EPJ Web of Conferences **245**, 04039 (2020) https://doi.org/10.1051/epjconf/202024504039
CHEP 2019

dCache - Keeping up With the Evolution of Science

Tigran Mkrtchyan[1,*], *Olufemi* Adeyemi[1], *Vincent* Garonne[2], *Dmitry* Litvintsev[3], *Paul* Millar[1], *Lea Morschel*[1], *Albert* Rossi[3], *Marina* Sahakyan[1], *Jürgen* Starek[1] and *Sibel* Yasar[1]

[1]Deutsches Elektronen-Synchrotron DESY, Hamburg, Germany
[2]Nordic e-Infrastructure Collaboration (NeIC), University of Oslo, Norway
[3]Fermi National Accelerator Laboratory, Batavia, USA

Abstract. The dCache project provides open-source software deployed internationally to satisfy ever more demanding storage requirements of various scientific communities. Its multifaceted approach provides an integrated way of supporting different use-cases with the same storage, from high throughput data ingest, through wide access and easy integration with existing systems, including event driven workflow management. With this presentation, we will show some of the recent developments that optimize data management and access to maximise the gain from stored data.

1 INTRODUCTION

The dCache project started in 2000 as a collaboration between Deutsches Elektron-Synchrotron (DESY) and Fermilab National Laboratory. The task was to develop a common storage software for these laboratories that combined commodity heterogeneous disk servers as a caching layer in front of tape storage. In contrast to earlier approaches, the software would provide an experiment agnostic solution, allowing different groups of particle physicists to use a shared infrastructure. A POSIX compliant namespace and the clean separation between that namespace and the location of file's data meant that various operator interventions were possible without requiring a downtime, and that the failure of some storage node resulted in the unavailability of only data stored exclusively on that node. The focus on network protocols that support transferring file's data directly between the client and the node with that file's data allowed dCache to scale, matching the storage demands.

At roughly the same time, CERN's LHC facility began adopting grid technology, resulting in the birth of WLCG: The World-wide Large hadron collider Computational Grid. WLCG is a collection of research institutes and universities across the world that collaborate to provide the storage and computing resources necessary to analyze the data coming from the four LHC experiments. At that time, WLCG followed a strict hierarchical model, with CERN as Tier-0, each region (typically a country) with a single Tier-1 center and multiple Tier-2 centers.

* Corresponding author: tigran.mkrtchyan@desy.de

The Nordic Data Grid Federation (NDGF) joined the dCache project to enhance dCache so it could support their specific use case. The countries contributing to NDGF wished to collaborate in providing a geographically distributed Tier-1 center that would accept data provided any institute in any country is running. This use case is discussed further in section federated systems.

In general, dCache software has proved popular within WLCG. Various laboratories and universities have deployed dCache. Combined, they provide some 50% of the overall WLCG storage capacity. These resources have proved critical to the recent discovery of the Higgs boson[1].

dCache is not limited to particle physics. Over time different user communities are making use of dCache to store their data, in fields as diverse as radio astronomy, biology, photon science, neutrino research, in addition to more prosaic applications, such as a sync-and-share service.

Now, dCache has been used in production for over eighteen years and is deployed throughout the world[2]. From its earliest versions forward, dCache has responded to the challenges presented by new user communities and new ways of working, developing and adopting innovative solutions aimed at satisfying the demands for increased productivity [3]. Such changes included added support for different network protocols (including GridFTP, SRM and HTTP/WebDAV, and multiple authentication schemes (X.509, Kerberos, username+password, OpenID-Connect).

2 Quality of Service

User communities are looking to extract the maximum output from a finite budget. Although the cost-per-terabyte of storage is generally decreasing, there is an increasing demand to store ever more data.

One possibility is to provide differentiated storage; that is, some storage capacity is made available at a lower cost, with a limited IO performance or reliability, while other storage is made available at a higher cost, with enhanced IO performance and reliability. This differentiation could come from many places: from the underlying storage technologies involved, from internal behavior of the storage system, or from procedures of the operations team.

