


FTS3

White Area Lecture

Michail Salichos
IT/SDC

04/12/2013

Overview

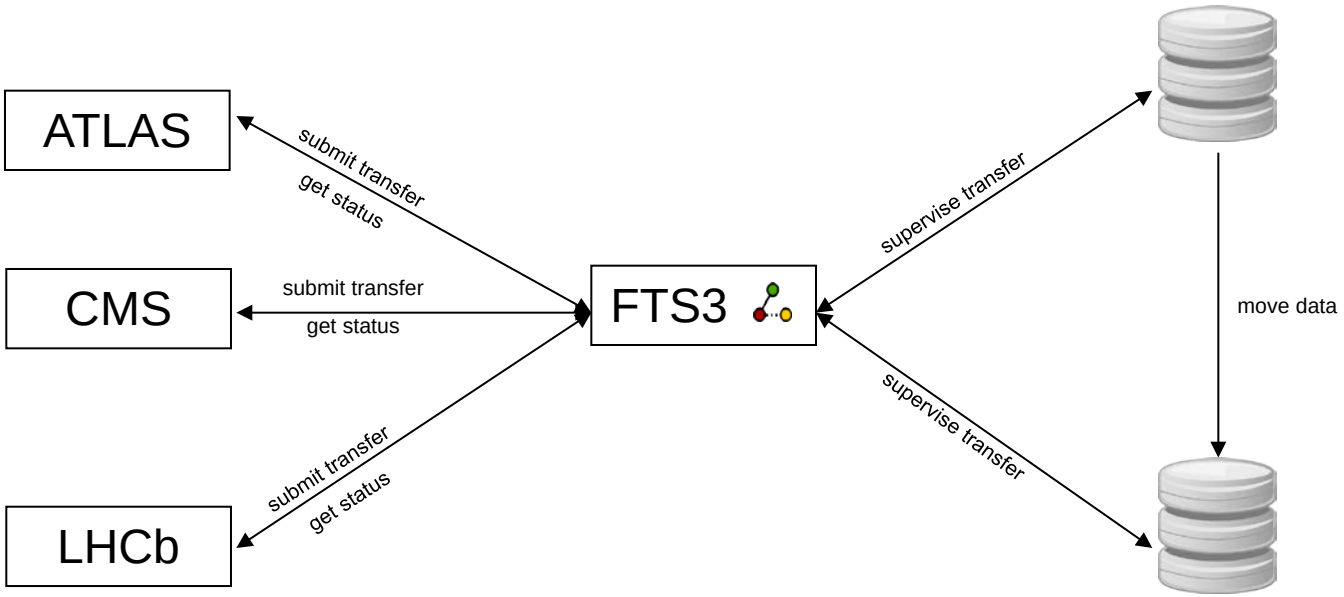
- Background
- Agile development
- Features
- Status
- Road-map
- Summary

What is FTS?

- The service responsible for distributing the majority of LHC data across WLCG infrastructure
  - mature service - running for almost 10 years
- Low level data movement service, responsible for moving sets of files from one site to another while allowing participating sites to control the network resource usage

How does it work?

- Users interact with FTS by submitting transfer jobs, that simply say "copy <source URL> to <destination URL>
  - FTS then queues, schedules and performs the transfer, retrying it if necessary



