# XRootd & XCache: News from the frontier

7th Rucio Community Workshop @ SDSC

Andrew Hanushevsky, SLAC
Guilherme Amadio, CERN
Matevž Tadel, UCSD

San Diego, October 2, 2024

# Overview

On behalf of the **XRootd collaboration** and, in particular:

- **Andy Hanushevsky**, SLAC, ATLAS & LSST
  - Project lead, core development
- **Guilherme Amadio**, CERN IT
  - Core XrdCl development, software management, release management
- **Matevž Tadel**, UCSD, CMS
  - XCache development

***Contents:***

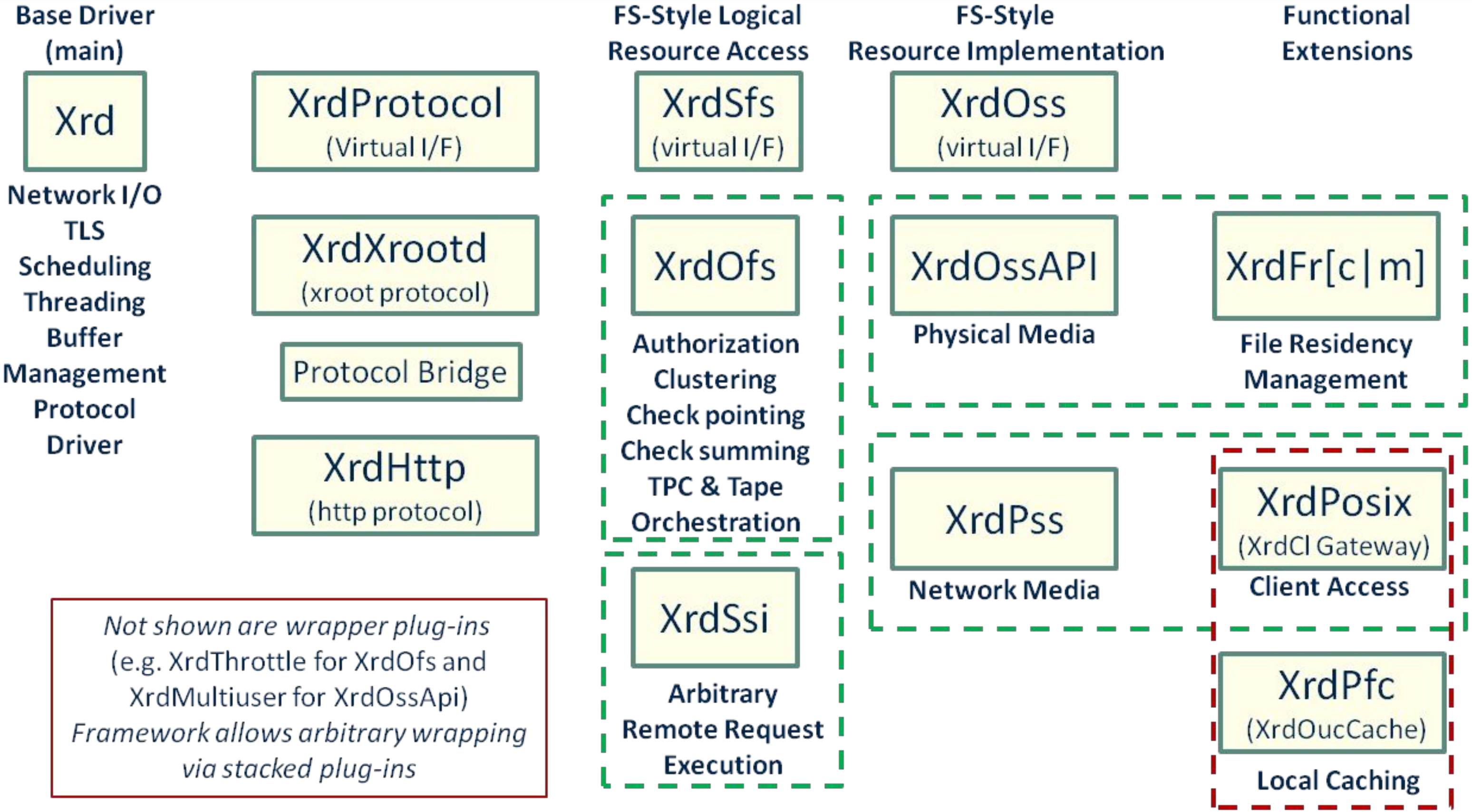
- Introduction – What is XRootd
- What XRootd can do for you
- Highlights of latest development & plans

See also Indico page of XRootd & FTS Workshop @ SFTC UK, Sept. 2024

# Introduction

- XRootd is the Formula 1-grade framework for data-access and data-delivery
- XRootd is the Formula 1-grade Plugin Framework to deploy C++ services
  - The initial hallmark plugins provided interface to distributed, load-balanced POSIX storage
  - Since then dozens of new plugins have been developed to address many community needs
- Strengths of XRootd
  - XROOT protocol – was 20-years ahead of time: vector reads, partial responses
  - XRootD plugins: file servers, data federations, EOS, XCache, Vera Rubin QServ,...
  - Own client – XrdCl – provide data to jobs / processes … but also connect server instances
    - Direct use, xrdcp / xrdfs, from ROOT, custom file access layers (*XrdAdaptor* in CMS)
    - In FUSE module and, of course, in XCache
  - Clustering of servers and caches
    - Creating a uniform namespace, even though storage is distributed→ data federations
    - Scalable, load-balanced services
  - XRootD Collaboration: flexible, open to community requirements
    - We work with you to make your problems go away

# XRootD Core



Base Driver (main)

Xrd

Network I/O
TLS
Scheduling
Threading
Buffer Management
Protocol Driver

XrdProtocol (Virtual I/F)

XrdXrootd (xroot protocol)

Protocol Bridge

XrdHttp (http protocol)

FS-Style Logical Resource Access

XrdSfs (virtual I/F)

XrdOfs

Authorization
Clustering
Check pointing
Check summing
TPC & Tape Orchestration

XrdSsi

Arbitrary Remote Request Execution

FS-Style Resource Implementation

XrdOss (virtual I/F)

XrdOssAPI

Physical Media

XrdPss

Network Media

Functional Extensions

XrdFr[c|m]

File Residency Management

XrdPosix (XrdCl Gateway)

