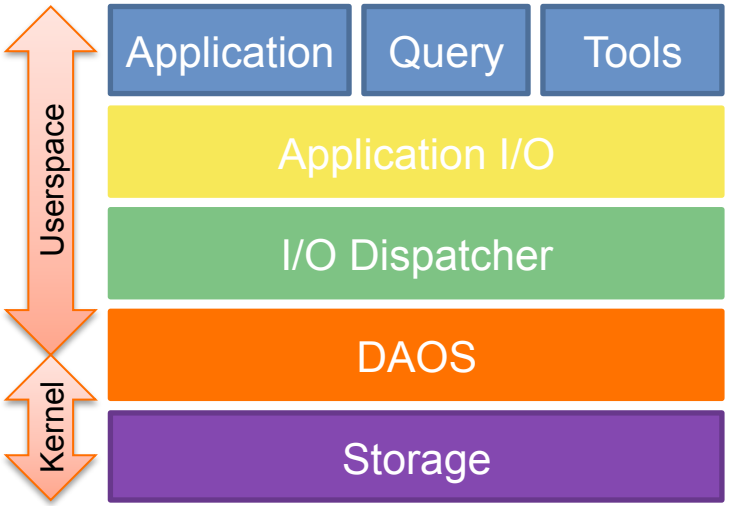
DAOS Changes to Lustre*

High Performance Data Division

Johann Lombardi & Liang Zhen

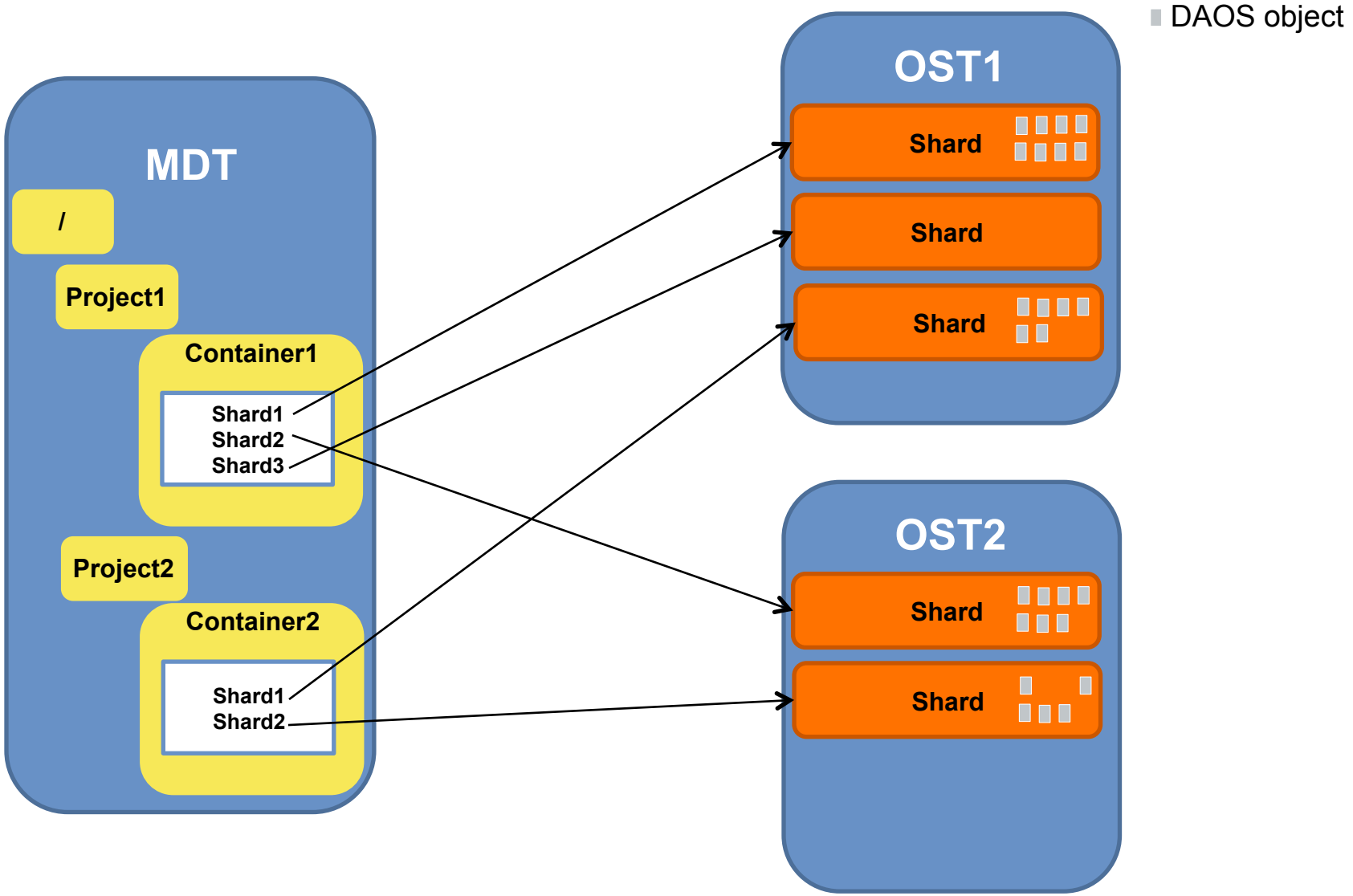
April 17, 2013

DAOS API Overview



- Distributed Application Object Storage
  - Replacement for POSIX
  - Event-based API
    - all operations are asynchronous
  - Distributed ACID transactions (epoch)
- Container
  - special file in a the POSIX namespace
  - can only be changed/accessed via the DAOS API
    - only stat(2) & unlink(2) return valid information
  - contains any number of shards
- Shard
  - virtual storage target
    - typically a new dataset/subvolume
  - maintain a collection of objects

