EPJ Web of Conferences **214**, 04045 (2019) https://doi.org/10.1051/epjconf/201921404045
CHEP 2018

Bootstrapping a New LHC Data Transfer Ecosystem

Brian Bockelman[1,*], *Andrew* Hanushevsky[2], *Oliver* Keeble[3], *Mario* Lassnig[3], *Paul* Millar[4], *Derek* Weitzel[1], and *Wei* Yang[2]

[1]University of Nebraska - Lincoln
[2]SLAC, Menlo Park, US
[3]CERN, Geneva, Switzerland
[4]DESY, Hamburg, Germany

Abstract. GridFTP transfers and the corresponding Grid Security Infrastructure (GSI)-based authentication and authorization system have been data transfer pillars of the Worldwide LHC Computing Grid (WLCG) for more than a decade. However, in 2017, the end of support for the Globus Toolkit - the reference platform for these technologies - was announced. This has reinvigorated and expanded efforts to replace these pillars. We present an end-to-end alternate utilizing HTTP-based WebDAV as the transfer protocol, and bearer tokens for distributed authorization.

This alternate ecosystem, integrating significant pre-existing work and ideas in the area, adheres to common industry standards to the fullest extent possible, with minimal agreed-upon extensions or common interpretations of the core protocols. The bearer token approach allows resource providers to delegate authorization decisions to the LHC experiments for experiment-dedicated storage areas.

This demonstration touches the entirety of the stack - from multiple storage element implementations to FTS3 to the Rucio data management system. We show how the traditional production and user workflows can be reworked utilizing bearer tokens, eliminating the need for GSI proxy certificates for storage interactions.

1 Introduction

The Worldwide LHC Computing Grid [1] is a global collaboration of over 170 computing sites, linking national and international computing infrastructures, with a mission to provide the data and computational infrastructure to the experiments on the Large Hadron Collider. One of the most important tasks the WLCG collaboration must accomplish is the movement of data between sites. Data movement is both a technical challenge and a social one. Approximately 50PB of data is generated by the LHC experiments each year and must be promptly distributed, processed, and analyzed. Socially, to perform these transfers, the sites participating in the WLCG must agree on a common approach to authentication and authorization infrastructure (AAI) and data movement protocols.

The WLCG does not develop its own software or protocols, but rather serves as a coordinating body between infrastructures such as the European Grid Infrastructure or the Open

*e-mail: brian.bockelman@cern.ch

Science Grid [2]. As such, the WLCG has often settled on technology pioneered by external projects; an important early collection of “grid" technologies was the Globus Toolkit. The Globus team behind the toolkit was an early promoter of the GridFTP data transfer protocol [3] and the “grid security infrastructure" approach to AAI; the toolkit has served as a reference implementation for nearly 20 years.

In 2017, the Globus team decided to end support for the Globus Toolkit [4], ending support for one of the two GridFTP server implementations used on the WLCG. This action reinvigorated efforts to examine additional transfer technologies. In this paper, we show one approach - based on the combination of HTTPS, WebDAV, and capability-token technology - and how these come together to meet the perceived requirements for the WLCG data transfer system.

2 Background

GridFTP has served as a workhorse for the WLCG for over a decade, powering effectively all of its data transfers. To provide a sense of scale, Figure 2 shows the average daily transfer rate using GridFTP over the first 6 months of 2018, as measured via WLCG monitoring. In this section, we go over technical strengths of GridFTP and GSI that have made them a lasting choice for the WLCG.

