Third-party transfers in WLCG using HTTP

MORGRIDGE
INSTITUTE FOR RESEARCH
CORE COMPUTATION

FEARLESS SCIENCE

HTTP-TPC: A protocol for moving bulk data using HTTP

We currently have an *opportunity and a need* to migrate the community's data movement protocols given where GridFTP is in its lifecycle.

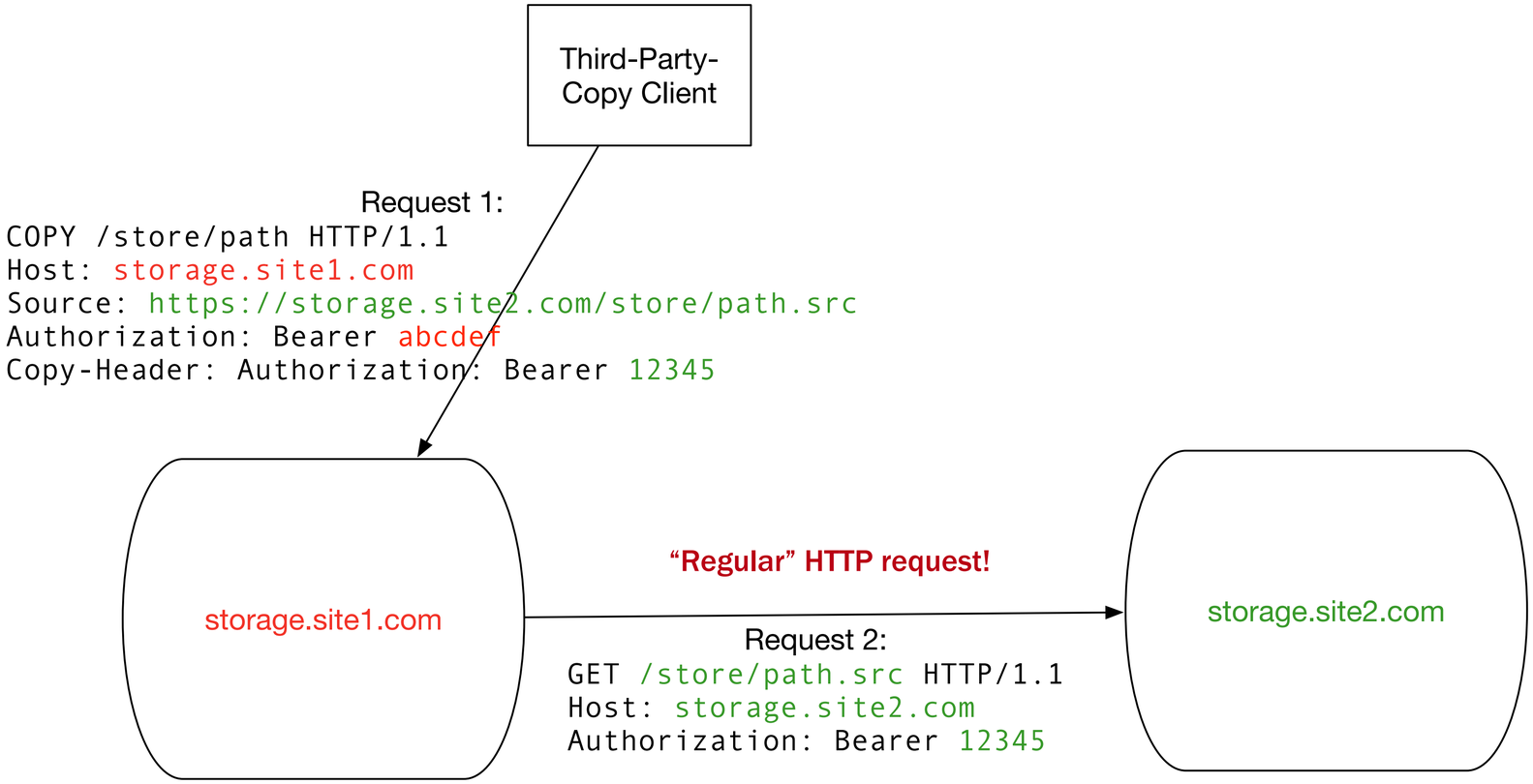
- For several years, there's been ongoing work to develop HTTP to meet our needs.
  - Our small HEP community can leverage the global effort to make HTTPS performant, interoperable, and ubiquitous.
- This builds on a common interpretation of the WebDAV standards, evolving into **HTTP-TPC**.

