Fast Crash Recovery in RAMCloud

Diego Ongaro, Stephen M. Rumble, Ryan Stutsman, John Ousterhout, and Mendel Rosenblum

Presented by Jian Guo and Zhehao Li
How to build persistent memory storage?
Background: How to combine RAM and Disk?

- How to remember information to disk?
  - Backup battery
  - Other magical hardware

- How to recover data from disk to RAM?
  - Online replication
  - Fast crash recovery
Background: Existing memory storage system

- **Memcached**
  - DRAM-based, temporary cache
  - Low latency & Low availability

- **Bigtable**
  - Disk-based, cache of GFS
  - High latency & High availability
RAMCloud: Design Goals

- Persistence: 1 copy in DRAM + n backups in disks
- Low latency: 5-10 µs remote access
- High availability: fast crash recovery in 1-2 seconds

Capacity: 300GB (typical back to 2009)
Requirement: Infiniband network
RAMCloud: Architecture

- Data model: key-value store
- Architecture: primary/backup + coordinator
- Persistence: 1x memory + Nx disk
Problem 1: How to get low latency and persistence?
Problem 1: How to get low latency and persistence?

- Asynchronous write
- Batched write
- Sequential write
Pervasive Log Structure

- Treat both memory and durable storage as an append-only log
- Backups buffer update to avoid synchronous disk writes
- Hash table for random access support
Pervasive Log Structure

- Only wait for backup to buffer write in DRAM
Pervasive Log Structure

- Bulk writes in background
Pervasive Log Structure

- Hash table: (key, location)
Problem 2: How to use full power for fast recovery?
Problem 2: How to use full power for fast recovery?

Scale up!*

* Terms and conditions apply:
** Actually they only have 60 machines
** Actually they used Infiniband to get 5us latency and full bidirectional bandwidth
Goal for Recovery

- Desired data size: 64GB
- Desired timeframe: 2s

However...

- Disk: 100MB/s $\rightarrow$ 10 min
- Network: 10Gbps $\rightarrow$ 1 min
Scattered Backup

- Divide log into segments, scatter across servers

Read logs from backup in parallel
Partitioned Recovery

- Partition missing key ranges, assign to recovery masters.

- Recover on one master: 64GB / 10Gb/second \( \approx 60 \) seconds

- Spread work over 100 recovery masters: 60 seconds / 100 masters \( \approx 0.6 \) seconds

Recover to hosts in parallel
Partitioned Recovery

- Masters periodically calculate partition lists and send to coordinator
- Coordinator sends partition to backup and recovery master
Recovery Flow

1. Read disk
2. Divide segment data
3. Transfer data to masters
4. Add objects to hash table and log
5. Replicate log data to backups
6. Write segment replicas to disk

Backups report its masters and send logs
Problem 3: How to avoid bottlenecks in recovery?
Potential Bottlenecks

- **Straggler**
  - Balance the load among recovery masters and backups

- **Coordinator**
  - Rely on local decision-making techniques
Balancing Recovery Master Workload

- Each master profiles the density of key ranges
  - Data is partitioned based on key range
  - Balance size and # objects in each partition
  - Local decision making
Balancing Backup Disk Reads

Solution:

- Masters scatter segments using knowledge of previous allocation & backup speed
- Minimize worst-case disk read time
- Local decision making
Evaluation
## Evaluation Setting

### Cluster Configuration

<table>
<thead>
<tr>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>60 Machines</td>
</tr>
<tr>
<td>2 Disks per Machine (100 MB/s/disk)</td>
</tr>
<tr>
<td><strong>Mellanox Infiniband HCAs (25 Gbps, PCI Express limited)</strong></td>
</tr>
<tr>
<td>5 Mellanox Infiniband Switches</td>
</tr>
<tr>
<td>Two layer topology</td>
</tr>
<tr>
<td>Nearly full bisection bandwidth</td>
</tr>
</tbody>
</table>

Not common setting in data centers
Eval1: How much can a master recover in 1s?

- 400MB/s ~ 800MB/s
- Slower if with 10Gbps Ethernet (300MB/s)
Eval2: How many disks needed for a master?

Optimal: 6 disks / recovery master
Eval3: How well does recovery scale? (Disk-based)

- 600MB in 1s with 1 master + 6 disks
- 11.7GB in 1.1s with 20 masters + 120 disks
- 13% longer
Eval3: How well does recovery scale? (Disk-based)

Total recovery time tracks straggling disk
Eval3: How well does recovery scale? (SSD-based)

- 1.2GB in 1.3s with 2 masters + 4 SSDs
- 35GB in 1.6s with 60 masters + 120 SSDs
- 26% longer
Eval4: Can fast recovery improve durability?

- RAMCloud: 0.001% / y
- GFS / HDFS: 10% / y
Conclusion and Future Work
Conclusion: Fast Crash Recovery in RAMCloud

- Pervasive log structure ensures low latency with durability
- Scattered backup & partitioned recovery ensure fast recovery
- Result:
  - 5-10 µs access latency
  - Recover 35GB data in 1.6s with 60 nodes
Potential Problems

- Scalability is skeptical for larger scale
- Recovery process could ruin locality
- Fast fault detection precludes some network protocols
Future work on RAMCloud

For an overview of RAMCloud, see [TOCS paper]. Here are a few of the interesting projects we have undertaken as part of building RAMCloud:

- **Fast crash recovery.** To minimize DRAM cost, RAMCloud keeps only a single copy of data in DRAM, with multiple copies on secondary storage. In order to avoid long gaps in availability, it harnesses hundreds of servers working concurrently to recover lost data within a few seconds after a crash. [SOSP2011 paper] [Ryan Stuttsman PhD thesis]

- **Log-structured memory.** RAMCloud manages in-memory storage using an approach similar to that of log-structured file systems. This allows RAMCloud to use DRAM twice as efficiently as traditional storage allocators such as malloc. [FAST 2014 paper] [Steven Rumble PhD thesis]

- **Rules-based programming.** RAMCloud contains several modules that must manage distributed resources in a concurrent and fault-tolerant fashion (DFCT modules). After struggling to write these modules, we discovered that a rules-based approach works well for them. In the rules-based approach, the algorithm is decomposed in a collection of rules, each of which makes incremental progress towards a goal. [USENIX ATC 2015 paper]

- **Implementing linearizability.** One of RAMCloud’s goals is to provide a high level of consistency. We developed a general-purpose infrastructure for implementing linearizability, which provides exactly-once semantics for RPCs even in the face of crashes and reconfiguration. We used this to implement a high performance transaction mechanism in RAMCloud. [SOSP 2015 paper]

- **Scalable indexes.** The original RAMCloud data model consisted of a key-value store, but we extended it to provide scalable and high-performance secondary indexes. [USENIX 2016 paper] [Ankita Kejriwal PhD thesis]
Q & A
Backup