Quota management in dCache or making a perfectly normal file system normal

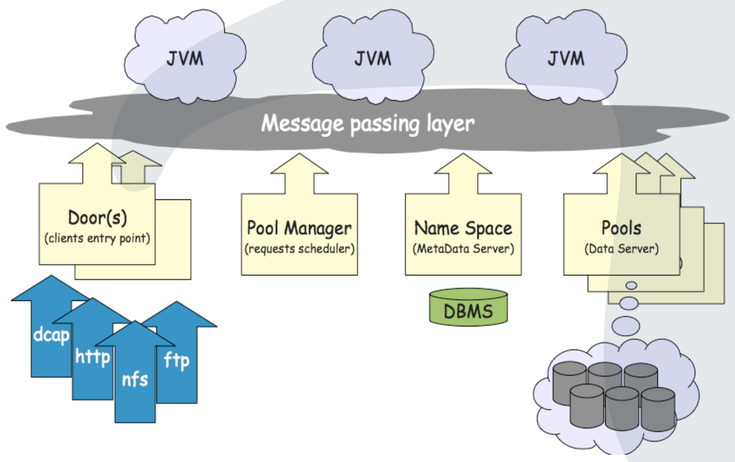
Authors: DMITRY LITVINTSEV, Albert Rossi(FNAL); Svenja Meyer, Paul Millar, Tigran Mkrtchyan, Lea Morschel, Marina Sahakyan (DESY); Krishnaveni Chitrapu (NSC)

Motivation

From its inception dCache was designed as a caching disk buffer to a tertiary tape storage system with an assumption that the latter has virtually unlimited capacity. No provisions for storge quota were made. In disk-only (or hybrid) configuration the need to manage space usage by users and groups has come to the fore. Here we present details of implementation of user and group quotas in dCache.

dCache in a nutshell

dCache is architected as an ensemble of microservices implemented in Java and communicating with each other by messages over TPC/IP network. Each microservice performs specific functions and together they act in concert to deliver data to/from clients over the network.



The core components of the system:

- I/O portals, or doors that implement a specific protocol - (G)FTP, WebDAV, NFS, XRootD etc.
- Namespace (aka PnfsManager) that uses database back-end to store file metadata, directory structure and file locations.
- PoolManager that selects destination (or source) data server based various selection criteria and pool cost function.
- Pool(s) – data severs that store data in complete files (called file replicas) on data partitions deliver data to/from clients and manager file repository cache. File replicas on pools play roles of inodes to dCache namespace entries.

On upload a client connects to a door to initiate a transfer to a destination path in dCache namespace. The door sends a message to the namespace server to check permissions in the destination directory, receives reply. If OK, a namespace entry is created and the door messages PoolManager to select a pool. The PoolManager selects the pool based on pool cost matrix and sends message to the selected pool to start a mover. The pool starts the mover, replies to PoolManager with mover socket address. PoolManager replies to the door with message containing the mover address. The door replies to the client with the mover address and client writes data to that socket.

Following WLCG SRM specification the lifecycle of file replicas in dCache is determined by their Access Latency (AL) and Retention Policy (RP). ONLINE/REPLICA files have disk-only replicas that are always in cache. NEARLINE/CUSTODIAL files have at least one replica stored on tape. Once stored on tape the cached replicas are subject to cache expungement based on LRU access time.

Requirements

Objects in dCache namespace are owned by users identified by UIDs and GIDs. UID and GID of dCache user is determined by mapping of credentials presented by the client to dCache authn/authz service.

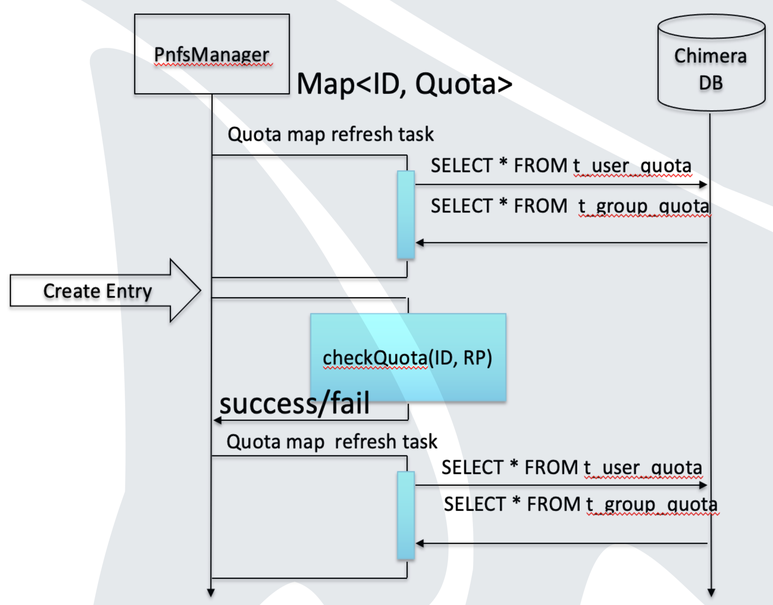
The requirements to storage quota system reflect budgetary and fair use concerns. Experiments requested to be able to:

- Control amount of data written to tape by a user/group.
- Control amount of disk-only space that a user/group can use.
- Quota check and aggregation must not slow down per file namespace operations.

