FTS3 – a file transfer service for Grids, HPCs and Clouds

Andrey Kiryanov[1]
PNPI, NRC KI, CERN
Gatchina, Russia
E-mail: Andrey.Kiryanov@cern.ch

Alejandro Alvarez Ayllon
CERN
Geneva, Switzerland
E-mail: Alejandro.Alvarez.Ayllon@cern.ch

Michail Salichos
CERN
Geneva, Switzerland
E-mail: Michail.Salichos@cern.ch

Oliver Keeble
CERN
Geneva, Switzerland
E-mail: Oliver.Keeble@cern.ch

FTS3, the service responsible for globally distributing the majority of the LHC data across the WLCG infrastructure, is now available to everybody. Already integrated into LHC experiment frameworks, a new Web interface now makes the FTS3's transfer technology directly available to end users. In this contribution we describe this intuitive new interface, "WebFTS", which allows users to easily schedule and manage large data transfers right from the browser, profiting from a service which has been proven at the scale of petabytes per month. We will shed light on new development activities to extend FTS3 transfers capabilities outside Grid boundaries with support of non-Grid endpoints like Dropbox and S3. We also describe the latest changes integrated into the transfer engine itself, such as new data management operations like deletions and staging files from archive, all of which may be accessed through our standards-compliant REST API. For the Service Manager, we explain such features as the service's horizontal scalability, advanced monitoring and its "zero configuration" approach to deployment made possible by specialised transfer optimisation logic. For the Data Manager, we will present new tools for management of FTS3 transfer parameters like limits for bandwidth and max active file transfers per endpoint and VO, user and endpoint banning and powerful command line tools. We finish by describing the impact of extending WebFTS's captivating graphical interface with support of Federated Identity technologies, thus demonstrating the use of Grid resources without the burden of X.509 certificate management. In this manner we show how FTS3 can cover the needs of wide range of parties from casual users to high-load services.

[1]Speaker

The evolution of FTS3 is addressing technical and performance requirements and challenges for LHC Run 2, moreover, its simplicity, generic design, web portal and REST interface makes it an ideal file transfer scheduler both inside and outside of HEP community.

International Symposium on Grids and Clouds (ISGC) 2015
15 -20 March 2015,
Academia Sinica, Taipei, Taiwan

1. Introduction

FTS3 (File Transfer Service, version 3.0) is one of the projects of critical importance for data management at CERN [1]. It has been designed in a modular and extensible way to allow good scalability.

Core functionality of FTS3 is extended with various Web-oriented tools like versatile monitoring and WebFTS user interface with support of Federated Identity (IdF).

2. System architecture

The components of the FTS (shown in Fig. 1) are: command-line clients, a daemon process responsible for transfer submission, status retrieval and general VO (Virtual Organization) and service configuration, another daemon process responsible for bulk deletion and stage-in of files from archive using the SRM [2] protocol (BringOnline operation), and, finally, the database back-end. The service scales horizontally very well by adding more resources with identical configuration into an FTS3 cluster, since the configuration is stored in the database and is being read during start-up of the service. A single configuration file is used only to store the database credentials.

