Challenges in Using Persistent Memory In Distributed Storage Systems

SNIA SDC 2016

Dan Lambright
Storage System Software Developer
Adjunct Professor University of Massachusetts Lowell
Aug. 23, 2016

Overview

- Technologies
  - Persistent memory, aka storage class memory (SCM)
  - Distributed storage
- Case studies
  - GlusterFS, Ceph
- Challenges
  - Network latency
  - Accelerating parts of the system with SCM
  - CPU latency

Storage Class Memory

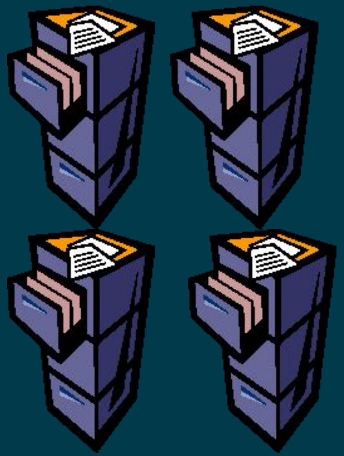
What do we know / expect?

- Near DRAM speeds
- Wearability better than SSDs (claims Intel)
- API available
    - Crash-proof transactions
    - Byte or block addressable
- Likely to be at least as expensive as SSDs
- Fast random access
- Has support in Linux

Distributed Storage

How to scale performance and capacity?

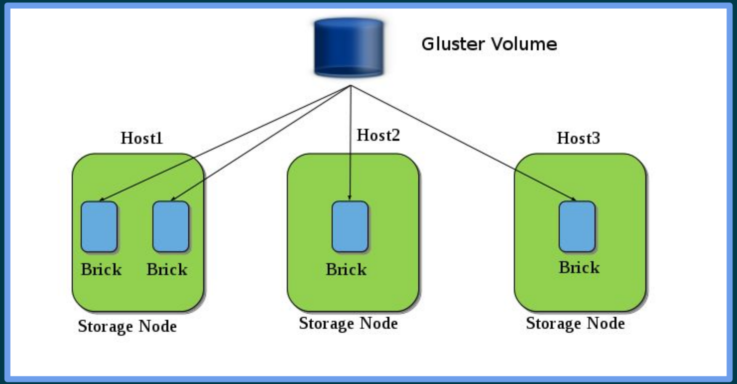
- Single server scales poorly
  - Horizontal scaling expensive
- Multiple servers in distributed storage scale well
  - Maintain single namespace
- Commodity nodes
  - Easy expansion by adding nodes
  - Good fit for low cost hardware
  - Minimal impact on node failure



Case Studies

GlusterFS

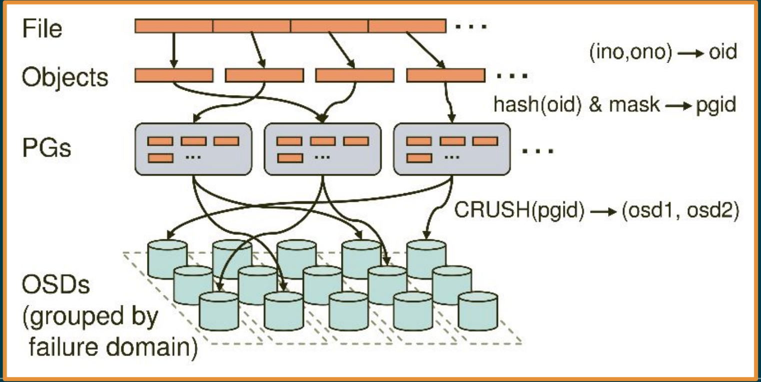
- Scale-out NAS
- Aggregates file systems (bricks) into single namespace
- Scalability limited - directories / management replicated across all nodes
    - No metadata servers



Case Studies

Ceph

- Metadata servers manage node membership
- Supports block, object, file
    - RADOS as intermediate representation adds overhead
    - Must translate ingress to: {objects, placement groups (PGs), OSDs}



