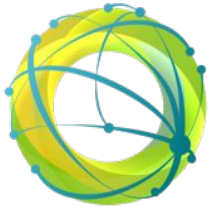
# FTS & XRootD Workshop

*The Cosener's House, Abingdon, UK, 9-13th September 2024*



FTS

File Transfer Service

# *FTS 2024: State of Affairs*

Mihai Patrascoiu
on behalf of the FTS team

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

Features:

- TPC Orchestration
- Tape Operations (over multiple protocols)
- Certificate and token auth
- Multihop transfers
- Transfer Optimizer
- Cloud support
- Python bindings + CLI clients

…and many others

# FTS Team

- Mihai Patrascoiu (Project Leader) [CERN]
- Steven Murray (Service Manager) [CERN]
- João Lopes (C++ / Python developer) [CERN]
- Shubhangi Misra (Web developer) [CERN]
- Louis Regnier (C++ / Python developer) *(1st July 2024)* [CERN]

…and thanks to many other past and present contributors

# FTS Ecosystem

*Projects under FTS umbrella*

| | | |
|---|---|---|
| - FTS | (Server + QoS daemon) | [C++] |
| - FTS-REST | (Submission server) | [Python, Flask] |
| - FTS-clients | (Python & CLI) | [Python] |
| - FTS-Monitoring | (Django Web UI) | [Python, Django] |
| - *webFTS* | *(decommissioned@CERN)* | [PHP] |

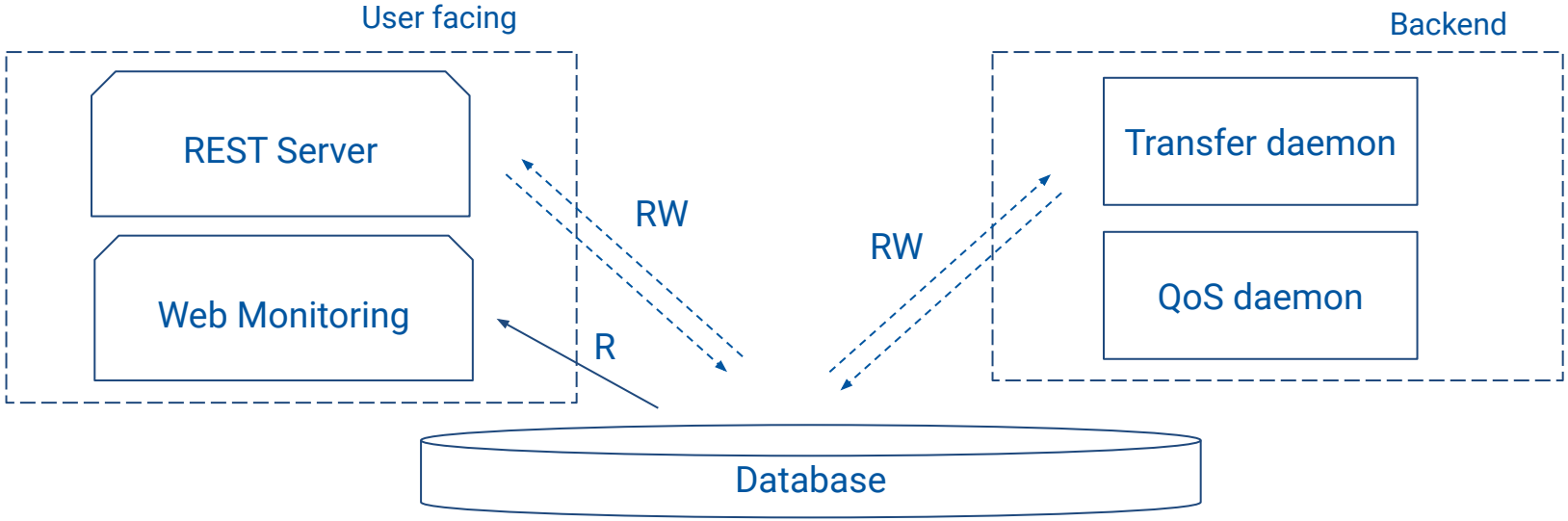
*Data Management Clients*

| | | |
|---|---|---|
| - Gfal2 | (Grid file access library) | [C++] |
| - Gfal2-python | (Python bindings) | [C++, BoostPy] |
| - Gfal2-util | (Python CLI) | [Python] |
| - Davix | (Grid HTTP client) | [C++] |
| - SRM-IFCE | (SRM interface for Gfal2) | [C, gsoap] |
| - CGSI-gSOAP | (gsi interface for Gfal2) | [C, gsoap] |