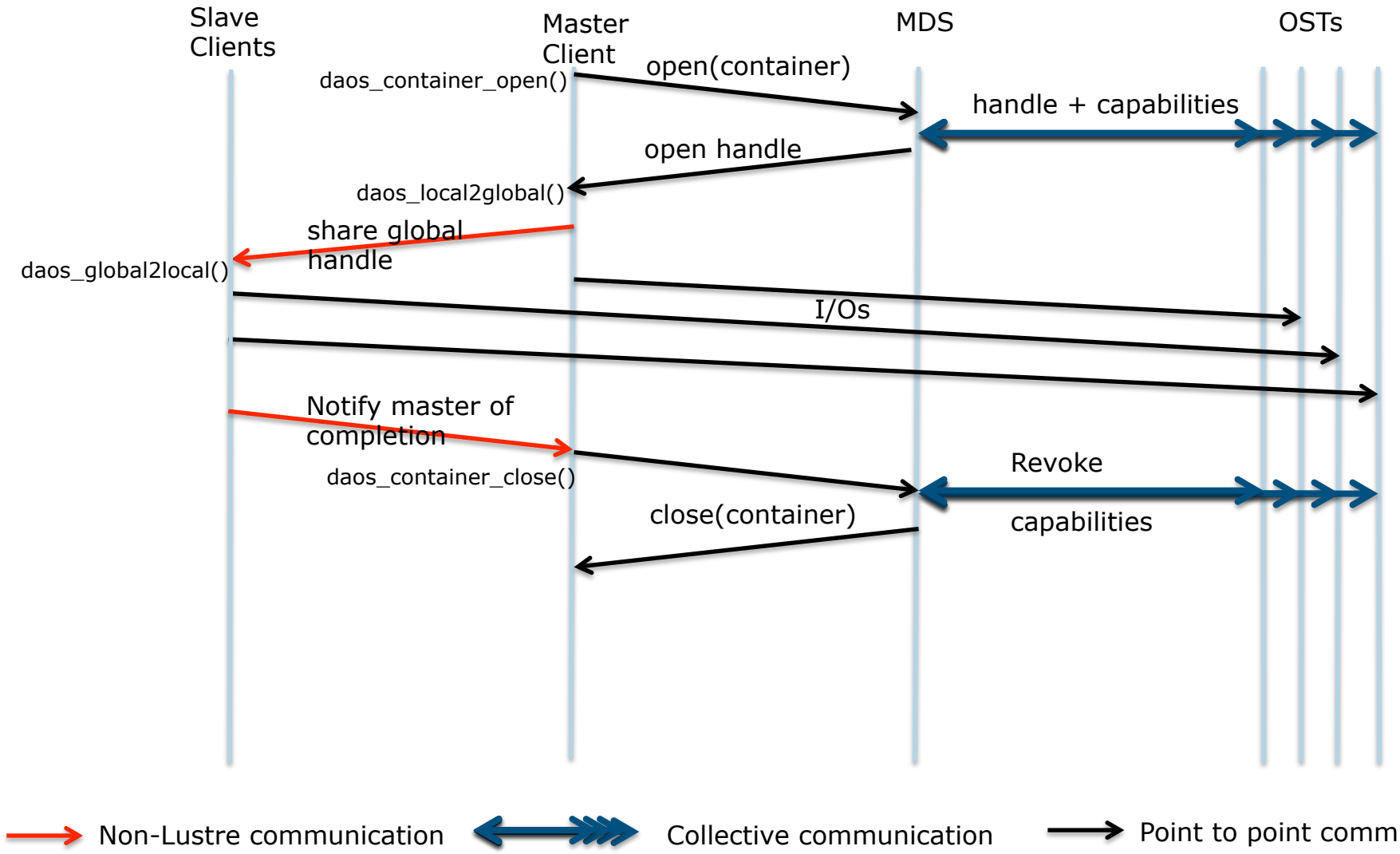
Container & Shard



Container Operations

- daos_container_open/close()
  - get/release a container handle
  - collective open/close
- daos_container_unlink()
  - destroy a DAOS container
  - All the shards associated with the container are also destroyed
- daos_container_query()
  - fetch container information
    - e.g. #shards, highest committed epoch, ..

Collective Open



Shard Operations

- daos_shard_add()
  - create a new shard
  - update layout (including all copies)
  - transfer capability list to new shard
- daos_shard_disable()
  - mark a shard as disabled in the layout
- daos_shard_list_obj()
  - Parse list of non-empty objects in a shard
- daos_shard_query()
  - fetch placement information, number of objects, …

