Features of the new XRootD client and ROOT IO plugin

Łukasz Janyst

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

ROOT Users Workshop
Saas Fee, 12.03.2013

Outline

- Motivation
- New client API
- Command line tools
- New ROOT plug-in
- Status

Motivation

- Fully asynchronous, call-back-based IO
- Thread safety
- Fork safety
- More efficient thread and syscall utilization
- Decouple the caching
- Overall maintainability

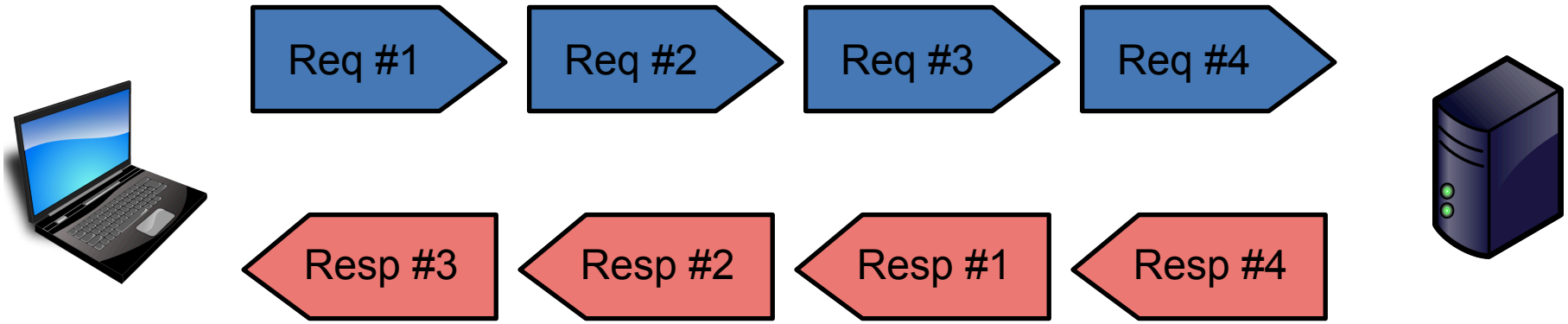
Asynchronous

