dCache: Inter-disciplinary storage system

Tigran Mkrtchyan[1,*], *Krishnaveni* Chitrapu[4], *Vincent* Garonne[2], *Dmitry* Litvintsev[3], *Svenja* Meyer[1], *Paul* Millar[1], *Lea* Morschel[1], *Albert* Rossi[3], *Marina* Sahakyan[1]

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

Abstract. The dCache project provides open-source software deployed internationally to satisfy ever more demanding storage requirements. Its multifaceted approach provides an integrated way of supporting different use-cases with the same storage, from high throughput data ingest, data sharing over wide area networks, efficient access from HPC clusters and long term data persistence on a tertiary storage. Though it was originally developed for the HEP experiments, today it is used by various scientific communities, including astrophysics, biomed, life science, which have their specific requirements. In this paper we describe some of the new requirements as well as demonstrate how dCache developers are addressing them.

1 INTRODUCTION

The dCache project started in 2000 as a collaboration between Deutsches Elektron-Synchrotron (DESY) and Fermilab National Accelerator 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 the 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 the 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, also knows as Monarc model.

**Corresponding author: tigran.mkrtchyan@desy.de

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].

Now, dCache has been used in production for over twenty years and is deployed throughout the world[2]. Increasingly, sites are using dCache to support communities that have different requirements from WLCG; for example, DESY facilities and services now support photon sciences, biology, future accelerators, R&D and others. 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].

2 Federated systems

With dCache, different institutes can contribute towards an aggregated storage system [4]. 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 operating 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 universities from Denmark, Sweden, Norway, Finland and Slovenia to form a single distributed Tier-1 center for WLCG (Fig. 1). 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 to. dCache ensures the Tier-1 center is always able to accept data from CERN, provided at least one site is up.

Figure 1: NDGF Tier-1 - Five countries (Slovenia is not shown), one dCache.

The AGLT2 is a WLCG distributed Tier-2 site, formed from a collaboration between the University of Michigan and Michigan State University. They provide an excellent example of dCache's federation ability. Data ingested at any campus is written locally, using storage nodes located on site. 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.

Though AGLT2 and NDGF sites operate over a decade, such distributed setups are exceptions. However, to reduce operational overhead and improve data availability, more and more small Tier-2s will come together to build a so-called data lake[5].

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 an automatic dCache 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. In addition to on-demand replication, dCache provides mechanisms for manual data placement. The replicas can be cached copies only, e.g. removed when space is needed, or have a guaranteed availability windows on the remote site, to support scheduled data processing activities.

Caching and data placement was always the foundation of the dCache design. Nevertheless, to get distributed deployments operational, a special configuration was needed. Moreover, addition and removal of cache nodes on a remote site required operation intervention at the main site as well.

To reduce these operation costs of distributed deployments, the dCache developers have introduced the concept of a *Zone*. A zone is a logical group that combines multiple dCache services and is available in data placement and replication rules as well as for dCache inter-component communication. In High-Availability setups, where redundant copies of critical components are running at different Zones, the Zone local instance of a component is preferred to reduce performance penalties introduced by network latency. As for data movement the standard dCache internal pool-to-pool copy is used, all data integrity mechanisms, like check summing or TLS are available for in-transit data protection.

Such caching sites can be spawned on-demand when the data processing should be extended by external compute resources without so-called managed storage.

The dCache services, Zones and the caching nodes can be dynamically added and removed in the distributed deployment without configuration changes on other sites. This reduces operation overhead of distributed deployments and provides a foundation for highly available dynamic storage federation. Moreover, in a combination with a site-local read-only namespace replica, which prefers availability over consistency in case of network partitioning, a full access to the data can be provided even 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.

3 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, like use of spinning disks and SSDs, from internal behavior of the storage system, like use of RAID systems and erasure coded JBODs, or from procedures of the operations team.

In dCache we have developed the concept of Quality of Service (QoS) for storage. This describes how data is stored within dCache and exposed as a REST API, which is developed as a part of INDIGO-DataCloud project[6]. This is an extension of the long-standing support for storing data on disk and tape.