Implementation

A quota handler has been added to PnfsManager service. It implements put/get/modify/remove functions that allow to manipulate user and group quota limits on a data store which is part of namespace database scheme. It functions as follows:

- Schedules periodic scans of all namespace entries to aggregate file sizes by UID, GID and Retention Policy (RP).
  - Updates underlying data store with the resulting usage data.
  - Maintains memory cache of aggregated data as `map<id, Quota>`. The `Quota` data structure contains space usage numbers and their limits for RP.({REPLICA, CUSTODIAL, OUTPUT}).
- When PnfsManager executes create entry function, the quota handler checks if used space for a given UID, GID and RP does not exceed a limit. If it does, the create entry call fails with “Quota exceeded” message.



The periodic collection of usage data has been chosen to avoid adverse impact on the performance of per file operations that would be especially noticeable when performing bulk operations interactively (e.g. `rm *`) if the usage space were to be re-calculated on each create/remove. The downside of this solution is that the quota usage numbers are always lagging behind. The severity of the lag can be mitigated by adjusting the frequency of the quota aggregation scans.

If user goes over quota the effect is not immediate and a user will still be able to write until next scan completes and usage numbers update. Conversely, once over quota, removal of the data will not enable the user to resume writing until next scan has completed.

Admin and user interface

PnfsManager admin interface has been augmented with remove/show/set commands allowing admin to manipulate user and group quota limits.

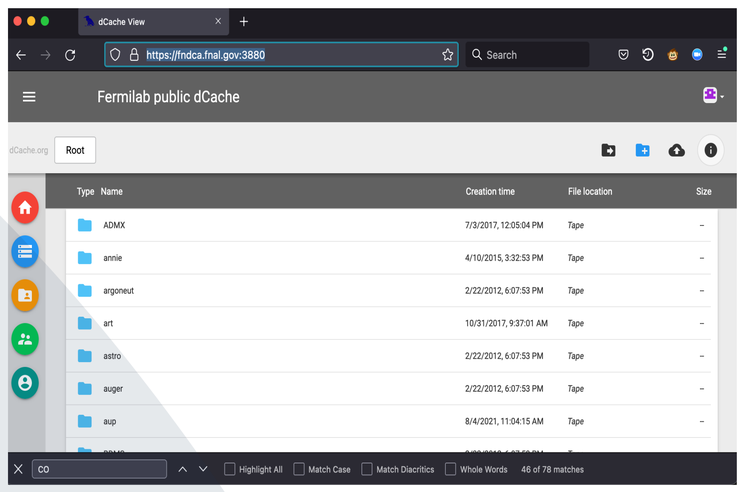
REST API

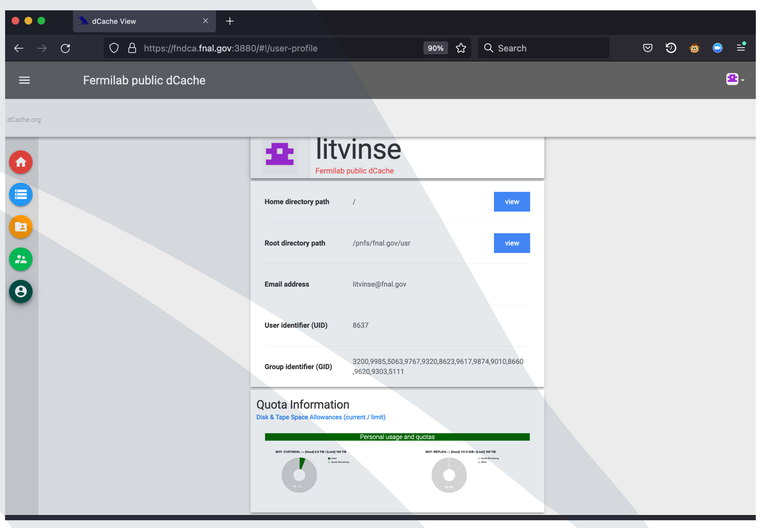
Users and admins can interact with quota handler using dCache REST API

(Consult https://dcache.org:3880/api/v1 for example usage and documentation):

dCache View

https://dcache.org:3880





Conclusion

We have implemented storage quota in dCache allowing to control how much disk-only and tape resident data a user/group can store in the system.

Multi-user, multi-VO installations are expected to benefit from this feature.

The quota feature has been rolled out in version 7.2 of dCache where it was not enabled by default. In version 8.2 it is enabled by default.

FERMILAB-POSTER-23-033-CSAID