

whamcloud

The logo for Whamcloud features the word "whamcloud" in a bold, dark grey, lowercase sans-serif font. A thick blue horizontal line underlines the text. On the right side, a blue graphic element consisting of two curved segments forms a stylized '3' or a partial circle that overlaps the end of the text and the underline.

Quota Enforcement for Orion

- Johann Lombardi

Today's Quota

- Not portable to other backend filesystems
- Quota on/off is on a per-target basis
- Changing a quota limit requires all slaves to be up and running
- Qunit broadcast isn't optimal
- Can't deal with OST addition
- No proper way to decommission a dead OST
- Master recovery requires all targets to be up and running
- Full quotacheck required after e2fsck

A few words on space accounting ...

- ZFS **permanently** tracks per-uid/gid disk usage
 - Even when there is no quota limit enforced
 - Only #blocks and not #inodes
- Same scheme adopted with ldiskfs
 - Quota as a new core ext4 feature
 - mkfs.lustre/mke2fs creates empty quota files
 - e2fsck can now fix quota files
- End of quotacheck
- Quota on/off **only** enables/disables **enforcement**
- Let's talk about enforcement in details now ...

Architecture Primer

- New Slave->Master connection
- Leverage the proven scalability of the LDLM for quota communications
- Master tracks on-disk quota space distribution
- Setquota can be issued with missing slaves
- Efficient handling of OST addition
- Huge fraction of the quota space granted to slaves initially
- Quota enforcement on/off managed globally
- DNE support
- Allow per-pool quota in the future

Slave - Master connection

- Tracking reverse MDT import on OST is a pain
- Cannot enqueue locks on reverse import
- Orion's OSP hides the connection
- FIDonOST needs a connection to MDT0 too

- New connection set up from slave to master
- Master still has to run on MDT0 for now
- Slaves can now enqueue locks ...

Quota Locks

- New class of locks to manage resources allocated to clients
 - Quota, grant, locks, permission to send RPCs, ...
 - New DLM namespace and lock type
 - Each component (i.e. quota, grant ...) is assigned a range of LDLM resource IDs
 - One step towards unification of grant and quota
- One global quota lock, namely **quota index lock**
 - Used to distribute the list of IDs having a quota limit
- Plus **per-ID quota locks**
 - Must be acquired by slave in order to hold unused quota space for that ID

Quota Index Lock

- Slaves enqueue one quota index lock per quota type
- Guarantee that the list of IDs with quota enforced is in sync between master & slave
- List of IDs is fetched via a bulk transfer
- Master sends glimpse callbacks to notify slaves of a new limit enforcement

Per-ID Quota Lock

- Slaves must hold the per-ID quota lock when caching unused quota space for a given ID
- Used to query/grant/cancel quota on that ID
- Master issues glimpse/blocking callbacks to claim quota space back for that ID
- Lock request packed in QUOTA_DQACQ/REL
 - If the slave does not own the quota lock for that ID, it packs the lock enqueue request in the DQACQ/REL RPC
 - If the slave already owns a lock for that ID, DQACQ/REL RPC just packs the lock handle
 - One exception is DQACQ requests issued after setquota to report initial usage

Quota Identifier

- Usually a 32-bit group or user ID
- Extended to support a FID
- Can be used to implement per-directory quota
 - As done by Fujitsu
- New quota_id union introduced:

```
union quota_id {  
    struct lu_fid qid_fid;  
    obd_uid      qid_uid;  
    obd_gid      qid_gid;  
};
```

