CERN

Michal Simon

XRootD5 landscape

Outline

- Introduction
- Adoption
- Encryption: the recap
- Paged read / write
- K8s
- Erasure coding
- Miscellaneous
- R&D

XRootD: the team

Server / OFS / OSS

- Andy

Client / EC / Python

- Michal

CMake / packaging / CI

- Michal

TLS

- Andy, Michal

XCache

- Matevz

xrootdfs

- Wei

HTTP protocol plugin

- Fabrizio

HTTP TPC plugin

- Brian, Cedric, Elvin

GSI authentication plugin

- Gerri, Michal

EPEL / Debian packaging

- Mattias

XrdOssCsi

- David

SciTokens plugin

- Derek

Ecosystem

XRootD is both a protocol and a framework (~400k lines of code) for low latency file access and as such is a **key component of many projects**:

- Software defined storage: **EOS** (our favorite customer :-), **CTA**, and many others
- **100s of XCache** deployments (USDC, RAL, NERSC, GSI, OSG, and many more)
- **HTTP TPC Proxy** for EOS (again, our favorite customer :-)
- GRID access and transfers: **JAliEn**, **FTS/gfal2**, Rucio
- Analysis: **ROOT**, Gaudi, Athena, CMSSW
    - uproot (scikit-hep, numpy), snakemake
- **xrdcp** / **xrdfs** (lxplus / lxbatch)

Adoption

XRootD5 adoption

- **EOS5 already released,** tailor made features:
  - redirect collapse (facilitates HA setup)
  - better error on write recovery (allows to recover almost all errors at MGM)
- **ROOT** moved their builds to R5
  - Implemented **root/roots support in RNTuple**
- **Alice**, all known **XCache** instances, RAL, **EPEL** (e.g. DPM)
- lxplus / lxbatch

Releases

Since last workshop we had:

- Three **feature releases:** 5.2.0, 5.3.0 and 5.4.0
- Five **bugfix releases:** 5.3.1, 5.3.2, 5.3.3, 5.4.1 and 5.4.2

Packages available in:

- Extra Packages for Enterprise Linux (**EPEL**) and Fedora
  - EPEL 7/8/9
- **Debian** (also available on Ubuntu)
- **PyPI** (many enhancements in the area of Python packaging)

Source code available at: https://github.com/xrootd/xrootd

- gitlab.cern.ch used for CI

Encryption: the recap

- On the client side the **roots/xroots** protocol;
  - --**notlsok** options allows to proceed without encryption if the server is too old to support it
  - --**tlsmetalink** option allows to apply encryption to all URLs in a metalink file

- On the server side the **xrootd.tls** configuration directive, with few compatibility options:
  - by default it is **off**
  - enforce encryption only for clients that support it (**capable**)
  - do encryption only at client discretion (**none**)

How flexible is it?

- **Encrypted and unencrypted traffic uses the same port** number (not like http vs https) to ease operators lives

- One can configure the server to encrypt:
  - only the **third-party-copy** orchestration
  - **control channel** after login (handy for GSI auth)
  - control channel before login
  - **data streams**
  - everything

- On the client side:
  - **--tlsnodata** allows to apply roots/xroots only to the control stream

Certificates, certificates, …

- XRootD server needs a host certificate in order to enable encryption
    - configurable with **xrd.tls** directive
- If roots/xroots is being used client will **enforce host verification**
    - the hostname must match the one in the host certificate (or one of the SAN extensions)

Certificates, certificates, …

- The client does not need to have a certificate
  - the user **may use his proxy certificate** in order to establish a TLS connection
  - server can be configured to enforce client certificate verification with: **xrd.tlsca**

- Allowing the client to establish the TLS connection based on user X509 proxy certificate opens door to a new **more concise implementation of gsi authentication** in the future

Paged read / write

- Detect and repair ‘in-transit’ data corruption with **4KB level of granularity**
  - **Hardware assisted crc32c per 4KB page**(throughput in order of ~10GB/s per core)
    - Hamming distance 6
  - Corrupted pages are **automatically resent**

Paged read / write

- **Critical for the XCache use-case**
  - All ingest happens with ***root* protocol via XRootD client**
  - **Boosts data integrity** (corrupted data tend to be sticky)
- **In-the-flight error recovery in xrdcp**
  - Strategic for big file (e.g. 100 GB) transfers
  - Throughput of **1.25 GB/s per stream** (optimized for aggregate throughput)

