Erasure Coding
Xrootd Object Store
Andy Hanushevsky, Michal Simon, Wei Yang
Introduction to EC in Xrootd

Originally developed for EOS, extended to work with any type of Xrootd storage.

- Writing:
  - A data block at client is divided into chunks
  - The chunks are erasure coded
  - EC is implemented in Xrootd client
  - Using Reed Solomon erasure coding from Intel® ISA-L
  - Calculate crc32 of all chunks (data/parity)
  - Spread chunks to Xrootd data servers, using ZIP archive to group individual chunks and crc32c

- Reading:
  - Only read data chunks, unless reconstruction / error correction

A simplified view of EC:

\[
\begin{align*}
  a_1 x_1 + a_{12} x_2 + \ldots + a_{1n} x_n &= p_1 \\
  a_{21} x_1 + a_{22} x_2 + \ldots + a_{2n} x_n &= p_2 \\
  \vdots \\
  a_{m1} x_1 + a_{m2} x_2 + \ldots + a_{mn} x_n &= p_m
\end{align*}
\]

- Data: \((x_1, x_2, \ldots, x_n)\)
- Parity: \((p_1, p_2, \ldots, p_m)\)
- Tricky to choose Vandermonde matrix \(A_{ij}\)
- Compare to RAID blocks, EC block sizes are usually much larger

Xrootd ZIP archives on data servers
Key question: should users use EC enabled Xrootd client?

- Using EC enabled Xrootd client directly
  - Only good for LAN environment, only work for xroot protocol
  - This mode fits administrators but not users.

- Using a Xrootd EC proxy as gateway to Xrootd storage
  - A proxy is both a Xrootd server and a Xrootd client
  - Enable EC in the proxy’s xrootd client component.
    - EC is invisible to the users
    - Users use existing xrdcp/xrdfs, gfal, curl
  - Support all WLCG security, protocols, TPC, etc.
  - Expect performance penalty
    - Performance test should show whether this penalty is acceptable.
  - The backend xrootd storage is plain and simple
    - No central metadata service
    - Prefer to keep the simplicity of the backend

- This is a user friendly mode
- The rest for the slides talks about this model.
Other Design Choices

● How to “locate” file zip archives for existing or new files. What is the cost?
  ○ Also eliminate qdl delay (aka “5 second” delay) when creating new files
● What if a zip archive was left behind during cleaning?
● How to calculate and store checksum?
● How to balance storage usage on backend servers?
  ○ Especially when a new server is added, often with a larger capacity
● What is the user facing impact/failure mode when a disk/server is down
● How to identify files that have lost zip archive?
  ○ Depend on the cause of loss, it is possible to identify degraded files through scanning of Xrootd storage namespace or actual storage.
● How to recover from corruption or HW failure
● How to apply services (patching, etc.) without interrupting operation?
Interface to Users

Nothing changed: users will still work with root(s) or http(s) URL:
- https://atlas.cern.ch:1094/atlas/rucio/user/jdoe/my.data or
- root://atlas.cern.ch:1094//atlas/rucio/user/jdoe/my.data
- Think of “atlas/rucio/user/jdoe” as bucket, folder, whatever you like.
  - Access permission is managed at the proxy/gateway, not backend storage.

CLI tools for GET/PUT/DEL/LIST/RENAME/TPC
- xrdcp/xrdfs: work mostly with root(s) URLs
- gfal2: works with both root(s) URL and http(s) URLs
- curl: works with http(s) URLs
- No overwriting of existing file: do explicit deletion first

API calls through xroot and http protocols
- All xroot native IO calls and xrootd posix IO calls:
  - Except: open() with O_APPEND or O_TRUNC, truncate() and perhaps writev()
  - Low expectation on small read() and vector readv() performance
- Expect similar situation for HTTP protocol
- S3 support (just an experiment using boto3, no relation to EC):
  - Authenticate with bear tokens (JWT or Macaroon)
  - Object operations work. Note: XrdHTTP responses to successful upload by an smiling emoji ← should be removed
  - Bucket operations do not work ← XrdHTTP should responds in XML (instead of HTML) if the client agent is boto3
Test Environment