FTS and DMC clients published to PyPi, EPEL, Debian*

*Special thanks to Mattias Ellert

# FTS components (overly-simplified in 1 slide)




REST Server → Accepts submissions; queried for transfer status

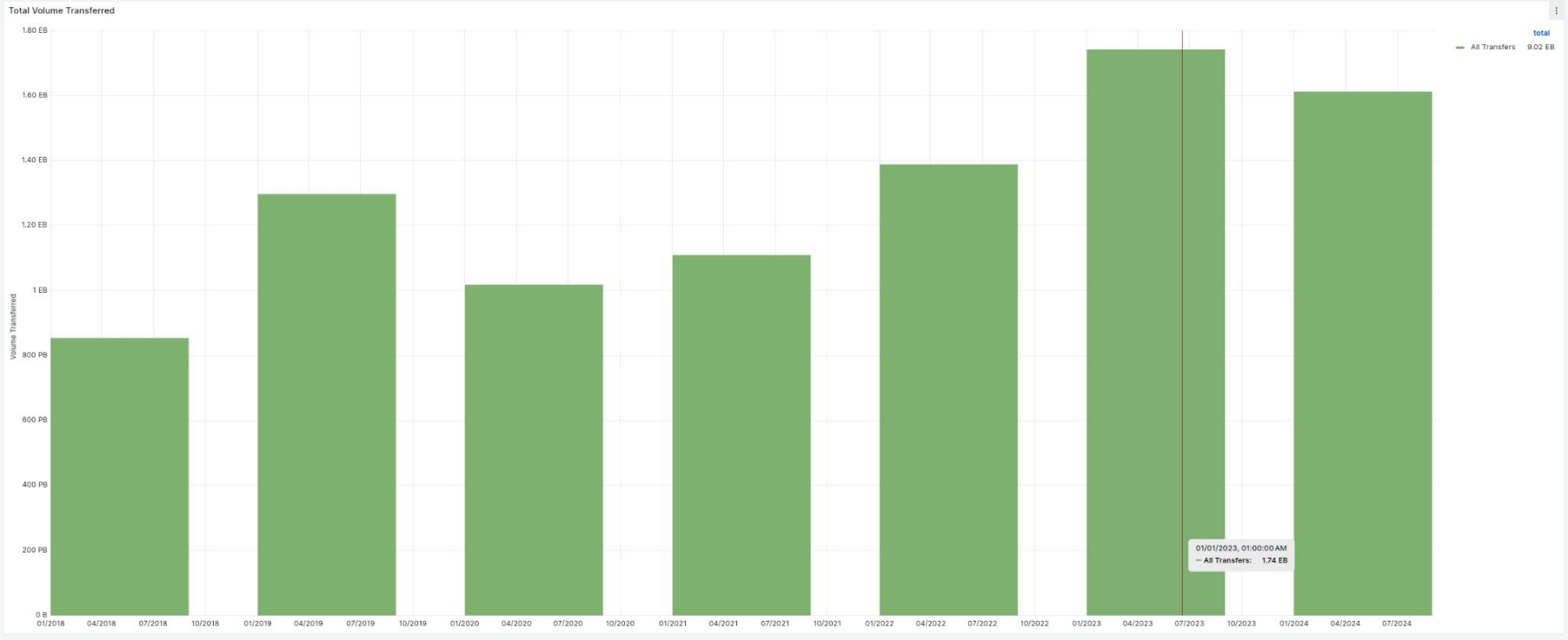
Web Monitoring → Visual webpage