A common example would be to provide an extensive “scratch” storage capacity, while also providing a more restricted “home” storage capacity. The scratch area is intended for data that may be relatively easily recreated, while the home storage contains much more precious information. This distinction in usage allows the site to keep the costs of the scratch area down by, for example, having reduced redundancy and not creating backups.

Often, this differentiation is presented to the users as distinguishable systems: either as distinct systems, or as different locations within some common filesystem. Moving data from one system to another involves copying the data, with a corresponding change to the path. Such changes are problematic, as all references to that path would need to be updated whenever such changes are made.

In dCache we have developed the concept of Quality of Service (QoS) for storage. This describes how data is stored within dCache. This is an extension of the long-standing support for storing data on disk and tape. In addition, we provide a REST API and web-based client that allows users to select and modify the QoS of files.

As part of the INDIGO-DataCloud project[4], the dCache team has developed support for exposing this QoS behavior through the CDMI protocol[5]. This project supported similar developments for other storage systems, so providing a standard mechanism for querying the QoS of data stored in a storage system and, when supported by the underlying storage (such as dCache), modifying that QoS.

The original stimulus for QoS and, indeed, dCache itself, comes from the use of disk as a cache in front of tape for particle physics analysis. Particle physics has long suffered from an excess of data: the process of discovery is akin to searching for a needle in a haystack.

Data is stored on tape media in order to keep down costs: storing all data on disk would be prohibitively expensive. However, different analysis steps require random access to the data, which tape media cannot provide reasonably. Therefore, a hybrid system is needed, with data stored on tape and staged back when necessary. Caching and careful use of resources hides much of the latency associated with tape access.

With the reduction of the price (per GiB) of disks, it has become economical for some data to be stored only on disk. In particular, data that is considered less important because it may be recreated easily by running software, may be stored on less reliable, cheaper disk media. Such changes in storage behavior are easily expressed by introducing new QoS; for example, a low-cost disk-only storage QoS.

3 Federated systems

With dCache, different institutes can contribute towards an aggregated storage system [6]. dCache may be configured so that data is preferentially accepted using local storage when available, falling back to using more remote storage if all local storage is either full or offline. Today, there are two WLCG sites operate in such distributed deployment: Nordic Tier-1, known as NDGF, and Atlas Great Lake Tier-2, known as AGLT2.

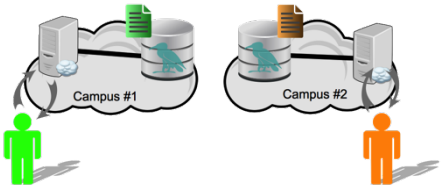
The NDGF Tier-1 is grouped by four Nordic countries to form a single distributed Tier-1 center for WLCG. For storage, there is a single dCache instance that has storage nodes in each of the member institutes. Several institutes have tape libraries to which dCache has access. dCache ensures the Tier-1 center is always able accept data from CERN, provided at least one site is up.

The AGLT2 is a WLCG distributed Tier-2 site, formed from a collaboration between the University of Michigan and Michigan State University. It provides an excellent example of dCache federation ability. Data ingested at any campus is written locally, using storage nodes located on the same campus. However, as this is a single dCache instance, the files are represented in a single namespace and may be accessed from any location.

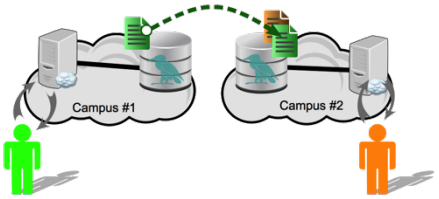
dCache may be configured to maintain multiple copies of certain data. The location of these copies may be constrained; for example, to different geographic locations. This ensures the

data is still available if any location suffers some disaster, or to provide local access to that data before any user attempts to access it.

The configured access constrains allow the configuration to force local access. If the data happens to reside on storage that is not local to the user making the request, then dCache creates an internal copy and allows access from that local copy. On such configured dCache, this ensures that data is always read from local storage. On correctly provisioned systems, the result is that each location will contain a cache of the working dataset.



(a) Data is initially stored at the same campus.



(b) Reading data triggers automatic replication, if necessary

Figure 1: Illustration of data placement in dCache. Data written to a local for user storage and replicated to the remote site when accessed by a remote user.

