
CERN

Michal Simon

Modernizing xroot protocol

Outline

- Introduction
- Ongoing developments
  - Vector Writes
  - Extended Attributes
  - Encryption
  - Bundled Requests

XRootD Framework

- Distributed **low-latency file access** system
- Originated at **SLAC** (Stanford)
- It is the **data access framework of choice for High Energy Physics** community
- **Backbone of EOS**, the main storage solution used at **CERN** (**250 PB** of raw disk space, access to over **1.8 billion files**)

XRootD Framework

- **Multithreaded C++ client/server framework**
  - scalable
  - very **stable** (high quality code base)
- Hierarchical filesystem-like namespace
  - storage clustering with **hierarchical redirections**
- **Plug-in** based architecture
- **Authorization / authentication** (X509, Kerberos, etc.)

XRootD Framework

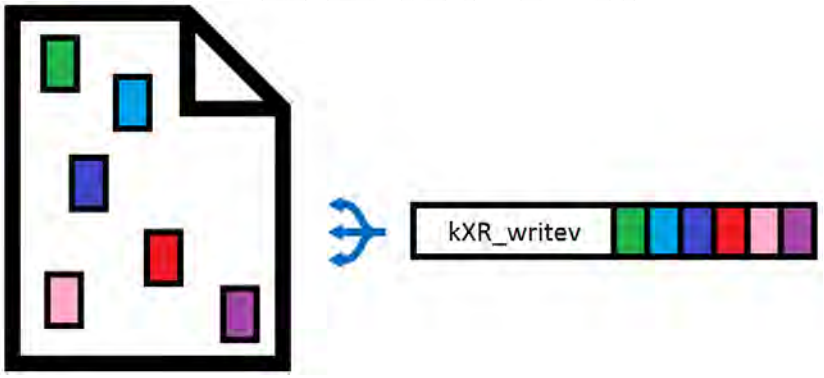
- Native xroot protocol has been designed for **efficient remote file access in LAN/WAN**
  - checksums, vector reads
  - redirections
  - third-party-copy
- Both **synchronous & asynchronous I/O** interfaces for data and metadata
- **Native RPC** mechanism (SSI) and native **caching** utility(XCache)

Ongoing developments

- Goals:
  - **reduced latency** (number of RTTs)
  - **increased security**
  - **simplified programing model** (new APIs)

Vector Writes

- Motivation: **reduce latency while flashing cache** (e.g. fuse mount caching)
- Write scattered data chunks in one operation (within a single file)



- The protocol also supports vector writing to multiple files (in case this is would be needed in the future)

Extended Attributes

- Motivation:
  - catch up with the world (common feature)
  - very convenient for developers
  - metadata (checksum, version, stripe index)

- Protocol extensions:
  - new request type for handling extended attributes (**kXR_fattr**)
    - deleting / retrieving / setting
    - support for **batch operations**

Extended Attributes

- Protocol extensions (cont.):
  - query request allows to retrieve information regarding fattr support, limits, etc.
- Implementation:
  - a fattr operation needs to be preceded by file open and followed by file close
  - makes access and privilege checking very easy
  - sounds like considerable overhead (3x RTT), so stay tuned for bundled requests

Encryption

- Motivation:
  - additional **protection against malicious attacks** (e.g. man-in-the-middle attack)
  - **symmetry with HTTP protocol**
  - opens XRootD to other communities that require data encryption (e.g. medical data)

- **Does not require protocol changes**, however once in place **sign requests will be deprecated**

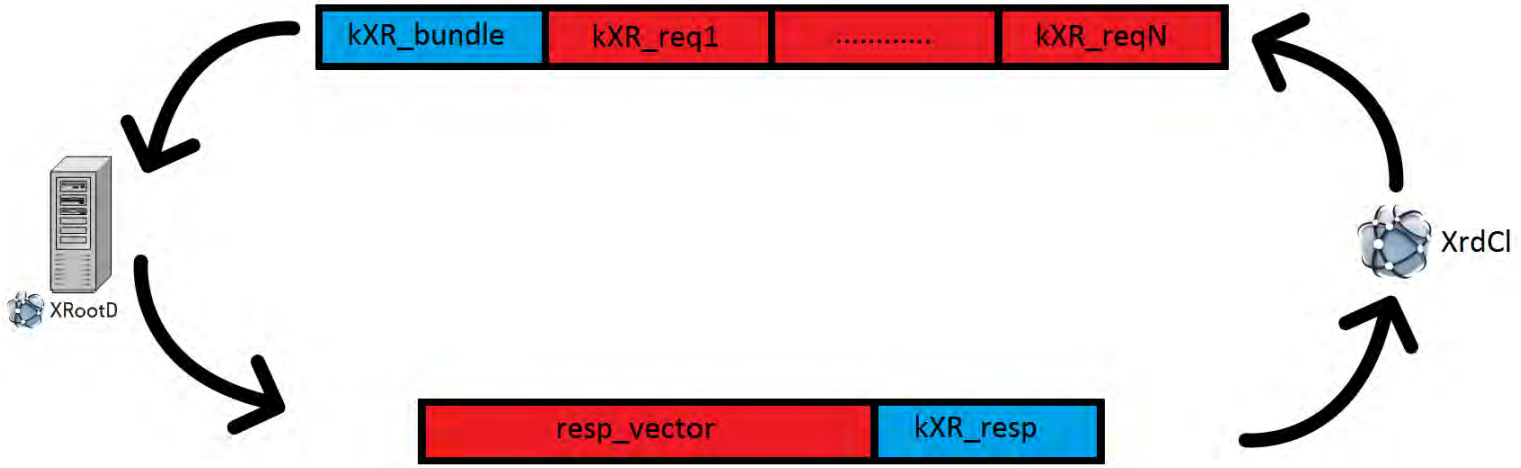
Encryption

- Implementation:
  - TLS tunneling based on **openssl async API and an event-loop**
    - fully async I/O no cheating
  - **ABI compatible**
  - **redirections** (or equivalent) **from unencrypted to encrypted** and vice versa
    - does not require additional login/authentication

Bundle requests

- Motivation
  - reduce the number of RTTs
  - provide more modern APIs for the users

- Protocol extensions
  - new request type for bundling other requests



Bundle requests

- Use cases:
  - extended attribute operation = open + fattr + close
  - extended stat = open + stat + fattr + close
  - object store like GET = open + read + close
  - object store like PUT = open + write + close

- **Modern C++ API**

  …

  workflow(open | read >> handler | write | close >> handler)

  …

Useful Links

- http://xrootd.org
- https://github.com/xrootd/xrootd.git
- http://storage-ci.web.cern.ch/storage-ci/xrootd/experimental/
- xrootd-dev@slac.stanford.edu

Questions?