FERMILAB-CONF-18-369-CD

Storage for advanced scientific use-cases and beyond

A. Paul Millar*, Olufemi Adeyemi*, Gerd Behrmann†, Patrick Fuhrmann*,
Vincent Garonne†, Dmitry Litvinsev‡, Tigran Mkrtchyan*, Albert Rossi‡,
Marina Sahakyan*, Jürgen Starek*
*Deutsches Elektron-Synchrotron (DESY)
Notkestraße 85,
22607 Hamburg, Germany
†Nordic e-Infrastructure Collaboration (NeIC)
Stensberggata 25,
NO-0170 Oslo,
Norway
†Fermilab National Laboratory
PO Box 500,
Batavia IL 60510-5011,
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 this paper, we describe some of the recent features that facilitate the use of storage to maximise the gain from stored data, including quality-of-service management, heterogeneous systems — both through integrated tertiary storage support and geographical locality — the parallel NFS (pNFS) extension, and innovative delegated authorisation schemes.

I. 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 a shared infrastructure, allowing different teams 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 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[1], [2]. WLCG is a collection of research institutes and universities across the world that collaborate to 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 centres.

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 III.

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[3].

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[4]. 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[5]. Such changes included added support for different network protocols (including GridFTP[6], SRM[7] and HTTP[8][9]/WebDAV[10]), and multiple authentication schemes (X.509[11], Kerberos[12], username+password, OpenID-Connect[13]).

In this paper we present a selection of these solutions that

This manuscript has been authored by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics.

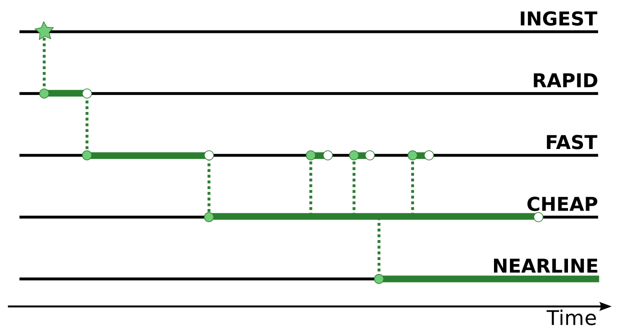


Fig. 1: A typical timeline showing how data changes QoS

have been recently adopted in dCache; specifically, the focus on storage quality of service, application of federated storage, support for NFSv4.1 / pNFS and delegated credentials. We also present some active areas of work that we anticipate will be available in future versions of dCache.

II. 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[14].

A reasonable example is illustrated in Fig. 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 that 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.

After some time has elapsed, with no further analysis expected, the data is no longer stored with QoS CHEAP. If the data is not stored with any other QoS it would be deleted at this point; however, in this case, the data still has QoS NEARLINE and is maintained. If subsequent access is needed, then it is temporarily promoted to QoS CHEAP.

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.

dCache provides a REST API and web-based client that allows users to select and modify the QoS of files. This allows dCache to react to changes in how files should be stored. Additionally QoS transitions may be configured to operate without requiring user interaction.

As part of the recent INDIGO-DataCloud project[15], the dCache team has developed support for exposing this QoS behaviour through the CDMI protocol[16]. 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.

Use cases

The QoS concept is one that is key to dCache. In this section we briefly outline various use-cases that demonstrate the benefits from QoS.

1) 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) Photon science: Although leaving out many details, the use-case shown in Fig. 1 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.

People using the facility have access to their data through a web portal, which will stage data back from tape (as necessary) and provide access to their data as a single file using a container format.

3) Delayed write to tape: Data that is written to tape is known as "precious" data; for example, in particle physics, the raw data cannot be reproduced due to the unpredictable nature of quantum mechanics. Similarly the "monte-carlo" simulated data requires considerable computing power and cannot easily be reproduced.

In some cases, although the data may be important, it often is not immediate clear if the data is valid when it is written. A subsequent analysis step is required to validate the data. Examples include the "reconstruction" step in particle physics or certain photon science experiments.

In effect, 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.

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. This requires, amongst other things, multiple copies of the data and a regular tests to verify the integrity of the data.