The problem

Must lower latencies throughout system : storage, network, CPU

Media	Latency
HDD	10ms
SSD	1ms
SCM	< 1us
CPU (Ceph, aprox.)	~1000us
Network (RDMA)	~10-50us

Framing The Problem

How to analyze distributed storage + SCM’s benefits

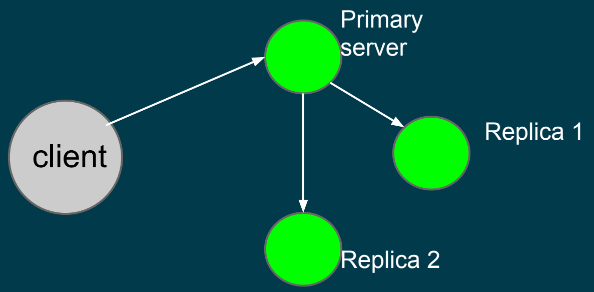
- Plethora of workloads and configurations
    - HPC, sequential, random, mixed read/write/transfer size, etc
    - # OSDs, nodes, replica/EC sets, …
- Benchmark *one*
    - 3X replication; one brick/OSD per node
    - Linux SCM support /dev/pmem
    - 14Gbps RDMA
    - Two clients 4KB/8KB random reads / writes

NETWORK LATENCY

Server Side Replication

Latency cost to replicate across nodes

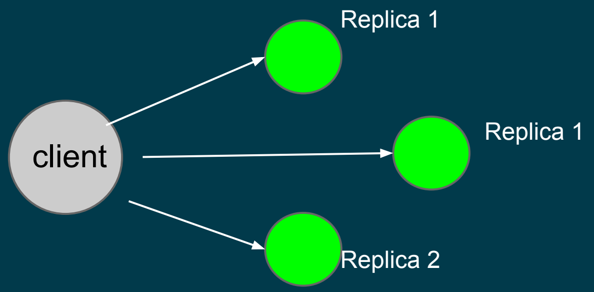
- “Primary copy” : update replicas in parallel,
  - processes reads and writes
  - Ceph’s choice, also Gluster’s “journal based replication” (under development)
- Other design options
  - Read at “tail” - the data there is always committed



Client Side Replication

Latency price lower than server software replication

- Uses more client side bandwidth
- Likely client has slower network than server.
- Gluster’s default replication strategy (AFR)



Consistency

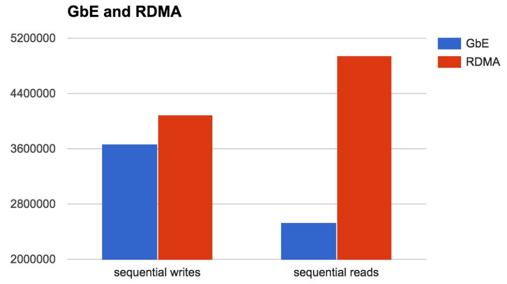
- Reads following writes
  - Read and write operations logically occur in some sequential order.
  - Completed write operations are reflected by subsequent read operations.
- In Ceph,
  - Writes to different objects (but same PG) are serialized
  - Serialized even if originated from different clients
  - There may be many PGs
  - PG size configurable online
  - (note many PGs are resource intensive)

Improving Network Latency

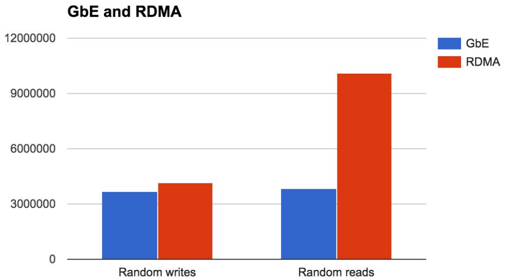
RDMA

- Avoid OS data copy; free CPU from transfer
- Application must manage buffers
  - Reserve memory up-front, sized properly
  - Both Ceph and Gluster have good RDMA interfaces
- Extend protocol to further improve latency?
  - Proposed protocol extensions, could shrink latency to ~3us.
  - RDMA write completion does not indicate data was persisted (“ship and pray”)
  - ACK in higher level protocol - adds overhead
  - Add “commit bit”, perhaps combine with last write?

