EPJ Web of Conferences **295**, 01031 (2024) https://doi.org/10.1051/epjconf/202429501031
CHEP 2023

FTS Service Evolution and LHC Run-3 Operations

Steven Murray[1,*] *Mihai* Patrascoiu[1,] *Luca* Mascetti[1,] *Joao Pedro* Lopes[1,] *Shubhangi* Misra[1,] *Eraldo* Silva Junior[2,]

[1]CERN, Geneva, Switzerland
[2]CBPF, Brazil

Abstract. The File Transfer Service (FTS) is a software system responsible for queuing, scheduling, dispatching and retrying file transfer requests, it is used by three of the LHC experiments, namely ATLAS, CMS and LHCb, as well as non LHC experiments including AMS, Dune and NA62. FTS is critical to the success of many experiments and the service must remain available and performant during the entire LHC Run-3. Experiments use FTS to manage the transfer of their physics files all over the World or more specifically all over the Worldwide LHC Computing Grid (WLCG). Since the start of LHC Run-3 (from 5th July 2022 to 31st August 2023), FTS has managed the successful transfer of approximately 1.2 billion files totalling 1.83 Exabytes of data.
This paper describes how the FTS team has evolved the software and the deployment in order to cope with changes in implementation technologies, increase the efficiency of service, streamline its operations, and to meet the ever changing needs of its user community. We report about the software migration from Python 2 to Python 3, the move from the Pylons web development framework towards Flask and the new database deployment strategy to separate the handling of the critical operations from the long duration monitoring queries. In addition, during 2022 a new HTTP based protocol has been finalised that can now be used between FTS and compatible WLCG tape storage endpoints.

1 Introduction

The File Transfer Service (FTS) [1] is a software system responsible for queuing, scheduling, dispatching and retrying file transfer requests all over the World. A single installation of FTS can be used by multiple user communities or Virtual Organizations (VOs). Figure 1 shows the architecture of FTS. A single FTS instance is a cluster of identical head-node machines that share a common DNS alias and are connected to a single MySQL or compatible (MariaDB) database. End users can use their favourite web browser to monitor and configure FTS. They can use client programs such as `fts-rest-transfer-submit` to submit, check the status of and cancel file transfers. Each FTS head-node runs an Apache web-server (`httpd`) in order to provide the HTTPS interface to its end users. The web-server hosts two web applications: `fts3rest` and `ftsmon`. The first provides the FTS REST API for submitting and cancelling transfers, as well as configuring FTS, whereas the latter provides human readable web pages for monitoring the current status of file transfers, reviewing file transfer logs and transfer statistics. Each head-node also runs a Quality of Service (QoS) or `fts_qos` daemon

*e-mail: steven.murray@cern.ch

and an FTS Server or `fts_server` daemon. The `fts_qos` daemon is responsible for executing tape related actions such as polling a storage endpoint to detect when a file has been written to tape. The `fts_server` daemon is responsible for scheduling and executing disk to disk transfers. The shared MySQL database is used to persist the state of all queued and on-going file transfers plus those that have been finished for less than a few days.

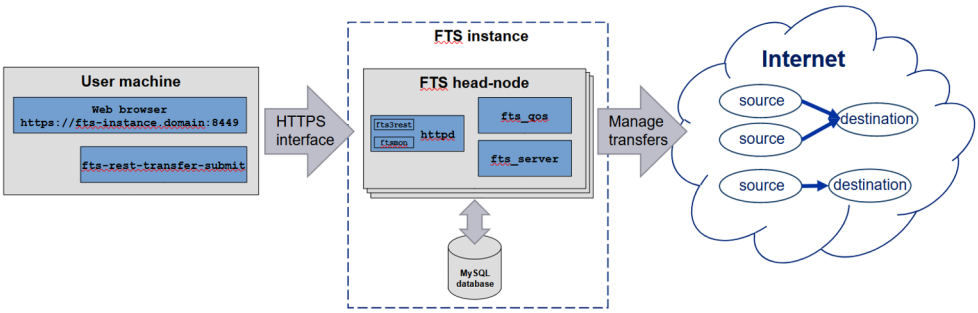

Figure 1. The FTS architecture

FTS is a critical component in the successful operation of three LHC experiments, namely ATLAS, CMS and LHCb. The FTS instances used by these experiments must remain available and performant during the entirety of LHC Run-3. Non-LHC experiments also use FTS, notable examples being AMS, Dune and NA62. FTS allows experiments to concentrate on deciding which files need to be transferred where, whilst it ensures these files are sent safely and the underlying network and storage systems are used efficiently.