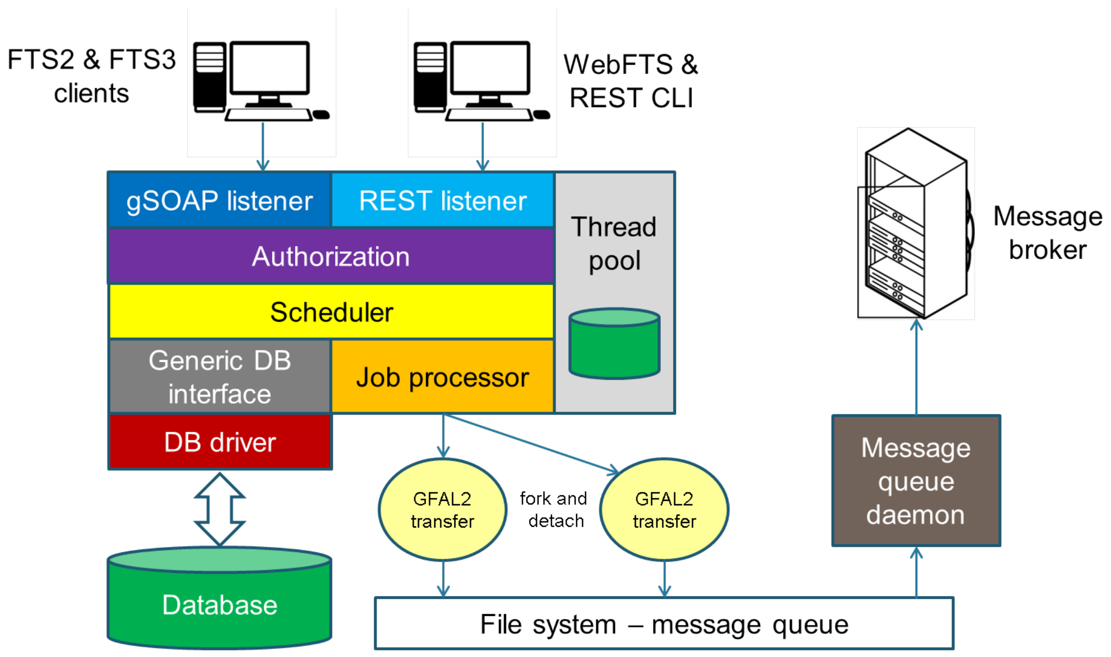

Fig. 1. FTS3 architecture

3. Main features

FTS3 offers features and functionality that have been requested by the experiments and sites following their initial usage of FTS2 [3, 4] and the needs that the new FTS3 service should address. Most of these have already been implemented or are in the process of being finalized. For example:

- transfer auto-tuning / adaptive optimization;
- endpoint-centric VO configuration;
- transfer multi-hop;
- VO activity shares;
- multiple replica support;
- REST-style interface for transfer submission and status retrieval;
- retry failed transfers mechanism;
- staging files from archive;
- bulk deletions;
- support for Oracle and MySQL database back-ends;
- transfer and access protocols support on top of GFAL2 plug-in mechanism (SRM, GridFTP [5], HTTP, xroot);
- session / connection reuse (GridFTP, SSL, etc), which is ideal for many small file transfer jobs.

In order to be a credible long-term platform for data transfer, FTS3 has been designed to exploit upcoming developments in networking, such as integrating monitoring data from perfSONAR for further transfer optimization, resource management and monitoring of network state.

3.1. Adaptive optimization

Adaptive optimization is the mechanism introduced in FTS3 to address the shortcomings of its predecessor – FTS2. FTS2 required file transfers to be performed on channels which must be defined and configured explicitly. The FTS3 auto-tuning algorithm and channel-less transfer model allows moving from a predefined structured topology to a full-mesh (as shown in Fig. 2) for network links and storage elements without any configuration and, at the same time, optimizing transfers based on information stored in the database.

