**QoS Rule Engine and Policy Management**

This document is simply meant to “prime the pump.”

The easiest way to do this for me is just to lay out my own inchoate thoughts and let the margins fill up with comments from you drawn from more concrete and connected user contexts.

Here’s what occurs to me might be a way to go.

1. The QoS Engine uploads qos policy template files. These files can be arbitrarily many, based on any number of schemes (standard types, experiment-specific types, etc.). The template could have something like the following JSON representation:

**{**

**"name": "DUNE",**

**"states": [**

**{**

**"duration": "30\*24\*60\*60", // (i.e., 30 days in seconds)**

**"media": [**

**{**

**"type": "fast",**

**"count": 3,**

**"partitions": ["hostname"]**

**},**

**{**

**"type": "slow",**

**"count": 1**

**}**

**]**

**},**

**{**

**"duration": "90\*24\*60\*60", (i.e., 90 days in seconds)**

**"media": [**

**{**

**"type": "slow",**

**"count": 1**

**},**

**{**

**"type": "CTA",**

**"count": 1**

**}**

**]**

**},**

**{**

**"duration": "-1", // indefinite**

**"media": [**

**{**

**"type": "CTA",**

**"count": 1**

**}**

**]**

**}**

**]**

**}**

It should be possible to add, remove and modify policy templates via a RESTful API. One could also do this manually by placing JSON files in the qos engine home dir.

2. The template should be used to create a profile instance for an individual file.

Where do we store this? in a specific DB, or should chimera be modified to record this information for each file inode?

3. We now use a single directory tag to define the file’s initial policy, e.g.:

QOS\_POLICY DUNE

4. QoS transitions are currently limited only to disk, tape and disk+tape. We can expand the RESTful API to accommodate the extended semantics. The most flexible way of doing this would be to expose the entire “states” array:

request submit -activity=UPDATE-QOS -expand-all -arguments=target-qos:{"duration": "90\*24\*60\*60","media":[{"type": "slow", "count": 1},{"type": "CTA","count": 1}]},{"duration": "-1","media":[{"type": "CTA","count": 1}]} /pnfs/fs/usr/dune/data/2021-04-22

It may be useful also to provide a push version where you can push a state onto the stack:

request submit -activity=UPDATE-QOS -expand-all -arguments=target-qos:{"duration": "90\*24\*60\*60","media":[{"type": "slow", "count": 1},{"type": "CTA","count": 1}]}]},push:true /pnfs/fs/usr/dune/data/2021-04-22

Or one could also provide append to alter the final state(s):

request submit -activity=UPDATE-QOS -expand-all -arguments=target-qos:{"duration": "90\*24\*60\*60","media":[{"type": "slow", "count": 1},{"type": "CTA","count": 1}]}]},append:true /pnfs/fs/usr/dune/data/2021-04-22

Or one could provide “set”, “insert” and “remove” options which take an index. It would not be difficult to expose all of these through the API.

We can also add a new QoS activity, SET-QOS, which would just globally reset the file’s policy to a templated type:

request submit -activity=SET-QOS -expand=all -arguments=policy:DUNE /pnfs/fs/usr/dune/data/2021-04-22

Obviously, any modification means the engine then recomputes the states queue and restarts from the first state.

5. We can continue to support AL/RP + storage-unit definitions, but I would suggest these be integrated/coordinated by the qos-engine into the file profile.

The most important design decision here is how to store the individual file qos profile. I would think the advantage of putting it in chimera is data consistency (all in one database instance). The disadvantage is of course more query pressure on chimera.

6. Aside from managing the templates and setting the file policy on new cache locations, the engine will also run a scheduled sweep of policies; expired policies will be popped off the stack, and trigger a verification request to the verifier which will check the file against the requirements of the successive state. The period for such sweeps can be similar to or shorter than the pool scans, according to need. However, since the sweep will be looking at every file, it can take a good while (though not as long as pool scan adjustments, since the verification request will be fire-and-forget). It will undoubtedly be necessary to batch the query in some way, though how to do this is currently unclear to me (we don’t want to do it like a pool scan by location, because we would be doing redundant checking on a good number of files).