CEPHALOCON APAC 2018
THE FUTURE OF STORAGE
22-23 March 2018 | BEIJING

Global deduplication for Ceph

Myoungwon Oh

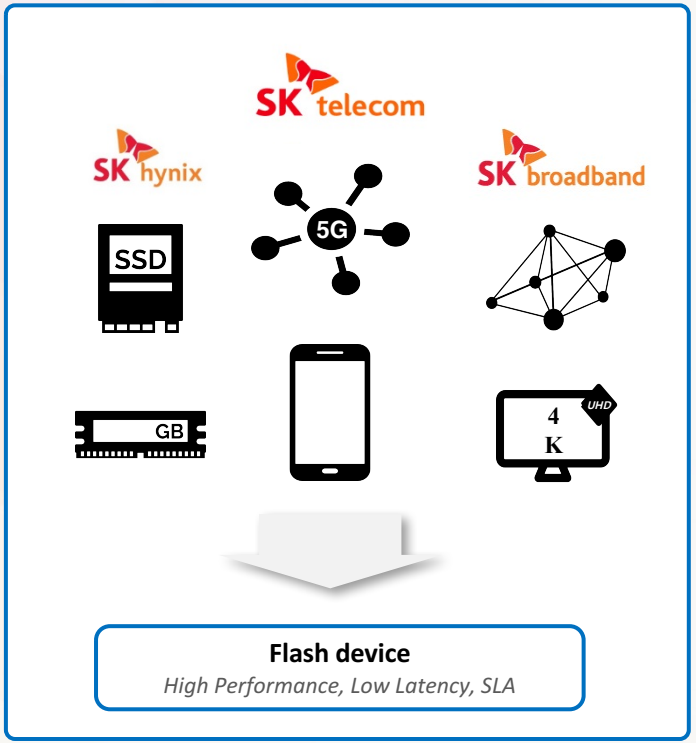
SW-Defined Storage Lab

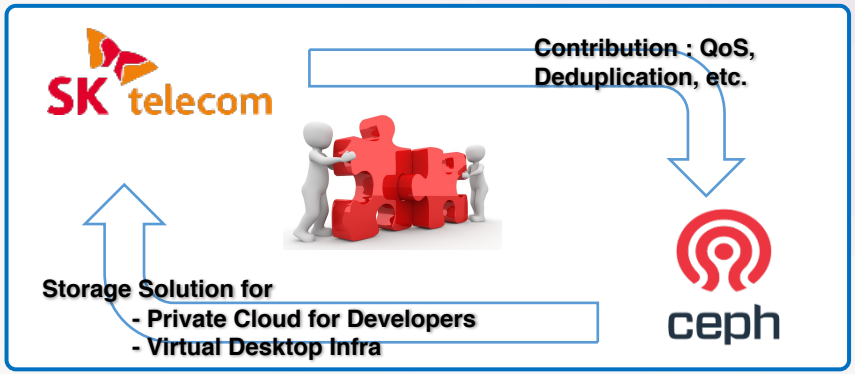
SK Telecom

Agenda

1. Why do we need Global dedup ?
2. Ceph deduplication design
3. Ceph extensible tier (implementation)
4. Upstream
5. Plan & issues

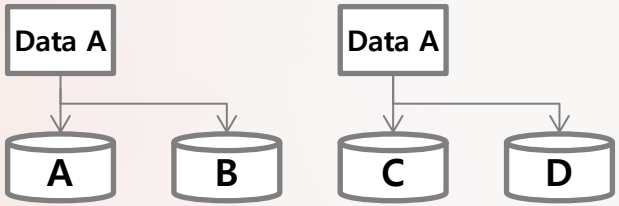
SK with software-defined storage



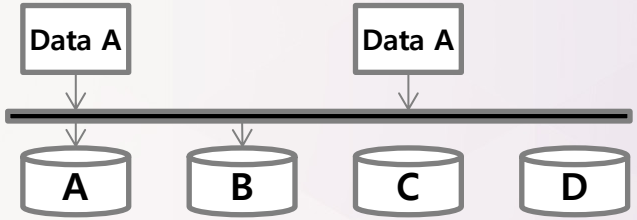


Why do we need global dedup?

