EOS XDC Developments
QoS & Converter Engine

Mihai Patrascoiu
CERN / IT- ST- PDS

About XDC

- The eXtreme DataCloud – a 2 year, multi-institute EU-funded software development and integration project started in February 2018
- Goal: improve existing Data Management Services by adding missing functionalities requested by research communities
- Involved@EOS: Oliver Keeble, Andreas Peters
  Mihai Patrascoiu, Fabrizio Furano

XDC website: http://www.extreme-datacloud.eu/

XDC Roadmap

2018

XCache integration
External storage
File adoption

2019

QoS data management
Converter Engine

2020

Final release

2018 developments: EOS Workshop 2019

QoS data management?

- Goal: accommodate different use-cases with storage policies that can achieve the cheapest solution
- Storage policy according to system rules or user-defined
- Implementation brings QoS classes and Converter Engine

Storage policies - examples

- Store files in replica or erasure encoding format
  - Store only files unused for 6 months in EC
    - Store only files unused for 6 months

      and larger than 5GB in EC
- Transition to tape if inactive for # months

QoS classes in EOS

- Abstraction entity over existing storage properties:
  - Discoverable
  - Configurable
  - User applicable on a per file/directory basis
- Maneuvering is done via a QoS API
- Transitioning is supported between QoS classes

How do QoS classes work?

- A QoS class configures the following properties
  - Layout
  - # Stripes
  - Checksum type
  - Placement type
- A QoS class provides guarantees

  E.g.: redundancy level, geolocation
- QoS transitions from one class to another must be explicitly allowed

  E.g: disk $\rightarrow$ tape, tape $\nrightarrow$ disk

How do QoS classes work? (cont'd)

File QoS class deduced at runtime

Extended attribute for mid-transition:
user.eos.qos.target

- QoS class applied to directory → propagates to files assigned in that directory
- Opaque info on Open to specify desired QoS

Structure of a QoS class

- Name
- Transitions : [qos_class, qos_class, …]
- Metadata: { expected_redundancy,
  expected_latency,
  expected_geolocation: [geotag, …] }
- Attributes: { layout_type,
  nstripes,
  checksum_type,
  placement_type }

Mandatory fields

QoS property map

EOS specific

Structure compatible with INDIGO CDMI QoS specification

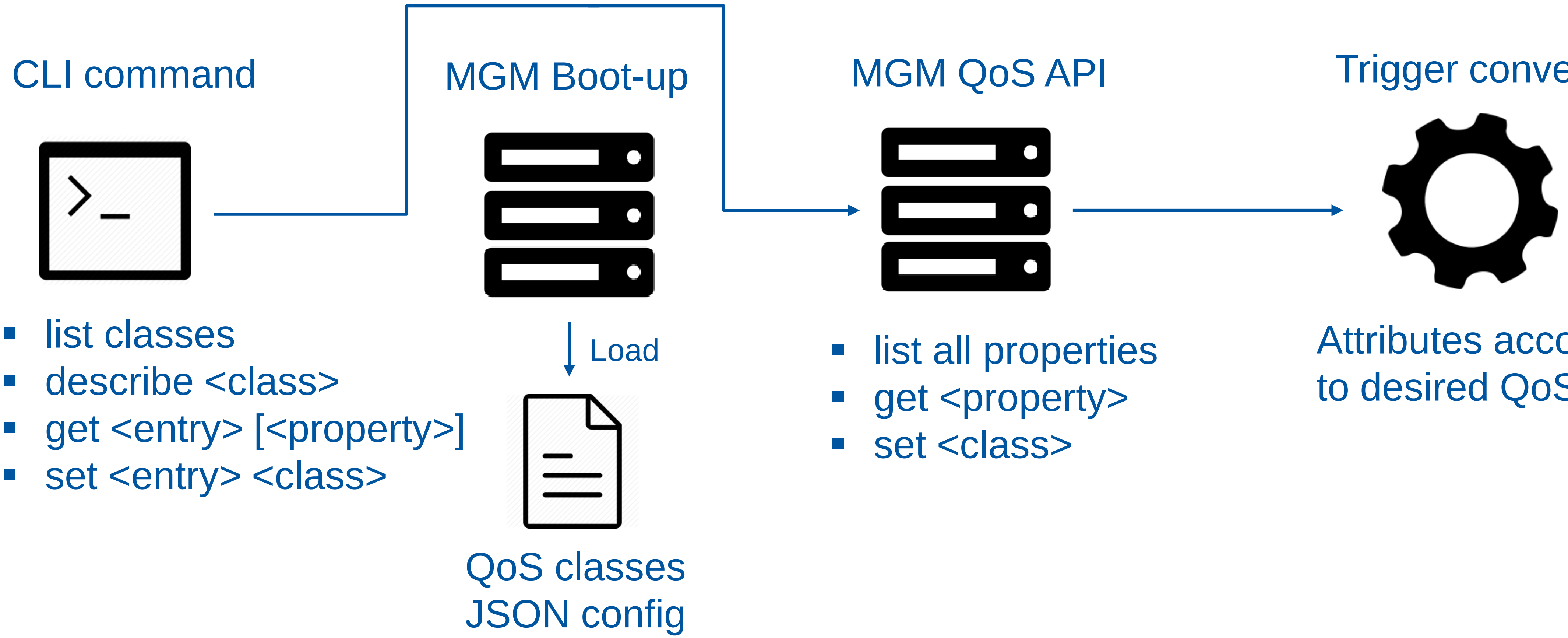
QoS class example

```
{
  "name": "disk_plain",

  "transition": [ "disk_replica“ ],

  "metadata": {
    "cdmi_data_redundancy_provided": 0,
    "cdmi_geographic_placement_provided":
    [ "CH“ ],
    "cdmi_latency_provided": 75
  },
  "attributes": {
    "layout": "plain",
    "replica": 1,
    "checksum": "adler32",
    "placement": "scattered"
  }
}
```

