A Low Latency and Scalable Key Value Store from Modern off the shelf Components

Dan Pollack
Data Storage Science LLC
Outline

- What am I talking about?
- Why do I think it’s worth doing?
- How does it work?
- How good is it?
- What’s next?
What am I talking about?

- A low latency Key-Value store
  - Small keys/values – operations not throughput
- Two basic components
  - Software – Minio
  - Hardware – NVMeoF/TCP storage system
What am I talking about?

- Minio
  - Simple – single command line operation
  - Scalable – single or many servers in a cluster
  - Performant – GB/s Throughput / >1ms Latency
  - Manageable – Multiple control and storage APIs supported
  - Open source
What am I talking about?

- NVMeoF/TCP storage system
  - Very high performance
    - 120us latency
    - 10GBps throughput
  - Common infrastructure – No new hardware
    - Standard server system
    - Standard ethernet networking
  - Disaggregated – No trapped resources
  - Open source software components
Why do I think it’s worth doing?

- Disaggregated resources are more flexible – any resource for any workload
- TCP is everywhere and rebuilding/deploying uncommon infrastructure is unpopular – (FCOE, iSCSI, RoCE, other niche technologies)
Why do I think it’s worth doing?

- Simple software is easier to understand and use
- Key/Value stores are a specific type of object storage
- DRAM Key/Value systems are common but -
  - NVMe storage 10X - 100X lower cost
  - NVMe storage 10X - 100X higher capacity
Why do I think it’s worth doing?

- Fulfills actual need people have with minimal effort and cost
- Rapid to evaluate
How does it work?

- NVMeoF/TCP storage system provides namespaces (LUNs/Volumes) to multiple host systems
- Minio cluster running on host systems exposes the NVMe devices as storage using object APIs
- Clients make Key/Value style requests to Minio
How does it work?
How does it work?

- NVMe storage discovery – once per new device
- NVMe storage access and format
- Minio command

minio server http://host{1...4}/mount{1...2}
How does it work?

- YCSB for workload evaluation
- Native python API tools for workload evaluation
How good is it?

Native API PUT Latencies (ms)
16Byte Key - 200Byte Value

- 2-3ms: 65%
- 1-2ms: 33%
- 3-4ms: 2%
How good is it?

Native API GET Latencies (ms)
16 Byte Key - 200Byte Value

- <5ms: 30%
- 1-2ms: 38%
- 2-3ms: 27%
- 3-4ms: 3%
- 4-5ms: 3%
How good is it?

Native API Delete Latencies (ms)
16 Byte Key - 200 Byte Value

- 1-2ms: 53%
- 2-3ms: 47%
How good is it?

YCSB Load Phase Insert Latency (ms)
16 Threads

Average OPS - 849

Avg, 1.88
Min, 1.14
Max, 9.50
99th, 5.65
99.9th, 7.64
99.99th, 8.94
How good is it?

YCSB Insert Latency (ms) - 90/10 Workload
16 Threads

Average OPS - 66.05

- Max, 72.36
- 99.9th, 60.91
- 99.99th, 60.91

Latency (ms):

- Avg, 4.37
- Min, 1.56
- 99th, 9.59
- 99.9th, 16.77

2020 Storage Developer Conference. © Data Storage Science LLC All Rights Reserved.
How good is it?

YCSB Read Latency (ms) - 90/10 Workload
16 Threads

Max, 69.33
99th, 5.70
99.9th, 9.05
99.99th, 22.76

Average OPS - 592.20
How good is it?

YCSB Read Latency (ms) - All Read Workload
16 Threads

Average OPS - 640.81

Latency (ms):
- Max, 405.09
- Average, 2.55
- Min, 0.83
- 99th, 4.95
- 99.9th, 12.14
- 99.99th, 75.80
How good is it?

- Round trip network latency is .5ms
- 1ms seems to be a key target for read response time
- API reads are 38% within 2ms
- YCBS reads average slightly more than 2ms
- This is OK – Not great so tradeoffs are important
What’s next?

- NVMeoF/TCP in common usage for most NVM block IO over ethernet
- Software enhancements for greater workload suitability
- Native NVMe Key/Value per namespace command set
What’s next?

- Minio enhanced for data access
  - Versioning of stored data
  - Storage protection classes
  - Replication between instances
  - Data lifecycle management
  - Immutable data
  - SQL queries on data
Please take a moment to rate this session.

Your feedback matters to us.