dCache+CEPH
Tigran Mkrtchyan for dCache Team
dCache User Workshop, Umeå, Sweden
Agenda (from)

- WebDAV
- xFTP
- XrootD
- NFS
- DCAP

DC POOL

RAID 6

DC POOL

RAID 6

DC POOL

RAID 6

DC POOL

RAID 6
Agenda (to)

WebDAV  xFTP  XrootD  NFS  DCAP

DC POOL  DC POOL  DC POOL  DC POOL

OSD  HDD  OSD  HDD  OSD  HDD  OSD  HDD  OSD  HDD  OSD  HDD  OSD  HDD  OSD  HDD  OSD  HDD  OSD  HDD  OSD  HDD  OSD  HDD  OSD  HDD  OSD  HDD  OSD  HDD  OSD  HDD  OSD  HDD  OSD  HDD  OSD  HDD
Final result
Why CEPH?

- Demanded by sites
  - deployed as objects store
  - used as back-end for OpenStack and Co.
- Possible alternative for RAID systems
  - no rebuilds on disk failure
  - one disk per OSD
  - allows to use JBODs and ignore broken disks
CRUSH in Action

\[ \text{HASH(Object)} \mod 3 = 2 \]
BUT, not only CEPH

- CEPH specific code only ~400 lines
- Other object store can be adopted
  - DDN WOS
- Swift/S3/CDMI
- Cluster file systems (as a side effect)
  - Luster
  - GPFS
  - GlusterFS
How it works?

• Pool still keeps it’s own meta
  • File state, checksum, etc.
• All IO requests forwarded directly to CEPH
• Each dCache pool is a CEPH pool
  • resilience
  • placement group
• Each dCache file is a RBD image in CEPH
  • striping
  • write-back cache
  • out-of-order writes
Pool internals

- cell communication
- mover queue
- flush queue

Data Mover

virtual repository

metadata

data repository
Pool internals

- cell communication
- mover queue
- flush queue

Data Mover

- POSIX IO
- XFS/ext4
- POSIX IO
- metadata
- ../data
- ../meta
Pool internals

- cell communication
- mover queue
- flush queue

Data Mover

librados

RDB

virtual repository

metadata

../meta

data repository

ceph
dCache setup

# layout.conf

pool.backend = ceph

# optional configuration

pool.backend.ceph.cluster = dcache
pool.backend.ceph.config = /.../ceph.conf
pool.backend.ceph.pool-name = pool-name
On the CEPH side

$ rados mkpool pool-name ....

$ rbd ls -p pool-name
0000000635D5968A4DD89E29C242185B2D82
00000001A770D854E41448D87C91822D90F0F
....

$
HSM script

- file:/path/to/pnfsid
  - shortcut to /path/to/pnfsid
- backend://
  - rbd://<pool name>/pnfsid

All files accessible in CEPH without dCache
Current Status

- Part of dCache-3.0
- Focus on stability and functionality first
  - all existing dCache feature set must be available
- uses RBD interface
  - striping
  - write-back caching
  - alterable content
- Thanks Johan Guldmyr for testing!
  - all (known) issues are fixed 3.0.4 & 3.0.13
- Part of my testing infrastructure
- Still missing on-the-field instance
Lightning talk #1
(SQL or noSQL?)
RECENTLY DURING THE JOB INTERVIEW

OK.
You are an expert in NoSQL. Are there any other technologies you know well?