**Backend**: Xrootd storage:
- 19 nodes of retired Dell R510s, each:
  - 24GB RAM, 1Gbps NIC, 12x 3TB HDD (some have 11)
  - Each HDD is presented to the OS as its own SCSI device (via LSI RAID controller)
  - CentOS 7, Xrootd 5.3.4 (later auto-updated to 5.4.0), xrootd “sss” security
- 312 pre-placed test files (ATLAS data files) ranging from 30MB to 1.1GB, all with known adler32 checksum

**Frontend**: Xrootd EC proxy
- 64 core, 128GB, 100Gbps NIC
- CentOS 7, unreleased Xrootd (2021-12-17+patch ← this is newer than 5.4.0)
- EC configuration: 8+2, chunk size 1MB (So a block has 8+2 MB)

*xroot protocol vs HTTP protocol*
- Most test will be done using the xroot protocol. HTTP protocol has higher overhead. It also have to be translated to xroot protocol in order to handle EC storage.
- Small performance degradation in HTTP protocol isn’t a concern. For HTTP protocol, we will primarily exam the error rate.
Performance

150 concurrent clients
Each read the pre-placed 312 data files, then repeat 5 times
Memory cache in Xrootd proxy clearly helped

- Randomly and continuously write files (from the 312 sample) and then read back.
- Reached to 200 concurrent clients (reached the plateaus at <100 clients)
Performance, cont’d

- Files operations
  - Put/Get/Delete
- 10,000 files, each is 27KB
  - Put/Get/Delete sequentially:
    - 10K Put, then 10K Get, then 10K deletion
- In 1,2,4,8,12 ... parallel operations
  - All ran on the same EC proxy machine
  - Managed by “xargs” in order to sustain large number of concurrent operations.
  - The machine is overloaded at 500+ parallel oprs

Likely will accommodate ~400Hz of small file Put/Get/Del in any combination.

Performance degradation beyond 500 clients
- Had to run all 500+ clients (xrdcp/xrdfs) and the proxy on the same machine to avoid cyber security rule on DoS attack.
Administrative Tasks and Tools

Main tasks for administration and operation
- Perform routine planned outage for OS patch, etc.
  - This can be done transparent to users because EC tolerants shutting down a data server.
- Discover and recover degraded files/objects (also see next slide)
- Discover new files for backup
- Clean debris left behind
- Move data to balance storage ← XrdEC already has such a capability built-in for new files

Tools and scripts for administrative tasks
- XrootdFS allows a mounted file system view for administrators
  - Can do almost all administrative tasks except discover of degraded files and file debris
- CLI based ingredients for high level tools/scripts already exist.
  - xrdcp: copy data and extract checksum
  - xrdfs: find data (zip) file location, move locations, get/set xattrs

Validated disaster recovery scenarios:
- Lost a disk or disk array: name space will tell what are on the lost disks
- Lost the namespace on a data server: can recover from metadata (xattr on disks)
- Lost both name space and some/all disks on a data server: same as losing a data server (below)
- Lost a data server
  - If this is non-storage related issue (CPU, RAM, NIC, Power): just fix it — there is no data loss. Operation not affected.
  - Otherwise need to scan the whole storage to see which files has missing zip archives.
Future Work

- Can we cache to zip archive location (node name) to speed up file look up for existing files
  - For example, cache the info in cmsd?
- Have XrdEC logging files with missing zip archive during operation.
  - To remind admins to run recovery.
- Develop tools to recover just the missing zip archive
  - Want a lightweight recovery tool compare to whole file copying over XrootdFS or xrdcp
- Mote test on modern HW
- Develop path to migrate from non-EC storage to EC storage
  - They can co-exist on the same storage cluster. But how to migrate?
- How to package XrdEC in RPMs?
  - Note there is an Intel ISA-L library involved
- Document and operational procedures
  - Many are already available at xrootd-howto in readthedocs.io
Summary

- Erasure Coding in Xrootd is already quite stable and useful.
- Due to the nature of EC, it is better to treat it as an object store
  ○ Though most of the Posix IO functions are still available
- It can be seamlessly integrated into the WLCG ecosystem
  ○ It can fill the gap where commercial EC storage systems are not a good fit.
- A prototype demonstrated impressive performance and resilience.
- Already have good documentation on how-to-use
- Future improvements largely depends on the real work usage feedback.