Log-structured Memory for DRAM-based Storage

Stephen Rumble, John Ousterhout

STARnet

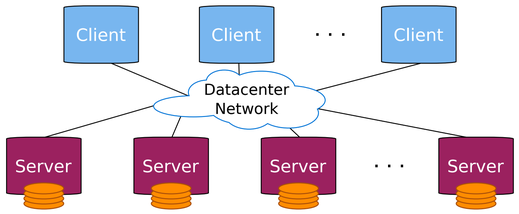
Center for Future Architectures Research

CFAR

2384.003 -- Storage

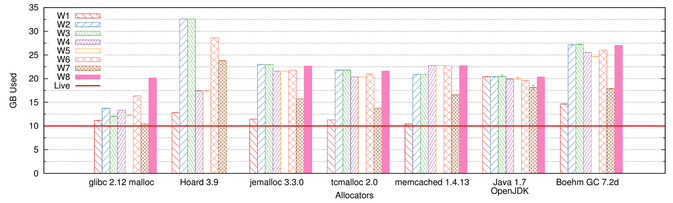
3.2: Architectures for big-data applications

RAMCloud Overview



- RAMCloud is a datacenter storage system focusing on:
  - *Large Scale*: 1,000 – 10,000+ servers
  - *Low-latency*: 5 – 10 microseconds per RPC across the datacenter
- Goal: Enable novel applications with 100 – 1,000x decrease in storage latency / increase in operations/second.
- All data stored in DRAM at all times.
- Data replicated to remote disks for durability
- Currently implements a simple key-value data model

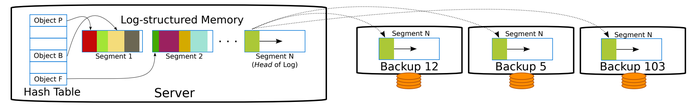
DRAM is expensive. How can we use it efficiently?



Total memory used under synthetic changing allocation patterns. Only 10GB of live data is allocated.

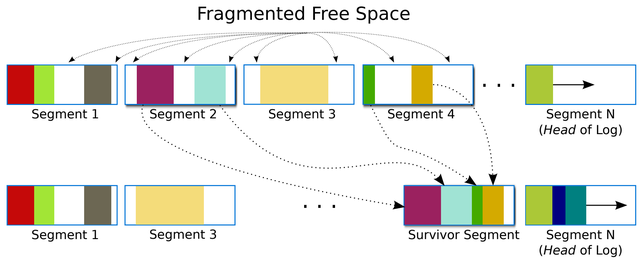
- Current memory allocators are unstable when the distribution of allocation sizes changes.
- Even copying garbage collectors that defragment memory are not designed to use that memory efficiently.
- RAMCloud needs a new memory management scheme that makes more efficient use of expensive DRAM.

Pervasive Log Structure



- Memory treated as a large contiguous array: a *log structure*
  - New and updated objects appended to end of log, replicated for durability. Same log exists in memory as on remote backup disks.
- Log split into evenly-sized *segments*
  - Scattered across backups, *cleaned* (defragmented) independently
- Hash table provides fast map from key to data in in-memory log
  - Not persistent. Rebuilt from disk log during crash recovery
- Log structure provides:
  - Memory efficiency: Trade off cleaning cost for memory utilization
  - Performance: Large disk I/Os for high bandwidth
  - Durability: Disk replication allows system to survive crashes
  - Consistency: New data is written only to the head of the log

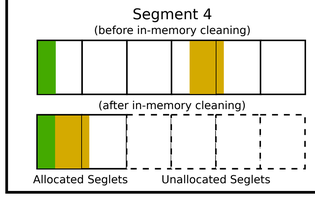
Parallel Cleaning: Minimizing Latency Impact



- When data is deleted, fragmented space accumulates in segments.
- Cleaning coalesces live data from old segments into *survivor* segments.
- Cleaned segments are then reused to store new data.
- RAMCloud's cleaner defragments in parallel with normal operation for high performance and minimal disruption of service, including concurrent writes.

