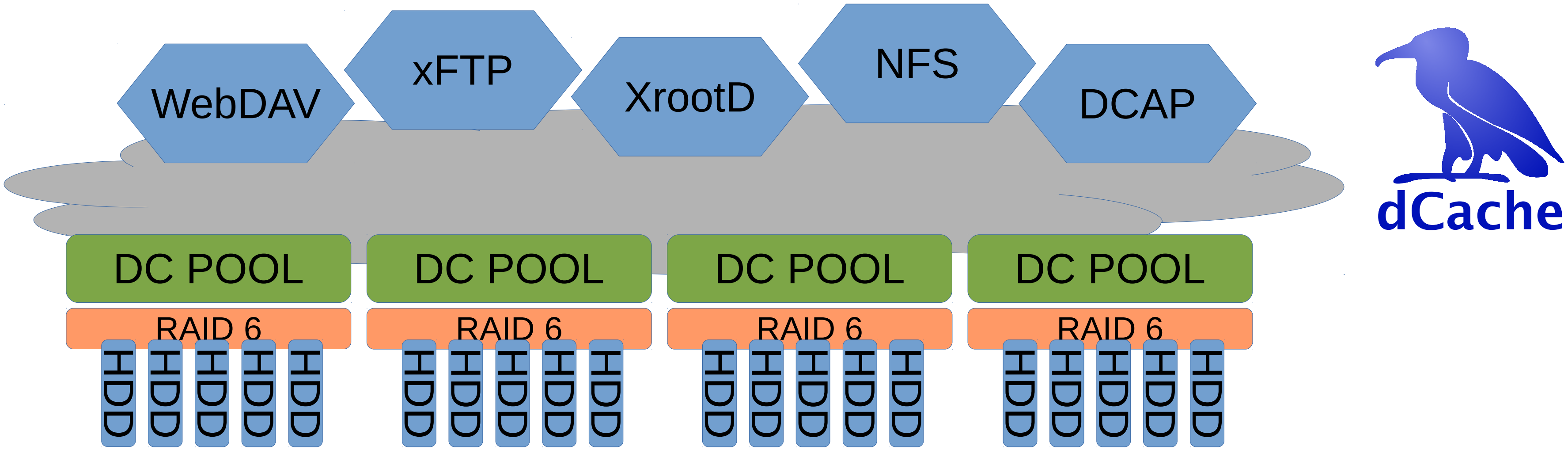
dCache - *outsourced* storage

Tigran Mkrtchyan for dCache Team

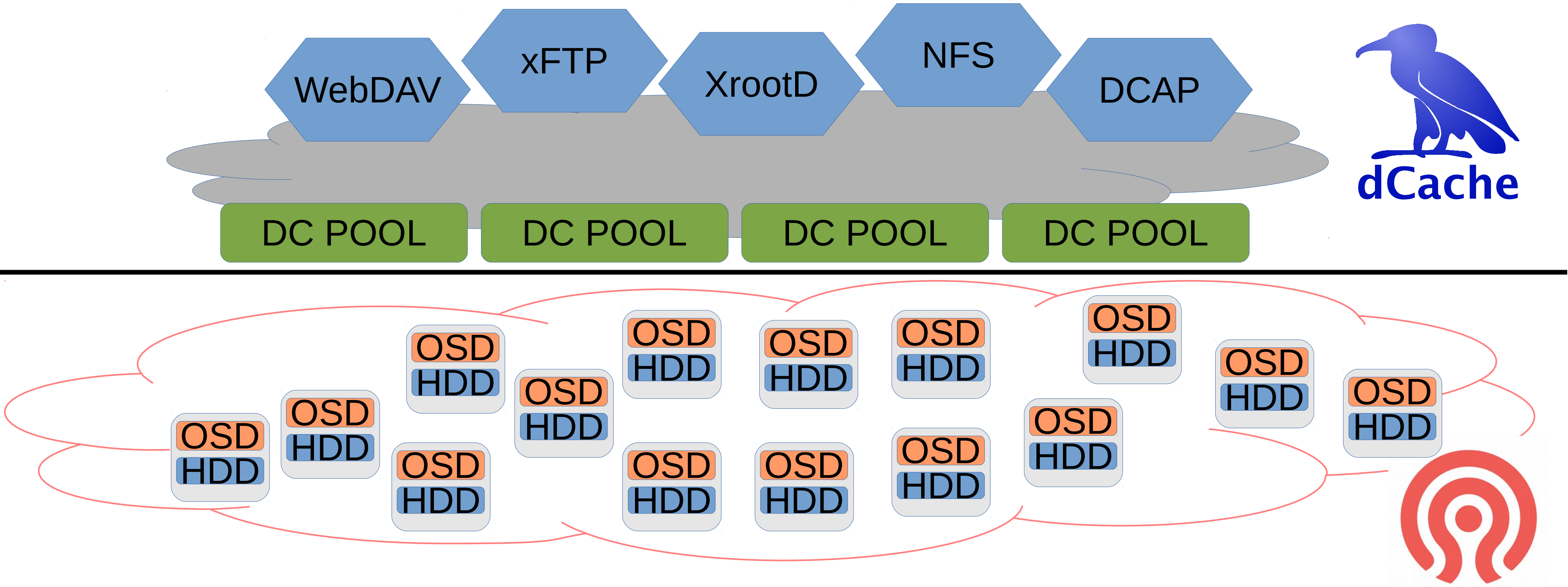
CHEP 2016, San Francisco

Fermilab
NDGF Nordic DataGrid Facility
INDIGO - DataCloud Better Software for Better Science
DESY
HELMHOLTZ | ASSOCIATION

Agenda (from)


WebDAV
xFTP
XrootD
NFS
DCAP
dCache
DC POOL
DC POOL
DC POOL
DC POOL
RAID 6
RAID 6
RAID 6
RAID 6
HDD
HDD
HDD
HDD
HDD
HDD
HDD
HDD
HDD
HDD
HDD
HDD
HDD
HDD
HDD
HDD
HDD
HDD
HDD
HDD

Agenda (to)


WebDAV
xFTP
XrootD
NFS
DCAP
dCache
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

