Quota on HEAD TOI

yong.fan@sun.com
Lustre Group
Sun Microsystems, Inc.

Motivation

- HEAD is original from CMD project which did not support quota
  - Porting quota from lustre 1.6 to HEAD with new MD stack on MDS
  - Security support for quota on HEAD
- Bz# 13058

Lquota overview

- Distributed
  - Only one quota master on MDS
  - One quota slave per MDS and OSS
- Quota master
  - Global quota management
  - Assign/revoke quota to/from slave
  - Does not directly participate in file operation related quota process
  - Interface for quota control/query from client

Lquota overview (cont'd)

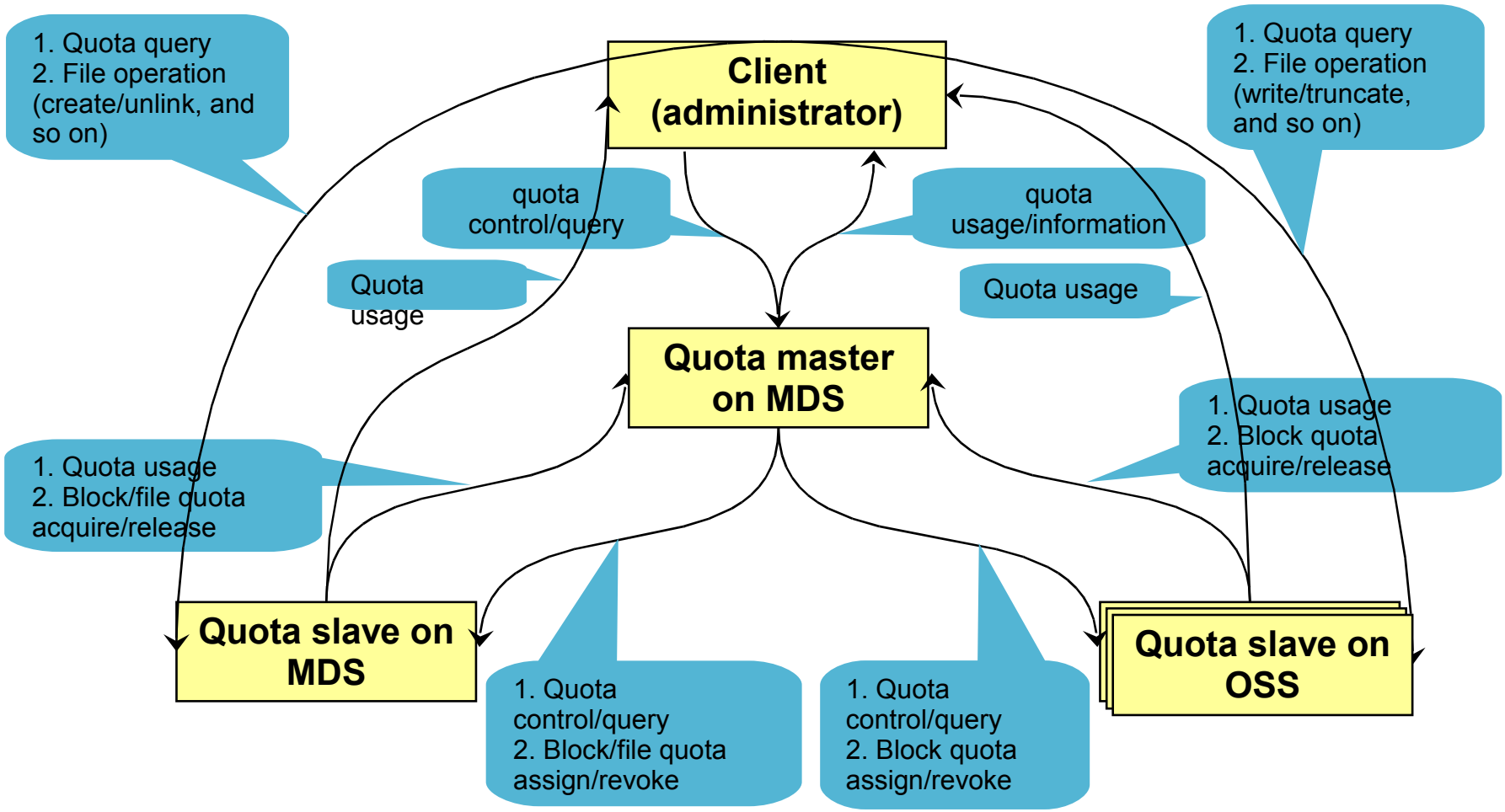
- Slave require/release quota from/to master
- Slave on MDS is the real maintainer of user/group metadata block/file limit/usage on this MDS

