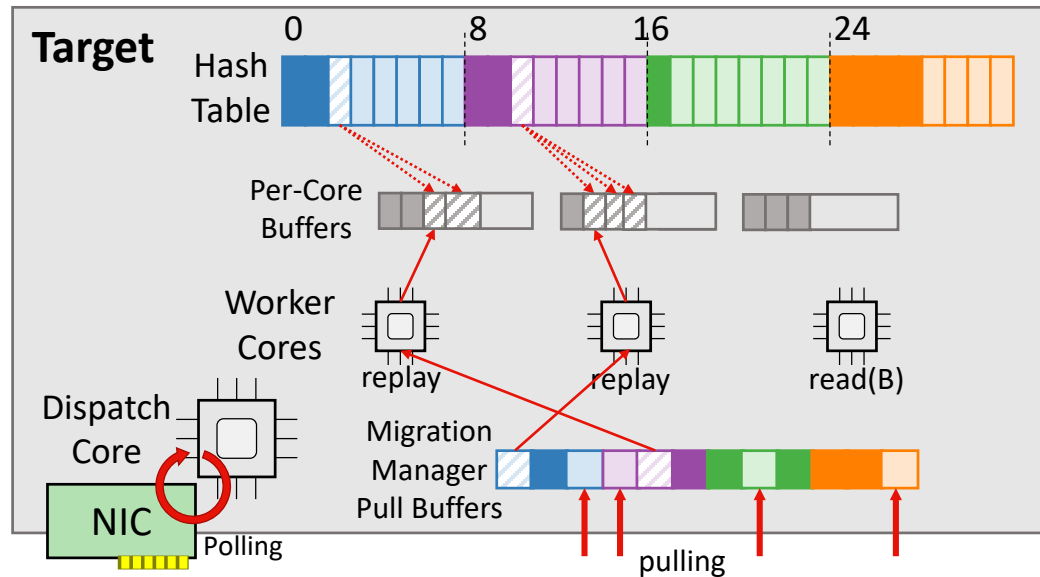
- **Target driven, migration manager**

- **Co-partitioned hash tables, pull from partitions in parallel**

- **Replay pulled data into per-core buffers**

Parallel, Pipelined, & Adaptive Pulls

- **Stateless passive Source**

- **Granular 20 KB pulls**

# Parallel, Pipelined, & Adaptive Pulls

- **Redirect any idle CPU for migration**

- **Migration yields to regular requests, on-demand pulls**

Rocksteady

- Instantaneous ownership transfer

- Leverage skew to rapidly migrate hot data

- Adaptive parallel, pipelined at source and target
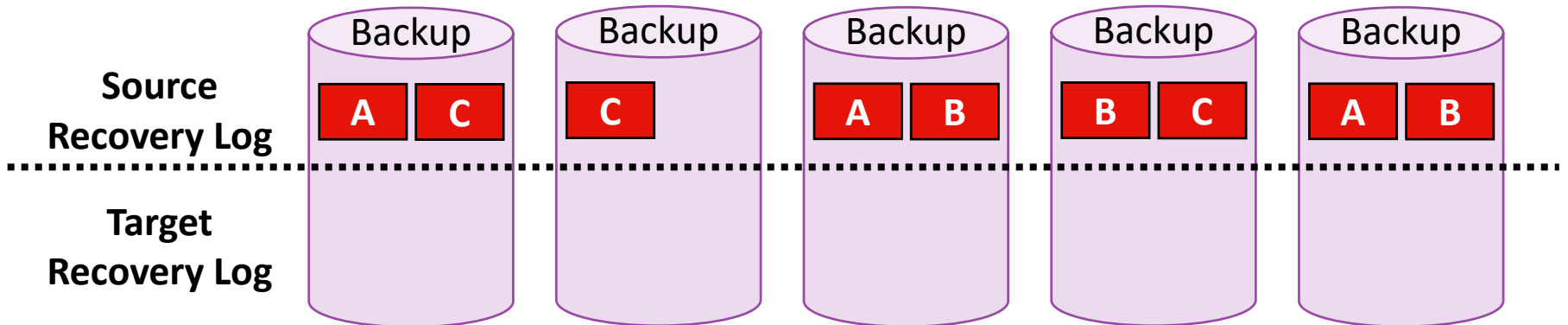
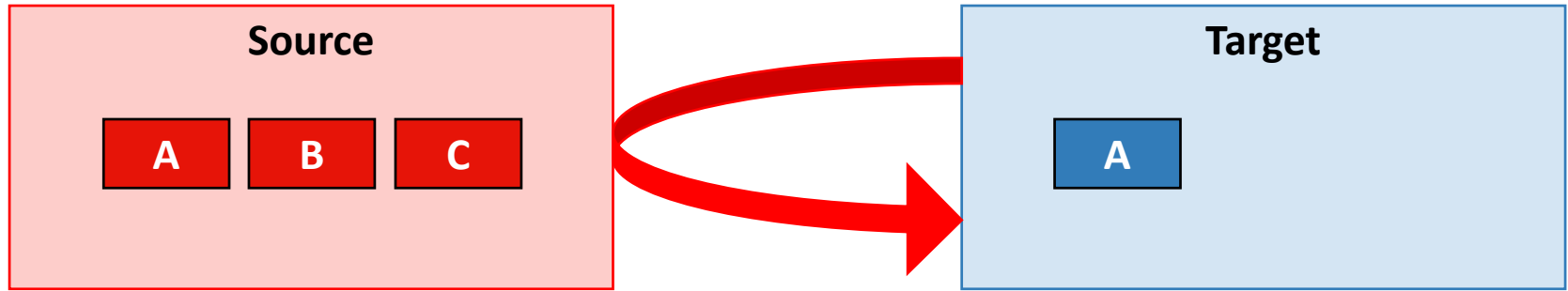- **Safely defer synchronous replication at target**

