Experiences with http/WebDAV protocols for data access in high throughput computing

2011 J. Phys.: Conf. Ser. 331 072003

(http://iopscience.iop.org/1742-6596/331/7/072003)

Journal of Physics: Conference Series **331** (2011) 072003 doi:10.1088/1742-6596/331/7/072003

Experiences with http/WebDAV protocols for data access in high throughput computing

Gerard Bernabeu[1,2], Francisco Martinez[1,2], Esther Acción[1,2], Arnau Bria[1,3], Marc Caubet[1,2], Manuel Delfino[1,4], Xavier Espinal[1,3]

[1] Port d'Informació Científica (PIC), Universitat Autònoma de Barcelona, Edifici D, ES-08193 Bellaterra (Barcelona) Spain

E-mail: bernabeu@pic.es

Abstract. In the past, access to remote storage was considered to be at least one order of magnitude slower than local disk access. Improvement on network technologies provide the alternative of using remote disk. For those accesses one can today reach levels of throughput similar or exceeding those of local disks. Common choices as access protocols in the WLCG collaboration are RFIO, [GSI]DCAP, GRIDFTP, XROOTD and NFS. HTTP protocol shows a promising alternative as it is a simple, lightweight protocol. It also enables the use of standard technologies such as http caching or load balancing which can be used to improve service resilience and scalability or to boost performance for some use cases seen in HEP such as the "hot files". WebDAV extensions allow writing data, giving it enough functionality to work as a remote access protocol. This paper will show our experiences with the WebDAV door for dCache, in terms of functionality and performance, applied to some of the HEP work flows in the LHC Tier1 at PIC.

1. Remote Storage Access and Worldwide LHC Computing Grid

Most of the experiments using the Worldwide LHC Computing Grid (WLCG) require access to one or many remote storage systems. In order to improve efficiency when using remote storage resources via IP networks many protocols have been implemented, adapted or just adopted.

In the WLCG community it is common to find a wide range of file-based protocols to remotely access network attached storage, composed by a mixture of proprietary/non-standard protocols (GPFS, LUSTRE, Hadoop), protocols developed specifically by the community (XROOTD, RFIO, [GSI]DCAP), protocols adapted to fit in the GRID (gridFTP) and standard protocols (NFS); each one with its advantages as well as caveats and restrictions.

The remote storage clustered file systems available in the WLCG are diverse and not all protocols are available at each system/site. At the moment it is not trivial to choose a single protocol which is a

[2]Also at Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas (CIEMAT), Madrid, Spain
[3]Also at Institut de Física d'Altes Energies, IFAE, Edifici Cn, Universitat Autònoma de Barcelona, ES-08193 Bellaterra (Barcelona), Spain
[4]Also at Universitat Autònoma de Barcelona, Department of Physics, ES-08193 Bellaterra (Barcelona), Spain

perfect fit for all use cases required by the different experiments and sites. This drives the sites contributing in the WLCG to the need of supporting several protocols and, at the same time, forces the experiments to be flexible to the protocols available at each site.

2. Remote Storage Access Protocols

In order to efficiently use a clustered file system on top of a Network Attached Storage (NAS) a file-based protocol which can solve both high and low throughput transfers under random and sequential data access patterns is needed.

At Port d'Informació Científica (PIC), the Spanish WLCG Tier-1 data center, all data stored under dCache[1] is available through several data access protocols: dCap, gsidCap, gridFTP, XRootd, and HTTP. At PIC the WLCG jobs mainly use dCap to read files from dCache to the Worker Nodes in high throughput bulk data transfers, using commands like *dccp*, or in a “read as need” stream-like data access pattern using low throughput data transfers for both random and sequential access. Job data outputs are usually written back to dCache using the gridFTP protocol with high throughput transfers. Whereas this data access model is well understood for most job classes of the WLCG experiments, it is not rare to find inefficiencies and bottlenecks when reading the job input data.

Getting all requirements and wishes implemented in a remote storage access protocol is almost impossible but, focusing in the experience acquired as a WLCG Tier-1 that also serves other experiments with a common system (dCache), it is possible to make a list of what a data access protocol should provide in order to provide fast and reliable WAN transfers and maximize local jobs data access efficiency:

- lightweight control protocol
- client caching capabilities
- fast and reliable data transfer mechanism
- random data access capabilities
- portable interface, easy to use (POSIX)
- standardized and fully described protocol

2.1. NFSv4.1

Network File System (NFS) version 4.1 is the first minor version of NFSv4[2] and extends it by adding support for parallel NFS (pNFS), sessions and directory delegations. What is more interesting for the scientific computing use case about NFSv4.1 is pNFS; it allows to separate the data and meta data paths of the file system, providing the necessary capabilities for a real distributed cluster file system.

