QoS and DLC in IaaS
INDIGO-DataCloud

Presenter: Patrick Fuhrmann

Contributions by:
Giacinto Donvito, INFN
Marcus Hardt, KIT
Paul Millar, DESY
Alvaro Garcia, CSIC
Zdenek Sustr, CESNET

And many more
## Quality of Service based on media

<table>
<thead>
<tr>
<th>Media Quality</th>
<th>Access Latency</th>
<th>Durability</th>
<th>Datarate</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HIGH</td>
<td>OK</td>
<td>OK</td>
<td>Very low</td>
</tr>
<tr>
<td></td>
<td>MEDIUM</td>
<td>MEDIUM</td>
<td>OK</td>
<td>Reasonable</td>
</tr>
<tr>
<td></td>
<td>LOW</td>
<td>Not so clear</td>
<td>MEDIUM</td>
<td>Very high</td>
</tr>
<tr>
<td></td>
<td>MEDIUM</td>
<td>Quite OK</td>
<td>OK</td>
<td>MEDIUM</td>
</tr>
</tbody>
</table>

19/12/2016
Not quite as easy as that

It looks simple, but there are issues.
Is there a sufficiently complete set of properties?

In WCLG we only had two properties:
- Access Latency
- Retention policy

That was already too much for most people 😊

Talking to Reagan Moore (IRODS) at the Paris RDA meeting:
- He is suggesting about 200 properties
- That might be a bit over the top for a start
QoS Property Value Ambiguity

Access Latency
- 1 day
- 1 hour
- 1 ms
- 1 ns

High Ambiguity
- Archive
- Backup
- Streaming
- HPC

Cheapest Fastest
Property dependencies

Durability

Access Latency
Property Quantization

Multi Dimensional Property Quantization

Cost

More Data

S3

Access Latency

Glazier
## Properties zoo of existing systems

<table>
<thead>
<tr>
<th>Service</th>
<th>Type</th>
<th>Description</th>
<th>Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Amazon</strong></td>
<td>S3</td>
<td>Standard</td>
<td></td>
</tr>
<tr>
<td><strong>Google</strong></td>
<td>Standard</td>
<td>Durable, reduces availability</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Correlates to HPSS Classes</td>
<td>(customizable)</td>
<td></td>
</tr>
<tr>
<td><strong>HPSS/GPSS</strong></td>
<td>Resilient</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>dCache</strong></td>
<td>Resilient</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Glacier</td>
<td>Nearline</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TAPE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Time to tidy up!

Starting with the unambiguous technical view, seen by the storage system.

Canonical Properties
### What are canonical properties?

<table>
<thead>
<tr>
<th>Class</th>
<th>Access Latency</th>
<th>Durability</th>
<th>Media</th>
<th>Replicas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class A</td>
<td>&lt; 1 ms</td>
<td>&gt; 0.9999</td>
<td>Disk / SSD</td>
<td>1 Disk</td>
</tr>
<tr>
<td>Class B</td>
<td>&lt; 10 min</td>
<td>0.99999999</td>
<td>Tape</td>
<td>2 Tape</td>
</tr>
<tr>
<td>Class C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Avoiding ambiguities*
How to get those properties out of existing storage systems?
Canonical Storage Properties

Slightly extended Information Provider (internal component)

Storage System
- dCache
- StoRM
- EOS

Canonical Storage Property Information

Storage Access
The **canonical view** only helps to describe the system on the **technical level**.

It’s not very helpful for the storage user.

We need to introduce more convenient **QoS views**.
QoS views

Ambiguous, non canonical, dependent, combined properties.

Examples:

Low latency & lowest price
High throughput & super durable
Large volume & cheap & archive
Therefore: Introducing a new service

Property Class ID

Class = XYZ
For that particular system

Discover and Match

Canonical Storage Property Information

Customer View

COST = Cheapest
Media = TAPE
Translation and discovery
The federated system provides additional QoS properties.

- Number of copies, not in the same location
- Minimum geographic distance for disaster cases. (fire, earthquakes)
- Legal implications: Privacy laws

Federated system might need more higher level services attached:

- FTS or Globus Online to create replicas
- DynaFed to federate distributed resources.
More problems to solve

- How does the client provide the storage class to the storage system?
  - Bucket
  - Directory
  - Additional argument in WebDAV, FTP etc

- The system only provides the class, it doesn’t ‘promise’ the space.
  - Do we need a space reservation protocol?
  - Similar to hotels.com. Check hotel pictures first, reservation only after payment.
  - Is reservation required in systems with unlimited space (Clouds)?

- Do we allow to change the storage class, assuming the system will do the necessary data movements?
  - This is of course just a storage system property.
    - Amazon and Goolge don’t
    - dCache and HPSS do.
Data Life Cycle is just the time dependent change of

- Storage Quality of Service
- Ownership and Access Control (PI Owned, no access, Site Owned, Public access)
- Payment model: Pay as you go; Pay in advance for rest of lifetime.
- Maybe other things

6 m  1 years  10 years
Current status (definitions)

- Introduced at the research data alliance (RDA) in Paris
- Lots of interested communities and sites.
- Creating of interest group in progress.
  - Name still in heavy discussion 😊
- 10 Committed members
- Will be followed up on in Tokyo end of Feb 2016
Current status (technically)

- Canonical Information providers are being build
  - dCache (internal)
  - Common external system for
    - GPFS/HPSS
    - CEPH
    - StoRM/GEMSS
    - Cloud (Amazon and Google)
- Information Provider Protocol in discussion (candidate : CDMI)
SUMMARY

- INDIGO provides funding to standardize QoS and possibly Data Life Cycle of systems.
- Scientific communities are showing great interest in those activities.
- Common definition of QoS is essential for Platform as a Service for storage.
- RDA ‘Interest Group’ being built to get in touch with more communities.
- Prototype systems, including:
  - IBM: GPSS, HPSS
  - Grid storage systems: dCache, StoRM, ...
  - Public Clouds: Amazon, Google
- Prototypes will be provided within the next 12 months.
- Contribution of ideas from your side is more than welcome.
- Contact: Dr. Paul Millar