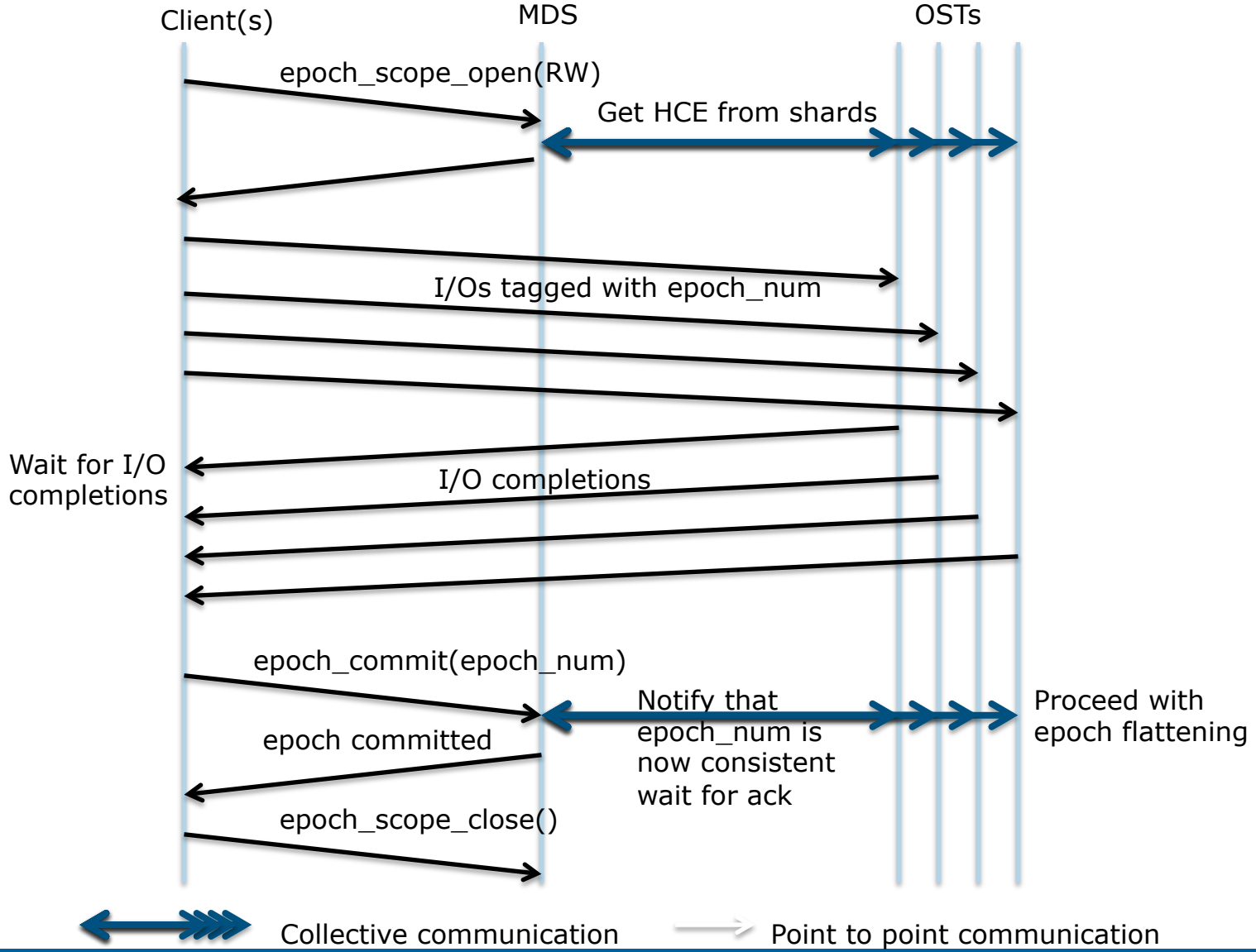
DAOS Objects

- No explicit create/destroy
  - assumes all object exist
  - objects are actually created on write (CROW)
  - objects have an infinite size
- daos_object_write()
  - write into a DAOS object in a given epoch
  - epoch value can be anything larger than the Highest Committed Epoch
- daos_object_read()
  - read DAOS object content from a committed epoch
  - read from unwritten objects/extents returns zeroes
- daos_object_punch()
  - discard data range