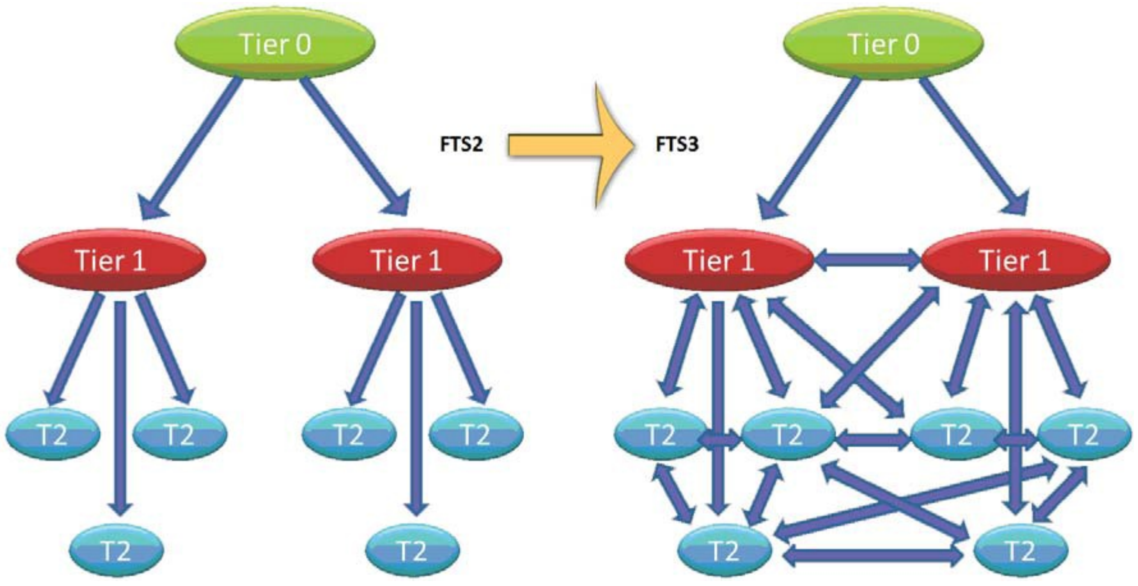

Fig. 2. move from hierarchical topology to full mesh using FTS3

The plot in Fig. 3 demonstrates how the adaptive optimization algorithm is influenced by the achieved throughput and success rate of distinct links, and dynamically adjusts the number of concurrent file transfers based on this information. A sample is taken every 30 seconds and the decisions of the optimization are stored into the database for persistence and service monitoring.

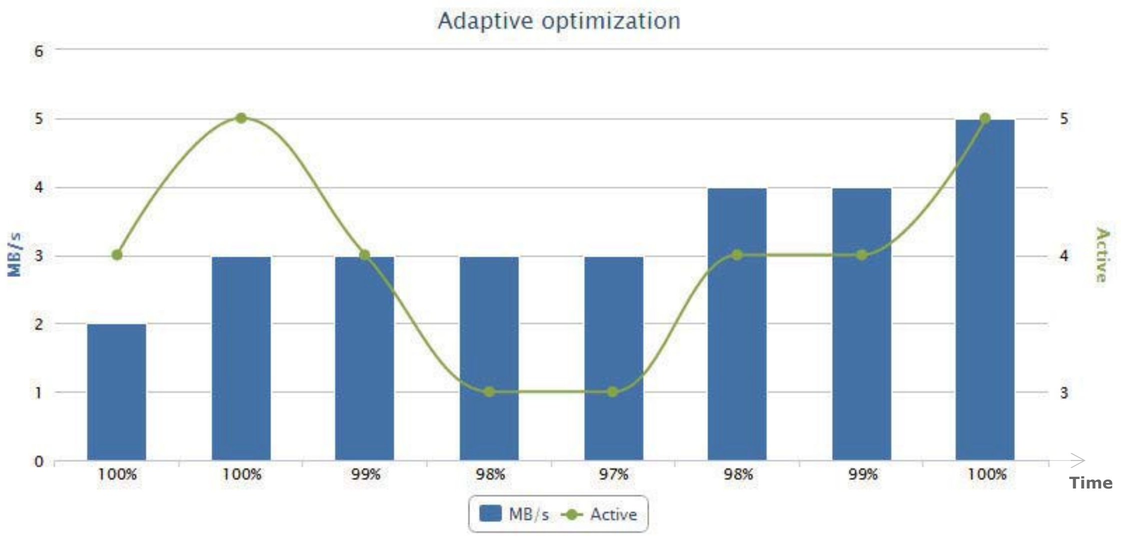

Fig. 3. Adaptive optimization

3.2. Managing resources

In the FTS3 world resource management is Storage Element (SE) centric and it can be done at three different levels: sharing the resources between Virtual Organizations, specifying the number of concurrent transfer jobs and defining custom protocol parameters (TCP buffer size, number of open sockets per transfer job, etc.). The motivation for the SE-centricity was the change in computing model from a strict hierarchy to a mesh [6]. While in the case of a channel based policy the number of configurations required for full-mesh connectivity grows quadratically with the number of SEs, in case of a SE-centric policy the growth is linear (for example, a set-up of 6 SEs requires either 15 channel configurations or 6 SE-centric configurations). There are four types of configurations available in FTS3:

- The standalone SE configuration type has been created so the user can address the problem of overloaded SEs by limiting the total number of outgoing and incoming transfers. Those limits can be imposed per-VO in order to create so called 'VO-shares'. While transfer jobs are carried out between two SEs with standalone configurations, the limits for both the source and the destination are respected. On the other hand, their protocol configurations are merged resulting in consistent settings for the given pair (e. g. FTS3 will pick the smallest time-out value).
- The SE pair configuration allows configuring explicitly the link between two SEs. This type of configuration should be used only in case when standalone configurations do not apply to this particular pair (e. g. the destination SE is in an unusual location and requires much bigger time-out than other SEs). The SE

pair configuration overrides all the settings from the respective standalone configurations.
- The SE group configuration is meant to address a particular use case when a group of SEs shares a common resource (e. g. a common network link) that may limit their aggregated performance. Hence, the SE group configuration allows imposing a limit on the total number of incoming and outgoing transfer jobs for the whole group of SEs. If an SE has a standalone configuration and is also a member of a group, both limits are respected: the one imposed by the standalone configuration and the one imposed by the group configuration.
- The SE group pair configuration (analogous to the SE pair configuration) allows to override the group configuration settings for a pair of two particular SE groups.

Each of the above-mentioned configurations can be hybridized with the auto-tuner mechanism through replacing any configured value with the 'auto' keyword.

3.3. REST interface

FTS3 provides a REST [7] interface to submit transfers and query their state. This interface is to a great extent self-discoverable, as it provides information about how to reach each resource, and which sort of interaction they support [8]. We have tried to honour the restrictions imposed by some authors to consider a RESTful interface implementation a proper RESTful interface [9]:

- Each resource has its own URI:
  Collection of jobs: https://fts3.cern.ch:8446/jobs/
  Single given job: https://fts3.cern.ch:8446/jobs/083ac623-5649-4127-1234-abcdef123456
- Meaningful use of HTTP verbs
  GET <job-URI> retrieves information about the given job
  DELETE <job-URI> cancels a job
  POST <job-collection-URI> submits a new job
- Hypermedia. Although this third point is not complete – not every resource provides information about the actions that can be performed – we do provide self-discovery information on the root level of our API using the JSON Hypertext Application Language [10].

Even though the default deployment configuration is to have all interfaces running on all the FTS3 machines, the FTS3 REST interface can easily be deployed separately.

It is worth noting that we also provide the JSON Schema of the expected JSON messages sent for submission. Using this schema, a client application can easily validate its messages before sending them to the server to check compliance with the expected format.

3.4. Protocol support

FTS3 relies on the GFAL2 [11] library, which provides an abstraction layer over the complex Grid storage systems. Support for several protocols is provided by a collection of plug-ins. This plug-ins based system has three big advantages:

1. Independence: The client application – FTS3 in our this case – can be written independently of the protocol, or set of protocols, that will be used. Thanks to this we can avoid pulling dependencies for unneeded protocols.
2. Isolation: The protocol-specific logic is contained inside the corresponding plug-in, so the addition of new functionality, fixes or any other changes can be done without risk of affecting the behaviour of the other protocols.
3. Extensibility: Adding a new protocol is just a matter of writing or installing a new plug-in.

As mentioned, FTS3 in principle can support any protocol that GFAL2 supports. Currently, these are DCAP, GridFTP, HTTP(S)/WebDAV, RFIO, SRM and XROOTD. It is worth mentioning that, from this list, only GridFTP and SRM were supported by the previous version, FTS2.

4. IdF-enabled Web interface

In the attempt of making all the features of FTS3 easily accessible to the end users a new standalone Web-oriented interface was developed: WebFTS. With its point-and-click commander-like two-panel interface individual users can submit and monitor their own transfer jobs. WebFTS provides the same multi-protocol support as FTS3 which makes it a very handy tool for transferring files between Grid and non-Grid resources.

In the first version of WebFTS it was necessary to manually delegate user's X.509 credentials to the service by providing unencrypted copy of the private key. There was no security compromise in this, because private key was only used inside browser's JavaScript context to sign delegation request and was never made available to any external service including FTS3 itself. However this operation was tricky and required access to the command line OpenSSL tool.

As an ultimate solution to X.509 burden in the view of accessing Grid resources with Web browser it was decided to add Federated Identity (IdF) support to WebFTS which would allow transparent transition from X.509-based credentials to Web-based ones (SSO) with help of two additional services: STS [12] and IOTA CA [13].