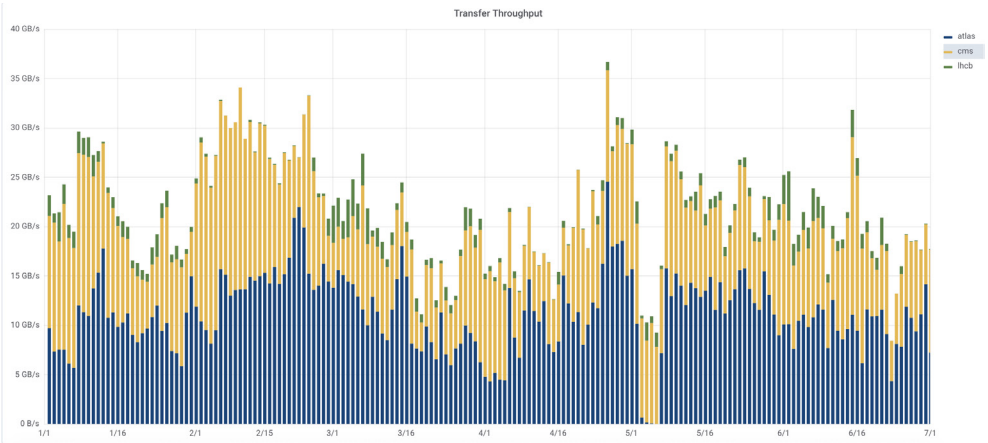

Figure 1. Average daily data rates across three WLCG VOs during the first half of 2018. A sustained rate of 30GB/s over a day corresponds to 2.5PB of data movement.

2.1 GridFTP

The GridFTP protocol is a set of extensions to the venerable File Transfer Protocol (FTP) [5]. These extensions focus primarily on:

- *Implementing third-party-copy* (TPC). To many, a “data transfer" implies data movement between a client and a server. For the WLCG, most data movement is between two sites and coordinated by a central “file transfer service" (FTS). No data is transferred through the client - only between two server endpoints. With GridFTP, the client – the “third party" in the TPC – establishes a session to each GridFTP server. The client instructs one server (the passive server) to start a data transfer and receives an IP address and TCP port in response.

The client then contacts the second server (the active server) and instructs it to send the file contents to the passive server's open TCP port.

- *Enabling data transfer modes for higher transfer performance*. In particular, GridFTP specifies several mechanisms for data to be transferred over multiple TCP streams in parallel. The technique of multiple parallel TCP streams works around a traditional TCP weaknesses for networks with high bandwidth-delay products, which are commonly found in R&D networks (such as the transatlantic links used by the WLCG). Striping modes in the protocol allow for structured subsets of the data to be striped across multiple destination servers, allowing single-file transfer speeds greater than what is available to any given data transfer node. WLCG has traditionally focused on achieving throughput through running many file transfers in parallel as opposed to maximizing single-transfer rates, thus not needing this feature in general.
- *Adding strong authentication, integrity, and encryption to the FTP control channel* (defined by RFC 2228 [6]; implementations are expected to utilize that RFC). Harking back to the earlier days of the Internet, FTP was designed without a mechanism for securing its control channel, allowing for any man-in-the-middle to insert arbitrary commands into the session. GridFTP utilizes GSI for authentication and privilege delegation; GSI is utilized to implement the GSS encryption and integrity layer specified by RFC 2228.

  Further, GridFTP implementations provide a mechanism for checksums, allowing the user to verify the data they expected was transferred.
- *Addition of authentication, integrity, and encryption to the FTP data channel*. GridFTP's GFD.20 specification provides a mechanism for data channel authentication, allowing transfers to also have encryption and integrity checking. However, this is rarely used in practice on the WLCG as one of the major implementations (dCache) does not support this extension and the Globus Toolkit client will silently downgrade transfers to not use encryption if both endpoints do not support it.

Figure 2 shows an overview of how a third-party-copy is performed with GridFTP.