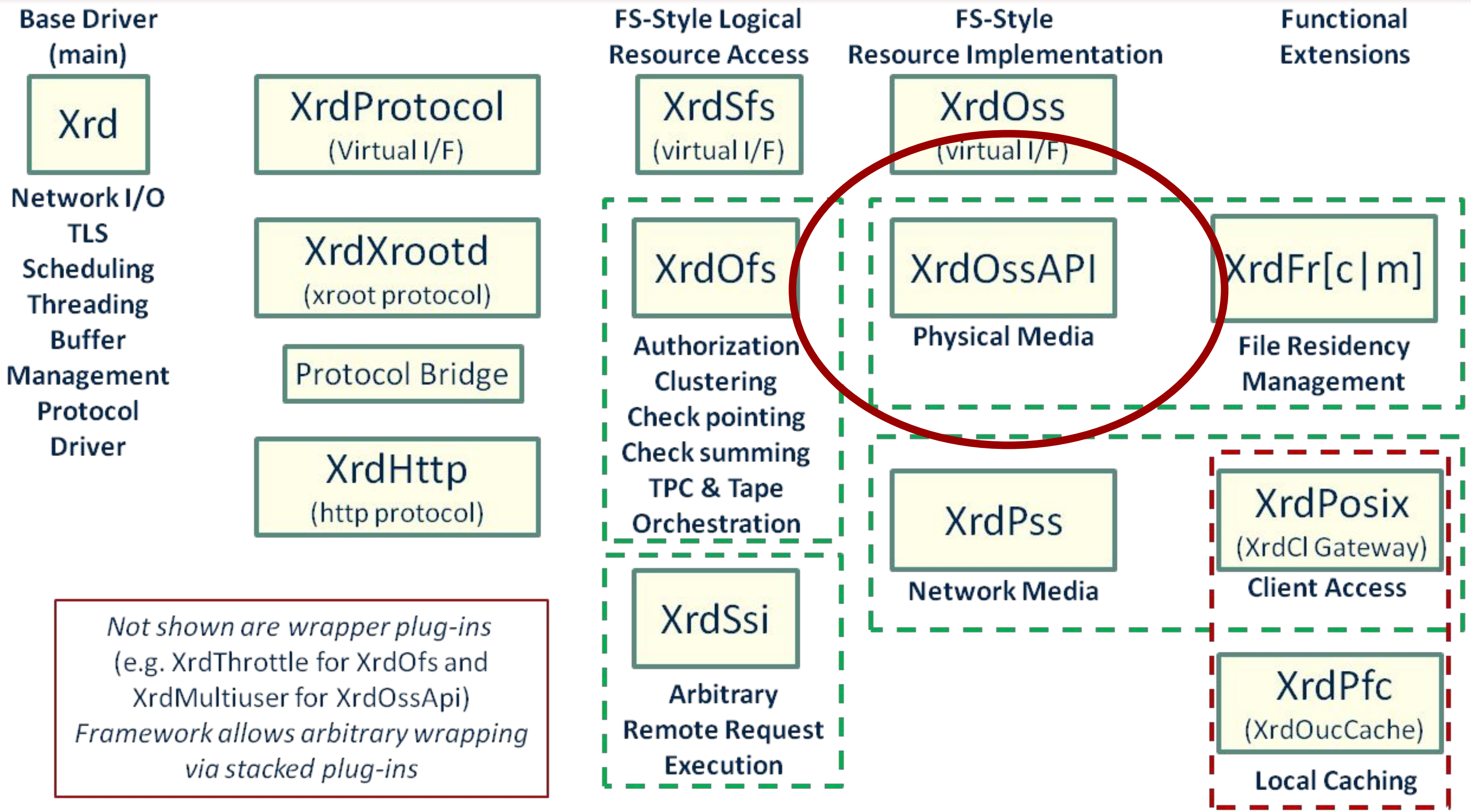
Client Access

XrdPfc (XrdOucCache)

Local Caching

*Not shown are wrapper plug-ins*
(e.g. XrdThrottle for XrdOfs and
XrdMultiuser for XrdOssApi)
*Framework allows arbitrary wrapping*
*via stacked plug-ins*

# XRootD Core – File server



Base Driver (main)