- Up to 40% of total storage space can be saved via deduplication (in our private cloud)
- Local dedup (in a block device level) can not cover whole data reduction in terms of Cluster-wide



1) Local deduplication



2) Global deduplication

A) Design comparison

	4 OSD	8 OSD	12 OSD	16 OSD
Local Dedup	15.5%	8.1%	5.5%	4.1%
Global Dedup	50%	50%	50%	50%

B) FIO workload with deduplication ratio of 50% (32KB block size)

Design challenges

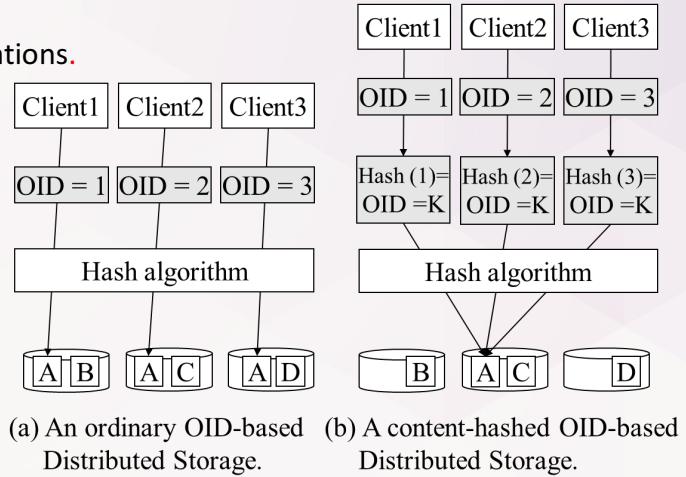
- Which implementation is the most appropriate for shared-nothing scale-out storage?
  - Applicable to existing source
  - Transparent to the application
  - Efficient metadata management
- How to manage dedup metadata?
- What is the most appropriate dedup method (e.g., inline or post)?
  - Performance
  - I/O cost

Design 1: Double distribution hash

- Do we need a new MDS (metadata server) for dedup?
  - Shared-nothing filesystem is scalable because there is no MDS.
    - MDS does not fit the Shared-nothing design
  - MDS needs additional I/Os to complete I/O requests
    - E.g., metadata query to MDS
  - How to do rebalance if we add MDS ?
  - Synchronization between MDSs

Design 1: Double distribution hash

- Can we implement dedup without MDS?
  - Yes, CAS (content-addressable storage) pool with double distribution hash!
    - (OID, Data) – chunking and fingerprinting -> (OID, Offests[], FPs[]) - FP is new OID -> (FP, Data)
    - Pros
      - No central MDS
      - Applicable to existing source without major modifications.
      - Transparent to the application
      - Efficient metadata management
      - Reusing existing architecture
        - DH, recovery, rebalance, data placement
    - Cons
      - I/O redirection (**need a translation layer**)

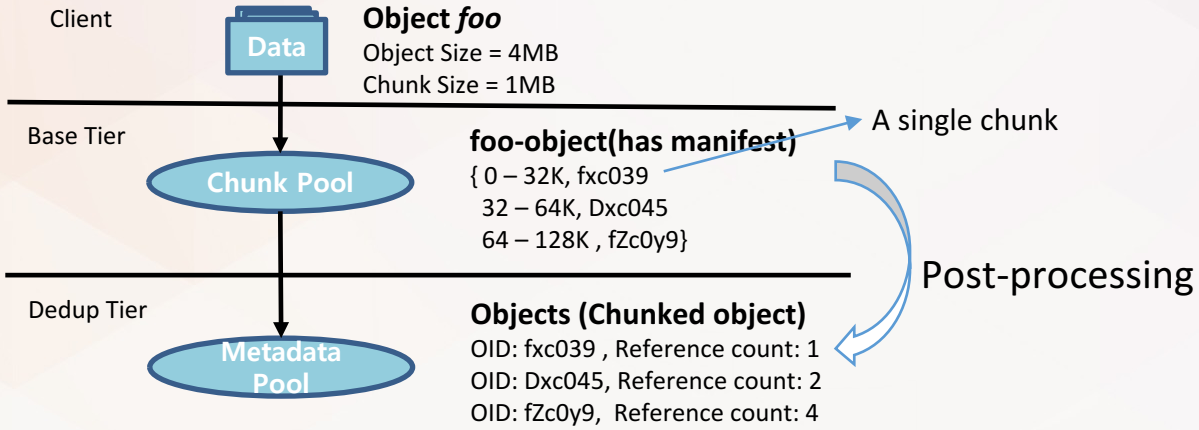


(a) An ordinary OID-based Distributed Storage.

