FTS3 Steering Meeting

9th July 2015

http://fts3-service.web.cern.ch/

@fts3

Current status

- Production version
  - 3.2.33
- Incoming version
  - 3.3.3 (3.3.0 + 4 bug fixes)

3.3.x improvements

- Less load on submission and querying
- ‘ priority’ and ‘ max_time_in_queue’ new submission parameters
- Expose if a file failed with recoverable or non-recoverable error
- Best replica chosen at submission time
- Coredump generation
- Web UI for configuration

http://fts3-service.web.cern.ch/content/fts-330

3.3.3 bug fixes

- New fixes over 3.2.33
  - [FTS-261] Context leak in the REST client
    - Affects LHCb
    - Actual bug in pycurl, workaround in FTS3 bindings
  - [FTS-270] TOC-TOU bug on submission
    - Race condition, affects everyone depending on the submission mechanism and rate
    - Regression test added

3.3.3 bug fixes

- Regressions introduced with 3.3.0
  - [FTS-265] Polling of multiple jobs with inner file fields cause duplication
    - Affects ATLAS
    - Regression test added
  - [FTS-269] /jobs/files leaks db connections
    - Validated in the development cluster and WebFTS machine

- Unexpected side effects of sqlalchemy + iterators

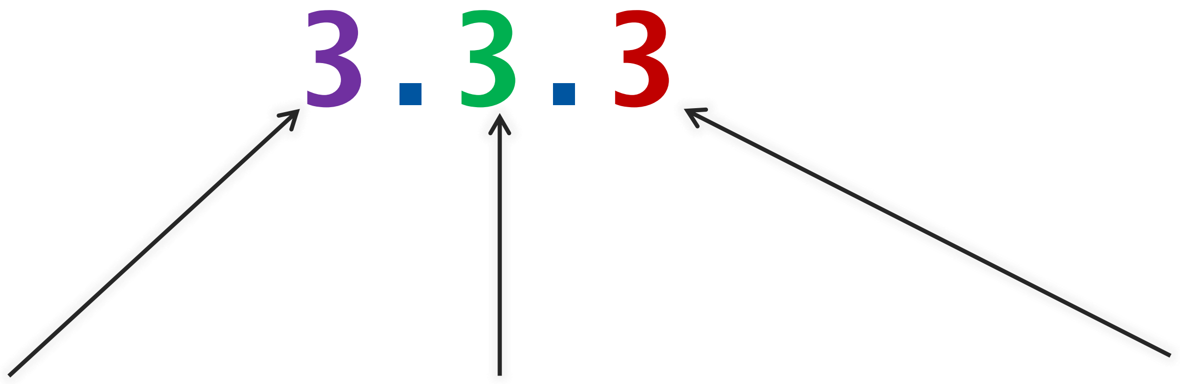
3.3.3 running in the CERN pilot

- As usual, when you consider it is good enough, ask sites to upgrade
  - More on this later…

Versioning (Before)

- Pretty much sequential revision
    - 3.2.30, 3.2.31, 3.2.32, 3.2.33

Versioning (Proposed)



Won’t change

New features, collection of several (or too complex) bug fixes

Hotfixes, no new functionalities

Versioning (Proposed)

- 3.3.3 -> 3.3.4 -> 3.3.5
  - Small changes, small impact, safe to upgrade
  - Little or no validation required
- 3.3.5 -> 3.4.0
  - Bigger (ish) changes, complex bug fixes, new functionalities
  - More validation required
- 3.4.0 -> 4.0.0
  - Not happening

Release pace

- Develop to master once, twice a month? Every week?
- If production is 3.3.3, and pre-production is 3.4.0, and a bug is found in production
  - Fix goes to 3.4.1 or 3.3.4?

Release pace (proposal)

- Merge to master (pre-prod) twice a month
  - If new features or big changes, minor increase
  - If only small bug fixes, revision increase
- Freeze merges while pre-prod does not become production
  - Cherry-pick bug fixes only and revision increase
- Once pre-prod hits production
  - Start merges from develop again
- gfal2 has been doing this already!

Distribution & upgrades (now)

- Two repositories
  - Development & Staging
  - Pilot instances upgrade automatically from Staging
  - Production instances manually upgraded when requested from the same repository as Pilot
- May need reconsideration?