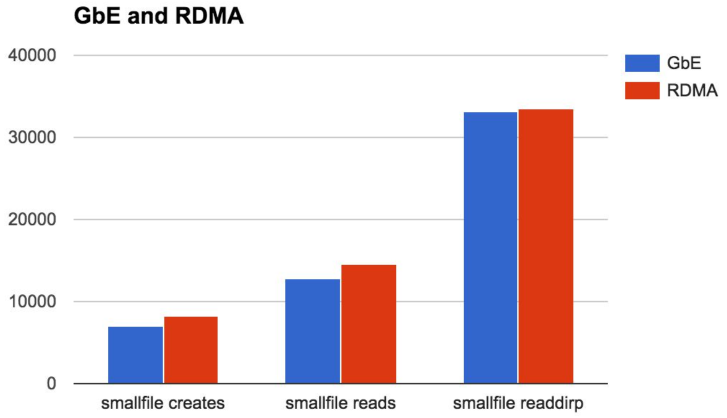
IOPS RDMA vs 10Gbps : Glusterfs 2x replication, 2 clients



Sequential I/O (1024 block size)



Random I/O (1024 block size)



1024 bytes transfers

Biggest gain with reads, little gain for small I/O.

Improving Network Latency

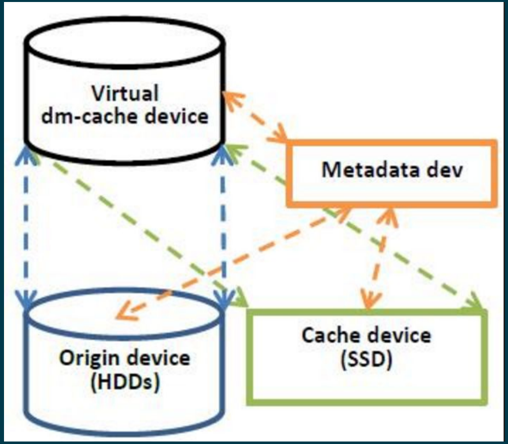
Other techniques

- Reduce protocol traffic (discuss more next section)
- Coalesce protocol operations
    - WIth this, observed 10% gain in small file creates on Gluster
- Pipelining
    - In Ceph, on two updates to same object, start replicating second before first completes

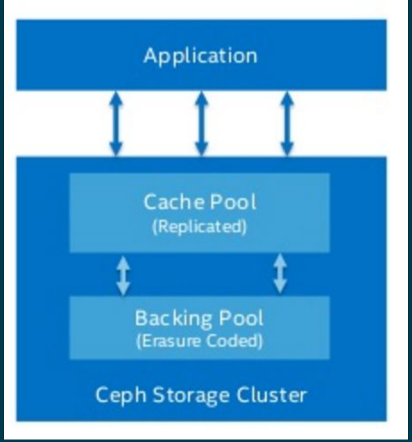
ACCELERATION

Adding SCM to Parts of System

Kernel and application level tiering



DM-cache

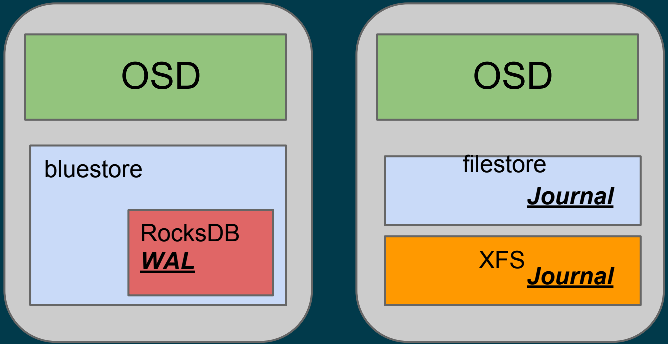


Ceph tiering

Adding SCM to Parts of System

Candidate destinations

- Ceph filestore's journal
- Ceph bluestore's - RocksDB write ahead log
- XFS journal

