XRootD Roadmap

GSI Virtual Micro Workshop

December 14-15, 2020

Andrew Hanushevsky, SLAC

Introduction

- This roadmap contains
  - The context in which it was created
  - The starting point and
  - What drives this map
- It has no specific destination
  - The world changes too quickly
    - But you’ll get the feel of where it can go and why
- Sit back and enjoy the journey!

The XRootD Project

- A structured Open Source community supported project to provide a framework for clustering distributed storage services available via github, EPEL, & OSG
  - The project also supplies the fundamentals
    - A packaged storage service that meets many needs
      - But one that is also highly customizable

What the project does

- Accepts contributions from all disciplines
  - Core team supplies architectural consistency, code vetting, integration, packaging, documentation inclusion, testing (via CI), maintenance and support *management*
  - Successfully doing so for 20+ years
  - We rely on the community to assist in testing, CI enhancements, support, and bug fixes
    - The project co-ordinates these activities
      - Keep in mind, we are not a software company!

The XRootD Project Software

- Framework runs on common platforms
  - Most popular Linux distributions & macOS
  - Includes full featured python bindings
- Focus on diverse community needs
  - Widely used in HEP and Astro communities
    - Significant use in many other disciplines
      - Via our community partner designed systems
        - Where framework is embedded in a larger system
          - Our unofficial logo is “**XRootD** inside!”
            - E.G. CTA, DPM, EOS, PRP, Qserv, StashCache

Current storage support

- Any kind of mounted Posix-like file system
- Unmounted file systems
  - Ceph (2nd party, originally developed by Sebastien Ponce - CERN EP-LBC)
  - HDFS (3rd party, originally developed by Brian Bockelman - Morgridge)
- Tape
  - CTA (3rd party, plug-ins developed by Michael Davis - CERN IT-ST-TAB)
  - HPSS (1st party, integration developed by SLAC)
  - Client access via **XRootD** prepare protocol
    - SRM support is not envisioned

Current storage access modes

- Posix-like file system access via
  - xroot[s] and http[s] protocols
  - FUSE mounted file system
- LAN clustered & distributed WAN clusters
  - Using **cmsd** clustering services
    - Independent of protocol used for access
  - Best LAN example is UCSD **Xcache**
  - Best WAN example is CMS AAA

Current storage caching modes

- Posix-like caching file system via
  - **FRM** (File Residency Manager) cache
    - Read/write whole file access
      - Supports all transfer protocols to/from cache
  - **Mcache** (memory caching only)
    - Read/write block level file access
      - Supports xroot[s] and http[s] to/from cache
  - **Xcache**
    - Read/only block level file access
      - Supports xroot[s] and http[s] to populate cache

Current QoS support

- WLCG QoS support in wait and see mode
  - We have not received *any* community requests for extensive QoS functionality (except for GSI)
  - Framework already provides QoS templates
    - Similar to SRM space tokens but more flexible
      - Tied to a logical path or selected via CGI element
    - This seems good enough for communities we serve

QoS templates

- A file may be created in a *cgroup*
  - E.g. xroot://*host*//*path*?oss.cgroup=*cgname*
- Each *cgroup* is tied to a particular QOS
  - I.E. the *cgroup* is effectively a QOS template
- Currently, QOS is determined by hardware
  - E.g. HD, SSD, etc though can be extended
    - Via external site-specific actions based on *cgroup*
      - These need to be provided & implemented by the site

QoS cgroup specification

- A *cgroup* is defined using **oss.space**
  - **oss.space** *cgroup mountpoint*
  - Logical file paths may be assigned a *cgroup*
    - **oss.space** *cgroup* {**assign** | **default**} *lfnpfx* [*lfnpfx* [...]]
      - Logical paths and *cgroups* are independent
        - Files in a directory can be in different *cgroups*
  - A file may be reassigned to a different *cgroup*
    - Admin function via the **frmadmin reloc** command
      - https://xrootd.slac.stanford.edu/doc/dev50/frm_config.htm#_Toc43844791
  - For *cgroup* implementation see
    - https://xrootd.slac.stanford.edu/doc/dev51/ofs_config.htm#_Toc53410343

Typical QoS cgroup usage

- Currently used in very limited domains
  - In ATLAS as SRM space tokens
    - DATASPACE, GROUPSPACE, SCRATCHSPACE
  - In **Xcache** for physical data separation
    - A *cgroup* for actual data files (usually HD)
    - A *cgroup* for metadata files (may be SSD)