Naïve Fault Tolerance During Migration

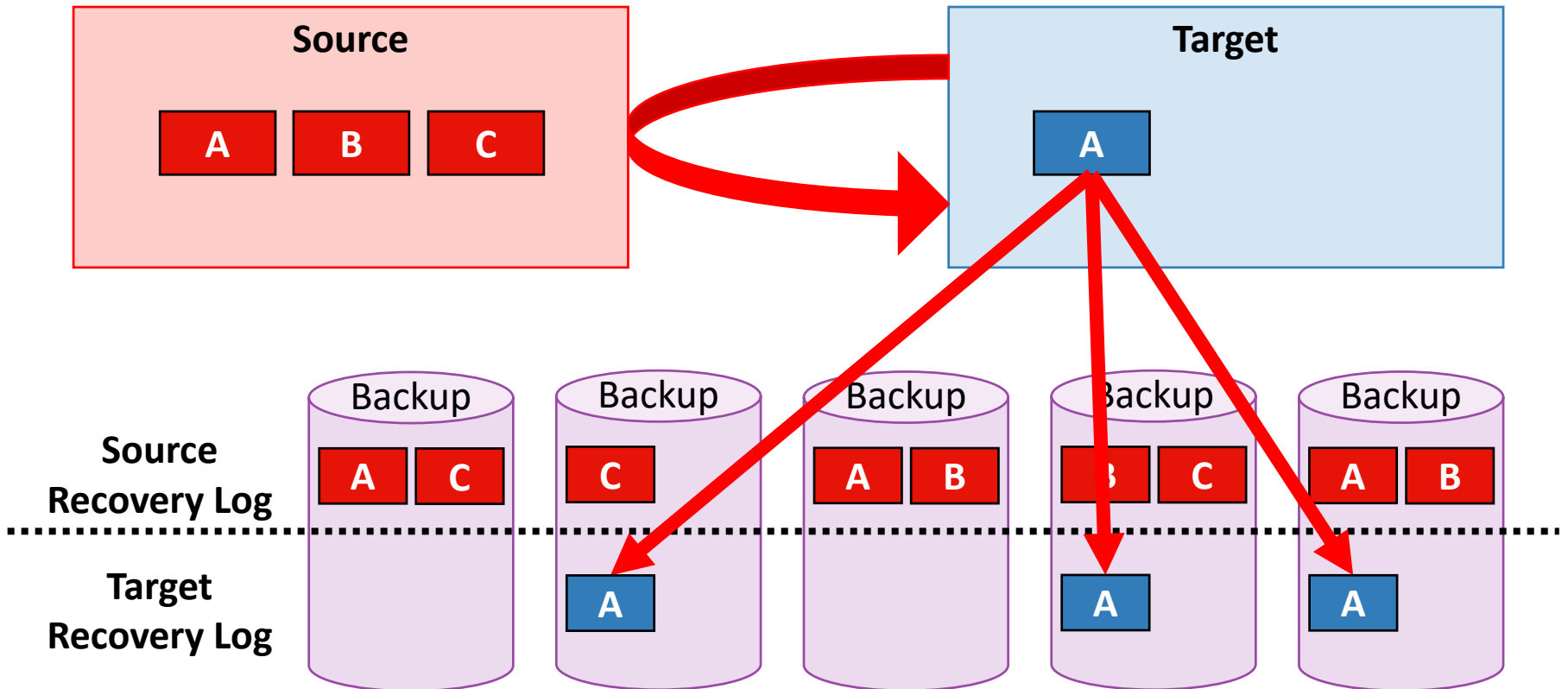**Each server has a recovery log distributed across the cluster**
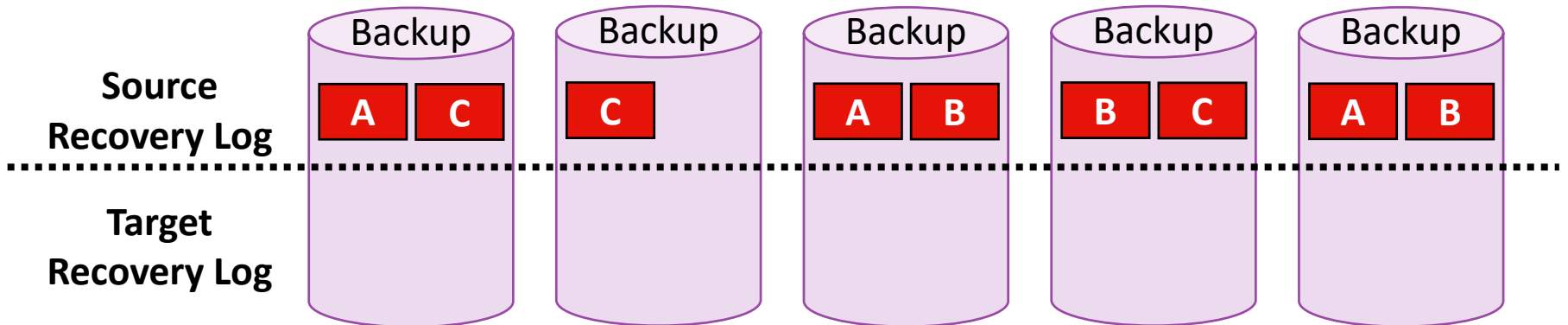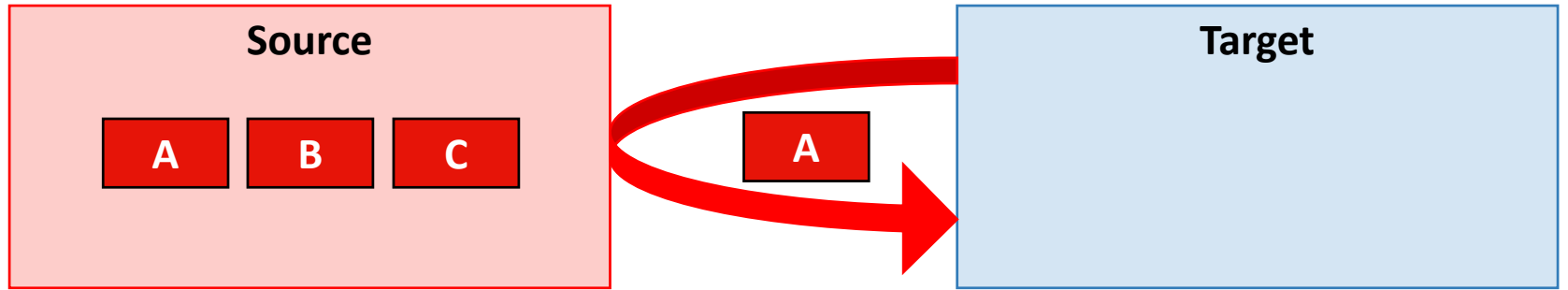
Naïve Fault Tolerance During Migration

