ceph days

Scale Multiple Ceph-Clusters Horizontally

Ansgar Jazdzewski, Object Storage Engineer, Hetzner Cloud GmbH

HETZNER

S3 AT SCALE

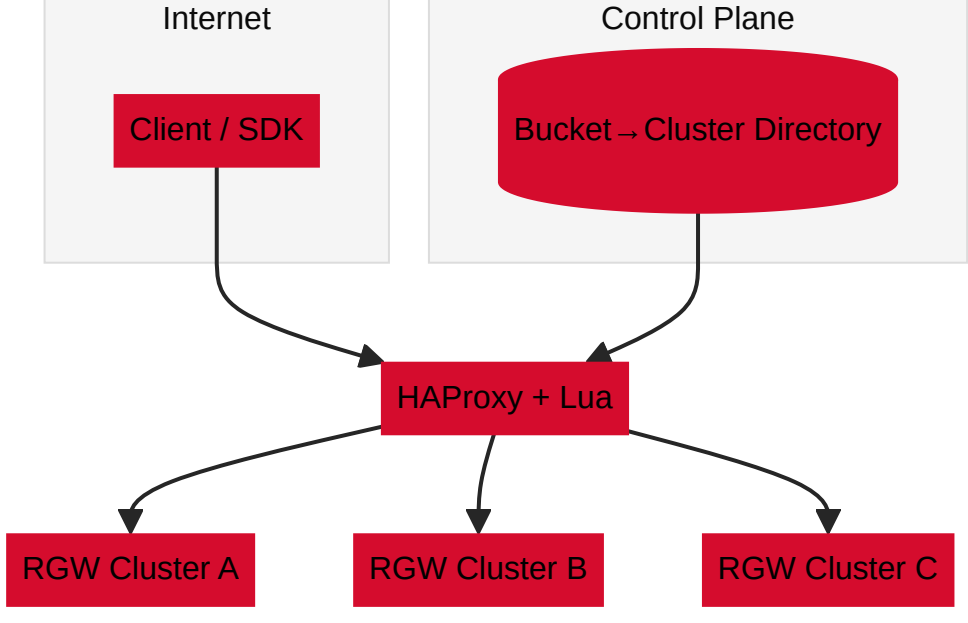
MULTIPLE CEPH-CLUSTER TRANSPARENT TO CUSTOMERS

Ansgar Jazdzewski • Hetzner-Cloud

WHY ONE CLUSTER ISN'T ENOUGH

- Physical Space
- Blast radius
- Rebalancing cost, recovery storms
- Dedicated failure domains
- Isolated maintenance windows

ARCHITECTURE



Overview

CHALLENGES TO OVERCOME

- Selct location
- Data Placement
- Move Data (Buckets)

HAPROXY + LUA (BUCKET-AWARE)

- read the bucketname from request
- create a LUA extention to lookup the cluster
- route to the correct cluster (backend)

PATH- AND DOMAIN- STYLE

Domainstyle

```
"https://[bucket].s3.example.com/[object]"
```

Pathstyle

```
"https://s3.example.com/[bucket]/[object]"
```

DECIDE IN HAPROXY

```
acl is_bucket_dns hdr_reg(host) -i ^[^.]*\.s3\.example\.com$
http-request set-var(txn.bucket) hdr(host),word(1,.) \
  if is_bucket_dns
http-request set-var(txn.bucket) path,word(2,/) \
  if ! is_bucket_dns
http-request set-var(txn.backend) lua.bucket2cluster \
  unless no_lua
http-response set-header X-Bucket %[var(txn.bucket)]
```

LUA STRUCTURE

How the Communication with Haproxy and Ceph work

LUA 1 (REQUIREMENTS)

```
local socket = require("socket")
local unix = require("socket.unix")
local socket_path = "/run/haproxy/bucket2cluster.sock"
```

LUA 2 (QUERY)

```lua
local function query_bucket(bucket)
  local client = unix()
  client:connect(socket_path)
  client:send("GET " .. bucket .. "\n")
  local response = client:receive("*l")
  client:close()
  return response
end
```

LUA 3 (RESULT)

```
function bucket2cluster(txn)
  local bucket = txn:get_var("txn.bucket")
  local response = query_bucket(bucket)
  if response:find("^200 ") then
    backend = string.sub(response, 5)
  else
    backend = "unknown"
  end
  txn:set_var("txn.backend", backend)
end
```

LUA 4 (REGISTER)

```
core.register_fetches("bucket2cluster",
  bucket2cluster_lookup)
```

HAPROXY PERFORMANCE OPTIMIZATIONS

Lua Load Per Thread

```
global
  lua-load-per-thread /etc/haproxy/bucket2cluster.lua
```

Load Lua script per thread to avoid lock contention

HAPROXY + CREATE/DELETE BUCKET

HAPROXY + CREATE/DELETE BUCKET

- Create ACL

HAPROXY + CREATE/DELETE BUCKET

- Create ACL
- Send to Service

HTTP METHODS

```
acl is_put method PUT
acl is_del method DELETE
```

TEST ROOT

```
acl path_is_root_host \
  path,normalize(),url_dec,lower \
  -m reg ^/$
acl path_is_root_path \
  path,normalize(),url_dec,lower \
  -m reg ^/[^/]+/?$
```

path "/" or "/bucket/"