Where we are today

- 5.0.3 with numerous requested features
  - **XRootD**
    - TLS with performance enhancements, JSON monitoring streams, credential forwarding, user file attributes, hardware CRC32C, plug-in stacking, K8s deployment options, enhanced tape support, universal multi-VO VOMS plug-in, and many more
  - **http**[**s**]
    - Full TPC, proxy cert handling, SciTokens, multi-VO support, and several more

Highlight: TLS core

- TLS core configured using directives:
  - **xrd.tls**, **xrd.tlsca** and **xrd.tlsciphers**
    - These can apply to **https** and **xroots**
      - For backward compatibility can still use **http.*xxx***
        - *xxx*: **cadir**, **cafile**, **cert**, **cipherfilter**, and **key**
          - Directive mode controlled via directive
            - **http.httpsmode** {**auto** | **disable** | **manual**}
  - For details see
    - https://xrootd.slac.stanford.edu/doc/dev51/xrd_config.htm#_Toc49272850

Highlight: TLS https & xroots

- https adds one new TLS directive
  - **http.tlsreuse <u>off</u> | on**
    - For backward compatibility at non-X509 sites
- xroots adds two new TLS directives
  - **xrootd.tls [capable]** *req*
    - *req:* **[-]all | [-]data | [-]login | <u>none</u> | off | [-]session | [-]tpc |** *req*
      - This is for optimization and backward compatibility
        - See https://xrootd.slac.stanford.edu/doc/dev51/xrd_config.htm#_tls
  - For details see
    - https://xrootd.slac.stanford.edu/doc/dev51/xrd_config.htm#_Toc49272850

Highlight: Automatic crl refresh

- The crls are automatically refreshed
  - Server side function
    - No need to restart server
- **xrd.tlsca noverify** | {**certdir** | **certfile**} *path* [*options*]
  - *options*: [**crlcheck** {**all** | **external** | **last**}]
    [**log** {**failure** | **off**}] [[**no**]**proxies**]
    [**refresh** *rint*[**h**|**m**|**s**]] [**verdepth** *vdn*]
- See https://xrootd.slac.stanford.edu/doc/dev51/xrd_config.htm#_Toc49272858

Highlight: JSON Monitoring

- New G-Stream monitoring added
  - For use in low to medium report rates
    - E.g. **Xcache** and TCP monitoring
      - Specifically geared for plug-ins
  - Data should be in JSON
    - Though that is determined by the plug-in
  - Easily ingestible by elastic search, etc
    - No need for specialized collectors

Highlight: Credential forwarding

- The **sss** authentication protocol enhanced
  - Can forward credentials of any other protocol
    - E.g. x509 -> sss -> x509 (*recreated*)
  - Used for server to server proxy authentication
    - Client x509 authenticates to server *A*
    - Server a requests action in behalf of client at *B*
      - Server *A* authenticates with server *B* using **sss**
    - Server *B* executes using client's original credentials
  - For details see
    - https://xrootd.slac.stanford.edu/doc/dev50/sec_config.htm#_Toc56021439

Highlight: User file attributes

- Directive added to control user settings
  - **ofs.xattr** [**maxnsz** *nsz*] [**maxvsz** *vsz*] [**uset** {**on** | **off**}]
- Underlying file system must support xattr
  - Some require mount option or config setting
    - E.g. ext*n* and lustre
- xrdcp is able to copy extended attributes
  - --xattr option similar to --preserve in cp
- For details see
  - https://xrootd.slac.stanford.edu/doc/dev51/ofs_config.htm#_Toc53410333

Highlight: Universal VOMS

- VOMS plug-in enhanced
  - Supports multiple VO's
    - Authorization can take into account user's VO
      - See https://xrootd.slac.stanford.edu/doc/dev50/sec_config.htm#_Toc56021456
  - Same plug-in for https and xroot[s] protocols
    - Simplifies deployment and configuration
      - Requires install of **libvomsapi.so** library for use

Highlight: Stackable plug-ins

- Most plug-ins can now be stacked
  - Addition of ++ option on directives
    - ofs: authlib, ctllib, osslib, preplib, and xattrlib
    - sec: entitylib
    - xrd: tcpmonlib
    - xrootd: fslib
  - Simplifies enhancing existing plug-ins
    - No need to rewrite just wrap it!