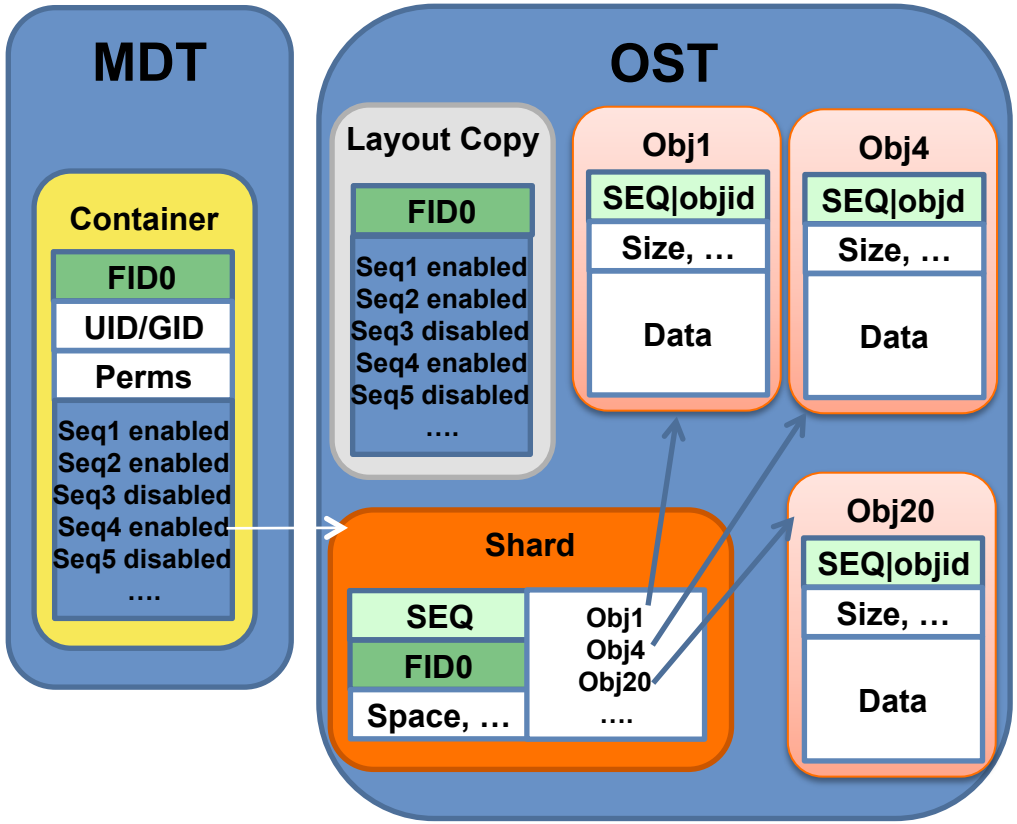
Epoch API

- Transaction identifiers passed in all DAOS I/O operations
  - readers can only read from already committed epochs
  - writers can only write to not-yet-committed epochs
- Epochs are per-container
  - may cross several containers in the future, notion of epoch scope
  - daos_epoch_scope_open/close
- Epochs are totally ordered
  - become persistent only after all prior epochs are persistent
  - explicit commit from library user when all writes completed & flushed
    - daos_epoch_commit()
  - Highest Committed Epoch (HCE)
- daos_epoch_slip/catchup