- 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 )
```

- The response handler is called when the response is ready
- No need to have cache to handle buffers
- Synchronous calls implemented using async ones

Async vs. stream multiplexing



- 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

Performance - metadata ops

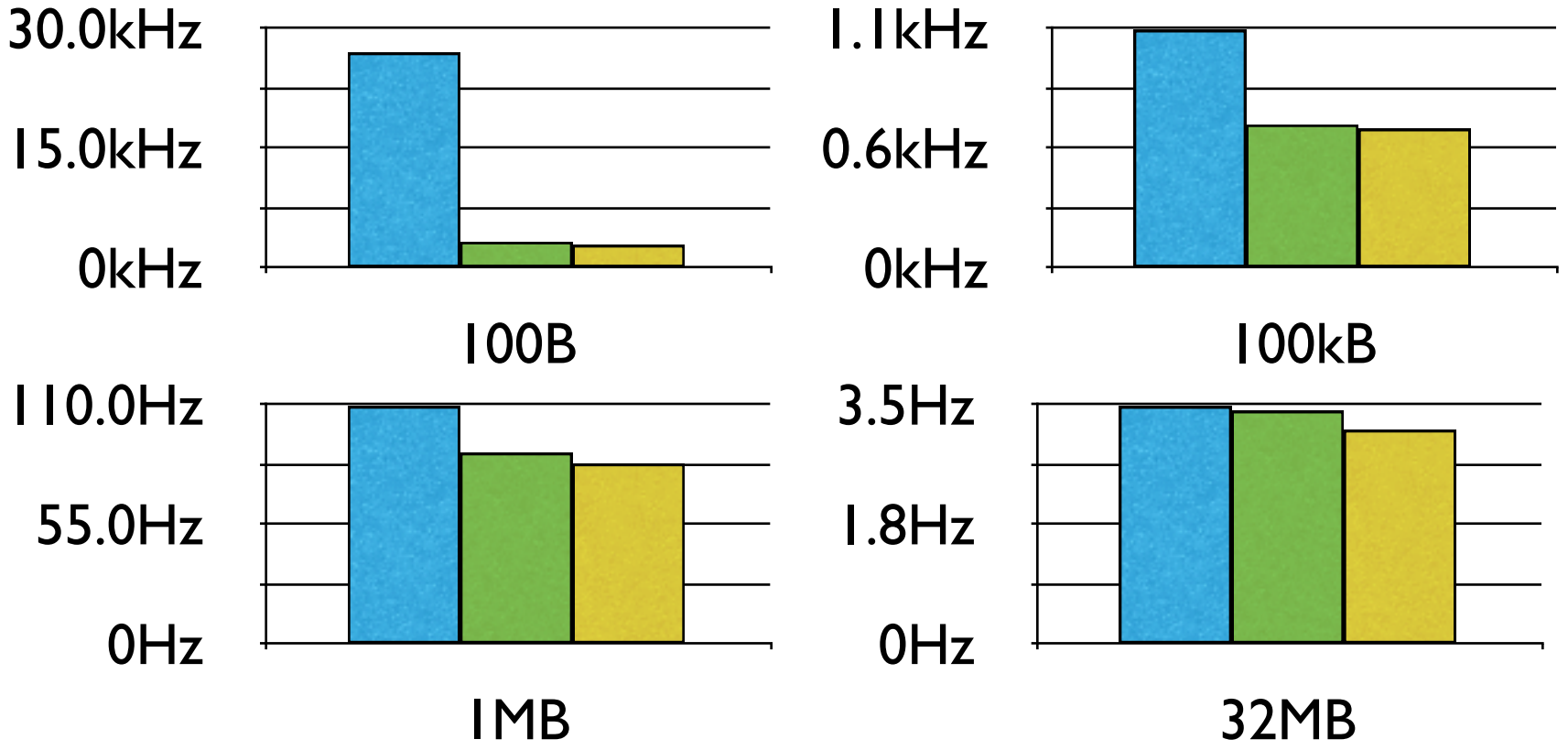

```
]==> time xrd metaman dirlist /data/bigdir > /dev/null
1.58s user 1.94s system 4% cpu 1:18.09 total

