FTS

File Transfer Service

Towards Tokens, QoS, Archive Monitoring and beyond

Mihai Patrascoiu
on behalf of the FTS team

CERN

FTS
File Transfer Service

Open Source software for reliable and large-scale data transfers within WLCG

Developed at CERN

Carles Garcia Cabot
Python Developer

Edward Karavakis
Project Leader

Mihai Patrascoiu
C++ Developer

FTS - Core features

Intuitive

- Simple interaction to submitting transfers via REST API
- UI-based WebFTS portal

Robust

- Checksums and retries are provided per transfer
- Deploy and run “zero config”

Flexible

- Multiprotocol support (HTTP, GridFTP, XRoot, SRM, S3, ...)
- Real-time monitoring and runtime config (via Web Admin)

Adaptive

- Runtime optimisation (maximise throughput without burning the storages)
- Priorities / Activities support for transfer classification

FTS - Latest Numbers

8 WLCG instances

BNL, CERN (4), FNAL, RAL, MIT

16 non-WLCG instances

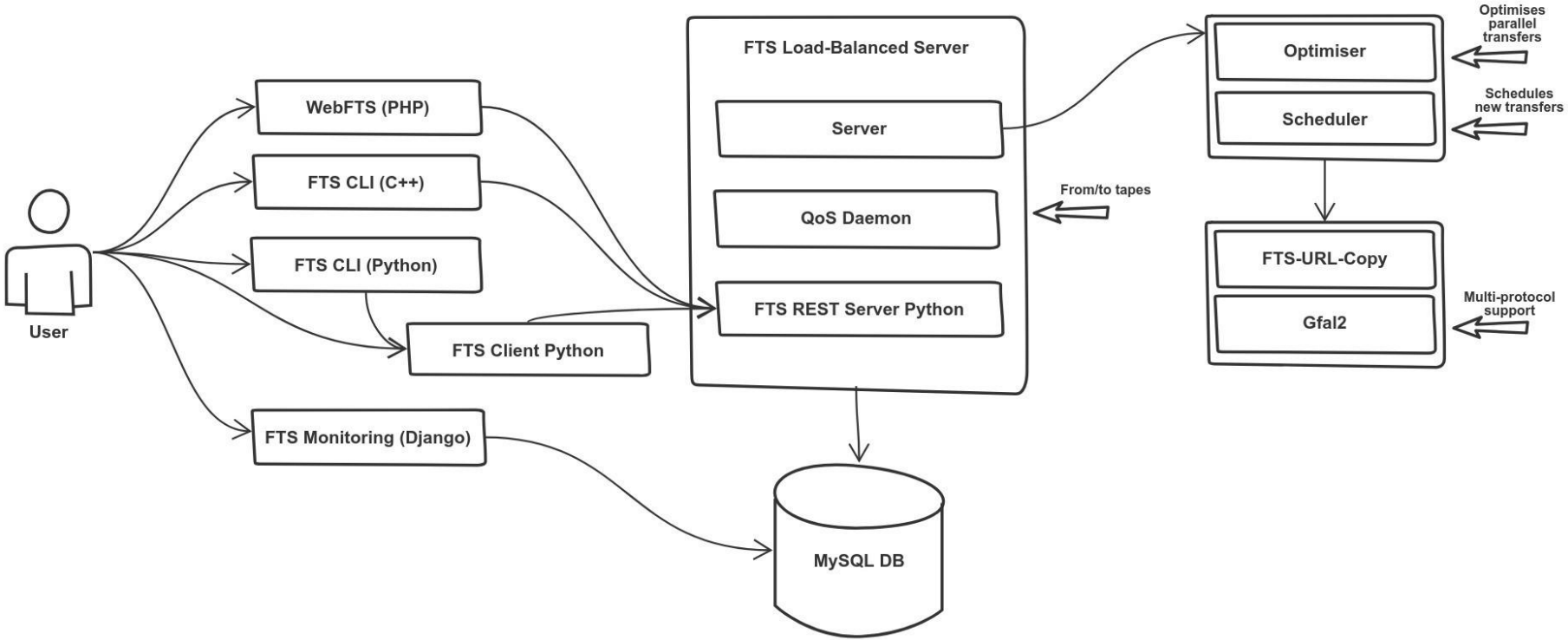
CERN (DAQ, Public), RAL, KEK(2), Imperial (also used by CMS), PIC, MWT2, CESNET (WebFTS + RCAuth), JINR, CNAF, SARA, SLAC, IHEP, Fermilab (containers), FENIX Research Infrastructure (Human Brain Project)