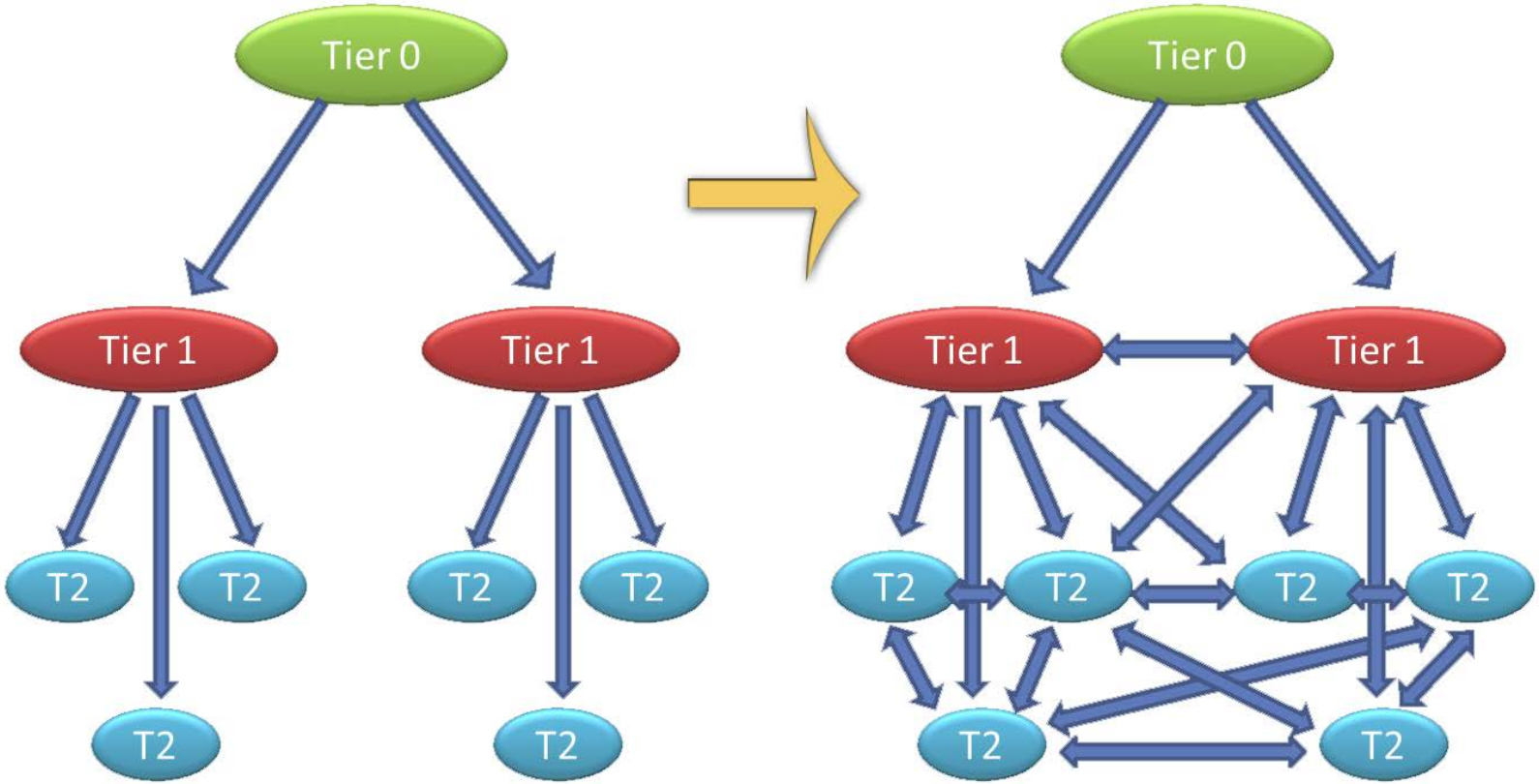
Motivation behind FTS3

- address a particular set of FTS2 shortcomings, e.g.
  - relax the requirement to configure channels
  - protocols support
  - code maintenance issues
- simple to install and configure
- easy to maintain and support
- light-weight and not “resource hungry”
- support transferring large volume of data
- scale well horizontally
- control and efficiently use resources (network, SEs)