CERN hosts six production FTS instances. Three are dedicated to and named after the ATLAS, CMS and LHCb experiments. The remaining three are named Public, DAQ and Pilot and are used by multiple VOs. Public is used by non-LHC experiments and individual end users, DAQ is used for data acquisition specific workflows and Pilot is used to validate new experiment workflows. FTS instances are deployed by other organisations including Brookhaven National Laboratory (BNL) and Fermi National Accelerator Laboratory (FNAL) in the USA. It should be noted that BNL can be used as a failover site if the CERN ATLAS instance fails and FNAL can likewise be used if the CERN CMS instance fails.

Figure 2 shows FTS orchestrated the successful transfer of 1.2 Billion files totalling 1.83 Exabytes from the start of LHC Run-3, 5th July 2022, until 31st August 2023.

This paper describes how the FTS team has evolved the FTS software and the way in which it is deployed in order to cope with changes in implementation technologies, to increase service efficiency, to streamline operations procedures and to meet the ever changing requirements of the user community. Specifically we report on the software migration from Python 2 to Python 3, the move from the Pylons web development framework to Flask, the new database deployment strategy to improve service performance and stability, the addition of support for the new HTTP based Tape REST API, the phasing out of the GridFTP protocol, and finally the future of the FTS project.

2 Migration from Python 2 to Python 3

The current deployment platform of FTS is CentOS 7. The default version of Python on CentOS 7 is 2.7.5, however Python 3.0 was released on 3rd December 2008 and Python 3.6.8 can easily be installed and used on CentOS 7. More so, Python2 end-of-life was announced

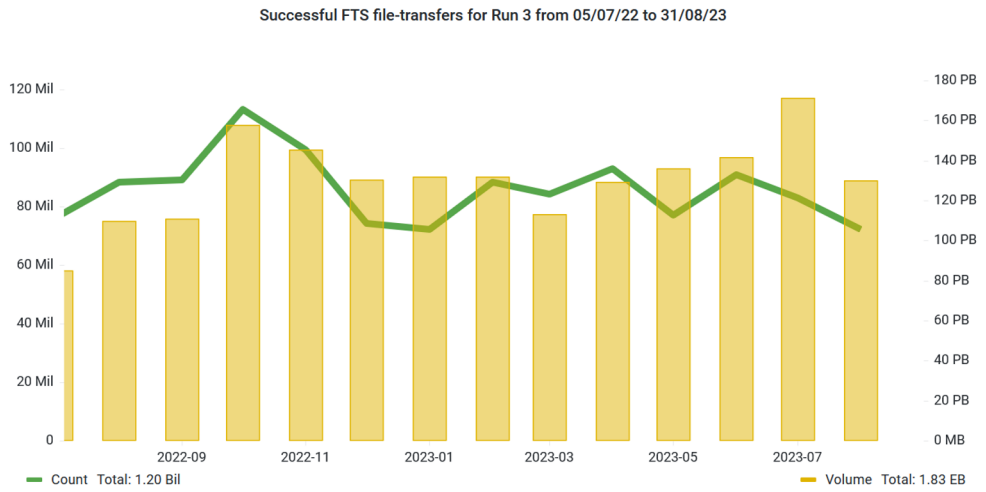

Figure 2. Successful FTS file-transfers from 5th July 2022 to 31st August 2023

for 1st January 2020 [2]. The FTS team therefore migrated all Python 2 code within FTS to Python 3 during 2020-2021, with first deploy to production in 2022. This included the FTS client applications (e.g. `fts-rest-transfer-submit`) , the FTS REST web application (`fts3rest`) and the FTS Web Monitoring application (`ftsmon`).

The migration from Python 2 to 3 did not bring any functional changes to end-users. The migration did help future proof the FTS code base, allowing the FTS project to move to the newer MySQL 8 version and keeping in sight the planned end of life of CentOS 7 on 30th June 2024. FTS will most probably move to a Linux version based on Red Hat Enterprise 9. The default Python version will then be 3.9.

The new Python 3 version of the FTS clients is distributed via the Python Package Index (PyPI) and can be installed using:

```
$ pip install fts3
```

A new RPM is also available in both the CERN FTS repository [3] and in the Extra Packages for Enterprise Linux (EPEL) repository [4]. Once access to at least one of these repositories has been configured the new RPM can be installed using:

```
$ yum install fts-rest-client
```

Please note that the old C++ client package will be deprecated after FTS version 3.12.