**Xrd**

Network I/O
TLS
Scheduling
Threading
Buffer Management
Protocol Driver

**XrdProtocol** (Virtual I/F)

**XrdXrootd** (xroot protocol)

**Protocol Bridge**

**XrdHttp** (http protocol)

FS-Style Logical Resource Access

**XrdSfs** (virtual I/F)

**XrdOfs**

Authorization
Clustering
Check pointing
Check summing
TPC & Tape Orchestration

**XrdSsi**

Arbitrary Remote Request Execution

FS-Style Resource Implementation

**XrdOss** (virtual I/F)

**XrdOssAPI**

Physical Media

**XrdPss**

Network Media

Functional Extensions

**XrdFr[c|m]**

File Residency Management

**XrdPosix** (XrdCl Gateway)

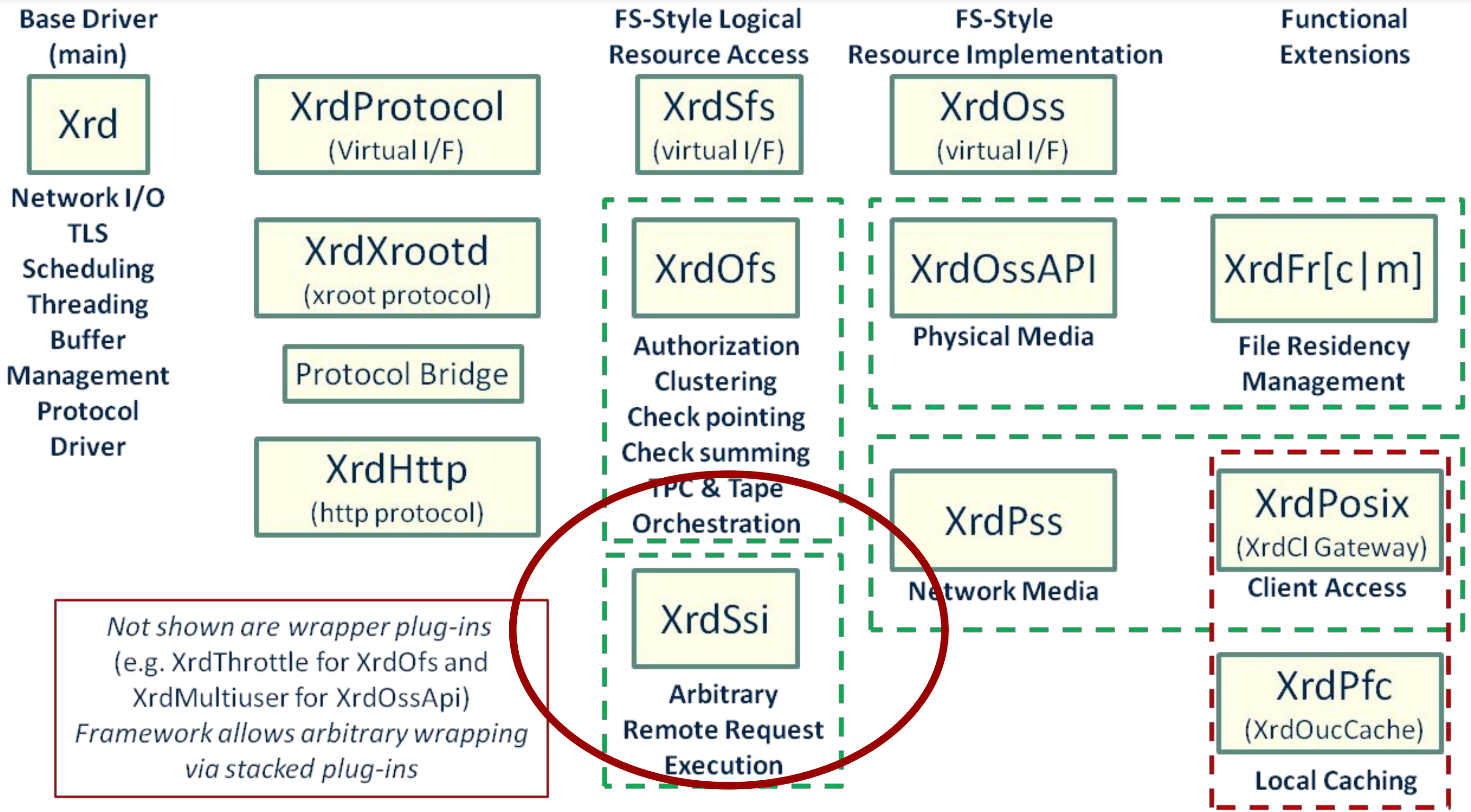
Client Access

**XrdPfc** (XrdOucCache)

Local Caching

*Not shown are wrapper plug-ins*
(e.g. XrdThrottle for XrdOfs and XrdMultiuser for XrdOssApi)
*Framework allows arbitrary wrapping via stacked plug-ins*

# XRootD Core – LSST / Vera Rubin



Base Driver (main)

Xrd

Network I/O
TLS
Scheduling
Threading
Buffer Management
Protocol Driver

XrdProtocol (Virtual I/F)

XrdXrootd (xroot protocol)

Protocol Bridge

XrdHttp (http protocol)

FS-Style Logical Resource Access

XrdSfs (virtual I/F)

XrdOfs

Authorization
Clustering
Check pointing
Check summing
TPC & Tape Orchestration

XrdSsi

Arbitrary Remote Request Execution

FS-Style Resource Implementation

XrdOss (virtual I/F)

XrdOssAPI

Physical Media

XrdPss

Network Media

Functional Extensions

XrdFr[c|m]

File Residency Management

XrdPosix (XrdCl Gateway)

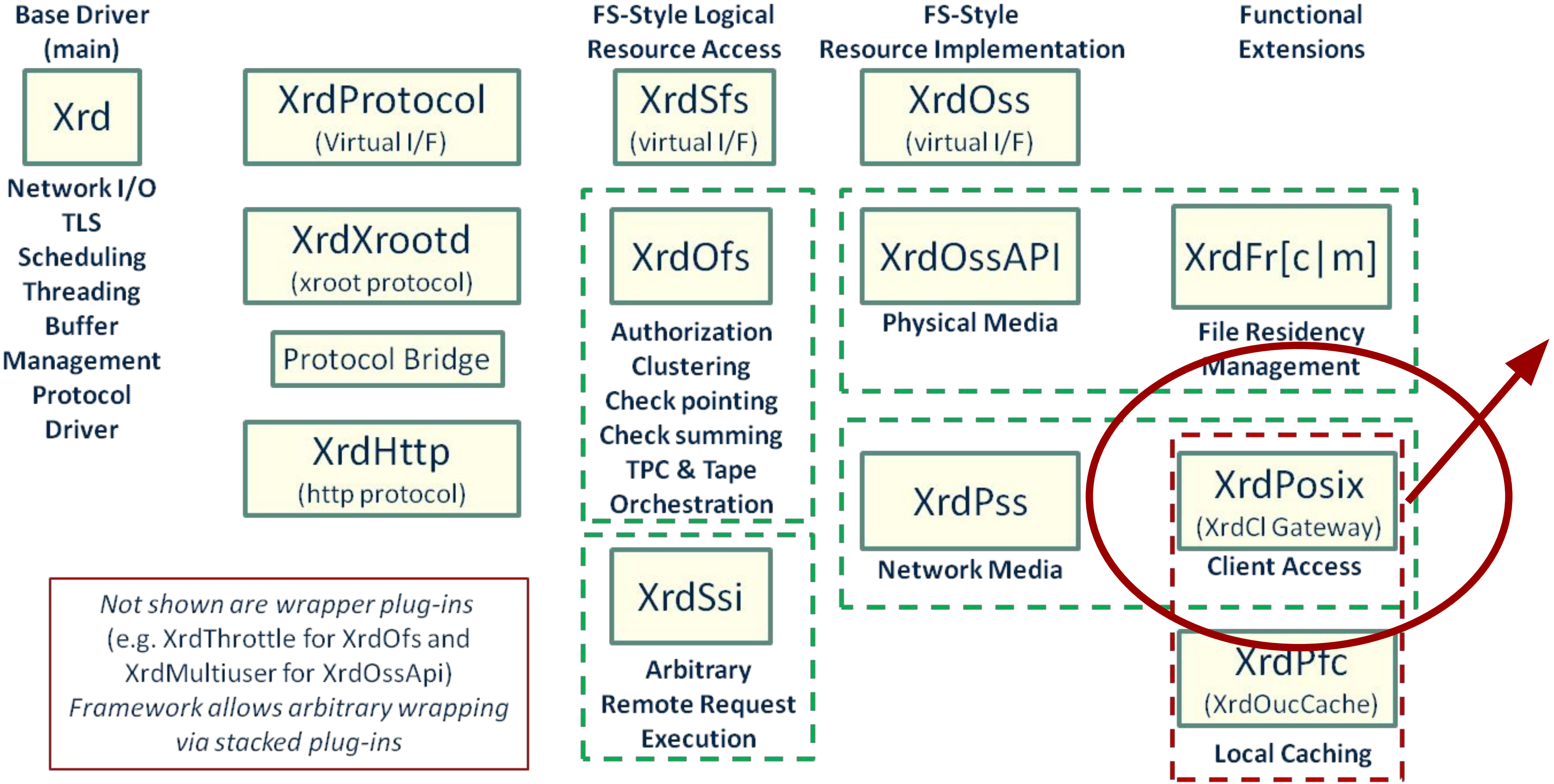
Client Access

XrdPfc (XrdOucCache)

Local Caching

*Not shown are wrapper plug-ins*
(e.g. XrdThrottle for XrdOfs and
XrdMultiuser for XrdOssApi)
*Framework allows arbitrary wrapping*
*via stacked plug-ins*

# XRootD Core – Pelican's XrdCl-http plugin



Base Driver (main)

Xrd

Network I/O
TLS
Scheduling
Threading
Buffer Management
Protocol Driver

XrdProtocol (Virtual I/F)

XrdXrootd (xroot protocol)

Protocol Bridge

XrdHttp (http protocol)

FS-Style Logical Resource Access

XrdSfs (virtual I/F)

