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

With kind contributions by
Shaun DEWITT, EUDAT
Content

- Introducing INDIGO-DataCloud.
- What is the issue with QoS in Storage?
- Which part are we trying to solve?
- What is our approach?
- **H2020 Project**
  - Approved Jan 2015
  - Started April 2015 – Ends Sep 2017 = 30 months
- 26 European Partners
- 11 European Countries
- > 11 Million Euros
- **Objective**: Develop an Open Source platform for computing and data, deployable on public and private cloud infrastructures.
- Requirements and use-cases collected from 11 INIDIGO communities.
- For further details: [http://indigo-datacloud.eu](http://indigo-datacloud.eu)
INDIGO DataCloud WP structure

Stolen from Alvaro’s, Andrea’s presentation
WP4 in detail

- **Virtualized Computing Resources**
  - Full Container support for Cloud Management Infrastructures and Batch
  - Container support for special hardware (Infiniband, GP-GPU’s)
  - Spot Instances
  - Fair Share Scheduling

- **Virtualized Storage Resources**
  - QoS and Data Life Cycle for storage (storage management)
  - Access to data by meta data instead of name space
  - Dual access to data (Object Store versus POSIX file name space)
  - Identity Harmonization for storage

- **Virtualized Network Resources**
  - Orchestrating local and federated network resources
  - “Software Defined Network” evaluation
  - Services and Appliances for virtual networks
Why QoS and DLC

- EU requires to provide a “Data Management Plan” from all data intensive EU projects.

- Problem:
  - No common way to describe QoS or Data Life Cycle
  - No common way to negotiate QoS with storage endpoints (except for SRM systems 😊)

- Common definitions for QoS would be very convenient in general but inevitable for PaaS layers, as the negotiation resp. brokering is done by engines. (Similar to hotel or flight finders)
Description of Work for WP4

1. Define a common vocabulary for QoS storage properties and their values based on use cases from scientific communities:
   - Involve standardization bodies, e.g. RDA, OGF
2. Define a semantics to negotiate QoS with endpoints
3. Find a real network protocol (prototype or demonstrator) and implement the defined QoS semantics for different systems.
Storage provisioning for large public infrastructures is facing two contradicting problems:

- The complexity of storage and storage management
- The large variety of sciences and their diverging expectations on storage
Infrastructure Problem

- **Infrastructures**
  - Are growing in
    - size of storage and
    - number of supported sciences and communities and
    - Number of direct customers accessing storage
  - They all have different ideas on how to use storage.
  - Serving them in the old fashion doesn’t scale any more
  - So you need an API’s or portals to let them select what they need

- **Infrastructures are used by platforms**, which
  - tend to federated resources from different locations and storage providers.
  - So storage needs to be brokered and procured automatically (or programatically)
Examples for Storage Complexity
## 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>HIGH</td>
<td>MEDIUM</td>
<td>LOW</td>
<td>MEDIUM</td>
<td>MEDIUM</td>
</tr>
<tr>
<td>OK</td>
<td>MEDIUM</td>
<td>Not so clear</td>
<td>Quite OK</td>
<td>OK</td>
</tr>
<tr>
<td>OK</td>
<td>OK</td>
<td>MEDIUM</td>
<td>OK</td>
<td>OK</td>
</tr>
<tr>
<td>Very low</td>
<td>Reasonable</td>
<td>Very high</td>
<td>MEDIUM</td>
<td>MEDIUM</td>
</tr>
</tbody>
</table>
Not quite as easy as that

It looks simple, but there are issues.

Starting with:

a) What are storage properties.
b) What are storage property values.
Access Latency
- How long does it take from the request for a byte to receiving that byte.

Retention Policy
- What is the probability of data loss.

Access Mechanisms
- http, GridFTP, NFS, ....

Security
- encrypted during the transfer, on disk, end – to – end.

Authentication
- SAML, Open ID Connect, Password, X509
How many QoS properties?

