RE-Store: Reliable and Efficient KV-Store with Erasure Coding and Replication

Yuzhe Li$^1$, Jiang Zhou$^1$, Weiping Wang$^1$, and Yong Chen$^2$

$^1$Institute of Information Engineering, Chinese Academy of Sciences
$^2$Texas Tech University Lubbock, Texas, USA

IEEE CLUSTER 2019
In-memory KV-stores

- KV-stores have been widely used in large-scale in-memory systems:
  - Data cache (e.g., Memcached in Facebook)
  - In-memory database

- Challenges for data availability:
  - In-memory failover is time-consuming
  - Larger systems have higher failure rates

- Common approach to achieve reliability:
  - Replication
  - Erasure coding
Replication

- **Advantage**
  - Fast data recovery

- **Disadvantage**
  - Space-consuming. (Need N times extra memory to tolerate N failures)
Erasure Coding

- **Advantage**
  - Space-efficient.

- **Disadvantage**
  - Lengthy recovery. (A lot of computing work and data transfer during data recovery)

![Diagram of Erasure Coding](image)

Li et al. (IIE, CAS)
RE-Store: Reliable and Efficient KV-Store with Erasure Coding
Cluster 2019 4 / 22
Our goal is to combine replication and erasure coding to achieve both efficiency and reliability.

To achieve this goal, we must deal with the following 3 challenges:
- Performance issues.
- Load imbalance.
- Data recovery mechanism for both replication and erasure coding.

We propose RE-Store, an in-memory key/value store system with a hybrid replication/erasure coding scheme, which solves the above three challenges.
The basic component in RE-Store is process group.

Each group comprises:
- K primary processes
  - for storing original data
- K secondary processes
  - for storing replica data
- M parity processes
  - for storing parity data

Metadata is backed up using only replication for better efficiency.
Asynchronous Update

Performance issues (1st challenge of hybrid scheme):
- In order to keep data consistent, an update has to be applied in the primary process, secondary process, and parity process.
- The nature solution (e.g., two-phase commit protocol) will cause a high latency overhead as the primary process must wait for all the update requests to finish.

RE-Store uses an asynchronous update mechanism to solve this problem.
Asynchronous Update

Client

Primary Process

Secondary Process

Parity Process

i_th Update Request

i_th Update Request

Patch i_th Diff

Apply i_th Update

Apply i-1_th Update

Apply i_th Update

Apply i_th Update

i_th ACK

i_th ACK

i+1_th ACK

i+1_th ACK

i+1_th Update Response

i+1_th Update Response

Apply i+1_th Update

Apply i+1_th Update

Apply i+1_th Update

Apply i+1_th Update

i+1_th Update

i+1_th Update

i+1_th Update

i+1_th Update

Li et al. (IIE, CAS)

RE-Store: Reliable and Efficient KV-Store with Erasure Coding and Replication

Cluster 2019 8 / 22
Load imbalance (2nd challenge of hybrid scheme):
- The secondary and parity processes only need to participate in put/set operations.
- They will become idle for read-dominant workloads, while busy for read-write/write-dominant workloads.

RE-Store uses a staggered layout to solve this problem.
Staggered Layout

- In traditional layout, different processes run on different nodes.
- In staggered layout, each node runs primary processes, secondary processes, and parity processes so that all nodes will equally utilized to service both update-dominant and read-dominant workloads.

(a) Traditional layout

(b) Staggered layout
Data recovery (3rd challenge of hybrid scheme):
- An error can occur at any location, including the original data, parity data, and replica data.
- We need to consider recovery mechanisms for both replication and erasure coding.

In RE-Store, a fast data recovery strategy is strongly desired under different scenarios.
• Single failures: primary failure, secondary failure, and parity failure.
• For each failure, we assume that a recovery process (RP) is started on a new node for recovery.

(a) Primary Failure
(b) Secondary Failure
(c) Parity Failure
During the recovery of a single-process failure, RE-Store will not block key/value operations.

For get/read operations, client requests will be handled by the primary process or the corresponding secondary process if a primary failure occurs.

For update/write operations, there are two situations to be considered.

- If the corrupted data block has been recovered, update operations will be applied to it normally.
- Otherwise, the corrupted data block will be repaired first and then update operations will be allowed.
Data Recovery

- Two-process failures: primary-secondary failures and other cases of failure.
- Other cases of failures: similar to single-process failure.
- Primary-secondary failures: only cases that require the erasure coding to participate in data repair.
Evaluation

- 5-node cluster for evaluation
  - 4 nodes for server and 1 node for client
  - Each node is composed of four 8-core 2.2GHz AMD 6132HE processors, 64GB memory, and one gigabit Ethernet card

- 3 process groups in RE-Store
  - Each process group consists of 3 data processes, 3 secondary processes, and 1 parity process

- 3 approaches to compare with:
  - replication, erasure coding and vanilla Memcached
  - We deploy Cocytus as an erasure coding system
Memory Consumption

(a) Memory consumption under different value sizes

(b) Memory consumption for different fault tolerance number
Performance

(a) Throughput($r:w = 50:50$)

(b) Throughput($r:w = 95:5$)
Recovery Efficiency

(a) Single-node failure

(b) Two-node failure
Coding Schemes

![Bar chart showing memory usage for different value sizes and erasure coding schemes.]

- **RE(3,3,1)**
- **RE(5,5,1)**
- **RE(8,8,1)**

Memory (GB) vs. Value Size:
- 1KB
- 2KB
- 4KB
- 8KB

Li et al. (IIE, CAS)
RE-Store: Reliable and Efficient KV-Store with Erasure Coding
Cluster 2019
Related Work

- Replication
  - Copysets, Scarlett, Mojim

- Erasure Coding
  - Xorbas, lazy recovery, EC-Store

- Hybrid scheme of replication and erasure coding
  - Cocytus, REC2, Ring
Conclusion

- RE-Store uses a new, hybrid scheme to combine space-efficiency advantages of erasure coding and the recovery-efficiency of replication.

- RE-Store can save 18%-34% memory consumption compared to the replication scheme and is able to achieve better recovery performance than erasure coding.
Thank You!