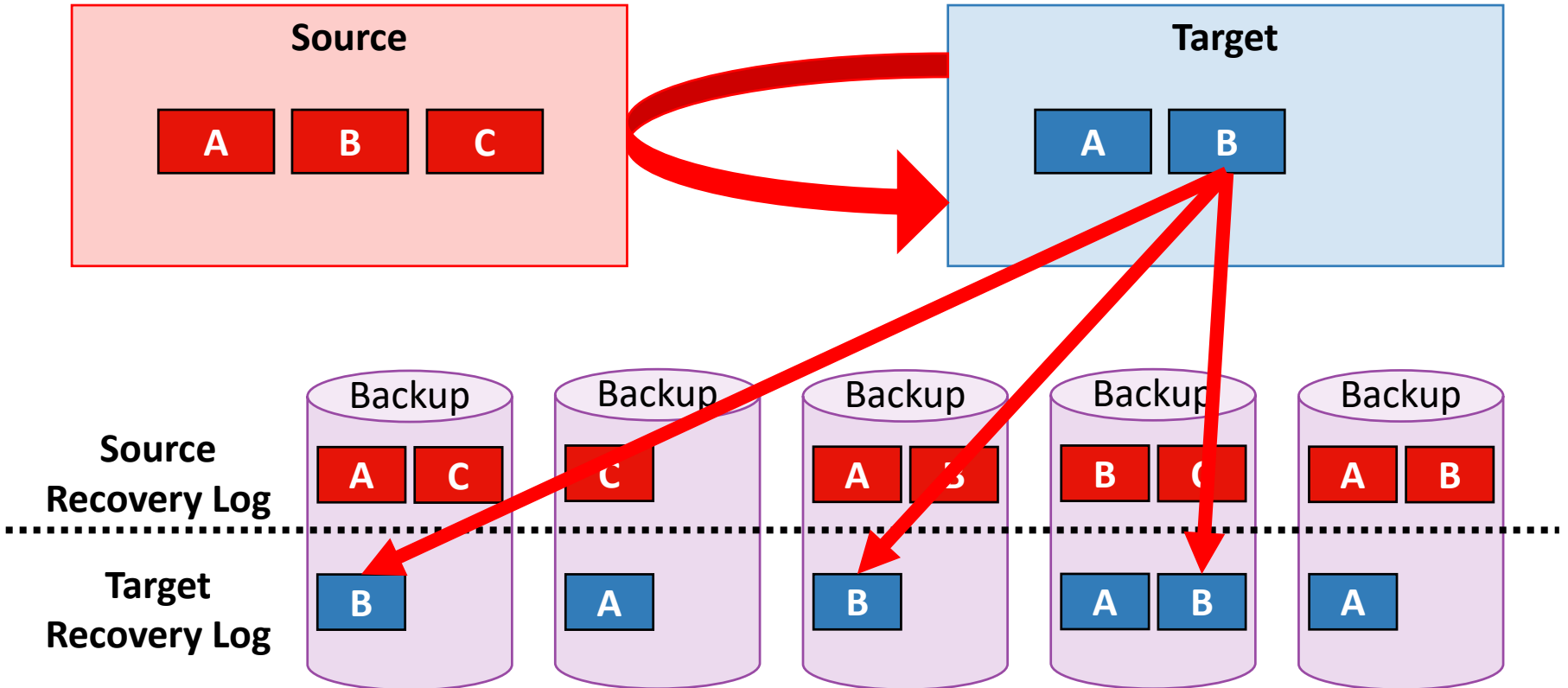**Migrated data needs to be triplicated to target's recovery log**