~36 Virtual Organizations

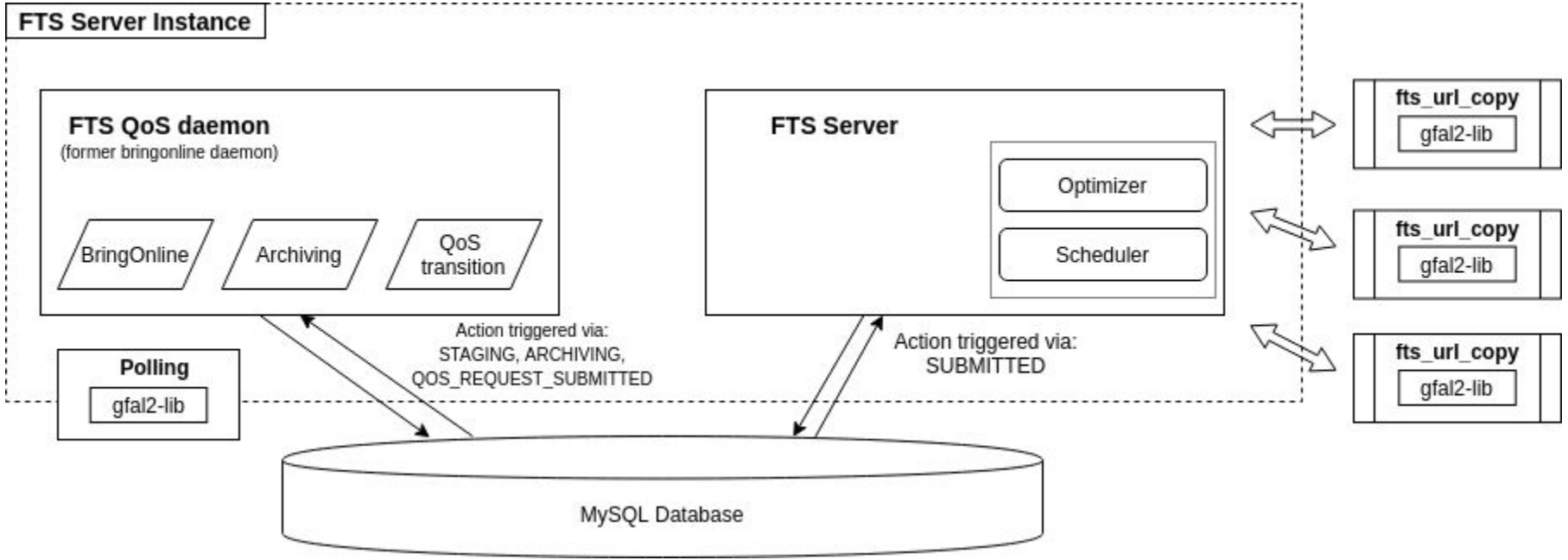
ATLAS, CMS, LHCb, AMS, NA62, Compass, ILC, Magic, Belle, Mice, Xenon, Snoplus, Gridpp, Dune, LZ, Solidexperiment.org, SKA, Ligo, Icecube, Elixir, NP02(part of Dune), CAST, ESCAPE, Eiscat.se, Virgo, Pierre Auger Observatory, BES III, JUNO, CEPC, FENIX-RI, CTA, T2K, Project8, ICARUS, FASER, Folding@Home

Transferred in total in 2018: 848 PBs and 986M files
Transferred in total in 2019: 1.27 EBs and 1.08B files
Transferred in total in 2020: 851 PBs and 760M files (so far) | *~65% by CERN FTS instances*

FTS ecosystem



FTS Server Instance Architecture



Updated Components

- FTS Rest Flask
  - New deployment of FTS Rest on Python Flask
  - Move the code base from from Pylons to Flask framework
  - Code migrated to Python3
  - Backwards compatible API

- Data Transfer Orchestrator (DTO)*
  - Former CDR (Central Data Repository): software used by NA62 for DAQ to move data to CASTOR
  - Generalised the functionality to support any VO for Data Acquisition
  - Code parallelisation, writing to CTA

