Pelikan with ADP

Yao Yue (@thinkingfish), Twitter, Inc
Pelikan: an Open-sourced, Modular Cache
Cache @ Twitter

Clusters
>400 in prod (single-tenant)

Hosts
many thousands

Instances
tens of thousands

Job size
2-6 core, 4-48 GiB

QPS
max 50M (single cluster)

SLO
p999 < 5ms*

Protocol
Memcached, Redis/RESP, thrift, ...

Data Structure
Simple KV, counter, list, hash, sorted map...
A Modular Architecture

process_request(struct response **, struct request *);
Pelikan + Intel® Optane™ DC Persistent Memory
Intel® Optane™ DC Persistent Memory

MEMORY MODE

APPLICATION

VOLATILE MEMORY POOL

DRAM AS CACHE

OPTANE PERSISTENT MEMORY

Affordable, large volatile memory capacity
No code changes

APP DIRECT

APPLICATION

OPTANE PERSISTENT MEMORY

Large capacity persistent memory

DRAM
Pelikan with Apache Pass (AEP)

**Motivation**

Cache more data per instance
- Reduce TCO if memory-bound
- Improve hit rate

Persistent data => *warmer* cache
- Improve operations w/ graceful shutdown and faster rebuild
- Higher availability during maintenance

**Constraints**

Maintainable Changes
- Same codebase
- Non-invasive, retain high-level APIs

Operability
- Flexible invocation
- Predictable performance
Data Pool Abstraction with PMDK

Persistenc w/ libpmem (PMDK)

“cc_alloc” (malloc)
Durable Storage with DRAM Compatibility

Cuckoo

Slab

occupied unoccupied heap/array PMEM

hash table slabs
Results
Benchmark Overview

Core Parameters

**Instance density**
18-30 instances / host

**Object size**
Between 64 and 2048 bytes, step x2

**Dataset size**
Between 4GiB and 32 GiB / instance, step x2

**# of Connection per Server**
100 / 1000

**R/W ratio**
Read-only, 90/10, 80/20

*Twemcache-only for this presentation*

Focus

**common**
- Serving performance (vs. DRAM)
- Perf scalability with different dataset size

**app direct mode**
- Rebuild performance

**app direct vs memory mode**

**Lab vs data center**

**Bottleneck analysis**
Stage 1: Serving Performance (memory mode)  
(aka “does this work at all?”)

**Hardware Config (Intel lab)**
- 2 X Intel Xeon 8160 (24)
- 12 X 32GB DIMM
- 12 X 128GB AEP
- 2-2-2 config
- 1 X 25Gb NIC
- CentOS 7

**Test Config**
- 30 instances per node
- key size is 32 byte
- connection count is 100
- NUMA-aware
- 90 R / 10 W
Stage 2: Serving Performance (app direct mode)

- **Test Config**
  - 24 instances per node
  - key size is 32 byte
  - connection count is 100
  - NUMA-aware

- **Hardware Config (Intel lab)**
  - 2 X Intel Xeon 8260 (24)
  - 12 X 32GB DIMM
  - 12 X 128GB AEP
  - 2-2-2 config
  - 1 X 25Gb NIC
  - CentOS 7
Stage 2: Recovery Performance

Status Quo

Data Availability
- No redundancy in cache by default
- Some clusters are mirrored

Backfill
- Mostly rely on organic traffic
- “Bootstrapper” bounded by QPS
- Full warmup takes from minutes to days

Constraints on maintenance
- 20 minute restart interval by default
- Large clusters take days to restart

Rebuild from AEP

Single instance
- 100 GiB of slab data
- complete rebuild: 4 minutes

Concurrent
- 18 instances per host
- complete rebuild: 5 minutes

Potential impact
- Speed up maintenance by 1-2 orders of magnitude (often needs other changes)
Stage 3: Testing In-house (memory mode)

Hardware Config (Twitter DC)
- 2 X Intel Xeon 6222 (20)
- 12 X 16GB DIMM
- 4 X 512GB AEP
- 2-1-1 config
- 1 X 25Gb NIC
- CentOS 7

Test Config
- 20 instances per node
- key size is 64 byte
- value size is 256 byte
- connection count is 1000
- NUMA-aware
- read-only

throughput 1.08M QPS

p999 max = 16ms
p9999 max = 148ms
Stage 3: Testing In-house (app direct mode)

Test Config
- 20 instances per node
- key size is 64 byte
- value size is 256 byte
- connection count is 1000
- NUMA-aware
- read-only

Hardware Config (Twitter DC)
- 2 X Intel Xeon 6222 (20)
- 12 X 16GB DIMM
- 4 X 512GB AEP
- 2-1-1 config
- 1 X 25Gb NIC
- CentOS 7

SLO: p999 < 5ms

throughput 1.08M QPS

p999 max = 1.4ms
p9999 max = 2.5ms
Conclusion

App direct mode
- Changes were modest
- Can serve all data structures
- Serving performance comparable to DRAM for tested Twitter workloads
- Recovery performance was good

Memory mode
- Fully-loaded config performs like DRAM
- Less scalable w/ wimpier config

Bottleneck
- Network is still primary

Next Step

Network
- Testing in-house with ADQ

Production canary
- Will we see the same performance?
- How does larger heap affect hit rate?

Performance
- Scaling with connection counts
- Profiling, especially for memory mode
- Tuning data structure/storage design
- Testing AEP with pelikan_rds
Further Read

Pelikan
- Redis at Scale
- Caching with Twemcache
- Why Pelikan
- Pelikan Github

Cache w/ AEP
- Redis-pmem
- Memcached with pmem

Contributors

Thank you Intel Team!
Ali Alavi (@TheAliAlavi), Andy Rudoff (@andyrudoff), Jakub Schmiegel, Jason Harper, Michal Biesek, Mauricio Cuervo (@mauriciocuervo), Piotr Balcer, Usha Upadhyayula

Thank you Twitter Team!
Brian Martin (@brayniac), Kevin Yang (@kevjyang), Matt Silver (@msilver)

#collaborate