]==> 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

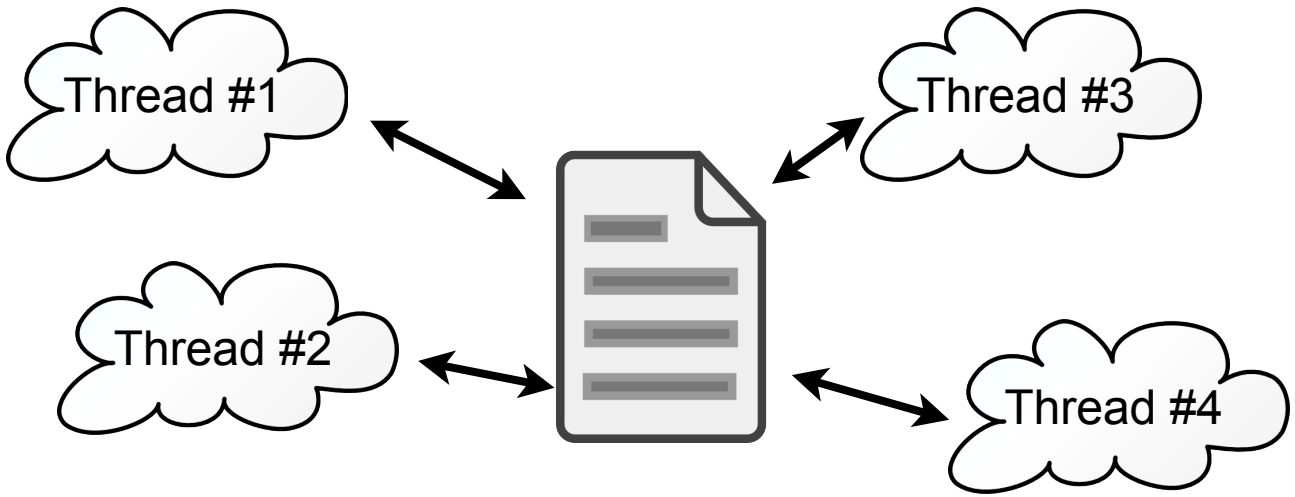
Data ops without cache

Link: 1000 Mbps, round-trip 0.3 ms



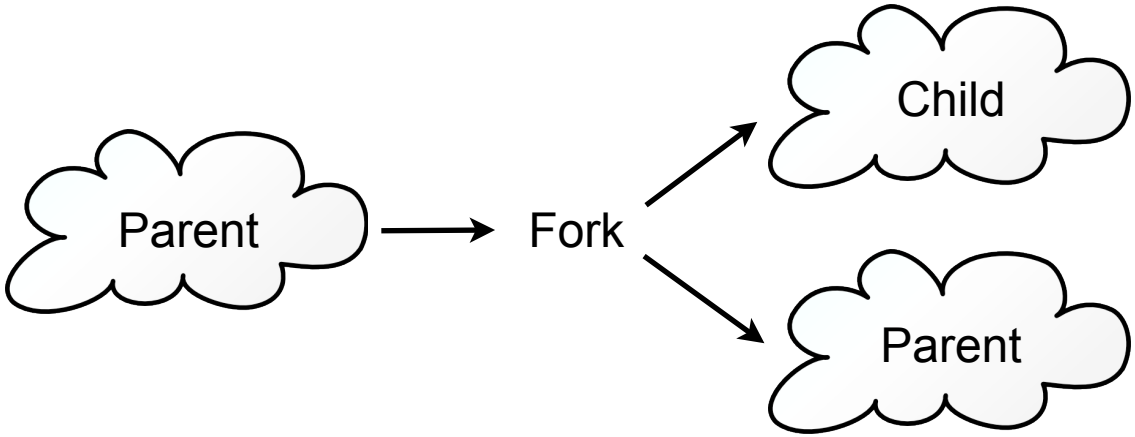
- Async means many requests sent at the same time
- Problem with sync requests has been identified and will be fixed in 3.3.2

Thread safe/aware



- File and FileSystem objects can be safely accessed from multiple execution threads
- Internally uses a worker thread pool to handle call-backs

Fork safe



- Can handle forking even when the IO operations are in progress
- File and FileSystem objects remain valid in both parent and child
- The operations in the parent continue after the fork
- The objects in the child will run recovery procedure (like in the case of a broken connection)