*Attributions to Marco Borreto and Cristina Voineag

New Features

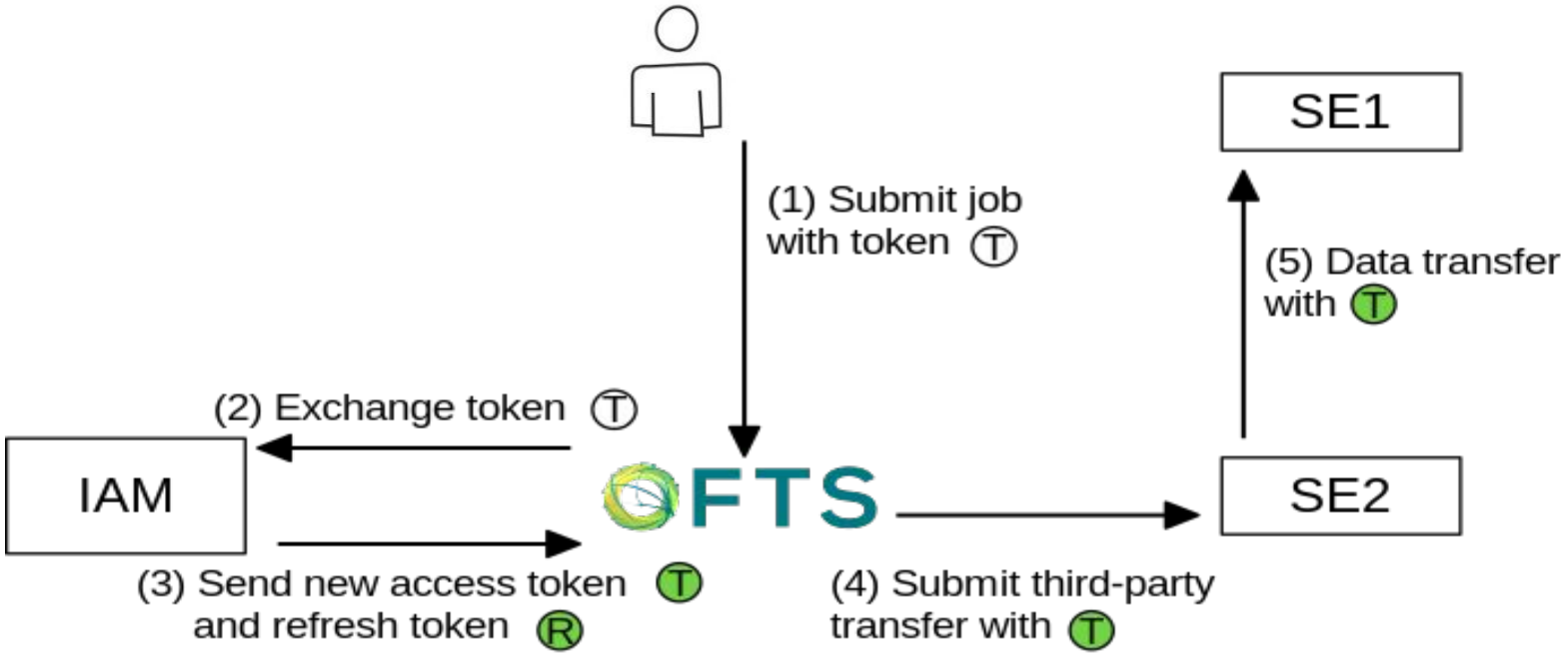
- OIDC Token support
  - FTS transfer workflow w/o x509 certificates
  - Multiple IAM support (XDC, WLCG)
- QoS daemon
  - Generalize the mechanism for transitions and polling
  - Supports CDMI QoS transitions
- Archive Monitoring
  - Follow a transfer until successful arrival (or not) on tape storage

OIDC Token Support

- A new authentication mode supported: OpenID Connect
- Token validation done offline or via introspection (based on PyOIC)
- Allows for Audience and Scope claims validation
- Refresh token + access token stored in database (to be used during transfer)
- Enabled by sending <access_token> to FTS-REST server

```
fts-rest-transfer-submit --access-token=<access_token>
        -s https://fts3-devel.cern.ch:8446/ <src> <dst>
```

OIDC Token Support



QoS daemon

- Triggers and monitors *long-standing* storage transitions
  - Tape → Disk, Disk → Tape, QoS transition
- Replaces the FTS-bringonline daemon
- Workload is split via database partitioning
- Interaction with storage endpoints done via Gfal2
- QoS transitions are implemented according to CDMI-QoS specification
  - Relies on Gfal2 QoS API (v2.18.0)