FTS3 goal

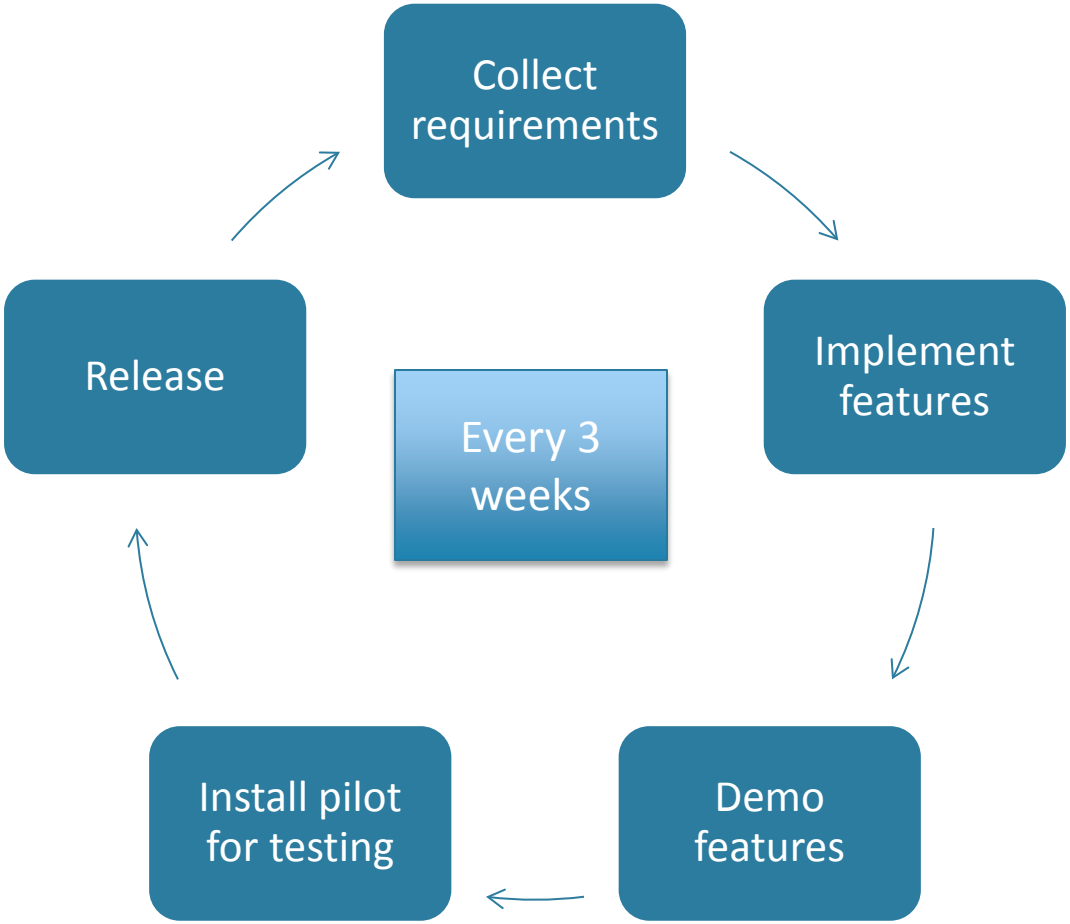
from hierarchical topology…

to a full mesh…



... and move data at a very large scale

Agile development



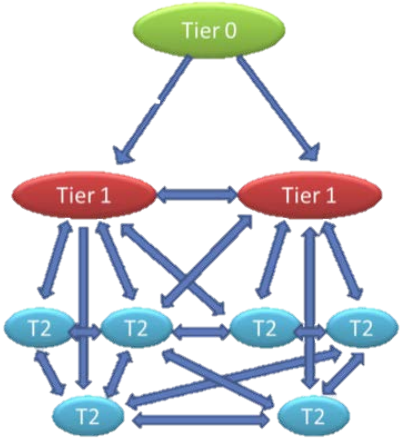
FTS highlights

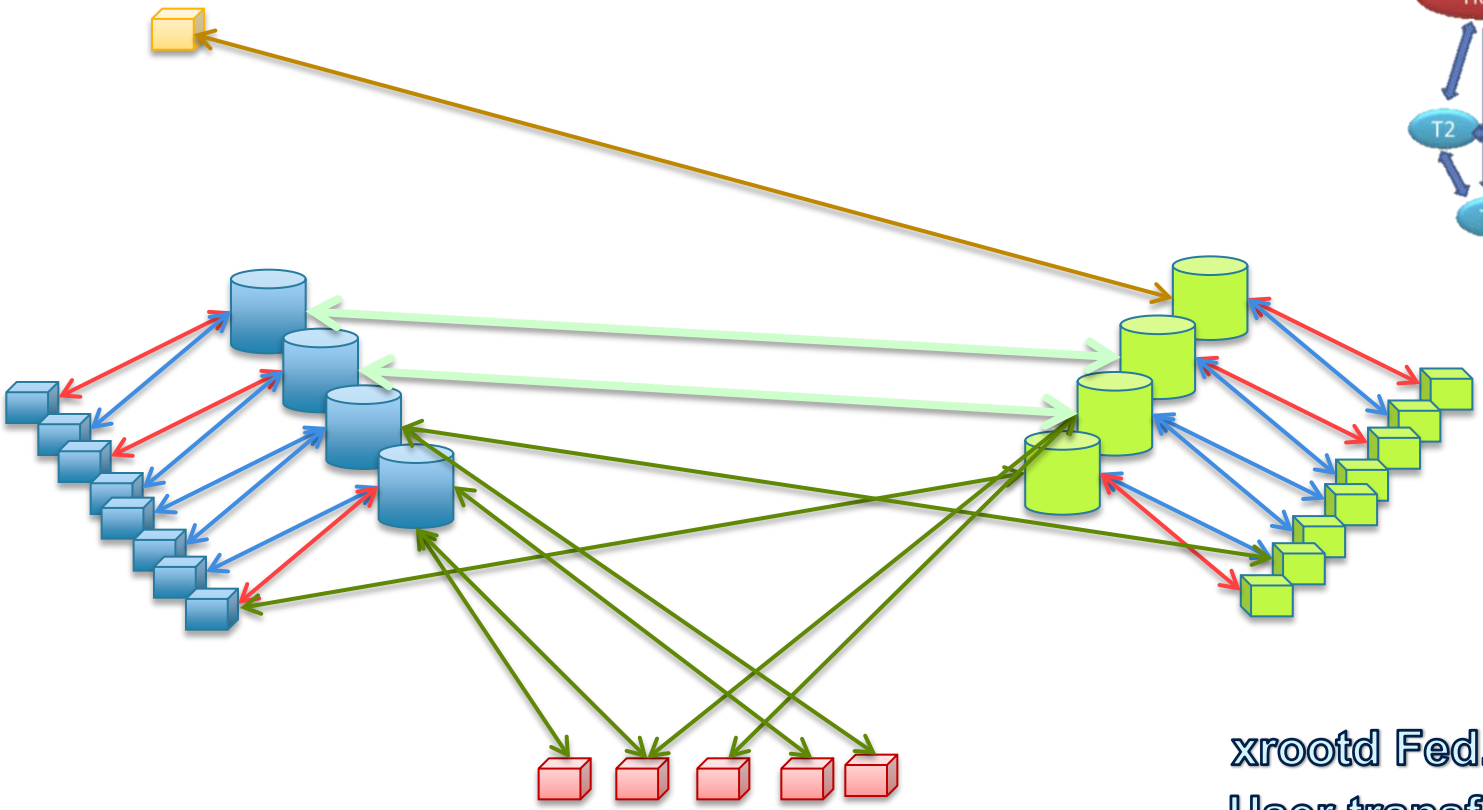
FTS2	FTS3
Mature service, ~10 years in prod	➢ 1 year as a Pilot ➢ 6 months ATLAS+ LHCb prod
Channel model – site pair	Endpoint-centric Adaptive optimization – zero config
SRM, gsiftp	SRM, gsiftp, HTTP, xroot
Oracle	Oracle, MySql (SQLite or PostgreSQL easily)
No horizontal scalability	Scales well horizontally

What effects transfers?

- For every transfer three entities and their state play a role
  - Source storage system
  - Network
  - Destination storage system
- All three are effected by multiple activities

Activities...





xrootd Fed.

User transfers

Scheduled

How to access the service

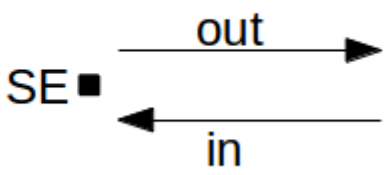
- Clients and interfaces
  - FTS2 clients compatibility
  - FTS3 clients with many new features
  - RESTful API for standard clients using JSON
  - Python bindings for custom clients
- FTS3 clients new features + python bindings
  - Job and file metadata
  - Session reuse, ideal for many small file size transfer jobs
  - JSON formatted output
  - Retry mechanism at the job level
  - Specify file size for validation with the surl
  - Transfer multi-hop

