QoS/DLC toy model: a proposal

Paul Millar

INDIGO-DataCloud WP4 virtual meeting (2015-09-02)

The “toy model” is...

- a high-level **protocol description** of how users interact with storage,
- a **framework** for defining QoS/DLC terms,
- a **starting point** for actual network protocol discussion,
- concrete enough that people can **criticise it**,
- stimulate generation of **open questions**.

Why is the toy model necessary?

- Cannot define terms in isolation:

  There's always some interaction model: let's make it explicit.

- We want a "reality check"

  Can we describe Amazon S3, Google Cloud Storage, WLCG Tier-1 and Tier-2, ...?

QoS attributes

- Some characteristic of the storage service when offering this QoS.
- Something that the service provider "promises" to deliver.

    usually backed by an MoU or SLA

- The differences between the available QoS options are explainable through different QoS attribute values.

Attributes as dimensions

- One view of QoS is to define each possible description as an axis in some *n*-dimensional QoS space.
- Attributes can be discrete or continuous:
  - Discrete: only accepting certain values,
  - Continuous: values can be somewhat arbitrary
- This concept is OK, but doesn't really work for users specifying desired QoS...

Specifying desired QoS

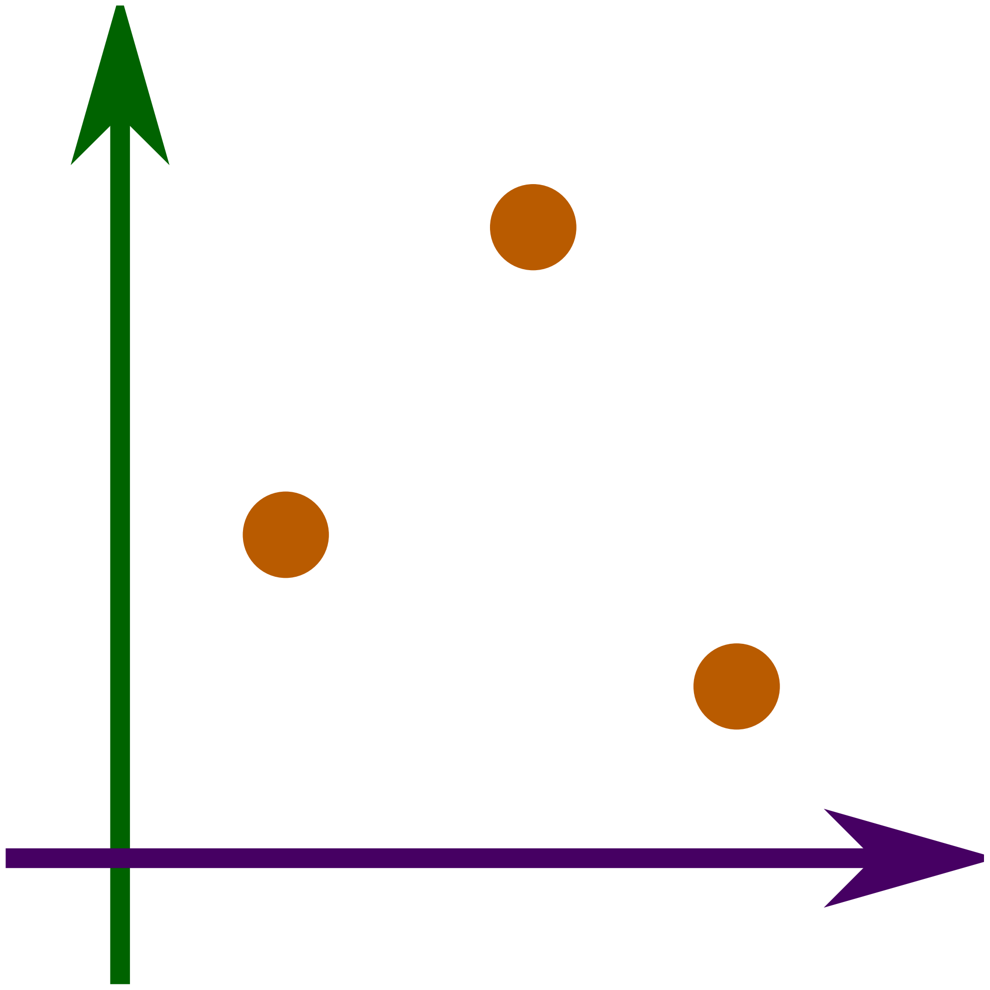
User specifying desired attributes is **awkward**:

- Clients don't know whether changing a value will alter the QoS
- Clients don't know whether there is a “better” QoS.
- Clients could specify too little information
- Clients could specify a conflicting.