XrdOfs

Authorization
Clustering
Check pointing
Check summing
TPC & Tape Orchestration

XrdSsi

Arbitrary Remote Request Execution

FS-Style Resource Implementation

XrdOss (virtual I/F)

XrdOssAPI

Physical Media

XrdPss

Network Media

Functional Extensions

XrdFr[c|m]

File Residency Management

XrdPosix (XrdCl Gateway)

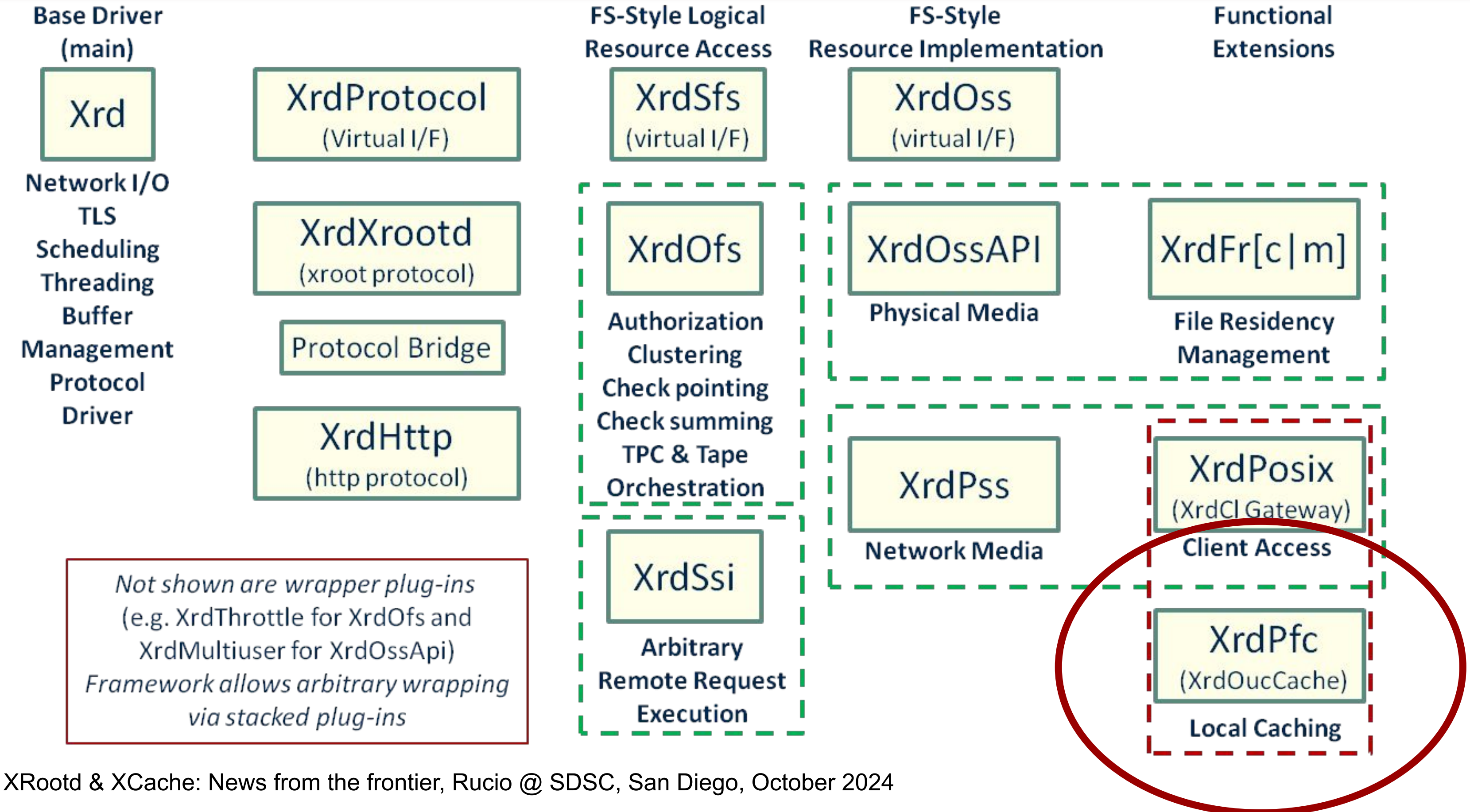
Client Access

XrdPfc (XrdOucCache)

Local Caching

*Not shown are wrapper plug-ins*
(e.g. XrdThrottle for XrdOfs and XrdMultiuser for XrdOssApi)
*Framework allows arbitrary wrapping via stacked plug-ins*

# XRootD Core – XCache – Xrd Proxy File Cache



Base Driver (main)

Xrd

Network I/O
TLS
Scheduling
Threading
Buffer Management
Protocol Driver

XrdProtocol (Virtual I/F)

XrdXrootd (xroot protocol)

Protocol Bridge

XrdHttp (http protocol)

*Not shown are wrapper plug-ins*
(e.g. XrdThrottle for XrdOfs and XrdMultiuser for XrdOssApi)
*Framework allows arbitrary wrapping via stacked plug-ins*

FS-Style Logical Resource Access

XrdSfs (virtual I/F)

XrdOfs

Authorization
Clustering
Check pointing
Check summing
TPC & Tape Orchestration

XrdSsi

Arbitrary Remote Request Execution

FS-Style Resource Implementation

XrdOss (virtual I/F)

XrdOssAPI

Physical Media

XrdPss

Network Media

Functional Extensions

XrdFr[c|m]