3 Migration from Pylons to Flask

Pylons [5] is a framework for developing web applications in Python. Pylons was used to implement the FTS REST web application (`fts3rest`). The Pylons project is now in maintenance-only mode and was never ported to Python 3 which made it incompatible with the migration of FTS from Python 2 to 3. Even though the Pyramid project [6] is the natural successor to Pylons, the FTS team decided to migrate to the Flask [7] framework because it has a big user community, good documentation, is simple, has a rich ecosystem and integrates well with SQLAlchemy [8].

The Flask version of the FTS REST web application is a brand new GitLab project and can be visited here:

`https://gitlab.cern.ch/fts/fts-rest-flask/`

No changes were made to the FTS REST API. The goal was to copy the structure and code of the original version as much as possible in order to avoid introducing any regressions.

The Flask version of the web application has been running in production at CERN since February 2022. The application RPM can be downloaded from the FTS YUM repository using:

```
$ yum install fts-rest-server
```

4 Database deployment strategy

The MySQL databases of the CERN FTS instances are hosted by the Database On Demand (DBoD) service [9] of the CERN IT department. This delegation of duty reduces costs by centralising the knowledge and people required to run hundreds of MySQL database at CERN. It also allows the FTS team to concentrate on developing, deploying and operating the FTS software.

Running an FTS instance on top of a database hosted by a "cloud" like service such as CERN DBoD does not come without its challenges. The read rate of the MySQL databases hosted by the DBoD service can drop as low as 20 MB/s if the underlying database files are highly fragmented. Unfortunately this happens naturally to the largest tables of the FTS database which are used to enqueue and dequeue millions of file transfers which translates to the execution of millions of `INSERT` and `DELETE` SQL statements which cause fragmentation over time. FTS is a legacy system that has built up a backlog of inefficient SQL queries. The FTS and DBoD teams have made FTS more efficient over the last 2 years in order to keep it within the performance bounds of the DBoD service. This work is useful to other organisations that also wish to run FTS on top of a database hosted by a "cloud" like service. In particular the two teams have:

- Migrated the CERN FTS databases to MySQL 8.0 which enables the use of on-line Data Definition (DDL) operations or schema changes. This removes the need to run the databases on high-speed locally connected flash drives. The schema changes can be applied whilst the FTS service is still running.
- Optimized the SQL query used to decide which files to stage from tape.
- Automated the execution of `OPTIMIZE TABLE` to run once a week to avoid the fragmentation of the large FTS tables exposed to high rates of `INSERT` and `DELETE` statements. This weekly cleanup allows for the MySQL RAM cache to warm up in less than 20 minutes after a cold start. The previous setup could take over an hour to warm up its cache.
- Deployed a read-only replica of each FTS production database. The main databases were then dedicated to the execution of mission critical queries or On-Line Transaction Processing (OLTP) queries. The newly deployed replicas were dedicated to execution of monitoring queries or On-Line Analytical Processing (OLAP) queries. This separation of query types means the execution of fast mission critical database statements is isolated from the execution of relatively slow and non-critical monitoring statements.

5 The Tape REST API

Data files are written to and read from disk storage systems using synchronous operations such as open, read, write, seek and close. Tape systems are asynchronous in nature. They are composed of one or more disk staging areas in front of one or more tape libraries. To archive a file to tape, it is first synchronously written to a disk staging area. This is followed by the asynchronous step of waiting for the file to be safely written to tape. To retrieve a file from tape a prepare or bring on-line request is sent to the tape library. This is then followed by the asynchronous step of waiting for the file to be written to the disk staging area. The file is then synchronously copied out. Due to this asynchronous workflow, tape storage systems usually have the following additional operations with respect to their pure disk counter parts:

- Check a file is on tape.
- Queue a file to be retrieved from tape.
- Cancel a request to retrieve a file from tape.
- Check a file is on disk.
- Release a file from disk whilst keeping a safe copy on tape.

These operations are defined in one form or another within the Storage Resource Manager (SRM) protocol [10].

The Worldwide LHC Computing Grid (WLCG) [11] community have decided to replace the legacy SRM protocol with a modern and uniform way of managing tape data movements across the World. This new way is the HTTP based Tape REST API.

The Tape REST API was jointly developed by the EOS [12], CTA [13], dCache [14], StoRM [15], FTS and GFAL2 [16] projects. The result was the creation of the WLCG Tape REST API specification [17]. FTS version 3.12.2 and GFAL2 version 2.21.0 introduce full support for the Tape REST API. Full support was deployed at CERN in January 2023. It should be noted that in addition to the above mentioned tape operations, the API also allows experiment and end users to pass both archiving and staging metadata to the tape storage endpoints. Archiving metadata could be used to label high and low priority tape data and also data that should be collocated together on the same set of tapes. Staging metadata could be used to label the retrieve priority of data.