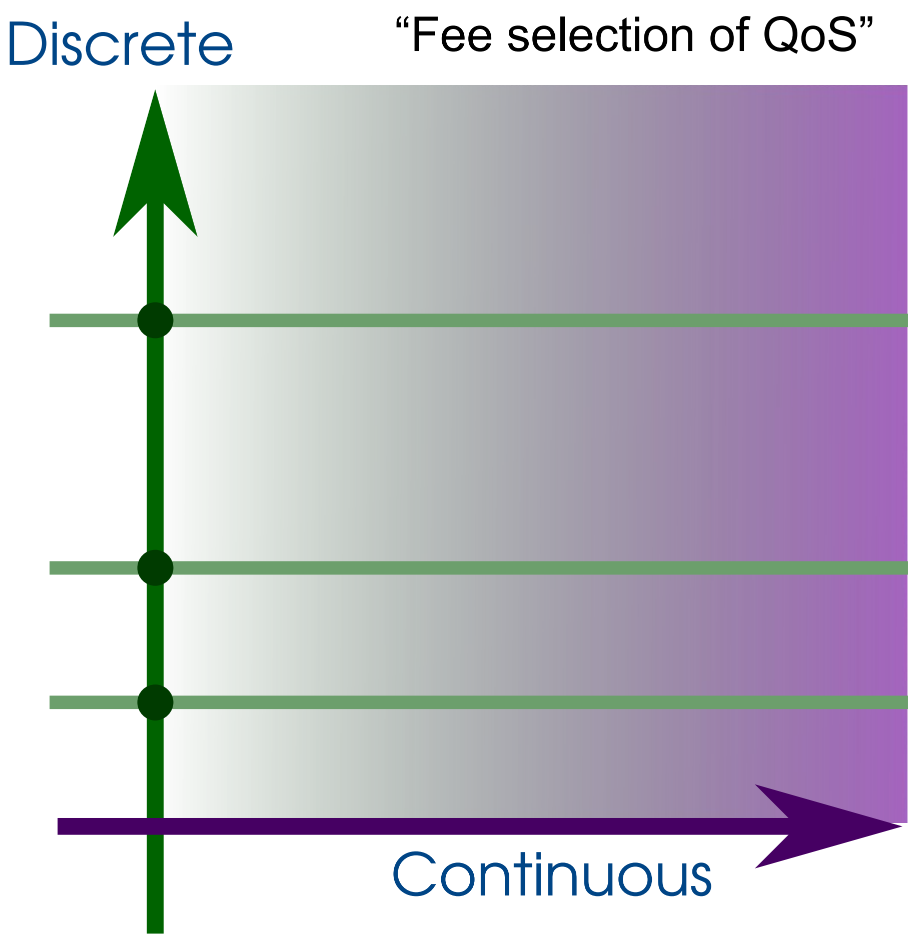
Alternative: list available **islands**

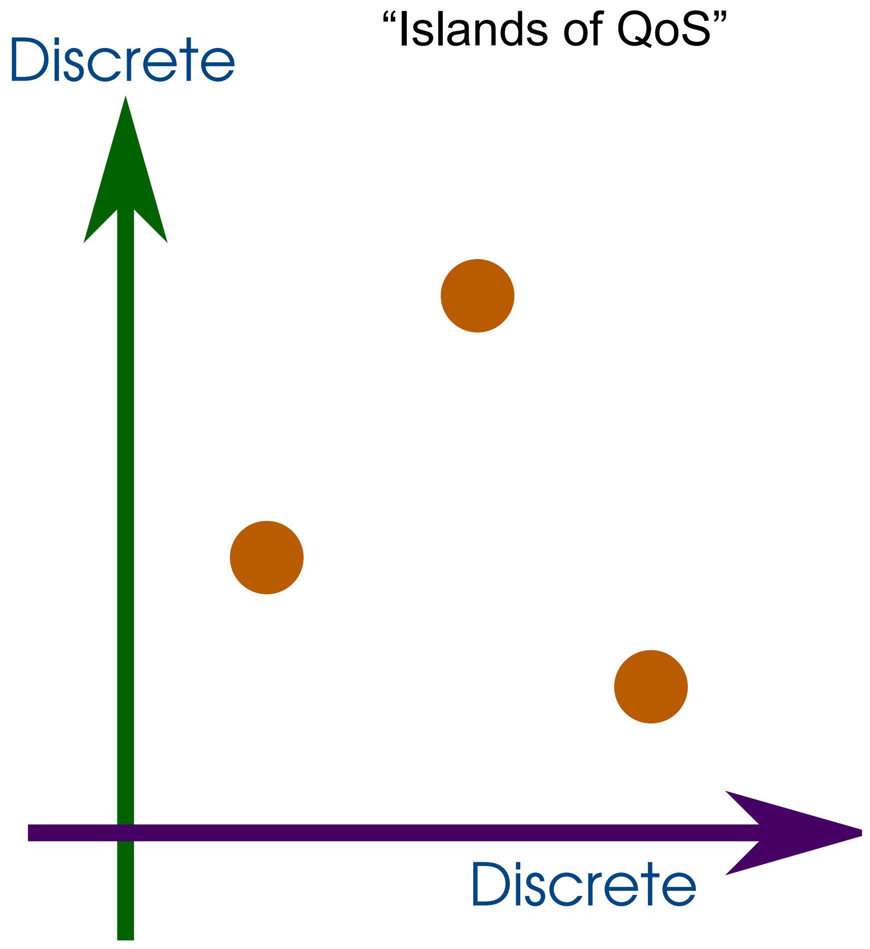
- Client can see what are available options,
- Client can choose exactly what they want,
- If user wants a different value for Attribute-X, she can see the consequence in the other attributes.
- For each QoS-island, user needs sufficient information to choose.