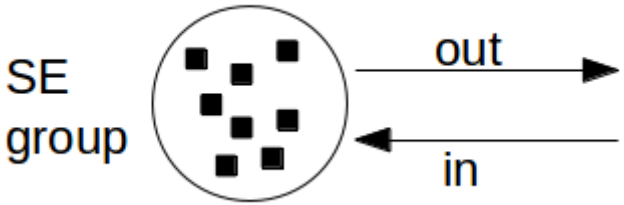
Resource optimization

- Adaptive optimization – let FTS3 decide
- Session reuse
  - GridFTP channel caching
  - SRM KeepAlive
  - HTTP SSL context reuse
- Multiple replicas support
- Smart transfer retry mechanism

Resource management

- Protocols support
  - GridFTP, SRM, HTTP, xroot
  - On top of GFAL2 – provides protocol plug-ins
- Blacklisting users (DN) and SEs
- Endpoint-centric configuration

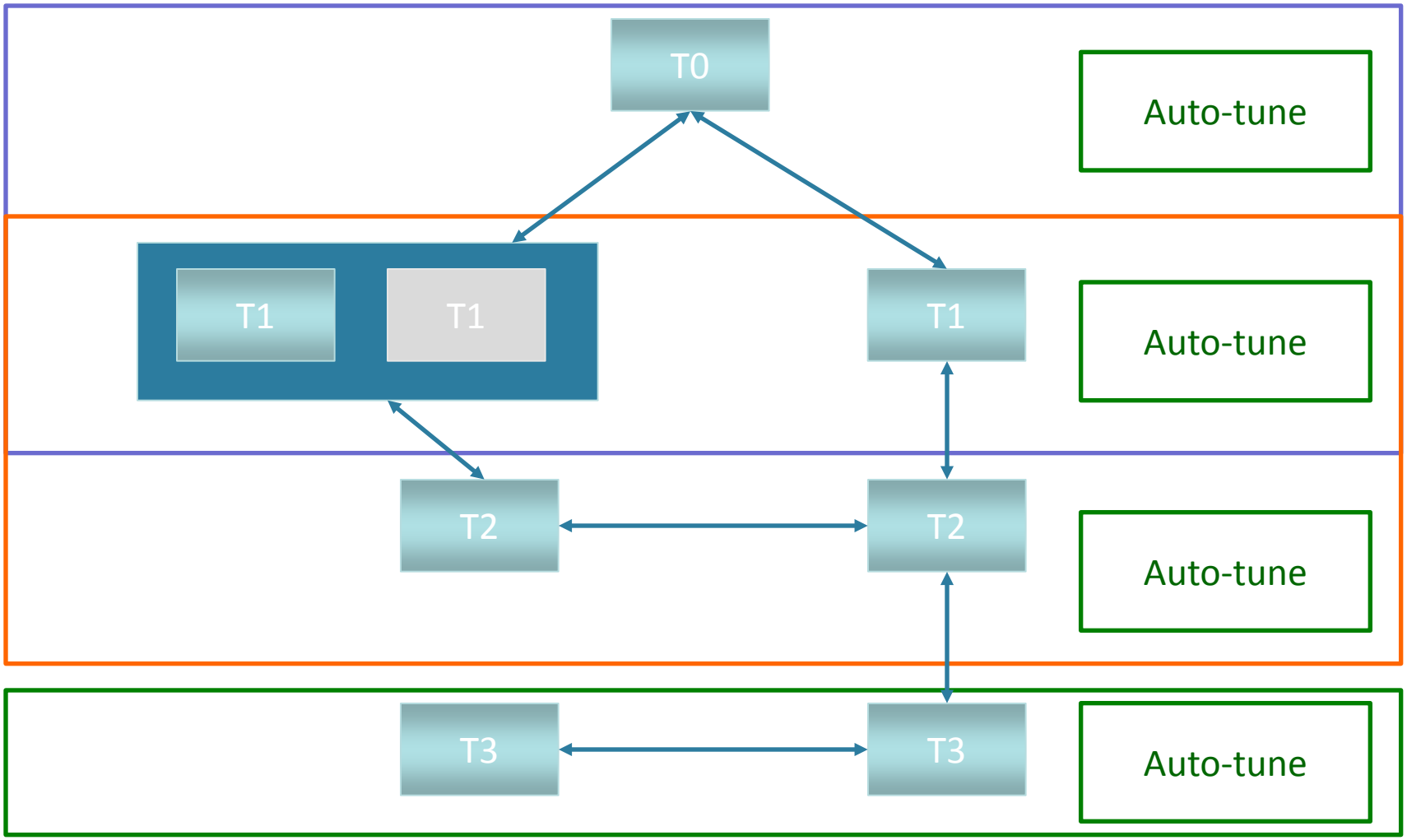




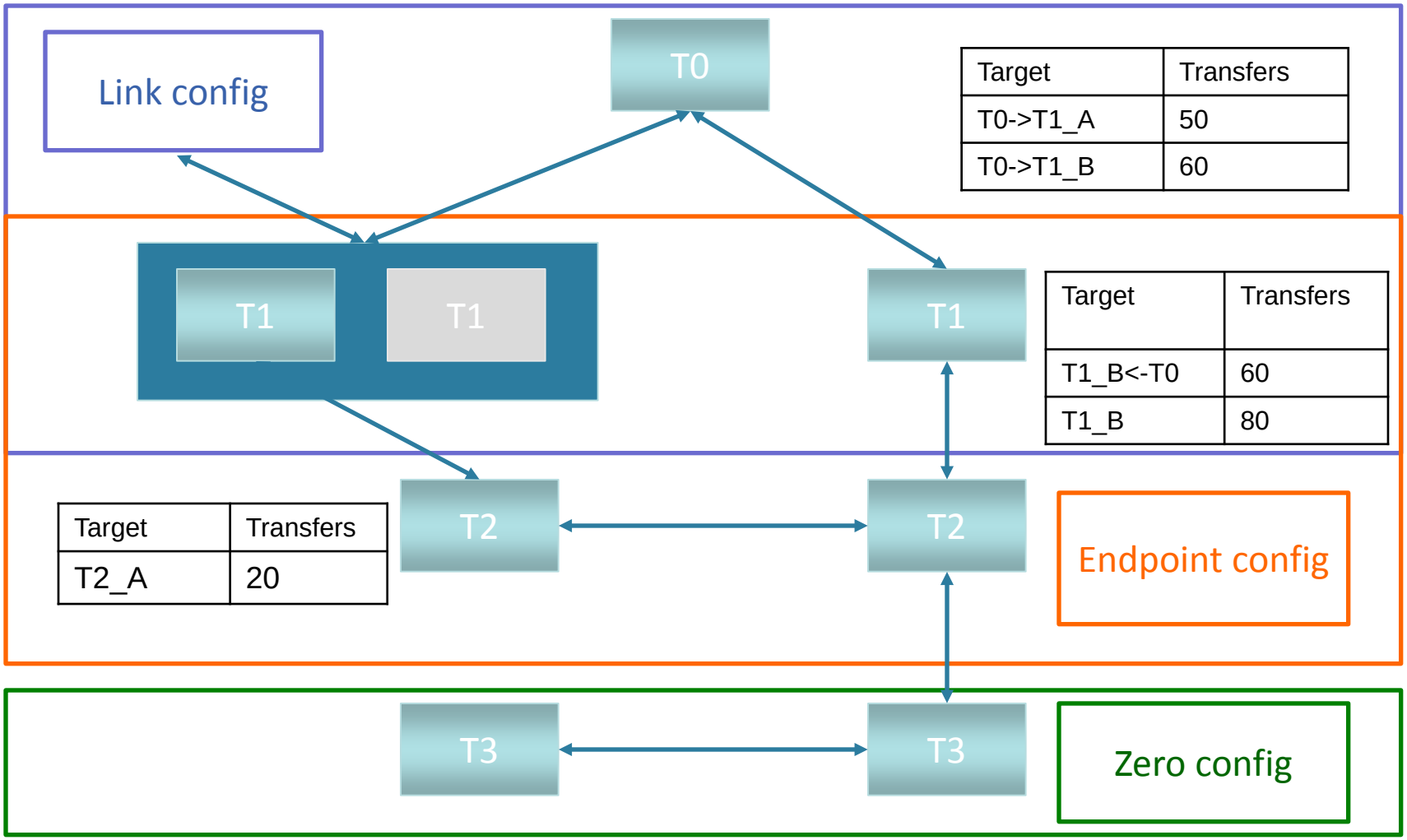
Deployment

- Horizontal scalability
- Minimal initial configuration
  - Mostly stored into the database
- DB backend support
  - MySql, Oracle
  - SQLite and/or PostgreSQL if requested

