EPJ Web of Conferences **214**, 04048 (2019) https://doi.org/10.1051/epjconf/201921404048
CHEP 2018

dCache - joining the noWORM storage club.

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. For over a decade, dCache.ORG has provided robust software, called dCache, that is used at more than 80 universities and research institutes around the world, allowing these sites to provide reliable storage services for the WLCG experiments and many other scientific communities. The flexible architecture of dCache allows running it in a wide variety of configurations and platforms - from all-in-one Raspberry-Pi up to hundreds of nodes in multi-petabyte infrastructures. The life cycle of scientific data is well defined - collected, processed, archived and finally deleted, when it's not needed anymore. Moreover, during all those stages the data is never modified: either the original data is used, or new derived data is produced. With this knowledge, dCache was designed to handle immutable files as efficiently as possible. Data replication, HSM connectivity and data-server independent operations are only possible due to the immutable nature of stored data. Nowadays many commercial vendors provide such write-once-read-many or WORM storage systems, as they become more and more demanded with grown demand of audio, photo and video content in the web. On the other hand by providing standard NFSv4.1 interface dCache is often used as a general-purpose file-system, especially by new communities, like photon scientists or microbiologists. Although many users are aware of data immutability, some applications and use cases still require in-place updates of stored files. To satisfy new requirements some fundamental changes have to be applied to dCache's core design. However, new developments must not compromise any aspect of existing functionality. In this presentation we will show new developments in dCache to turn it into a regular file system. We will discuss the challenges to build a distributed storage system, 'life' with POSIX compliance, handling of multiple replicas and backward compatibility by providing WORM and noWORM capabilities within the same storage system.

1 INTRODUCTION

Any data is associated with a life time and a life cycle and scientific data is not an exception. The raw data is collected at experiment's detectors, filtered, indexed converted into different representation formats and finally published or get referenced in scientific papers. However, during all these stages the data itself is never modified - it is either used

1 tigran.mkrtchyan@desy.de

as a source to produce new derived data or is removed if it doesn't fulfill scientific needs. Such data handling pattern is know as Write-Once-Read-Many, or WORM[1] data. The WORM property of the data allows underlying storage systems to easily provide functionality which requires data consistency, like replication or offloading to tertiary storage systems, like tape robots.

Although WORM storage sounds like a requirement for scientific data, in reality it's not the case. Most of the data in the web is immutable: Flickr, iTunes Netflix, Spotify, Youtube and many other services provide immutable content. Moreover, legal organizations require that storage systems guarantee that documents are never modified. As a result, many storage vendors have WORM storage solutions in their portfolio.

2 noWORM problem statement

Although scientific data is never modified, a new generation of experiments has introduced a different way to collect data. For decades high energy physics experiments had a concept of a data-taking run. During a single run, experiment trigger configurations were unchanged and all events during that run were stored in a single file. The duration of a run was defined by the scientists based on the final file size, number of events or changes in experiment conditions. The photon science experiments have a slightly different approach. Instead of data runs there is the concept of beam time, which can spread over multiple days. Instead of events there are so called images, which are stored in container files. HDF5[2] is one of the popular container file formats used. As beam time can spread over multiple days, those container files can be updated to add new images into a set. Moreover, as containers usually have headers which have to be updated as well, adding a new image to the set is not an append only operation. To support data collected by such experiments the underlying storage system must support updating of existing files at a random offset. Obviously, when such a container is frozen, when beam time is over, the experiment workflow assumes that the storage system will not allow any modification to the collected data set.

3 WORM in dCache

As dCache[3] was developed as a scientific data storage the WORM nature of the data is an essential part of its design. By splitting a file's metadata and data dCache uses a unique identifier for each file which is independent from the file's name and location. By using this level of indirection, dCache can store multiple copies of a file, dynamically add or remove new locations and make use of external storage like S3 or tape. Although dCache provides POSIX-like[4] access with dcap, nfs or xrootd protocols, once a file is closed it can be opened for reading only. In addition to standard file attributes like size, creation time and ownership, dCache stores the file's checksum, which is used for periodic consistency scans and integrity check when a file is moved between different storage nodes or when a file is restored from external location.

Changing this fundamental behavior has a deep impact on the way dCache is deployed and operates today:

- The unique file ids may point to different data.
- Multiple replicas must be kept in sync.
- External tape copy must be kept in sync.
- File's checksum can't be used for data integrity validation.

Moreover, POSIX compliance adds additional requirements to noWORM storage systems:

- Multi-writer support: all updates issued by one client must be visible to all other clients independent from the access protocol.
- Byte-range locking; an advisory and mandatory file locking mechanisms.

4 Quality of (Storage) Service

Not all data require the same capabilities from the storage system. Obviously, storage of unique raw experiment data must be very reliable. However, derived data can be stored in a less reliable way (since it can be reproduced) and, as a result, needs less expensive storage. These storage capabilities are called Quality on Service, or QoS[5].

dCache and other storage systems used by the HEP community support only two QoS types - disk and/or tape. This means that files can reside on disk storage, to provide desired access latency, on tape, which has a higher reliability, or a combination of disk and tapes. However, QoS, for storage, is not limited to access latency and reliability. One of the possible QoS types can be a retention policy (here is an unfortunate name clash with SRM[6] spec which uses the term 'retention policy' in a slightly different context).