Transfer daemon → Schedules transfers

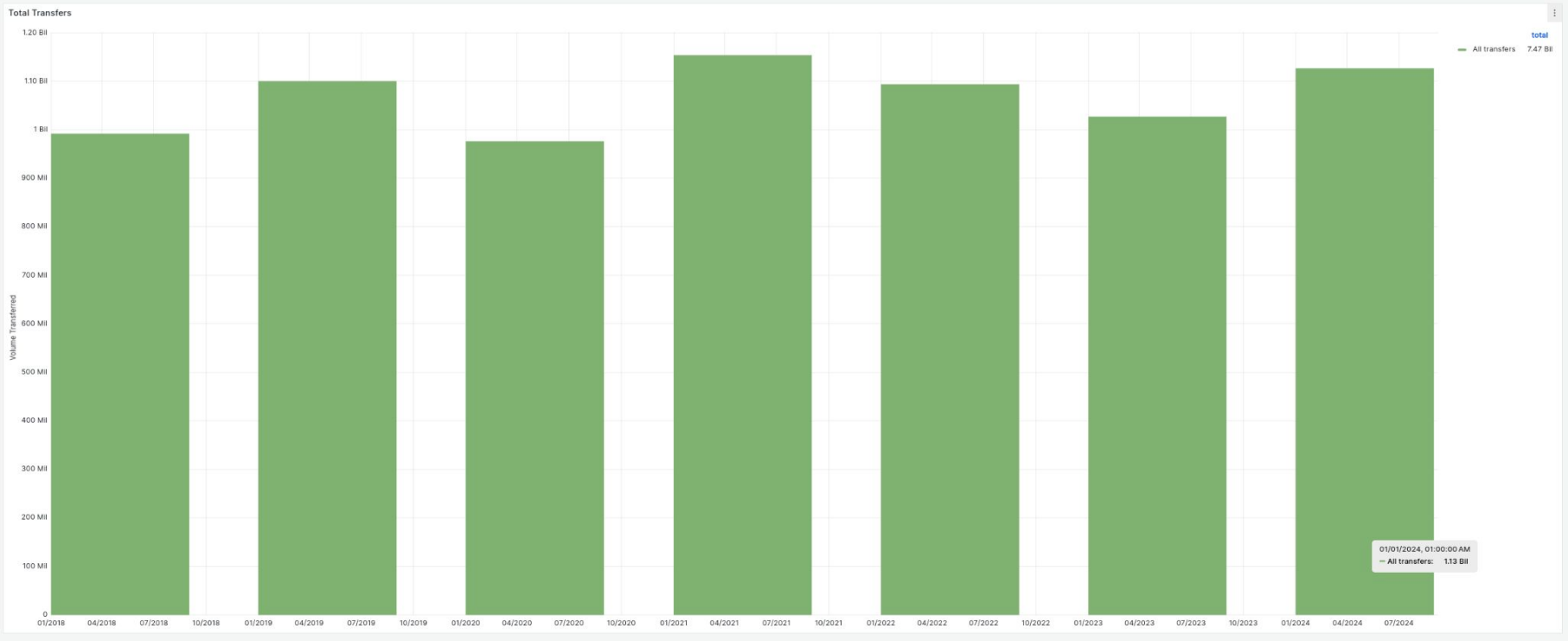
QoS daemon → Handles tape work

# *Service operations insights*

# FTS - Transferred Volume (2018-2024)



# FTS - Number of Transfers (2018-2024)



# FTS – Service changes at a glance

- Alma9 fully deployed (starting June 2024)