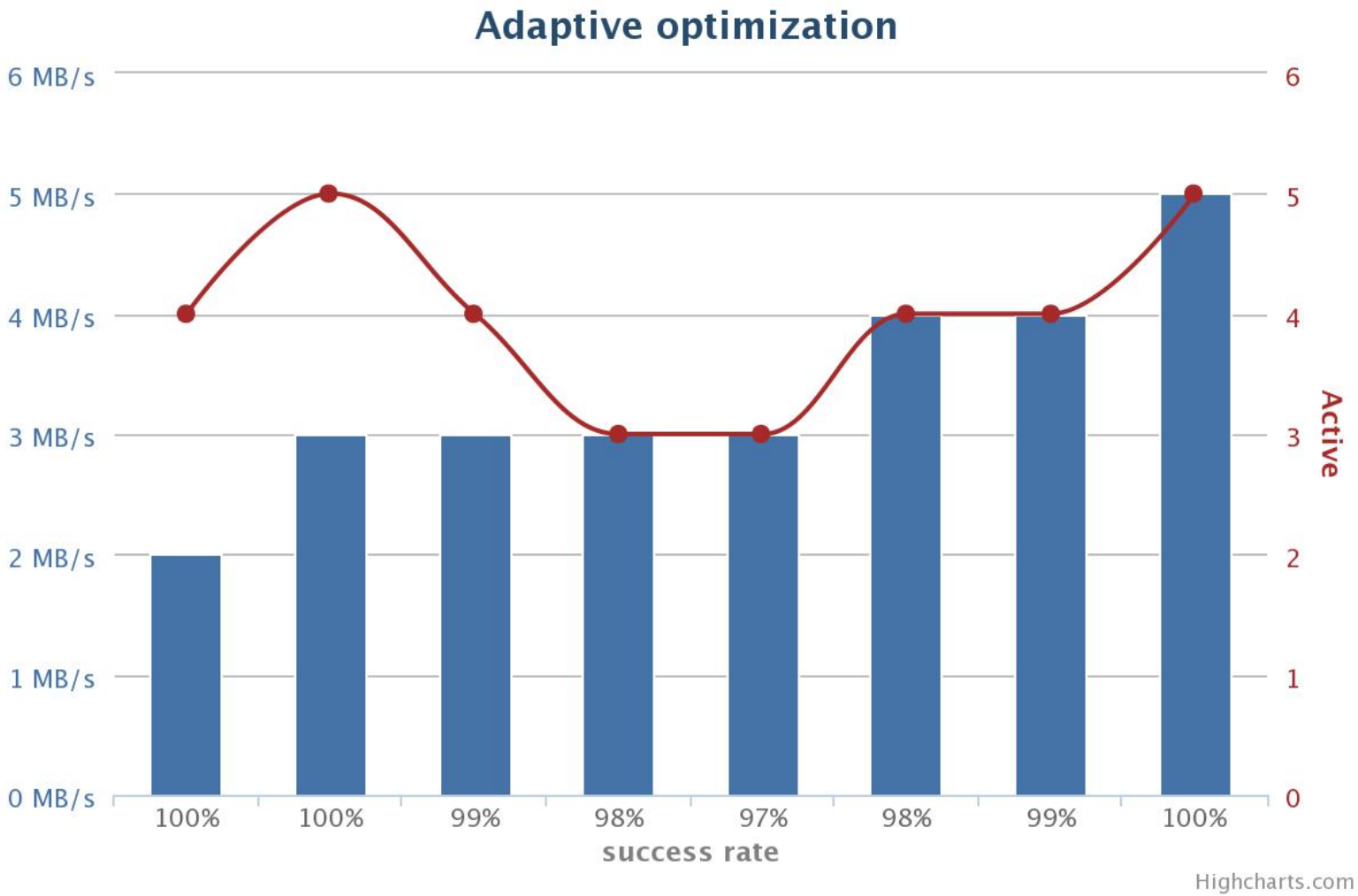
Configuration model



Configuration model (2)



Core features

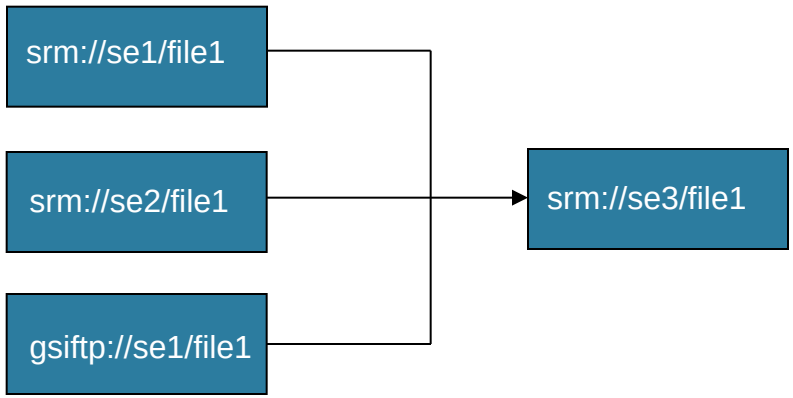


Core features (2)

- RESTful API
  - clients installation not needed
    - standard clients and/or libraries can be used
      - [lib]Curl, Python's urllib2...
  - well defined JSON schema
  - https://svnweb.cern.ch/trac/fts3/wiki/FTSRest

Core features (3)

- Multiple replicas support
  - Modes
    - Automatic – let FTS3 decide the order of replicas based on historical information
    - Manual - respect order set



Core features (4)

- Retry failed transfers
  - per individual job or globally set
  - failures classified as recoverable or not
  - Non-recoverable
    - No such file or directory
    - No space left on device
    - Permission denied
    - Read-only file system
    - etc
  - Recoverable – to be retried
    - All the rest
- More information in FTS3 wiki page

Monitoring

- WLCG Dashboard transfers UI
  - Developed by CERN Dashboard team
  - A single entry point to the monitoring data collected from the distributed systems of the LHC
  - Monitor multiple FTS3 instances
  - each FTS3 server publishes messages to a message bus to report transfer status and state transitions
- Web interface for individual FTS3 server monitoring
  - In-depth details about job information, queued jobs, audit-trails, etc
- Nagios probes