We have used the last 12 months to greatly mature the implementations and integration with the storage software used in HEP.

- All major storage implementations have demonstrated a HTTP-TPC implementation.
- Except EOS, all have a production version of the protocol.

HTTP-TPC: The Basic Idea

- The client selects either the source or destination to be the **active** side.
- A COPY request is sent, including the headers & URL to use for the real transfer.
- The active side does GET or PUT as needed to move the resource.



Opportunities in HTTP-TPC

- HTTP-TPC is, in the end, just HTTP. You can drive transfers with "any old HTTP client".
  - `curl` is enough to move files (but FTS is better!).
  - In fact, our verification tests are written in `bash+curl` for extra simplicity!
- Flexibility in authentication mechanisms – we've demonstrated pure X509, hybrid X509/token, and pure token.
  - The two servers need not have a common mechanism or trust each other; only the client needs to manage the token.
- Only the **active** side needs to support HTTP-TPC; the other side sees *pure HTTP*.
  - Example: HTTP-TPC can be used to move directly to/from a Ceph S3 instance.
- Actual data movement can be done using a second protocol.
  - dCache implements HTTP-TPC where data is moved via GridFTP.
- Active server can support multiple streams per transfer, reuse TCP streams across transfers, and load-balance multiple user's transfers over the same stream.

HTTP-TPC in WLCG DOMA

The WLCG DOMA TPC group aims to cultivate alternatives to GridFTP.

The group runs continuous, nightly testing of 45 endpoints; on a good day, about 40 function.

- No current version of HTTP-TPC implementations require non-standard work-arounds.
- We keep a "score" of how many daily tests have succeeded over the past 20 days.
- For functioning endpoints, we then run a low-level test matrix using Rucio.

DOMA-TPC smoke test, started 2019-10-30T12:00+0100, took 39:38.