FTS facilitates experiments and end users in accessing their tape storage systems. The `fts_qos` daemon is responsible for managing tape specific operations. This daemon uses the GFAL2 library to implement these tape operations. All of the code used to support the new Tape REST API is contained within the GFAL2 library. FTS needed a small amount of changes and the latest version 2.21.0 of GFAL2 in order to offer support for the new API.

6 GridFTP is being phased out

Figure 3 primarily plots the monthly counts of GridFTP and HTTP transfers managed by the CERN FTS instances between 1st January 2020 and the 31st July 2023. The plot clearly shows GridFTP is being phased out and HTTP is being used more. In order to put the two protocols into perspective the figure also plots the combination of all transfer types, where all refers to GridFTP, HTTP, XRoot and S3.

7 The future of FTS - Exascale tokens for FTS

X509 proxy certificates are currently the main authentication mechanism used by clients connecting to FTS and by the transfers that FTS schedules and executes. This will change

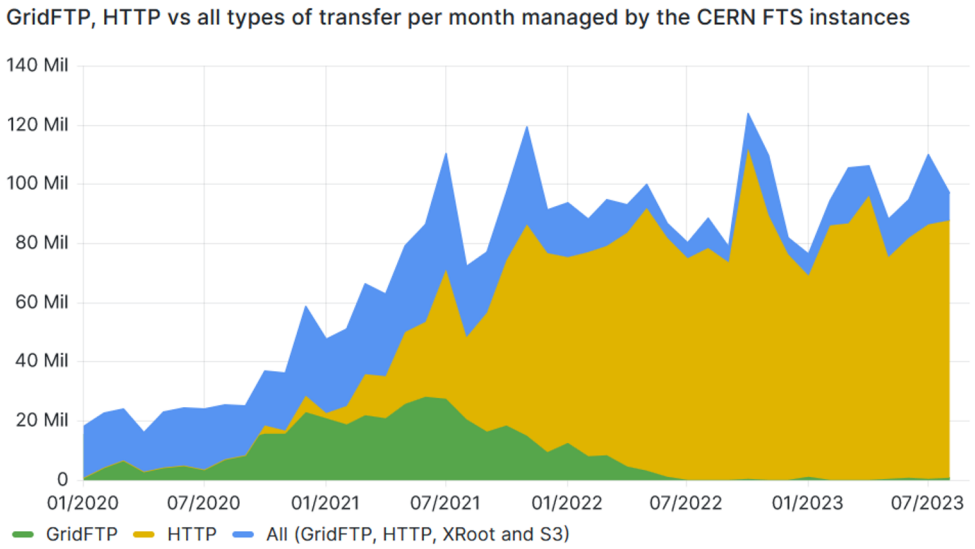

Figure 3. GridFTP, HTTP vs all types of transfer per month managed by the CERN FTS instances from 1st January 2020 to 31st July 2023

in the future as the WLCG community moves towards the use of token based authentication [18].

FTS currently provides "unrefined" support for authenticating HTTP file transfers using tokens. Building on this work the FTS team started the "Exascale tokens for FTS" project in February 2023. This project will last 2 years and should enable end users and storages to transition from X509 proxy certificates to a 100% token based authentication workflow. Given the central nature of FTS within the WLCG, this project is a prerequisite for the WLCG's transition from X509 certificates to tokens. The main stakeholders of the "Exascale tokens for FTS" project are:

- Worldwide LHC Computing Grid (WLCG)
- Large Hadron Collider (LHC) experiments
- Storage providers
- Authentication and Authorization services
- European Grid Infrastructure (EGI) community
- Open Science Grid (OSG) community

Discussions in the first half of 2023 centered around how WLCG specific token support within FTS should be. On the one hand implementing WLCG specific workflows would offload development work from Rucio [19], DIRAC [20] and small to medium sized experiments and would also centralize that work within FTS. On the other hand being fully generic would allow FTS to be used by "everything". The conclusion of these early discussions is for the FTS team to implement a fully generic solution in time for the WLCG Data challenge planned for February 2024.