(b) A content-hashed OID-based Distributed Storage.

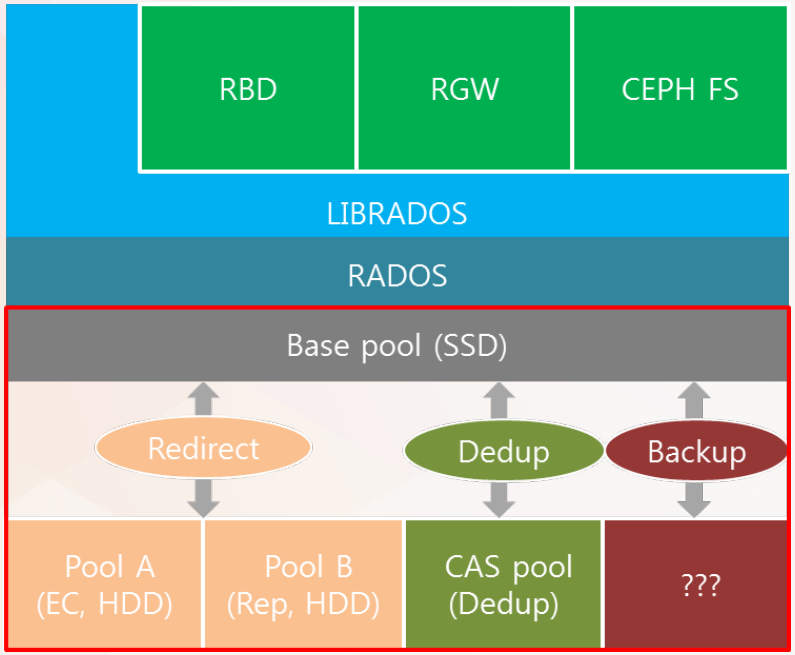
Design 2: Self-contained object for deduplication

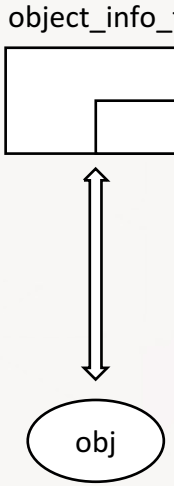
- External metadata structure needs additional complex linking between deduplication metadata and existing scale-out storage system
- Self-contained object can be answer
  - Dedup metadata is included in the original object



1. Find dirty metadata object which contains dirty chunks from the dirty object ID list.
2. Find the dirty chunk ID from the dirty metadata object's chunk map.
3. The deduplication engine generate a chunk object and send it to the chunk pool.
4. In the chunk pool, the chunk object generated in step 3 is placed
5. Add reference count information to the object.
6. When the chunk write at the chunk pool ends, update the metadata object's chunk map.