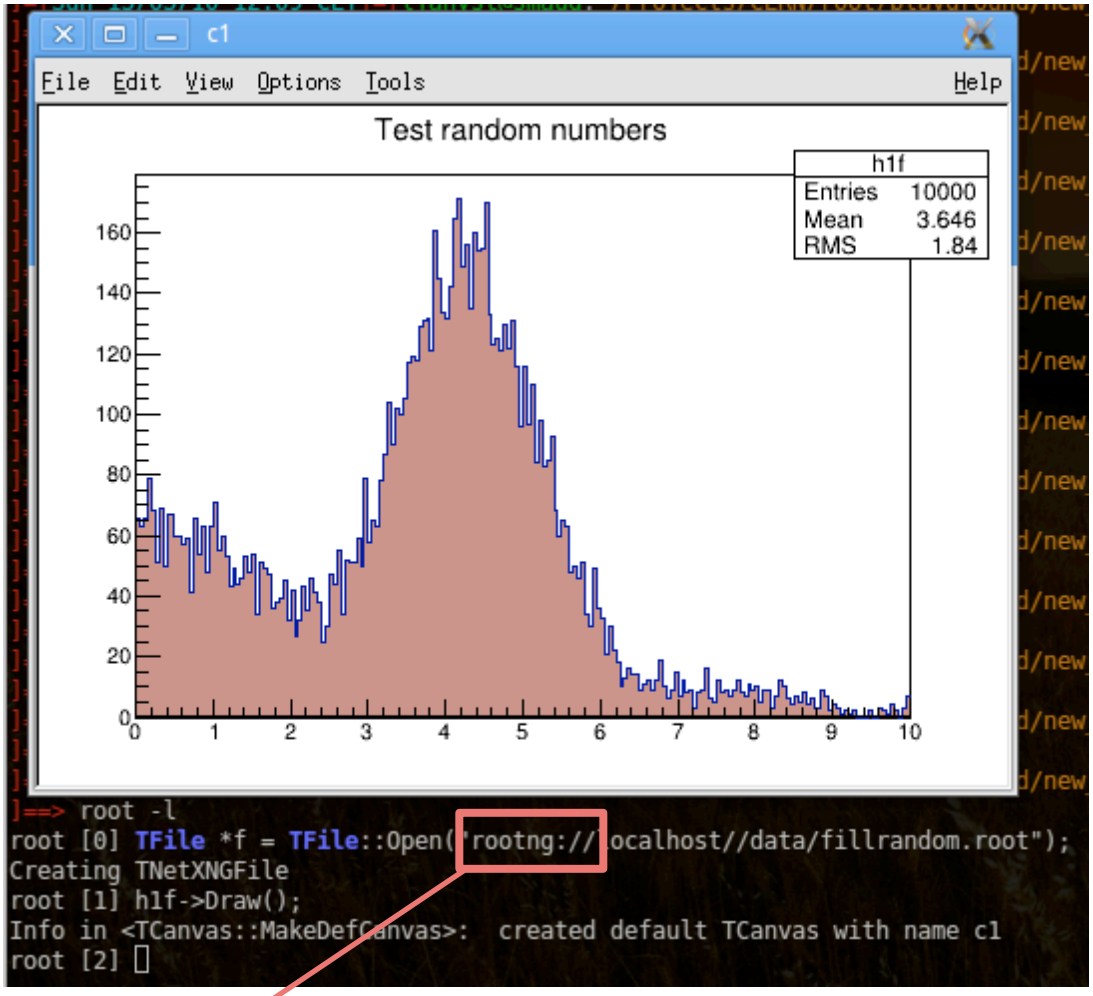
User classes

- **XrdCl::FileSystem** for meta-data requests
  - mkdir, rmdir, query, locate, move truncate, chmod, ping, stat...
- **XrdCl::File** for data operations
  - read, write, readv...
- Redesigned API, not backwards compatible but rarely used directly (interfaced by ROOT)

Command line

- **xrdcopy** (replacement for **xrdcp**) - backwards compatible interface, drop-in replacement, heavily used
- **xrdfs** (replacement for **xrd**) - cleanups to the interface, rarely used

ROOT plug-in status



Added a new protocol to simplify testing

- Prototype of TFile plug-in works
- It's fairly simple, mostly 1-to-1 mapping of methods
- Need to performance test and optimize
- Expect to submit the code in the next weeks

ROOT plugin

- Both new and old plug-in can co-exist at runtime
- For testing purposes selection can be done:
  - by changing the file URL (ie. **root://** to **rootng://**) or
  - setting the **XRD_CLIENT** environment variable or
  - setting a variable in a **.rootrc** file or
  - using **gEnv**
- In the final version the new plugin should replace the old one as the default

Migration

- The new client libraries and executables can co-exist with the old ones
- The two ROOT plugins can co-exist
- Old client will be provided for two years from the 4.0.0 release
  - critical bug fixes
  - no new features

Status

- Implements all the planned functionality
  - input on new features is welcome
- Focusing on performance tuning and bug hunting
- Successfully tested:
  - server-side EOS (will be part of the next production release)
  - FTS plug-in doing third party copies
  - CMS IO prototype
- The ROOT TFile plug-in expected in April

Summary

- Released with xrootd 3.3.0
  - However performance tuning work is still on-going, and we expect improvements in 3.3.2
- The API is unlikely to change incompatibly, but we don't guarantee binary compatibility until 4.0.0
- ROOT plugin (TNetXNGFile) will come in the next weeks

Questions

Thanks for your attention!

Questions? Comments?