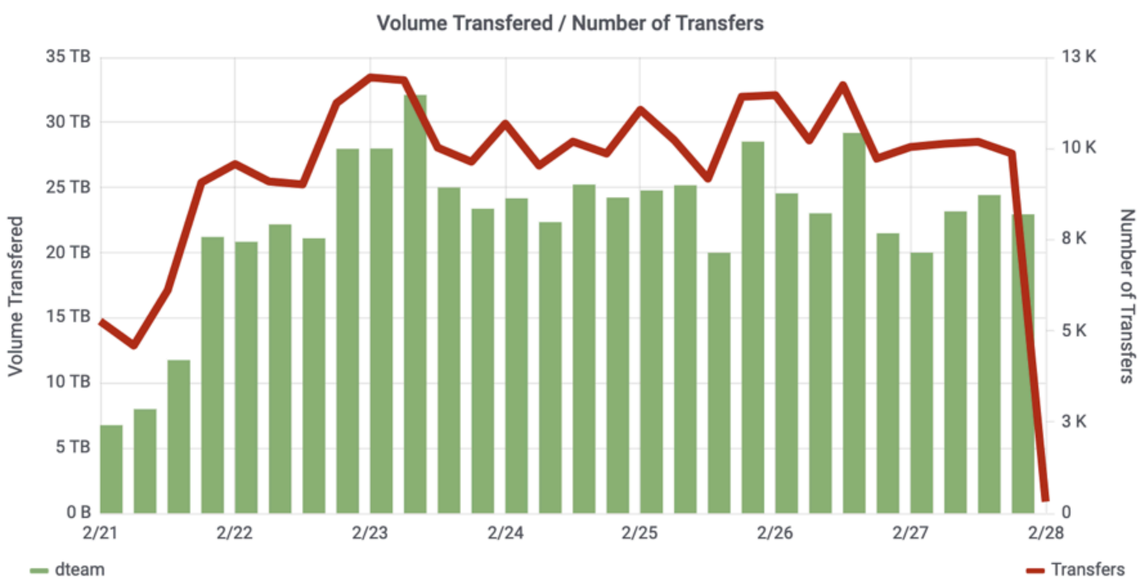
SOUND ENDPOINTS

SCORE	ENDPOINT	SOFTWARE	WORK-AROUNDS
20	AGLT2	dCache	[in 01:21]
20	BEIJING	DPM	[in 02:32]
20	BNL	dCache	[in 00:39]
20	BRUSSELS	dCache	[in 00:47]
20	CA-IAAS	DynaFed	[in 01:00]
20	CALTECH	xrootd-D/HDFS	[in 01:05]
20	CERN-DYN-S3	DynaFed/S3	[in 00:38]
20	CERN-TRUNK	DPM	[in 00:48]
20	DESY-DOMA	dCache	[in 00:25]
20	DESY-PROM	dCache	[in 00:27]
20	FLORIDA2	xrootd-R/Lustre	[in 02:16]
20	FLORIDA	xrootd-D/Lustre	[in 01:31]
20	FNAL	dCache	[in 00:48]
20	IN2P3	dCache	[in 00:24]
20	INFN-T1	StoRM	[in 00:23]
20	KIT	dCache	[in 00:24]
20	LRZ-LMU	dCache	[in 00:42]
20	NEBRASKA2	xrootd-R/HDFS	[in 01:18]
20	NEBRASKA	xrootd-D/HDFS	[in 01:06]
20	PIC-PROD	dCache	[in 02:01]
20	PRAGUELCG2	DPM	[in 00:37]
20	SARA-test	dCache	[in 00:19]
20	TOKYO-LCG2	DPM	[in 01:24]
20	TRIUMF-DYNAFED	DynaFed/S3	[in 01:13]
20	TRIUMF-PROD	dCache	[in 00:50]
20	UKI-BRUNEL	DPM	[in 00:30]
20	UKI-IC	dCache	[in 00:26]

HTTP-TPC Performance

Isn't HTTPS slow?

- Perhaps the most common question we get!
- For nearly a decade, TLS encryption has been performed in hardware: yesterday's server can encrypt faster than today's network card can send. **Encryption is not a bottleneck!**
- A HTTP host with many transfers in flight should achieve within 10% of its `iperf` speed.
- Most common bottleneck: **TCP**. As with GridFTP, we scale aggregate rates through multiple streams.
  - XRootD implements multi-streamed HTTPS for single transfers: not clear this is worthwhile.



See Poster **"Testing the limits of HTTPS single point third party copy transfer over the WAN"** for more information!

Authentication and Authorization

All implementations can move file transfers with X509.

- The active side must be able to authenticate with the inactive. This can be accomplished by:
  - **Delegating** the proxy to the active side, OR
  - Having the client use its the proxy to **generate a bearer token (preferred)** at the inactive side; the client subsequently passes the bearer token to the active side.
- Bearer tokens provide an enormous amount of flexibility:
  - It's the defacto authentication mechanism on the Internet, used by other authorization frameworks such as OAuth2.
  - Both sides need not support the same token format.
- Importantly, WLCG has settled on a token profile for VO-issued tokens: this provides the path forward for a **interoperable, "X509-free" authorization**.

Authorization

The WLCG JWT profile finalized last month – with .

- SciTokens client library has committed to being "dual-profile"; existing users.
- As with SciTokens, this is a very lightweight layer on top of JWT. dCache, XRootD, and StoRM are racing to finish support.

Basic idea: allow VOs to issue tokens that dictate the file access permissions inside their own storage areas.

Next up? Using token exchange to allow FTS to start transfers without proxies.