Epoch Overview

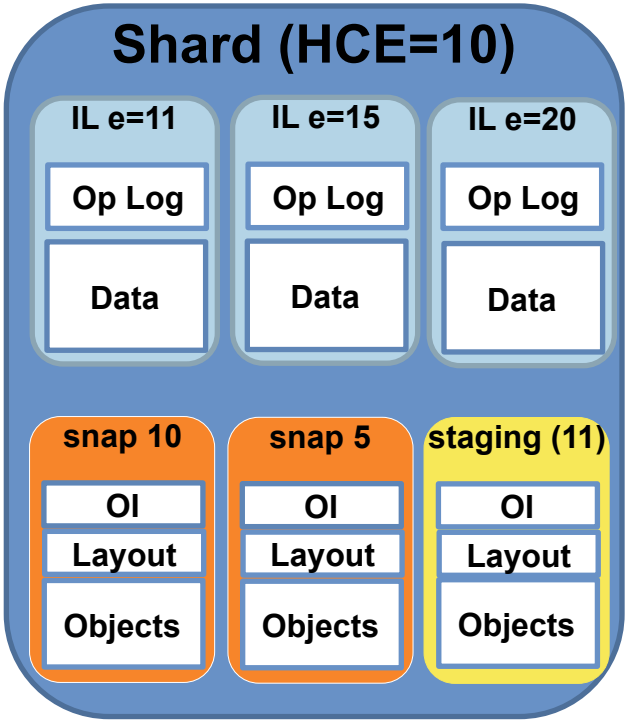


Container & Shard Representation



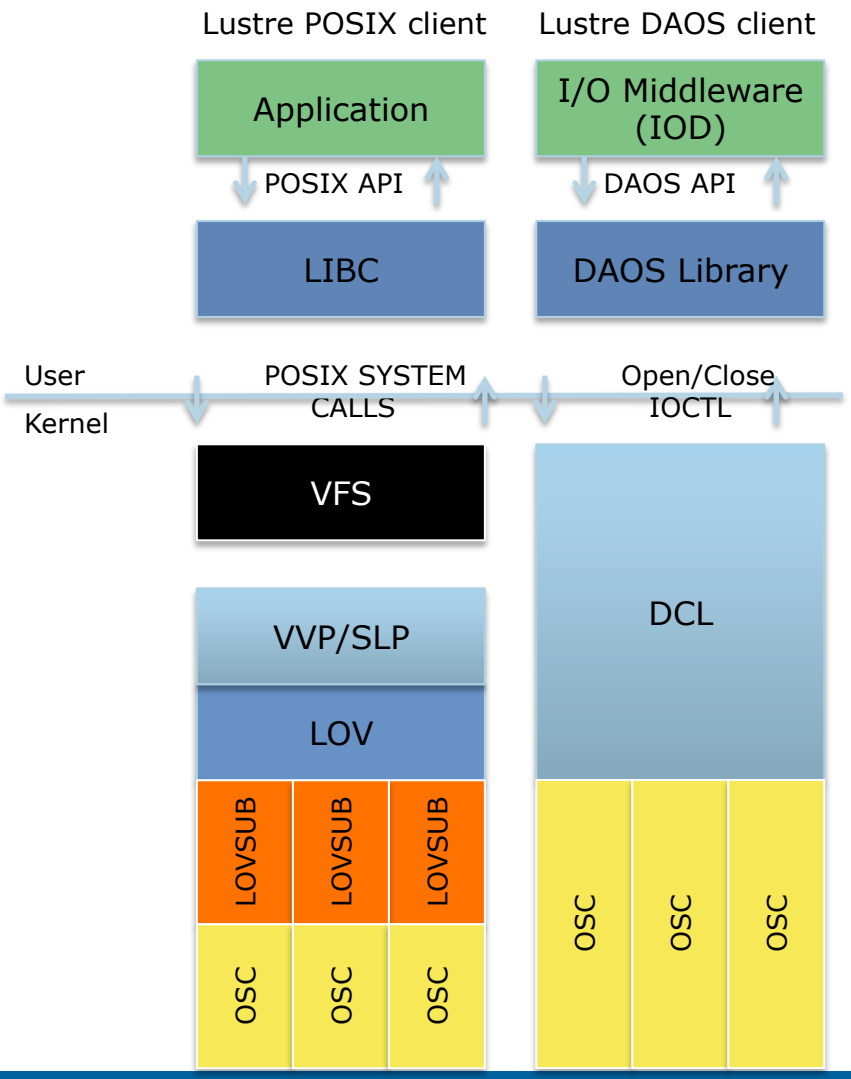
- MDT FID associated with container
- Each shard is assigned a FID sequence (SEQ)
- Container layout stores list of sequences
  - Layout is replicated on all the shards
- Each shard maintains its own object index

Shard on-disk Representation

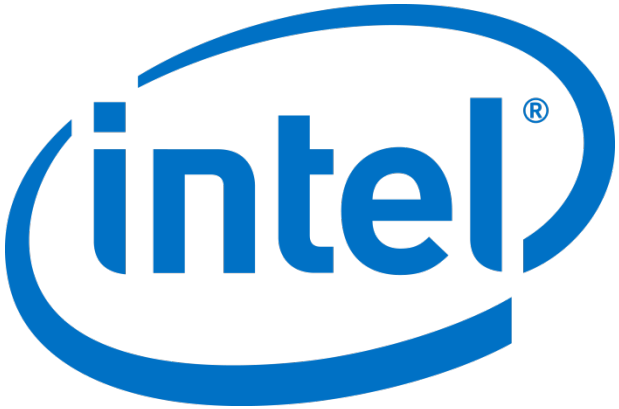


- Considered btrfs, but finally chose ZFS
- Intent logs (IL)
  - operation log (objid, offset, length, …)
  - log file storing data blocks from writes
  - intent log flattening
    - Parse intent log and execute operations
- A shard is represented by multiple ZFS datasets
  - root dataset storing intent logs and configuration
  - staging dataset where flattening takes place
  - HCE snapshots
  - one LU stack per dataset
- Extensions to OSD API and shard-aware OFD (SFD)

DAOS Library / Lustre Client Interface



- DAOS I/O path quite simple
  - no locking & no striping
  - does not require CLIO complexity
  - DAOS Client (DCL) sits directly on top of OSCs
- Also LMV/MDC changes to support container & epoch
- All operations are asynchronous
  - POSIX AIO not generic enough
  - own event & event queue mechanisms
  - relies on ptlrpcd


intel