To reflect a file's mutability change we have decided to bind such behavior change to a QoS change. A new QoS type WORM is introduced which can be applied to existing and newly created files. For files with noWORM QoS additional restrictions apply - they can't be migrated to external storage systems, i.e. such files have a disk-only policy. However, after QoS transition to WORM, the data can be migrated to tape storage, if desired. This behavior matches with the experiment's workflow described earlier: after the container of the data set is declared as 'closed' it becomes immutable and can be archived if needed.

5 Data replication and mirroring

Although noWORM QoS constrains data to disk-only storage we still must guarantee high data reliability. One of dCache's core functionalities is data replication. The newly written data is replicated to a new location based on a predefined policy. However such a replication model has it's limitations. First of all, this replication happens in an asynchronous fashion. This means that the required replicas are created after the client finishes the transfers and there are small time windows, during which only a single copy of the data exists. Secondly, while a single copy of a file is being modified other copies of the same file can be accessed and modified by other clients, thus having multiple copies with different content. Moreover, when some data nodes are unavailable during updates older versions of the files are being offered when back online again.

To address the data replication challenge we use client side replication, known as mirroring. When client side mirroring is used, the client's application is responsible for creating all replicas required by the server. All updates must be sent to all copies of the files. This guarantees that all other clients will see the same data, independent of the data server they are talking to. To ensure strong data consistency byte-range file locking can be used.

Obviously, to use client-side replication a client with such functionality should be used. One of the access protocols that dCache supports is NFSv4.1/pNFS, defined in rfc5661[7]. The pNFS extension to NFSv4.1 defines the semantics for accessing data which is spread across multiple data servers in a distributed NFS server environment. Although clients see a single-rooted file system, the access protocol to distributed data is defined by so called pnfs layout types. Moreover, NFSv4.1 specification allows new layout types to be defined in addition to standard block-,object- and file-layouts.

One of such 'new' layout types is the flex-files layout type, defined in rfc8435[8]. As the flex file layout type supports data server mirroring, it became an obvious choice for dCache to offer flex file layout type for NFS clients to use.

Although the flex file layout was considered to be the recommended standard in August 2018, it's already supported by modern Linux distributions as well as RedHat Enterprise

Linux 7.5 [9] (kernel-3.10.0-862.el7). The dCache project as one of the early adopters offers flex file layout support starting version 3.0[10] (Nov 2016).

5.1 Compatibility with non NFS clients

Over many years one of the strong sides of dCache is that full functionality is available to all supported protocols. This should not be an exception for noWORM QoS support. However, as other protocols don't support client-side mirroring, we have to allow a small compromise. Any update by such clients for existing files will be rejected. An initial upload of new files is allowed, though. To satisfy the desired level or redundancy a usual replication mechanism is used, e.g. new copies will be created asynchronously. During a period when not all replicas are available, updates by NFS clients will be rejected as well, however, as long as any of the replicas are still online, the data can be accessed for read.

6 Current status

The dCache team works hard to make this functionality available as a part of the standard dCache version. The current working prototype provides required functionality and ability to change file's WORM/noWORM quality of service. A parallel development has introduced mirroring on write. To guarantee stability of the implementations we use xfstests test suite. Obviously, our current implementations don't pass all tests. As not all test cases can be applied to NFS storage we are working to identify relevant test cases and document cases which will not be supported by dCache. Changing dCache's fundamental design paradigm requires major architectural changes. To minimize the impact of new changes and ease of testing, the new functionalities are added into main dCache code base as a series of incremental changes.

7 Summary

With the growing demand to support mutable files for modern experiments the dCache team has added a new quality of service type to control WORM capability. To provide data reliability, client side mirroring is used to guarantee required data protection. As not all protocols can update multiple copies of the files dCache exposed noWORM functionality to NFSv4.1 clients with flex file layout type, which is supported by modern Linux clients.

8 References

1. https://en.wikipedia.org/wiki/Write_once_read_many [accessed 2019-02-24]
2. The HDF Group, HDF5 format. Version 1.10.4. Available from https://portal.hdfgroup.org/display/HDF5/HDF5 [accessed 2019-02-24]
3. https://dcache.org/ [accessed 2019-02-24]
4. https://en.wikipedia.org/wiki/POSIX [accessed 2019-02-24]
5. P. Millar *et al*., Journal of Physics: Conf. Series **898** (2017), DOI:10.1088/1742-6596/898/6/062043
6. https://sdm.lbl.gov/srm-wg/index.html [accessed 2019-02-24]
7. Shepler S., Eisler M. Noveck D. "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 2019-02-24]

8. Halevy B, Haynes T, "Parallel NFS (pNFS) Flexible File Layout ", RFC 8435, 10.17487/RFC8435 (2018), Avaliable from https://tools.ietf.org/rfc/rfc8435.txt [accessed 2019-02-24]
9. https://www.redhat.com/en/technologies/linux-platforms/enterprise-linux
10. dCache Project, dCache [software], Version 3.0.0. Available from https://github.com/dCache/dcache/releases/tag/3.0.0 [accessed 2019-02-24]