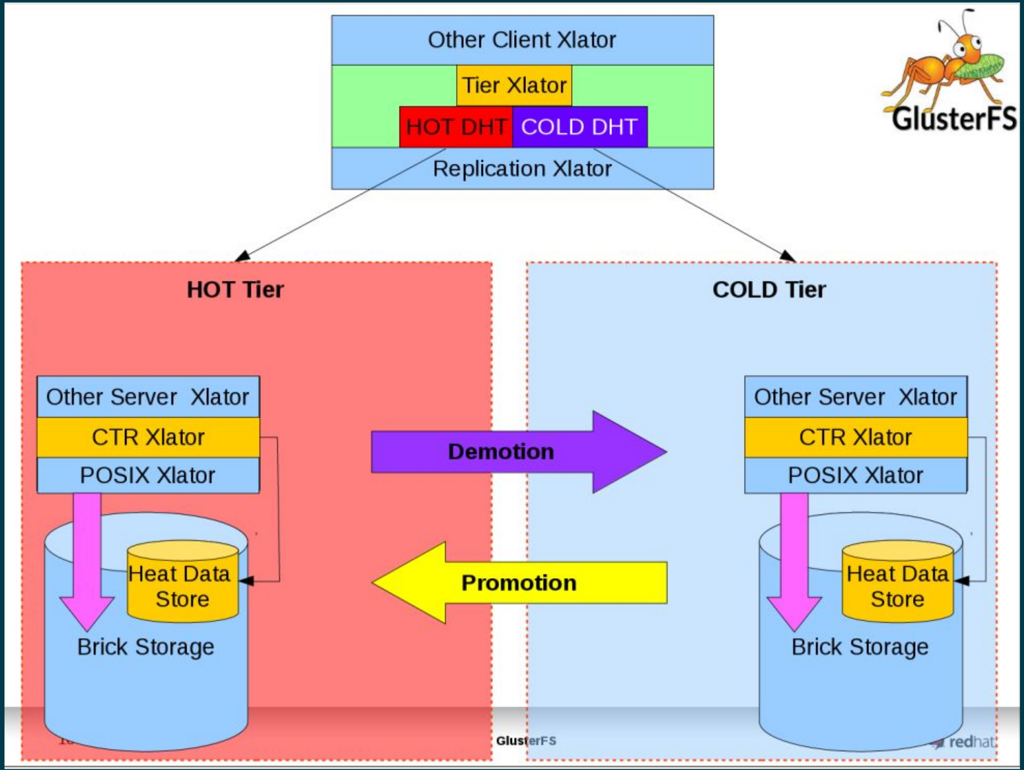


In Depth: Gluster Tiering

Illustration of network problem

- Heterogeneous storage in single volume
  - Fast/expensive storage cache for slower storage
  - Introduced in Q1 2016
  - Fast “Hot tier” (e.g. SSD, SCM)
  - Slow “Cold tier” (e.g. erasure coded)
- Policies:
  - Data put on hot tier, until “full”
  - Once “full”, data “promoted/demoted” based on access frequency

