TREQS Overview

by Bernard Chambon

NCSA $\leftrightarrow$ CC-IN2P3 meeting

Thursday, 14th January 2016

Outline

- TREQS's aim
- Technical details
  - Functionalities
  - Architecture
  - Development process
- Current status, schedule

TREQS aim 1/2

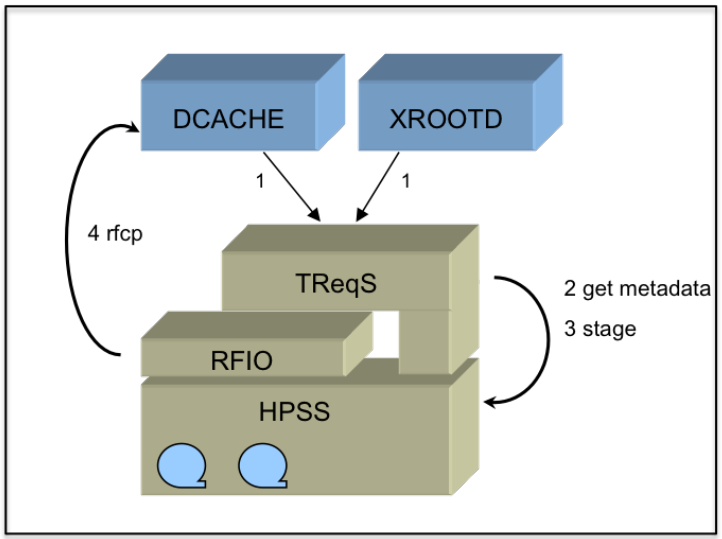
- Memo about HPSS at CC-IN2P3
  HPSS used for 15+ years, by 80 groups ; Currently 57 M files, amount of ∼32 PB
  Wired to Oracle SL8500 (40,000 tape cells), with 3 media types T10K-B..D (D = 8.5 TB)
  ∼100 drives dedicated to HPSS

  Access via RFIO using CLI [1] tools (rfdir, rfcp). (⇒ CERN RFIO built using HPSS API)
  Major access from DCACHE, XROOTD
  HPSS average access per day :
  2000 tape mounts ; 50 K files accessed (16 K for reading), 35 TB (18 TB for reading)
  16 K for reading : 10 K served by TREQS-1 [2], 8 K are staging (2 K files already on disk)
- TREQS motivation : regulation and optimisation of staging
  Regulation
  HPSS requests are processed in FIFO mode
  No drive number limit per COS [3], for reading (and HPSS needs drives for writing)
  Optimization
  Several requests for files on same tape could lead to several mounts of that tape
- TREQS main objectives (details in slide #5)

Optimisation	Decrease number of tape movements (mount, rewind)
Optimisation	Increase throughput for staging (ordering files)
Regulation	Provide control upstream to HPSS

1. CLI : Command Line Interface ; WUI : Web User Interface
2. TREQS-1 : Old version of TREQS running on production (see next slide)
3. COS : HPSS Class Of Service

- TREQS positioning



- TREQS story
  A TREQS-1, inspired by a work at BNL (ERADAT, formally BNL Batch) from BNL [1]
  Initial version, running for 5 years, but without any evolution
  TREQS-1 usage per day :
  10,000 avg | 25,000 max staging requests, mainly from ATLAS experiment

  A new TREQS-2 architecture and implementation, started at fall 2015

1. BNL : Brookhaven National Laboratory

- Decrease number of tape movements, Increase throughput for staging
  - Aggregate requests over time, per tape, to reduce number of tape mounts
  - Sort files to be staged in most efficient way, currently FPOT [1], (RAO [2] in the future ?)
  - Possibly add files to be staged while a tape is currently being read
- Provide control upstream to HPSS
  - Limit number of allocated drives per drive-model (e.g. 25 drives for T10K-D tape model).
  - Fair share of drives among users (planned, but not yet available)
  - Limit access to granted users (white list)
- About client
  - CLI to submit, but also to query and cancel requests, possibly per tape
  - Provide synchronous mode (rfcp for file transfer) and asynchronous mode (= pre-staging)
  - Provide bulk staging mode (= staging from a filelist)