Distribution & upgrades (thoughts)

- If version has meaning
  - Revision increase
    - Automatically picked by Production instances
    - This does *not* mean the fix doesn’t go to the pilot first
    - Faster reaction time
  - Minor increase
    - *Not* automatically picked by Production instances
    - An experiment needs to request the upgrade (?)
  - Major increase
    - Still not happening…

Distribution & upgrades (thoughts)

- Is the mailing list (fts3-steering) working fine for rollouts?
- Would automatic updates work even for major changes?
  - After prudential time in Pilot services!
  - See ‘future’ work for relevant points to increase the reliability of this behavior
- Would be possible to have updates applied progressively?
  - Pilot -> (CERN -> BNL -> RAL) (in no specific order)

Communication

- Bugs, help requests, etc… should go to
  - fts-support@cern.ch / dmc-support@cern.ch
  - GGUS
  - fts-devel@cern.ch within reason
  - Please, avoid personal mails
    - Since the person may be on holidays
    - Difficult rotas

Future

- Workforce available
  - Alejandro Alvarez Ayllon (75% + 20% = 95%)
    - To be reduced around 20% (75%)
  - Andrea Manzi (20% for *WebFTS*)
  - David Smith (40%)
    - Still need to ramp up
  - Getting a new LD soon…
    - Board end of July

Future (short term)

- Service stabilization
- Bug fixes
- Chase dead code
- Apply QA tools
- Refactor some parts of the code
- Continuous tests instead of nightly tests
- Evaluate other transfer optimization possibilities

Future (medium/long term)

- Same as short term
- S3 support
  - 3rd party copy from the Grid* to S3 is supported
  - From S3 to the Grid would be interesting
    - Otherwise FTS3 becomes a network bottleneck
    - Need support on the storages
- GridFTP pipelining
  - Need input (have been waiting for ~1 year)
- Reduce database load (!)

(*) True for DPM and dCache

Future (long term)

- SDN integration somehow?
  - A draft of a GFAL2 plugin that triggers before transfers
  - Could be used to send information to SDN controllers?
  - Information is limited
    - Final destination IP and port available
    - Final source IP and port not available

Future (long term)

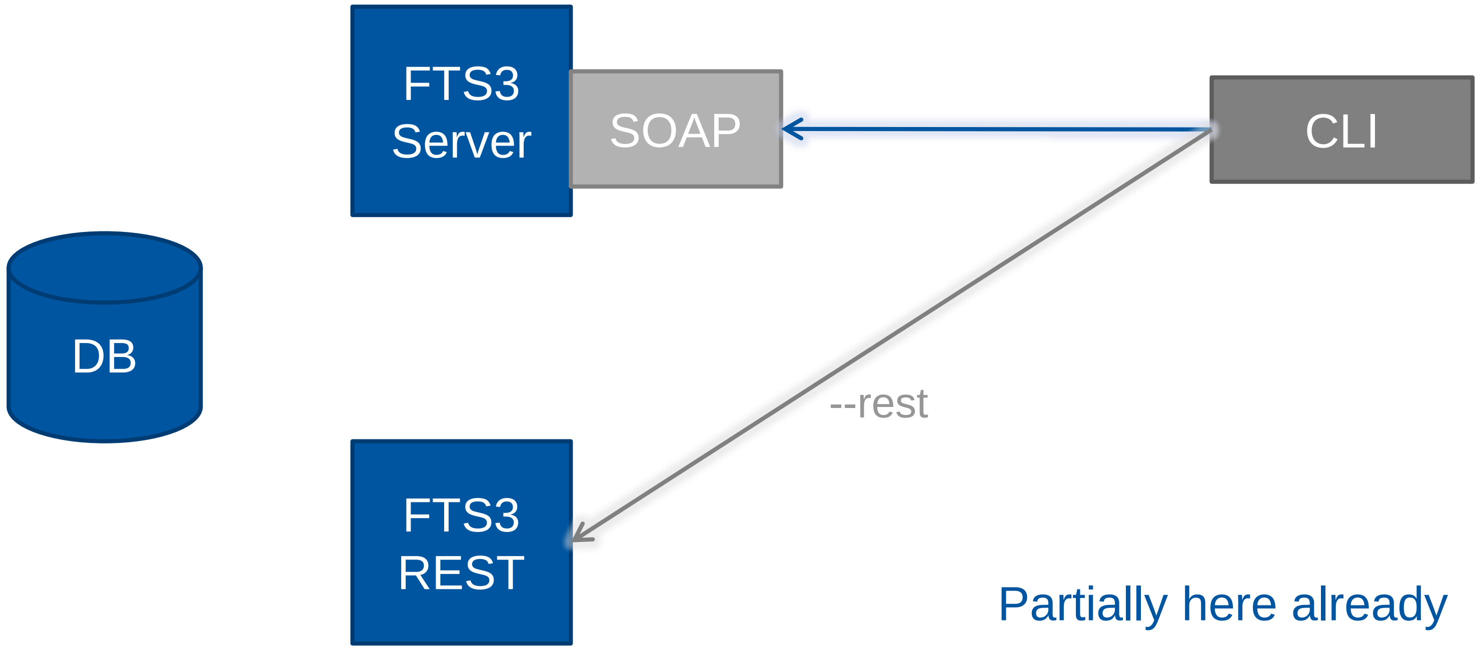
- Reduce codebase size
    - Less code, less maintenance effort
    - Easier to add functionality
    - Less bugs
- So cut loose all we can
- And what could we cut loose eventually?