- 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
Even more complexity

- QoS Property “Value Ambiguity”
- Property dependencies
- Property Quantization
- Non standard property zoo of existing system
QoS Property  Value Ambiguity

Access Latency

- 1 day
- 1 hour
- 1 ms
- 1 ns

High Ambiguity

- Archive
- Backup
- Streaming
- HPC

Cheapest  Fastest

19/12/2016
Property dependencies

Durability

Access Latency
Multi Dimensional Property Quantization

Cost

More Data

S3

Glacier

Access Latency
## Properties zoo of existing systems

<table>
<thead>
<tr>
<th>Provider</th>
<th>System</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td>S3</td>
<td>Standard</td>
</tr>
<tr>
<td>Google</td>
<td>Standard</td>
<td>Durable Reduces Availability to the HPSS Classes</td>
</tr>
<tr>
<td></td>
<td>HPSS/GPSS</td>
<td>Corresponds to the HPSS Classes (customizable)</td>
</tr>
<tr>
<td></td>
<td>dCache</td>
<td>Resilient</td>
</tr>
<tr>
<td></td>
<td></td>
<td>disk+tape</td>
</tr>
<tr>
<td></td>
<td>Glacier</td>
<td>Nearline</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TAPE</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>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class A</td>
<td>&lt; 1 ms</td>
<td>&lt; 0.9999</td>
<td>Disk, SSD</td>
<td>1 Disk</td>
<td>10 E/m/GB</td>
</tr>
<tr>
<td>Class B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class C</td>
<td></td>
<td></td>
<td>Tape</td>
<td>2 Tape</td>
<td>20 E/m/GB</td>
</tr>
</tbody>
</table>

!!! For EUDAT, those “Classes” are close to their “Services”
So after having defined Canonical Storage Properties and their values ..... How to get them out of existing storage systems ?
Canonical Storage Properties

Canonical Storage Property Information

Storage System
- dCache
- StoRM
- EOS

Storage Access

Slightly extended Information Provider (internal component)
The **canonical view** only helps to describe the system on the **technical level**.

It’s not very helpful for the storage enduser.

We need to introduce more convenient **QoS views**.
Examples on how a user would describe his/her needs

- Low latency & Lowest price
- Highest possible throughput & Short term
- Scratch & Very cheap
- Long Term Storage & Price not important
That's what customers would expect

**Your Magic Storage Wand**

**Basic**

How much storage do you need?

- 100 G
- 1 T
- 10 T
- 100 T
- 1 P
- Dynamic

**Quality**

- Scratch
- Pretty Good
- Rock Solid

**Access**

- WebDAV
- GridFTP
- NFS 4.1 / pNFS

**Euros/Month**

1,05

**Advanced**

**Expert (Extra Costs may apply 😊)**
That's what customers would expect

Your Magic Storage Wand

Basic

Advanced

Expert (Extra Costs may apply 😊)

Media

- Disk
- Tape
- SSD
- Tape Remote

Access Latency: 100 Nano Seconds

Retention: 0.999999 Absolute

Access

- http
- WebDAV
- GridFTP
- NFS 4.1 / pNFS

Security

- X501
- SAML
- Open ID Connect
- Password

Extre

- Attach OID’s
- Support Macaroons

Euros/Month: 1,05
Therefore: Introducing a new service
Discover and Match

Canonical Storage Property Information

Customer View

Optional Properties

COST = Cheapest
MEDIA=Tape
ACCESS=medium

Property Class ID

Class = XYZ
For that particular system

Discover & Match
Translation and discovery

Canonical Storage Property Information

Discover & Match

GUI

Platform Service Or
High level Broker

REST API
Canonical property federation

Platform as a Service

Canonical Storage Property Information System

D&M

GUI

REST API

IaaS

D&M

GUI

REST API
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
To summarize the procedure

- Storage Systems provide a set of ‘classes’ describing standardized storage properties with standardized values.
- Neither the name of the classes nor the combination of properties are standardized, they depend on the storage system.
  - Like S3 and Glacier are the names of the class
- **Matchmaking software** tries to match the various classes to the non standard and site specific requirements of the communities or individuals and returns the closest match to the customer.
- For further requests, the customer will use the ‘class name’ in the request. That could be a directory, a space token or a container.
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.
Current status

- Creating a RDA working group (Paris and Tokyo)
  - Name: Quality of Service and Data Life Cycle Definitions WG
  - Currently agreeing on a Charter.
  - 10 Committed members (sites and communities, Flexier ...)

- Contributing to the SNIA CDMI reference implementation, as this is our planned transport for QoS steering.

- Defined version 1 of RESTFUL API

- Defining a CDMI extension to describe the storage properties and values.

- Implementations are ongoing for dCache, StoRM and the GPFS and TSM pluggins.

INTERESTED?

Paul.millar@desy.de
INDIGO provides funding to standardize QoS and possibly Data Life Cycle of systems.

Scientific communities and EUDAT are showing 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 implementations are in progress (dCache, StoRM, HPSS, ...)

Contribution or ideas from your side are more than welcome.
Further reading

First Proposal for restful representation of our ideas.