Gluster Tiering



Gluster's "Small File" Tiering Problem

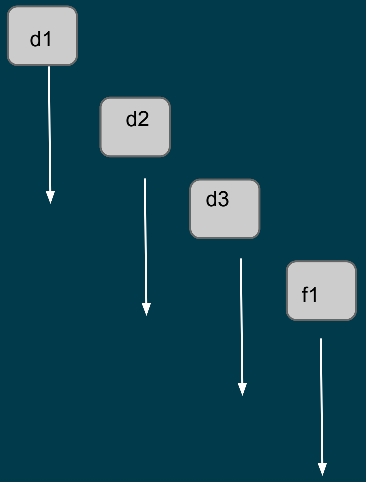
Analysis

- Tiering helped large I/Os, not small
- Pattern seen elsewhere ..
  - RDMA tests
  - Customer Feedback, overall GlusterFS reputation …
- Observed *many* "LOOKUP operations" over network
- Hypothesis: metadata transfers dominate data transfers for small files
  - small file data transfer speedup fails to help overall IO latency

Understanding LOOKUPs in Gluster

Problem : Path Traversal

- Each directory in path is tested on an open(), by client's VFS layer
  - Full path traversal
  - d1/d2/d3/f1
  - Existence
  - Permission



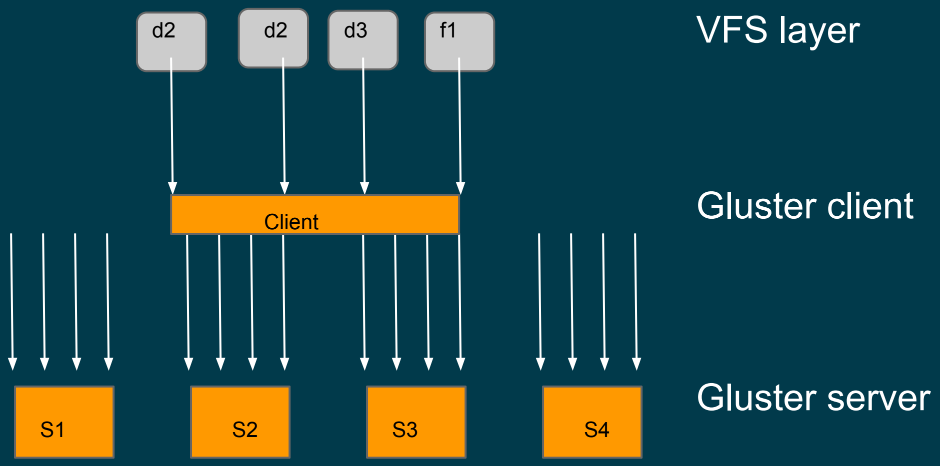
Understanding LOOKUPs in Gluster

Problem : Coalescing Distributed Hash Ranges

- Distributed hash space is split in parts
  - Unique space for each directory
  - Each node owns a piece of this “layout”
  - Stored in extended attributes
  - When new nodes added to cluster, rebuild the layouts
- When file opened, entire layout is rechecked, for each directory
  - Each node receives a lookup to retrieve its part of the layout
- Work is underway to improve this.

LOOKUP Amplification

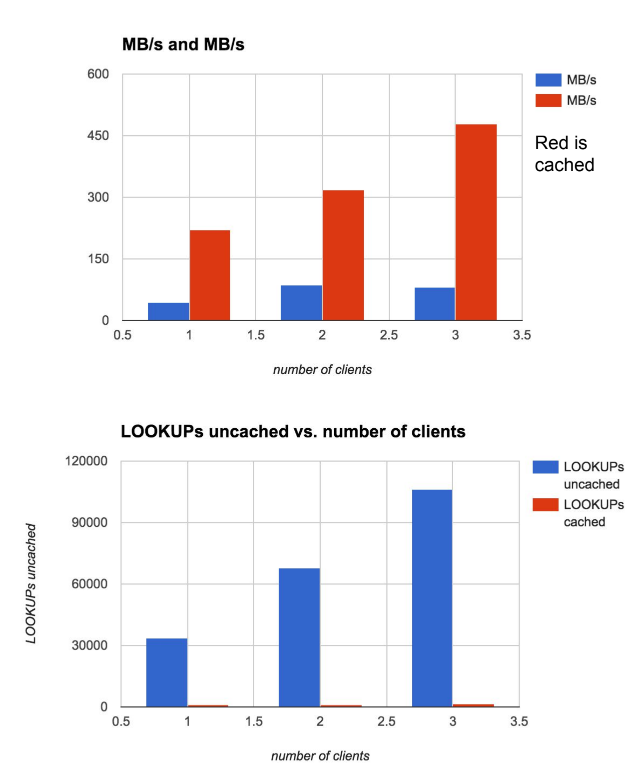
d1/d2/d3/f1
Four LOOKUPs
Four servers
16 LOOKUPs total in worse case