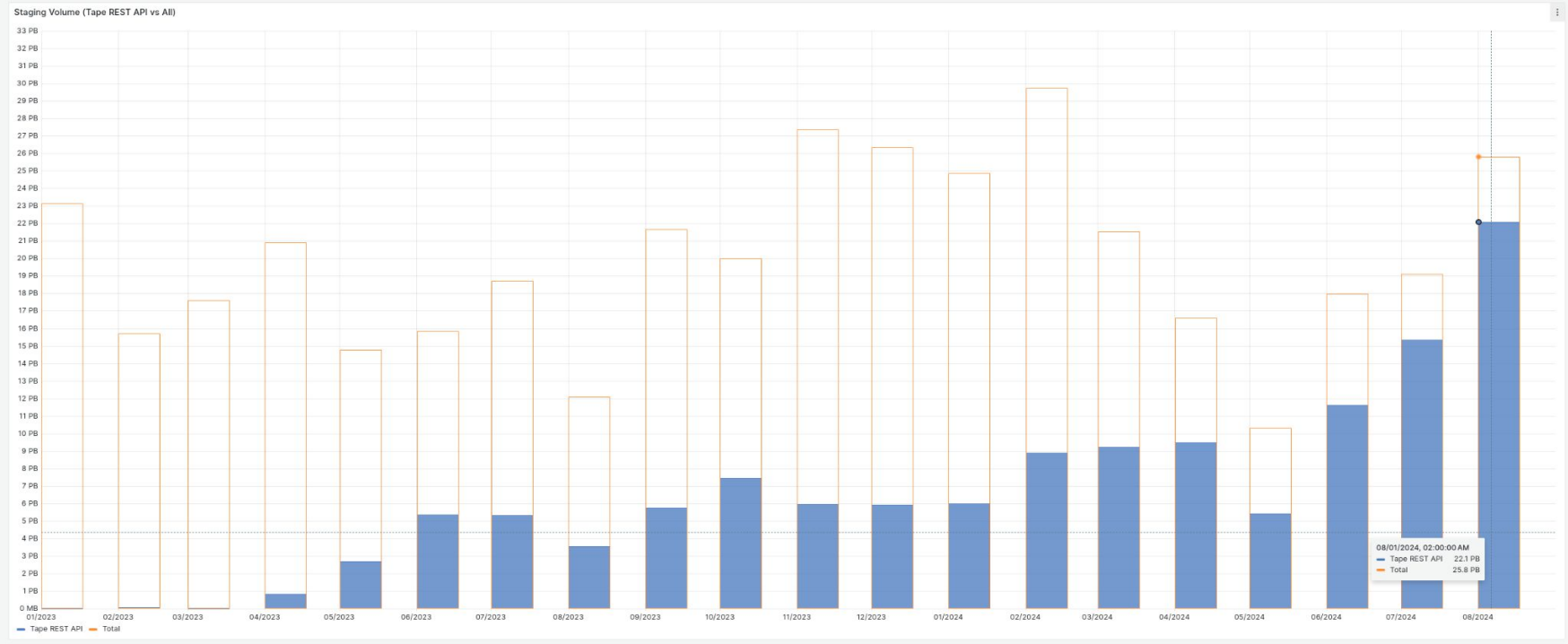
  → gradual deployment since March 2024

- GridFTP decommissioning on FTS instances for LHC experiments

  → remaining only on FTS3-Public @ CERN

- Increase of the HTTP Tape REST API usage

# Tape REST API adoption (2022-2024)



# *Development insights*

FTS File Transfer Service

## FTS v3.10 (2020)

- Addition of Archive Monitoring feature
- Appearance of FTS-QoS daemon (Bringonline daemon deprecated)
- First support for OIDC tokens introduced

## FTS v3.11 (2021)

- Destination file integrity report feature
- SE-issued tokens support built-in (Gfal2)
- Improvement of QoS staging database query (performance greatly improved after algorithm change)

## FTS v3.12 (2022)

- FTS-REST-Flask released (Python3 based)
- Movement to MySQL8 permitted (allowed by new FTS-REST)
- Schema updates w/o downtime
- Tape REST API

## FTS v3.14-alpha (Nov 2023) (tokens!)

- Beta version of token support
- Prepared for the DC'24 Workshop (Nov 2023)
- Fine-tuned for the DC'24 (Feb 2024)

## FTS v3.13 (2024)

- Alma9 migration (June 2024)
- Post-DC'24 improved token support
- Overwrite-when-only-on-disk feature
- Configurable TPC support level per SE

## FTS v3.14 (202x)

- Optimizer improvements
- Token "just-in-time" refresh
- Transfer agent improvements
- Groundwork for "Future of FTS"

# FTS – v3.13 Alma9 series

- Alma9 becomes the officially supported platform (released 30 May 2024)
- Core Server:
  - Advanced to C++20 features
  - GridFTP components made optional on RPM install
  - SciTag support /  TPC Support-level per SE
- FTS-REST-Flask
  - Python3.9 becomes the default platform (also tested for 3.10 → 3.12)
- Web Monitoring (moved to Django3)
- CI drops CC7 builds / Docker images updated to Alma9
- Custom dependencies rebuilt for Alma9 (+ newer versions)

# FTS – Packages and Platforms

**Server**

- `fts-server / fts-rest-server / fts-monitoring`
- Only available on Alma9 ! (nu support planned for RHEL8 / CS8)
- Packages (+ dependencies) only available via the FTS repositories