Future (long term)

Future (long term)

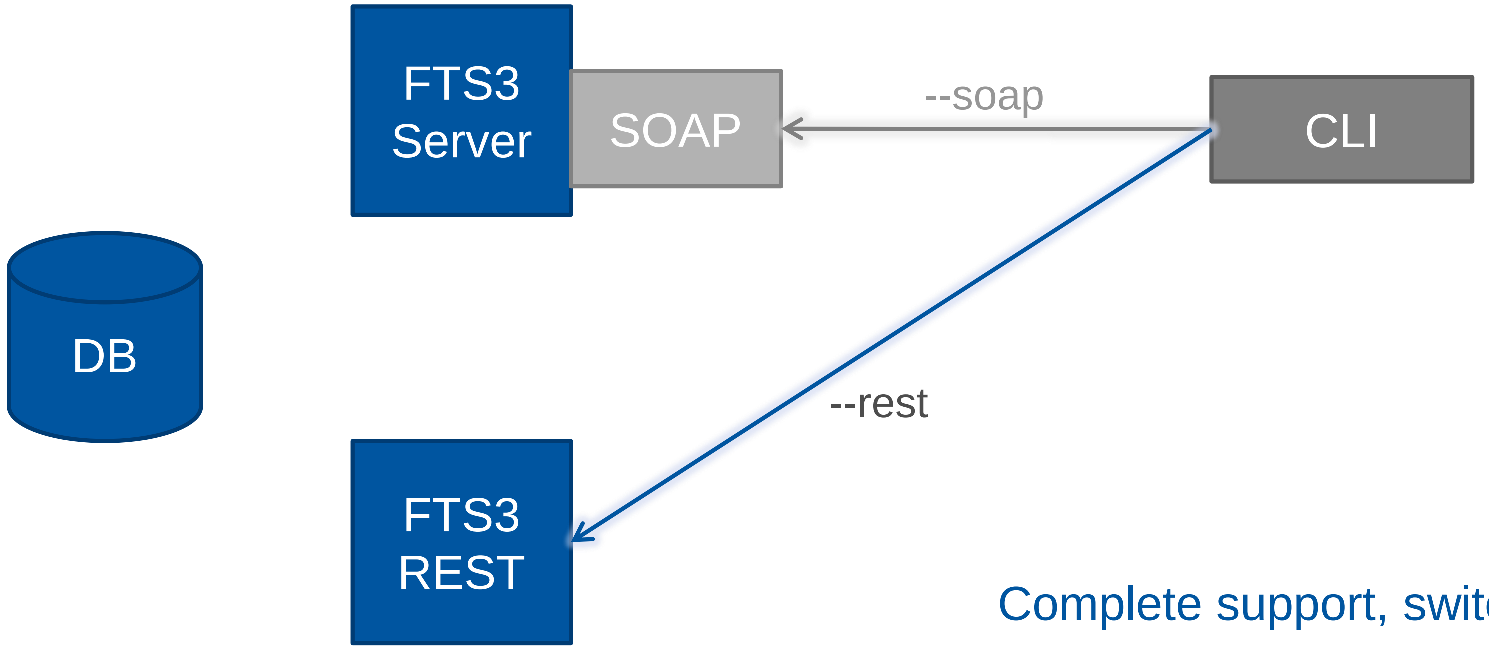
- SOAP API has 5.3k lines of C++ code
  - Out of 46.4k (11%!)
- Very hard to extend
- A bad bug in a client-facing code can cause a crash on the process that runs the transfers
- We have another API for submission anyway…
  - And experiments moved/will move/are moving towards it