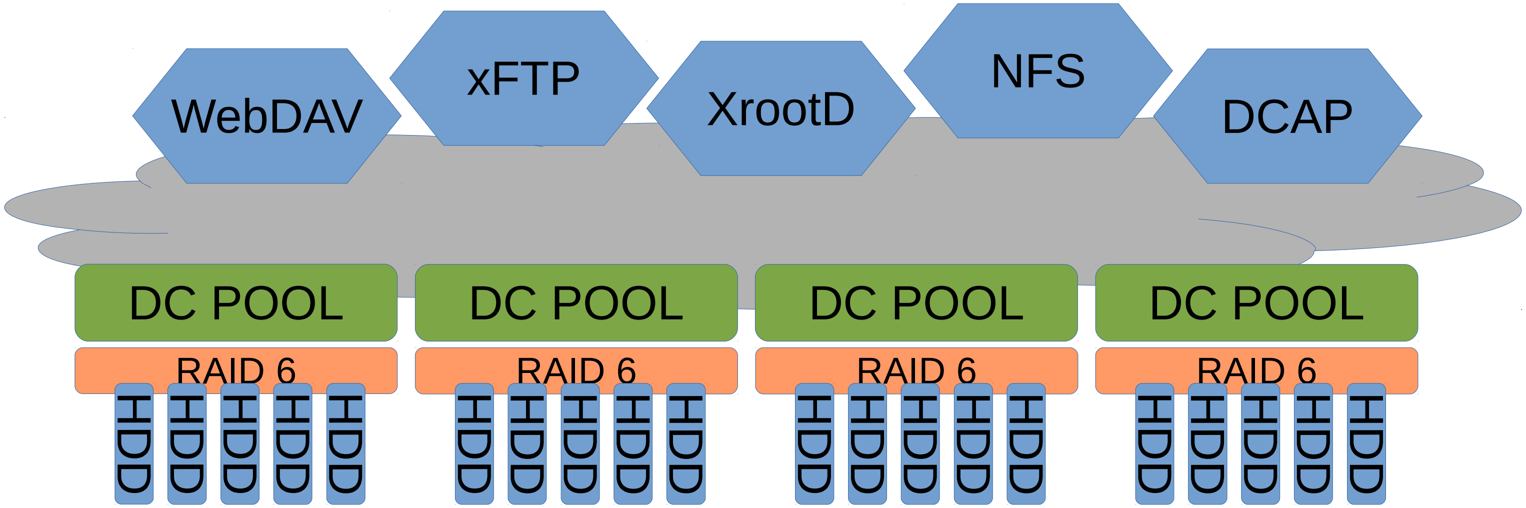
dCache as Storage System

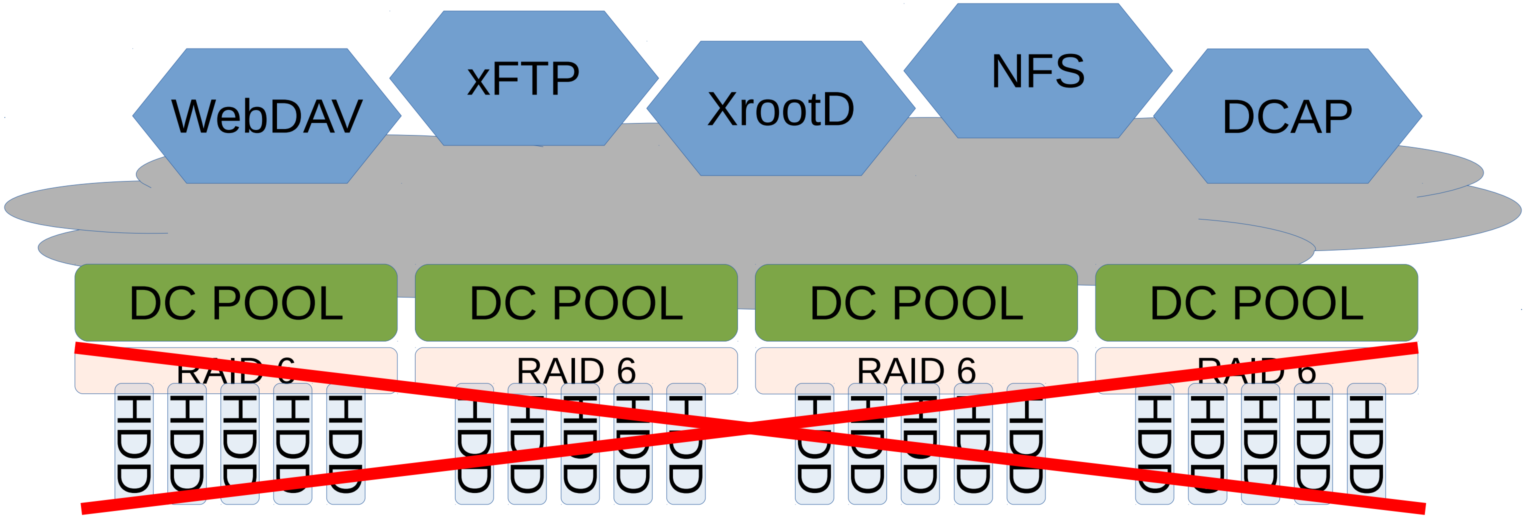
- Provides a single-rooted namespace.
- Metadata (namespace) and data locations are independent.
- Aggregates multipe storage nodes into a single storage system.
- Manages data movement, replication, integrity.
- Provides data migration between multiple tiers of storage (DISK, SSD, TAPE).
- Uniquely handles different Authentication mechanisms, like x509, Kerberos, login+password, auth tokens.
- Provides access to the data via variety of access protocols (WebDAV, NFSv4.1/pNFS, xxxFTP. DCAP, Xrootd, DCAP).

dCache as Storage System

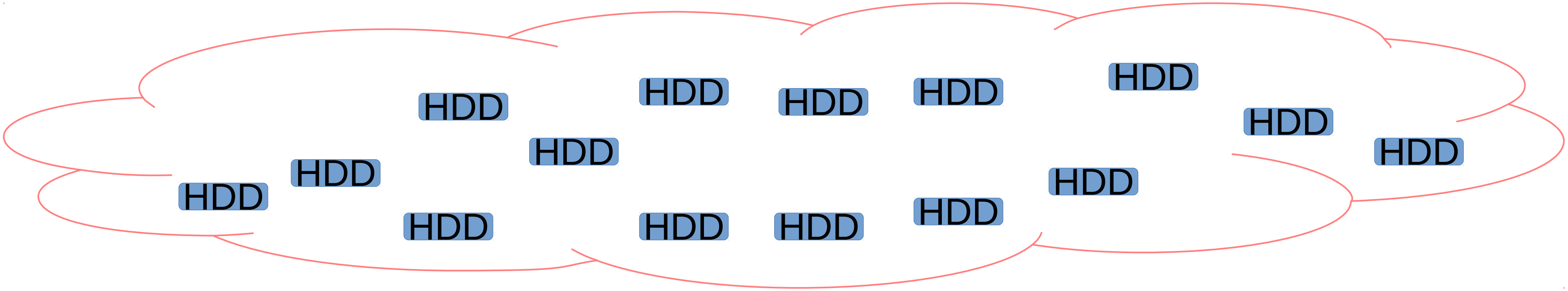
- Provides a single-rooted namespace.
- Metadata (namespace) and data locations are independent.
- Aggregates multipe storage nodes into a single storage system.
- Manages data movement, replication, integrity.
- Provides data migration between multiple tiers of storage (DISK, SSD, TAPE).
- Uniquely handles different Authentication mechanisms, like x509, Kerberos, login+password, auth tokens.
- Provides access to the data via variety of access protocols (WebDAV, NFSv4.1/pNFS, xxxFTP. DCAP, Xrootd, DCAP).