- Currently in the design phase…

For more information, see Track 3 "WLCG Authorization; from X.509 to Tokens" later today and "Beyond X.509: authN and authZ in practice" on Thursday

Implementation progress over the last year - XRootD

To enable HTTP in your config:

```
xrd.protocol http:1094 libXrdHttp.so
http.cadir /etc/grid-security/certificates
http.cert /etc/grid-security/xrd/xrdcert.pem
http.key /etc/grid-security/xrd/xrdkey.pem
http.secxtractor /usr/lib64/libXrdLcmaps.so
http.listingdeny yes
http.staticpreload http://static/robots.txt \
    /etc/xrootd/robots.txt
http.desthttps yes
```

For several years, the XRootD server software has been multi-protocol: instances can speak either the `xrootd` or HTTPS protocol. Recent improvements:

- Implemented RFCs for checksum calculation.
- Add support for OAuth2-based token request. Internal token format is based on macaroons.
- Matured implementation via a steady stream of bugfixes: no HTTP-TPC work-arounds needed for clients!

Implementation progress over the last year - dCache

dCache support for macaroon-based HTTP-TPC is feature complete for well over a year, so not much development.

- In DOMA testing, dCache has the most test endpoints and provides very reliable, consistent test results.

HTTP-TPC works out-of-the-box with default configuration:

- However, it's possible to configure dCache so it doesn't work,
- Some sites need to update their configuration.

Note this is separate from ATLAS' move away from SRM

- Changes also needed to support non-SRM based uploads.

Future work:

- Help sites enable HTTP-TPC in production instances,
- Update dCache's SciToken support to allow SciToken-authorised macaroon requests,
- Add support for WLCG AuthZ JWT tokens.
- Participate in HTTP-Token testbed,
- Additional monitoring.

Implementation progress over the last year - StoRM

See poster for more info

WebDAV third-party transfer introduced in v. 1.1.0

- In production!

Token-based authorization and delegation

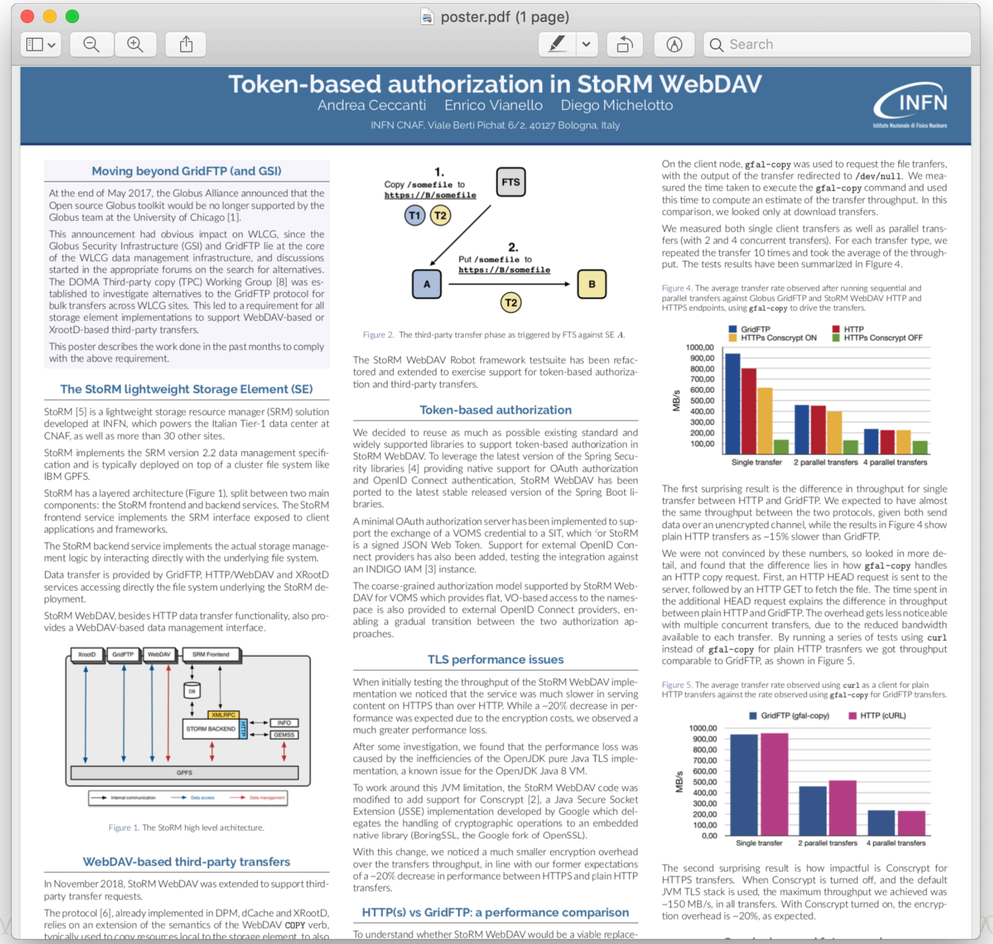
- No GSI/Gridsite delegation support
- OAuth2 endpoint for VOMS credential → token translation