Write and read locality are illustrated by the simple example given in Figure 1. Data written in Campus #1 is shown as green, while data written by users in Campus #2 is shown as brown. When a user at either campus writes data, local resources are used. When a user in Campus #2 first requests data written in Campus #1, an automatic, internal copy is made so that the data is also stored on Campus #2 resources. This and subsequent requests from Campus #2 users for that data are satisfied by that local copy.

The cache copies of file data are stored within the dCache-managed storage capacity. If this capacity is exhausted, then cached copies are garbage-collected. The cache management algorithm is plugable, with the default scheme following an LRU strategy.

A client requesting access to data stored on campus-local resources will read the data from those resources. However, attempts to read data only stored non-locally will trigger automatic, internal replication of that data to the local campus resources. This and subsequent access will use the local resources, with the local cache copy being available until it is deleted to allow newer replicas.

Though AGLT2 and NDGF sites operate over a decade, such distributed setups are exceptions. However, to reduce operational overhead, more and more small Tier-2s will come together to build so-called data lake [7].

Caching and data placement was always the foundation of dCache design. Nevertheless, to get operational the distributed deployments, a special configuration was needed. Moever, addition and removal of *cache* nodes on a remote site required operation intervention at the main site as well. Thus, such distributed setup might introduce an additional operational overhead.

To reduce operations costs of a distributed deployments, the dCache developers have introduced a concept of a *Zone*. A zone is a virtual container that collects multiple dCache services into a logical group. Those virtual groups are available in data placement and replication rules as well as for dCache inter-component communication. As those grouping rules can be defined to be dynamic, the *caching* nodes can be dynamically added and removed from the system without manual operation on the main site. Moreover, for internal data movement the standard dCache *pool-to-pool* copy is used, which provides data protection and integrity mechanisms, like checksumming or TLS.

With a fine grained configuration, an automatic data replication can be triggered from one zone into another one when data is located in a zone remote for the clients. Such data replication can be triggered for some selected protocols and allow direct access for the others to separate streaming read by HTTP and FTP from latency secetive random access by NFS or XrootD.

In a combination with a read-only namespace replica at the *caching site* a full access to the data can be provided when connection to the main site is lost without compromising authorization and data integrity.

This model can help smaller Tier-2s to become an integral part of their Tier-1s, but benefit from local expertise and technology used in place.

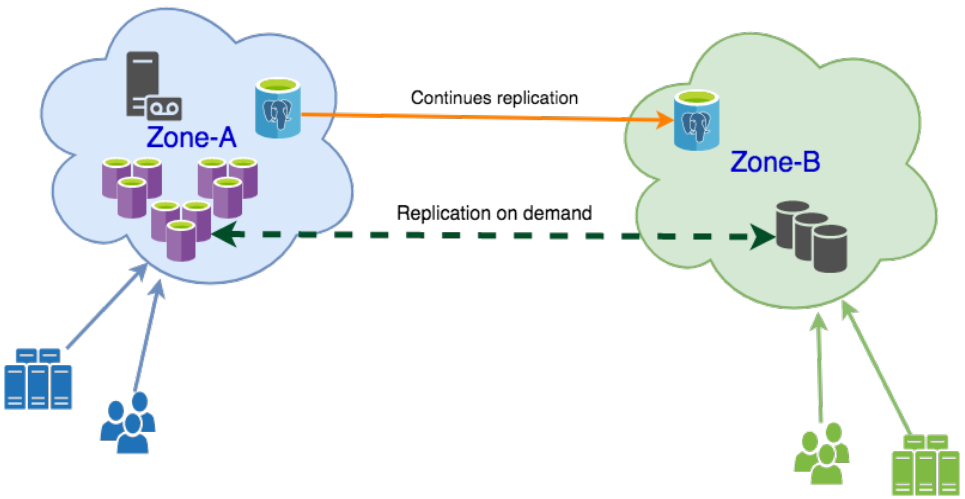


Figure 2: Zone aware dCache deployment

4 NFS/pNFS

One issue when dealing with large volumes of data is how to allow access. Many standard protocols lack the features to benefit from distributed data, common in multi-petabyte storage systems.