33

Naïve Fault Tolerance During Migration

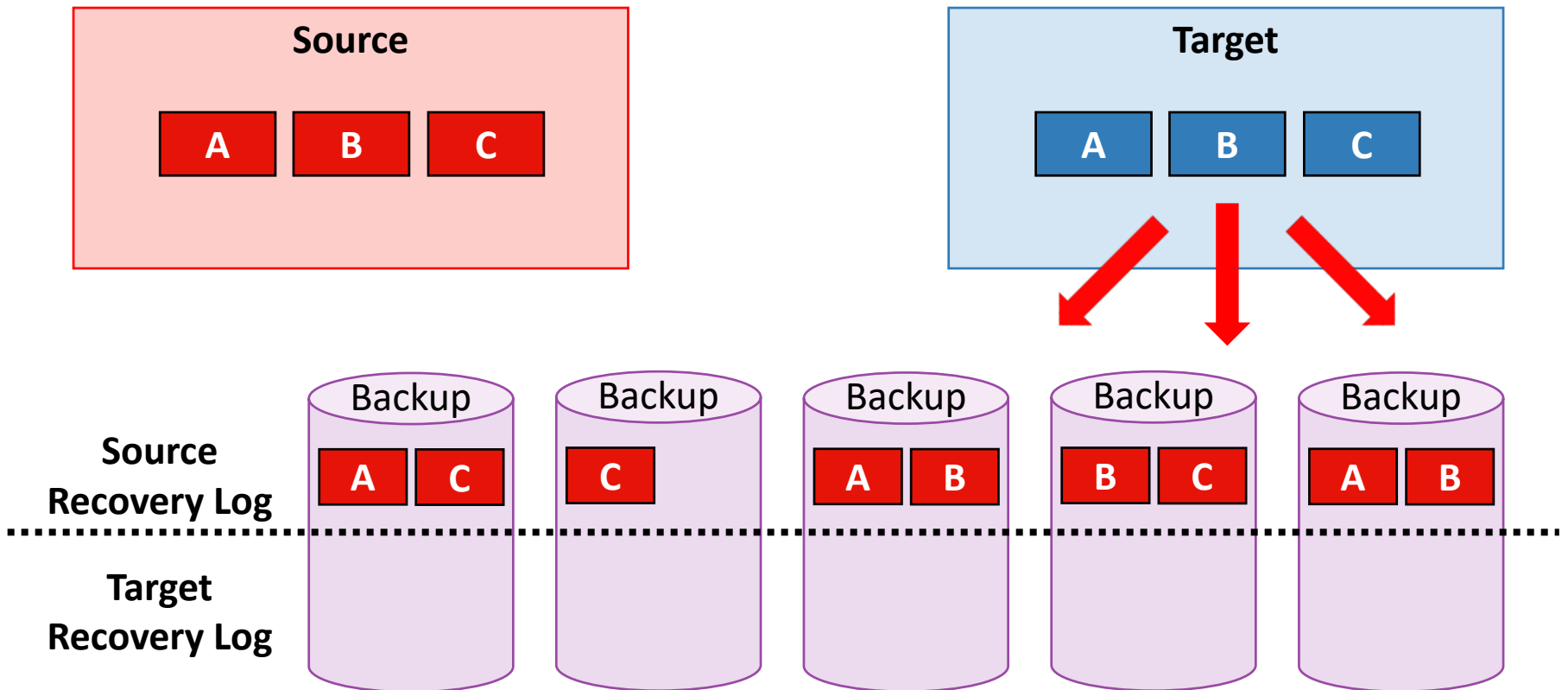**Migrated data needs to be triplicated to target's recovery log**

34

Synchronous Replication Bottlenecks Migration

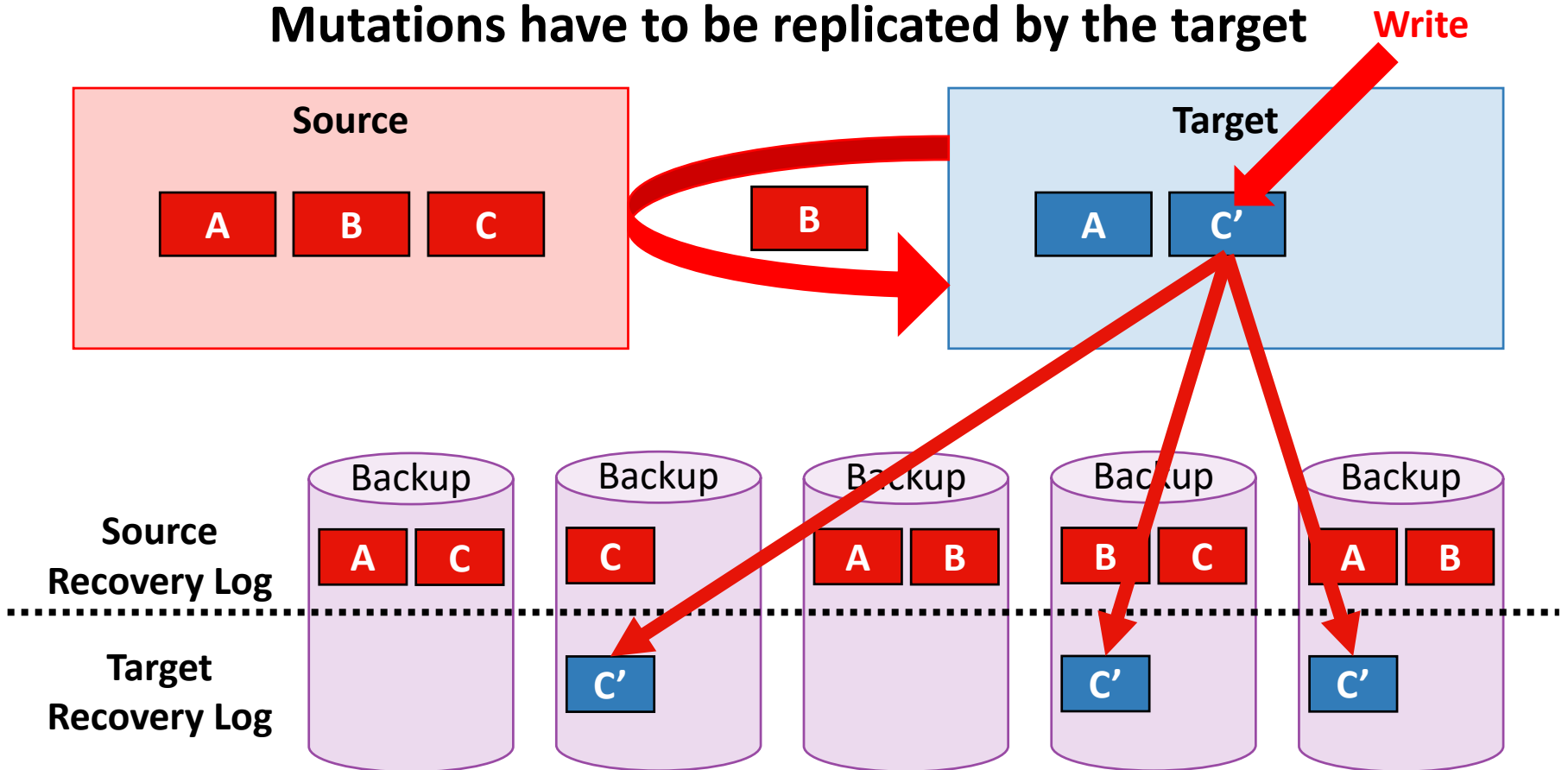**Synchronous replication hits migration speed by 34%**