Client Metadata Cache

Gluster's "md-cache" translator

- Cache file metadata at client long term
  - WIP - under development
- Invalidate cache entry on another client's change
  - Invalidate intelligently, not spuriously
  - Some attributes may change a lot (ctime, ..)



CPU LATENCY

CPU Latency

Services needed to distribute storage add to CPU overhead

- It takes CPU cycles to…
    - Distribute data over nodes
    - Perform replication / ec
    - Manage a single namespace
    - Convert between external and internal representations of data

Ceph Datapath - Micro Optimizations

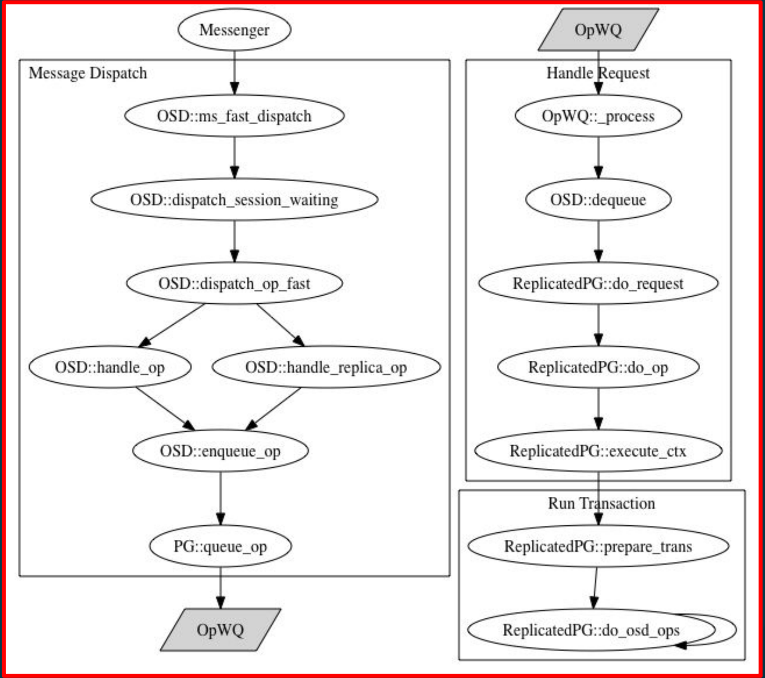
session_dispatch _lock

- Upper (fast) and lower (slow) halves of I/O path
- Context switch between halves - consider run to completion?
- Many locks taken- use lockless algorithms?
- Memory allocation matters (Jmalloc)
- Etc