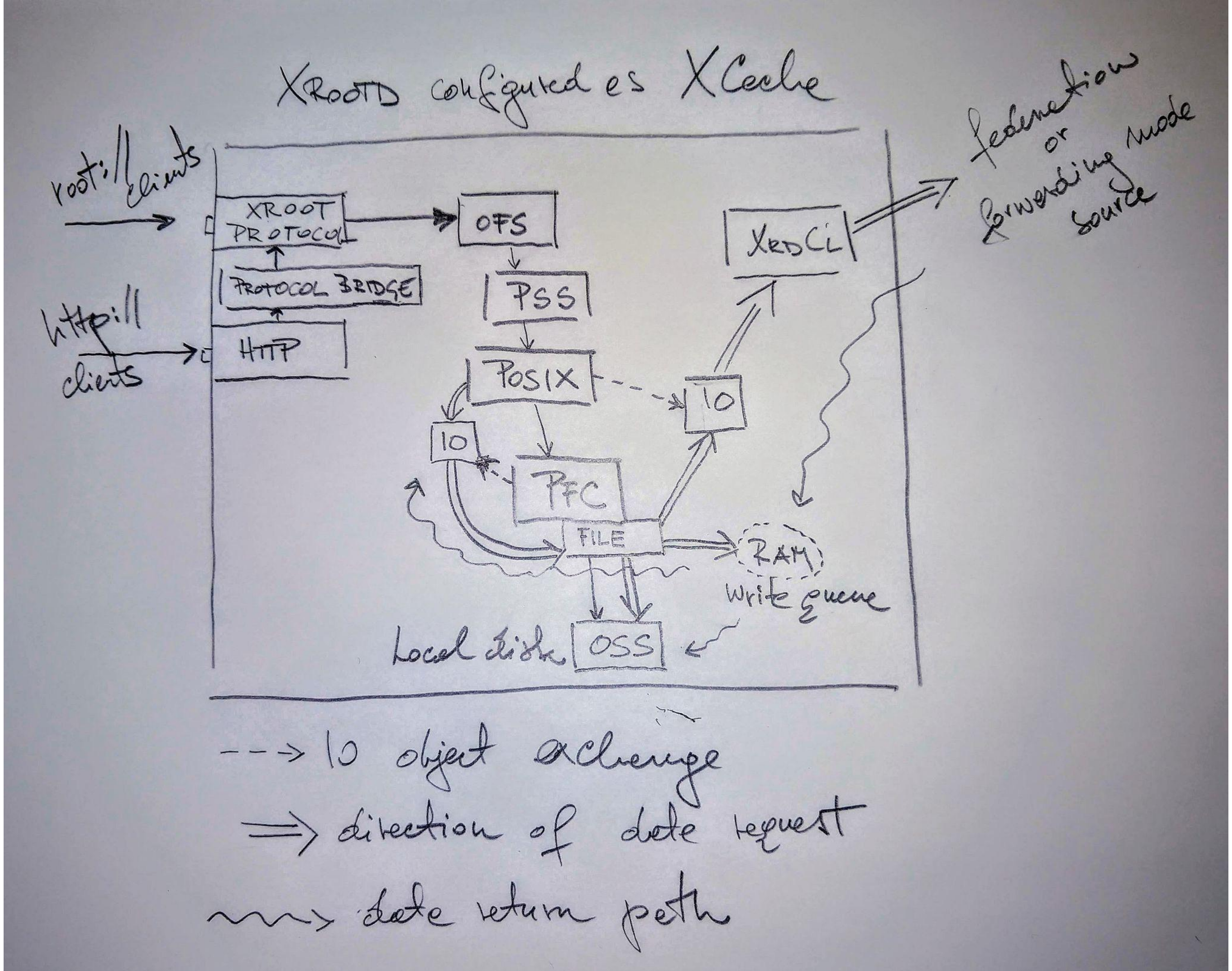
File Residency Management

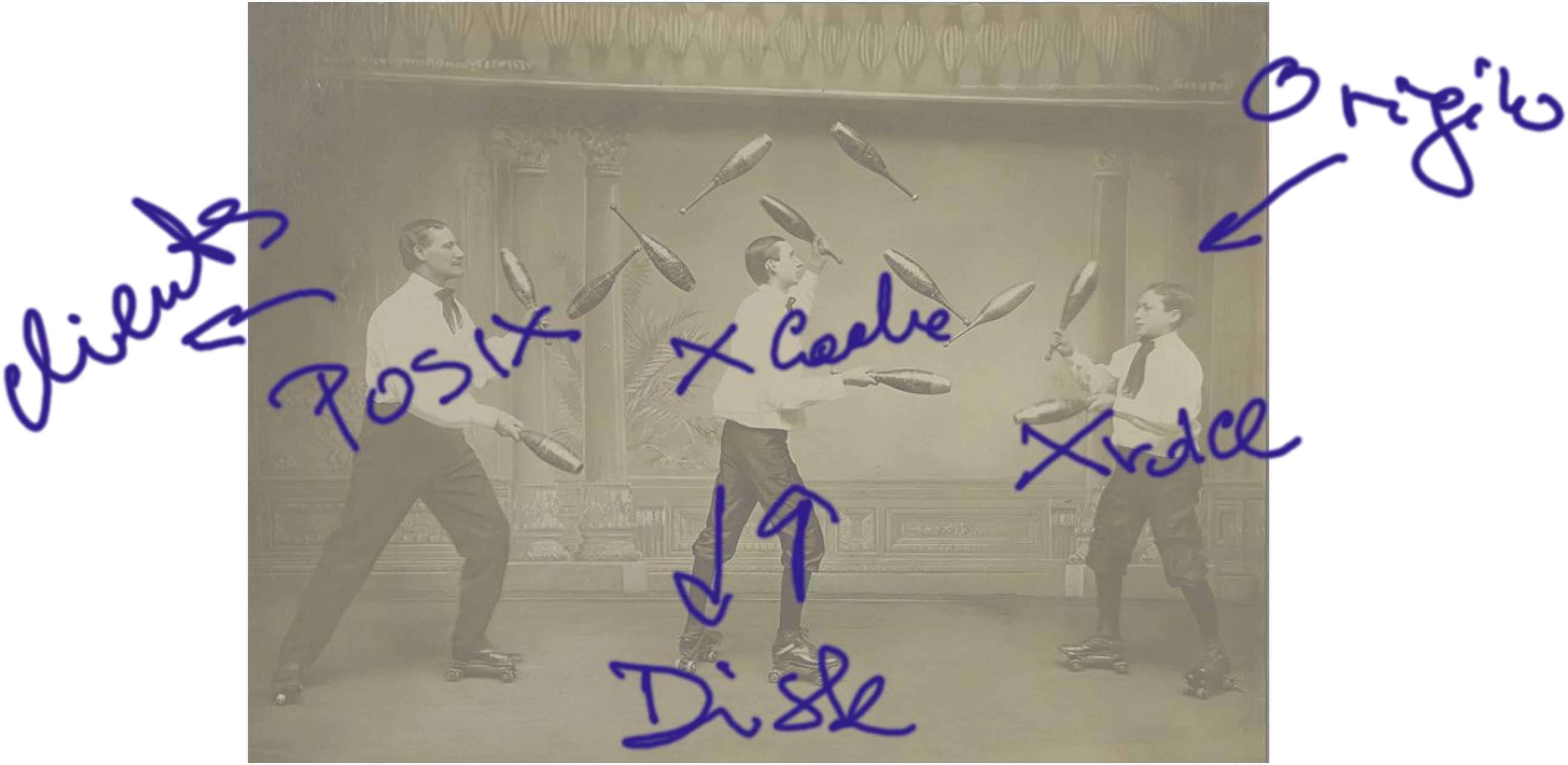
XrdPosix (XrdCl Gateway)

Client Access

XrdPfc (XrdOucCache)