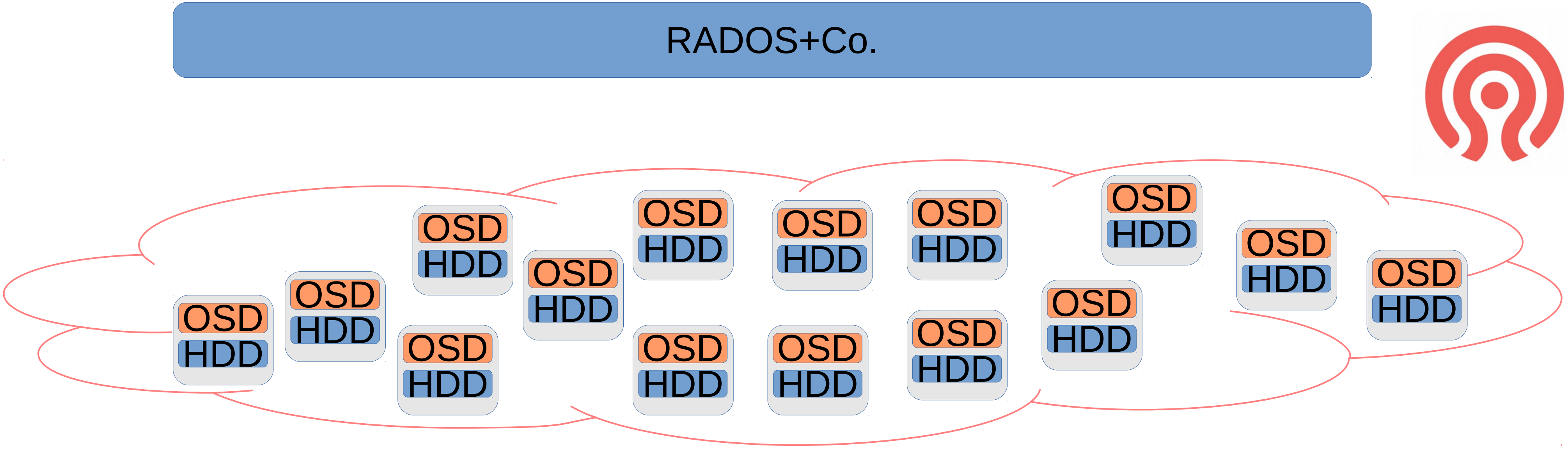
dCache building blocks




WebDAV
xFTP
XrootD
NFS
DCAP
DC POOL
DC POOL
DC POOL
DC POOL
RAID 6
RAID 6
RAID 6
RAID 6
HDD
HDD
HDD
HDD
HDD

dCache


HDD
HDD
HDD
HDD
HDD
HDD
HDD
HDD
HDD
HDD
HDD
HDD
HDD
HDD
HDD
HDD
HDD



RADOS+Co.