QoS overview



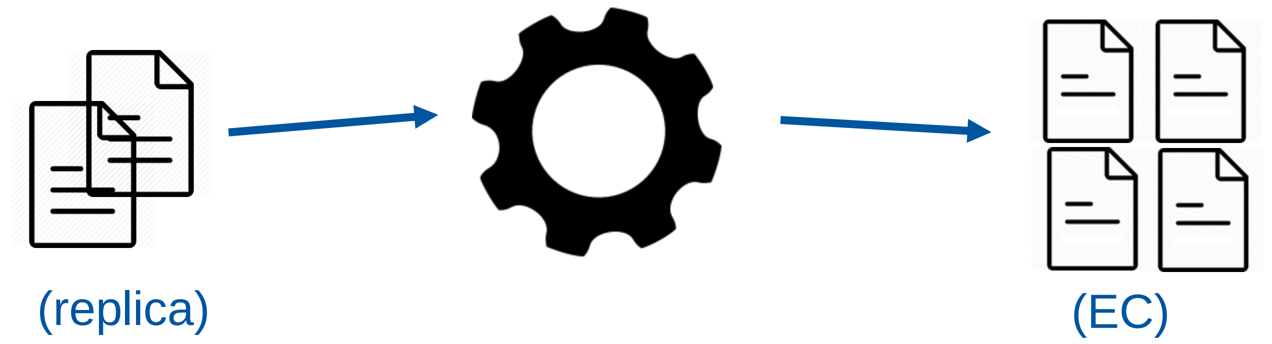
- list classes
- describe <class>
- get <entry> [<property>]
- set <entry> <class>

- list all properties
- get <property>
- set <class>

Attributes according to desired QoS class

Converter Engine

- Rewrite of the converter daemon
- One single converter instead of one per space
- Converts files from one layout/QoS class to another using ThirdPartyCopy



Converter Engine (cont'd)

- Persistent conversion jobs storage by using QuarkDB
- Jobs are fetched in batches of 1000
- Runtime scalable threadpool
- Interact via new `eos convert` command

```
$ eos convert status
Threadpool: thread_pool=converter_engine min=16 max=400
size=16 queue_size=82
Running jobs: 100
Pending jobs: 176
Failed jobs: 0
Failed jobs (QDB): 2


$ eos -j convert file /eos/xdc/test/convert replica:4
                                           default adler32
{
   "conversion_id" : "000000000009dc:default#00650312",
   "path" : "/eos/xdc/test/convert"
}
```

Converter Engine - Improvements

- Allow directory conversions
- Support periodic conversion rule on directory
- Testing at scale

Thank you for your time!

Icons: Computer Network Icons collection - openclipart.org

CERN