Local Caching


XROOTD configured as XCache
root:// clients
http:// clients
XROOT PROTOCOL
PROTOCOL BRIDGE
HTTP
OFS
PSS
POSIX
IO
IO
PFC
FILE
RAM
write queue
Local disk
OSS
XrdCl
federation or forwarding mode source
---> IO object exchange
⟹ direction of data request
~~> data return path


clients
POSIX
Xcache
Xrdcl
Disk
Origin

# Notes on XCache

- XCache is fully asynchronous, three ways
- The hardest setup XrdCl is exposed to
  - O(1k - 10k) open files – this is way beyond what would be seen in, say, EOS-FUSE
- XCache is magick – but the hardware it is running on isn't
  - Limited by network and, most often, disk I/O limits
  - There are many things one can tune:
    - block sizes, numbers of various threads, …
    - disk – use OSS LVM / space management to use disks independently
  - This all depends on the actual use-case:
    - full-file, burst mode (as in OSDF / Pelican)
    - partial-file, relatively slow direct reading of data directly from a lot of jobs (CMS)
    - XCache as buffer for large-block-based storage (e.g., fronting Ceph object store)

# What XRootD can do for you

# Relevance of XRootd in scientific communities

- Own the protocol, server and client implementations
  - Gives one full control … but potentially complicates deployment and operations.
    - XRootd is quite easy to deploy and operate … secure - should not get flak from security folks
  - Can interface to all relevant storage solutions – or can be extended with some sweat
  - XRootd and XrdCl (and therefore also XCache) all speak HTTP(S)
    - Providing an easy way to mingle with existing data sources & clients
- Data / storage access is something that can easily go wrong, in many ways
  - Fewer and fewer of us live in the world of dedicated Tier-[012] centers
    - or … of over-provisioned dedicated resources that let you get away with squander
    - some data-servers might really not like how your jobs access the data
  - Access through XRoot, with appropriate plugins, gives you predictable access patterns
    - Reduces load on storage
    - Can provide meaningful caching in the job / node context
  - Never underestimate the havoc your users can unleash upon your resources.

# XRootd & RUCIO

- Presumably, Rucio is managing your data, placing it where you want it.
  - [ XRoot might even be involved in some of that business ]
- Then, this data needs to be accessed from your jobs
  - Local jobs: direct reading off a shared system might work (see previous slide)
  - Nearby jobs: for smallish files and modest job inputs … one might be able to get away with the *"copy input to job"* paradigm
  - This, however, is often not the case … and is becoming less and less of the case:
    - Data files are growing large (e.g., LHC: 2–4 GB → 10–20 GB)
    - By consequence, partial file reads are becoming increasingly important
    - Jobs often require several inputs, e.g., calibration, conditions, simulation data, backgrounds
      - many of which are read extremely sparsely
    - Golden age of *"all your network are belong to us"* are pretty much over
  - HPC resources, some with very strict policies, are being gently pushed down our throats
  - Campus resources have limited connectivity / access to non-cluster-local storage
    - E.g., UCSD Physics T2 .vs. SDSC

# XRootd as the last 100-mile solution

- The Pelican project is, in a sense, a response to this realization
  - Providing an easy bridge for the newcomers to cross into the distributed data world
  - Using XCache for just-a-bit-before-time data-placement close to where jobs will run …
  - … but still uses / promotes *"copy data to the job"*
- Solution to problems on the previous page really is to use direct, partial reads from the nearby storage or cache.
  - XRootd ecosystem gives you means to do that on your I/O layer, at a relatively low cost
  - High read-rates can be compensated with prefetching (up to a point)
    - And predictable / tuned reads can improve even local storage access
  - Problems of course still exist:
    - Integration with WMS / jobs execution – getting data access points into the job
    - Giving jobs access to your (remote) storage – which HEP has been doing since ~ever

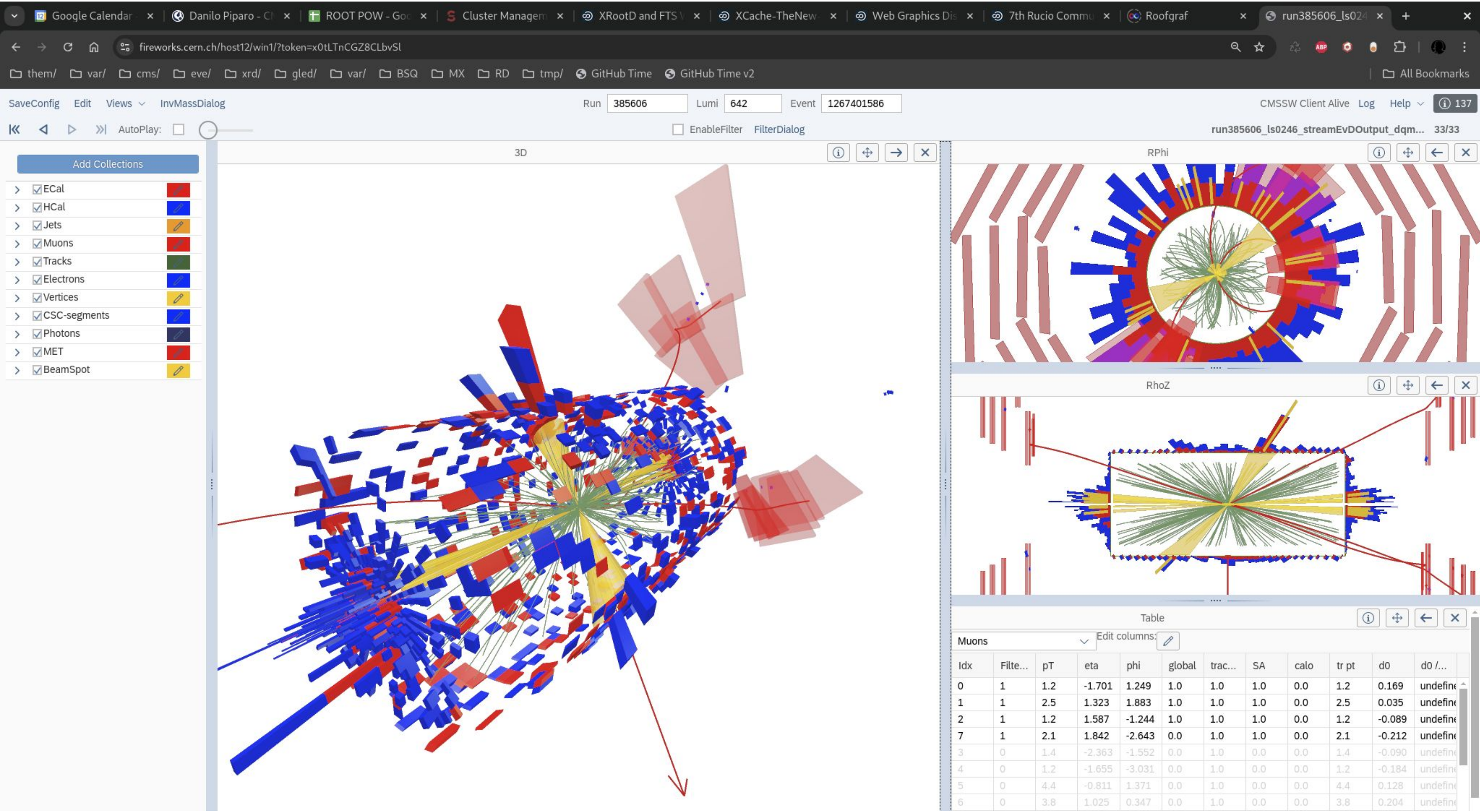
The following are some highlights from a collaboration that went XRoot all the way