Rocksteady: Safely Defer Replication At The Target

Replicate at Target only after all data has been moved over

Writes/Mutations Served By Target

Mutations have to be replicated by the target

Write

Source

A B C

B

Target

A C'

Backup | Backup | Backup | Backup | Backup

Source Recovery Log

A C | C | A B | B C | A B

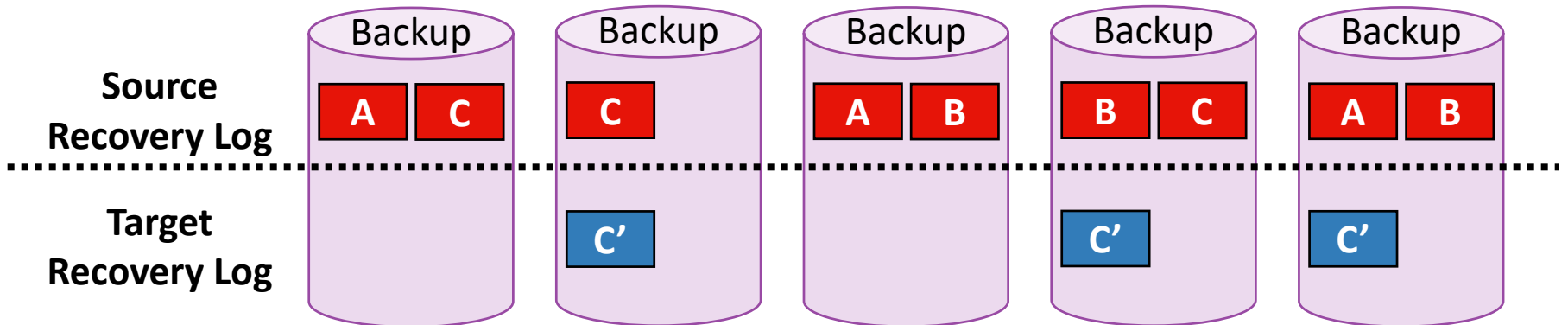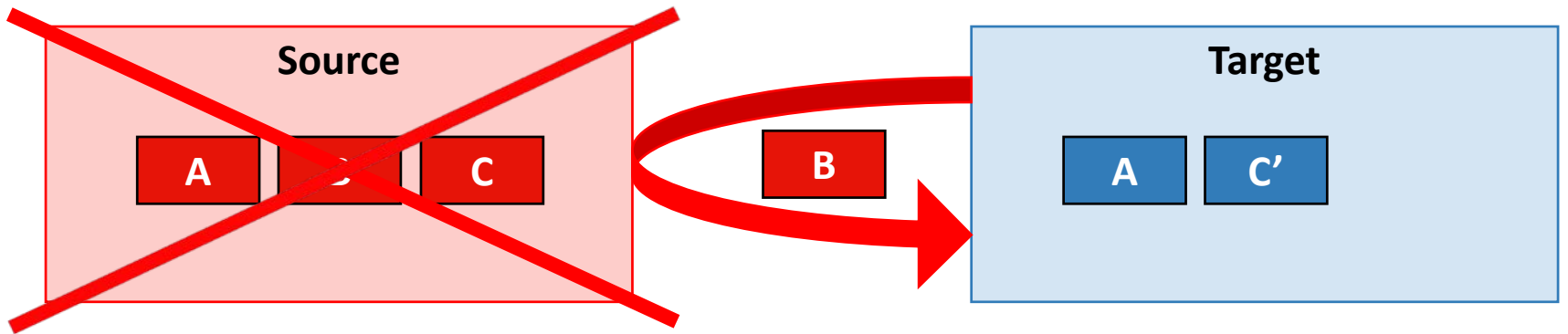**Target Recovery Log**

C' | C' | C'

Crash Safety During Migration

- **Need both Source and Target recovery log for data recovery**
  - Initial table state on Source recovery log
  - Writes/Mutations on Target recovery log

- **Transfer ownership back to Source in case of crash**
  - Migration cancelled
  - Recovery involves both recovery logs

- **Source takes a dependency on Target recovery log at migration start**
  - Stored reliably at the cluster coordinator
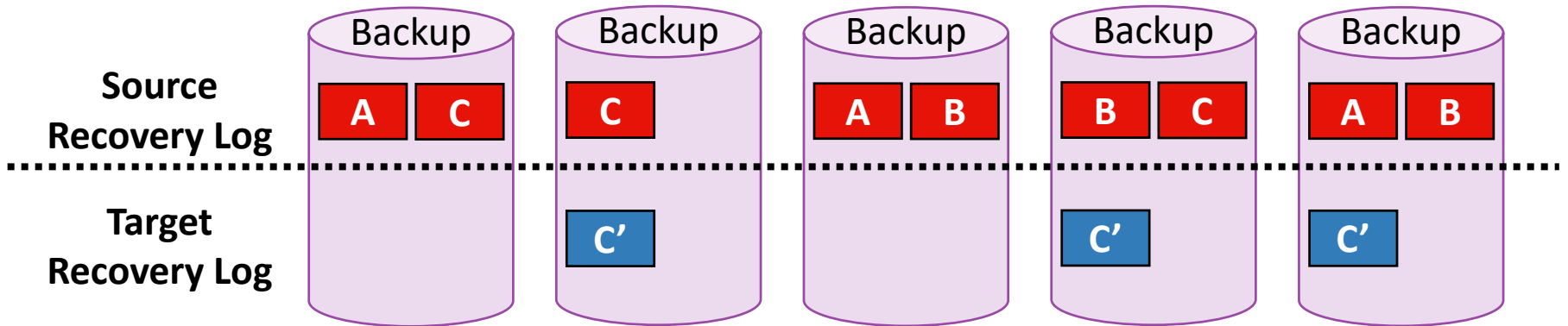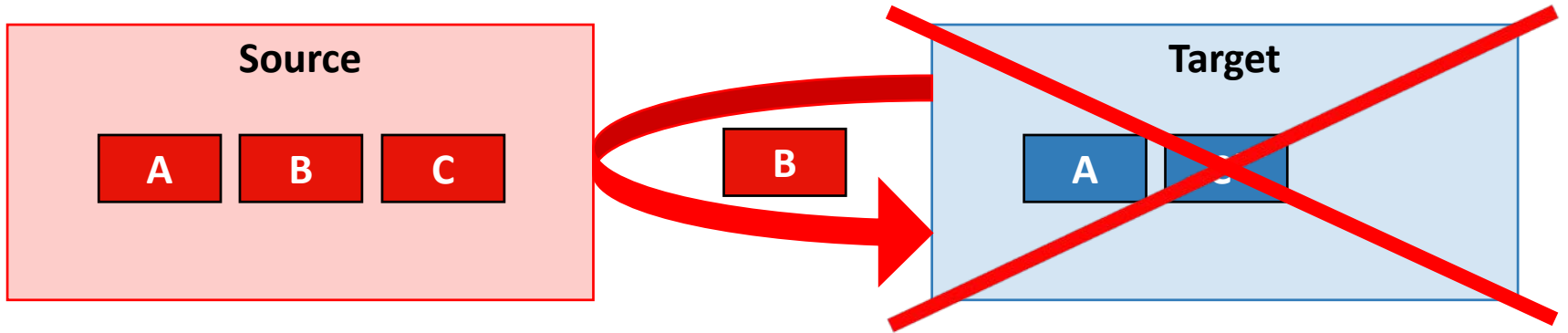 - Identifies position after which mutations present