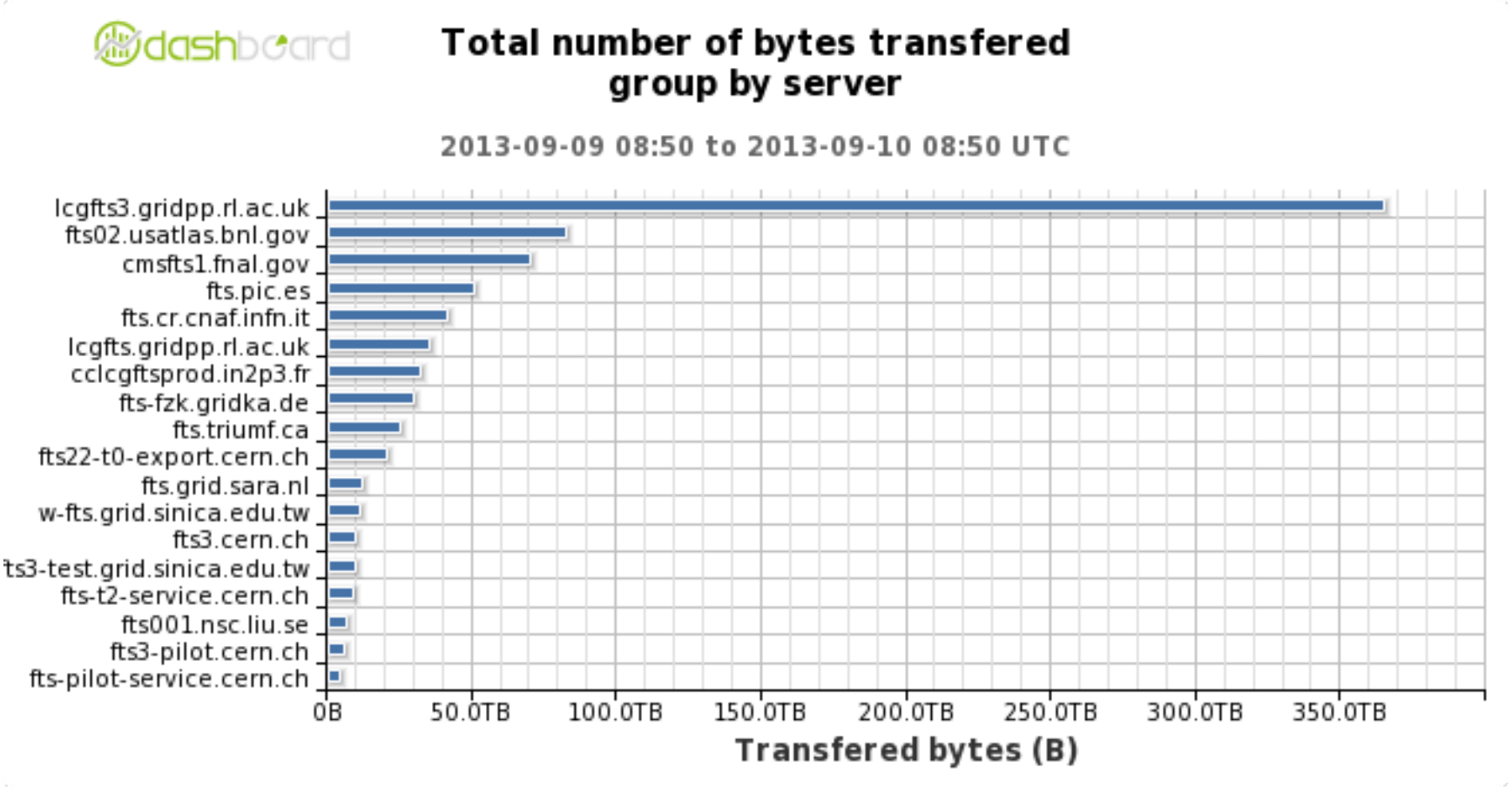
Releases

- Available in
  - EPEL 6 (fts-*)
  - our continuous integration repository (stable)
- Platform supported
  - SL6 / 64 bit
  - Clients support on diff platforms (if requested)

Testing and evaluation

- Installed at CERN, RAL, PIC, KIT, ASGC, BNL, IN2P3 and PNL
  - production and testing
  - > 1 year as a Pilot service
- Heavily used by ATLAS for prod jobs
  - avg weekly transfer volume from RAL ~1.5PB
- LHCb: all WAN transfers are executed by FTS3 (using the CERN instance with RAL as backup)
- Tested by
  - LHC experiments
  - EGI/EUDAT against globus GridFTP, dCache GridFTP and GridFTP interface for iRODS (Griffin)
  - many other VOs already tested and using it successfully: snoplus.snolab.ca, ams02.cern.ch, vo.paus.pic.es, magic, T2K, NA62, etc
- Planned to run a “service challenge”

Sample volume



Roadmap

