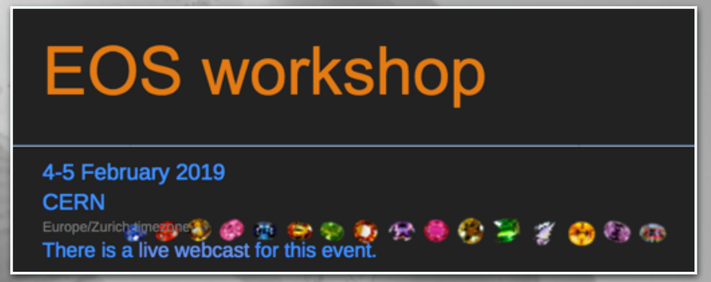
EOS Open Storage

HTTP(S) based APIs for EOS



http://eos.cern.ch

Andreas-Joachim Peters
CERN IT-ST

OVERVIEW

- HTTP(S) server in EOS
- GRPC server in EOS
- REST APIs in EOS

Introduction

- **HTTP(S)** is industry standard and a vast majority of languages provides many client implementations to access HTTP(S) based servers
  - WLCG initiative to establish as mainstream access protocol since many years
- **REST** interfaces have become a standard for service communications
- **GRPC** is an HTTP based **R**emote **P**rocedure **C**all implementation provided by GOOGLE using Google protocol buffers (simple on the wire data representation - multi language)
  - GRPC would be a perfect match for the new micro service architecture in CERNBOX

HTTP(s) in EOS today

- HTTP(S) access is implemented in EOS using a patched **libmicrohttpd** and a patched **NGINX** module
  - reasons:
    - interference of XRootD & libmicrohttpd EPOLL loops
    - NGINX sends immediately PUT body to MGM (does not use 100-CONTINUE)
- HTTP server has separate thread pool and runs on dedicated ports (default MGM 8000, FST 8001)
- HTTP(S) is only provided via NGINX reverse proxy gateway

HTTP(s) in EOS today

- libmicrohttpd handler in EOS supports **GET, HEAD, PUP, DELETE** for data handling and WebDAV for meta-data handling **PROPFIND, PROPSET, MKCOL, DELETE, OPTIONS**
  - **S3** handler exists, but **does not support the full specification**
  - **OC handler** extends WebDAV with OwnCloud specific chunked upload mechanism
  - libmicrohttpd with EOS shows a strong **performance bottleneck** in terms of request/s (~ 100-150 Hz) and is CPU/memory wasteful due to separate IO buffers and thread pool

XrdHttp

- **Since 2013** XRootD implemented a **protocol bridge** allowing to add any protocol inside the XRootD server - even on the same port as XRootD protocol
- XrdHttp had been implemented by **Fabrizio Furano (IT-ST)** and is used inside **DPM**
  - the plug-in bridges **HTTP GET,PUT,HEAD** requests to the internal XrootdXeq XRootD protocol handler
  - the plug-in supports **HTTP(S)** on the same socket with **client authentication**
  - the plug-in supports **HTTPS to HTTP redirection** using a token mechansim
  - in **2017** an **plug-in handler** has been added to HTTP to be able to **extend/overwrite** the default **functionality** of XrdHttp

XrdHttp

- the **native XrdHttp** implementation is **not compatible** out of the box with EOS - only if used in a **gateway** machine **with a proxy** plug-in to talk to EOS

- it is a bigger development project to move all functionality from the EOS HTTP integration into and XrdHttp plug-in

- the simplest model feasible is just to **bridge** the **XrdHttp request handler** to the **EOS HTTP request handler** - this keeps all the original EOS protocol implementations but allows to benefit from the XRootD based thread pool etc.

  - two patches were needed in the API of the external handler plug-in, which are available in **XRootD 4.9**

XrdHttp in EOS

- the **MGM plug-in** has **141 lines of code** to act as a request bridge
- the **FST plug-in** has **238 lines of code** to act as a request bridge
- the **code** is currently **not merged** into a release branch and can be found here including testing RPMs from the CI: https://gitlab.cern.ch/dss/eos/tree/xrdhttp
- the **plug-in co-exists** in this version **with** the **libmicrohttpd** server and both can be used - however if FSTs are configure with XrdHttp, the MGM redirects to to the XrdHttp port
- requires **XRootD 4.9**

XrdHttp in EOS ⇒ MGM configuration

XrdHttp

HTTP(S) using the XRootD thread-pool and XrdHttp can be enabled in `/etc/xrd.cf.mgm` like:

```
if exec xrootd
   xrd.protocol XrdHttp:9000 /usr/lib64/libXrdHttp-4.so
   http.exthandler EosMgmHttp /usr/lib64/libEosMgmHttp.so eos::mgm::http::redirect-to-https=1
   http.cert /etc/grid-security/daemon/host.cert
   http.key /etc/grid-security/daemon/privkey.pem
   http.cafile /etc/grid-security/daemon/ca.cert
fi
```

To disabel HTTPS you can remove the cert/key/cafile directives. If you want to redirect the data transfer from HTTPS to HTTP you can change the configuration key to `eos::mgm::http::redirect-to-https=0` or just put `none` at this place.

The targetport in redirection is currently taken from the sysconfig file and uses `EOS_FST_HTTP_PORT+1000`.

XrdHttp in EOS

⇒ FST configuration

XrdHttp

HTTP(S) using the XRootD thread-pool and XrdHttp can be enabled in `/etc/xrd.cf.fst` like:

```
if exec xrootd
   xrd.protocol XrdHttp:9000 /usr/lib64/libXrdHttp-4.so
   http.exthandler EosFstHttp /usr/lib64/libEosFstHttp.so none
   http.cert /etc/grid-security/daemon/host.cert
   http.key /etc/grid-security/daemon/privkey.pem
   http.cafile /etc/grid-security/daemon/ca.cert
fi
```

To disable HTTPS you can remove the cert/key/cafile directives. The targetport in redirection is currently taken from the sysconfig file and uses `EOS_FST_HTTP_PORT+1000`. The protocol used for data transfers is configured on the MGM. By defaul HTTPS access redirects to HTTP on the data server if not modified in the MGM configuration file.