If The Source Crashes During Migration

Recover Source, recover from Target recovery log

If The Target Crashes During Migration

Recover from Source and Target recovery log, recover Target

Crash Safety During Migration

- **Need both Source and Target recovery log for data recovery**
  - Initial table state on Source recovery log

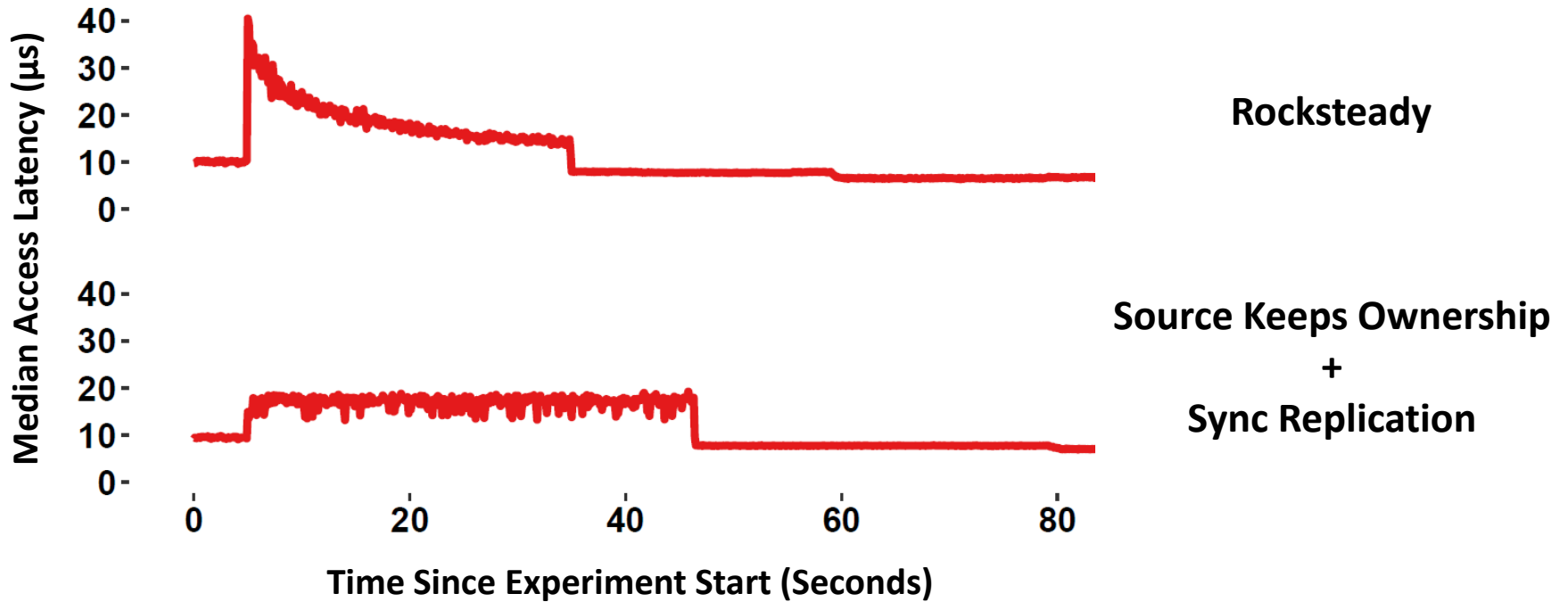**Safely Transfer Ownership At Migration Start**