Such preservation activities can be expensive, therefore it is important that they are applied only to the data deemed worth preserving. 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.

III. Federated systems

Collaborations are a long-standing feature of scientific endeavour; however, with the increased size of datasets, both the handling of data and access to the data becomes more of a concern. Ad hoc mechanisms often do not scale well, and simply copying all the data requires equal storage capacity in all locations.

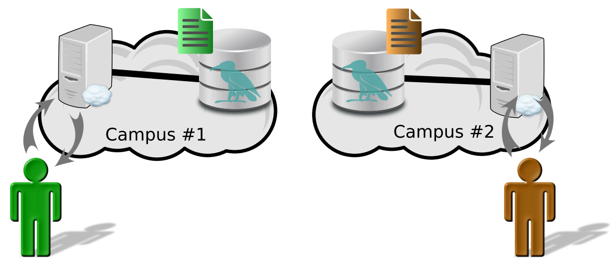
With dCache, different institutes can contribute towards an aggregated storage system[17]. dCache may be configured so that data is preferentially accepted using local storage when available, falling back to using more remote storage if all local storage is either full or offline.

In general, a file's data can reside within dCache in multiple locations. This may be to satisfy redundancy constraints, as an autonomous response to reduce load, or to satisfy configured access constraints.

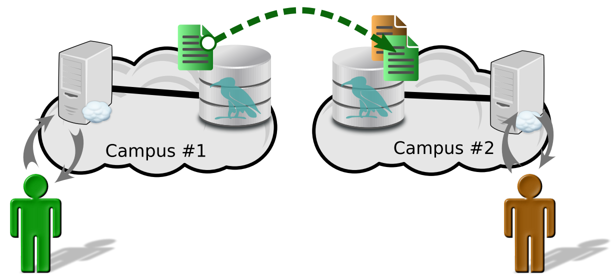
dCache may be configured to maintain multiple copies of certain data. The location of these copies may be constrained; for example, to different geographic locations. This ensures the data is still available if any location suffers some disaster, or to provide local access to that data before any user 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.

Write and read locality are illustrated by the simple example given in Fig. 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



(a) Data is initially stored at the same campus



(b) Reading data triggers automatic replication, if necessary

Fig. 2: Illustration of data placement in dCache

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 copied.

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 pluggable, with the default scheme following an LRU strategy.

Use cases

The ability to steer where data is written and to trigger internal copies within dCache provides support for some interesting federation use-cases. In this section we identify two such use-cases.

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.

2) Midwest Tier-2: The Midwest Tier-2 is a WLCG distributed Tier-2 site, formed from a collaboration amongst the University of Chicago, Indiana University and University of Illinois. 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

IV. 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 behaviour: all data access must go through a single server, providing a limitation on the overall performance. However, with the introduction of the pNFS extension in NFS v4.1[18], the NFS protocol has become a practical way of accessing large-scale storage[19]. This is shown in Fig. 3, where data access is directly between the storage nodes and the clients. This allows a correctly deployed dCache to achieve the desired throughput.

For this reason, dCache supports NFS, with an emphasis on pNFS[20]. To the best of our knowledge, dCache is the first storage system to have pNFS placed in production. 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.

Use cases

The pNFS extension, as part of the NFS v4.1 specification, is perhaps the first commercially adopted, standard protocol to support the server redirecting the client to the server hosting the data. This provides key benefits that are described in this section.