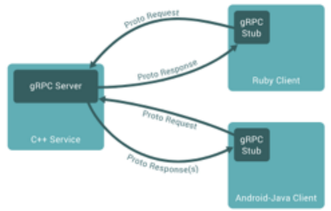
XrdHttp Release/ Integration Plan

- **XrdHttp** needs sufficient testing with the CERNBOX test suites (smashbox etc.) before it can be used in production
  - code **can be merged** without risk **to EOS 4.X** - will happen latest Q2
  - **libmicrohttdp** will be **deprecated in EOS 5.X**
  - approach is compatible with adding additional external handler plug-ins e.g. macaroons, tokens
    - at this moment unclear, if third-party copy HTTP plug-in can be easily used

GRPC

https://grpc.io/

- **GRPC server** implementation is provided in C++ by GOOGLE as an embedded server running on a dedicate port (default port 50051)
  - supports **HTTP** and **HTTPS** transport



- as most of GOOGLE products **no** proper **RPM packaging** available - GRPC libraries are built in our CI `grpc & grpc-devel` and available in EOS dependencies repository
- GRPC port can be specified in sysconfig file via **EOS_MGM_GRPC_PORT**
- GRPC server uses a **fanout file** in the logging class writing to `/var/log/eos/mgm/Grpc.log`

GRPC

https://grpc.io/

- HTTPS is configured when providing the following sysconfig variables
  `EOS_MGM_GRPC_SSL_CERT`
  `EOS_MGM_GRPC_SSL_KEY`
  `EOS_MGM_GRPC_SSL_CA`

  - once these are configured clients can connect with a certificate which will be verified against the given CA chain and the **client DN** is **extracted** server side

EOS Identity Mapping for GRPC

https://grpc.io/

- virtual identity interface (**vid**) supports *tident* mapping based on **client DN** or an application specific **authorisation key**
  - the mapping has to be defined for each client host accessing the GRPC server or a host name wildcard if it should apply to all GRPC calls
  - GRPC calls appear with the protocol name `grpc`
  - all **client** requests can be **mapped to a static role/identity** in EOS **based** on the client **DN** or the application **authorisation key**
  - it is possible to use the **authorisation keys in ACL** rules e.g. allow an application with a given authorisation key to access certain directories and/or act as the owner of that directory
    `sys.acl=k:990A425C-5903-4ECC-AB64-A3C2CB48D21D:rwx`

Identity Mapping for GRPC

- Map a given authorisation key to a certain role (e.g. adm):
  `eos vid set map -grpc key:990A425C-5903-4ECC-AB64-A3C2CB48D21D vuid:3 vgid:4`

- Map a given DN to a certain role (e.g. adm) :
  `eos vid set map -grpc “key:DN=….CN=xyz” vuid:3 vgid:4`

GRPC Service Protocol

```
package eos.rpc;

option java_multiple_files = true;
option java_package = "io.grpc.eos.rpc";
option java_outer_classname = "EosProto";
option objc_class_prefix = "EOS";

service Eos {
        // Replies to a ping
        rpc Ping (PingRequest) returns (PingReply) {}

        // Replies to MD requests with a stream
        rpc MD (MDRequest) returns (stream MDResponse) {}
}
```

GRPC Service Protocol

GRPC service	Description	IO mode	Remark
ping	service roundtrip	synchronous	used to measure default latency
md	retrieve meta-data	streaming	request single file, directory or listing

GRPC Test Clients

- **eos-grpc-ping** [—key <ssl-key-file> —cert <ssl-cert-file> --ca <ca-cert-file>] [--endpoint <host:port>] [--token <auth-token>]
  - ping given endpoint or localhost:50051
- **eos-grpc-md** [--key <ssl-key-file> --cert <ssl-cert-file> —ca <ca-cert-file>] [—endpoint <host:port>] [--token <auth-token>] [-l] <path>
  - retrieve MD for path, if -l specified return children if path is a directory

GRPC Future

- EOS CLI used to provide 100% REST API via URL syntax
  - argument escaping hell
  - fully synchronous api
  - started migration of EOS CLI commands to protobuf based messages (base64 encoded serialisation stream) e.g. **acl, route, ns, fs, find** commands …
  - migration of EOS CLI commands to protobuf based will continue
  - todo: automatically **bridge** all client **protobuf** requests **to GRPC server**
    - will provide the complete API to interact with EOS via GRPC from any computing language for MGM operations
    - **file IO** operations are available via **HTTP/WebDAV** interfaces

GRPC in EOS

- GRPC is not merged into the release branch - mainly because of new RPM dependency
- You can get the source tree here, also testing RPMS https://gitlab.cern.ch/dss/eos/tree/dev_grpc

Summary

- The integration of **HTTP(S)** via XrdHttp will provides a **complete and secure access protocol** with high scalable performance to EOS without a risk in touching generic existing and working implementation
  - allows deprecation of microhttpd dependencies in next major version and might become THE default protocol in the future
- Adding **GRPC** protocol to EOS will offer a generic interface to a large variety of clients with **full security** (authN,authZ) including **asynchronous meta-data operation** for large responses
  - GRPC can replace and improve the current URL based REST API

THANK YOU

QUESTIONS ?

EOS workshop

4-5 February 2019

CERN

Europe/Zurich timezone

There is a live webcast for this event.