- Entirely determined by experiment requirements and prioritization
- What's next
  - Global scheduling and shared VO configuration across distributed FTS3 servers
  - Integration and testing of perfSonar information (bandwidth & ping tests) for transfer optimization
  - deeper integration with archival storage and include high performance file management capabilities (deletes, renames...)
  - Web-app for transfer submission and status retrieval
  - Keeping an eye on bandwidth reservation evolution
  - Embedded / light-weight FTS3 version on top of SQLite

Users - Usage View

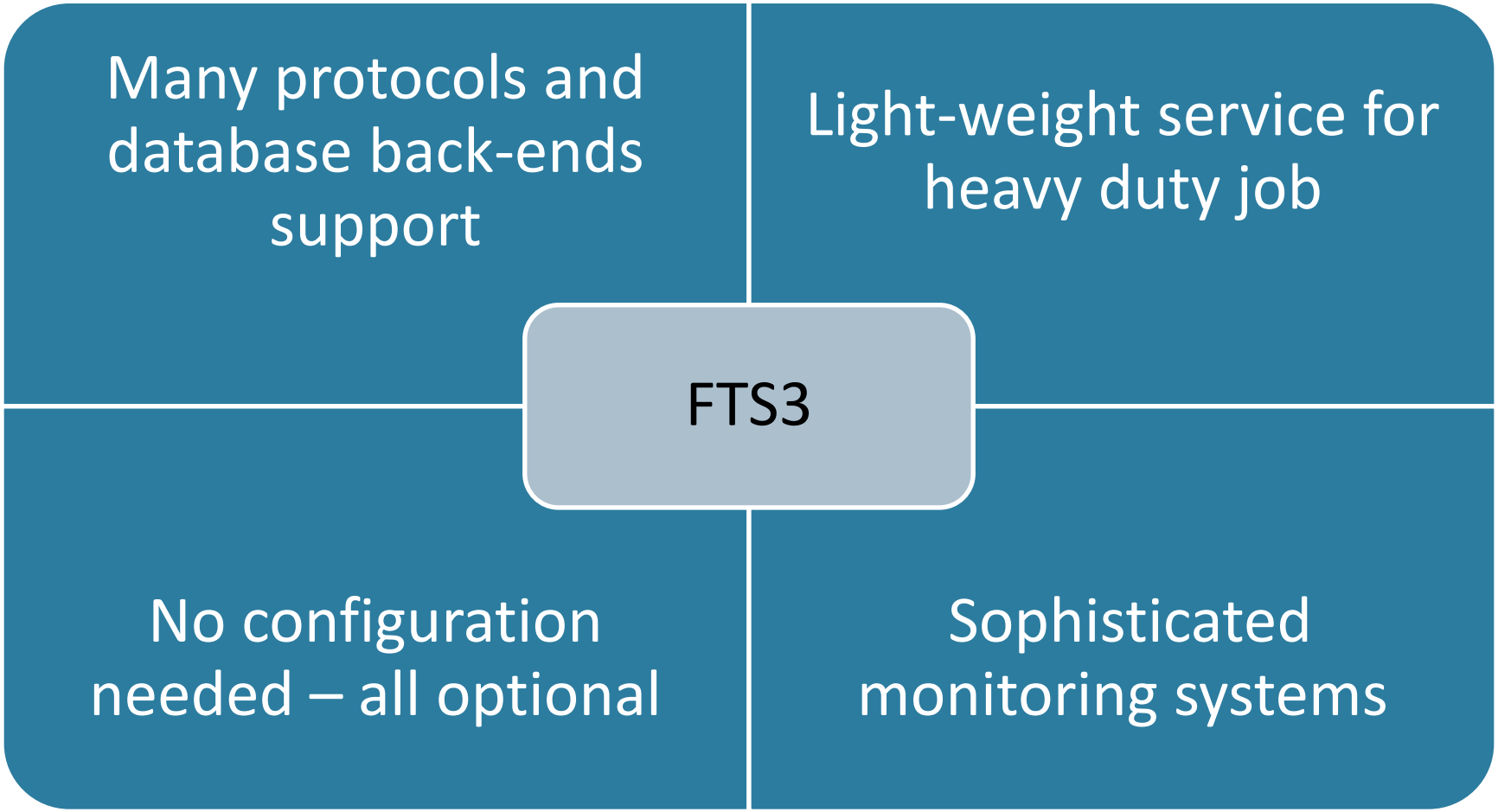
- RUCIO
  - The RESTful interface is great. We use it exclusively in Rucio.
  - The RESTful interface was fast enough for our times-4 scaling tests.
  - The FTS developers quickly adapted a few corner cases to our needs.
  - The state machine messaging queuing is great and we use it intensively.