Highlight: Tape support

- New plug-in directive for tape support
  - **ofs.preplib** [**++** | [**+noauth**]] *path* [*parms*]
- Plug-in to handle xroot prepare request
  - Used to prime redirectors
  - Used to facilitate access to offline files
    - E.g. “bring online”
- For details see
  - https://xrootd.slac.stanford.edu/doc/dev51/ofs_config.htm#_Toc53410327

Highlight: Caching exports

- Seamless support of cacheable paths
  - **all.export** *path* … [**no**]**cache**
    - Automatically supplies all the required boilerplate needed to export **Xcache** managed paths to a redirector
      - Also applies to **FRM** caches

Highlight: Kubernetes support

- Support to ease k8s deployments
  - New **cms** directive for virtual networking
    - **cms.vnid** {=*id* | <*path* | @*libpath* [*parms*]}
      - Establishes a network namespace to track servers
        - Normally DNS name or IP address would be used
      - See https://xrootd.slac.stanford.edu/doc/dev50/cms_config.htm#_Toc53611101
  - Enhanced **xrd** directive for k8s DNS
    - **xrd.network** … [[**no**]**dyndns**]
      - Accommodates the volatile nature of k8s DNS
      - See https://xrootd.slac.stanford.edu/doc/dev51/xrd_config.htm#_Toc49272864

Highlight: SciTokens

- SciTokens plug-in available
  - Token based authorization
    - Requires use of a recognized token issuer
      - Infrastructure for issuing tokens is still in flux
    - Requires TLS support (i.e. token encryption)
  - Available for https and xroots
    - Doing seamless integration with xtootd
      - Now plug-in is a 3rd party addon

Highlight: Extended https x509

- https protocol has full x509 cert support
  - Recognizes non-proxy certificates
    - This is the standard
  - Recognizes proxy certificates (new)
    - Along with VOMS extension

Highlight: HTTP TPC

- The http plug-in now supports TPC
  - Third party copy push and pull modes
    - Based on special headers (non-standard)
    - Uses libcurl to implement transfer agent
  - Relies on Macaroon support (included)
    - Server to server TPC authorization
  - No plan to support macaroons for xroot

Highlight: Command options

- Two command line options added
  - **[-a | -A]** *path*
    - Set admin path via command line
  - **[-w | -W]** *path*
    - Set homepath (cwd) path via command line
- Better support for systemd setups

Highlight: New commands

- xrdpinls
  - List all recognized plug-ins
    - Also provides required version information
      - Lists where a version tag is required, minimum version allowed, and associated directive
        - Optional >= 5.0 bwm.policy
        - Required >= 5.0 cms.perf
        - Required >= 5.0 cms.vnid
        - Optional >= 5.0 gsi-authzfun

What are the possible plug-ins?

- There are 27 plug-in points
  - 25 for the server
  - 2 for the client
- Most plug-ins are not exclusive
  - Either they run in parallel or are stackable
    - E.G. Protocol plug-ins run in parallel
- Plug-ins allow system customization
  - Most are supplied in the **XRootD** core

Plug-ins I

@logging	Log message handler (server – cli option)
bwm.policy	Network bandwidth management
cms.perf	Performance monitor for cmsd (not script based)
cms.vnid	Virtual network identifier generator for cms
gsi-authzfun	Specialized gsi authz function
gsi-gmapfun	Specialized gsi gridmap function
gsi-vomsfun	Specialized gsi VOMS function
http.exthandler	HTTP authentication post processing
http.secxtractor	HTTPS security information extraction
ofs.authlib	Authorization plug-in
ofs.ckslib	Checksum plug-in
ofs.cmslib	Cluster management service client plug-in
ofs.ctllib	Specialized file system control plug-in
ofs.osslib	Storage system plug-in
ofs.preplib	Prepare request plug-in

Plug-ins II

ofs.xattrlib	**Extended attribute handler plug-in**
oss.namelib	**Name mapping plug-in**
oss.statlib	**Functional stat() plug-in**
pfc.decisionlib	**Cache purging decision plug-in**
pss.cachelib	**Cache implementation plug-in**
pss.ccmlib	**Cache context management plug-in**
sec.protocol	**Authentication protocol plug-in**
xrd.protocol	**Communications protocol plug-in**
xrdcl.monitor	**Client-side action monitor plug-in**
xrdcl.plugin	**Client-side API implementation plug-in**
xrootd.fslib	**File system plug-in**
xrootd.seclib	**Security manager plug-in**

Why so many plug-ins?

