CERN

Ceph development in HEP

Sébastien Ponce
`sebastien.ponce@cern.ch`

Outline

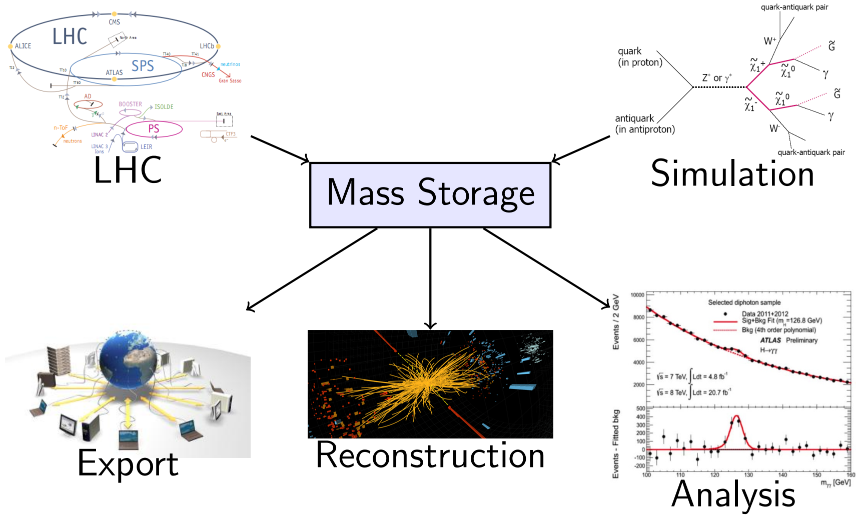
Our problem

Why going to Ceph ?