- STS (Security Token Service) is a service that consumes SAML2 assertion produced by IdF-enabled SSO and transforms it into a short-living X.509 certificate.
- IOTA CA (Identifier-Only Trust Assurance Certification Authority) is a CA with specific profile that is eligible of issuing short-living X.509 certificates that are necessary for STS.

All the security machinery that is happening behind the scenes and transparently to the user is shown on Fig. 4:

1. When user opens WebFTS page a standard SSO login link is shown ("LogIn" or "LogOut <user name>" depending on the state).
2. User logs in to IdF-enabled SSO and SAML2 assertion is made available to the server hosting WebFTS.
3. WebFTS JavaScript application in user's browser fetches SAML2 assertion from the server, generates a key pair, and forwards public key along with the assertion to STS.
4. STS uses IOTA CA to issue a certificate and sends signed certificate back to WebFTS JavaScript application. VOMS extensions may be added to the certificate if requested by the user.
5. WebFTS JavaScript application uses certificate and private key to delegate user's credentials to FTS3 server via REST API.

It's important to mention that all the sensitive information like private keys never leaves user's computer and all the cryptographic operations necessary for delegation are performed in the browser context by JavaScript code.

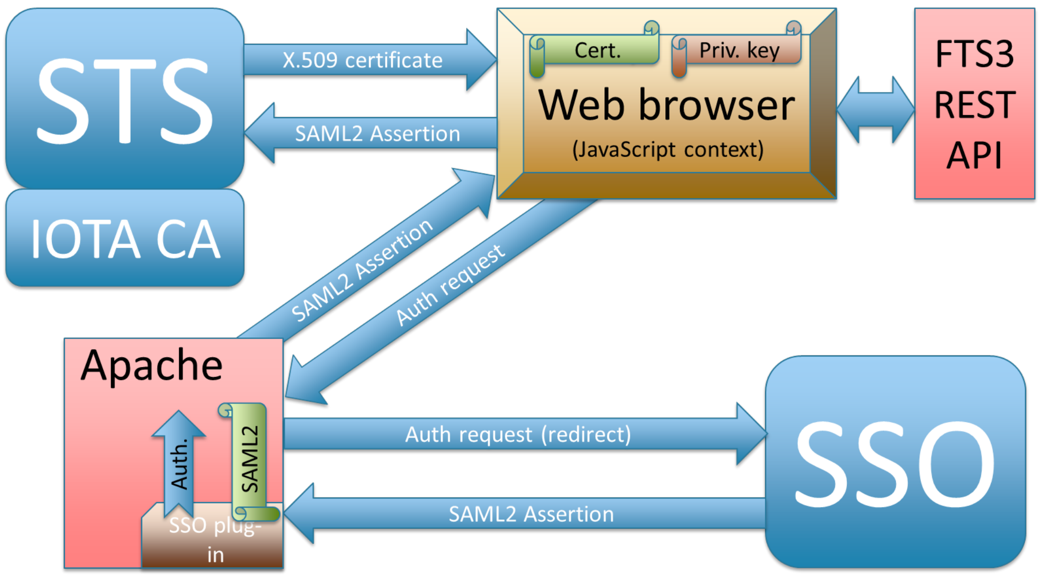

Fig. 4. IdF-enabled WebFTS

IdF integration allows to make WebFTS completely X.509-free from the user's point of view. With Identity Federations like eduGAIN it's now possible to expose FTS3 powers to user communities that were never accustomed to X.509 infrastructure necessary for the Grid.

One of the ramaining problems though is that in order to have a chance of accessing Grid storage endpoints with X.509 credentials received from STS such endpoints need to trust IOTA-profile CAs. This involves administrative work of convincing resource owners into making their resources available to IdF users.

5. HPC integration

LHC experiments show more and more interest in using available HPC resources in addition to the usual Grid ones. The difference here is that HPCs usually have unique architecture that does not match one of a normal Grid site from WLCG. In order to submit tasks to HPCs one have to use a software that was designed to work specifically with a given HPC.

Arisen from the LHC ATLAS experiment workload management system software called PanDA, which eventually turned into not-ATLAS-specific Big PanDA, have shown to be very effective in handling millions of jobs, and also acted as a bridge for accessing supercomputer resources in Oak Ridge LCF.