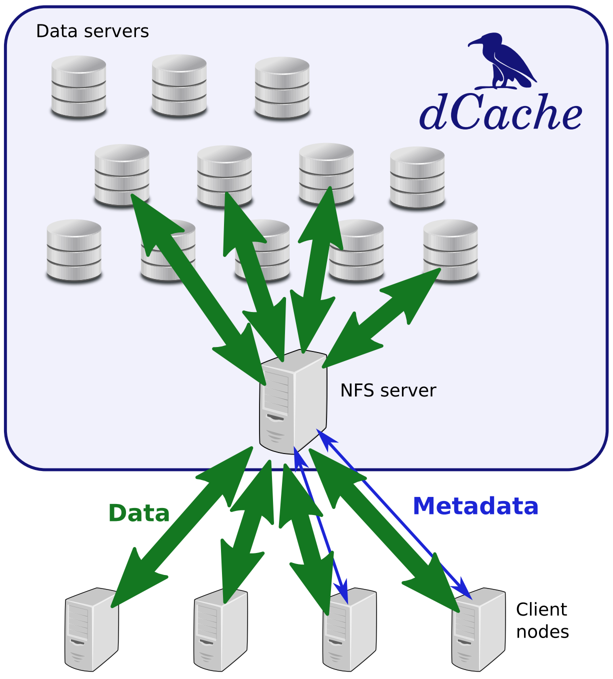
1) Belle II analysis: The analysis framework developed and maintained at CERN (ROOT) supports various network protocols, with the ability to add more. This allowed the use of ROOT with dCache via the dCache proprietary protocol: dcap.

In contrast, the analysis software used by the Belle II collaboration is available in a binary-only form. This precludes adding support for proprietary network protocols.

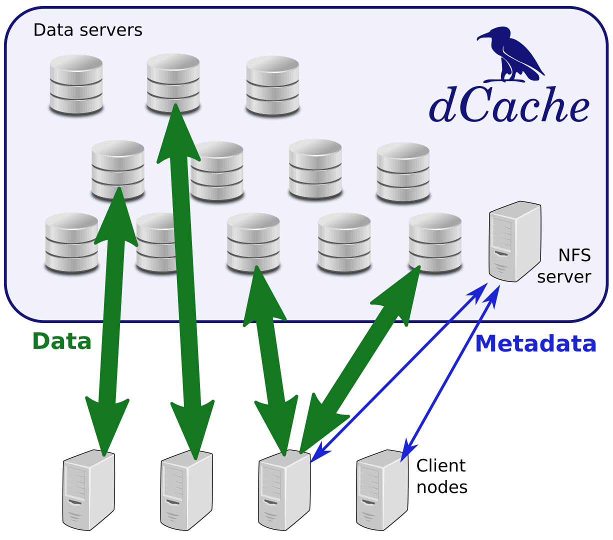
The availability of NFS-mounted dCache allows stock Linux machines to support Belle II analysis of data stored in dCache.

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 ownCloud 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 ownCloud software to access dCache through the regular VFS abstraction.



(a) Traditional NFS (without pNFS) proxies all data transfers



(b) With pNFS extension, data and metadata traffic are separated

Fig. 3: Graphical representation showing the benefits of the pNFS extension

3) Easy analysis: DESY provides scientists with servers for direct analysis work: small analysis runs and to try out algorithms before they are deployed (at scale) using the grid.

DESY also hosts the German Nation Analysis Facility (NAF), that allows particle physics analysis in a more interactive fashion that is currently possible through grid resources.

In contrast to grid-based analysis, both these facilities drop the storage abstraction. By NFS-mounting dCache, analysis work can access files stored in dCache like any other files on their computer.

4) SMB support: Support for SMB was added to dCache services allowing a small number of Windows users to store and access data stored in dCache. Rather than implementing the SMB protocol support directly in dCache, a protocol translation server was established that runs the open-source SAMBA software, while providing access to dCache storage via NFS.

V. 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.

Such complications make this relatively common desire sufficiently difficult that users may resort to non-recommended practise to achieve the same goal; e.g., sharing passwords.

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 [21].

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.

The use-case that first drove the introduction of macaroons was to provide better support for a web-portal. This web-portal manages scientific data collected by people who are granted short-term access of a facility located at DESY. As the web-portal has access to domain-specific metadata, it provided an enhanced view of the data stored in dCache. The users may then select data for downloading based on metadata criteria.

A further complication is that the users of this web-portal are not necessarily known to dCache; instead, the portal accesses an independent user database. The web-portal has credentials that allow access to the data. In effect, this delegates the authorisation decision to the web-portal. As it is the web-portal's credential that is authorised in dCache, only it may access the data. This means that all downloaded data must go through the portal, which introduces performance limitations.