# CMS: ROOT & XRootD

- File format: ROOT TTrees
  - EDM (Event Data Model):
    - classes for data storage with dictionaries
    - File format / TTree structure that represents the data / collections in columnar format
    - Lightweight version of the framework, FWLite, for use with plain ROOT
- CMSSW framework – custom Storage API – wrapper over data-sources
  - *RootFile / RootTree / XrdAdaptor* for reading via XRoot
    - Uses the knowledge of ROOT TTree file structure and XrdCl C++ bindings
    - Does partial reads, auto-detection of used branches and precise prefetching
    - Multi-source with source blacklisting
- Final user analysis uses simpler ROOT files these days – plan ntuple TTrees
  - NANOAOD and custom derivations and deviations + truly custom analysis formats
    - the latter, might not even be ROOT – depending on the required sample size
- Event size depends on the data tier: from 4 MB down to 1.5 kB for NANO

# CMS: XRootd in action

- AAA – Any data, Anytime, Anywhere– Global XRootd federation for CMS
  - All storage is exported through a three layer redirector structure:
    - global, regional (US vs. EU), site-level
  - LFN is all you need to access a file
- Every job or user can access all the data – so one can do:
  - Job overflow (too many jobs queued on a site) & fallback (local file open fails) to remote open
  - Run with zero storage setup for opportunistic sites – just read from AAA (or nearby T2 / T1)
  - Several workflows use remote reads exclusively, e.g.:
    - Re-Reco – raw data is available at high-quality sites, e.g., Fermilab T1, low read-rate
    - Pile-up event mixing: very large files, random min-bias samples pulled in by the jobs
- Certain analysis jobs can be trickier
  - High filtering rate, high processing rate (little compute per event)
    - *RootTree* is smart enough to detect the primary / high-rate branches – prefetching!
  - XCache – pin a sub-set of namespace to a specific site / neighborhood
    - Somewhat tricky setup between site-config and WMS / Crab



Run 385606 Lumi 642 Event 1267401586

run385606_ls0246_streamEvDOutput_dqm... 33/33

- ☑ ECal
- ☑ HCal
- ☑ Jets
- ☑ Muons
- ☑ Tracks
- ☑ Electrons
- ☑ Vertices
- ☑ CSC-segments
- ☑ Photons
- ☑ MET
- ☑ BeamSpot

Muons

| Idx | Filte... | pT | eta | phi | global | trac... | SA | calo | tr pt | d0 | d0 /... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1.2 | -1.701 | 1.249 | 1.0 | 1.0 | 1.0 | 0.0 | 1.2 | 0.169 | undefine |
| 1 | 1 | 2.5 | 1.323 | 1.883 | 1.0 | 1.0 | 1.0 | 0.0 | 2.5 | 0.035 | undefine |
| 2 | 1 | 1.2 | 1.587 | -1.244 | 1.0 | 1.0 | 1.0 | 0.0 | 1.2 | -0.089 | undefine |
| 7 | 1 | 2.1 | 1.842 | -2.643 | 0.0 | 1.0 | 1.0 | 0.0 | 2.1 | -0.212 | undefine |
| 3 | 0 | 1.4 | -2.363 | -1.552 | 0.0 | 1.0 | 0.0 | 0.0 | 1.4 | -0.090 | undefine |
| 4 | 0 | 1.2 | -1.655 | -3.031 | 0.0 | 1.0 | 0.0 | 0.0 | 1.2 | -0.184 | undefine |
| 5 | 0 | 4.4 | -0.811 | 1.371 | 0.0 | 1.0 | 0.0 | 0.0 | 4.4 | 0.128 | undefine |
| 6 | 0 | 3.8 | 1.025 | 0.347 | 0.0 | 1.0 | 0.0 | 0.0 | 3.8 | 0.204 | undefine |

# FireworksWeb – CMS physics analysis event display

- Event Display as a Service
  - CMS members can access dedicated servers at CERN & UCSD
  - Access data from eoscms, AAA (through XCache), CERNBox *(share with cms-vis-access)*
  - Proto-app with preloaded data-formats forks off an instance (fast!, can serve multiple users/tabs)
- Uses FWLite to only read the required collections
  - Configures ROOT TTreeCache to do the prefetching
  - Use XCache to store file fragments locally
- A typical use-case:
  - A 4 GB file with 10k+ events
  - The user wants to only view events with, say, N_muons >= 2 && pT_M1 > 50 GeV …
    - The filter only needs to read two columns out of O(5000)!
    - Efficient & fast!
  - … and then display O(10) collections out of O(100) available in the file.
- Now being extended to also show CMS Open Data.

# New stuff & plans

# Overview of stuff that's happened over last year++

- Detailed reports from XRootd & FTS Workshop one month ago:
  - *Catching Up With XRootD I. & II.:* Andy's and Guilherme's talks
  - *Contributing to XRootD:* News about the repo, build system, CDash, github-actions
- Highlights from 5.6 and 5.7 release cycles
  - XCache: eviction API, use local FS as origin (front DFS that hates small reads)
  - Performance: default # of event loops for proxies, avoid OpenSSL-3.0 performance hit, …
  - Tokens: various extensions / fixes / improvements from real-life cases
  - Security: use at least SHA-256, require min 2048 RSA keys
  - Improved HTTP conformance
  - Support musl libc
  - Bump to C++-17
  - Improved / modernized build and CI system, github & friends
- A lot of development happens in plugins!
  - E.g., XrdCeph is now back in the main repo, S3 reader developed as part of Pelican, …

# XRootd development trajectory

- Current stable release: 5.7.x
  - No major outstanding issues or requests from the community
    - → no pressure for new releases
  - 5.8 planned as XCache Resource Monitor & Purge plugin feature release
    - Driven by the Pelican requirements
    - Includes standalone directory quota manager
- 6.0 is well underway, planned for late '24 / early '25
  - Major version change is the only time we can break ABIs of components.
    - XRootd is available natively for all major GNU/Linux distributions & MacOS
      - Their library packaging rules are very strict about ABI compatibility.
  - Improved error messages and scalability enhancements
  - Planning additional features in security and token handling
  - Enhanced Ceph support as an object storage system
  - Finally dropping Python-2 & CentOS-7 support
  - Move to C++-20

# XRootD Core Plans

- For future work we focus on HL-LHC and community needs
  - Significant work to further improve performance
    - Using **io_uring** *(kernel level async I/O)* where possible
    - Using **kTLS** *(kernel-level TLS)*
    - Provide RDMA capabilities for data transfers
    - Integrate SDN support (Software Defined Networking, Sense Project)
  - We are always gathering community needs and feeding them into the development plan
- OSDF / Pelican is a strong driver for XCache & XrdCl-http development
- Additional XCache plans:
  - improve prefetching strategies … auto-detect burst and slow-read modes
    - consider file structure hints – relevant for non-root data formats

# Closing words from our release manager

- **XRootD is a core component of HEP software ecosystem**
  - Depended on by CTA, FTS, EOS, ROOT, Rucio, experiment frameworks, etc
- Exabytes of data processed each year (including CERN LHC Tier0 operations)
- **Needs security, stability, scalability, sustainability & performance**
  - Code scanning (CodeQL), security policy setup on GitHub (allows private bug reporting)
  - Continuous effort to improve testing infrastructure
    - Measure and expand test coverage, use static analysis tools, automatic testing in CI
  - Performance analysis of production workloads to guide performance optimizations
  - Lower barrier for contributors and users as much as possible
  - Make it as easy as possible to configure, build, run tests, and create packages
- Consider this an invitation to try it … or at least think how you could use it.