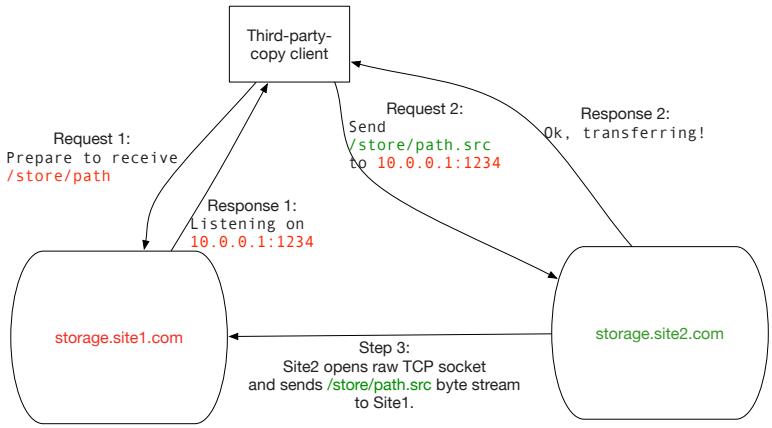

Figure 2. An overview of a GridFTP third-party-copy. The client contacts the passive server and instructs it to prepare to receive a file; the server responds with one-or-more TCP socket addresses. The client will then contact the active server, instructing it to send the source file's bytestream to the open TCP socket. If data channel authentication is not needed, no mutual authentication is done between the two servers.

2.2 GSI

The Grid Security Infrastructure (GSI) is an authentication and authorization infrastructure based upon the more common Public Key Infrastructure [7] used on the Internet with a few extensions and agreed-upon semantics.

One key difference between the infrastructure used in your browser and GSI is the concept of *proxy delegation*: the ability for an entity to take an established identity and create an "impersonation" that is to be treated as a proxy for the original identity. The entity that holds the proxy is entrusted with the rights and privilege of the original identity. Proxies, however, can be revoked and are generally short-lived (a small number of days), reducing the risk compared to giving the original credential to a delegate.

At a technical level, proxies are implemented as short-lived X509 certificates (see RFC 3820 [8] for a technical specification) issued by a non-CA (a so-called end-entity-credential). These X509 certificates can be used as the client certificate in a TLS session (it would, of course, require special validation rules) or as an identity in a GSSAPI session [9]. When transfers are performed using GridFTP, it is common to authenticate the FTP session using GSI and delegating a proxy to the active server. The active server can then utilize the delegated identity to perform data channel encryption.

3 HTTP/WebDAV for the WLCG

Hyper-Text Transfer Protocol (HTTP) [10] is one of the most well-known transfer protocols, underpinning the World Wide Web. It is popular to the point of being "universal" - almost any general-purpose operating system platform will have some sort of HTTP client and server. Compared to GridFTP, where there are a handful of known implementations – and the reference one is no longer maintained – HTTP has a thriving ecosystem of clients. HTTP's popularity means there is significant investment that the WLCG can utilize and reduces the possibility of an implementation monoculture.

GridFTP, like FTP, roughly allows remote procedure calls (uploads, downloads) using a filesystem-like model: there is a concept of directories, listing directory contents, files, etc. HTTP, on the other hand, is more generic: URLs in HTTP are generic resources, there is no concept of "listing" a URL, and HTTP itself is modeled around a request / response session. Since WLCG primarily moves data between filesystems (or filesystem-like storage systems), we utilize the WebDAV [11] extensions on top of HTTP to add the missing filesystem semantics.

In addition to treating the HTTP endpoint as a remote filesystem with WebDAV, other aspects of HTTP utilized by WLCG [12] include:

- *Partial file download support*: A HTTP request can specify that the server should only return a subset of the bytestream behind the request; this capability will be used in Section 3 to partition the transfer over multiple TCP streams as is done with GridFTP.
- *Checksum support*: RFC 3230 [13] specifies an extension to HTTP allowing a server to return a checksum corresponding to a given resource; this allows post-transfer verification of a file's integrity.
- *Built-in Redirections*: Common storage systems on the WLCG achieve high throughput by load-balancing transfers across several disk servers; HTTP's built-in redirection status response allows a single endpoint for the storage system to redirect the clients to the actual server performing the transfer, eliminating the need to proxy data or use virtual IP setups for data transfers.

- *Flexible authorization model*: HTTP requests can contain an opaque value in the 'Authorization' header [14]; as this is an opaque value, the authorization and authentication schemes can be completely decoupled from the transfer protocol. GSI can also be used at the TLS layer of an HTTPS connection; however, this limits the approach to a very specific AAI.