**Clients**

- `fts-rest-client` (Python) / ~~`fts-client`~~ (deprecated C++ client)
- `gfal2 / gfal2-python / gfal2-util / davix`
- Packages available via EPEL (covers EL8, EL9, active Fedora)
- EPEL packages: your best shot at FTS & DMC for other archs (e.g.: ARM)

# FTS – Configurable TPC Support per SE

- FTS v3.13 allows configuring the level of HTTP-TPC support an SE offers
  - Possible values: `Full Support` (default), `Pull only`, `Push only`, `Not Supported` (e.g.: Cloud Storage)
  - When orchestrating the transfer, FTS creates the possible HTTP-TPC combinations
  - Previously, only possible by playing with Gfal2 config files

Example:

$SE_{SRC}$ (Full support)[push & pull]
$SE_{DST}$ (Push only)

Transfer $SE_{SRC} \rightarrow SE_{DST}$ (HTTP-PUSH)

$SE_{SRC}$ (Not supported)
$SE_{DST}$ (Push only)

Transfer $SE_{SRC} \rightarrow SE_{DST}$ (Stream)

# FTS – Overwrite-when-only-on-disk

- New submission and overwrite mode (since FTS v3.13.1)
  - Add `--overwrite-when-only-on-disk` flag
  - FTS will overwrite the file **if the file has only disk locality**
- Will streamline CMS workflows, available for other use-cases
- Flag is enabled per-SE (via the SE configuration page)

```
$ fts-rest-transfer-submit  --overwrite-when-only-on-disk
                                        -s <fts-server> <src> <dst>
```

# *DataChallenge'24 Reflections*

# FTS & DC'24

## *The good*

- FTS upheld its contract to saturate StorageEndpoints (according to configured limits)
- FTS successfully pioneered WLCG transfers with token support in a high stress environment

## *The bad*

- Saturating StorageEndpoints does not correlate with maximizing throughput
- No way to prioritize faster links (T0→T1) over slower ones (T0→T2)

## *The ugly*

- Problems with congested database queries

  ( → token queries further refined post-DC’24)

- FTS3-ATLAS instance overloaded

  *( → must impose limits and refuse submissions)*

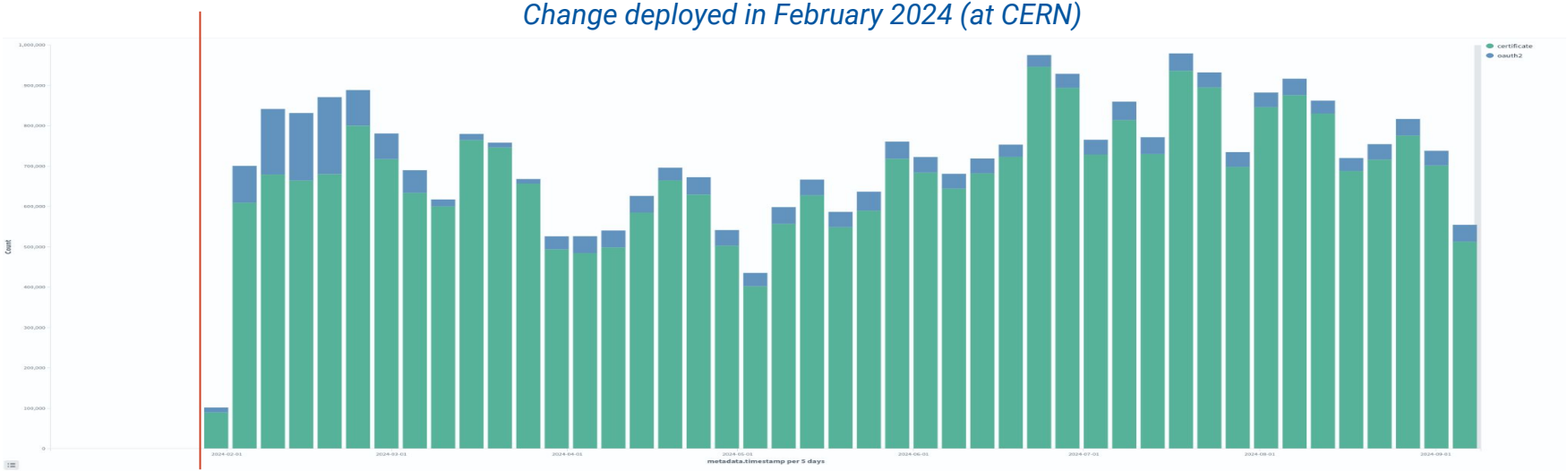
- FTS3-ATLAS Optimizer malfunctioned
  - Optimizer would never finish a cycle through the database
  - No link decision would ever be updated

