EPJ Web of Conferences **214**, 04042 (2019) https://doi.org/10.1051/epjconf/201921404042
CHEP 2018

dCache - storage for advanced scientific use cases and beyond

Tigran Mkrtchyan[1,1], *Olufemi* Adeyemi[1], *Patrick* Fuhrmann[1], *Vincent* Garonne[2], *Dmitry* Litvintsev[3], *Paul* Millar[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 storage software deployed internationally to satisfy ever more demanding scientific storage requirements. 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. In supporting new communities, such as photon science and microbiology, dCache is evolving to provide new features and access to new technologies. In this paper, we describe some of these recent features that facilitate the use of storage to maximise the gain from stored data, including quality-of-service management, support for distributed and federated systems, and improvements with support for parallel NFS (pNFS).

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

1Corresponding author: tigran.mkrtchyan@desy.de

provide the storage and computing resources necessary to analyse 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 centre and multiple Tier-2 centers.

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 centre 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 fifteen 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 its corresponding less desirable characteristics, while other storage is made available at a higher cost, with its corresponding more desirable characteristics. This differentiation could come from many places: from the underlying storage technologies involved, from internal behaviour 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 the above simple example, data may seldom change from scratch to home or vice versa; however, in a more general scientific workflow, this is more common.

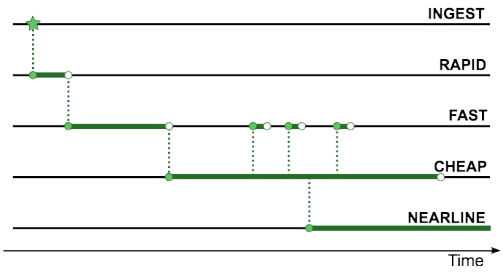


Figure 1: A typical timeline showing how data changes QoS

A reasonable example is illustrated in Figure 1. In this example, data is first accepted from scientific instruments using hardware dedicated to fast ingest of data (INGEST). This hardware is dedicated to streaming access, ensuring it can maintain the high bandwidth necessary to keep up with the incoming data.

Once written, the data is then immediately moved to hardware that supports random access (RAPID). This is to allow an initial analysis that performs some basic checks, to decide whether this data contains any meaningful results.

As the data is accepted, it is moved to similar storage capacity that allows more detailed scientific analysis (FAST). This is distinct hardware from RAPID to avoid any potential competition of resources from regular user scientific analysis, which would risk slowing down the accept/reject decision.

The data continues with FAST QoS for an initial period as freshly received `hot' data will likely see an initial surge of analysis work. Later on, with the data becoming `cold' and less interesting, it is moved to slower and cheaper technology (CHEAP).

Although the data is stored in CHEAP, it is still subject to occasional analysis. Such requests for access may be satisfied either directly from this QoS or by giving the data FAST QoS for a limited time. In particular, if the data is part of some analysis campaign then it may make sense to pin the data with FAST QoS for the period of that campaign.

The data forms part of a publication and so is cited in the paper. Therefore, it must be kept for an extended period --- longer than it would otherwise. To ensure this, the data is given an additional QoS: NEARLINE. This storage further reduces the likelihood of data loss and cost, but with increased latency when accessing the data.

All these QoS changes should take place without the users having to modify how they access the data: neither the endpoint nor the path should change. Instead, data should be accessible from the same path, triggering changes to the QoS automatically, as necessary.

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 behaviour 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.

2.1 QoS in Particle physics

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 cheaper disk media. Such changes in storage behaviour are easily expressed by introducing new QoS; for example, a disk-only storage QoS.

2.2 QoS for Photon science

Although leaving out many details, the use case shown in Fig1 describes the data flow for the PETRA-III and the European-XFEL facilities, located at DESY campus. The initial fast data ingest happens close to the detectors, using physically close storage media. This provides fast access for initial analysis and other time-critical computing activities, using computers that are also located physically close to the beam-lines. The data is copied to the DESY IT building, into hardware that is dedicated to streaming writes or reads. Once ingested, an internal copy is made to support both analysis and writing to tape archive.

2.3 Delayed write to tape

Deleting data from tape leaves ``holes'' in the tape: areas of tape that do not hold useful information but which cannot be used to write fresh data, since data has been written subsequently in the tape. These holes may be recovered through a process called reclamation: the valid contents of a tape is read and written to other tapes. Since reclamation requires exclusive access to tape drives for extended periods, it is desirable to reduce the need for this operation. To achieve this, precious data is written initially to disk. Once the data passes the validation process, the QoS is modified and the file is migrated to tape.

2.4 Data preservation

With data preservation, scientific data is maintained beyond the lifetime of the collaboration that created it. This is done so that the data may be used in new and novel ways not originally foreseen; for example, as an educational resource or as part of new scientific research.

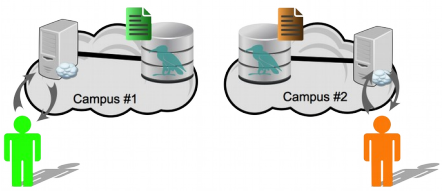
This topic covers a wide range of activity. One (small) part is the ability to store data reliably over time: bit preservation. By identifying bit preservation with a specific storage QoS, it becomes easier to identify which data should be handled carefully, so greatly reducing the likelihood of data loss and a subject of regular integrity validations.

3 Federated systems

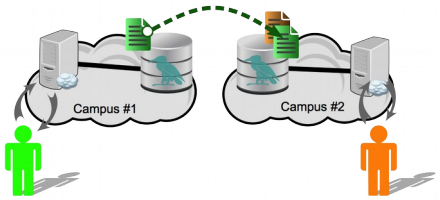
With dCache, different institutes can contribute towards an aggregated storage system[6]. dCache may be configured so that data is preferentially accepted using local storage when available, falling back to using more remote storage if all local storage is either full or offline.

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 2: Illustration of data placement in dCache

Write and read locality are illustrated by the simple example given in Figure 2. 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 a LRU strategy.

3.1 NDGF Tier-1

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

3.2 Atlas Great Lakes Tier-2 (AGLT2)

The AGLT2 is a WLCG distributed Tier-2 site, formed from a collaboration betweens 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.

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.

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[7], the NFS protocol has become a practical way of accessing large-scale storage[8]. 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[9]. 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.

4.1 Non-ROOT bases analysis

The analysis framework developed and maintained at CERN (ROOT) 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. In contrast, the analysis software used by other communities, for example Belle-I collaboration, is expecting POSIX interface. 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[10] software, while providing access to dCache storage via NFS.

4.2 DESYcloud: a sync-n-share service

DESY provides its scientists with a sync-and-share service. This service is based on dCache and currently uses nextCloud to support synchronisation 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 authorisation

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

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

These delegated authorisation 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 authorised, from which IP addresses operations are allowed, for which time period the token is valid. For further details, see [11].

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.

5.1 Third-party transfers

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 authorise 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.

5.2 Ad hoc data sharing

If a group of scientists wish to collaborate when analysing 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 authorisation. 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 2019-03-24]
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. 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 2019-03-24]
8. 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).
9. 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).
10. http://www.samba.org [accessed 2019-03-24]
11. 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*, in NDSS, (2014).