Implementation: Extensible tier



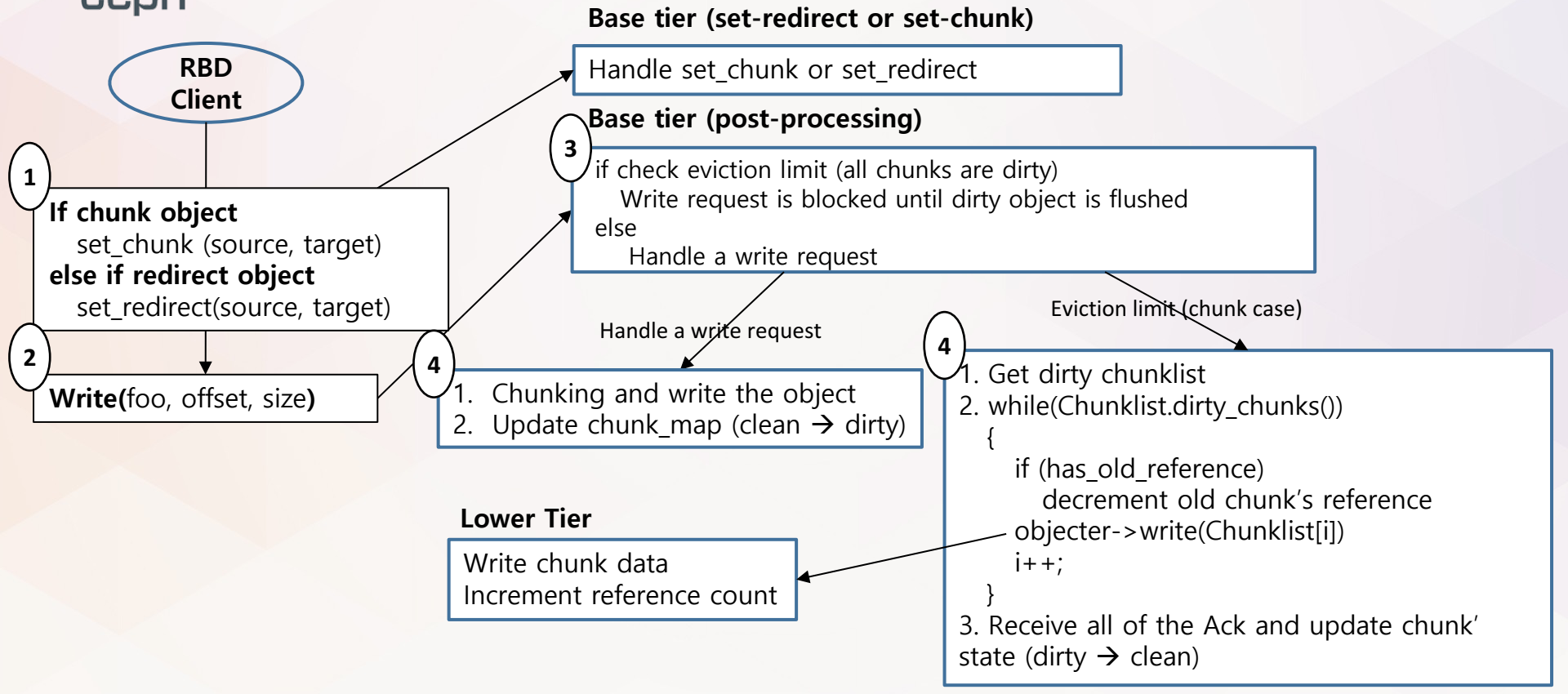


- **The key structure for extensible tier**

```
struct object_manifest_t {
        enum {
           TYPE_NONE = 0,
           TYPE_REDIRECT = 1,
           TYPE_CHUNKED = 2,
           TYPE_DEDUP = 3,
        };
        uint8_t type;  // redirect, chunked, ...
        ghobject_t redirect_target;
        map <uint64_t(offset), chunk_info_t>
chunk_map;
};
```

- **Operations**
  - Proxy read, write
  - Flush, promote

Implementation: write path



Upstream

- Proposal
  - http://marc.info/?l=ceph-devel&m=148172886923985&w=2
- Design
  - http://marc.info/?l=ceph-devel&m=148646542200947&w=2
  - Pad document (with Sage Weil)
    - http://pad.ceph.com/p/deduplication_how_dedup_manifists
    - http://pad.ceph.com/p/deduplication_how_do_we_store_chunk
    - http://pad.ceph.com/p/deduplication_how_do_we_chunk
    - http://pad.ceph.com/p/deduplication_how_to_drive_dedup_process
- Progress
  - osd,librados: add manifest, redirect https://github.com/ceph/ceph/pull/14894
  - osd,librados: add manifest, operations for chunked object https://github.com/ceph/ceph/pull/15482
  - osd: flush operations for chunked objects https://github.com/ceph/ceph/pull/19294
  - osd, librados: add a rados op (TIER_PROMOTE) https://github.com/ceph/ceph/pull/19362
  - WIP: osd: refcount for manifest object (redirect, chunked) https://github.com/ceph/ceph/pull/19935

Plan & Issues

- Plan (To Do)
  - Reference counting methods and data types for redirect and chunk
  - Offline fingerprinting and then storing of dedup chunked manifest (whole object or parts of it)
  - Dedup processing
    - Background dedup worker
    - Refcount manager and methods for dedup (http://pad.ceph.com/p/deduplication_how_do_we_store_chunk)
      - Fixed-sized backpointers
      - Scrub
  - Test cases

- Issues
  - Small chunk (< 64 KB)
  - Minimizing performance degradation
  - Dedup methods (inline?)
  - CDC (contents defined chunking)