SET ROOT

```
acl is_bucket_dns hdr_reg(host) -i ^[^.]*\.s3\.example\.com$
acl is_bucket_path hdr(host) -i "s3.example.com"
acl is_bucket_root \
  ( is_bucket_dns path_is_root_host ) \
  or ( is_bucket_path path_is_root_path )
```

CREATE OR DELETE

```
acl has_authorization hdr_cnt(authorization) gt 0
acl is_bucket_create \
  is_put has_authorization is_bucket_root
acl is_bucket_delete \
  is_del has_authorization is_bucket_root
```

USE BACKEND

```
use_backend be_bucket_create if is_bucket_create
use_backend be_bucket_delete if is_bucket_delete
```

SERVICE CREATE/DELETE BUCKET

SERVICE CREATE/DELETE BUCKET

- get Bucket request

SERVICE CREATE/DELETE BUCKET

- get Bucket request
- validate

SERVICE CREATE/DELETE BUCKET

- get Bucket request
- validate
- find cluster

SERVICE CREATE/DELETE BUCKET

- get Bucket request
- validate
- find cluster
- update db / check quota

SERVICE CREATE/DELETE BUCKET

- get Bucket request
- validate
- find cluster
- update db / check quota
- create / delete bucket

WHY WE DO THIS

WHY WE DO THIS

Quota work across clusters / locations (num buckets)

WHY WE DO THIS

Quota work across clusters / locations (num buckets)

Unified view Buckes can be managed with the UI (Hetzner Console) and S3-API

BUCKET MIGRATION

to maintain cluster utilisation we have to move buckets from time to time

BUCKET MIGRATION

to maintain cluster utilisation we have to move buckets from time to time

- Storage capacity

BUCKET MIGRATION

to maintain cluster utilisation we have to move buckets from time to time

- Storage capacity
- Noisy neighbors

CEPH RAOSGW MULTISIDE

We use multiside to have all metada in sync Users and Bucketnames

CHANGE THE DEFAULT

```
radosgw-admin sync group create \
  --group-id=default \
  --status=allowed
```

CREATE A PIPE

```
radosgw-admin sync group pipe create \
  --group-id=default \
  --pipe-id=pipe-main \
  --source-zones='*' \
  --source-bucket='*' \
  --dest-zones='*' \
  --dest-bucket='*'
```

MOVING A BUCKET

MOVING A BUCKET

- Identify a Bucket

MOVING A BUCKET

- Identify a Bucket
- Create Pipeline

MOVING A BUCKET

- Identify a Bucket
- Create Pipeline
- Run the actual Sync

MOVING A BUCKET

- Identify a Bucket
- Create Pipeline
- Run the actual Sync
- Change Database (Cluster)

MOVING A BUCKET

- Identify a Bucket
- Create Pipeline
- Run the actual Sync
- Change Database (Cluster)
- Delte data on Source-Side

BUKETS

our Goal is to move wehn possible easy Bukets

BUKETS

our Goal is to move wehn possible easy Bukets

- Size of the buket (medium)

BUKETS

our Goal is to move wehn possible easy Bukets

- Size of the buket (medium)
- Traffic read/write

BUKETS

our Goal is to move wehn possible easy Bukets

- Size of the buket (medium)
- Traffic read/write
- Used features (not hit Bugs)

CONFIGURATION (GROUP)

```
radosgw-admin sync group create \
  --bucket="${BUCKET}" \
  --group-id="group-${BUCKET}" \
  --status="enabled"
```

CONFIGURATION (PIPE)

```
radosgw-admin sync group pipe create \
  --bucket="${BUCKET}" \
  --group-id="group-${BUCKET}" \
  --pipe-id="pipe-${BUCKET}" \
  --source-bucket="${BUCKET}" \
  --source-zones="${SOURCE_ZONE}","${DEST_ZONE}" \
  --dest-bucket="${BUCKET}" \
  --dest-zones="${SOURCE_ZONE}","${DEST_ZONE}"
```

COPY DATA

```
radosgw-admin bucket sync run \
  --source-zone=fsn1-prod1-ceph3 \
  --bucket="${BUCKET}" \
  --debug_rgw=1
```

DELETE SYNC

```
radosgw-admin sync group remove \
 --bucket="${BUCKET}" \
 --group-id="group-${BUCKET}"
```

CHALLENGES

CHALLENGES

- Custom Domains + HTTPS

CHALLENGES

- Custom Domains + HTTPS
- Cross-DC/Location bucket placement

CHALLENGES

- Custom Domains + HTTPS
- Cross-DC/Location bucket placement
- Keymanagemet (RO-Keys)

CHALLENGES

- Custom Domains + HTTPS
- Cross-DC/Location bucket placement
- Keymanagemet (RO-Keys)
- STS-Lite + IAM

ADVANCED FEATURES

ADVANCED FEATURES

- Automated bucket rebalancing

ADVANCED FEATURES

- Automated bucket rebalancing
- Predictive capacity planning

Q&A

What’s your biggest scale blocker today?