OSD
HDD

OSD
HDD

OSD
HDD

OSD
HDD

OSD
HDD

OSD
HDD

OSD
HDD

OSD
HDD

OSD
HDD

OSD
HDD

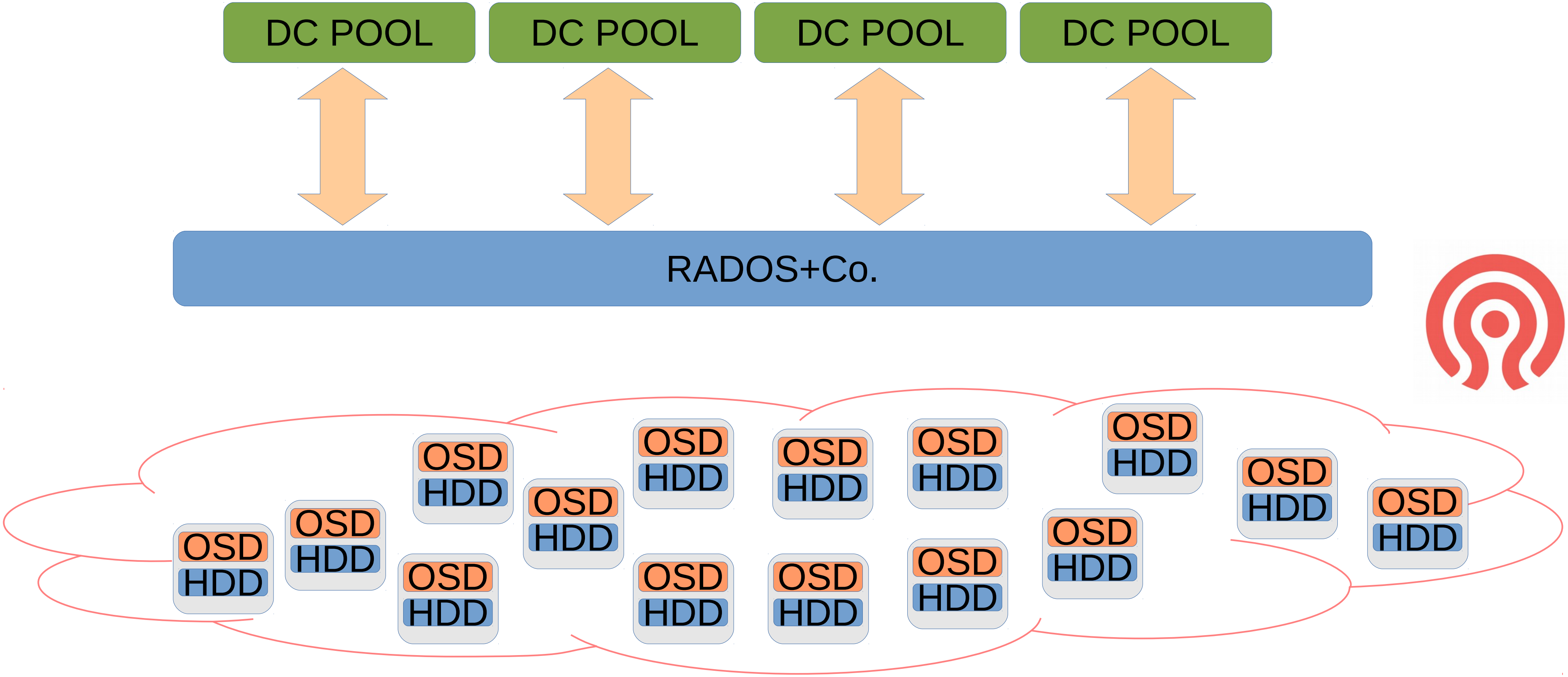
OSD
HDD

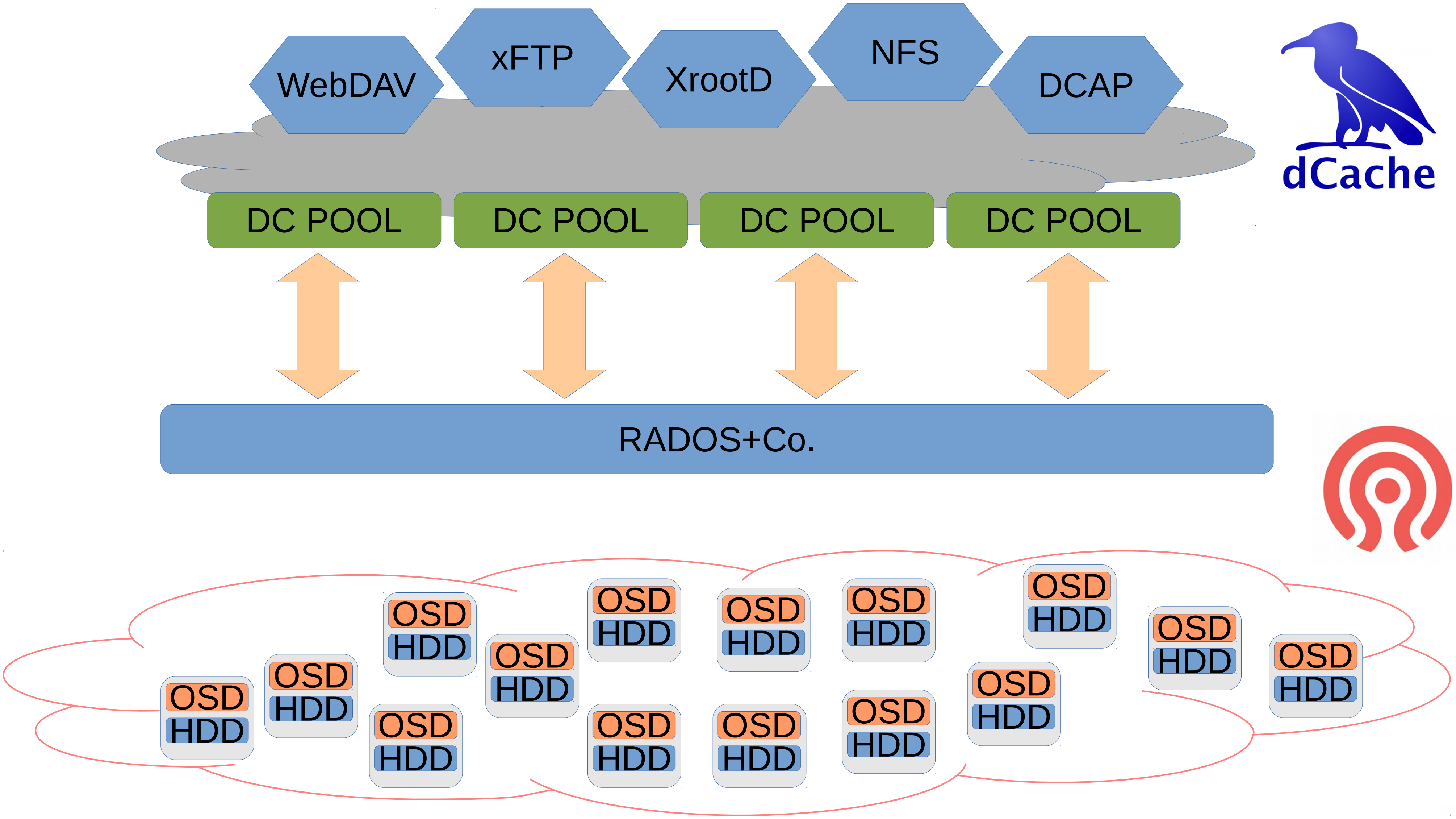
OSD
HDD

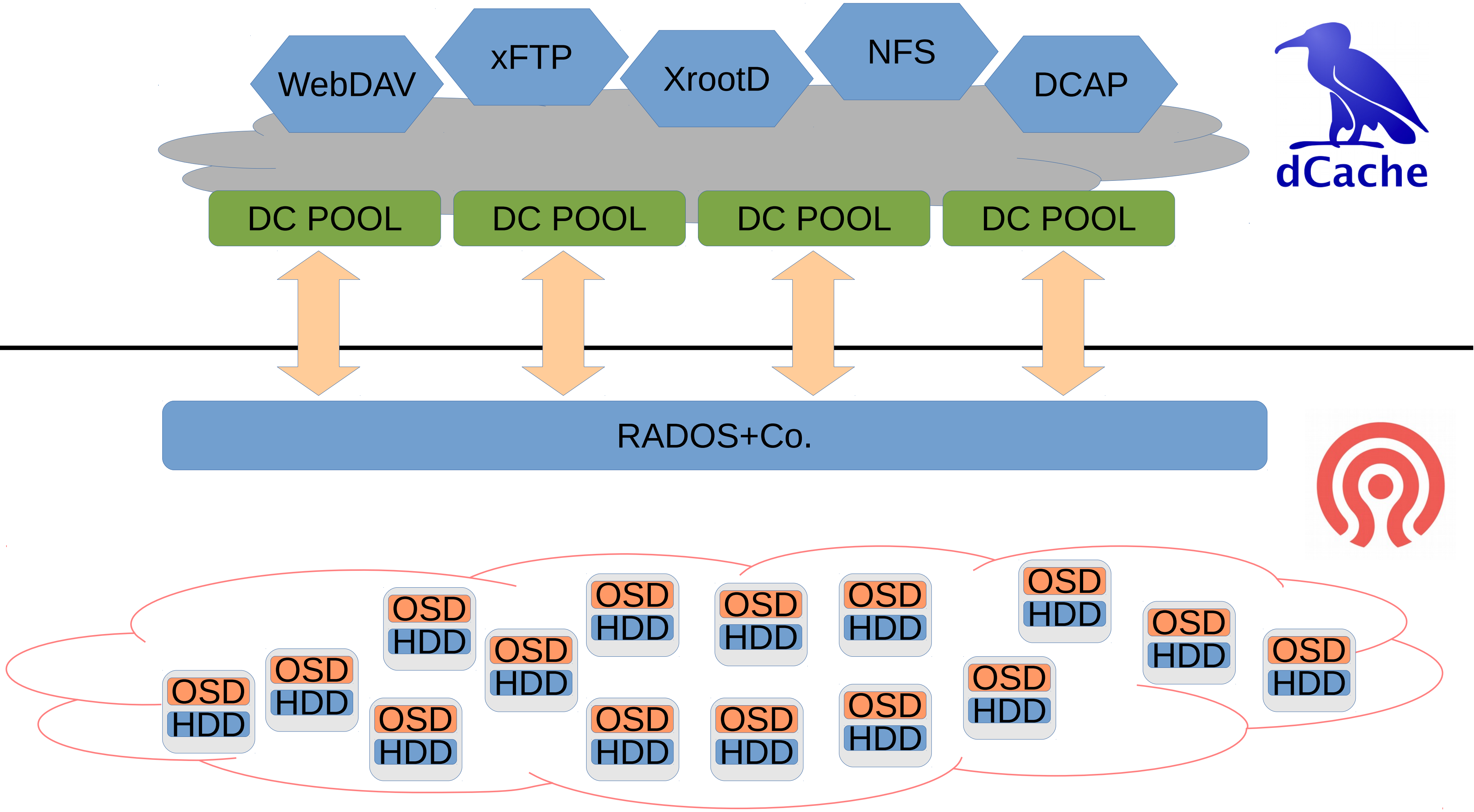
OSD
HDD

OSD
HDD

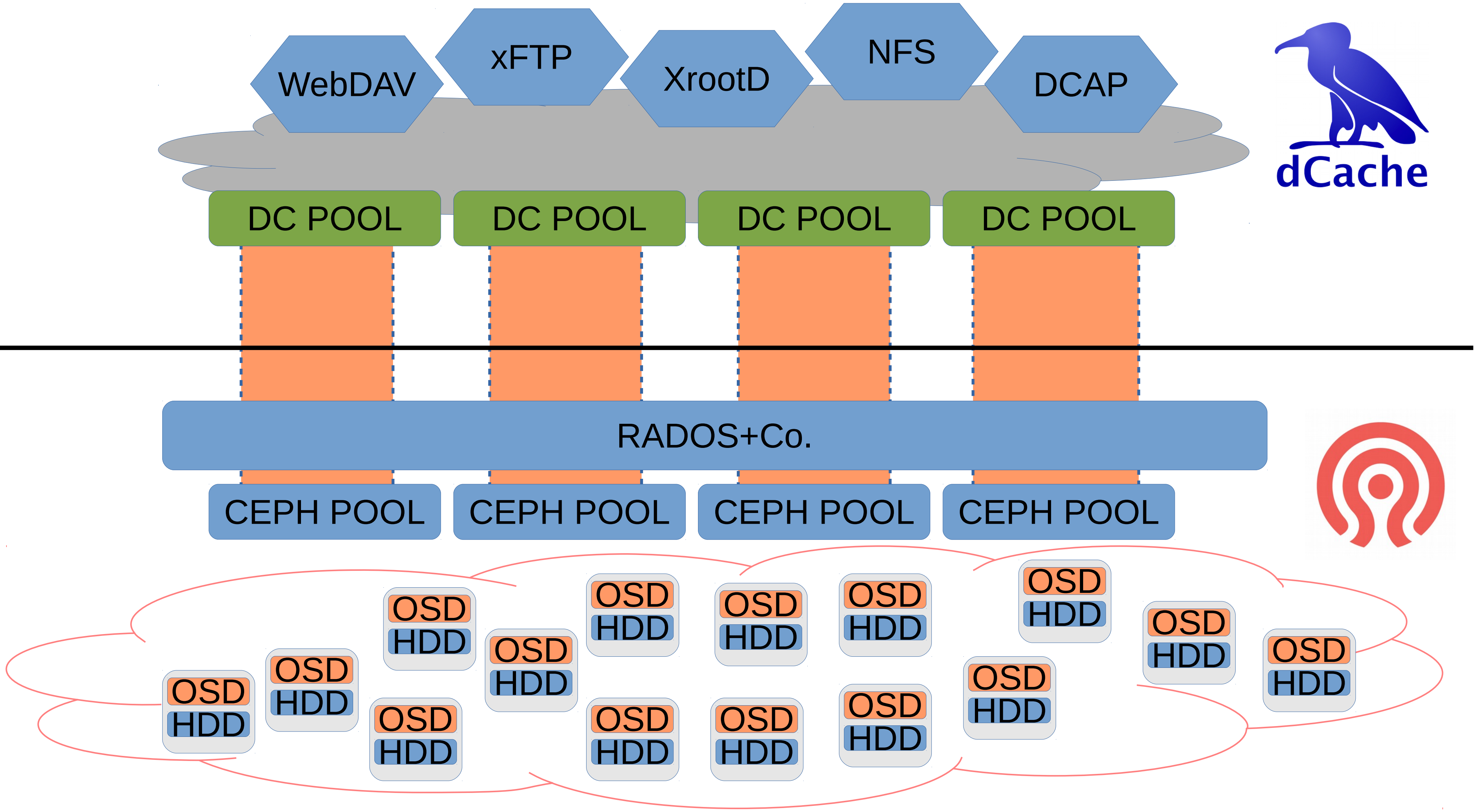
OSD
HDD


DC POOL
DC POOL
DC POOL
DC POOL
RADOS+Co.
OSD
HDD


WebDAV
xFTP
XrootD
NFS
DCAP
dCache
DC POOL
DC POOL
DC POOL
DC POOL
RADOS+Co.
OSD
HDD

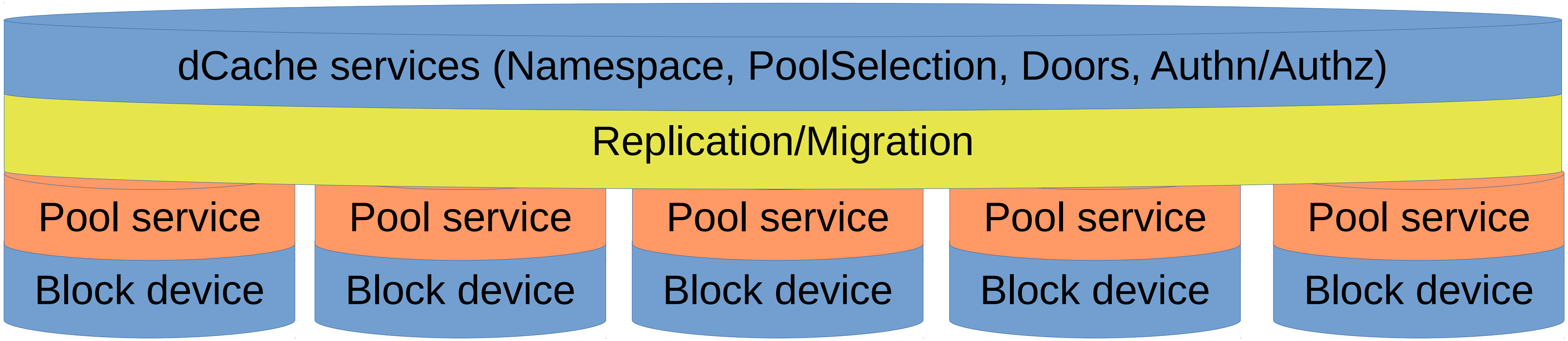

WebDAV