Future (long term)



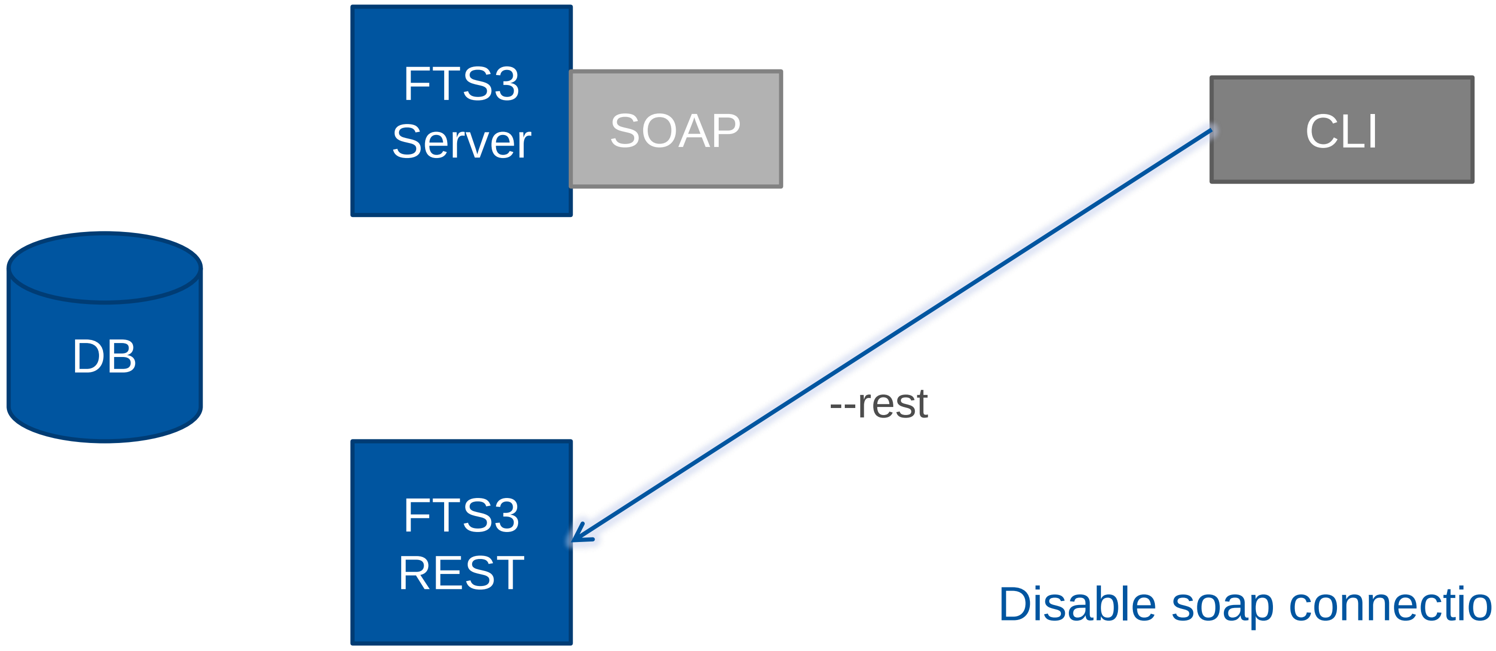
Partially here already

Future (long term)