Two-Level Cleaning: Reducing I/O Overhead

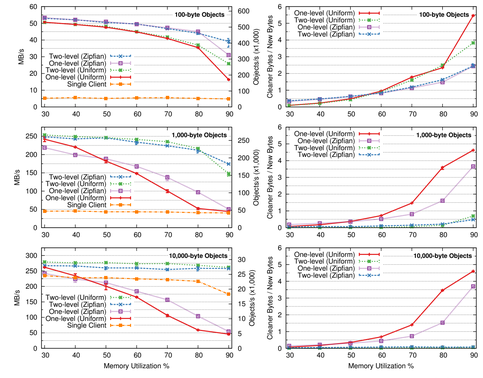
- Problem: When cleaning at high utilization, I/O overheads are large
- For example: if 90% of a segment has live data, cleaning it requires copying 9 bytes for every 1 byte freed. The survivor segment is sent over the network to multiple backup disks at significant I/O cost.
- Solution: Clean disk and memory independently, take advantage of their strengths and weaknesses:
  - DRAM has high bandwidth to absorb overheads of running at high utilization.
  - Disks have poor bandwidth, but much higher capacity.



Segment compaction reclaims space in memory without changing segments on backups.

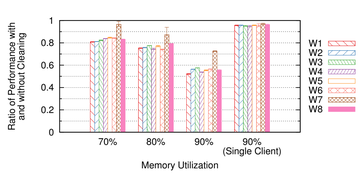
Compaction delays cleaning on disk, so disk segments drop in utilization and are cheaper to clean.

High Write Performance, Low I/O Overhead



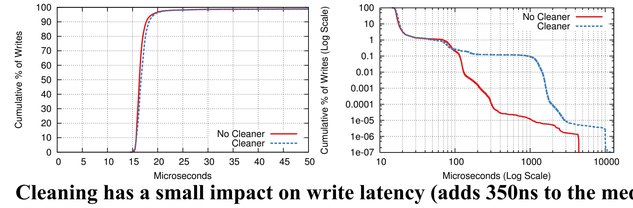
- High performance, even at high memory utilization
- Cleaning I/O overheads reduced up to 87x

High Memory Efficiency, Low Latency Impact



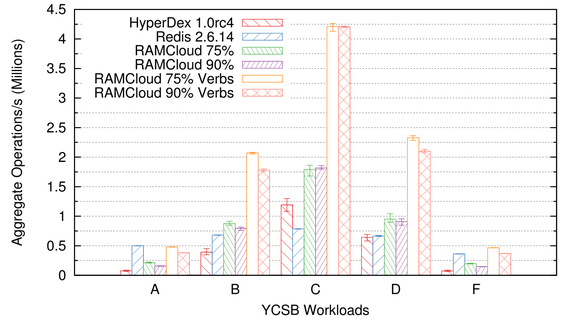
RAMCloud can run at very high memory utilization with good performance, even under changing workloads.

Users choose how to trade performance for mem efficiency.



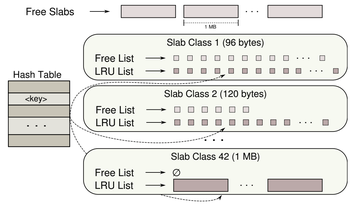
Cleaning has a small impact on write latency (adds 350ns to the median time for a small 100B write operation).

Higher Performance / Better Durability



- RAMCloud provides better durability than Redis, higher read throughput, and similar write throughput.
- RAMCloud is better than HyperDex in all workloads with similar durability.

Generality of Log-structured Memory



To show that log-structured memory can be applied beyond RAMCloud, we replaced memcached 1.4.15's slab allocator (left) with RAMCloud's log and cleaner.

Average number of objects stored per MB of cache

Allocator	Fixed 25-byte	Zipfian 0 - 8 KB
Slab	8737	982
Log	11411	1125
Improvement	30.6%	14.6%

Result:

- Up to 31% more space efficient
- No loss in write throughput
- Very low CPU overhead

Maximum Write Throughput and Cleaner Overhead

Allocator	Average Throughput (writes/sec x1000)	% CPU in Cleaning/Rebalancing
Slab	259.9 ± 0.6	0%
Log	268.0 ± 0.6	5.37 ± 0.3 %