Support for external trusted OAuth2/OIDC authorization server

- Coarse-grained, VO-level authZ

Current work:

- WLCG JWT profile support
- Fine-grained capability/group-based authZ



poster.pdf (1 page)

Token-based authorization in StoRM WebDAV

Andrea Ceccanti Enrico Vianello Diego Michelotto

INFN CNAF, Viale Berti Pichat 6/2, 40127 Bologna, Italy

Moving beyond GridFTP (and GSI)

At the end of May 2017, the Globus Alliance announced that the Open source Globus toolkit would be no longer supported by the Globus team at the University of Chicago [1].

This announcement had obvious impact on WLCG, since the Globus Security Infrastructure (GSI) and GridFTP lie at the core of the WLCG data management infrastructure, and discussions started in the appropriate forums on the search for alternatives. The DOMA Third-party copy (TPC) Working Group [8] was established to investigate alternatives to the GridFTP protocol for bulk transfers across WLCG sites. This led to a requirement for all storage element implementations to support WebDAV-based or XrootD-based third-party transfers.

This poster describes the work done in the past months to comply with the above requirement.

The StoRM lightweight Storage Element (SE)

StoRM [5] is a lightweight storage resource manager (SRM) solution developed at INFN, which powers the Italian Tier-1 data center at CNAF, as well as more than 30 other sites.

StoRM implements the SRM version 2.2 data management specification and is typically deployed on top of a cluster file system like IBM GPFS.

StoRM has a layered architecture (Figure 1), split between two main components: the StoRM frontend and backend services. The StoRM frontend service implements the SRM interface exposed to client applications and frameworks.

The StoRM backend service implements the actual storage management logic by interacting directly with the underlying file system.

Data transfer is provided by GridFTP, HTTP/WebDAV and XRootD services accessing directly the file system underlying the StoRM deployment.

StoRM WebDAV, besides HTTP data transfer functionality, also provides a WebDAV-based data management interface.

Figure 1. The StoRM high level architecture.

WebDAV-based third-party transfers

In November 2018, StoRM WebDAV was extended to support third-party transfer requests.

The protocol [6], already implemented in DPM, dCache and XRootD, relies on an extension of the semantics of the WebDAV `COPY` verb, typically used to copy resources local to the storage element. To also

Figure 2. The third-party transfer phase as triggered by FTS against SE A.

The StoRM WebDAV Robot framework testsuite has been refactored and extended to exercise support for token-based authorization and third-party transfers.

Token-based authorization

We decided to reuse as much as possible existing standard and widely supported libraries to support token-based authorization in StoRM WebDAV. To leverage the latest version of the Spring Security libraries [4] providing native support for OAuth authorization and OpenID Connect authentication, StoRM WebDAV has been ported to the latest stable released version of the Spring Boot libraries.

A minimal OAuth authorization server has been implemented to support the exchange of a VOMS credential to a SIT, which 'or StoRM is a signed JSON Web Token. Support for external OpenID Connect providers has also been added, testing the integration against an INDIGO IAM [3] instance.