One thing that was missing from PanDA is a data management subsystem, which became a bottleneck with HPCs where input and output data for thousands of concurrently running jobs have to be effectively staged in and out of the internal file system. Doing these transfers from inside HPC have shown to be ineffective because of wasted CPU time that could have been used for computation if needed data were pre-staged and made available before the job start, not to mention that computing nodes of many HPCs simply do not have usual TCP/IP network connectivity and can only access data on shared internal file system.

There's a work in progress shown on Fig. 5 of integrating FTS3 with PanDA and making FTS3 capable of relaying file transfers between non-Grid HPC resources and standard Grid storage endpoints.

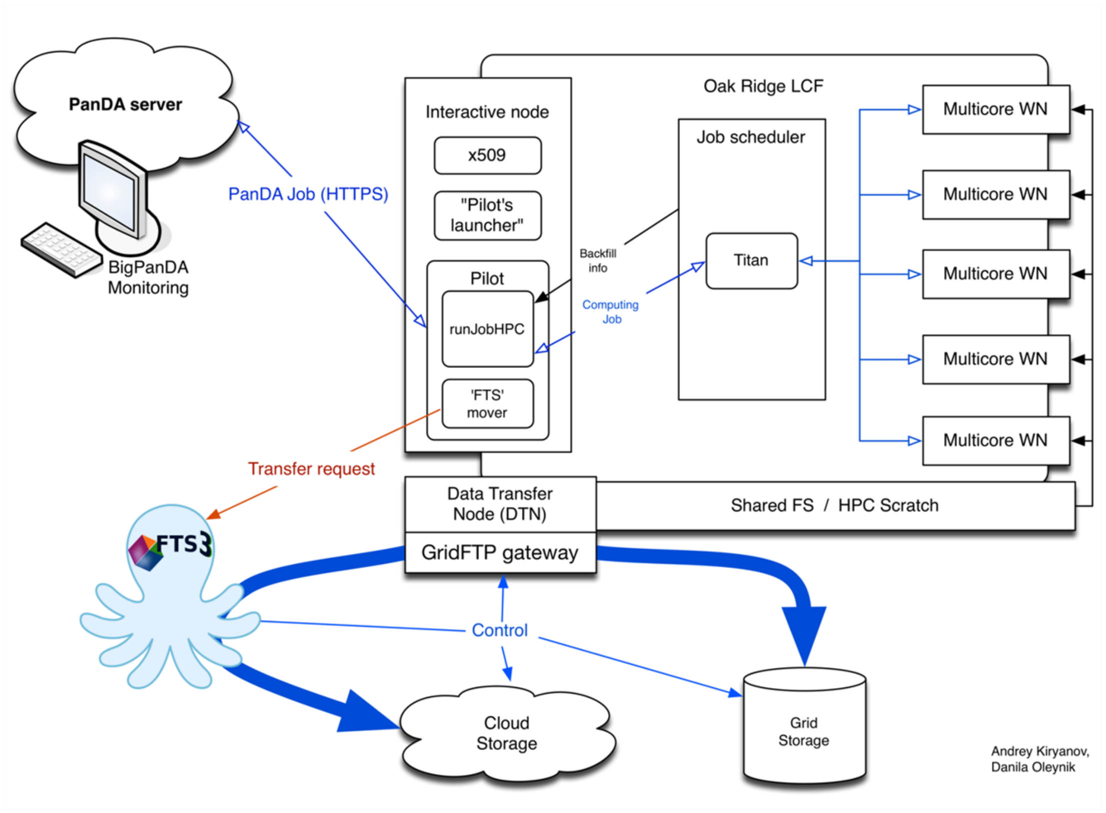

Fig. 5. FTS3 and HPC

6. Conclusions

The service has already undergone extensive pre-production validation and we have demonstrated the results of high volume production transfers performed on the pilot service and production-ready instances.

The plot in Fig. 6 shows the usage of FTS3 instance at GridPP, used by ATLAS and CMS for production and debug transfers. It is worth mentioning that the volume shown below (first bar = 370TB), was achieved using zero configuration (adaptive optimization) on top of MySQL database running on a VM.

