The new XRootD client

Łukasz Janyst

Architects Forum Meeting
Geneva, 12.04.2012

Outline

- Motivation
- Key points of the implementation
- Migration & backwards compatibility
- Status & next steps

Motivation

- Threading issues
  - file objects cannot be safely shared between execution threads
  - heavy - one thread per physical connection, lock contention
- Caching issues
  - use cache to handle read & writes (semi) asynchronously
  - cannot be easily disabled when needed
- Overall maintainability
  - hard to extend and fix bugs
  - some features not useful anymore

New implementation-features

- Fully asynchronous
  - all requests may be handled asynchronously, not only reads and writes
    - listing of huge directories an order of magnitude faster
  - callback model instead of request-and-wait-for-the-cache model
  - no need to have a cache to handle async communication
  - synchronous requests implemented in terms of asynchronous (with a semaphore)
  - avoid ambiguity and conflict between TTree cache and internal cache

New implementation-features

- Thread safe
  - the user API classes hold very little or no mutable state at all
- Lighter
  - one extra thread to handle socket events
  - one extra thread to handle time events
  - no need to spawn extra thread for every new connection
  - uses host system optimized polling (through libevent) instead of block+timeout model, which should reduce number of syscalls

Migration

- Discussed within the XRootD collaboration
- The new client libraries and executables can coexist with the old ones
- We will keep the old client for two years from the release of the new one
  - critical bug fixes
  - no new features
- New ROOT plugin needed

User classes

- **XrdClient::Query** for stateless requests
  - mkdir, rmdir, query, locate, move truncate, chmod, ping, stat...
- **XrdClient::File** for (stateful) file operations
  - read, write, readv...
- Redesigned API, not backwards compatible but almost never used directly (interfaced by ROOT)

Command line

- **xrdcopy** (replacement for **xrdcp**) - backwards compatible, heavily used, work on-going
- **xrdquery** (replacement for **xrd**) - backward compatible to some extent, cleanups to the interface, rarely used

ROOT plugin

- New ROOT plugin is needed
- No need to change any user code depending on ROOT IO
- The new and the old plugin can co-exist and be switched dynamically at runtime thanks to ROOT's plugin manager:
  - by changing the file URL (ie. root:// to newxroot://) or
  - setting an environment variable or
  - setting a variable in a .rootrc file

Status & next steps

- User API and most of the application code is DONE
- Still some work needed on:
  - xrdcopy
  - failure recovery and timeouts
- ROOT plugin (similar to TXNetFile)

- Test & iron-out the rough edges
  - use internally in EOS
  - planned to use in CASTOR when migrating to pure XRootD internal data transfers

Proposed timeline

- **June-July** - Test the new client inside EOS components
- **August** - ready for experiment functional tests
- **October (?)** - Start of deployment tests
- Aim for production ready-ness and integration in production releases of the experiment frameworks early in the long shutdown?

Questions

Thanks for your attention!

Questions? Comments?