Version 4.0 and earlier of the NFS protocol are examples of such behavior: all data access must go through a single node, providing a limitation on the overall performance. However, with the introduction of the pNFS extension in NFS v4.1[8], the NFS protocol has become a practical way of accessing large-scale storage[9]. By separating metadata and the data access paths, clients are able to talk directly to the storage nodes. This allows distributed storage systems to grow throughput and capacity by increasing the number of storage nodes.

For this reason, dCache supports NFS, with an emphasis on pNFS[10]. This allows the storage system to be mounted using standard clients (such as the Linux kernel) without the need for any driver or changes to the application. To the best of our knowledge, dCache is the first storage system to have pNFS placed in production.

The analysis framework ROOT[11] developed and maintained at CERN supports various network protocols, with the ability to add more. This allowed the use of ROOT with dCache via the HEP proprietary protocol, like dcap or XRootD. With a growing demand of non HEP software like Jupiter Notebooks and Apache Spark a POSIX-like data access is expected. Moreover, with availability of opportunistic HPC resources to HEP experiments, the provided POSIX interface must not require any special software installed on the worker nodes. The NFS-mounted dCache allows such communities to use stock Linux machines to access data stored in dCache without needing to adopt their analysis software, which is not always possible.

Another case of non-ROOT based analysis is the use of commercial applications, such as MATLAB, under OS Windows. In order to support Windows users to store and access data stored in dCache, SMB protocol was added. Rather than implementing the SMB protocol support directly in dCache, a protocol translation server was established that runs the open source SAMBA[12] software, while providing access to dCache storage via NFS.

DESY provides its scientists with a sync-and-share service. This service is based on dCache and currently uses nextCloud to support synchronization access and to give users a web-based user interface.

In part, this was made possible because dCache could be NFS mounted, allowing nextCloud software to access dCache through the regular VFS abstraction.

5 Delegated authorization

Traditionally, storage services have authorized requests based on the user's identity. This requires that either the users identify themselves (e.g., username and password, X.509), or that some trusted agent asserts their identity on their behalf (Kerberos, OpenID-Connect, SAML, trusted-host). While a powerful approach, this lacks certain useful characteristics.

For example, a user may wish to allow another person to act on their behalf, but only for a limited time, from a limited location, or limited to certain operations.

With traditional identity-based authorization, delegated authorization is only possible by creating an account for the delegated user. Such delegated authorization (allow Y the same as X, except for...) is often only expressible by copying the authorization rules and amending accordingly. Time-limited authorization is often unavailable, requiring some external process to remove these changes after the desired time has elapsed.

To provide these desired modes of authorization, dCache has introduced delegated authorization: a user may request a token that represents the user's authorized activity. The authorization may be attenuated (for example, making the token authorize only read activity), and allow for the introduction of limitations on time and IP address.

These delegated authorization tokens are based on macaroons, a technology developed by Google in which a chain of Hash-based Message Authentication Code (HMAC) protected strings, called caveats. Each caveat limits some aspect of the macaroons use; for example, what operations are authorized, from which IP addresses operations are allowed, for which time period the token is valid. For further details, see [13].

One advantage of macaroons over simpler tokens, such as JSON Web Token (JWT), is that it supports autonomous attenuation: any agent with a macaroon token can create a new, more limited version of the token offline, but removal of any existing caveats is cryptographically hard. This allows an agent to receive a somewhat limited token (e.g.,
read-only and valid for one day) and to create a further-limited macaroon (e.g, only a single file is readable, valid for one minute) quickly and without any interaction with any other system.

Third-party transfers are when data is copied between servers without that data travelling through the client. For network protocols with a separate control channel (e.g., FTP) and that support redirection (the GridFTP extensions), third-party transfers are easily achieved.

For protocols without a separate control channel (e.g., HTTP), one server must itself act as the client and make requests to the other. When acting as a client, the server needs some delegated credential with which to authorize the transfer.

If the client requests a macaroon as the delegated credential, that credential may be limited to transfer a specific file and be valid for only a relatively short duration. This limits how much damage may be done if that credential is misused (e.g., is stolen), so limiting the trust that user places in the server.

