XrdHTTP from 2013 to today

Fabrizio Furano (CERN IT-ST)

XrdHTTP, since 2012

- In 2012 the “HTTP Ecosystem” project, at the end of EMI
- We were looking for a way to allow pure Xrootd storage sites to interact with HTTP/HTTPS clients and browsers

- **NOTE: we write HTTP, we mean also a pragmatic subset of WebDAV**

- AndyH provided the XrdBridge classes, a sort of internal proxy
- FabrizioF provided the first XrdHTTP implementation to the Xrootd framework, and maintained it so far

XrdHTTP

- On top of simplicity (compared to Apache…) the new component had very interesting features, some were pretty original
  - Some came from the Xrootd framework, others came from the HEP experience

- Successive evolution milestones were contributed by Brian Bockelman, e.g.
  - Correct managing of SSL connection timeouts (https)
  - Added flexibility to the extension modules
  - TPC plugin
  - SciTokens authorization support
  - Macaroons authorization support

- Thanks to the low-level Xrootd disk access implementation, today XrdHTTP is a top performer in the HTTP realm
- With all the features that are needed by Grid/scientific computing
  - e.g. vector reads
- IMO with unmatched **simplicity of configuration and extension through plugins**

Most interesting features

- Port sharing with Xrootd
- Security extractor plugins:
  - e.g. to recognize VOMS proxy certs
- Extendable through plugins
  - e.g. to build real high performance REST services, or override/add the behaviour of requests (e.g. COPY)
- HTTPS to HTTP redirection to disk servers with security token
  - no compromise for performance on small transactions
- HTTPS to HTTP redirection to itself, with security token
  - no compromise for performance on small transactions
  - The DPM core: DOME uses this flavour for its internal REST protocol

Port sharing

- Normally the xrootd proto stays on port 1094
- Also XrdHTTP can go to port 1094, and live together

- The server socket is managed by the internal xrootd guts, Hence Xrootd can
  - analyze the first bytes of the initial handshake
  - understand which protocol the new client speaks
  - activate the right protocol instance, XrdXrootd or XrdHTTP

- The default XrdHTTP port is 1094. Sysadmins can activate HTTP(s) without having to fix firewalls over a previous Xrootd service

```
if exec xrootd
  xrd.protocol XrdHttp /usr/lib64/libXrdHttp.so
fi
```

Security Extractor

- Once a new HTTPS client has connected, XrdHTTP extracts the security info from the connection
  - And fills the internal security structure (XrdSecEntity)
  - However, VOMS certificates have additional info that is relevant for authorizing the access: FQANS (treated as groups). Moreover, validating a VOMS certificate requires special code.

- XrdHTTP's intelligence in filling that can be extended through a "security extractor plugin", with a pretty generic approach

- XrdHTTPVOMS is an example of such plugin. It validates a client that comes with a VOMS proxy and extracts the auth info from it into XrdSecEntity
- Regular clients go through it normally

```
http.secxtractor /usr/lib64/libXrdHttpVOMS.so
```

Security Extractor

- Now the VOMS secXtractor RPM is in EPEL
  - the development builds are done with the rugged DPM build system :-) maybe once a year
- The code is in Gitlab
  - https://gitlab.cern.ch:8443/lcgdm/xrdhttpvoms
- XrdHttpVOMS is an optional component linking to libvoms2 (EPEL, C++)
- Would be more coherent if it became an optional component residing in the Xrootd source tree, like others
  - Of course built conditionally by the Xrootd CMakefiles

XrdHTTP is extendable

- Add new commands, or substitute the default implementations with new ones
  - DPM uses this (more details in the DPM presentation today) for its DOME core
  - The performance one can get from this approach is only comparable to a native Apache plugin
  - Writing an extension to XrdHTTP is quite simple. Writing an Apache module is difficult

- Brian Bockelman used extensively this, and proposed and made several enhancements to this simple API
  - e.g. now this API can also transfer large data, which was not in the original implementation, oriented only to commands
  - Brian's extension for Third Party Copy (TPC) is a good example of such extensions, AFAIK used in OSG
- Andreas Peters used this to use XrdHTTP within EOS. This includes the OwnCloud support and its particularities

Redirectors

- An XrdHTTP redirector is simply XrdHTTP loaded by an Xrootd redirector
- It will produce … HTTP 30X redirections :-) for clients accessing data
  - Normal redirections: HTTPS->HTTPS
  - HTTPS->HTTP+security token
    - encrypts in a security token the security information
    - enhances the performance for small transactions with persistent conns
    - The DPM REST intf (DOME) uses that, the difference is 3-4 times

Vanilla Xrootd HTTP redirectors

- **Vanilla** Xrootd redirectors cannot provide listings. Only data servers produce meaningful (yet local) listings in the xrootd typical cluster
- Hence also XrdHttp in a **vanilla** redirector cannot produce listings
  - Better … a vanilla Xrootd redirector leads to partial listings, unless special Xrootd components are configured [does XrdCns still exist?]
  - pointing a browser to a data server directory will work (limited to that server)
  - pointing a browser to a redirector will generally not produce the expected content

- For HTTP this is a limitation, as pointing a browser to the storage is very useful and friendly, and it's a feature everyone expects

- Hey I wrote **vanilla**! Xrootd redirectors in other systems (e.g. DPM or EOS or others) do produce 100% valid listings in the redirector
- Also the dCache implementation does provide the expected output

The future

- Today XrdHTTP is quite complete and works amazingly well
- I also find it much simpler to setup than Apache. Once you have a working xrootd you almost have a working XrdHTTP

- One of the next items it may need is an OpenID-Connect implementation, depending on the choices of the WLCG Auth WG
- It's unclear to me if the internal XrdSecEntity will be descriptive enough for that
- Will need to manage arbitrary attributes from OIDC and allow the authorization to decide based on their content (e.g. the email address coming from Google/Facebook/CERN, …)