Ins & Outs of the New Client

Łukasz Janyst

CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

XRootD

XRootD Workshop
San Diego, 28.01.2015

CERN

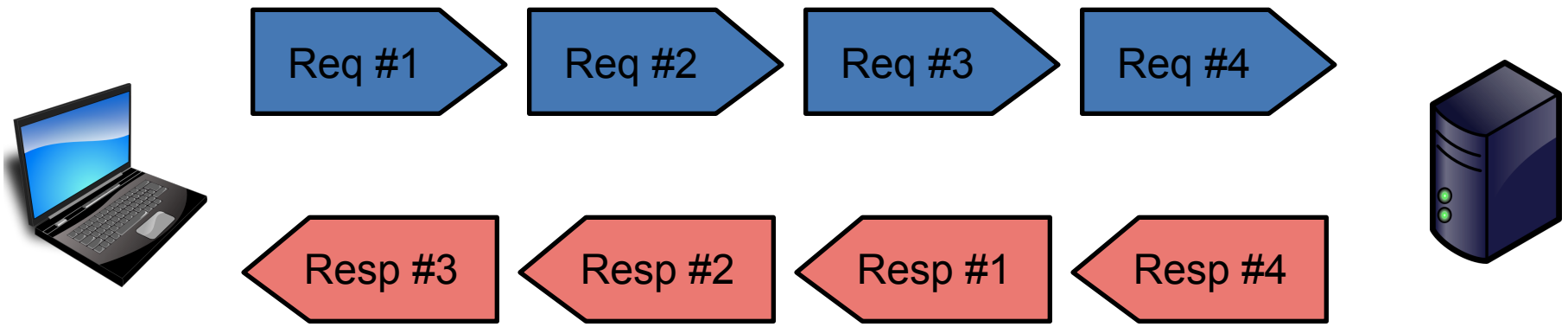
Status

- The new client (**XrdCl**) actually is **the client** since XRootD 4.0
- The old client (**XrdClient**) is **deprecated** and on its way to be removed
  - Only critical bugs will be fixed

The sales pitch - the whys

- **Asynchronous**
  - for all possible operations
  - use callbacks instead of half-blocking cache ops
- **Thread safe**
  - allow multiple threads to access the same file objects
- **Fork-safe**
  - preserve file object validity and system consistency
- **Lighter and more performant**
  - go away from one thread per socket model
  - saturate 10Gbps links
- **Maintainable and extendable**
  - careful about abstractions, support plug-ins

Importance of asynchrony



- The XRootD protocol supports virtual streams
- There may be many requests outstanding and the server may respond in the order it chooses
- The new client handles responses as soon as they come calling the user call-back function

Importance of asynchrony


```
]==> time xrd metaman dirlist /data/bigdir > /dev/null
1.58s user 1.94s system 4% cpu 1:18.09 total
```

```
]==> time xrdfs metaman ls -l /data/bigdir > /dev/null
1.26s user 0.46s system 64% cpu 2.678 total
```

- List a directory of 40k files spread across 4 servers
- Link: 100 Mbps, round-trip 1.8 ms

The async API

- All of the xroot protocol requests implemented as asynchronous methods
- The calls queue the request and return, **never block**

```
XRootDStatus File::Open( const std::string &url,
                         OpenFlags::Flags   flags,
                         Access::Mode       mode,
                         ResponseHandler   *handler,
                         uint16_t           timeout )
```

- No need to have cache to handle buffers
- Synchronous calls implemented using async ones