xFTP
XrootD
NFS
DCAP
dCache
DC POOL
DC POOL
DC POOL
DC POOL
RADOS+Co.
OSD
HDD

Final result



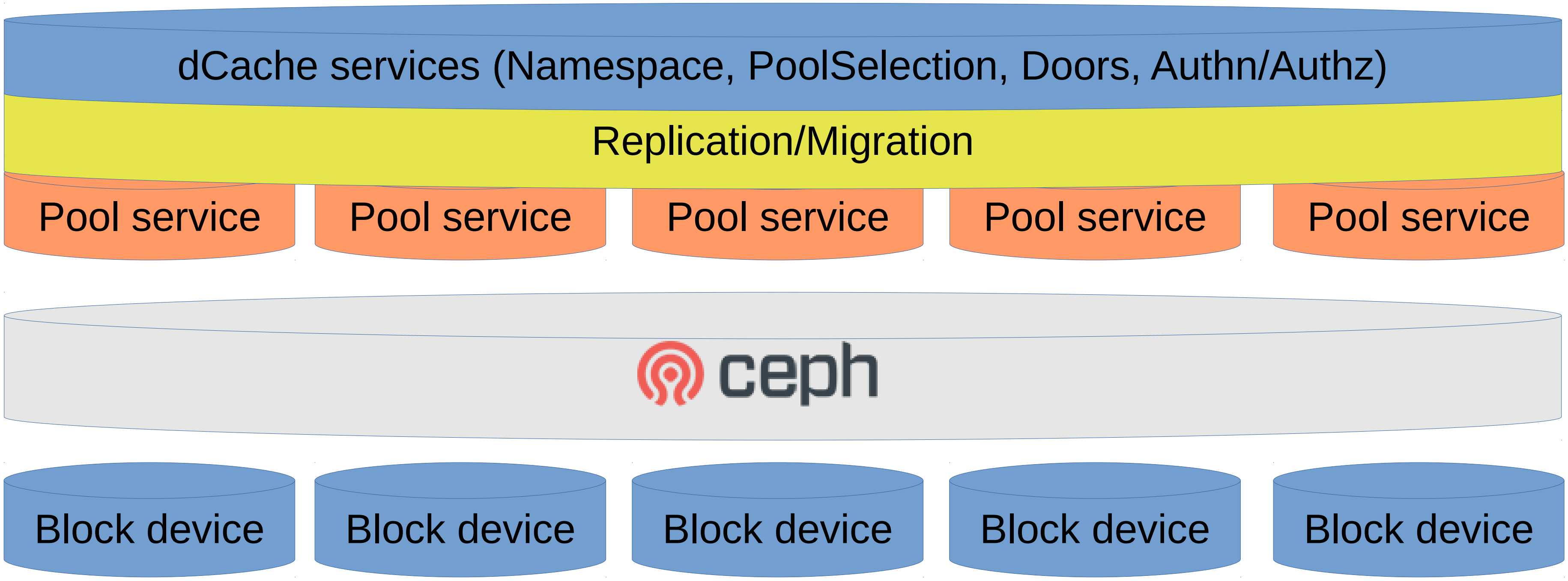
Storage in dCache (what we have)

- dCache provides high level service
- Data replication and management core dCache service
- Each pool attached to own disks



Storage in dCache (outsourcing, phase 1)

- dCache provides high level service
- Data replication and management core dCache service
- Each pool has it own 'partition' on shared storage