3.1 Third-Party-Copy for WebDAV

WebDAV includes the concept of a HTTP method for creating a copy of a resource; the example provided in RFC 2518 [11] is as follows:

```
COPY /~fielding/index.html HTTP/1.1
Host: www.ics.uci.edu
Destination: http://www.ics.uci.edu/users/f/fielding/index.html
```

The 'COPY' method describes the resource to work on (`/~fielding/index.html`) and a destination resource (`http://www.ics.uci.edu/users/f/fielding/index.html`). If successful, it should result in a copy of the resource at `http://www.ics.uci.edu/~fielding/index.html` to be made at `http://www.ics.uci.edu/users/f/fielding/index.html`. While the intent of the 'COPY' method appears to originally be for transfers within the same endpoint, we have interpreted this to allow for transfers to any external resource. Additionally, we have agreed upon the `Source` header when the service the client is interacting with is the destination of the transfer.

The server where the COPY method is invoked is the *active server*; it will perform any data transfer. If the active server receives the `Destination` header with a HTTPS URL, the simplest implementation would be to perform a HTTP PUT against that destination; more complex implementations are discussed in Section 3.2. Only the active server must understand the 'COPY' verb; it will perform a standard HTTP GET or PUT against the passive server. In this setup, *the passive server need not understand WebDAV*; only a basic HTTP implementation is needed. The flow of WebDAV copy is outlined in Figure 3.

Note the WebDAV scheme provides the client with the ability to specify a full URL for the source or destination; the scheme is not necessarily limited to HTTP or HTTPS. The active server may be asked to do an upload or download in a different protocol – including GridFTP if desired. This allows for a smooth transition from a GridFTP-based ecosystem to a WebDAV-based one.

The passive server needs a mechanism to authorize the transfer request from the active server; this authorization must be provided by the client. As HTTP request authorization are simply passed by the `Authorization` header, we use a mechanism for the client to specify entire headers to copy over. Any header in the client COPY request named `Copy-Header` will have its value copied verbatim into the active server's request to the passive server. That is, if the following header is present in the 'COPY' request:

```
Copy-Header: Authorization: Bearer foo
```

then the HTTP request to the passive server will include:

```
Authorization: Bearer foo
```

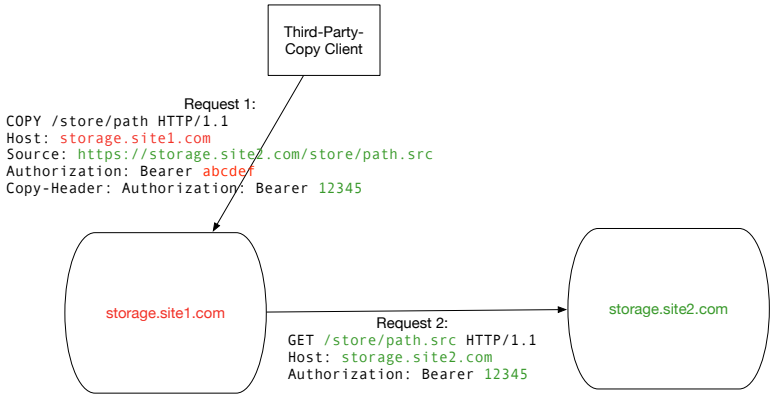

Figure 3. An overview of a WebDAV-based third-party-copy. The client initially contacts the active server, requesting it to perform a COPY. The active server will, in turn, send a GET request to the passive server and respond appropriately to the client with success or failure.

3.2 Multistream

A download of a resource over HTTP is typically envisioned as a single HTTP GET request and a single corresponding response of the file contents. Within HTTP 1.1, this response is delivered over a single TCP stream. In this way, HTTP downloads are typically thought of as single-stream. As one desirable property of GridFTP is the ability to transfer over multiple TCP streams in order to work around well-known TCP weaknesses for high bandwidth-delay-product networks, we have implemented a multistream scheme for WebDAV-based downloads in the implementation for the XRootD framework [15, 16].