Attributes and islands



Combining QoS attributes





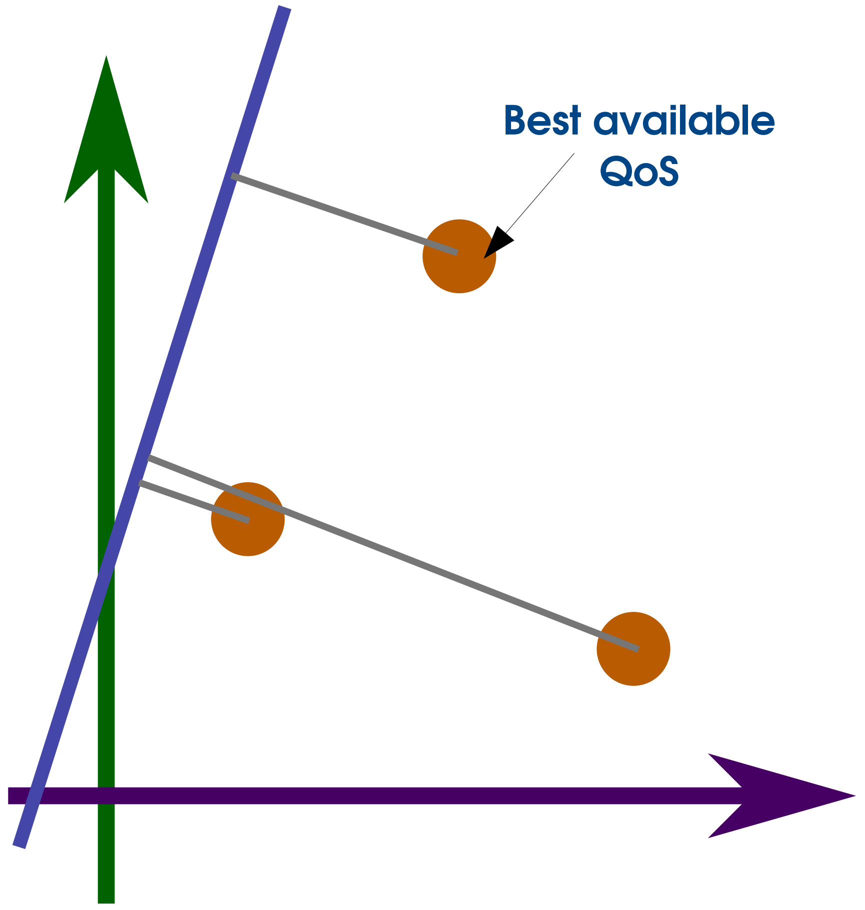
Independent **Dependent**

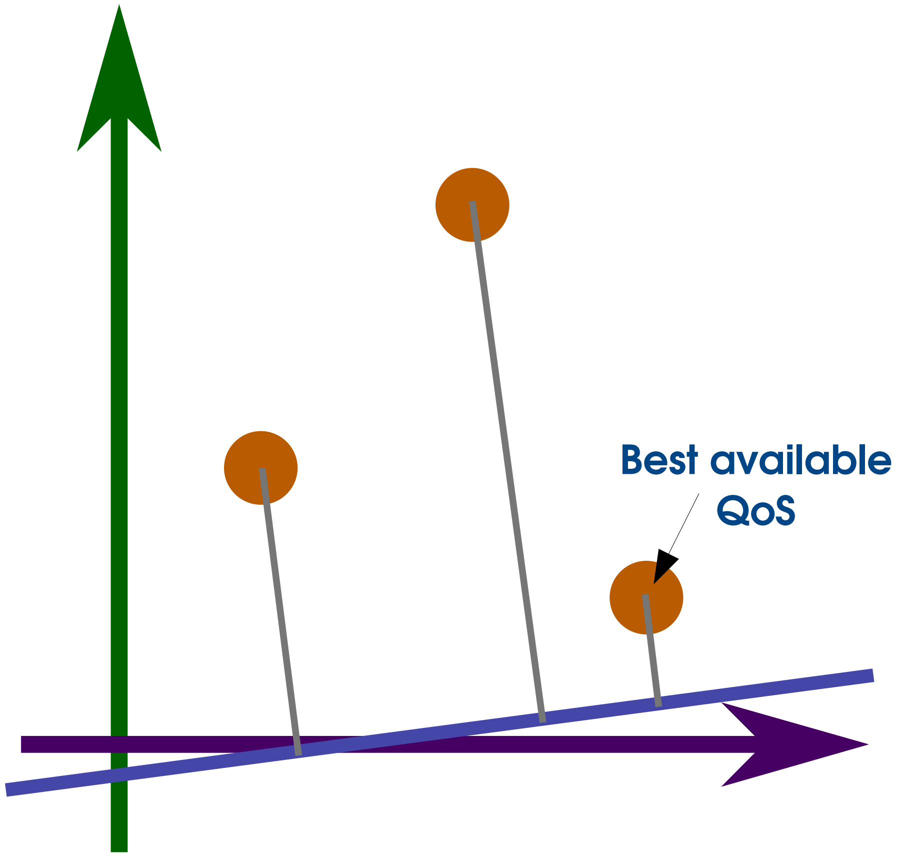
An aside: handling independence

- Example: Google/Amazon locality (ASIA, EU, USA)
- Two approaches:
  - Enforce the Islands-of-QoS view

    Simpler, but risks the combinatory explosion.
  - Allow independent definition: choose an island and allow setting the independent attributes separately.

