Transparent handling of small files with dCache to optimize tape access

Karsten Schwank, Dirk Krücker, Patrick Fuhrmann, Tigran Mkrtchyan, Paul Millar, Christian Bernardt (DESY, Hamburg, Germany), Dmitry Litventsev, Albert Rossi (FNAL), Gerd Behrmann (NDGF)
Deutsches Elektronensynchrotron DESY, Notkestrasse 85, 22607 Hamburg, Germany
E-mail: support@dcache.org

Abstract. We introduce a service for dCache that allows automatic and transparent packing and extracting of file sets into large container files. This is motivated by the adoption of dCache by new communities that often work with rather small data files which causes large time overheads on reads and writes by the underlying tape systems. Our service can be attached to existing dCache instances and we were show that it is able to optimize average access times to tape access.

1. Introduction
Increasingly, sites are using dCache to support communities that have different requirements from WLCG; as an example, DESY facilities and services now support photon sciences and biology groups. This presents new use-cases for dCache. Of particular interest is the chaotic file size distribution with a peak towards small files. This is problematic because tertiary storage systems, and tape in particular, are optimized for storing large files. Direct storing of the user files results in unacceptably poor performance. As dCache bridges the filesystem view with the underlying storage and manages transitions between media, it is the natural place to solve the poor performance from storing small files on tape. We achieved this by introducing a new service that reconciles user demand with tape behavior. The service is transparent to the users and packs files into containers (currently zip files) based on configurable policies. These containers are written directly into the same dCache, which then stores them on tape. Both the small files and the container files benefit from dCache features, such as caching and load-balancing. No additional storage is necessary and the service itself scales by running multiple instances within the same dCache, sharing the load. We describe the design and report on DESY’s experience of running the service over the past six months.

2. Implementation
dCache stores files on disk pools. These disk pools can be configured to store files on tape. It is configured by assigning a set of tags to dCache directories. Dependent on these tags, dCache executes external commands. Usually these commands are scripts that write a file to an attached tape system and return the bfdid identifier to dCache. We make use of this mechanism, but instead of writing the file to tape we mark the file as to be handled by the Small Files Service.
Once the Small File Service has finished packing the file, the same mechanism is used to retrieve a special identifier and return it to dCache.

The tagging of a file happens by the creation of an entry in a MongoDB database, consisting of the files dCache id (i.e., the PnfsId). The entries are then picked up by the first of three scripts the service consists of. It enriches the entries with more file metadata (e.g., file path, size). This metadata is then used by the second script to create sets of files to be stored in a container file. After a container file has been created, a metadata entry is created in the database. This entry is then used by the third script to finalize the file entries by adding the special identifier describing the connection between files and containers.

After this identifier has been read by the script on the pool, the information is stored in the central dCache metadata database and the entry is removed from the MongoDB.

The implementation ensures a minimal amount of metadata requests towards the dCache database. Every step in the process is secured against failures to be able to recover from any intermediate step.

3. Deployment
For our test cases we deployed the packing scripts and the MongoDB on strong multi-core, dedicated machines with 1Gbit Ethernet interface to the connected dCache instance. This setup has proved to work very well and allows running and managing the service independently of the connected dCache instance.

In this setup each involved pool is configured to execute the special script that will create the tags for each small file in the remote MongoDB database on the dedicated machine. On the same machine we have the pack-system service consisting of the previously mentioned three scripts, usually located in /usr/local/bin and the start-stop-Script to control the service.

4. Performance
We made different tests to investigate the performance properties of the service.
Figure 2. Reading files of different sizes from container files shows a constant overhead of ~0.2s per file compared to reading unpacked files

5. Use Cases
The initial motivation for the development of the service was the outlook of dCache being used by the photon science community, with a large amount of small files, but other communities have since then shown interest.

The DPHEP-Group at DESY has performed multiple experiments to use the service for preserving experiment data from the HERA accelerator [2].

6. Summary and Outlook
The service has proven to be applicable to many different use-cases and has been proven to greatly improve the average reading and writing time for small files on tape. We are currently in the process of facilitating the installation process. As a next development iteration we will investigate, how the service can be integrated into dCache as a regular dCache service.

References
[1] http://www.dcache.org
[2] Krücker D. et al. 2015 Data preservation for the HERA experiments at DESY using dCache technology, proceedings CHEP-2015, Okinawa