Safely Delay Replication Till All Data Has Been Moved

migration start
  - Stored reliably at the cluster coordinator
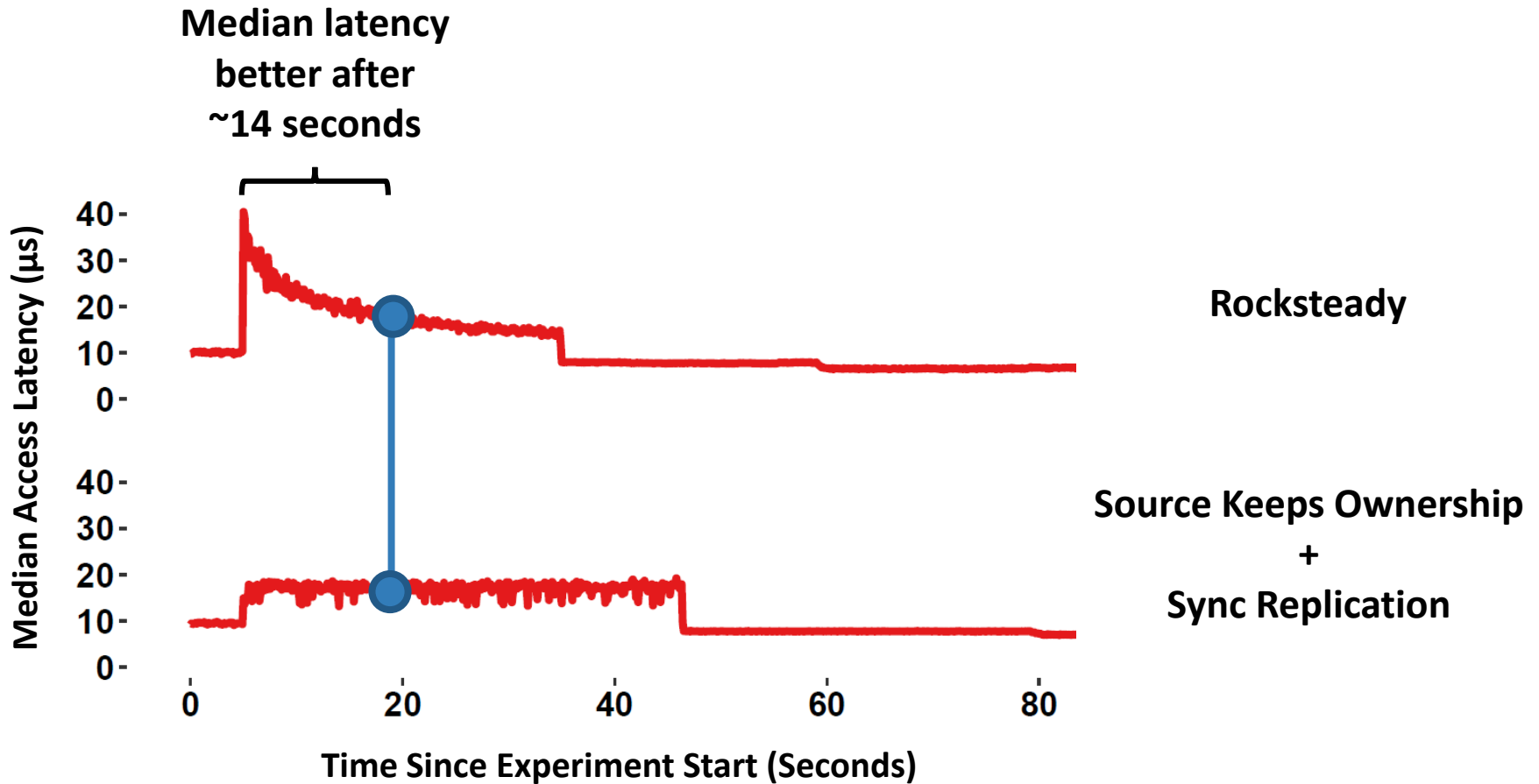 - Identifies position after which mutations present

Performance of Rocksteady

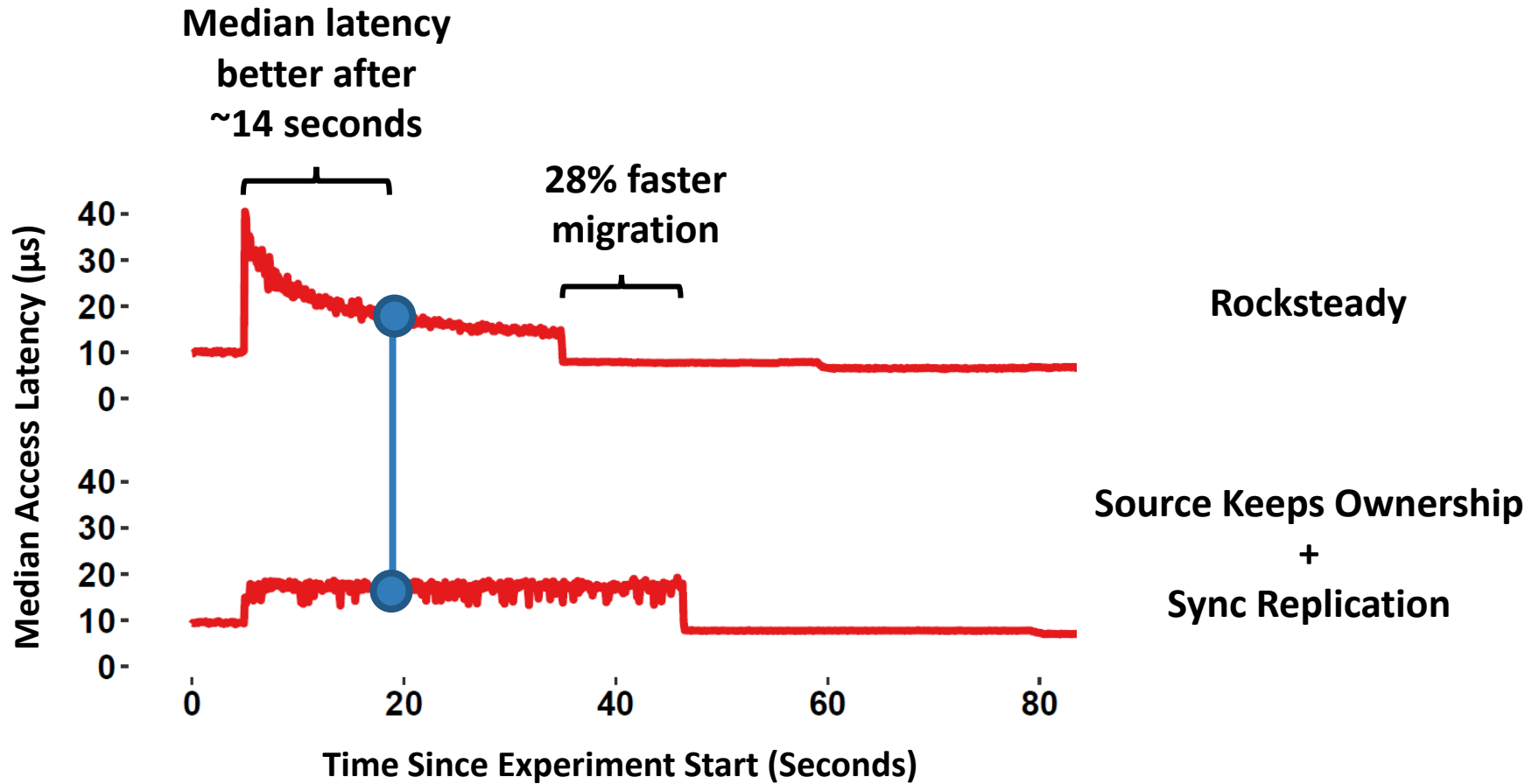**YCSB-B, 300 Million objects (30 B key, 100 B value), migrate half**