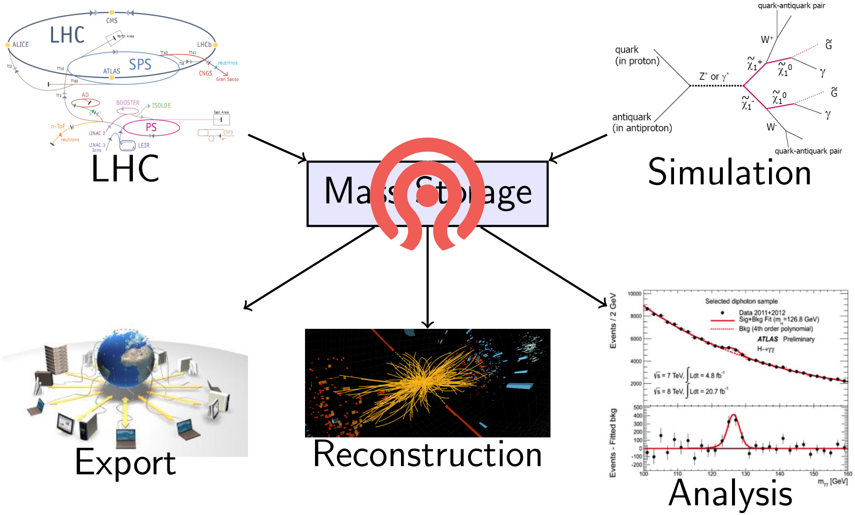
Building a complete solution

Our problem

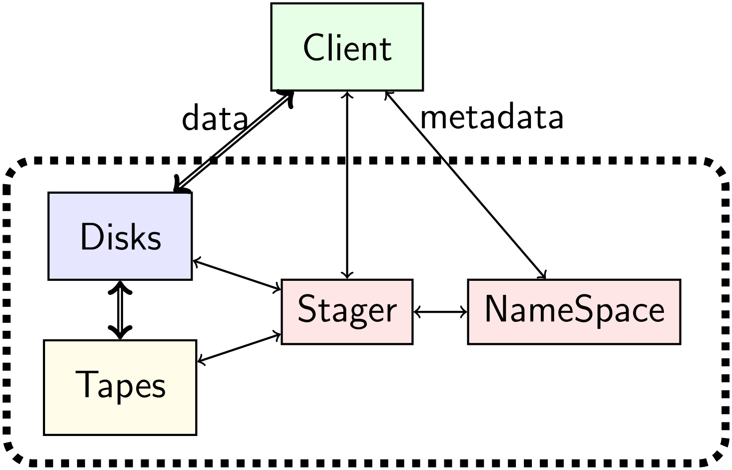
The big picture



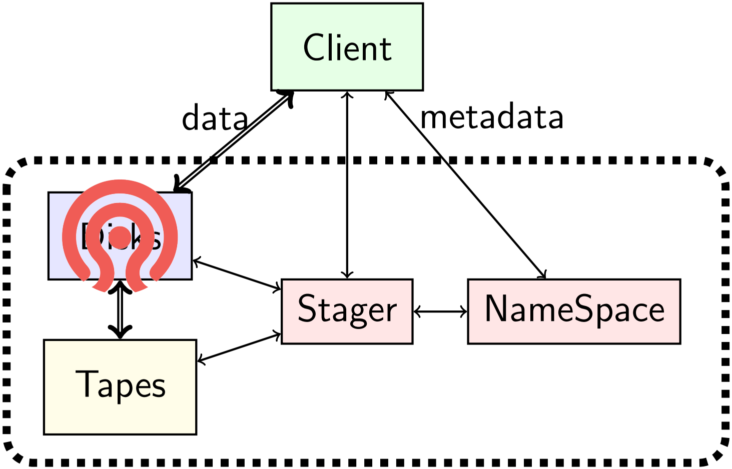
The big picture



Anatomy of a Mass Storage System



Anatomy of a Mass Storage System



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

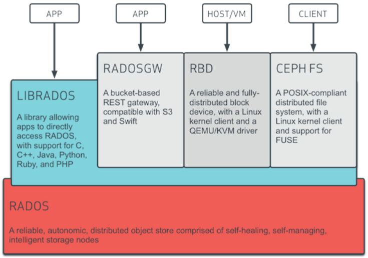
Why going to Ceph ?

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 eficiency
- 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



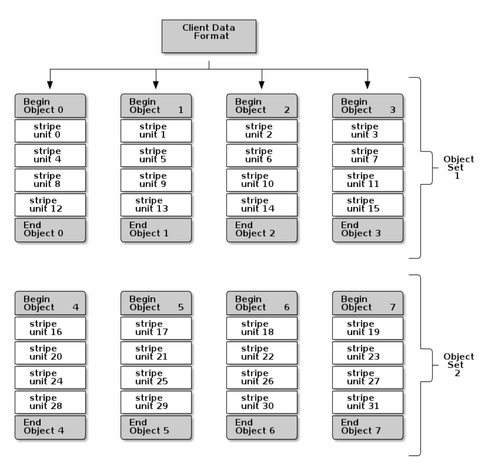
APP
APP
HOST/VM
CLIENT
LIBRADOS
A library allowing apps to directly access RADOS, with support for C, C++, Java, Python, Ruby, and PHP
RADOSGW
A bucket-based REST gateway, compatible with S3 and Swift
RBD
A reliable and fully-distributed block device, with a Linux kernel client and a QEMU/KVM driver
CEPH FS
A POSIX-compliant distributed file system, with a Linux kernel client and support for FUSE
RADOS
A reliable, autonomic, distributed object store comprised of self-healing, self-managing, intelligent storage nodes

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)

Building a complete solution

Interfacing our code with Ceph

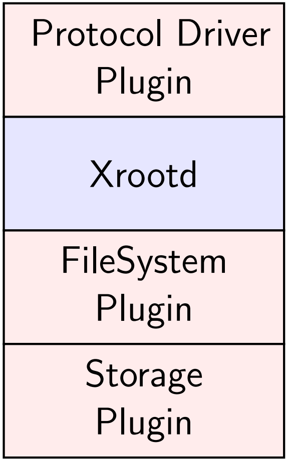
- mainly change POSIX semantic to object store
  - convert open/read/write/close to read/write
  - listing not supported
- introduce plugins to the transfer protocols
- slightly modify some tools
  - e.g. garbage collection daemon approach

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

GridFTP

Build on Xroot case

- reuse mapping of fds to objects
- first prototype took 2h

Production is another story

- pool mapping
- low throughput and chunk size issues
- authentification to be solved

see next talk

Conclusions

Ceph is suitable for our usage

- after a bit of learning
- and a couple of fixes (pull requests begin prepared)
- see next talk on optimization for throughput

Next steps

- stress testing (ongoing)
- test tape streams
- production for the Alice experiment