# Getting in touch with the XRoot / XCache crew

- XRootd developers + community of main users:
  - Weekly *xcache-devops* meeting (Thursday 11am Pacific)
    - OSG, Pelican, ATLAS, CMS + others, as needed – or desired
    - *xcache@opensciencegrid.org*
    - slack *OSG#xcache*
  - Advise, improve existing features, develop extensions
  - Help with debugging, analysing issues

- General user / developer support
  - Ask questions: *xrootd-l <[xrootd-l@slac.stanford.edu](mailto:xrootd-l@slac.stanford.edu)>*<br>*[https://github.com/xrootd/xrootd/discussions](https://github.com/xrootd/xrootd/discussions)*
  - Report problems: *[https://github.com/xrootd/xrootd/issues](https://github.com/xrootd/xrootd/issues)*

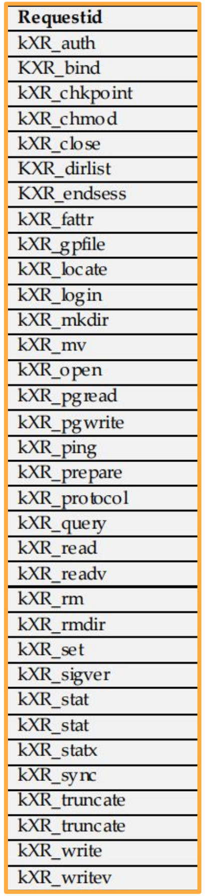
- New: a yearly XRootd & FTS Workshop, the 2nd week of Sept.

# Gentle Introduction to XRoot
by Guilherme

XRootD

# Guilherme – for Kernel Recipes '24: EOS via FUSE

- XRootD is a system for scalable cluster data access
- Initially developed for BaBar experiment at SLAC (~2002)
  - The Next Generation ROOT File Server
- Written in C++, open source (LGPL + GPL)
- Available in EPEL and most Linux distributions
- You can think of XRootD as nginx + curl + varnish
  - Besides HTTP it also supports the in-house XRoot protocol `root://`
    - a stateful, POSIX-like protocol for remote data access
  - TLS (`roots://` and `https://`) support since XRootD 5.0
    - Supports TLS for control channel only, or control + data channel
- Can be configured as proxy / caching server – XCache
- Authentication via Kerberos, X509, shared secret, tokens
- Not a file system & not *just* for file systems

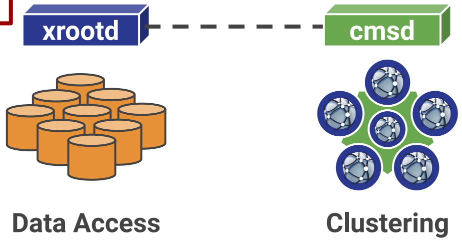


| **Requestid** |
|---|
| kXR_auth |
| KXR_bind |
| kXR_chkpoint |
| kXR_chmod |
| kXR_close |
| KXR_dirlist |
| KXR_endsess |
| kXR_fattr |
| kXR_gpfile |
| kXR_locate |
| kXR_login |
| kXR_mkdir |
| kXR_mv |
| kXR_open |
| kXR_pgread |
| kXR_pgwrite |
| kXR_ping |
| kXR_prepare |
| kXR_protocol |
| kXR_query |
| kXR_read |
| kXR_readv |
| kXR_rm |
| kXR_rmdir |
| kXR_set |
| kXR_sigver |
| kXR_stat |
| kXR_stat |
| kXR_statx |
| kXR_sync |
| kXR_truncate |
| kXR_truncate |
| kXR_write |
| kXR_writev |

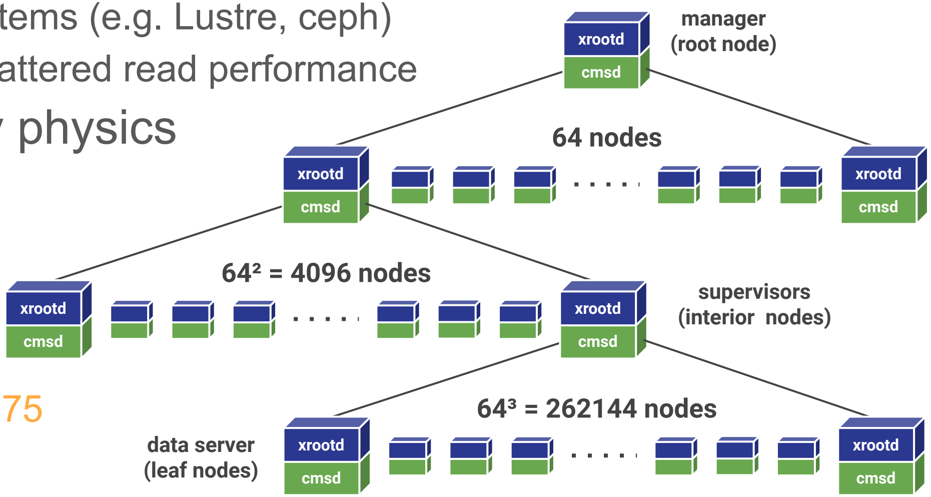
M. Tadel, XRootd & XCache: News from the frontier, Rucio @ SDSC, San Diego, October 2024

XRootD

# Guilherme – for Kernel Recipes '24



- XRootD clustering has many uses
  - Creating a uniform namespace, even though it is distributed
  - Load balancing and scaling
    - Proxy servers and caching servers (XCache)
    - Serving data from distributed filesystems (e.g. Lustre, ceph)
    - Ceph + XCache good to improve scattered read performance
- Wide deployment across high energy physics
  - EOS@CERN, CMS AAA Data Federation
- Highly adaptable plug-in architecture
  - If you write a plug-in, you can cluster it
  - Used by LSST Qserv, clustered MySQL
    - https://inspirehep.net/literature/716175
- Extensive support for monitoring



M. Tadel, XRootd & XCache: News from the frontier, Rucio @ SDSC, San Diego, October 2024