- Some people ask why so few
  - It’s a matter of perspective and needs
- XRootD architecture is highly modularized
  - Allows for specific functional replacement
    - Approach supports a myriad of authentication & authorization schemes, storage systems, clustering, and protocols among many other variations
      - This has allowed for long-term (i.e. 20+ years) evolution
- For simplicity every plug-in has a default!

Where do we go from here?

- Obvious next step is 5.1.0
  - Available in RPM form within days
- Recommend to deploy 5.1.0
  - 5.0.3 useful for testing
    - However, it still contains a number of bugs
      - All corrected in 5.1.0
- Plus 5.1.0 contains more features!
- Let’s look at the roadmap

XRootD roadmap drivers

- Experimental needs
  - We also try to anticipate future needs
    - Different perspective outside the trenches
      - Especially when considering a diverse community
- Balance between competing desires
  - Stability, performance and features
    - Roadmap tilts toward the former for start of run
- Commitment to backward compatibility
  - Can still mix circa 2000 clients and servers

Planned release schedule

- 5.1.x 4Q20 (almost if not there)
- 5.2.x 1-2Q21
- 5.3.x 3-4Q21
- Feature addition schedule is fluid
  - While we have plans experimental needs take precedence and may shuffle the schedule
- So, on to the highlights!

New Integrity Features in 5.1.0

- Data in motion integrity
  - CRC32C checksum for each 4K xmit unit
    - Dynamic substitution of checksum equivalent (i.e. TLS)
    - Real-time error correction using CRC32C
      - Only blocks in error are retransmitted (not for TLS)
        - Potential to substantially reduce network usage
          - Consider a 10GB file transfer with a 1 bit error
  - First deployment will be in **Xcache**
    - Subsequent rollout for xrdcp in 5.2.0

New Integrity Features 5.2.0

- Data at rest integrity
  - CRC32C checksum for each 4K disk block
  - Real-time error detection
- First usage will be in **Xcache**
  - Where only blocks in error will be re-fetched
- However, this is a universal plug-in
  - Any storage system may use it (e.g. ext4, xfs, etc)
    - Kudos to David Smith (CERN IT-SC-RD) who developed it

Using Xcache integrity features

- **pss.cschk** *opts*
  - *opts*: **[[no]cache] [[no]net] [off] [[no]tls] [uvkeep** { *n*[**d**|**h**|**m**|**s**] | **lru** }]
- Integrity feature is on by default
  - Substituting TLS when CRC is unavailable
    - Can switch this off with **notls**

Xcache integrity confidence

- Storage system tracks CRC confidence
  - Verified
    - Server sent CRC or TLS was used
  - Unverified
    - CRC locally generated to detect media errors
  - None
    - No CRC is available
- Unverified blocks may be re-fetched
  - See https://xrootd.slac.stanford.edu/doc/dev51/pss_config.htm#_Toc50581514

New Integrity Features III

- R 5.2.0 or 5.3.0
  - Data in motion integrity for writes
    - CRC32C checksum for each 4K transmission unit
    - Real-time error correction using CRC32C
      - Only blocks in error are retransmitted
        - Potential to substantially reduce network usage
    - Write integrity is far more difficult than reads
      - Different set of edge cases most of which are problematic
  - First deployment will be xrdcp

New ACID* Features (5.3.0)

- File checkpoints
  - Allows safe recoverable in-place updates
    - Server-side updates for Zip, Zarr, HDF5, etc files
      - Especially needed by other communities
  - Completes **XRootD** native Zip file support
    - Extraction, listing, and now appends
  - Driven by increasing use of Zip archives
    - E.G. Log files in ATLAS

*Atomicity, Consistency, Isolation, and Durability

New HPC oriented features I

- Fast data paths
  - Ability to selectively use faster data interfaces
    - Extends current multi-stream support to multi-path
      - This is peculiar to but common in HPC systems
        - Control interface is slow but data interface is fast
  - During logon client told of faster interfaces
    - Allows subsequent use for data transfer
      - Site can restrict fast interfaces to data only

New HPC oriented features II

- RDMA for data transport
  - Common in HPCs but is spreading
    - Driven by adoption of InfiniBand networks
      - LCLS-II at SLAC will use an internal InfiniBand network
    - Already have implicit RDMA via DCA feature
      - Direct Cache Access using Lustre based **Xcache**
        - Being used by GSI and NERSC

