dCache and QoS

Paul Millar
(on behalf of the dCache team)

DOMA-QoS workshop 2020 at CERN; 2020-02-07
https://indico.cern.ch/event/873367/

QoS REST support in dCache

- Support for controlling a file's QoS through the **REST API**
  - One file may be changed at a time.
  - INDICO-DataCloud CDMI interface has a plugin to drive this REST API.
  - Elements of the REST API have been implemented by EOS, allowing the same plugin to target either dCache or EOS.
- The REST API allows the client to **introspect** available QoS classes and metadata about them.
- dCache supports **three QoS classes**: DISK, TAPE and DISK_AND_TAPE

Bulk operations

Improving REST API to support bulk operations

- One benefit of SRM is a single request can target **multiple files**.
- This is currently not possible with dCache's REST API.

  Identified as a significant limitation by euXFEL users.
- We are adding bulk operations.
- API is documented here:

  https://docs.google.com/document/d/14sdrRmJts5JYBFKSvedKCxT1tcrWtWchR-PJhxdunT8/edit?usp=sharing

  The API is not fixed and may change as we gain experience.

Bulk operations: creating a request

- Make a bulk request by issuing a POST operation to the endpoint:

```
curl -X POST d '{...}' https://dcache.example.org/api/v1/bulk-requests
```

- The response contains the location of a URL representing this request.
- There will be some authorisation: not all users may make bulk operations.
- Clients may have multiple requests concurrently.

  But there will be a limit.

Bulk operations: interacting with bulk operations

- Clients can use the returned URL to **interact** with the request once they are made:
  - A **`GET`** request to learn the current status,
  - A **`PATCH`** request to cancel the bulk operation,
  - A **`DELETE`** request to clear the bulk operation (canceling if not already finished).
- Currently the API requires clients to **poll** for the current status.

  Subsequent update will add support for storage events.

Bulk operations: targets and activity

- Client has different options when **specifying targets**:
  - The request could target a list of files (SRM style),
  - One (or more) directories targeting the immediate children,
  - One (or more) directories with full recursion.
- Various **activities** will be supported:
  - A single bulk-operation request has a **single activity**.
  - **Deleting**, **pinning** and **changing QoS** are three examples of activities.

Storage Events

Existing support for QoS storage events

- **In Kafka**, events are generated when a pool writes data to tape, and reads data back from tape.
- **In SSE**, the inotify support will generate the IN_ATTRIB event whenever the QoS changes.
- Problems with these approaches:
  - Kafka events require site deployment and configuration.
  - Inotify targets individual directories (not recursive) and, in general, contains no metadata. IN_ATTRIB says "something" about a file has changed, which might be the file's QoS.

Improvements for storage events QoS

- Add **QoS notification** (via SSE) to describe QoS events.
  - Would potentially see QoS events of all files, not limited to a directory.
  - Contain metadata describing what just happened.
- Add **data loss notification** (via SSE and Kafka).
  - Allow Rucio to learn when data is lost
  - Reduce the operational cost of using unreliable/opportunistic storage.
  - Already possible with weird configuration, publishing to Kafka.
- Add **bulk-request notification**.
  - Monitor the progress without polling.

QoS classes

Existing support for different QoS classes:

- Currently can bind different pools to different portions of the namespace. These pools can have different QoS
  - For example, configure some pools to store data on SSD and bind these pools to the **`/data/ssd`** directory.
  - Similar to how dCache may be configured so data written into **`/data/tape`** is written to tape.
- **QoS transitions** would involve an external agent (Rucio + FTS) copying data from one portion of the namespace to another.

  Optionally, can delete the old file once the new file is created.
- Exploring this option through ESCAPE project, as a proof-of-principle.

Providing richer per-file QoS choice

- Currently can choose between DISK and TAPE
- We will extend the per-file QoS choices.
- Use pool tags (key-value pairs) to define media and other characteristics.
- Use admin-defined policies to drive data locality (within dCache).
- Open question whether QoS classes may be composed:

  QoS DISK and QoS TAPE vs QoS DISK_AND_TAPE

Conclusions

- Many things are already possible, but with weird configuration.

  Working to make weird configuration main-stream, and put these choices in the hands of users.

- Bulk REST API to allow bulk operation transitions to scale.
- Storage events to avoid polling.
- Improving single-file QoS options.

Thanks for listening!