- IT-PES
  - FTS3 very easy service to run, in particular:
    - All nodes identical, FTS2 had each channel on exactly one node.
    - Installation guide is complete and flexible, just documents what files need to be edited rather than wrapping complexity in scripts.
  - For the future
    - learn it's operation better. It unfortunately has not failed often enough to create any necessity to do so yet
    - Work auto scale out and shrinkage of service - no real hurry but academically interesting to do

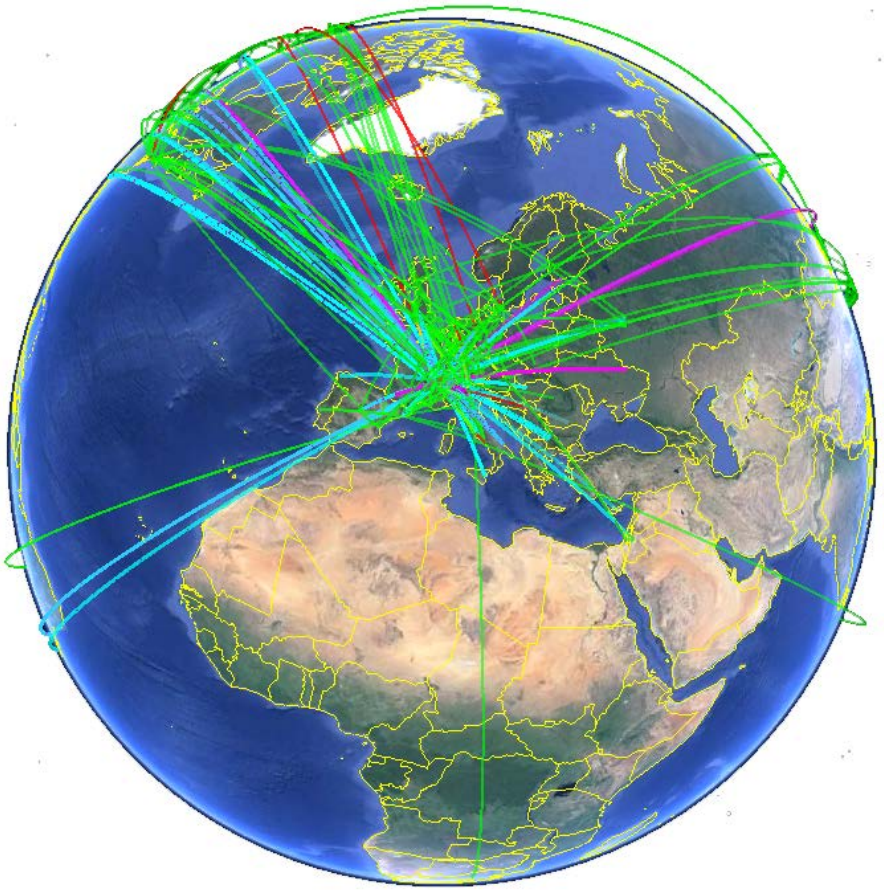
Users - Usage View (2)

- ATLAS ?
- CMS ?
- LHCb computing operations
  - used in production for all WAN transfers with the CERN instance (RAL as backup)
  - interested in a single instance deployment of the FTS3 server
  - currently interacting in "FTS2" mode but going to FTS3 native in the mid-term future
- SITES:
  - RAL
    - from the site perspective FTS3 is a significant improvement over FTS2. It's very easy to install and configure; its ability to use different database backends and ease of adding additional servers to balance load are both useful important features
  - PNNL
    - it is indeed straightforward to set up and run, the CLI is very compatible with the glite-transfer-* tools which makes it nice for transitioning, and the stock web monitoring is highly useful for debugging.
    - Also developers personal commitment to helping us to get this software running properly has been outstanding

FTS3 – WLCG new data movement service



FTS3



Thank you!