The multistream implementation is based on the observation that the HTTP GET method is idempotent while PUT is not; hence, for WebDAV copies in "pull mode" (the active server is the destination, downloading the file from the passive server), we can partition the transfer into multiple chunks, each specifying a byte range within the file. Currently, the chunksize is set to 8MB; optimization of the size is left as future work.

The XRootD server internally maintains a set of TCP connections for each TPC transfer; the list of file chunks to download are added to a work queue. For each idle connection, the transfer thread pulls a unit of work to the work queue and issues the corresponding HTTP GET request; HTTP pipelining is used to ensure two requests to the remote server are always outstanding. The responses are buffered in memory to allow reordering as a single bytestream; this allows the file's checksum to be calculated and persisted.

Currently, transfers in the XRootD implementation are single stream unless the client explicitly specifies the number streams through the `X-Number-Of-Streams` header in the COPY request.

3.3 Performance Markers

When a client issues a COPY request to the active server, determining an appropriate response is surprisingly difficult. If a response is given when the transfer is complete, it may come minutes or hours after the request – it is impossible for the client to differentiate an unresponsive server from a long transfer time. The server could respond immediately that the transfer has started and the client could perform multiple subsequent HEAD requests to see if

the file is growing. However, this leaves no clear mechanism for the active server to indicate the file is done or any subsequent error messages.

We have taken a third approach: the server responds immediately with a failure or that the transfer has been accepted. The response body, however, utilizes HTTP chunked transfer encoding, which allows the response body to be sent as a series of distinct chunks. The client can decode each chunk separately, allowing the server to send a periodic update of transfer progress. We refer to these as "performance markers" after the equivalent GridFTP concept. The performance marker format is as follows (newline characters are shown for readability)

```
Perf Marker\n
Timestamp: $(UNIX TIMESTAMP)\n
Stripe Index: 0\n
Stripe Bytes Transferred: $(BYTES)\n
Total Stripe Count: 1\n
RemoteConnections: $(CONNECTIONS)\n
End\n
```

The `$(CONNECTIONS)` value is a comma-separated list of TCP sockets to the passive server in active use at the time the performance marker was sent. With performance markers, the client can monitor transfer progress – providing it with a mechanism to abort the transfer (shutdown the TCP connection) if insufficient progress is being made. Finally, the last response chunk should be either `success: Created` or `failure: $(ERROR MESSAGE)` with an appropriate, human-readable error message.

4 Conclusions and Future Work

In this work, we have outlined an alternate technical approach to transfers within the WLCG. This work, based on HTTPS / WebDAV, provides features similar to the existing GridFTP ecosystem. There have been multiple implementations of this setup and basic interoperability has been demonstrated. We have shown that the WLCG's transfer service, FTS, has the ability to perform third-party-copies using this mechanism – allowing the higher-level data transfer tools to leverage the new ecosystem with minimal changes.

However, a common understanding and inter-operable implementations are insufficient: we must demonstrate transfers can be done reliably and at the same data rates and scale as GridFTP. For this work, instead of utilizing GSI to authorize the third party copy, we have performed authorizations using the SciTokens [17] and Macaroon [18] token formats. Work must be done to standardize the bearer tokens themselves, as well as agreeing upon standard token acquisition workflows analogous to what is done with GSI. Once technical issues are resolved, there are organizational considerations: one must coordinate site deploys, integration with the higher-level components, and finally switchover of production instances.

Luckily, there is an appetite for change within the WLCG: based partially on this initial work, the Data Organization, Management, and Access (DOMA) working group has formed the "Third Party Copy" task force. We look forward to pursuing this avenue of inquiry within that context and continue to nurture the ecosystem.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 1738962, 1624356, and 1148698.