The original stimulus for QoS and, indeed, dCache itself, comes from the use of disks as cache in front of tape for particle physics analysis. However, QoS has more dimensions than just latency and price. The new requirements like durability, availability and technology independence play an important role in the data life cycle of scientific communities, like EuFXEL[7].

To address those requirements the *Resilience*[8] subsystem of dCache, which is responsible for data durability, is evolving into *QoS Engine*. A significant amount of Resilience's architecture needed to be refactored in order to be placed on top of the already implemented functionality. Those changes are described in a separate paper.

4 Non HEP Analysis

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, resulting in a limitation on the overall performance. However, with the introduction of the pNFS extension in NFS v4.1[9], the NFS protocol has become a practical way of accessing large-scale storage[10]. 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[11]. 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[12] developed and maintained at CERN supports various network protocols, with the ability to add more. This allowed the use of ROOT with dCache via HEP proprietary protocol, like dcap or XRoot. With a growing demand of non HEP software like Jupyter 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 adapt their analysis software, which is not

always possible. With recent developments in the dCache inter-component communication protocol[13], the overhead of the internal communication is reduced, which improves HPC job efficiency, where the file's metadata access is as important as the data access itself.

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, the 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[14] software, while providing access to dCache storage via NFS.

In close collaboration with Linux kernel developers, dCache is an early adopter and demonstrator of the evolution of the NFS protocol and gives an *Early Access* to new developments, like *extended file attributes*[15] over NFS or *NFS-over-TLS*[16].

5 Tape interface

Some scientific communities, like photon scientists, for example, show a chaotic file size distribution with a peak toward small files. Direct storing of the user files results in unacceptable pool efficiency of tertiary storage systems and tape in particular, as tapes are optimized for streaming large files. As dCache bridges the file system view with the underlying storage and manages transitions between media, it is the natural place to solve the pool performance of storing small files on tape. Since 2015 a so-called *Small File Service*[17] has been deployed at DESY, that packs files into containers based on specified aggregation policies. As existing detectors have been updated to run under higher trigger rates and new beamlines become operational, the number of arriving files has increased drastically, bringing the pack service to its limits.

To cope with increased file arrival rate, the Small File Service is redesigned for better resource utilization and scalability. First of all, a shell-script based interaction with the pack service, which spawns a new process for each file that should migrate to a tape, is replaced with a dCache nearline-storage driver implementation. The new driver runs as an integral part of the dCache disk server and is responsible for all interactions between dCache and the pack service. It shares all allocated resources, thus drastically reducing the overall load on the system. Moreover, the nearline-storage driver has direct access to the files that have to be packed and can expose them directly to pack service by-passing dCache request scheduler and increases pack system throughput by reducing file access latency, as shown in Fig. 2. Moreover, the nearline-storage driver, that has been developed for Small-File service is the foundation of future work on dCache with the CERN Tape Archive (CTA) [18] integration.

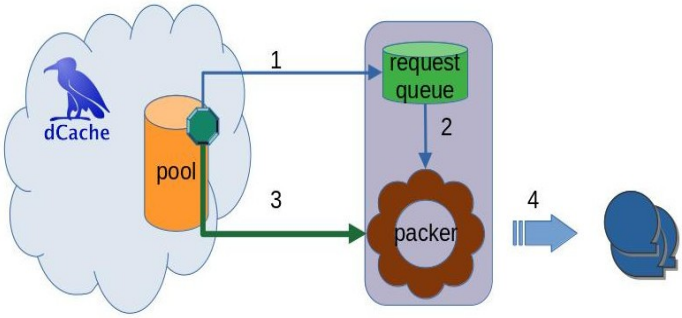

Figure 2: Small File Service operation: (1) request pushed into job queue; (2) policy based packing; (3) reading files to be packed directly from data server; (4) ready container stored no tape.

Originally dCache's tertiary connectivity interface was optimized for efficient writing to tapes and file recall was viewed to be a rare event. The recent ATLAS *data carousel*[19] activities with large-scale recalls have disproved this assumption. Unordered requests to tape libraries resulted in a high number of tape mounts and reposition requests, thus overall low tape access efficiency. After analyzing tape access logs provided by participating sites bulk-recall optimization strategies were identified and implemented[20]. Those changes are described in a separate paper.

