Using Ceph for Mass Storage in High Energy Physics

Sébastien Ponce
sebastien.ponce@cern.ch
Our problem
The big picture

LHC

Mass Storage

Simulation

Export

Reconstruction

Analysis

Cern for Mass Storage in HEP
The big picture

LHC

Mass Storage

Simulation

Export

Reconstruction

Analysis

Cern

June 14th 2016

Ceph for Mass Storage in HEP
Anatomy of a Mass Storage System

Client

Disks

Tapes

Stager

Namespace

data

metadata

June 14th 2016 Ceph for Mass Storage in HEP
Anatomy of a Mass Storage System

- Client
- Disks
- Tapes
- Stager
- NameSpace

Data flows from Disks to Client and Tapes to Stager.
Metadata flows from Client to Stager and Stager to NameSpace.
Mass Storage in Physics specificities

- mainly used for archiving of big files
  - with tape backend, disk being a cache
  - average file size 2GB, but up to 100GB
- we are big
  - total current data volume above 150PB
  - disk cache in 10s of PB
  - constant throughput in 10s of GB/s
  - single stream throughput to tape: 400MB/s
- we have specific protocols
  - xrootd and gridFTP
Why using Ceph?
Why using Ceph?

• delegate disk management to external software
  • avoid duplication of effort
  • homogenize our solutions for different services
• benefit from new features
  • rebalancing, striping, erasure coding, ...
• improve scheduling efficiency
• solve the tape bandwidth issue
What does (did) ceph provide?

- **LibRados**
  - Object store
  - No striping
- **RadosGW**
  - S3 compatible
- **RBD**
  - Block device
- **Ceph FS**
  - Kernel code
Best candidate was librados

- Has most things we need:
  - Scalability
  - External attributes
  - Number of replicas tunable per object
  - Erasure coding was under work
  - Rebalancing, draining, easy management

- Is missing two features:
  - Striping
  - Support for big files
    - Objects should be rather small (1-100 MB max)
Striping could be added

- File is mapped to a set of objects
- Objects’ names are `<filename>_<nb>`
- Excellent performances
libradosstriper

- We added a Striper interface to Ceph
  - API very similar to Rados
  - implemented on top of Rados
  - reusing existing ceph striping code
- Available since the hammer release
  - stable from infernalis onwards
  - enhancements to come in kraken (see coming slides)
Porting code to Ceph
Interfacing our code with Ceph

• mainly change POSIX semantic to object store
  • convert open/read/write/close to read/write
  • listing not supported

• introduce plugins to the transfer protocols

• slightly modify some tools
  • e.g. garbage collection daemon approach
Example of the Xrootd protocol

- very popular protocol for data transfers in High Energy Physics
- provides more than a protocol, a framework
  - supporting client redirection
  - embedding data federation
  - integrating seamlessly with ROOT, the main physics data format
**Xrootd plugins**

- different levels of interfaces
  - protocol (defaults xrootd)
  - filesystem (default POSIX)
  - storage (open/read/write/close)
- used storage
  - based on radosstriper
  - 100 lines of code mapping fds to ceph objects
  - and reused in our tools
Ceph as a generic Xrootd backend

- allows integrate Ceph into the computing Grid
- one can build a storage element on top of ceph
- several collaborations and institutes interested

But

- no directory listing
Stress Testing Ceph
The setup

- **Clients**: 8 boxes
- **CASTOR**: 5 boxes
- **Ceph**: 18 boxes

Some details:
- Client and CASTOR machines are batch nodes.
- All machines have 10 Gb/s connection.
- Ceph machines have 540 disks in total.
- Ceph cluster has 2 PB of effective space.
The setup

- **Clients**: 8 boxes
- **CASTOR**: 5 boxes
- **Ceph**: 18 boxes

**Some details**

- Client and CASTOR machines are batch nodes
- All machines have 10 Gb/s connection
- Ceph machines have 540 disks in total
- Ceph cluster has 2 PB of effective space
Default test conditions

Client setup
• pure writing, 300MB files
• 10 concurrent threads per node, 8 nodes

CASTOR setup
• using xrootd protocol, with 64MB buffers
• 16 slots per disk server

Ceph setup
• erasure coded pool (8+3)
• striping into 8MB rados objects
First test Results

- We manage to write at 3GB/s
- We are confident we can scale with nb of nodes
- We needed to “tune” quite some things
  - buffer sizes, parallelization level
  - make everything async
    - thus extending rados for remove/stat/xattrs
  - make async callbacks multithreaded
    - may be another extension to rados
- See CephForHighThroughput presentation for details
  - or watch pull requests in the near future