K8s support

- **Virtual network overlay**
  - Namespace where each node has an internal name
  - Use case: **allows cmsd in a XCache cluster to track file location by dependable name**
    - Does not relay on IP address or hostname
- **Dynamic DNS**
  - Hostnames are available in local DNS only if container is up
  - **Resolve IP addresses at time of contact** and not during initialization
- **Network namespaces**
  - Accommodate K8s network namespaces

Erasure Coding

- The EC module has been originally designed for EOS, now it is also **compatible with vanilla XRootD servers**
  - No need to have separate metadata file
  - Store additional information (i.e. file size) in extended attributes
- Staring with 5.2.0 XRootD comes with **default erasure coding plugin**
  - The plug-in can be loaded either by
    - **special redirect request** (generated by MGM)
    - standard **plug-in configuration file** (EC proxy)

Miscellaneous

- Atomic **ZIP append** (append new files to archive)
    - Checkpointing mechanism that ensures **atomicity**
        - Checkpointed write / rollback / commit
    - Atlas use case: **merge log files**
- Reproxy option for proxies doing TCP
    - Use case: **enable FTS performance markers with EOS TPC gateways**
    - Makes sure the proxy server forwards stat requests against data servers and not the head node (MGM)

Miscellaneous

- **S3 gateway**
  - Used in US in front of **Google Cloud**
  - XRootD proxy + **client HTTP plug-in** (based on Davix)
- Packet marking (experimental)
  - Based on Firefly protocol
- **Access tokens**
  - **ZTN authentication** protocol
    - Verify that client is **capable of obtaining a valid token**
    - Enforces **encryption**
  - SciTokens authorization plug-in

For developers

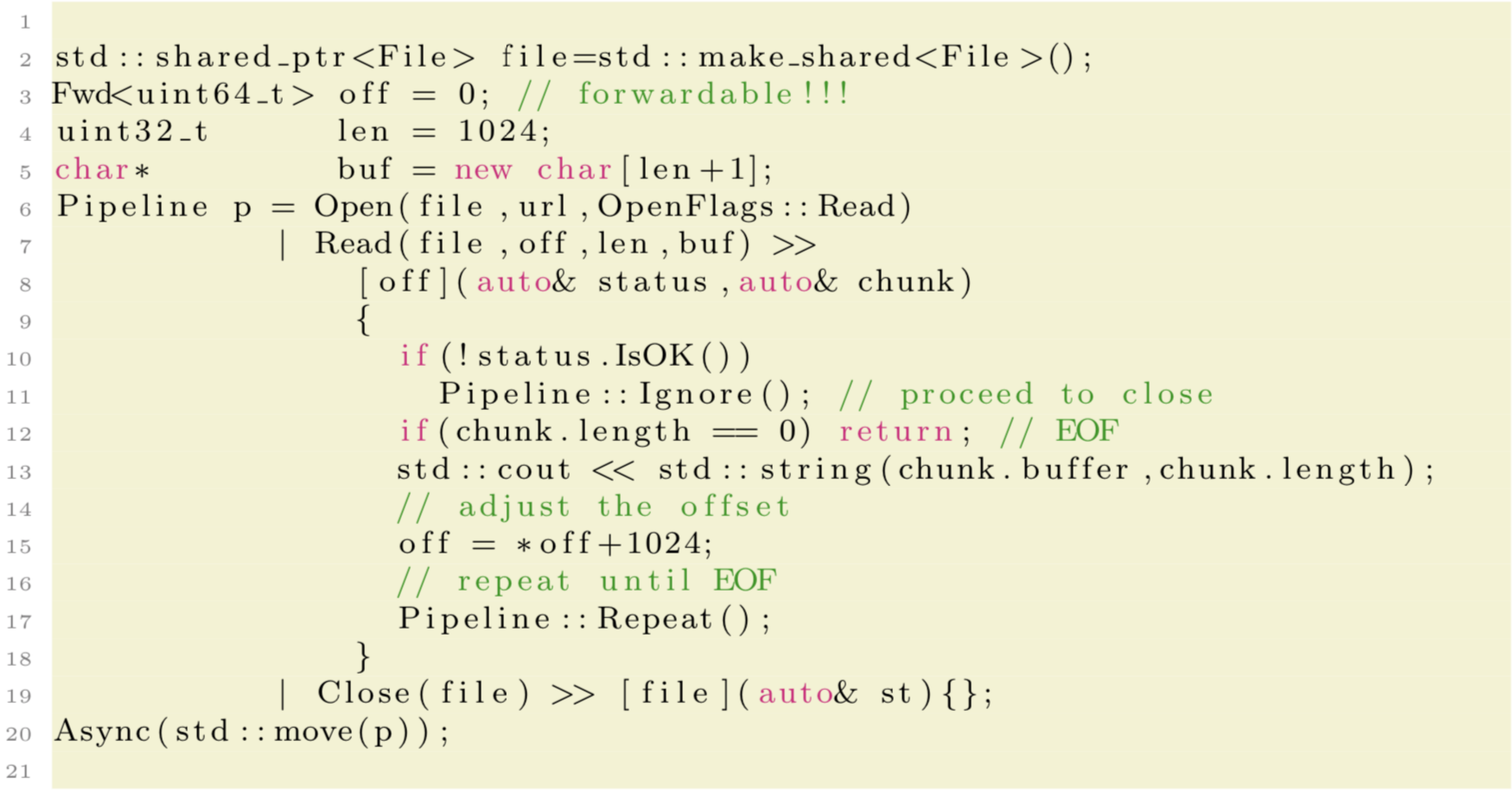
Server:

- Server side **plug-in stacking** with `++` directive
  - User plugin gets a pointer to the level-up plugin so it can call it’s implementation

Client:

- Automatically generate completion handlers from lambdas
  - *ResponseHandler::Wrap(…)*

Client declarative API



```

std::shared_ptr<File> file=std::make_shared<File>();
Fwd<uint64_t> off = 0; // forwardable!!!
uint32_t      len = 1024;
char*         buf = new char[len+1];
Pipeline p = Open(file, url, OpenFlags::Read)
           | Read(file, off, len, buf) >>
               [off](auto& status, auto& chunk)
               {
                 if(!status.IsOK())
                   Pipeline::Ignore(); // proceed to close
                 if(chunk.length == 0) return; // EOF
                 std::cout << std::string(chunk.buffer,chunk.length);
                 // adjust the offset
                 off = *off+1024;
                 // repeat until EOF
                 Pipeline::Repeat();
               }
           | Close(file) >> [file](auto& st){};
Async(std::move(p));

```

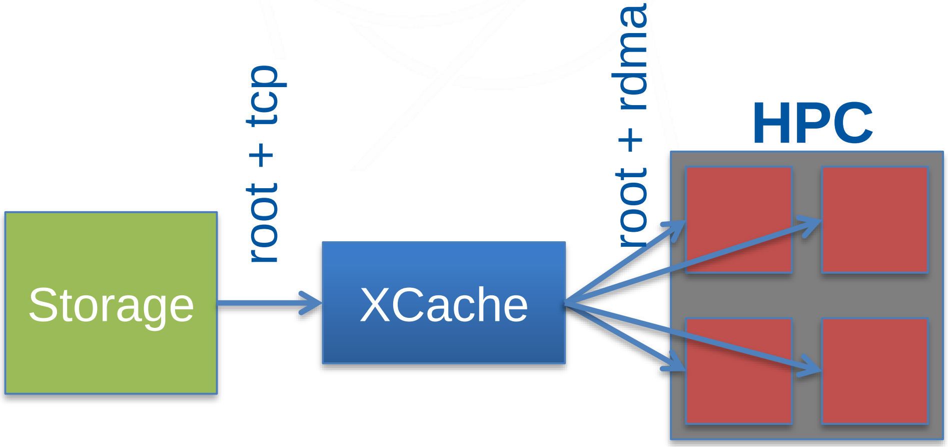
R&D: HPC support

Bind data channels to different address than the control channel

- Allows easier migration from *gridftp* to *root* protocol

RDMA support (R&D)

- Initial prototype **exchanging data over RDMA using libfabric** developed by 2 summer students
- Out of the box solution for accessing data in HPCs (or exporting from DAQs)



Questions?