To alleviate this, macaroons support was introduced into dCache. This allows the web-portal to acquire a macaroon from dCache that authorises the download of a specific file for a short time, based on its credential. This macaroon is then handed to the client for this specific file download, which then downloads the file directly from dCache.

Use cases

Macaroons provide a basic building block with which sites and users can extend the services that dCache provides. In this section we describe some of these use-cases.

1) Portal access: A web-portal may enhance the user experience by allowing file or dataset navigation, selection and transfers based on domain-specific metadata. The portal might also allow users to authenticate in ways that are not configured within dCache, or the portal knows about users that the dCache instance does not.

Without support for delegated authorisation, the portal would have to act on behalf of its users. In particular, all data transfers would have to go through the portal.

With macaroons, the portal can create an authorisation token that authorises a specific transfer by a specific client, that is valid for a short period. The portal can do this by either taking an existing macaroon and adding caveats, or by requesting a fresh macaroon from dCache. The portal then redirects the client to dCache with the macaroon embedded in the URL.

2) 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.

3) 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 access only being possible from certain IP addresses.

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

VI. Future work

The information we have presented in this paper so far has focused on the work already achieved. In this section, we outline some of the areas in which dCache is continuing to evolve, providing innovations for storage users.

A. REST API

dCache has focused on delivering support for features based on standard network protocols, such as NFS v4.1/pNFS. However, some features of dCache are not present in existing network protocols. Therefore, dCache is developing a REST based API to expose both these dCache-specific features and the activity that is availabel through other network protocols.

This API is documented using OpenAPI, allowing automatic client and library creation. Amongst other benefits, this will simplify creating web portals that interact with dCache.

B. Events / inotify support

Work is underway in providing support for events. This is a notification system where clients can register for events and learn when certain activities take place. One important aspect is support for inotify-like events: those indicating namespace and file-access activity.

The inotify support will allow users to build complicated, domain-specific behaviour without modifying dCache. Such possibilities include processing data as it is ingested, such as automatic metadata extraction or some initial analysis.

Notification also allows external software to remain synchronised with a dCache instance's namespace as files are uploaded, renamed and deleted. For example, large collaborations typically maintain a database of which files or datasets are available in different storage systems; keeping this information up-to-date is challenging. Additionally, domain-specific metadata may be held outside of dCache; this metadata catalogue should be updated whenever files are uploaded or deleted.

C. XDC, smart caches and data orchestration

The eXtreme DataCloud (XDC) project is an European Horizon-2020 funded project (Number 777367–XDC–H2020-EINFRA-2016-2017). As a partner in this project, DESY will be improving dCache to support smart caches and data orchestration.

Smart caches involve software where some central storage service provides the main storage capacity. All new data is written into that system. Additionally various satellite storage systems provide caches of the data stored in that central storage system. This differs from traditional caches by the caches acting autonomously independent of user activity; for example, a cache may decide to fetch data that has just been written, or data that has become more popular.

Data orchestration involves triggering computing activity based on events within the storage system. Complex workflows may be constructed that are followed when data is ingested, using computing resources provisioned from cloud systems as demand requires.

D. Expanding sync and share support

Currently shared files within DESYcloud are only visible through the front-end layer: either the web-based UI or via the sync client. Direct access to the underlying dCache will allow a user to access their own data, but not any of the data shared by other users.

We anticipate improving dCache so that shared files are also visible when accessing directly. This would be for all dCache supported protocols, including NFS, HTTP/WebDAV and GridFTP. An NFS-mounted dCache would provide instant high-performance access to all data without requiring mirroring of that data.

E. Improve support for object stores

dCache already supports using CEPH to store file data. We anticipate expanding this storage support to include S3 devices. Much as dCache currently provides a unified view of heterogeneous storage, we will also investigate how to federate different object storage under a single namespace and management.

VII. 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 been 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.

References

[1] D. Deatrich, S. Liu, C. Payne, R. Tafirout, R. Walker, A. Wong, and M. Vetterli, “Managing petabyte-scale storage for the atlas tier-1 centre at triumf,” in *High Performance Computing Systems and Applications, 2008. HPCS 2008. 22nd International Symposium on*. IEEE, 2008, pp. 167–171.