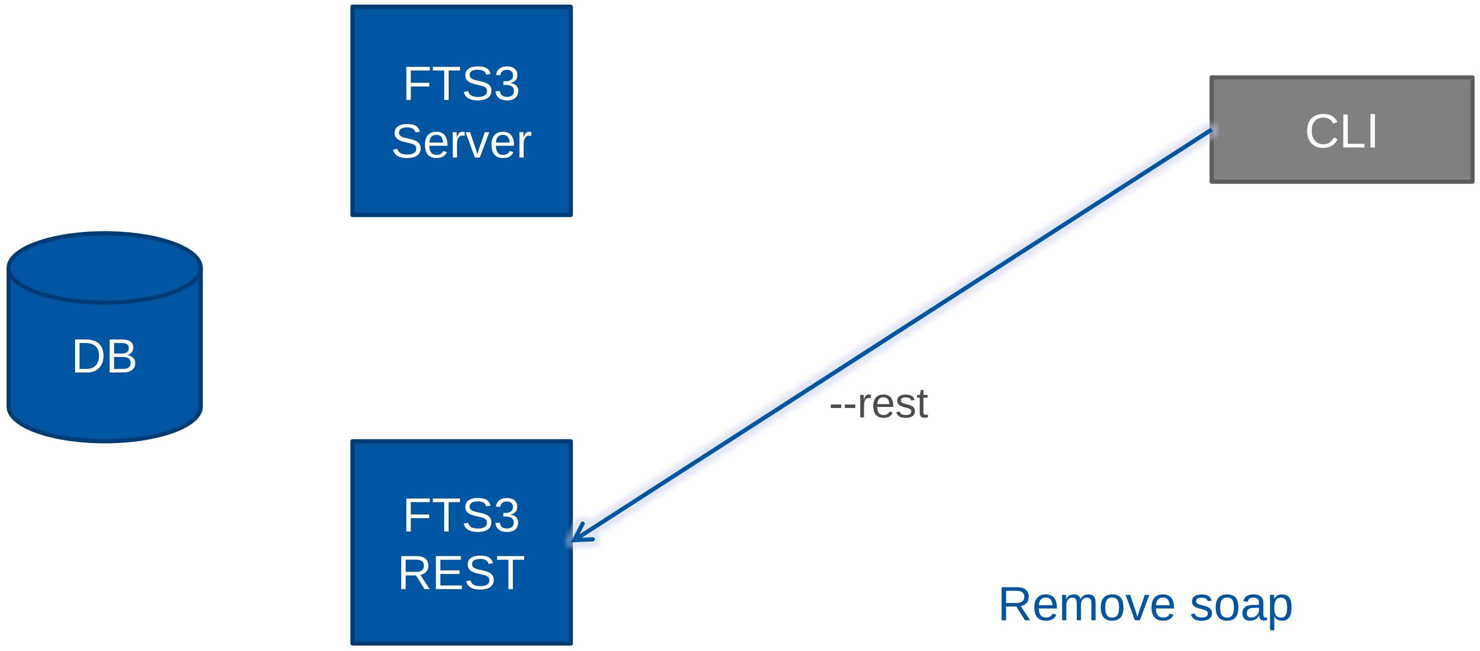
Complete support, switch default

Future (long term)



Disable soap connection

Future (long term)



Remove soap

Developer issues

- Log & History
  - Current log and database lifetime insufficient
    - On the short term, could the transfer log lifetime be expanded?
  - Querying the backup tables is impractical
    - Need reconsideration, and probably development effort
      - Table rotation? (one per week or day)
      - Store somewhere else?
  - (Speculative) Maybe even store both logs and job info together somewhere else
    - By this point, mostly job-id is needed for post mortem
    - But submission/finish time would be nice too

Developer issues

- Current monitoring lacking
  - How many connections per second? How many submission? Polls? Delegations? 500 errors? 400? 404? ….
  - If a server misbehaves, need to wait for someone to see it (experiment, sysadmin, developer peeking at logs)
  - This is an open question for FTS3 sites
    - You have the experience running services!
    - Can we agree on a base suitable framework/tool and start from there?
    - Or see what we got and choose the framework based on this?

KEEP
CALM
AND
COMMENT

CERN