1. FPOT: Logical File Position On Tape
2. RAO: Recommended Access Order

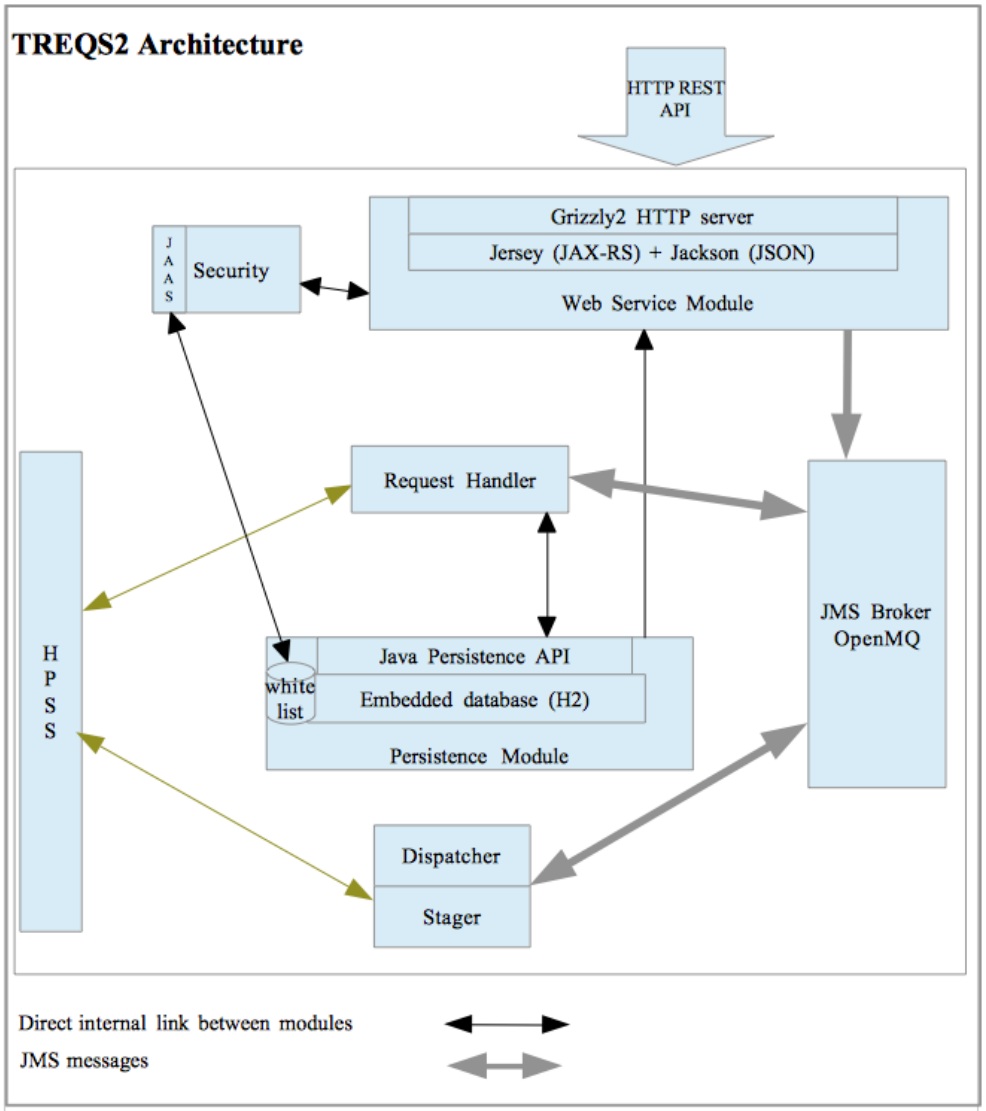
- Main components (guideline = use standards !)
  Client / Server : Java multithreaded server, python for client

  REST API (http, json) : Lightweight client (CLI or WUI), easy to develop
  JMS [1] : Components with well delimited scope, less shared data structures
  H2 DB as persistence : Fast, embedded (or server), 100% java, freely available
  JAAS [2] : To plug your own module (= site customization)
  HPSS API via JNI [3]
  Mustache+Datatable for (minimal) real time monitoring

- Schema of architecture ...

1. JMS : Java Messaging Service
2. JAAS : Java Authentication & Authorization Service
3. JNI : Java Native Interface

- Schema


TREQS2 Architecture
HTTP REST API
Grizzly2 HTTP server
Jersey (JAX-RS) + Jackson (JSON)
Web Service Module
JAAS
Security
Request Handler
HPSS
Java Persistence API
white list
Embedded database (H2)
Persistence Module
JMS Broker OpenMQ
Dispatcher
Stager
Direct internal link between modules
JMS messages

- Tools
  Maven for project mgmt
  Git as code repository (gitlab.in2p3.fr, private access)
  Jenkins for continuous integration process
  Sonar for code audit
- Tests methods
  Unit & integration tests
  Load test (with elapse time metrics)
  Metrics on staging throughput (e.g. without|with TREQS, without|with FPOT sort, etc.)
- Our main concerns (guideline = quality)
  - Provide a scalable and reliable server
  - Provide a configurable, maintainable and portable software
    No hard code value, use config file
    Use standards (REST, JSON, JAAS,)
    Modularity (authentication, scheduling algo)
    Open source license
  - Provide operation tools
    Admin tool (possibly WUI)
    Monitoring (integrated tool, detailed logs for external tool)

Current status, schedule

- Current status about TREQS-2
  Server, version 0.2, main components defined and implemented at 75%
  Client commands available to submit, query and cancel requests
- Key dates as schedule
  Alpha version for 2016 Q1
  Beta version for 2016 Q2
  We aim a production version 1 for fall 2016 (2016 Q3)
- Next steps
  Advanced functionalities (Fair share, priority staging,) planned for the following version
- For their feedback on this presentation, let me thank ...

  Fabio Hernandez
  Lionel Schwarz (TREQS software developer)
  Pierre-Emmanuel Brinette (HPSS & TREQS administrator)
  Yvan Calas (dCache & Xrootd service manager)