Anticipating the upcoming data movement needs of WLCG, and building on the lessons learned during the first run of LHC, we present a new, scalable and highly-optimized data movement service, which provides a simple interface for transfer job submission, status retrieval, advanced monitoring capabilities, multiple access and transfer protocols support and simplified configuration.

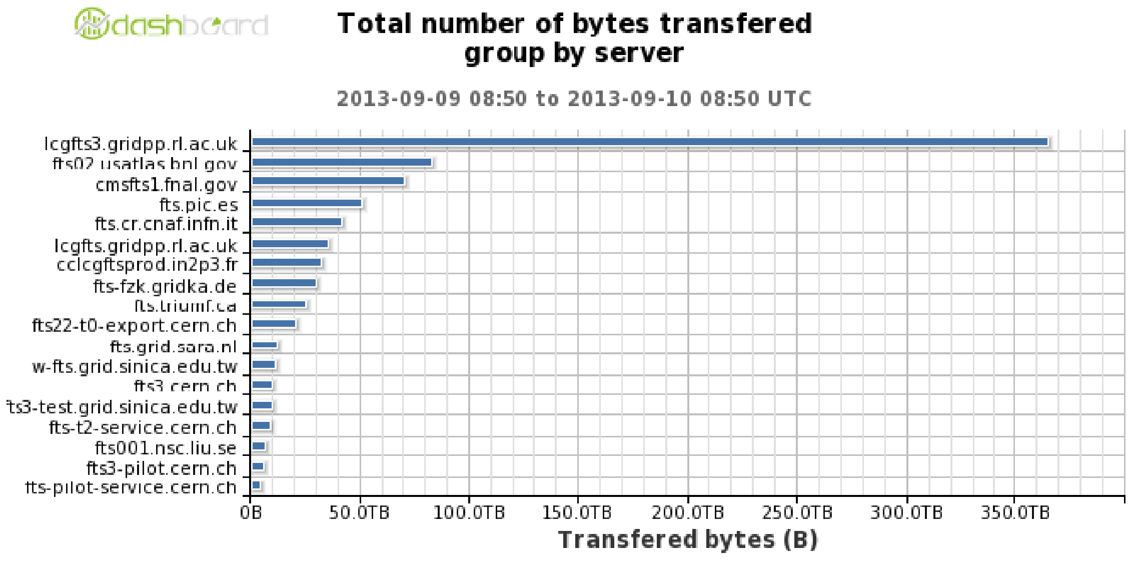

Fig. 6. FTS3 usage at GridPP

7. Acknowledgements

Andres Abad Rodriguez
Martin Hellmich
Andrea Manzi
Markus Schulz
Michal Simon
Christina Skarpathiotaki

This work was funded in part by the Russian Ministry of Education and Science under contract №14.Z50.31.0024

References

[1] Critical services in the LHC computing, A Sciaba 2010 *J. Phys.: Conf. Ser.* 219
[2] Abadie L et al 2007 24th IEEE Conference on Mass Storage Systems and Technologies pp.47,59, 24-27
[3] Data management in EGEE, A Frohner et al 2010 J. Phys.: Conf. Ser. 219

[4] Abadie L et al 2007 24th IEEE Conference on Mass Storage Systems and Technologies pp.60,71, 24-27
[5] Allcock W et al 2005 ACM/IEEE Supercomputing pp.54,54, 12-18
[6] The evolving role of Tier2s in ATLAS with the new Computing and Data Distribution model, S Gonzalez de la Hoz 2012 J. Phys.: Conf. Ser. 396
[7] Architectural Styles and the Design of Network-based Software Architectures, Roy T. Fielding Dissertation, 2000
[8] Principled Design of the Modern Web Architecture, Roy T. Fielding and Richard N. Taylor, ACM Transactions on Internet Technology, Vol. 2, No.2, May 2002, Pages 115 - 150
[9] Justice Will Take Us Millions Of Intricate Moves, Leonard Richardson, talk at the International Software Development Conference (QCon), 2008
[10] JSON Hypertext Application Language (Draft), M. Kelly, IETF Draft, October 2013
[11] GFAL 2.0, Adrien Devresse, Presentation at the GDB, November 2012
[12] Security Token Service, Henri Mikkonen, Presentation at EGI Technical Forum, September 2012
[13] https://www.igtf.net/ap/iota/