# FTS & DC’24: Auth method reporting

- Added new field to MONIT reporting:

```
auth_method=<certificate|oauth2>
```

*Change deployed in February 2024 (at CERN)*



# *Optimizer Improvements*

# DC’24 Optimizer Malfunction

- Optimizer runs in the same process as the server
  - A server restarted implies an Optimizer restart
- Optimizer works only with one link (src, dst pair) at a time
- Optimizer works sequentially through all the links

---

- On FTS3-ATLAS, during DC’24:
  - 600 links, ~6 seconds / per link → 3600+ seconds for a full Optimizer cycle
- At this rate, one Optimizer decision / hour

→ effectively running without an Optimizer

→ all link decisions frozen at last value

# Optimizer Improvements

- Turn Optimizer into standalone process
  - Can supervise separately to Transfer process (different restart needs, etc)
- Make Optimizer per-link work multithreaded
  - All computations are bound within the link, links are independent

    → perfect for parallelism
- Improved database index used by the Optimizer (thanks to Joao for discovering this!)
  - Despite per-link computations running against a recent time window, indexing based on the “finish_time” field was not the correct choice

# Optimizer Improvements – Results

| **Threads** | **Single-threaded** | **Multi-threaded** (n=10) | **Single-threaded** | **Multi-threaded** (n=10) |
|---|---|---|---|---|
| **Index** | Non-efficient | Non-efficient | Efficient | Efficient |
| **Measurement** | 600 links ~ 600 seconds | 600 links ~ 120 seconds | 600 links ~ 120 seconds | 600 links ~ 20 seconds |
| | ~ 1 second / link | ~ 0.2 seconds / link | ~ 0.2 seconds / link | ~ 0.03 seconds / link |
| **Speed-up factor** | **1x** | **5x** | **5x** | **30x** |