Horizon 2020 - eXtreme DataCloud
https://extreme-datacloud.eu/

QoS daemon

Staging

Active operation
Initiated by prepare request

Polling
Check file online

SRM: XAttr user.status
CTA: Xrd query prepare

Archiving

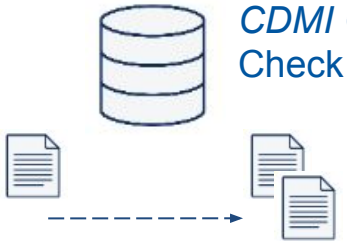
Passive operation
Initiated by transfer to disk buffer

Polling
Check file nearline

SRM: XAttr user.status
CTA: Xrd query prepare

QoS Transition

Active operation
Initiated by CDMI QoS Transition Request



CDMI QoS Polling
Check target QoS achieved

FTS Archive Monitoring

- FTS Server
  - executes file transfer as normal
  - If file transfer is completed and archive-timeout > 0 → ARCHIVING

- FTS QoS daemon
  - Fetches ARCHIVING files from the database
  - Polls ARCHIVING files until:
    - on-tape
    - timeout reached (timeout propagated to client)
    - error encountered (propagated to client)
  - Relies on Gfal2 for the polling query

Gfal2 Archiving API

- Introduced Archiving API for Gfal2
  - `gfal2_archive_poll (context, url, error)`
  - `gfal2_archive_poll_list (context, nbfiles, urls[], errors[])`
- Supported by XRootD & SRM plugins
- Returns:
  - 0 for archiving in progress
  - 1 for all files archived
  - 2 for files archived + files with error
  - -1 for all files with errors

Archiving Job

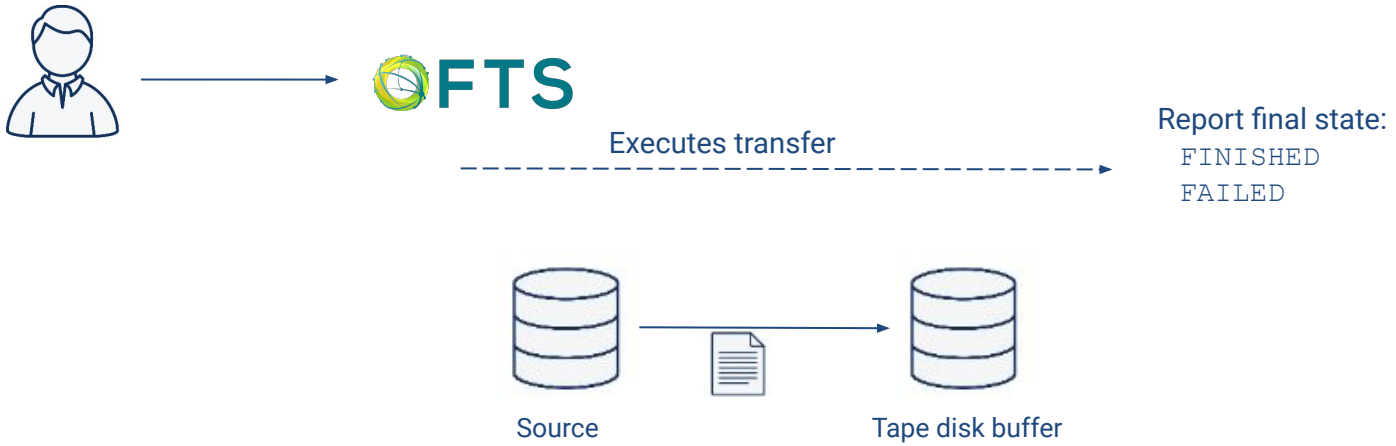
- New `ARCHIVING` state
  - File goes to `ARCHIVING` when transfer finished & `archive_timeout > 0`
  - Job goes to `ARCHIVING` when all files are in `ARCHIVING`
  - `ARCHIVING` transitions to `FAILED` or `FINISHED` (`CANCELED` support to be added)
  - Multihop jobs: only last hop is eligible for `ARCHIVING`

- Initiate by submitting `archive_timeout=<seconds>` to FTS-REST

```
fts-rest-transfer-submit --archive-timeout=<seconds>
        -s https://fts3-devel.cern.ch:8446/ <src> <tape_dst>
```