Metadata block: directory context, extended attribute, symbol link context

- Slave on OSS is the real maintainer of user/group block limit/usage on this OSS

Both data block and metadata block are counted as block usage on OSS

Lquota framework



Lquota utilities

- Turn on quota
  - lfs quotaon [-ugf] <fs>
- Turn off quota
  - lfs quotaoff [-ug] <fs>
- Recalculate disk space usage
  - lfs quotacheck [-ug] <fs>
- Set quota
  - lfs setquota [-ug] <name> -b <block-softlimit> -B <block-hardlimit> -i <inode-softlimit> -I <inode-hardlimit> <fs>
  - lfs setquota -t [-ug] -b <block-grace> -i <inode-grace> <fs>
- Query quota
  - lfs quota [-v] [-o obd_uuid | -i mdt_idx | -I ost_idx] [{-u | -g <name>} | -t] <fs>

Client side change

- LMV is introduced for CMD support, the abstract layer to control lower multi-MDCs
  - In 1.6

    LLITE=>MDC
  - In HEAD

    LLITE=>LMV=>MDC
  - lmv_quota_interface

```
quota_interface_t lmv_quota_interface = {
        .quota_ctl   = lmv_quota_ctl,
        .quota_check = lmv_quota_check,
};
```

MDS side change (1)

- New MD stack layers of "MDT=>CMM=>MDD=>OSD" replace the original single "MDS" layer.
- "MDS" layer is used for the communication with OSS server: "MDT=>CMM=>MDD=>MDS=>LOV=>OSC".
- Reverse the shared LQUOTA and LVFS quota interfaces as "MDS" obd_device's "obd_fsops".
- All the quota related process in original "MDS" layer are move to other layer(s).

MDS side change (2)

- Trigger quota environment process in MDT layer
  notify (mdt_obd_notify)
  setup (mdt_init0)
  cleanup (mdt_fini)
  recovery (mdt_upcall)
- RPC handler for quota control/query from client is in MDT layer
  mdt_quotacheck_handle (quotacheck)
  mdt_quotactl_handle (quotactl)

MDS side change (3)

- Quota application related with some special file operations (create, unlink, rename, chown, and so on) are reimplemented in MDD layer

```
lquota_chkquota, lquota_pending_commit, lquota_adjust
static int mdd_create(const struct lu_env *env, ...)
{
        mdd_create_sanity_check();
        lquota_chkquota();
        other create process ...;
        lquota_pending_commit();
        lquota_adjust();
}
```

- Lightweight push_ctxt/pop_ctxt are introduced in OSD layer for transferring new current user identity (md_ucred) to quota stuff

MDS side change (4)

- Disassemble quotactl (in 1.6) interface into multiple "md_quota_operations" methods, each method only perform definite function