NFSv4.1 is probably the most solid candidate as a standard distributed storage access protocol: it is an industry standard supported by many vendors[3], offers a POSIX interface, random access capabilities, fast and reliable data transfer over the TCP/IP protocol stack, default Linux kernel client caching is provided by the Linux client implementation and, thanks to the compound RPC calls, NFSv4.1 provides a better control protocol than its predecessors.

The NFSv4.1 protocol implementation has already shown good performance in the first early tests shown[4] to the WLCG community and it has been included in dCache's supported protocol list[5].

On the other hand, NFSv4.1 is still (November 2010) in experimental stage, so it requires an experimental kernel in the client and has limited functionality from the dCache (1.9.10) server side too. Although it is possible to efficiently transfer data using NFSv4.1 over the WAN, the protocol has not been designed for this purpose and the minimum required components to do it are not yet (November 2010) available at the dCache server; dCache NFSv4.1 server development is focused on exporting data.

2.2. HTTP/WebDav

Web-based Distributed Authoring and Versioning (WebDAV[6]) is a standardized[7] extension of the Hypertext Transfer Protocol (HTTP) protocol adding support to common file operations like file browsing, writing and name space management with support for operations like rename, move and delete files. WebDAV extends HTTP so that it provides most of the capabilities of a regular POSIX file-based protocol while keeping the high data transfer efficiency.

2.2.1. Access HTTP/WebDav.

A HTTP/WebDav share can be accessed in several ways. GNOME, KDE, MS Windows and MAC OS X natively implement a client which can connect to a WebDav server to access and modify remotely stored files. It is possible to access the whole file or just a few random bytes of it.

Most common standard web browsers also support connecting to a WebDav server, even using X.509 certificates for authentication. Traditional HTTP CLI based clients like wget or curl can also access WebDav as any other HTTP URL, making it easier to do high throughput bulk data transfers. The ROOT Data Analysis Framework can use WebDav shares as standard HTTP data sources too.

Under Linux it is possible to access a WebDav server as a mounted POSIX file system using FUSE based user space file system drivers like fusedav[8] or davfs2[9], both having built-in client caching.

2.2.2. dCache as HTTP/WebDav server.

After stability and performance, one of the most recurrent requirements from experiments is to ease the data access, in which dCache helps by providing standardized data access protocols like HTTP/WebDav and NFSv4.1 while keeping files accessible by the more HEP centric protocols like [gsi]dCap, gridFTP or Xrootd, currently in use by the production of the WLCG Tier-1 project. All data stored under dCache can also be managed using the SRMv2.2 protocol and is immutable; it is not possible to open dCache stored files in both read and write modes (RW) at the same time.

In order to provide performance under HTTP/WebDav dCache uses redirect calls for reads, so that the data server (also known as dCache pool) is the source of the data transfer, avoiding bottlenecks. Since some clients, or restricted network setups, might not work with the redirections dCache provides a way to disable them via the dCache configuration file (dcache.conf): webdav.redirect.on-read=false. When redirections are disabled all data transfers flow through the HTTP/WebDav dCache door, which might become a bottleneck.

The latest ROOT client available (20091028-1003) is unable to directly use the dCache 1.9.10-1 WebDav server. It is due to a dCache bug which will be solved in future dCache releases.

3. Performance Comparison On Remote Storage Access Protocols

Performance tests focused in bulk data transfers have been performed in order to determine the suitability of some of the protocols that dCache offers. The comparison wants to emulate high throughput bulk data transfers like the ones observed in production jobs from several WLCG experiments; it focuses on unauthenticated protocols for LAN data read access.

3.1.1. Test bed description.

The testbed infrastructure emulates an standard grid site, in order to avoid disk access bottlenecks all the transfers are memory to memory; source file is cached in the server RAM and destination file is /dev/null.

As described on figure 1 the dCache 1.9.10-1 service is composed by one dCache door dealing with the control part of the transfers and offering several protocols (gsiFTP, [gsi]dCap, XRootd, HTTP/WebDav), one dCache pool (2*10GE NIC, 48GB of RAM) serving the data and the dCache server where all other dCache required services are running (Chimera name server, PoolManager, SRM, InfoServer, etc). On the client side a default SLC5.3 x86_64 Worker Node has been taken from

the computing farm and dedicated to the tests. The Worker Node server has 1GE NIC, two X5355CPUs, 16 GB of RAM and runs the following tests:

- HTTP/WebDav using wget-1.11.4-2: *wget -q -O /dev/null $dCacheHTTPurl*
- Xroot using xrootd-20091028-1003: *xrdcp -s -f $dCacheROOTurl /dev/null*
- dCap using dcap-2.47.2-0 : *dccp $dCacheDCAPurl /dev/null*
- Tuned dCap using dcap-2.47.2-0 with 10MB transfer buffer size: *dccp -B 10000000 $dCacheDCAPurl /dev/null*