Enhanced Parallel XRootD

- XRootD runs on each worker node
  - There could be hundreds of these
- Data flow needs to minimize network use
  - Data source to running application
- Needs real-time data flow scheduling
  - Partly addressed but needs improvements
    - Driven by large scale sites (e.g. U Wisconsin)

Enhanced Write Support (backend)

- Distributed write recovery
  - For systems that support it (e.g. EOS)
    - Eliminates full file retransmission upon error
      - Writes can proceed using another data server
- Part of **XRootD** file copy framework
  - Automatically extends to gfal and xrdcp

Redirect minimization

- Ability to always use primary head node
  - Targeted toward consensus driven services
    - EOS is one such service
  - Several head nodes but only one is the primary
    - New one chosen after a failure
  - Client told redirect target is the primary
    - Subsequent requests only go to primary head node

Performance Improvements

- xrdcp
  - Simplify buffer management
  - Use kernel space buffers
  - Approximately 3-4x reduction in CPU usage
  - Up to a 40% increase in transfer speed
    - Depending on target device

Universal Third Party Copy (TPC)

- Ability to copy from/to using any protocol
  - To/from local file system from/to elsewhere
  - To/from elsewhere from/to elsewhere
- Simplifies current TPC implementation
  - Leverages the **kXR_gpfile** protocol element
  - Compatible with any authentication scheme
- Currently we support **XRootD** (pull mode) and **http**[**s**] (push and pull modes)

Plug-In Roadmap

- Previous slides were core enhancements
  - Either server or client based features, but…
- Large part of roadmap centers on plug-ins
  - Most have been developed elsewhere
- These support AAI and backends
- Let's take a test drive….
  - Stops in no particular order

SciToken plug-in (AAI)

- Based on existing OSG plug-in
    - Add security enhancements for **XRootD** use
        - Already available via **http**[**s**] plug-in
            - Being used by several sites
    - Will become part of the **XRootD** core

XcacheH plug-in (other communities)

- Accessing Xcache origins using http[s]
  - Broadens data access reach
    - Oriented toward multi-discipline sites
  - Can be used as a Squid replacement
    - Better performance and scalability
  - Based on the plug-in by Radu Popescu
    - Formerly at CERN now at Proton Tech AG
      - Further developed by Wei Yang - SLAC
  - Prototype being tested by ESNET & ESCAPE

Erasure coding plug-in (backend)

- Client side plug-in to support EC writes
  - Based on Intel ISAL
    - Hardware accelerated encoding
  - Leverages **XRootD** pgWrite capability
    - Data in motion integrity with recoverability
- Driven by ALICE requirements
  - Direct writes from the DAQ system to EOS
- Developed by Michal Simon (CERN IT-ST-PDS)

Unix Multi-User plug-in (other communities)

- Allow file ownership based on uid-gid
  - Access is based on Unix permission bits
    - **XRootD** no longer owns the file
    - A.K.A. uid-gid file tracking
- Builds on the OSG multi-user plug-in
- Popular at small sites as an NFS alternative
  - Especially as a drop-in replacement

Enhanced SSI* plug-in (other communities)

- Detachable tasks
  - Results collected from alternate locations
- Task grouping
  - Dynamically consolidate sharded requests
    - Eases task management scaling
- Driven by LSST qserv requirements
  - Typically run 200,000 parallel query tasks
    - Coordinated by one or more master nodes

*Scalable Service Interface – an **XRootD** specialization plug-in

Other developments

- Improved Ceph plug-in
  - Addition of more features
    - Vector reads/writes
- Packet marking
  - Labeling purpose of data in network packets
    - IPv6 only
  - XRootD will be used as a demonstrator

Conclusion

- This is a diverse roadmap
  - Features needed by one or more experiments
    - Not always in the HEP community
      - 73% of github tickets are enhancement requests
        - For features missing in other open source systems
- As we approach HL-LHC
  - Feature additions will diminish
  - Performance and stability enhancements will increase

A Word Of Thanks

- We are grateful for our core partners

  CERN SLAC UCSD

- We are also grateful for our community & funding partners and their support
  - OSG WLCG Worldwide LHC Computing Grid ATLAS EXPERIMENT ALICE CMS LSST dCache GSI NSF
    - Plus way too many other logos to fit (I should work on that)!
- And of course, the front-line people that make it all actually work!