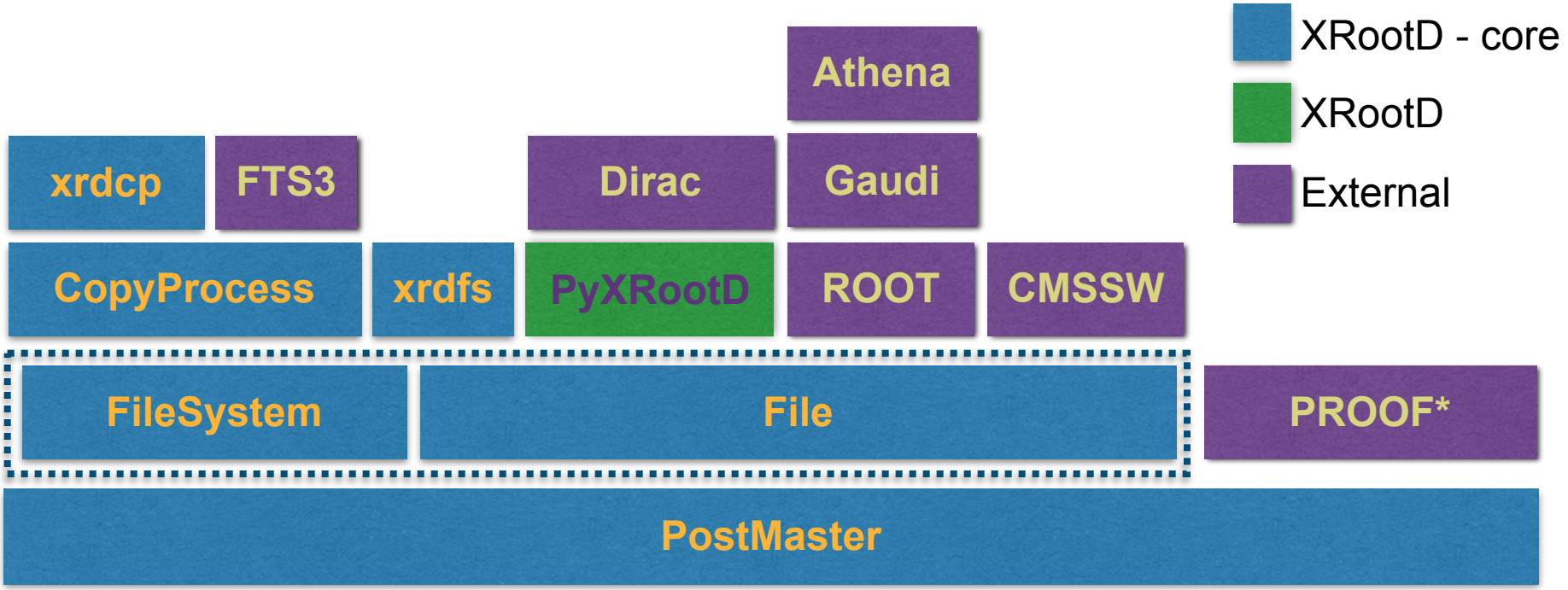
The user API

- **XrdCl::FileSystem** for meta-data requests
  - mkdir, rmdir, query, locate, move truncate, chmod, ping, stat...
- **XrdCl::File** for data operations
  - read, write, readv...
- **XrdCl::CopyProcess** for running copy operations
  - configure checksumming, third-party mode…
  - follow progress, get detailed results

The command line

- **xrdcp** (the old one is now **xrdcp-old**) - the copy command, backwards compatible with the old one
- **xrdfs** (replacement for **xrd**) - filesystem queries, not backwards compatible, cleanups to the interface

The API stack



Recovery & the knobs

- The client comes with many recovery strategies that may be tuned to your liking
  - mainly via various timeout settings
- There is also **plenty of knobs** concerning:
  - IP stacks
  - connections options
  - plug-ins
  - monitoring

Configuration

- **Configuration** file
  - read configuration from `/etc/xrootd/client.conf` or `~/.xrootd/client.conf`
  - the local config takes precedence over the global one
  - key-value pairs
- **Environment** variables
  - the same keys as in the config file but capitalised and prefixed with `XRD_`
- Command line **parameters** for xrdcp
  - `-DS` and `-DI` parameters
  - the same keys as in the config file

IPv6 vs IPv4

- The knob controlling the transport layer protocol is called **NetworkStack**, it may be set to:
  - **IPAuto** - detect which stack to use
  - **IPAll** - use IPv6 stack and both IPv6 and IPv4 addresses
  - **IPv6** - use IPv6 stack and addresses
  - **IPv4** - use IPv4 stack and addresses
  - **IPv4Mapped6** - use IPv6 stack but IPv4 addresses

Simple monitoring

- Client can insert some user-set local job properties into the server's monitoring stream
- Useful when trying to do analytics of access patterns based on server logs
- Envvars:
  - **XRD_APPNAME** - application name, defaults to executable name
  - **XRD_MONINFO** - custom monitoring info

Monitoring plug-ins

- Be informed about events happening in the XRootD client code
  - stream connections and disconnections, file opens and closes, errors, file transfers, checksum calculations
- Write a C++ class extending **XrdCl::Monitor**
- Put it in a shared lib
- Let the client know about it via **ClientMonitor** setting (**XRD_CLIENTMONITOR** envvar)

Client plug-ins

- The plug-in API is exactly the same as the **File** and **FileSystem** API - except for the **virtual** keyword
- Only asynchronous calls may be overloaded

```
virtual
XRootDStatus File::Open( const std::string &url,
                         OpenFlags::Flags   flags,
                         Access::Mode       mode,
                         ResponseHandler   *handler,
                         uint16_t           timeout )
```

Client plug-ins


```
]==> cat eos.conf
# example configuration

url = eosatlas.cern.ch;eoscms.cern.ch
lib = /usr/lib64/libXrdEosClient.so
enable = true
customarg1=customvalue2
customarg2=customcalue2
```

- The plug-ins are discovered and configured by scanning configuration files
- There is one config file per plug-in
- It's a set of key value pairs

Client plug-ins

- The plug-in manager will search for global configuration files in:

  /etc/xrootd/client.plugins.d/

- The global settings may be overridden by configuration files found in:

  ~/.xrootd/client.plugins.d/

- Any of the previous settings may be overridden by configuration files found in a directory pointed to by:

  XRD_PLUGINCONFDIR

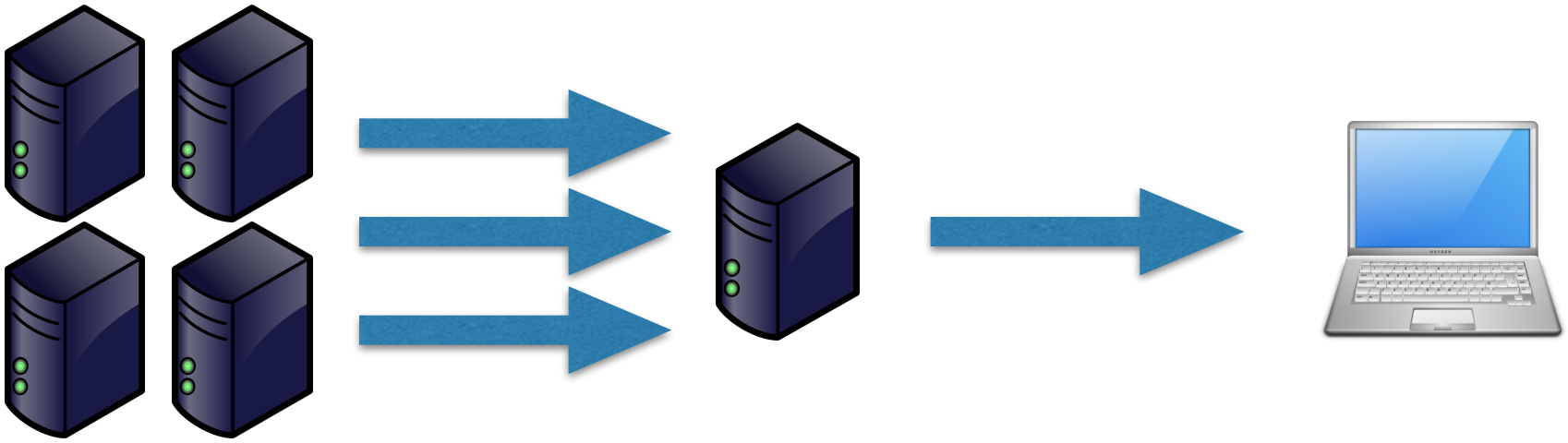
Client plug-ins - Motivation

RAIN

Redundant **A**rray of **I**ndependent **N**odes

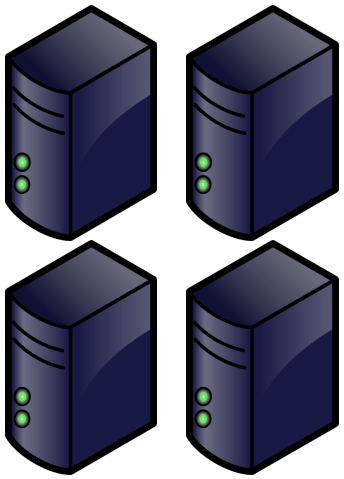
- Stripe files and use erasure coding to increase fault tolerance
- Primarily for archiving and similar use-cases
- Multiple techniques:
  - Hamming parity
  - Reed-Solomon error correction
  - Low-density parity-check

Client plug-ins - Motivation

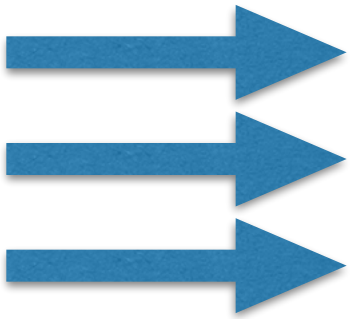


- The client needs to see the file as a whole
- File reconstruction needs to be done at a gateway
- CPU and bandwidth scalability issues

Client plug-ins - Motivation



Stripe servers



User

- When contacting EOS the client is able to execute specialised code
- Can contact the stripe servers directly
- Can reconstruct the data at the client machine
- **Transparently to the users - whatever they are!**

Client plug-ins - Motivation

- The client plug-ins are generic and all possible calls may be overridden
- You could use them to do things like:
  - implement a multi-source client, Brian's style
  - support local caching
- Client plug-ins provide a way for the XRootD community to **play, tinker and hack** the client
  - exactly what made the XRootD server so successful!
- Everything is **transparent** to the layers above!

Staff changes

- My staff contract expires with the **end of April**
  - Therefore I leave CERN and XRootD
- I would like to **thank you all** for the last five years of fruitful cooperation
- CERN is in the process of finding a replacement
- **Elvin Sindrilaru** will take over some of my duties

Thanks!

Thanks for your attention!

Questions? Comments? Concerns?