Ceph
Successes and challenges with open source distributed storage
Carlos Maltzahn, standing in for Sage Weil
DOMA Workshop, 11/16/17
Purpose of this talk

• Share important challenges
• Look forward (a little bit)
• Collect feedback
Network, Storage, & DRAM trends

Log scale

- Use DRAM Bandwidth as a proxy for CPU throughput
- Reasonable approximation for DMA and poor cache performance workloads (e.g. Storage)

Network, Storage, & DRAM trends

Linear scale

- Same data as last slide, but for the Log-impaired
- Storage Bandwidth is not literally infinite
- But the ratio of Network and Storage to CPU throughput is widening very quickly

The CPU/DRAM Bottleneck

Effects Of The CPU/DRAM Bottleneck

- Storage Cost = Media + Access + Management
- Shared nothing architecture conflates access and management
- Storage costs will become dominated by Management cost
- Storage costs become CPU/DRAM costs
ARCHITECTURAL COMPONENTS

**RGW**
A web services gateway for object storage, compatible with S3 and Swift

**RBD**
A reliable, fully-distributed block device with cloud platform integration

**CEPHFS**
A distributed file system with POSIX semantics and scale-out metadata management

**LIBRADOS**
A library allowing apps to directly access RADOS (C, C++, Java, Python, Ruby, PHP)

**RADOS**
A software-based, reliable, autonomous, distributed object store comprised of self-healing, self-managing, intelligent storage nodes and lightweight monitors
Challenges

• Current Architecture works great for multiple interfaces and directly-attached HDDs (aka spinners)

• Challenges
  • Storage devices are getting too fast
  • OSD peering too expensive for fabric-connected storage devices
  • Control of tail latency
  • Global name space scalability
    • frequent reason why people switch from files to objects
Storage devices are getting too fast

• NVMe is too fast
  • Migrating OSD critical path from thread-based to *futures*-based
  • Probably going to use Seastar library (seastar-project.org)
  • Most code doesn’t have to change, e.g. peering and consistency code
  • Will greatly reduce the CPU cost for all storage
  • Will take at least a year

• Fabric-based storage devices (NVMeoF)
  • OSD to OSD replication causes too much overhead
  • Planning a new pool type where one OSD manages all replica
  • Fabrics might be too expensive, especially if already CPU-limited
  • New device interfaces might make OSD to OSD replication affordable again
    • For example, NVMe key/value interface standardization effort
Challenges

• Tail latency control
  • Planning a new pool type for quorum-based consistency
  • Different writers to different replica in parallel with eventual consistency
  • Strong consistency: read all replica and resolve conflicts
  • Weak consistency: read one replica only

• Files vs objects
  • Objects are considered to be more scalable than files
  • Most file systems have *one* name server: that doesn’t scale
  • Many file system workloads with > 50% metadata operations
  • Luminous release introduces multiple active metadata servers
Global Name Spaces

• Multiple active name servers
  • first proposed by Sage Weil at SC04

• Challenges
  • Load balancing
  • Consistency
  • Logical names vs physical names (e.g., mapping files to objects)

Credit: http://ceph.com/community/new-luminous-multiple-active-metadata-servers-cephfs/
Global Name Spaces


SUBTREE PARTITIONING

MDS cluster

Hierarchical Namespace

RADOS

Client Perspective

Hierarchical Namespace

Composed from metadata stored in MDS and objects

Objects

MDS cluster

RADOS

Mantle

CephFS

Mantle

Hooks

rebalance

migrate?

partition

cluster

partition

namespace

send HB

recv HB

fragment

weak consistency

HDFS semantics

strong consistency

POSIX semantics

decoupled, durable
BatchFS semantics

decoupled, no durability
RAMDisk semantics

Summary

• Data management CPU overhead will dominate cost of storage
• Ceph data path will get a lot shorter, check out seastar-project.org
• Tail latency control via quorum-based consistency (new pool type)
• Global name space scalability via
  • Subtree-specific load balancing policies
  • Subtree-specific consistency semantics that can be dynamically changed
  • Decoupling logical names from physical metadata management
• Contact: Carlos Maltzahn, carlosm@ucsc.edu
Ceph Pools

- Grouping of objects into sets that differ by the following
  - Resilience (number of replicas or erasure code parameters)
  - Placement groups
  - CRUSH rules
  - Snapshots
  - Ownership
  - Future object management alternatives (see below)