8 The future of FTS - A true micro-service deployment model

FTS is composed of three main daemons, Apache (`httpd`), FTS server (`fts_server`) and FTS QoS (`fts_qos`). These daemons have different responsibilities and different workloads, however all three must be installed on each and every FTS head-node machine. In order to scale a given FTS deployment in a cost-effective manner one should be able to deploy different types of daemons on different machines. If the highest load is handled by `httpd` then it should be possible to deploy that daemon on its own on one or more dedicated machines. The remaining two daemon types can then be deployed on a smaller set of machines. The FTS team plan to move FTS to a true micro-service deployment model in a future release. Figure 4 shows this graphically.

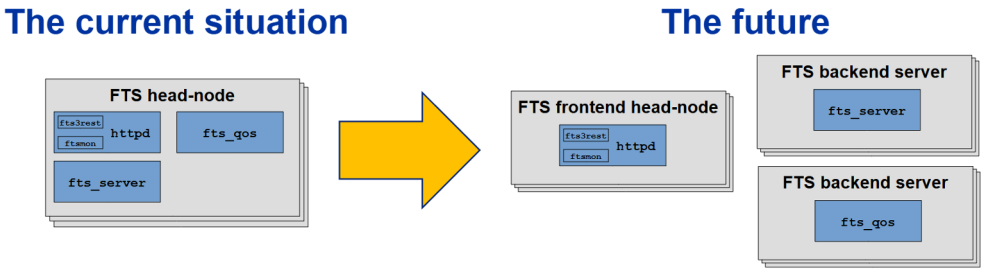

Figure 4. Moving FTS towards a true micro-service deployment model

9 The future of FTS - Outlook

The FTS and GFAL2 projects must continuously look for and remove legacy code, protocols and data transfer workflows. Without this effort the two projects would become too large and unwieldy. In this regard FTS and GFAL2 are closely watching the phase out of the GridFTP protocol and the replacement of the SRM protocol with the HTTP Tape REST API. The projects hope to be able to drop support for these two protocols in the future.

The scheduling logic of FTS is critical to its success and usefulness. The current architecture of this logic splits scheduler decisions across all of the head-nodes within an FTS instance. This split unfortunately breaks the strict FIFO order between some transfer requests. The FTS team hope to modify this logic and architecture in order to make it more predictable.

Amongst the many file transfers being carried out across the WLCG, there are frequent cases of multiple source storage endpoints sending files to a common destination. End users have identified the need to prioritise some of these source storage endpoints over others. In particular users have pointed out that tape storage endpoints with relatively small output buffers should be prioritised over pure disk storage endpoints in order to avoid the need to repeatedly retrieve the same files from tape due to garbage collection removing those files before they are consumed by the next storage endpoint. The FTS team plan to investigate the adding of this functionality.

The future of FTS and GFAL2 is to scale based on experiment demands in order to support LHC Run-4.

References

[1] File Transfer Service, https://fts.web.cern.ch/fts/

[2] Python2 End-of-Life, Sunsetting Python 2, https://www.python.org/doc/sunset-python-2/
[3] FTS project, FTS YUM repository, https://fts-repo.web.cern.ch/fts-repo/el7/x86_64
[4] Extra Packages for Enterprise Linux (EPEL) repository, https://docs.fedoraproject.org/en-US/epel
[5] Pylons web framework, https://pylonsproject.org/about-pylons-framework.html
[6] Pyramid web framework, https://trypyramid.com
[7] Flask web framework, https://flask.palletsprojects.com
[8] SQLAlchemy The Python SQL Toolkit and Object Relational Mapper, https://www.sqlalchemy.org/
[9] CERN-IT Database-on-Demand Service, https://cern.service-now.com/service-portal?id=service_element&name=database-on-demand
[10] Storage Resource Management (SRM) Working Group, https://sdm.lbl.gov/srm-wg
[11] Worldwide LHC Computing Grid, https://wlcg.web.cern.ch
[12] EOS Open Storage, https://eos-web.web.cern.ch/eos-web/
[13] EOSCTA Docs, https://eoscta.docs.cern.ch/
[14] dCache.org distributed storage for scientific data, https://www.dcache.org/
[15] StoRM (STOrage Resource Manager), https://italiangrid.github.io/storm/index.html
[16] Grid File Access Library, https://dmc-docs.web.cern.ch/dmc-docs/gfal2/gfal2.html
[17] WLCG Tape REST API, https://github.com/wlcg-storage/wlcg-tape-rest-api
[18] WLCG Authorization Working Group, https://twiki.cern.ch/twiki/bin/view/LCG/WLCGAuthorizationWG
[19] Rucio Scientific Data Management, https://rucio.cern.ch
[20] DIRAC The Interware, https://dirac.readthedocs.io/en/latest/index.html