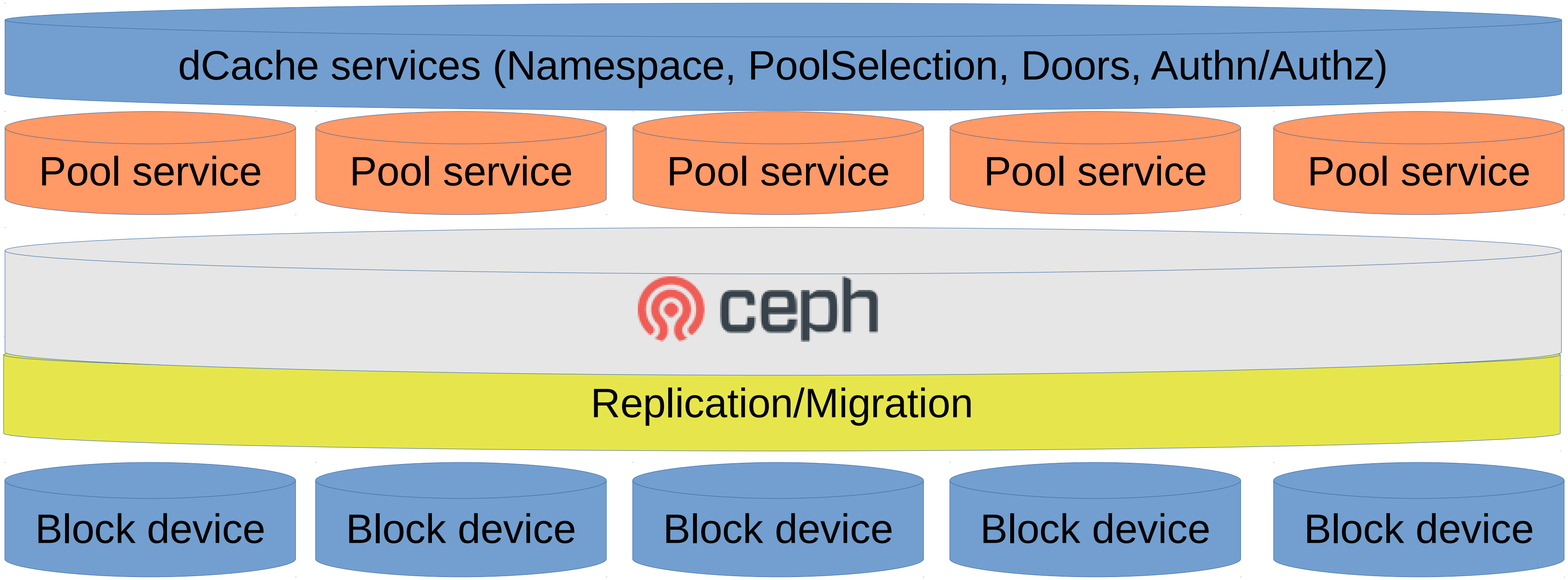


Phase 1 (changing IO layer)

- Single data server owns the data
- Single data server manages data
  - flush to tape
  - restore from tape
  - removal
  - garbage collection

Storage in dCache (outsourcing, phase 2)

- dCache provides high level service
- All pool see all 'partition' on shared storage
- Any pool can deliver data from any partition
- Object store takes care about replication and reliability


dCache services (Namespace, PoolSelection, Doors, Authn/Authz)

Pool service	Pool service	Pool service	Pool service	Pool service

Replication/Migration

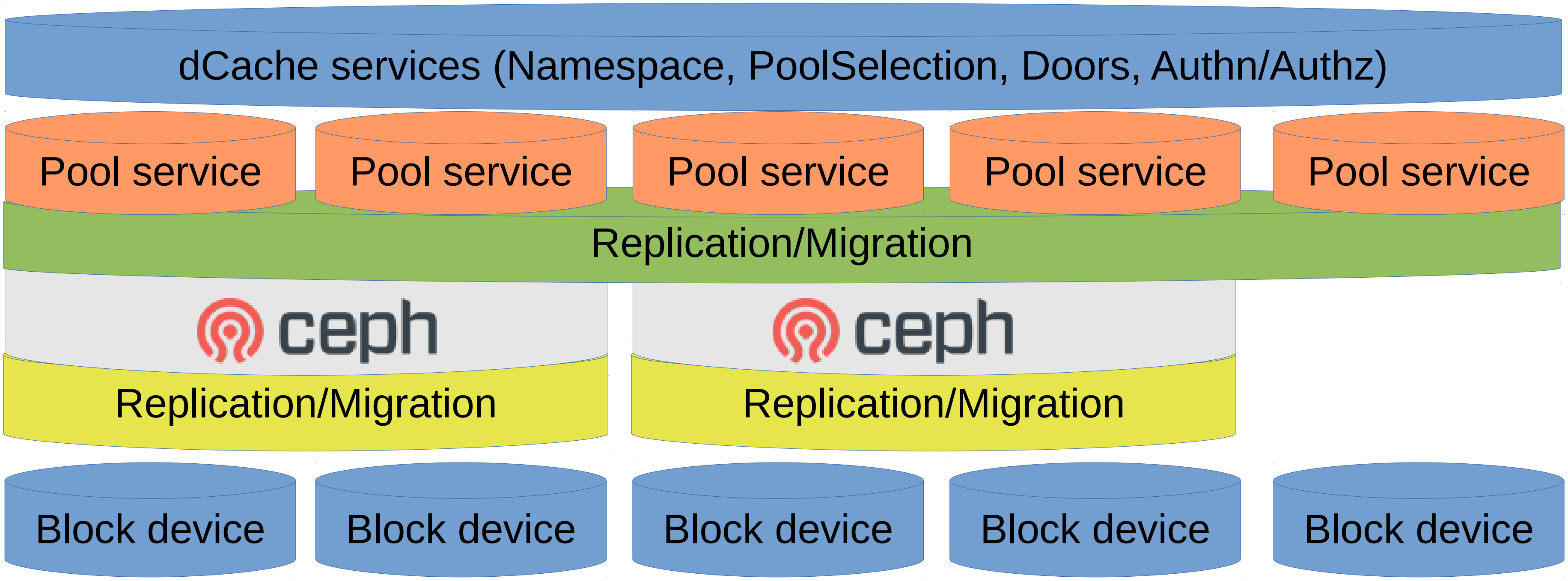
Block device	Block device	Block device	Block device	Block device

Phase 2 (Changing core philosophy)

- All data managed by 'quorum'
  - group decision who interact with tape
  - group decision who/when file is removed
  - File location is always 'known'

Storage in dCache (outsourcing, phase 3)

- dCache provides high level service
- dCache can move data between regular and OS pools



Phase 3 (mixed environment)

- Mixed setup
- Islands of storage servers
- dCache managed replication and data movement between islands

Why CEPH?