The coarse-grained authorization model supported by StoRM WebDAV for VOMS which provides flat, VO-based access to the namespace is also provided to external OpenID Connect providers, enabling a gradual transition between the two authorization approaches.

TLS performance issues

When initially testing the throughput of the StoRM WebDAV implementation we noticed that the service was much slower in serving content on HTTPS than over HTTP. While a ~20% decrease in performance was expected due to the encryption costs, we observed a much greater performance loss.

After some investigation, we found that the performance loss was caused by the inefficiencies of the OpenJDK pure Java TLS implementation, a known issue for the OpenJDK Java 8 VM.

To work around this JVM limitation, the StoRM WebDAV code was modified to add support for Conscrypt [2], a Java Secure Socket Extension (JSSE) implementation developed by Google which delegates the handling of cryptographic operations to an embedded native library (BoringSSL, the Google fork of OpenSSL).

With this change, we noticed a much smaller encryption overhead over the transfers throughput, in line with our former expectations of a ~20% decrease in performance between HTTPS and plain HTTP transfers.

HTTP(s) vs GridFTP: a performance comparison

To understand whether StoRM WebDAV would be a viable replace-

On the client node, `gfal-copy` was used to request the file tranfers, with the output of the transfer redirected to `/dev/null`. We measured the time taken to execute the `gfal-copy` command and used this time to compute an estimate of the transfer throughput. In this comparison, we looked only at download transfers.

We measured both single client transfers as well as parallel transfers (with 2 and 4 concurrent transfers). For each transfer type, we repeated the transfer 10 times and took the average of the throughput. The tests results have been summarized in Figure 4.

Figure 4. The average transfer rate observed after running sequential and parallel transfers against Globus GridFTP and StoRM WebDAV HTTP and HTTPS endpoints, using `gfal-copy` to drive the transfers.

The first surprising result is the difference in throughput for single transfer between HTTP and GridFTP. We expected to have almost the same throughput between the two protocols, given both send data over an unencrypted channel, while the results in Figure 4 show plain HTTP transfers as ~15% slower than GridFTP.

We were not convinced by these numbers, so looked in more detail, and found that the difference lies in how `gfal-copy` handles an HTTP copy request. First, an HTTP HEAD request is sent to the server, followed by an HTTP GET to fetch the file. The time spent in the additional HEAD request explains the difference in throughput between plain HTTP and GridFTP. The overhead gets less noticeable with multiple concurrent transfers, due to the reduced bandwidth available to each transfer. By running a series of tests using `curl` instead of `gfal-copy` for plain HTTP trasnfers we got throughput comparable to GridFTP, as shown in Figure 5.

Figure 5. The average transfer rate observed using `curl` as a client for plain HTTP transfers against the rate observed using `gfal-copy` for GridFTP transfers.

The second surprising result is how impactful is Conscrypt for HTTPS transfers. When Conscrypt is turned off, and the default JVM TLS stack is used, the maximum throughput we achieved was ~150 MB/s, in all transfers. With Conscrypt turned on, the encryption overhead is ~20%, as expected.

Outlook & Future

<u>There is a healthy community building around HTTP-TPC!</u>

What's next for HTTP-TPC?

- Work with the WLCG DOMA TPC group to continue rolling out support at additional sites.
  - We are still missing a handful of T1 sites as well as a CERN EOS instance.
- **Absolutely need to get EOS instances deployed!** Last major storage system that hasn't deployed an endpoint for testing.
- Tackle completely X509-free file movement with FTS.
- Work with the experiments to move production data through their transfer frameworks.
  - Goal: US CMS would like to have at least one site with 30% of its traffic through HTTP-TPC.

We need YOU in the HTTP-TPC Transfer Tests!

Questions?

MORGRIDGE
INSTITUTE FOR RESEARCH
CORE COMPUTATION

morgridge.org

This material is based upon work supported by the National Science Foundation under Grant No. 1836650. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

FEARLESS SCIENCE