Figure-of-merit: how users choose





Open issues

- Which attributes are actually useful

    e.g., file replication

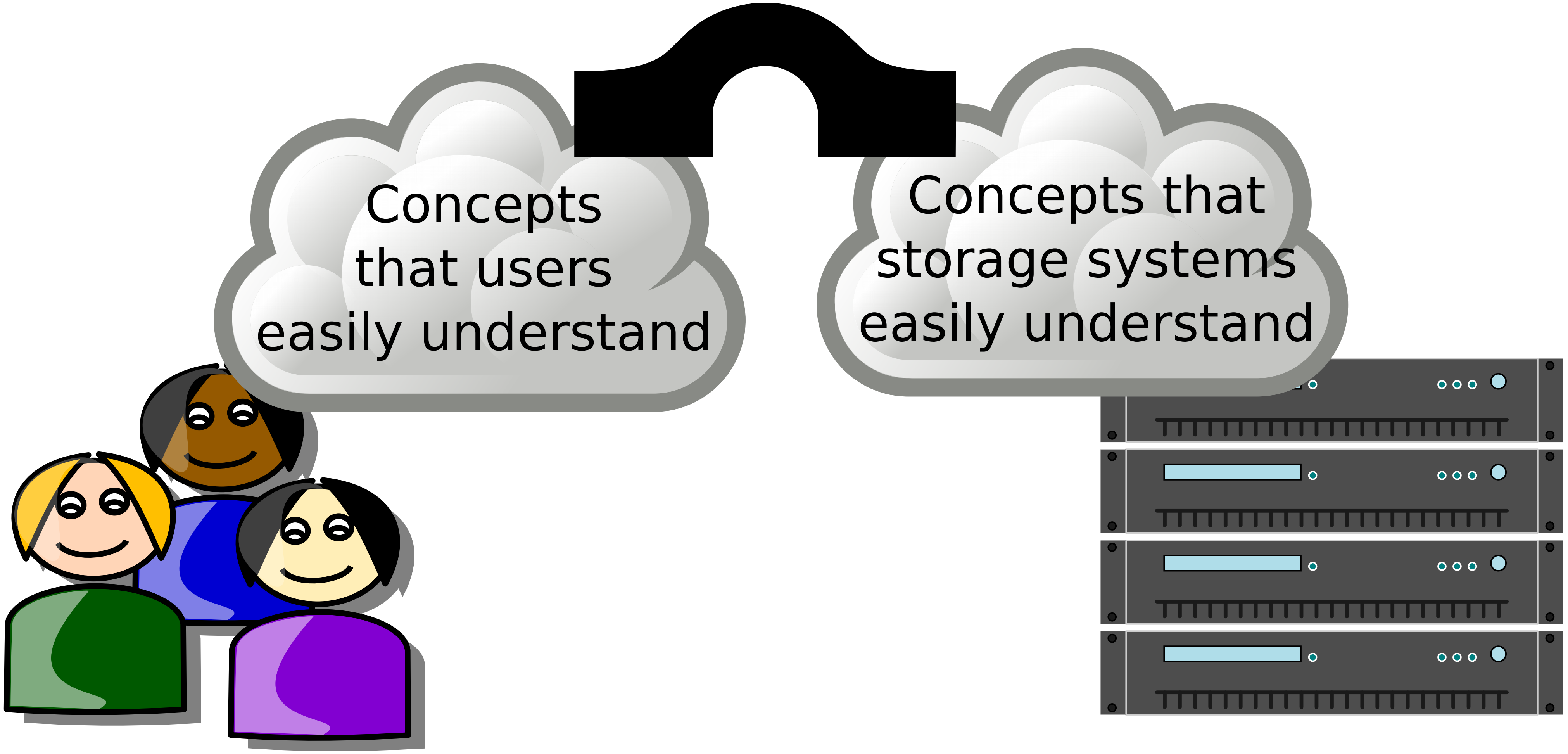
- How about availability and durability?

    Who really can distinguish between a 4x “9”s and a 5x “9”s value?

    How do service providers provide this level of service?

    How does the system know its values?

Bridging the gap


Concepts that users easily understand
Concepts that storage systems easily understand

Handling QoS of datasets

- Almost all users group data into datasets.
- There is no single, universal definition of a dataset.

  Datasets within datasets (subsets)? Files that are members of multiple, independent datasets? Mutability of datasets?

- Here's a model that could work:
  - A **label** is some arbitrary name that has either some QoS definition or is ***not specified***.
  - Each file has a **default QoS**.
  - Each file also has an **ordered list** of (zero or more) labels.
  - There is **last-one-wins** rule for selecting the QoS

    start with the default-QoS, then resolve each label's QoS, skipping any that are "not specified".

Data-LifeCycle

- Usually a fuzzy definition
  - Any operations that are applied from when the data is created to when it is deleted.
- Limit to **autonomous** data-lifecycle:
  - DLC where the storage acts autonomously.
  - Exclude cases where storage only assists in DLC operations.
- However, boundary is somewhat arbitrary:
  - Maintaining a backing up data,
  - Data validation,
  - Integrity policies,
  - Event notification.

Data-LifeCycle format

For each file, the DLC is a list of:

<predicate> <action>

Where:

<predicate> is when something should happen.

<action> is what should happen.

Deciding when something should happen

Define <predicate> as:

- <metric> <comparison> <value>

For example:

- File-age >= “6 months” (or 1.5x10^7 s, or ...)
- File-age >= 10 years
- Last-used >= 1 week

What should happen

- Modify the QoS of a file

    (e.g., move a file from SSD to disk after week of inactivity)
- Modify the ACL of a file

    (e.g., make private data public after 6 months)
- Transfer file into some other storage

    (e.g., copy data into some archive storage)
- Delete file

… other actions ?

Open issue:

- Do we need chaining in DLC actions:

  (e.g., transfer file into archive then delete)

- How to handle DLC assignment in datasets

  (assign DLC to QoS-labels, or is DLC independent to QoS?)

Backup slides