Performance of Rocksteady

YCSB-B, 300 Million objects (30 B key, 100 B value), migrate half

Performance of Rocksteady

YCSB-B, 300 Million objects (30 B key, 100 B value), migrate half

Median latency better after ~14 seconds

28% faster migration

Rocksteady

Source Keeps Ownership + Sync Replication

Median Access Latency (μs)

Time Since Experiment Start (Seconds)

Related Work

- **Dynamo:** Pre-partition hash keys

- **Spanner:** Applications given control over locality (Directories)

- **FaRM and DrTM:** Re-use in-memory redundancy for migration

- **Squall:** Reconfiguration protocol for H-Store
  - Early ownership transfer
  - Paced background migration
  - Fully partitioned, serial execution, no synchronization
    - Each migration pull stalls execution
 - Synchronous replication at the target

Conclusion

- Distributed low-latency in-memory key-value stores are emerging
  - Predictable response times ~**10 µs median, ~60 µs 99.9th-tile**

- **Problem:** Must migrate data between servers
  - Minimize performance impact of migration → **go slow?**
  - Quickly respond to hot spots, skew shifts, load spikes → **go fast?**

- **Solution: Fast** data migration with **low impact**
  - **Leverage skew**: Transfer ownership before data, move hot data first
  - Low priority, parallel and adaptive migration

- **Result:** Migration protocol for RAMCloud in-memory key-value store
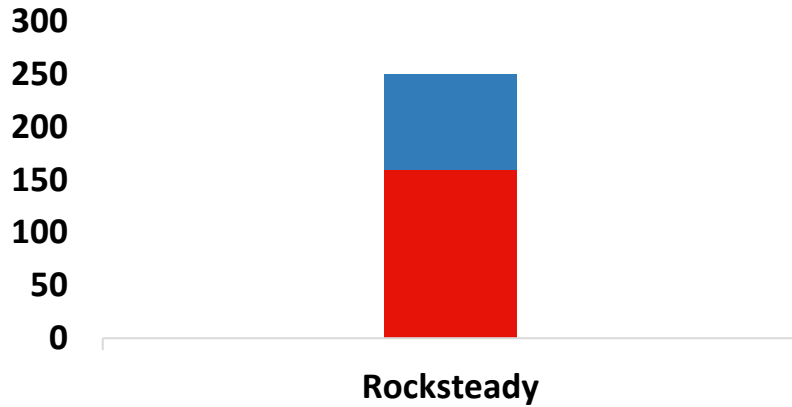 - Migrates at **758 MBps** with 99.9th-tile latency **< 250 µs**

Source Code: https://github.com/utah-scs/RAMCloud/tree/rocksteady-sosp2017

Backup Slides
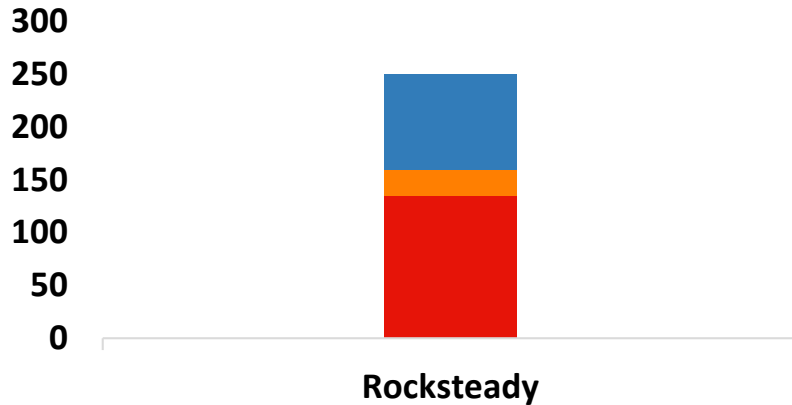
Rocksteady Tail Latency Breakdown
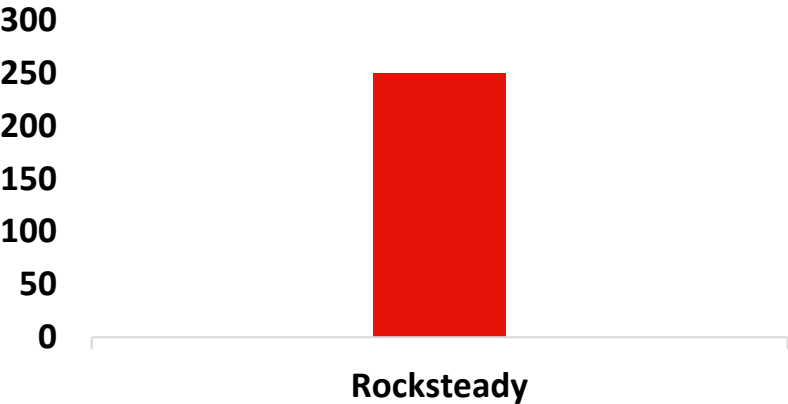
Rocksteady Tail Latency Breakdown

- Disabling parallel pulls brings tail latency down to 160 μsec

Rocksteady Tail Latency Breakdown

- Disabling parallel pulls brings tail latency down to 160 μsec

- Synchronous on-demand pulls further brings tail latency down to 135 μsec