pg_map_lock

PG::map_lock

OpWQ lock



Community Contributions

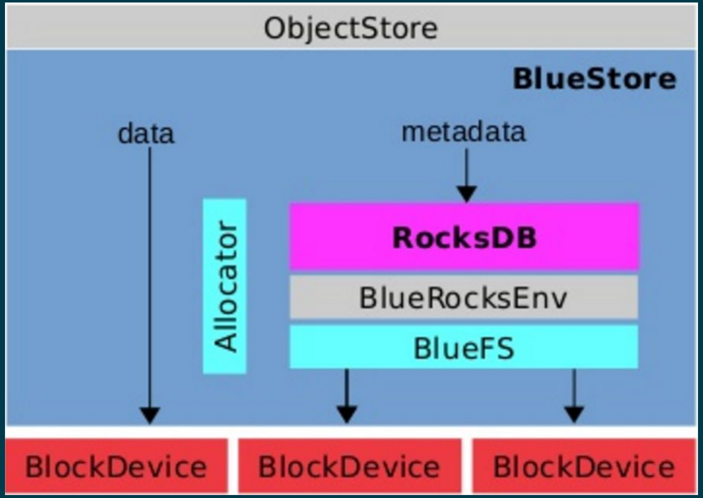
SanDisk, CohortFS, many others

- SanDisk
  - Sharded worker thread pools
  - Bluestore optimizations
  - Identified TCMalloc problems, introduced JEMalloc
  - .. much more ... ongoing
- CohortFS (now Red Hat)
  - Accelio RDMA module
  - Divide and conquer performance analysis using “infinite backend” (memstore)
  - Lockless algorithms / RCU (coming)

Ceph Datapath - Macro Optimizations

Bluestore - a key value database as backend

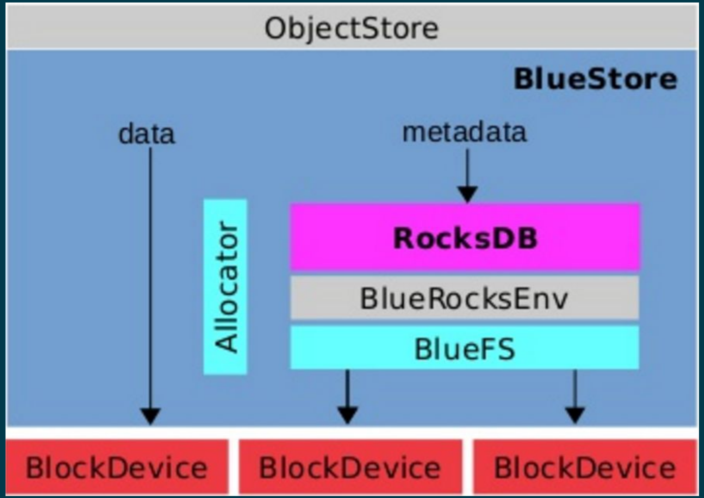
- No longer run Ceph over a file system
- Motivation
  - Transactions difficult to implement with posix
  - Writing to Ceph's journal first meant 2X writes
  - Object enumeration inefficient
- Manage metadata with a database
  - ACID semantics for transactions
  - No longer a double write