- Demanded by sites
  - deployed as objects store
  - used as back-end for OpenStack and Co.
  - Possible alternative for RAID systems
    - one disk per OSD
    - allows to use JBODs and ignore broken disks

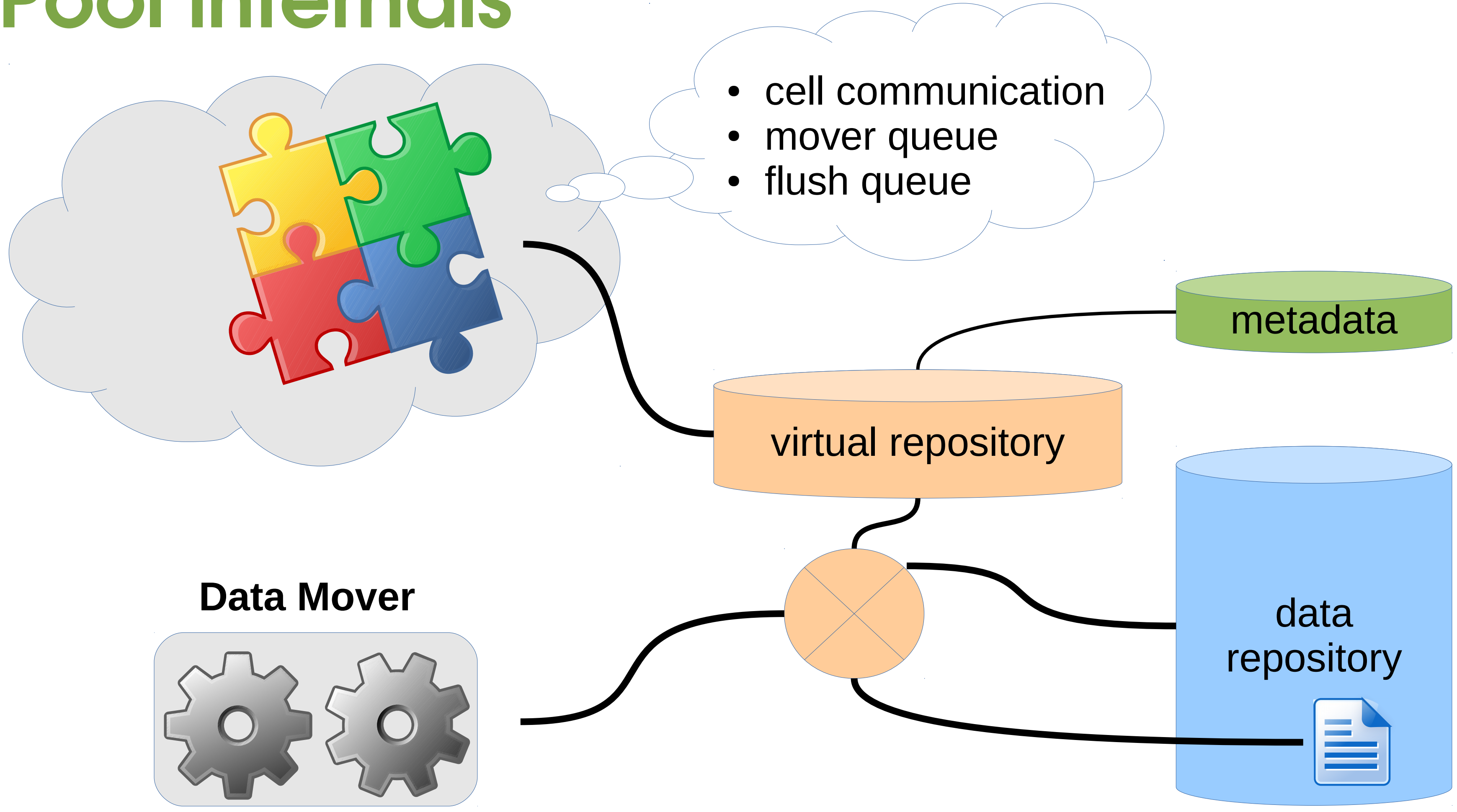
BUT, not only CEPH

- 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?

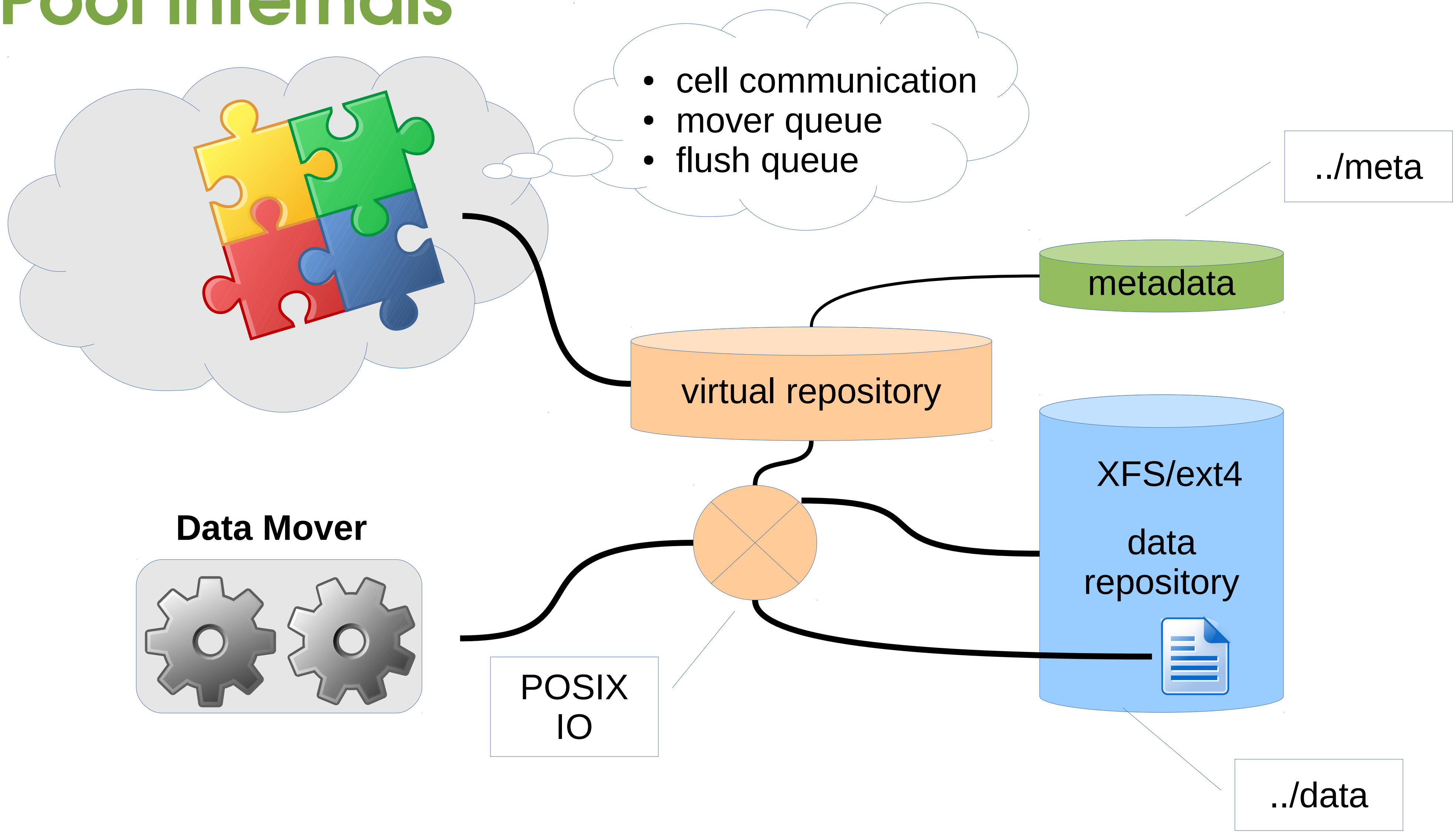
- Pool still keeps it's own meta
  - File state, checksum, etc.
- All IO requests forwarded directly to CEPH
- Each dCache pool is a CEPH ***pool***
  - resilience
  - placement group
- Each dCache file is a ***RBD image*** in CEPH
  - striping
  - write-back cache
  - out-of-order writes