References

[1] C. Eck, J. Knobloch, L. Robertson, I. Bird, K. Bos, N. Brook, D. Düllmann, I. Fisk, D. Foster, B. Gibbard et al., *LHC computing Grid: Technical Design Report. Version 1.06 (20 Jun 2005)*, Technical Design Report LCG (CERN, Geneva, 2005), http://cds.cern.ch/record/840543

[2] M. Altunay, P. Avery, K. Blackburn, B. Bockelman, M. Ernst, D. Fraser, R. Quick, R. Gardner, S. Goasguen, T. Levshina et al., *A Science Driven Production Cyberinfrastructure—the Open Science Grid* (2011), https://doi.org/10.1007/s10723-010-9176-6

[3] W. Allcock, *GridFTP: Protocol Extensions to FTP for the Grid* (2003), https://www.ogf.org/documents/GFD.20.pdf

[4] I. Foster, *Support for open source Globus Toolkit will end as of January 2018* (2017 (accessed 28 November 2018)), https://github.com/globus/globus-toolkit/blob/4c88c9ca1423e2af806714a2eca54f6eb5d9fd4e/support-changes.md

[5] J. Postel, J. Reynolds, *FILE TRANSFER PROTOCOL (FTP)*, Internet Requests for Comments (1985), https://tools.ietf.org/pdf/rfc2068.pdf

[6] M. Horowitz, S. Lunt, *FTP Security Extensions*, Internet Requests for Comments (1997), https://tools.ietf.org/html/rfc2228

[7] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, W. Polk, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, Internet Requests for Comments (2008), https://tools.ietf.org/html/rfc5280

[8] S. Tuecke, V. Welch, D. Engert, L. Pearlman, M. Thompson, *Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile*, Internet Requests for Comments (2004), https://tools.ietf.org/pdf/rfc3820.pdf

[9] I. Foster, C. Kesselman, G. Tsudik, S. Tuecke, *A Security Architecture for Computational Grids*, in *Proceedings of CCCS* (ACM, New York, NY, USA, 1998), CCS '98, pp. 83–92, ISBN 1-58113-007-4, http://doi.acm.org/10.1145/288090.288111

[10] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee, *Hypertext Transfer Protocol – HTTP/1.1*, Internet Requests for Comments (1997), https://tools.ietf.org/pdf/rfc2068.pdf

[11] Y. Goland, E. Whitehead, A. Faizi, S. Carter, D. Jensen, *HTTP Extensions for Distributed Authoring – WEBDAV*, Internet Requests for Comments (1999), https://tools.ietf.org/pdf/rfc2518.pdf

[12] J. Elmsheuser, R. Walker, C. Serfon, V. Garonne, S. Blunier, V. Lavorini, P. Nilsson, Journal of Physics: Conference Series **664**, 042014 (2015)

[13] J. Mogul, A.V. Hoff, *Instance Digests in HTTP*, Internet Requests for Comments (2002), https://tools.ietf.org/pdf/rfc2068.pdf

[14] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, L. Stewart, *HTTP Authentication: Basic and Digest Access Authentication*, Internet Requests for Comments (1999), https://tools.ietf.org/html/rfc2617

[15] C. Boeheim, A. Hanushevsky, D. Leith, R. Melen, R. Mount, T. Pulliam, B. Weeks, *Scalla: Scalable cluster architecture for low latency access, using xrootd and olbd servers*, http://www.xrootd.org/papers/Scalla-Intro.pdf

[16] B. Bockelman, *Third-party-copy plugin for xrootd* (2018), https://github.com/bbockelm/xrootd-tpc/tree/v0.4.2

[17] A. Withers, B. Bockelman, D. Weitzel, D. Brown, J. Gaynor, J. Basney, T. Tannenbaum, Z. Miller, *Scitokens: Capability-based secure access to remote scientific data* (2018)

[18] A. Birgisson, J.G. Politz, U. Erlingsson, A. Taly, M. Vrable, M. Lentczner, *Macaroons: Cookies with Contextual Caveats for Decentralized Authorization in the Cloud.*, in *NDSS* (2014)