Of course!
geek & poke
NoXML, NoUML, NoWSDL, NoSAP, NoIBM, ...
Pool internals

- cell communication
- mover queue
- flush queue

Data Mover

virtual repository

metadata

../meta

data repository

librados RDB

Pool internals

- cell communication
- mover queue
- flush queue

Data Mover

virtual repository

metadata

../meta

data repository

librados RDB
Remote Metadata (oh, no!)

```java
pool.plugins.meta=
    o.d.p.r.m.m.MongoDbMetadataRepository

pool.plugins.meta.mongo.url=
    mongodb://nodeA:27017,nodeB:27017

pool.plugins.meta.mongo.db=pdm
```
> db.poolMetadata.findOne()

```json
{
    "_id" : ObjectId("5901d0dcd23064c72fec70dd"),
    "pnfsid" : "0000852CC74061FF4669B3F3DD0D0F0DA468",
    "pool" : "dcache-lab001-A",
    "version" : 1,
    "created" : NumberLong("1493290829481"),
    "hsm" : "osm",
    "storageClass" : "<Unknown>:<Unknown>",
    "size" : NumberLong(801954),
    "accessLatency" : "NEARLINE",
    "retentionPolicy" : "CUSTODIAL",
    "locations" : [ ],
    "map" : {
        "uid" : "3750",
        "gid" : "3750",
        "flag-c" : "1:bbfc21ed"
    },
    "replicaState" : "PRECIOUS",
    "stickyRecords" : {} 
}
```
Aggregation: Files with #replica > 1

> db.poolMetadata.aggregate(
  {
    "$group":
    {
      "_id": "$pnfsid", "count": { "$sum": 1 }
    }
  },
  {
    "$match":
    {
      "count": { "$gt": 1 } 
    }
  }
)

{ "_id" : "000053626EFD641344CF98674F2DB177A557", "count" : 2 }
{ "_id" : "0000DA769FF39DB645D98C2FBCBCB03940D1", "count" : 2 }
{ "_id" : "00004FB135CB3D5D44A4A01A6986D0FC379F", "count" : 2 }
{ "_id" : "0000180828ED01F248B2932D803988BAAD68", "count" : 2 }
{ "_id" : "0000F47168DD3FDE41D1882397AF1F5605B9", "count" : 2 }
{ "_id" : "000081F065EE796E4895BB4A7808A723588C", "count" : 2 }
{ "_id" : "0000E00132BF82C54048885E534AA7E8098D", "count" : 2 }
{ "_id" : "0000A2434F3051D340B79DE69E76932B24E1", "count" : 2 }
{ "_id" : "0000987BE0D888E04E9598ABE826990D347B", "count" : 2 }
{ "_id" : "00002832C952394D4B4399D077DA8162F58D", "count" : 2 }
{ "_id" : "000051EC4E1A48B741E4830712869B0595E8", "count" : 2 }
MapReduce: total sizes by state

> db.poolMetadata.mapReduce(
    function (){
        emit(this.replicaState, this.size);
    },
    function(k, v) {
        return Array.sum(v)
    },
    {
        out:{inline : 1}
    }).results
{
    "_id" : "BROKEN",
    "value" : NaN
},
{
    "_id" : "CACHED",
    "value" : 2635758434
},
{
    "_id" : "PRECIOUS",
    "value" : 1834228442752
}
Summary

- Distributed metadata required for pools on shared storage
- NoSQL databases on possibility
- We are working on best solution
- Stay tuned!
Links

- https://www.dcache.org/
- https://en.wikipedia.org/wiki/Software-defined_storage
- http://ceph.com/
CEPH vocabulary

- OSD – object storage device
  - Minimal storage unit, usually a single disk.
- Primary-Affinity – primary OSD for a object
  - CEPH clients only read and write objects from/to PA.
  - Each OSD has a weight to be a PA
    - PA (HDD) == 0; PA (SSD) > 0 → all client IO from SSDs only
- RF – replication factor
  - Number of replicas per object.
- PG - placement group
  - Logical storage unit. Each object stored in a placement group. PG creates required number of object replicas on one or more OSDs.
- POOL – logical container,
  - contains one or more placement groups
  - Replication factors are assigned to POOLs
- CRUSH - Controlled Replicated Under Scalable Hashing
  - Each client uses CRUSH algorithm to find out object location based on cluster map, which contains list of OSDs
- MON – cluster coordination daemon.
  - The entry point for the clients to discover CRUSH-maps