Quota Space Allocation Tracking

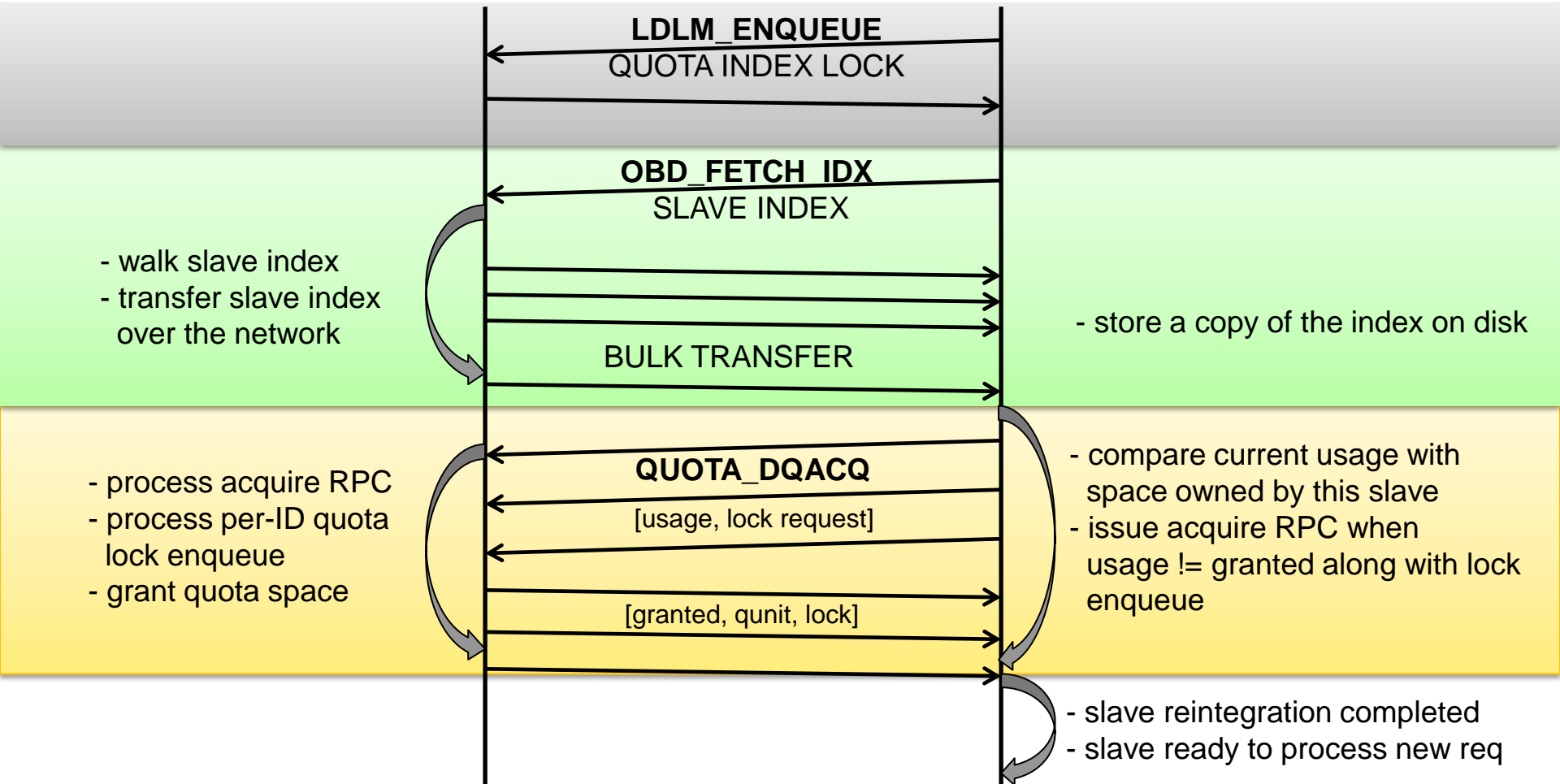
- Master now aware of the quota space distribution
- One index per slave in addition to the global index
- Global index records for each ID:
 - global soft/hard limits
 - how much space is granted in total
 - grace time
- Per-slave index tracks how much quota space is owned by the slave for each ID
- Slaves fetch their dedicated index from the master just after enqueueing the quota index lock
- Master's indexes are the reference
 - Slaves' on-disk copy of the index is just a cache, crushed each time the quota index lock is re-acquired

Slave (re)Integration Procedure

- Replace existing quota recovery
- #1 slave enqueues the quota index lock
 - After this point, the slave will be notified of IDs added/removed to/from the enforced list via glimpse callbacks
- #2 slave fetches their private index via a bulk transfer
 - That's the current list of IDs subject to a quota limit
 - This index also includes how much space is granted to this slave for each ID
- #3 slave re-acquires quota space
 - If current usage == granted, no need to send any acquire RPC
 - If current usage != granted, send a QUOTA_DQACQ RPC with lock enqueue packed (the master might grant us back more than usage)
 - This way, only locks for "active" IDs are replayed

Slave (re)Integration

Master Slave

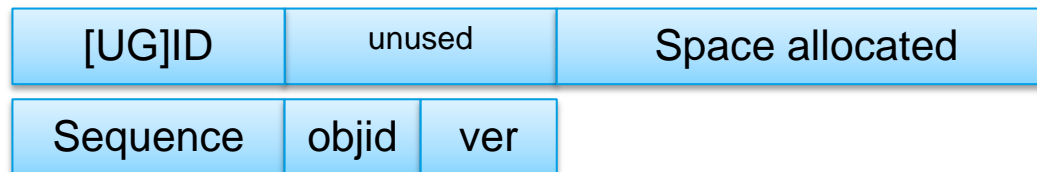


Index Network Transfer

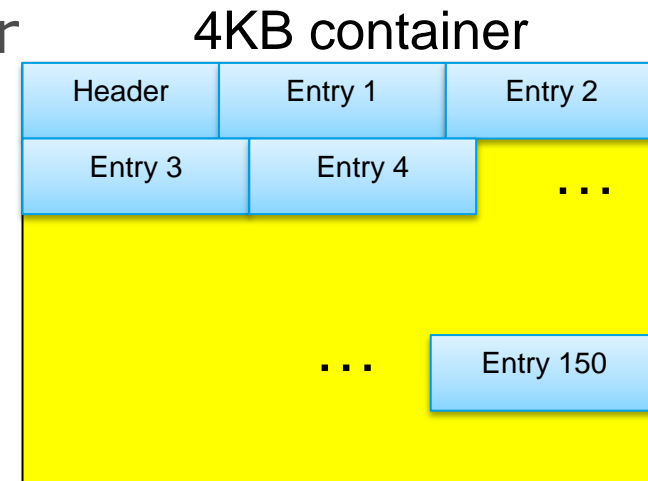
- Generic mechanism for reading index files over the network
 - Serialize the index into a byte stream on the “server”
 - Deserialize it on the “client”
- Only used to transfer the per-slave index from master to slave for the time being
- Probably many other use cases in the future
 - Quota master migration
 - Repquota
 - Directory split
- Credits to Andreas

Index Transfer Format

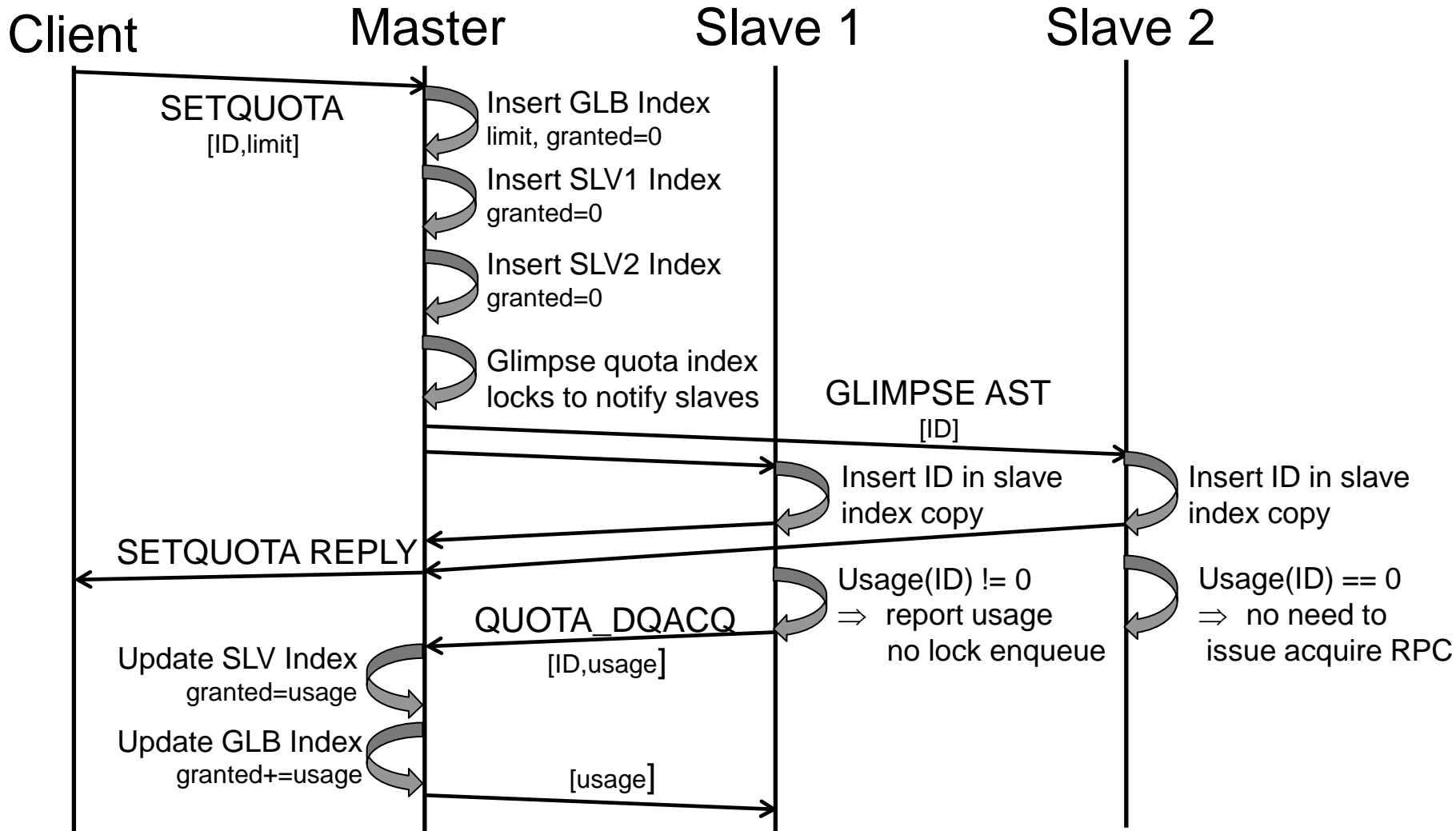
- Compact record format: 24-byte



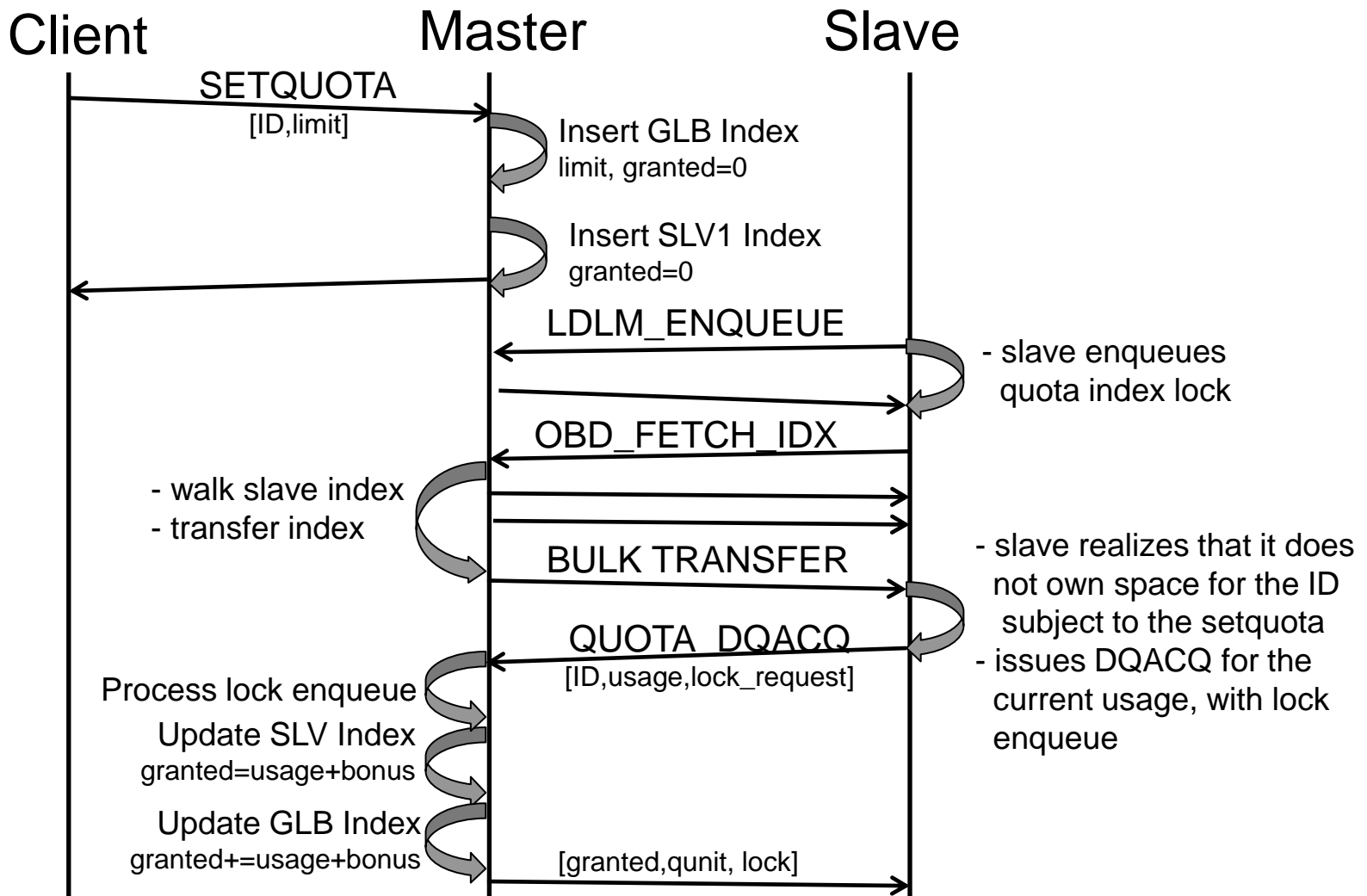
- Union between 16-byte FID and 8-byte [UG]ID
- 8-byte space representing the amount of quota space allocated to the slave for this ID
- Records are grouped in a container
 - Independent of page size, always 4KB
 - Header with magic, flags, format version, #entries and padding, resp.
=> $4+4+1+1+6 = 16$ bytes
 - 170 IDs per 4K container
 - 43,520 IDs in a 1MB bulk transfer



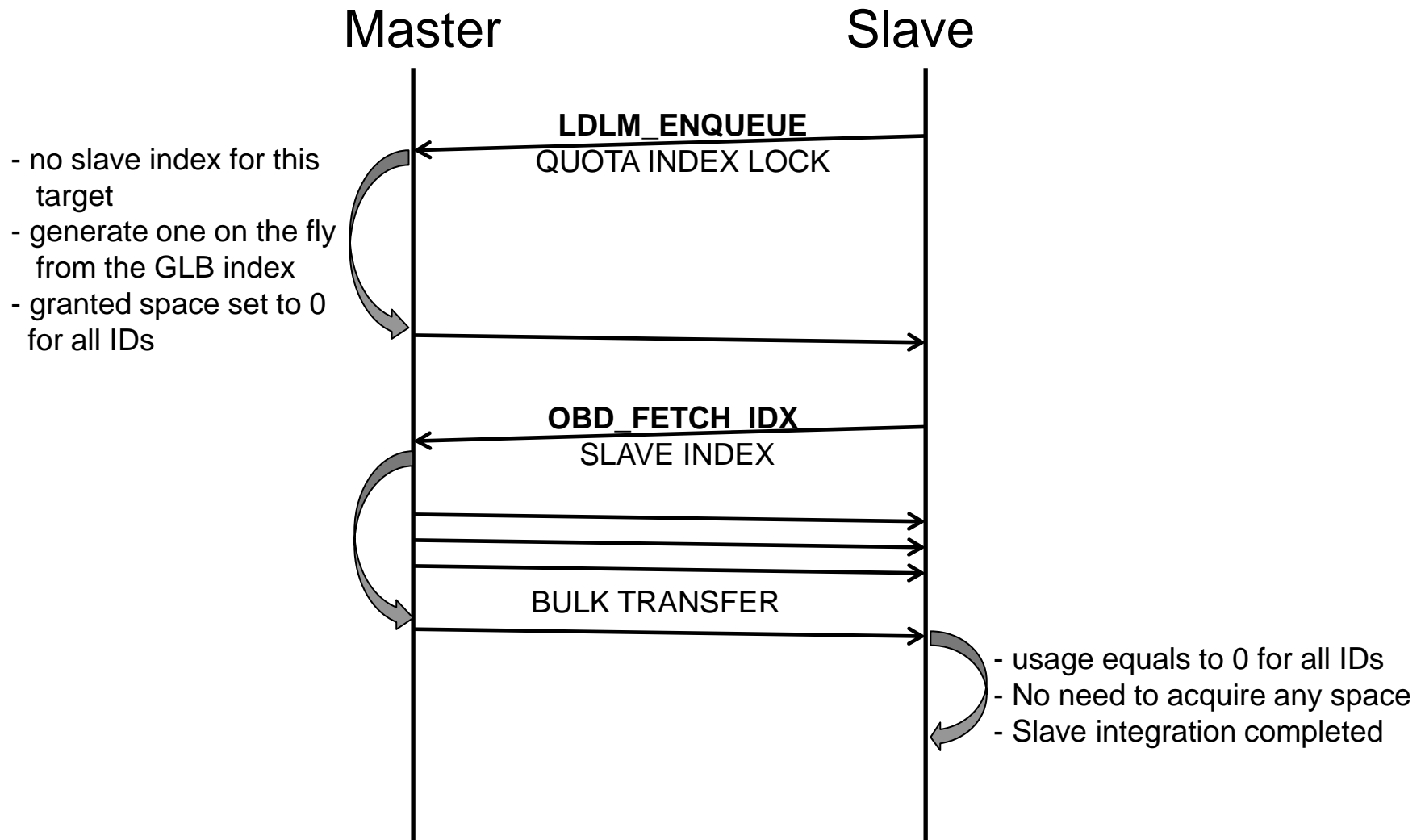
Setquota with slaves connected



Setquota with disconnected slaves



Online OST/MDT Addition



Granting more to Slaves

- Quota space allocated to slave on first write/create
- Huge fraction of the quota space is granted
 - Less DQACQ RPCs
 - More autonomy to slaves
 - Mitigate space overestimation issue with ZFS
- Possible thanks to a reliable quota space revocation mechanism based on the LDLM
- Connected slaves that potentially owned unused quota space for a given ID must have a per-ID quota lock
- Master issues glimpse or blocking callbacks on the per-ID quota locks
- The master can now be more selective since it is aware of the quota space distribution

Quota Space Rebalancing

- **Glimpse callbacks** sent on the per-ID locks to ask slaves to release a fraction of the unused quota space, if any
 - Replace the blind qunit broadcast
 - OST_QUOTA_ADJUST_QUNIT RPC not used any more
- Slaves release space in glimpse reply
 - Information packed in a new quota LVB
- **Blocking callbacks** sent as a last resort to claim all unused quota space back

New layering

- MDD/obdfilter don't deal with quota any more
- Space estimated in OSD layer
- Routines handling enforcement called directly from OSD layer
- Quota space acquired in ->declare
 - DQACQ sent while transaction isn't started
- Released at transaction stop time

Cascading timeouts

- Client's RPC processing might be stuck waiting for DQACQ RPC to master to complete
- Waiting for too long might cause the initial RPC to time out
 - client has to reconnect and resend all RPCs in flight, painful
- Client nodes can get evicted if lock cancellation underway
- Current quota code drops the reply
 - Quota not the only one to do that, check `ost_brw_read/write()`
- **-EINPROGRESS** now returned to clients
 - Client should retry indefinitely
 - Lock timeout will be extended thanks to HP request handling
 - New connect flag to detect clients that support EINPROGRESS

Slave Eviction

- Can happen when glimpse callback on quota locks not acknowledged in a timely manner
 - Might impact other services using this connection
- From the master perspective, the slave has gone “disconnected”
 - new ID can be added/removed to/from the slave index w/o issuing glimpses
 - space reserved by this slave cannot be claimed back
- From the slave point of view
 - Quota locks must be re-enqueued ASAP
 - Meanwhile, continue to operate with the on-disk copy
 - If one ID runs out of local quota space, requests are failed with -EINPROGRESS
 - Once the lock is requeued, all in-memory structures & on-disk index are cleaned up and recreated

DNE support

- Inode quota managed in the same way as block quota
- All MDTs are slaves
- EINPROGRESS support to be extended to metadata
- Metadata targets no longer acquire block quota space and only deal with inode limit
- No inode quota with ZFS OSD