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 $\sim 10 \, \mu s$ median, $\sim 60 \, \mu s$ 99.9th-tile

- **Problem:** Must migrate data between servers
  - Minimize performance impact of migration $\rightarrow$ go slow?
  - Quickly respond to hot spots, skew shifts, load spikes $\rightarrow$ 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 \, \mu s**
  - Median latency recovers from **40 \, \mu s to 20 \, \mu s in 14 s**
Why Migrate Data?

Poor spatial locality $\rightarrow$ High multiGet() fan-out $\rightarrow$ More RPCs
Migrate To Improve Spatial Locality

Client 1

Client 2

multiGet( A B )

multiGet( C D )

Server 1

Server 2

A
B
C
D

No Locality
Fanout=7

6 Million
Spatial Locality Improves Throughput

Better spatial locality → Fewer RPCs → Higher throughput
Benefits multiGet(), range scans
The RAMCloud Key-Value Store

Data Center Fabric

All Data in RAM
Kernel Bypass/DPDK
10 µs reads
The RAMCloud Key-Value Store
The RAMCloud Key-Value Store

Data Center Fabric

Client

Coordinator

Client

Write RPC

1x in DRAM
3x on Disk

Master
Backup

Data Center Fabric

Master
Backup

Master
Backup

Master
Backup
Fault-tolerance & Recovery In RAMCloud

Data Center Fabric

Coordinator

Client

Master
Backup

Master
Backup

Master
Backup

Master
Backup
Fault-tolerance & Recovery In RAMCloud

Data Center Fabric

Coordinator

Client

Client

Client

Client

2 seconds to recover
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

![Diagram showing flow of clients and servers]

- **Client 1**
- **Client 2**
- **Client 3**
- **Client 4**

**Source Server**

**Target Server**
Rocksteady Overview: Early Ownership Transfer

**Problem:** Loaded source can bottleneck migration

**Solution:** Instantly shift ownership and all load to target

![Diagram of server and client connections](image)

- **Source Server**
- **Target Server**
- **Client 1**
- **Client 2**
- **Client 3**
- **Client 4**

- Instantly Redirected
- 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

![Diagram](image-url)
Rocksteady Overview: Leverage Skew

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

**Solution:** On demand migration of unavailable data

- **Client 1**
- **Client 2**
- **Client 3**
- **Client 4**

**Diagram:**
- Source Server to Target Server
- Hot keys move early
- Median Latency recovers to 20 µs in 14 s
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

![Diagram of Rocksteady](image-url)
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

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

150 Million Records
22.5 GB

150 Million Records
22.5 GB

Target Server

150 Million Records
22.5 GB
Instantaneous Ownership Transfer

Before migration: Source over-loaded, Target under-loaded
Ownership transfer creates Source headroom for migration

- Source CPU Utilization:
  - Before Ownership Transfer: 80%
  - Immediately After Transfer: 25%
  - Created 55% Source CPU Headroom
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

After ownership transfer, hot keys pulled on-demand

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

Before Migration:
Median=10 µs
99.9th = 60 µs

<table>
<thead>
<tr>
<th>Skew Type</th>
<th>99.9th Latency</th>
<th>Median Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform (Low)</td>
<td>240µs</td>
<td>75µs</td>
</tr>
<tr>
<td>Skew=0.99</td>
<td>245µs</td>
<td>28µs</td>
</tr>
<tr>
<td>Skew=1.5 (High)</td>
<td>155µs</td>
<td>17µs</td>
</tr>
</tbody>
</table>
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.
Naïve Fault Tolerance During Migration

Migrated data needs to be triplicated to target’s recovery log
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

Source
A B C

Target
A B C

Write

Backup
A C

Backup
C

Backup
A B

Backup
B C

Backup
A B

Backup
C'

Source Recovery Log
A C

Target Recovery Log
C'

Recovery Log
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

Source

A  B  C

Target

A  C'

Backup

A  C

Backup

C

Backup

A  B

Backup

B  C

Backup

A  B

Backup

C'

Backup

C'

Backup

C'
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

Median latency better after ~14 seconds

Source Keeps Ownership + Sync Replication
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

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 \( \sim 10 \ \mu s \) median, \( \sim 60 \ \mu s \) 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 \(< 250 \ \mu s\)

**Source Code:** https://github.com/utah-scs/RAMCloud/tree/rocksteady-sosp2017
Backup Slides
Rocksteady Tail Latency Breakdown

![Rocksteady Tail Latency Breakdown Chart]
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