Bluestore

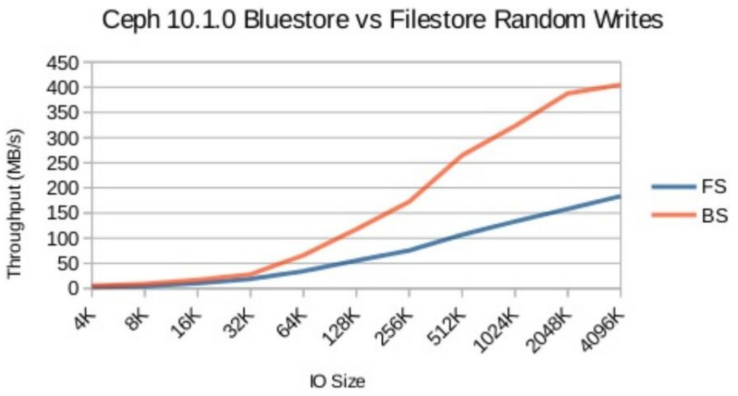
How does it help CPU latency

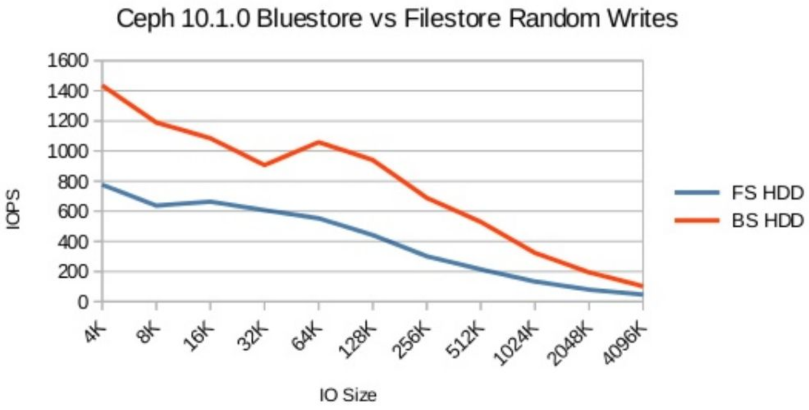
- Shorter code path
- More of stack customizable for Ceph’s needs
    - BlueFS allocates from block device



Some results (4/16)

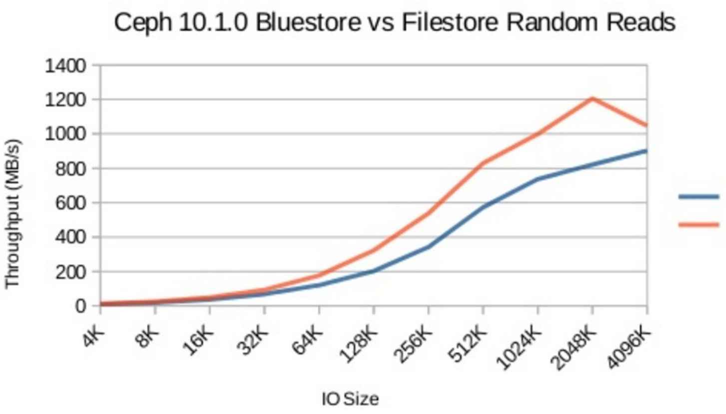
Code in flux - YMMV !

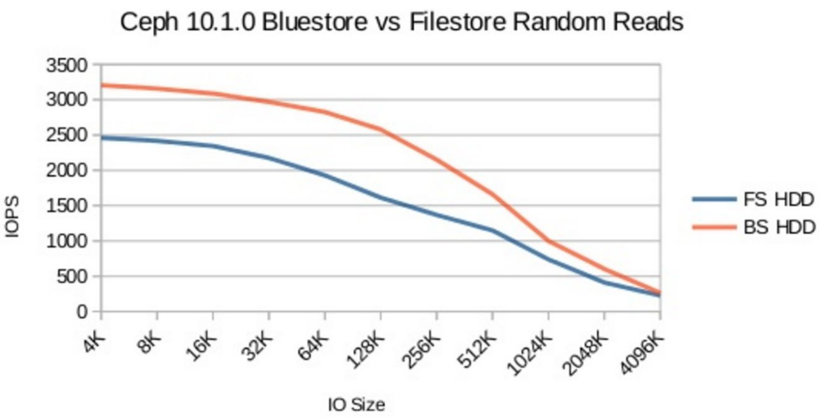




Some results (4/16)

Code in flux - YMMV !





What comes next?

Further optimizations for SCM

- RocksDB is an log structured merge-tree (LSM) database
  - Optimized for sequential access
  - Has periodic background compaction, write amplification, ...
  - Must carefully tune RocksDB “compaction” options
  - A good fit for disks, not so much for SCM
  - Use different DB, e.g. SanDisk’s ZetaScale ?
- Write to persistent memory directly

SUMMARY

Summary and Discussion

Distributed storage and SCM pose unique problems with latency.

- Network latency reductions
  - Use RDMA
  - Reduce round trips by streamlining protocol , coalescing etc
  - Cache at client
- CPU latency reductions
  - Aggressively optimize / shrink stack
  - Remove/replace large components
- Consider
  - SCM as a tier/cache
  - 2 way replication

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos