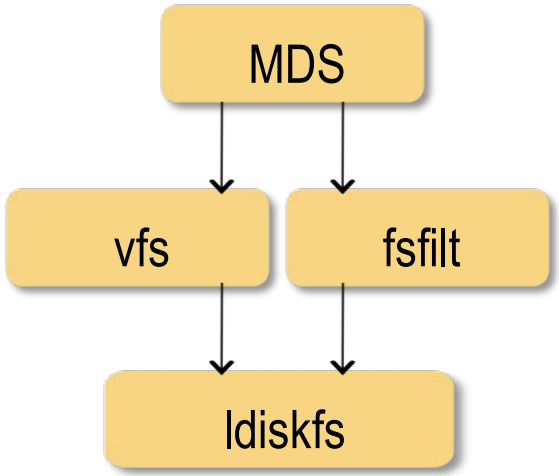
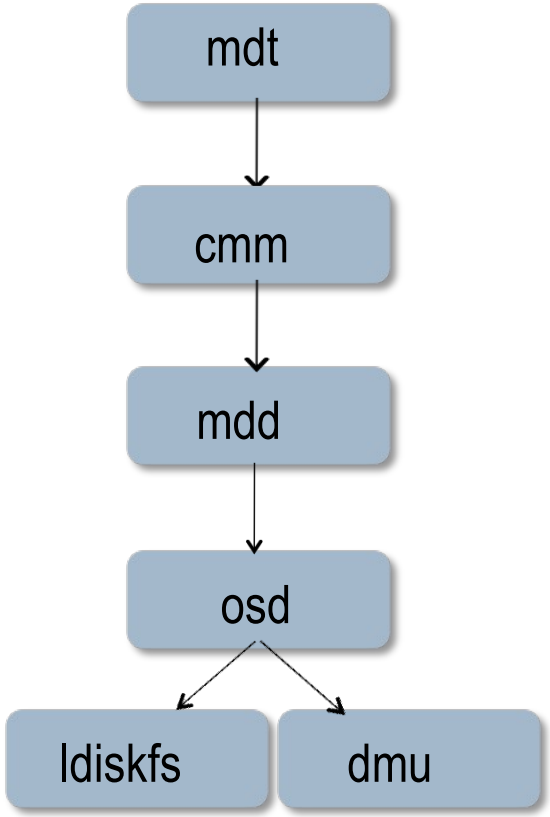
MDT Stack

Nikita Danilov
Staff Engineer
Lustre Group

Old (1.4-1.6) and new (HEAD) MDT





- fids
- md stack rewrite
- Integration of new features
- Don't lose performance

Aside: fids

- 1.6: "storage cookie" (MDS ino)
  - Fid: not bound to the specific MDS
  - Fid: is never reused.
  - Fid: **can be generated by client**.

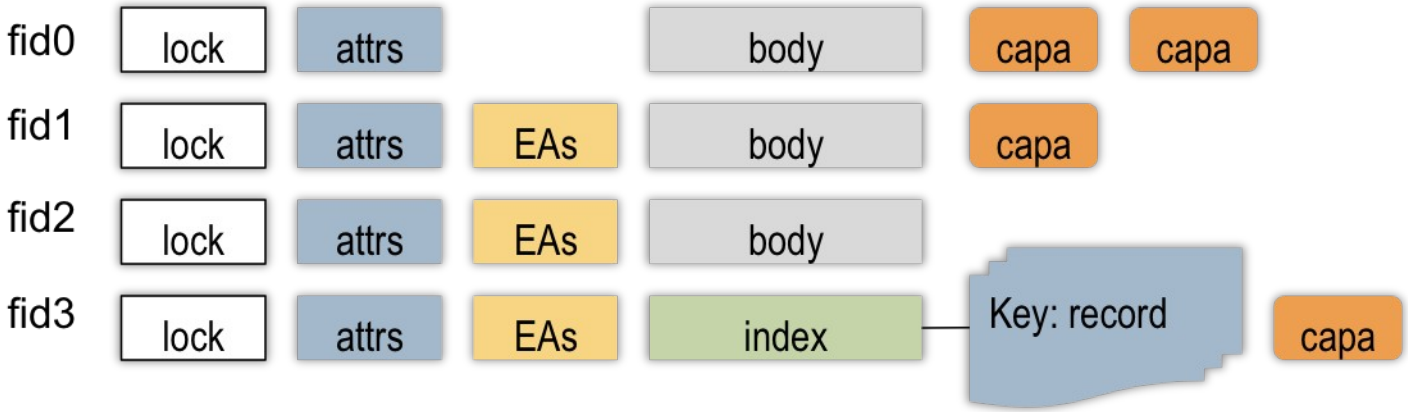
Problems:

- how clients generate fids locally without colliding with each other?
- how to locate a server where fid lives?
- how a server locates an object?

Answer:

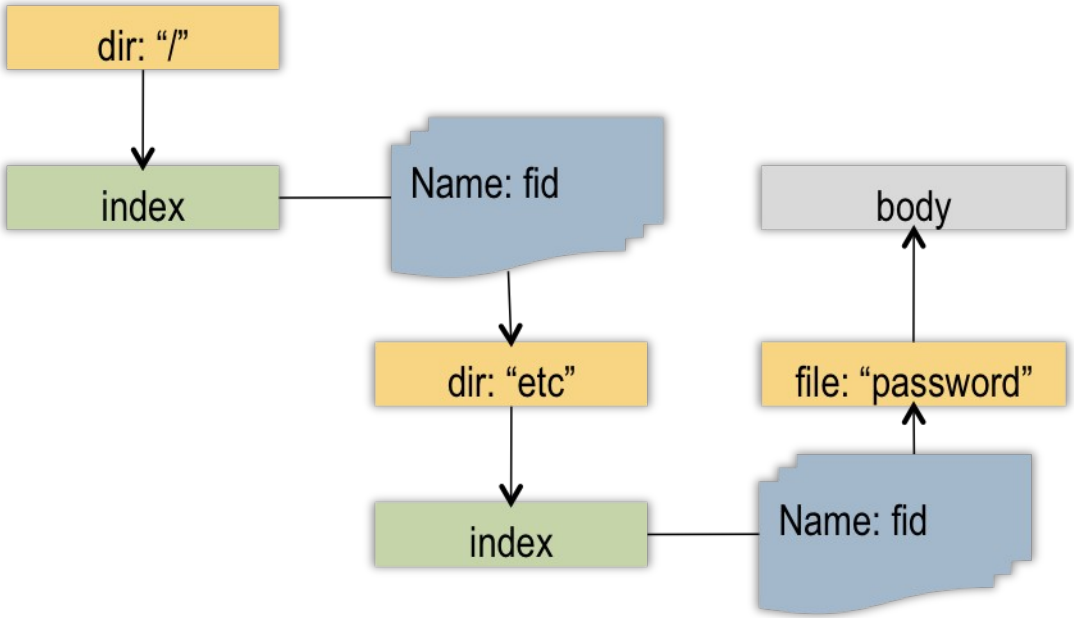
- seq; fld; object index.

osd: object storage device



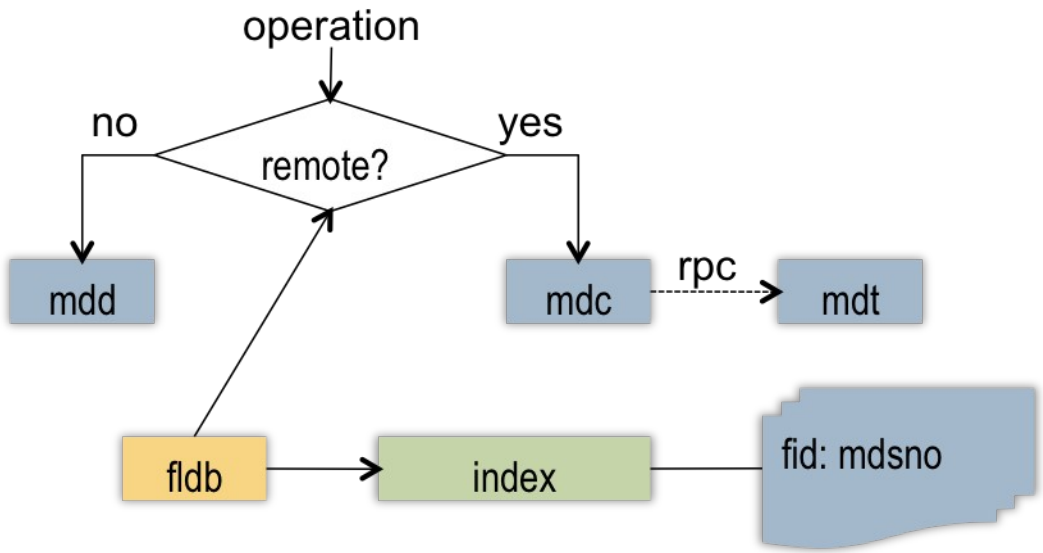
- fid-addressable objects
- Body (read, write, truncate)
- Fixed set of attributes
- Extended attributes
- Indices
- Object lock, transactions, capabilities

mdd: meta-data device



- Scalable directories (lookup, (un)link, readdir)
- Permission checks
- acls
- Orphan handling

cmm: clustered meta-data module



- Remote operations
- Partial operations
- Split directories
- fldb data-base

mdt: meta-data target

- Request packing and unpacking
- Replies
- Recovery
- dlm
- intents
- ptlrpc services, threads
- Reintegration

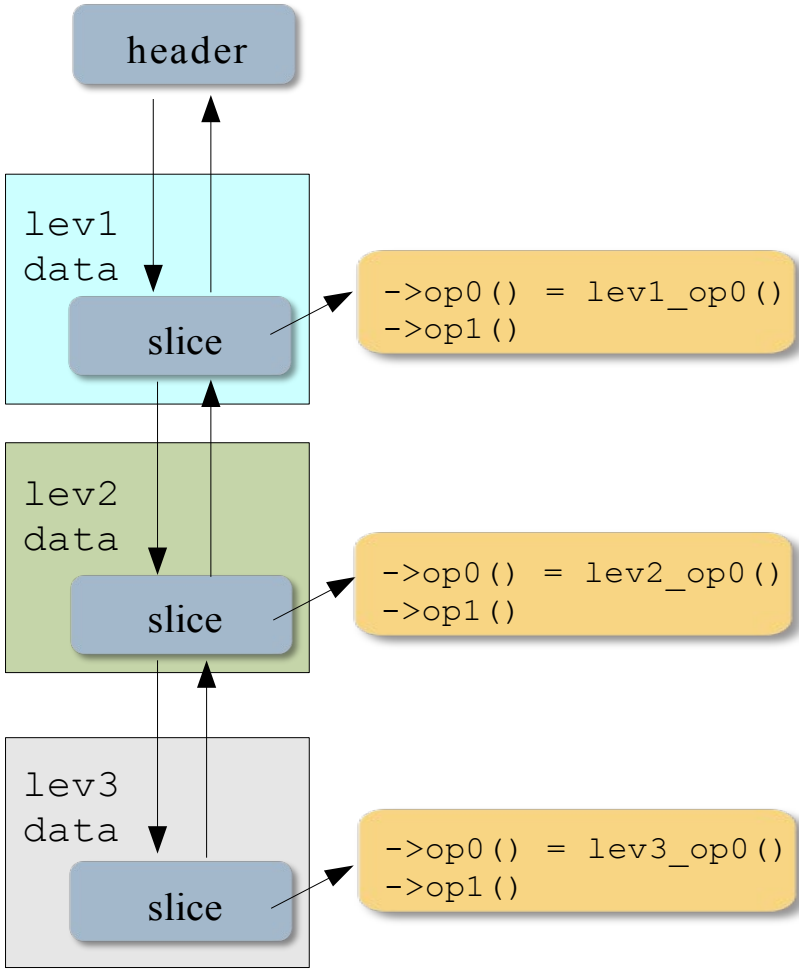
An example: file creation

- client: CREATE(pfid, "foo", cfid) rpc
- mdt:
  - create objects for pfid and cfid
  - take dlm locks
- cmm
  - execute remote object creation (mdc...mdt)
  - insert name locally
- mdd
  - call osd to insert a record (cfid) with a key (“foo”) into index, associated with pfid
- osd
  - insert (key, record) into an index (iam, zap)

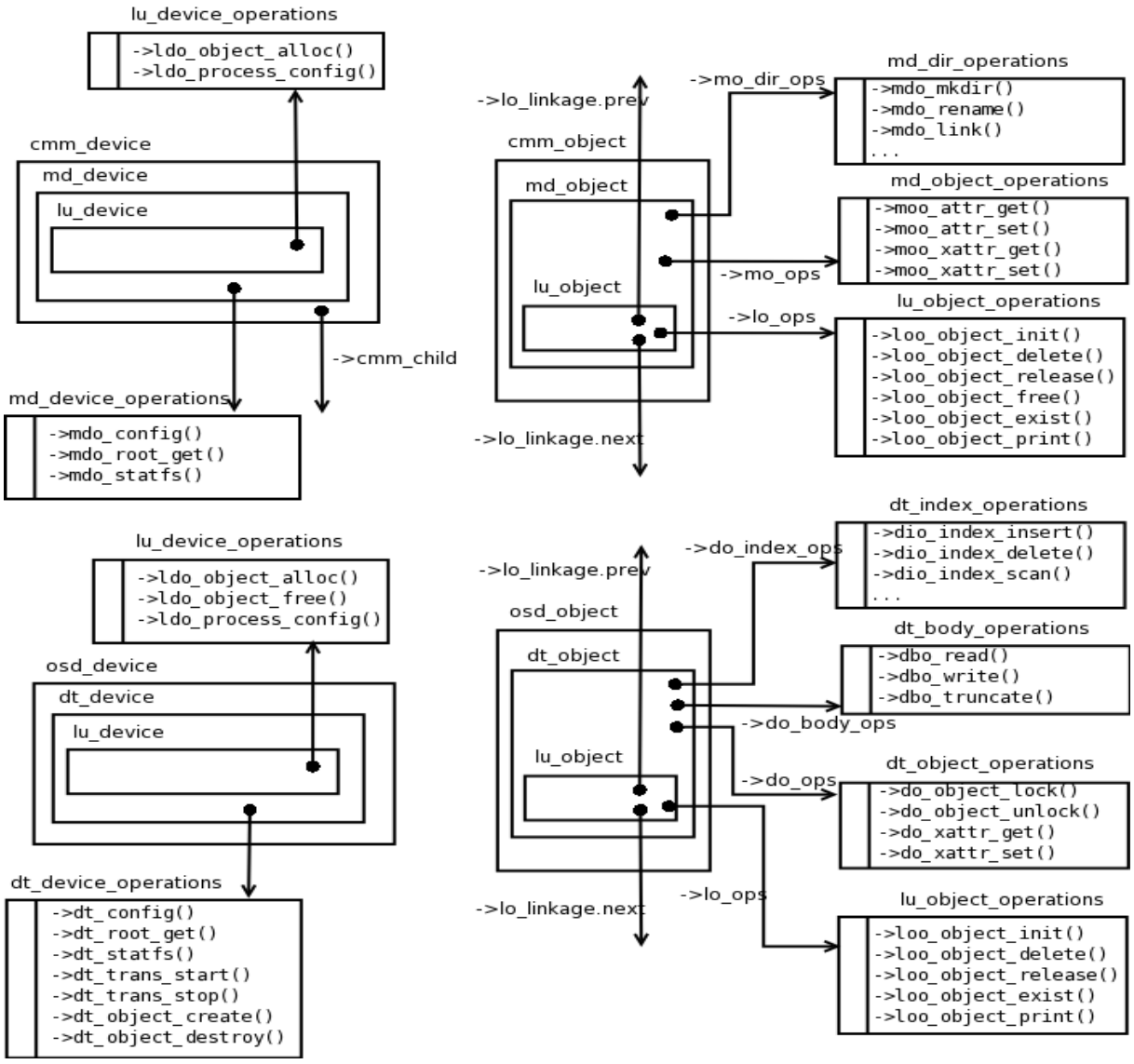
Nikita.Danilov@sun.com

Layered objects

- Compound object: a header and a sequence of layers
- Layer-private state
- Per-layer operation vectors
- Generic code invokes operations on each layer, delegating behavior



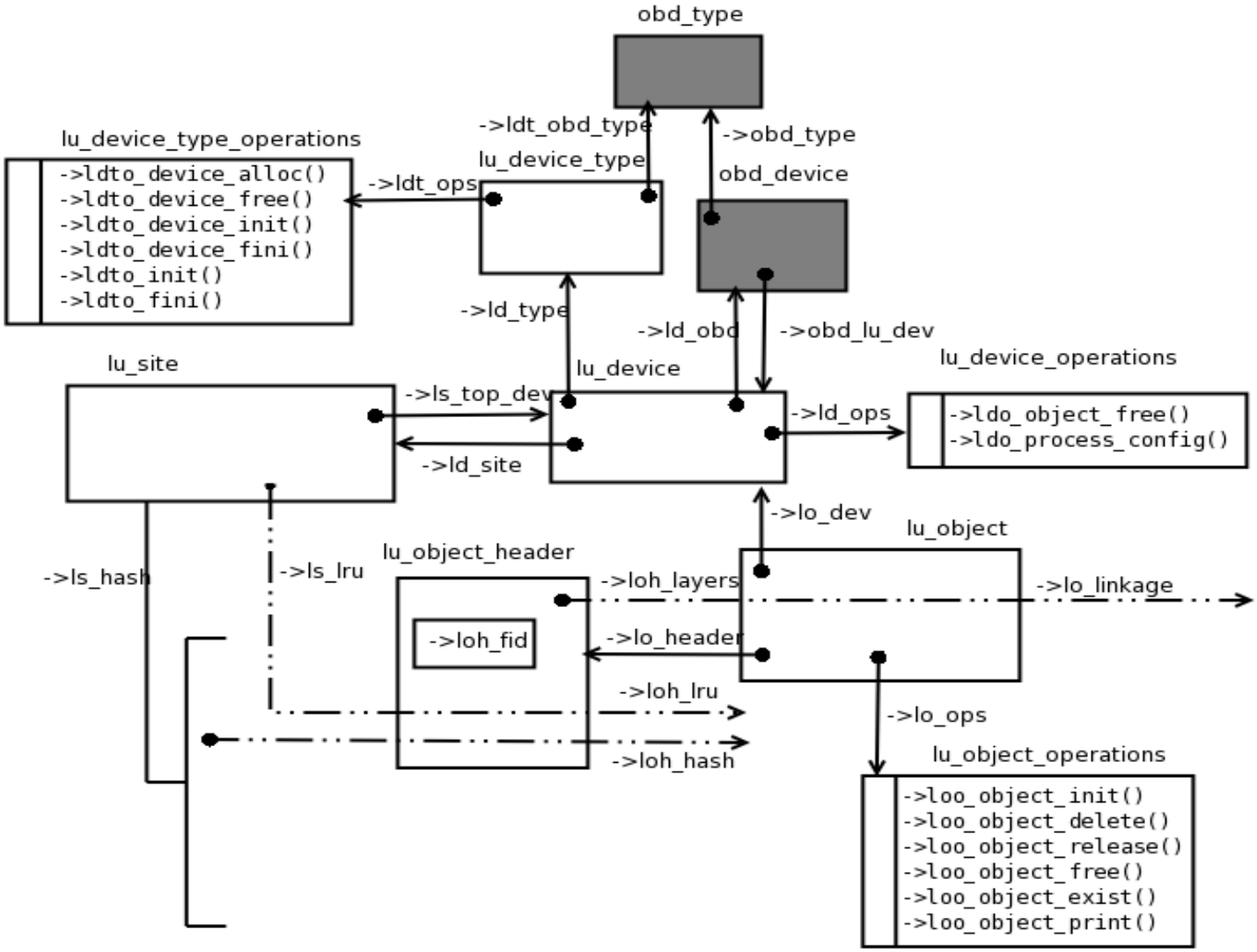
Operation Vectors



Layering

- Layer interfaces:
  - lu: all devices: creation, configuration, destruction
  - md: mdt, cmm, mdd: meta-data operations
- Layer data-types:
  - device type: setup, configuration
  - device: mdt_device, osd_device
  - object: cmr_object, cml_object, osd_object

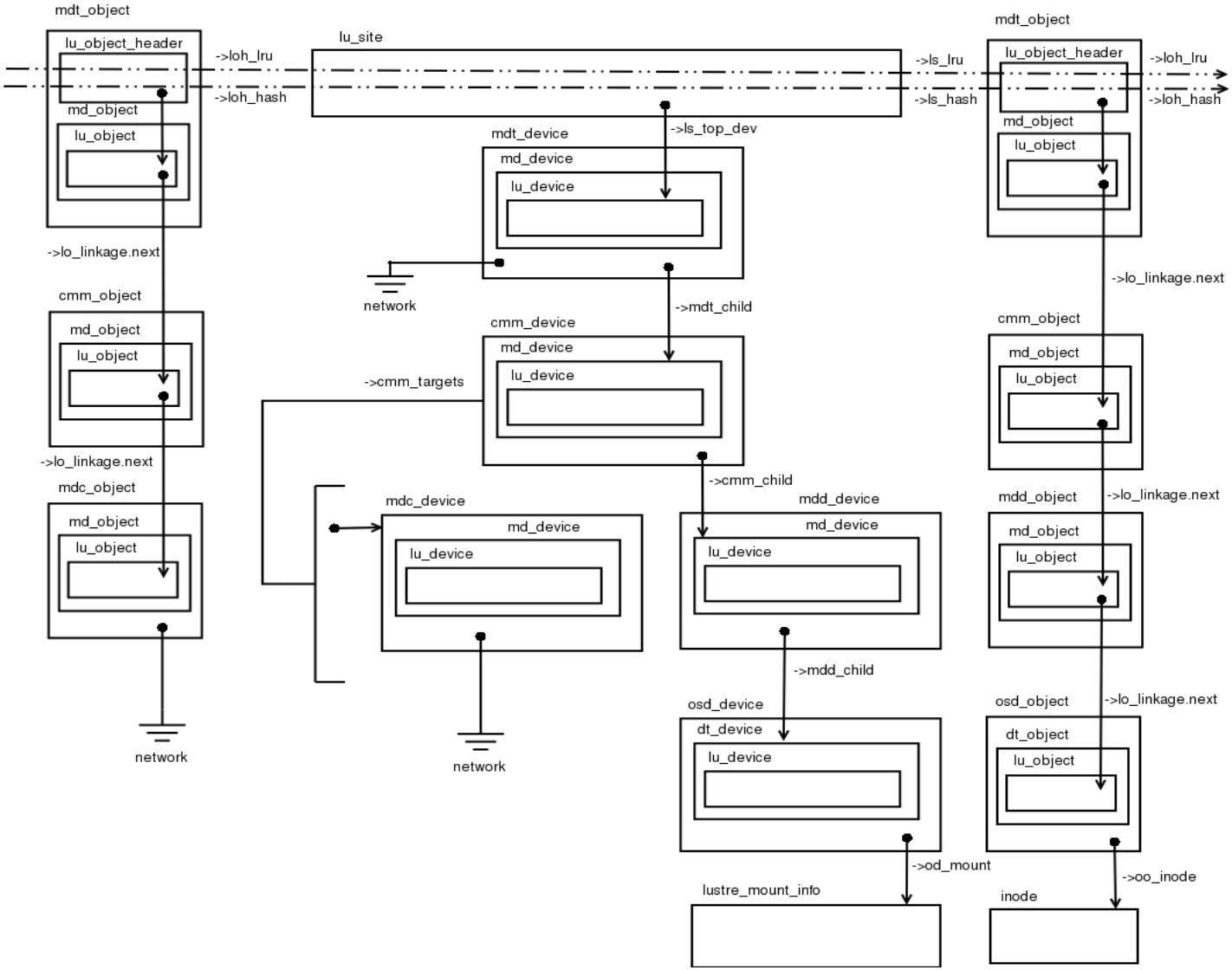
data-types


obd_type
->ldt_obd_type
->obd_type
lu_device_type_operations
->ldto_device_alloc()
->ldto_device_free()
->ldto_device_init()
->ldto_device_fini()
->ldto_init()
->ldto_fini()
->ldt_ops
lu_device_type
obd_device
->ld_type
->ld_obd
->obd_lu_dev
lu_site
lu_device
lu_device_operations
->ls_top_dev
->ld_ops
->ldo_object_free()
->ldo_process_config()
->ld_site
->lo_dev
lu_object
lu_object_header
->ls_hash
->ls_lru
->loh_layers
->lo_linkage
->loh_fid
->lo_header
->loh_lru
->lo_ops
->loh_hash
lu_object_operations
->loo_object_init()
->loo_object_delete()
->loo_object_release()
->loo_object_free()
->loo_object_exist()
->loo_object_print()

Layered objects

- Unique fid
- State private for a layer
- Operation vectors at every layer
- Caching
- Indexing (hashing)
- LRU, purge

Object Stack



Nikita.Danilov@sun.com