Pool internals

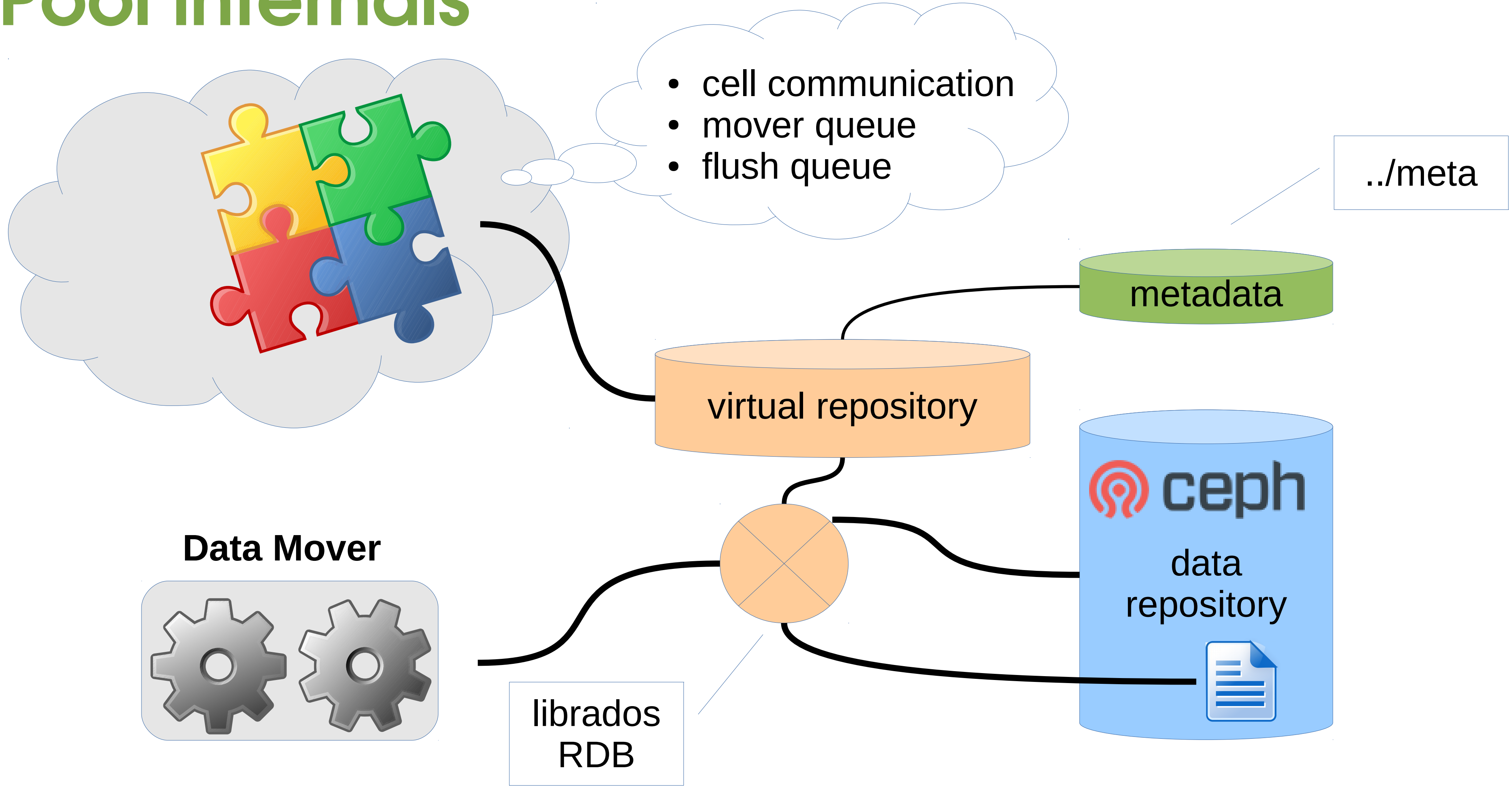


- cell communication
- mover queue
- flush queue

Pool internals



Pool internals



- cell communication
- mover queue
- flush queue

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
00000000635D5968A4DD89E29C242185B2D82
0000001A770D854E41448D87C91822D90F0F
....
$
```

HSM script

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

All files accessible in CEPH without dCache

Roadmap

- Phase 1
  - available in dCache-3.0
  - HSM integration under testing
  - performance/scale-out tests are required
    - sites are CEPH experts
- Phase 2/3
  - depends on user demand
  - operational overhead, if any
  - support overhead, if any
    - ***we don't want to convert into CEPH call center***

Current Status

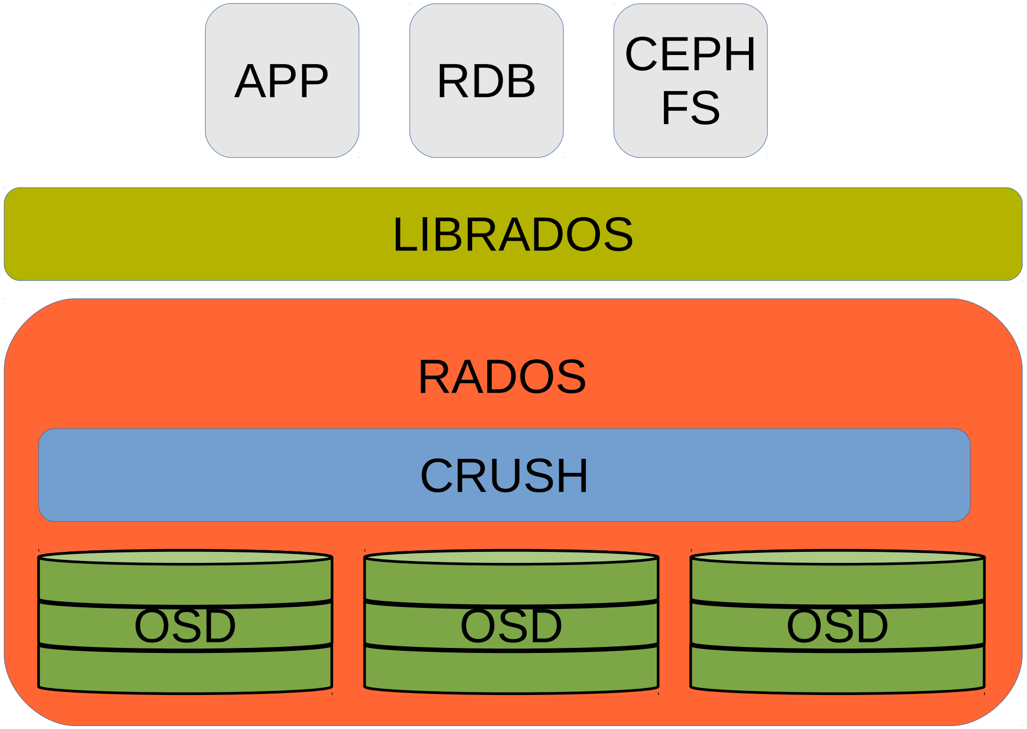
- Part of dCache-3.0
  - release end of October 2016
- Focus on stability and functionality first
  - all existing dCache feature set must be available
- uses RBD interface
  - striping
  - write-back caching
  - alterable content

dCache

Links

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

CEPH (extremely simplified)



- OSD ~ a physical disk
- CRUSH - determines how to store and retrieve data by computing data storage locations.
- RADOS - distributes objects across the storage cluster and replicates objects
- librados - provides low-level access to the RADOS service.

Software-defined storage

- Abstraction of logical storage services and capabilities from the underlying physical storage systems
- Automation with policy-driven storage provisioning with service-level agreements replacing technology details.
- Commodity hardware with storage logic abstracted into a software layer.