If a group of scientists wish to collaborate when analyzing some data, they all require access to that data. If some of those scientists are from a different institute, then accessing the data becomes problematic. Possible solutions include obtaining accounts on the storage system, making the data publicly (albeit obfuscated) accessible, or sharing credentials. None of these solutions is particularly appealing: creating accounts is often a bureaucratic process that might not be possible for non-local users, making data public may be unacceptable, and sharing credentials goes against security best practice.

A potential solution is for the off-campus scientists to receive a macaroon (for example, as a macaroon-embedded URL). This would provide time-limited access to some subset of the data, optionally with additional safeguards such as access only being possible from certain IP addresses.

Such an approach would foster collaborations that would otherwise be difficult to achieve.

6 Conclusion

In this paper, we have presented some of the challenges faced by scientific communities: fluid and changing expectations on storage, geographically spread communities, use of commodity software, and community-driven authorization. We have also outlined the various solutions adopted by dCache to tackle these problems. The various use cases that have been presented demonstrate real-world usage of the concepts that dCache supports, while being sufficiently generic that they may support other user communities.

Finally, the dCache team continues to work on innovative and useful additions to dCache, which will pave the way for future scientific discovery.

7 References

1. J. Wengler, *How grid computing helped cern hunt the higgs,* 2012, Available from https://sciencenode.org/feature/how-grid-computing-helped-cern-hunt-higgs.php [accesses 2020-03-01]
2. P. Fuhrmann and V. G¨ulzow, *dCache, storage system for the future,* in European Conference on Parallel Processing. Springer, pp. 1106–1113, (2006),
3. A. Millar, T. Baranova, G. Behrmann, C. Bernardt, P. Fuhrmann, D. Litvintsev, T. Mkrtchyan, A. Petersen, A. Rossi, and K. Schwank, *dCache, agile adoption of storage technology,* in Journal of Physics: Conference Series, **vol. 396**, no. 3. IOP Publishing, pp. 32 077–087, (2012).
4. D. Salomoni, I. Campos, L. Gaido, G. Donvito, M. Antonacci, P. Fuhrman, J. Marco, A. Lopez-Garcia, P. Orviz, I. Blanquer et al., *Indigo-datacloud: foundations and architectural description of a platform as a service oriented to scientific computing,* arXiv preprint arXiv:1603.09536, (2016).
5. *Cloud Data Management Interface (CDMI) v1.1.1*, SNIA, iSO/IEC 17826:2016.
6. G. Behrmann, P. Fuhrmann, M. Grønager, and J. Kleist, *A distributed storage system with dcache,* in Journal of Physics: Conference Series, **vol. 119**, no. 6. IOP Publishing, p. 062014, (2008).
7. I.Bird, S.Campana, M.Girone, X. Espinal, G. McCance and J. Schovancová, *Architecture and prototype of a WLCG data lake for HL-LHC*, EPJ Web of Conferences **214**, 04024 (2019)
8. S. Shepler, M. Eisler, D. Noveck, *Network File System (NFS) Version 4 Minor Version 1 Protocol*, RFC 5661, DOI:10.17487/RFC5661 (2010), Available from https://tools.ietf.org/rfc/rfc5661.txt [accessed 2020-03-01]
9. D. Hildebrand and P. Honeyman, *Exporting storage systems in a scalable manner with pnfs,* in Mass Storage Systems and Technologies, Proceedings. 22nd IEEE/13th NASA Goddard Conference on. IEEE, 2005, pp. 18–27, (2005).

10. J. Elmsheuser, P. Fuhrmann, Y. Kemp, T. Mkrtchyan, D. Ozerov, and H. Stadie, *LHC data analysis using NFSv4.1 (pNFS): A detailed evaluation,* in Journal of Physics: Conference Series, **vol. 331**, no. 5. IOP Publishing, p. 052010, (2011).
11. https://root.cern.ch/ [accessed 2020-03-01]
12. http://www.samba.org [accessed 2020-03-01]
13. A. Birgisson, J. G. Politz, U. Erlingsson, A. Taly, M. Vrable, and M. Lentczner, *Macaroons: Cookies with contextual caveats for decentralized authorization in the Cloud,* DOI: 10.14722/ndss.2014.23212
14. *cloud*, in NDSS, (2014).