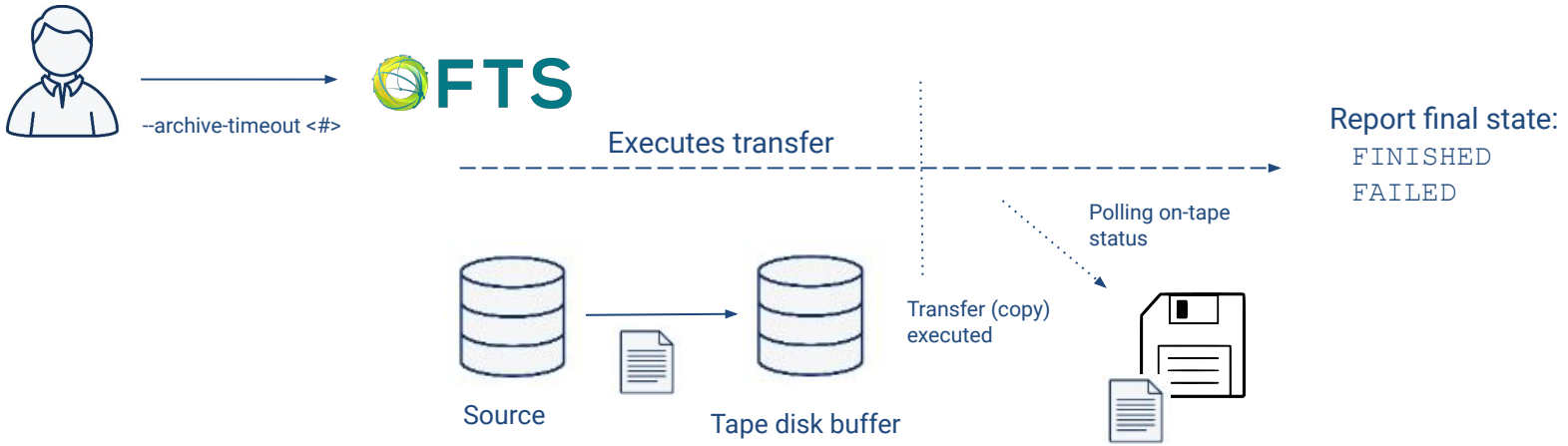
Transfer to Tape - Current Status

- Transfer considered successful when data arrives at tape disk buffer
- On-Tape status must be checked externally by **clients**



Transfer to Tape - Archive Monitoring

- Transfer considered successful when data arrives **on tape**
- On-Tape status checked by FTS
- Enabled by `--archive-timeout <seconds>` job argument



FTS-CTA Test Setup

Test archiving monitoring functionality on CTA PPS instance
File structure: 10MB, highly compressible, distinct checksum
Max volume: 100k files (1TB)

Tested on prototype instance
Multiple runs: gradual increment from 1k → 100k files
Identified bug in Archiving jobs recovery (fixed now)
Useful to assess functionality

Current testing on preproduction instance (FTS3-devel)
Planning more extensive tests with Rucio team
Assessing FTS Optimizer and Scheduler among multiple QoS daemons

FTS v3.10.0 (+ Gfal2 v2.18.3)

- QoS daemon to replace bringonline daemon
- Support for Archiving & (CDMI) QoS transitions
- Full support for TPC with token authorization
- Improved CLI clients

> Stress tested with CTA
> Available on FTS3-devel instance
> Planned release in November 2020

What's next?

- Active participation in DOMA Working Groups
  - Upcoming SRM → SRM transfers using HTTP-TPC
  - More granular authorization for JWT token submissions
  - Revise the *QoS language*
- Explore different backends for the QoS daemon
  → Decouple from MySQL database
- Deploy the new FTS-Flask REST server (Python 3)
- CentOS 8 releases for FTS & Gfal2

Thank you!

Contributions:

- Edward Karavakis
- Carles Garcia Cabot
- **Andrea Manzi*

- cern.ch/fts
- fts-devel@cern.ch
- https://gitlab.cern.ch/fts/fts3
- https://gitlab.cern.ch/dmc/gfal2

[Backup] QoS Transition

- FTS implements QoS transitions via the CDMI QoS specification
- The transition is initiated by a CDMI QoS request (request `target_qos`) and considered finished when the `current_qos = target_qos`
- The transition is monitored by FTS via CDMI QoS polling

```
fts-rest-transfer-submit --target-qos=<qos_value>
                         -s https://fts3-devel.cern.ch:8446/ <src> <dst>
```

[1] CDMI: Open data management interface, standardised by SNIA

[2] CDMI-QoS: Extensions to the interface brought by the IndigoDatacloud

[1] https://snia.org/cdmi
[2] https://www.rd-alliance.org/groups/storage-service-definitions-wg