# *Core Problems of FTS3*

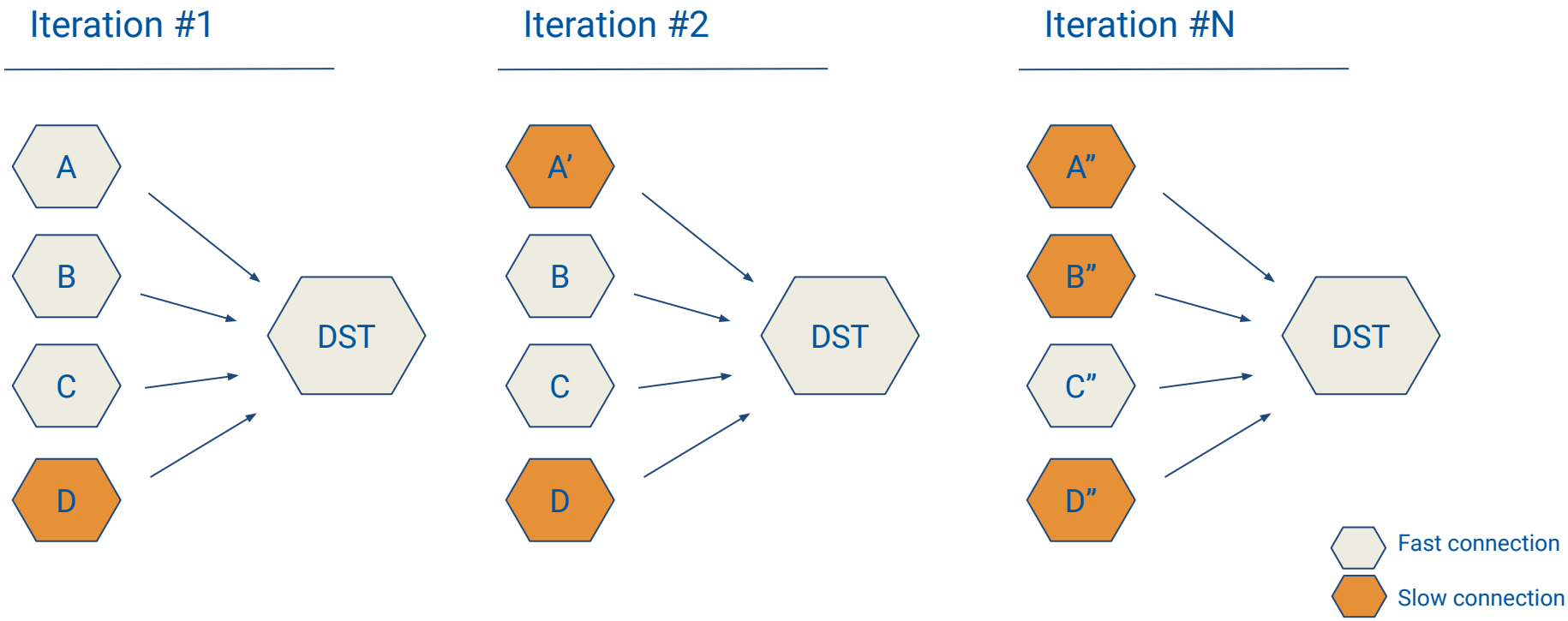
# I. Scalability

- In order to make the Scheduling decision, FTS sifts through most of the database
  - Time and time again, a breaking point arrives in the proximity of 2-3M submitted transfers
  - Difficult to reproduce production load with artificial tests (e.g.: test with 5M submitted did not reveal any slowdown)
  - Time complexity a function of number of links x submitted transfers
- A new approach is needed, where the scheduler can take the decision in constant O(1) or logarithmic O(logN) time

# II. Scheduling per link, not per SE

- The FTS Scheduling is structured around the link (src, dst pari) concept
  - Within a link, shares are assigned between VOs
  - Within a VO share, priority channels are created
  - Within priority channels, further shares are assigned between activities
- By the end of the scheduling cycle, FTS knows which transfers are next within each link
    - …but SEs are involved in multiple links
    - Practice shows SE limits are the limiting factor (most of the time)
- Round-robin(-ish) scheduling between links ensues
    - This approach will eventually saturate the destination SE with only slow links
    - In other words, the slow ones hang with you

# II. Scheduling per link, not per SE



# III. Optimizer diminishing returns

- The Optimizer’s role is to decide how many transfers are assigned per link

- Simple approach: While throughput is good, increase concurrency

  When throughput degrades, decrease concurrency

- The approach ensures concurrency stabilizes just before “breaking point”
  - Unnecessary stress on the Storage Endpoints for the same throughput

# III. Optimizer diminishing returns

# *Looking towards the Future of FTS*

# Towards the “Future of FTS”

- Work already started into improving the data store
- Work already started into improving the time complexity of the scheduling decision
  - Implies a complete Scheduler rework
  - New design will be designed with StorageEndpoint needs in mind
- Iterative approach: components being extracted one-by-one from the main “FTS server monolith”
- New approach will see most services running as standalone processes
  - Greater decoupling and possibility to scale out the service

# Conclusions

- Three main activities underway in 2023 - 2024
  - Tokens, Alma9 migration, “Future of FTS”
- Upcoming FTS v3.14.0 release (est. 2024 / early 2025)

  starts the decoupling of the main services into standalone units
  - Greatly improved Optimizer
  - Last missing token component: “just-in-time” refreshing
  - Reducing the transfer agent’s reliance on Gfal2
- The new scheduler will model the current Experiments and SEs needs
- Further discussion awaited:

  → Experiment Input to FTS (Tuesday @ 11:00)

  → Token discussion (Wednesday @ 11:00)

# Thank you!

- Issue tracking: JIRA FTS / DMC
- Code: Gitlab (CERN) → mirrored on Github

https://gitlab.cern.ch/fts/fts3
https://gitlab.cern.ch/dmc/gfal2
https://gitlab.cern.ch/dmc/davix

- **Discussion:** fts-devel@cern.ch / dmc-devel@cern.ch
- **Announcements:** fts3-steering@cern.ch
- **Support:** fts-support@cern.ch / ServiceNow / GGUS / ~IT-FTS Mattermost

→ ***Will create SNow tickets for all support requests***

- Documentation:

cern.ch/fts3-docs
cern.ch/dmc-docs
cern.ch/fts