6 Storage Events

Some communities, especially those, who process images, need to populate metadata catalogs before the data or data set can be processed. Traditionally, this is done by repeatedly querying a storage system to learn if new data is available. This approach doesn't scale with the number of clients, as each query uses resources to obtain the current status, resulting in some upper limit on the number of concurrent activities a service may sustain.

Storage events are a new way of interacting with storage. In contrast to the traditional request-response pattern, the storage system generates events when something happens, like file creation, access, removal or migration to/from a tape storage, and asynchronously delivers them to the clients. As storage events do not require polling, the client learns of changes very quickly. It also scales well, since the subscriptions typically require very few resources.

dCache provides two ways for clients to receive events: either using Apache Kafka or via W3C's Server-Sent Events (SSE) protocol. The two event delivery mechanisms have different trade-offs with different intended audiences: they complement each other rather than providing alternatives. Currently, these two event delivery mechanisms each provide a non-overlapping set of possible events: some types of event are only available via Apache Kafka while other event types are only available via SSE. The Apache-Kafka based events

are suitable for integration with IT infrastructure, like monitoring system or transfer analytic, while SSE based events are for web-based end-user applications.

A new development activity within the dCache project tries to extend the namespace component with a metadata catalog functionality, that will be populated automatically when new data arrives by utilizing storage events to execute scientific community specific metadata extraction application.

7 Summary and Outlook

In this paper, we have presented some of the challenges faced by scientific communities: fluid and changing expectations on storage, geographically spread deployments, use of commodity software and archival of huge amount of small files. 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.

8 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 2021-02-21]
2. P. Fuhrmann and V. Gülzow, *dCache, storage system for the future,* in European Conference on Parallel Processing. Springer, pp. 1106–1113, (2006), doi: 10.1007/11823285_116.
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), doi: 10.1088/1742-6596/396/3/032077
4. 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), doi: 10.1088/1742-6596/119/6/062014
5. 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), doi: 10.1051/epjconf/201921404024
6. 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).
7. https://www.xfel.eu/ [accessed 2021-05-20]
8. A L Rossi, et al., "Data Resilience in the dCache Storage System", Journal of Physics: Conference Series, Volume **898**, (2017), doi :10.1088/1742-6596/898/6/062024

9. 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 2021-02-21]
10. 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).
11. 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), doi: 10.1088/1742-6596/331/5/052010
12. https://root.cern.ch/ [accessed 2021-02-21]
13. L. Morschel et al., dCache – Efficient Message Encoding For Inter-Service Communication in dCache, EPJ Web of Conferences **245**, 05017(2020), doi:10.3204/PUBDB-2020-04903
14. https://www.samba.org [accessed 2021-02-21]
15. M. Naik, M. Eshel, File System Extended Attributes in NFSv4, Available from https://tools.ietf.org/rfc/rfc8276.txt, doi:10.17487/RFC8276 [accessed 2021-02-21]
16. T. Myklebust, C. Lever, Towards Remote Procedure Call Encryption By Default, Available from https://datatracker.ietf.org/doc/draft-ietf-nfsv4-rpc-tls/ [accessed 2021-02-21]
17. K. Schwank et al., Transparent handling of small file with dCache to optimize tape access, Journal of Physics: Conference Series664(2015) 042048, doi:10.1088/1742-6596/664/4/042048
18. E. Cano, V. Bahyl, C. Caffy, G. Cancio, M. Davis, J. Leduc, O. Keeble, V. Kotliar, S. Murray, CERN Tape Archive: a distributed, reliable and scalable scheduling system, same conference, (2021)
19. M. Barisits et al., ATLAS Data Carousel, EPJ Web of Conferences **245**, 04035(2020), https://doi.org/10.1051/epjconf/202024504035
20. L. Morschel, Improving Tape Restore Request Scheduling in the Storage System dCache, Bachelor Theses, FH Wedel, (2020), doi:10.3204/PUBDB-2020-01394