[2] M. Crawford, C. Dumitrescu, D. Litvintsev, A. Moibenko, and G. Oleynik, “Scalability and performance improvements in the fermilab mass storage system,” in *Journal of Physics: Conference Series*, vol. 396, no. 5. IOP Publishing, 2012, p. 052024.

[3] J. Wengler, “How grid computing helped cern hunt the higgs,” 2012.

[4] P. Fuhrmann and V. Gülzow, “dcache, storage system for the future,” in *European Conference on Parallel Processing*. Springer, 2006, pp. 1106–1113.

[5] 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, 2012, pp. 32 077–32 087.

[6] I. Mandrichenko, W. Allcock, and T. Perelmutov, “Gridftp v2 protocol description,” OpenGridForum, GFD-R-P 47, May 2005. [Online]. Available: \url{https://www.ogf.org/documents/GFD.47.pdf}

[7] F. Donno, L. Abadie, P. Badino, J.-P. Baud, E. Corso, S. D. Witt, P. Fuhrmann, J. Gu, B. Koblitz, S. Lemaitre, M. Litmaath, D. Litvintsev, G. L. Presti, L. Magnoni, G. McCance, T. Mkrtchan, R. Mollon, V. Natarajan, T. Perelmutov, D. Petravick, A. Shoshani, A. Sim, D. Smith, P. Tedesco, and R. Zappi, “Storage resource manager version 2.2: design, implementation, and testing experience,” *Journal of Physics: Conference Series*, vol. 119, no. 6, p. 062028, 2008. [Online]. Available: http://stacks.iop.org/1742-6596/119/i=6/a=062028

[8] R. Fielding and J. Reschke, “Hypertext transfer protocol (HTTP/1.1): Message syntax and routing,” Internet Requests for Comments, RFC Editor, RFC 7230, June 2014. [Online]. Available: \url{http://www.rfc-editor.org/rfc/rfc7230.txt}

[9] ——, “Hypertext transfer protocol (HTTP/1.1): Semantics and content,” Internet Requests for Comments, RFC Editor, RFC 7231, June 2014, http://www.rfc-editor.org/rfc/rfc7231.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc7231.txt

[10] L. Dusseault, “HTTP extensions for web distributed authoring and versioning (WebDAV),” Internet Requests for Comments, RFC Editor, RFC 4918, June 2007. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4918.txt

[11] S. Tuecke, V. Welch, D. Engert, L. Pearlman, and M. Thompson, “Internet x.509 public key infrastructure (pki) proxy certificate profile,” Internet Requests for Comments, RFC Editor, RFC 3820, June 2004.

[12] C. Neuman, T. Yu, S. Hartman, and K. Raeburn, “The kerberos network authentication service (v5),” Internet Requests for Comments, RFC Editor, RFC 4120, July 2005, http://www.rfc-editor.org/rfc/rfc4120.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4120.txt

[13] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, and M. C., “Openid connect core 1.0 incorporating errata set 1,” http://openid.net/specs/openid-connect-core-1_0.html, OpenID, Tech. Rep., 2014.

[14] C. Jung, M. Gasthuber, A. Giesler, M. Hardt, J. Meyer, F. Rigoll, K. Schwarz, R. Stotzka, and A. Streit, “Optimization of data life cycles,” in *Journal of Physics: Conference Series*, vol. 513, no. 3. IOP Publishing, 2014, p. 032047.

[15] 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.

[16] *Cloud Data Management Interface (CDMI) v1.1.1*, SNIA, March 2015, iSO/IEC 17826:2016.

[17] 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, 2008, p. 062014.

[18] D. N. S. Shepler, M. Eisler, and D. Noveck, “Rfc5661-network file system (nfs) version 4 minor version 1 protocol,” *URL: http://tools. ietf. org/pdf/rfc5661. pdf, j an*, 2010.

[19] D. Hildebrand and P. Honeyman, “Exporting storage systems in a scalable manner with pnfs,” in *Mass Storage Systems and Technologies, 2005. Proceedings. 22nd IEEE/13th NASA Goddard Conference on*. IEEE, 2005, pp. 18–27.

[20] 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, 2011, p. 052010.

[21] 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.