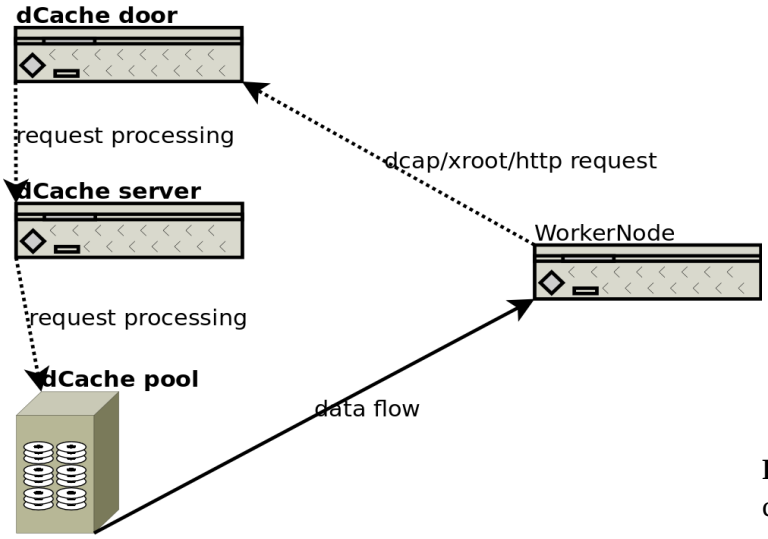

Figure 1. dCache protocol comparison testbed infrastructure

3.1.2. *Big file.*

An ATLAS standard ROOT file of 1496615973 bytes (1427MB) file has been selected for the big file protocol comparison tests, figure 2 shows the required transfer time (lower is better) for 100 runs. Since untuned dCap results are significantly worse than all the other protocols, only the tunned dCap results are taken. The average data transfer and CPU usage time per protocol are:

- wget: average 12453ms (114.6MB/s), 1797ms CPU
- xrdcp: average 12382ms (115.2MB/s), 2914ms CPU
- dcap -B10M: average 14481ms real (98.5MB/s), 1970 CPU

From the tested protocols/commands, the fastest is XRootd/xrdcp shortly followed by HTTP/wget which required 0,57% more time while having lower CPU usage., tunned dCap/dccp is 16% slower than HTTP/wget. It is worth to mention that there was no client/server error in any of the tests.

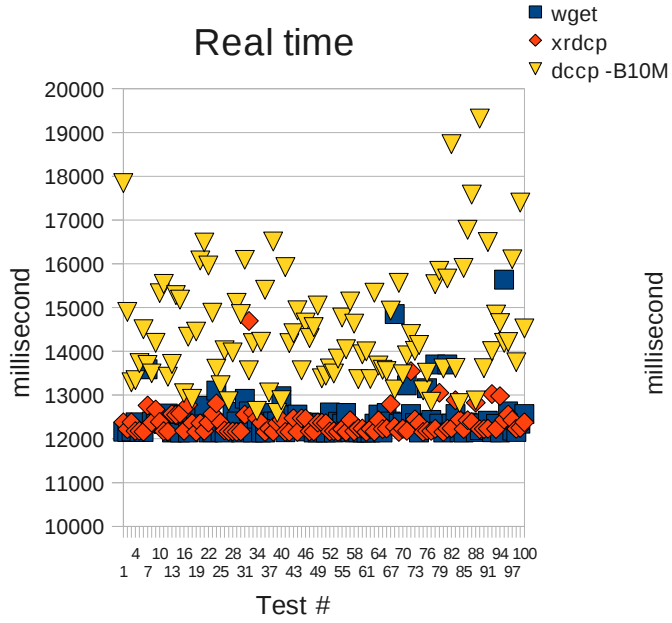

Figure 2. per protocol real time required for 1.4GB file read

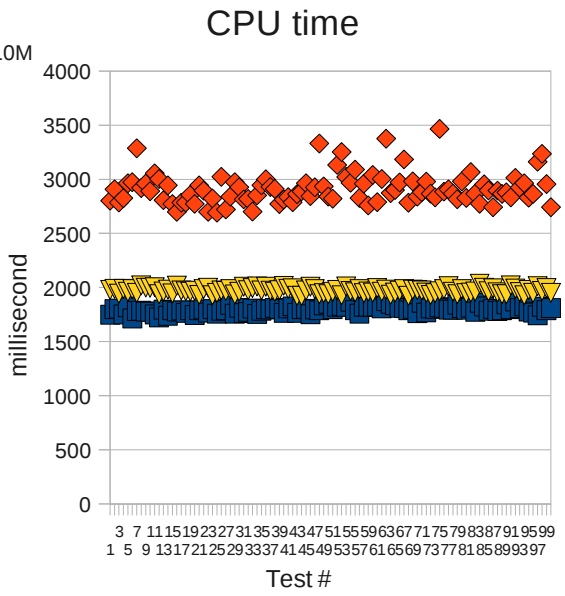

Figure 3. per protocol CPU time required for 1.4GB file read

3.1.3. Small file.

An ATLAS standard ROOT file of 2128456 bytes (2MB) file has been selected for the small file protocol comparison tests. figure 4 shows the required transfer time (lower is better) for 100 runs. In this case dCap tunning makes no difference but for consistency with 3.1.2 tunned dCap results are shown. The average data transfer and CPU usage time per protocol are:

- wget: average 147ms (11.47MB/s), 6.9ms CPU
- xrdcp: average 176ms (9.85MB/s), 20.2ms CPU
- dcap -B10M: average 227ms (10.85MB/s), 13.8 CPU

From the tested protocols/commands, the fastest is HTTP/wget followed by a 19% slower Xrootd/xrdcp that needed 3 times more CPU, tunned dCap/dccp is 54% slower than HTTP/wget. It is worth to mention that there was no client/server error in any of the tests.

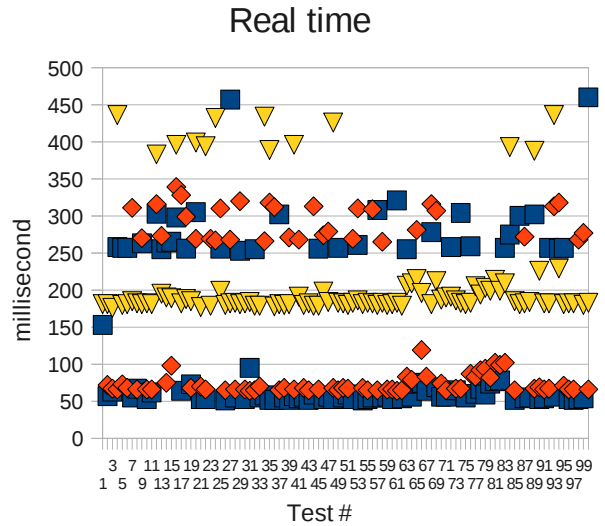

Figure 4. per protocol real time required for 2MB file read

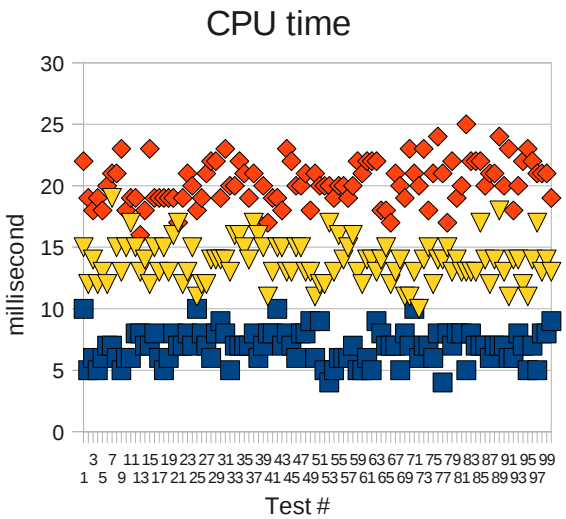

Figure 5. per protocol CPU time required for 1.4GB file read

4. Conclusions

This work shows that HTTP/wget is a lightweight, fast and reliable bulk file transfer solution for both big and small files which can be used to simplify and improve data access efficiency in high throughput computing.

While NFSv4.1 is not yet in production, HTTP is a well known and widely adopted standard protocol that can be used by many clients and it is now available for those sites using recent dCache versions as their storage system, or for sites that want a standard proxied access to distributed storage. More efforts on dCache are required to use dCache's HTTP/WebDav server as a mounted file system, but NFSv4.1 might be a better choice for this.

Acknowledgments

The Port d'Informació Científica (PIC) is maintained through a collaboration between the Generalitat de Catalunya, CIEMAT, IFAE and the Universitat Autònoma de Barcelona. This work was supported in part by grant FPA2007-66152-C02-00 from the Ministerio de Educación y Ciencia, Spain.

References

[1] http://www.dcache.org
[2] Network File System (NFS) version 4 Protocol
http://www.ietf.org/rfc/rfc3530.txt?number=3530
[3] Servers under development at NetApp ONTAP OS, Panasas, EMC Celerra, Oracle. Clients under development at least for Linux, Windows and OpenSolaris.
http://www.pnfs.com/
[4] http://www.dcache.org/manuals/20100419-hepix-dcache.pdf
[5] dCache going standard - The NFS v4.1 dCache implementation.
http://www.dcache.org/articles/i,article-20100401001.html
[6] http://www.webdav.org/
[7] http://tools.ietf.org/html/rfc4918
[8] http://0pointer.de/lennart/projects/fusedav/
[9] http://savannah.nongnu.org/projects/davfs2