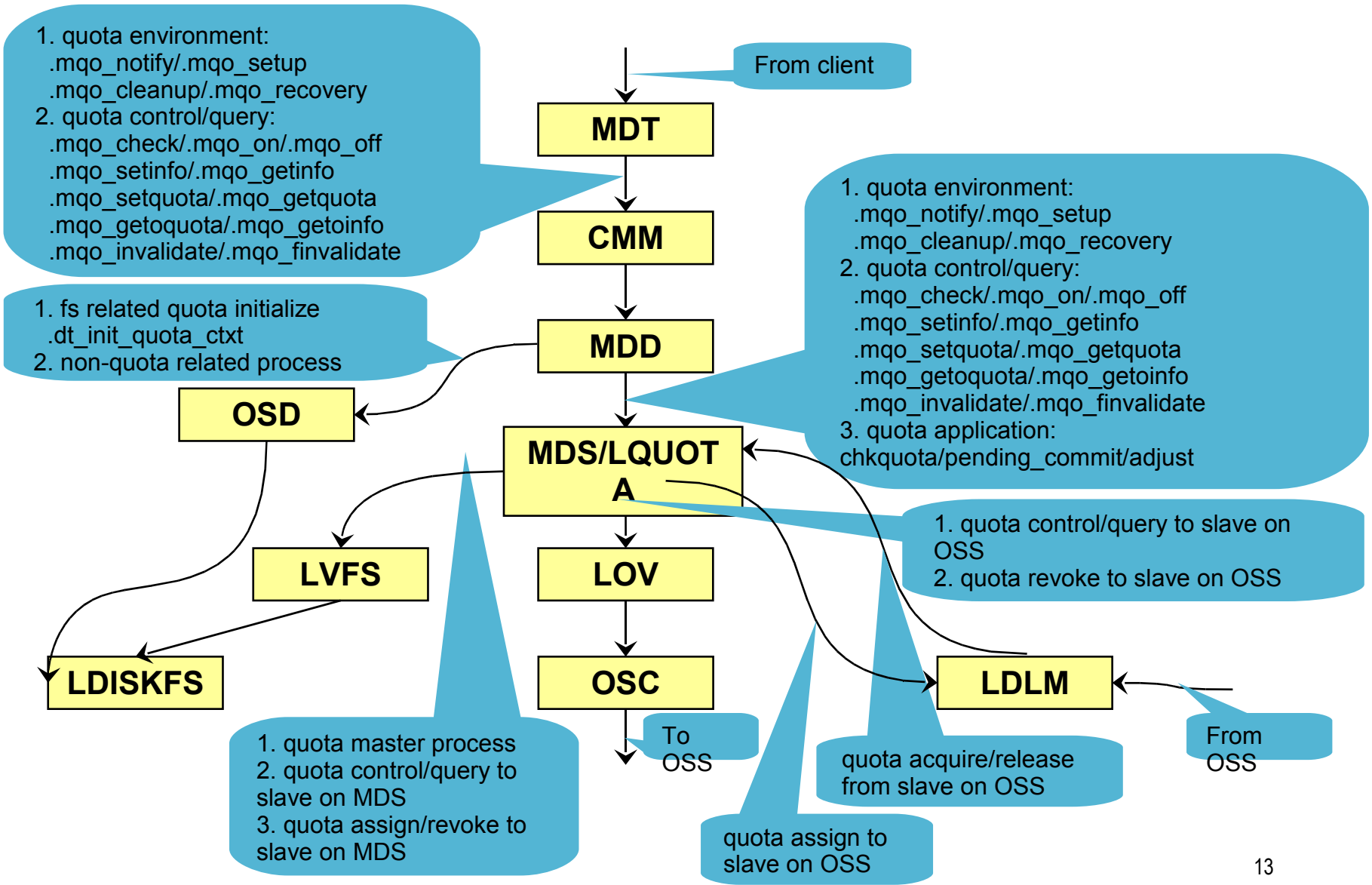
```
struct md_device_operations {
        ...
        struct md_quota_operations {
                int (*mqo_notify)(const struct lu_env *env, struct md_device *m);
                ...
                int (*mqo_check)(const struct lu_env *env, struct md_device *m, struct
                    obd_export *exp, __u32 type);
                int (*mqo_on)(const struct lu_env *env, struct md_device *m, __u32 type);
                ...
                int (*mqo_finvalidate)(const struct lu_env *env, struct md_device *m,
                    __u32 type);
        } mdo_quota;
};
```

MDS side change (5)

- Each MD stack layer registers its own "md_quota_operations" interface

```
static const struct md_device_operations cmm_md_ops = {
        ...
    .mdo_quota      = {
            .mqo_notify     = cmm_quota_notify
                ...
            .mqo_finvalidate = cmm_quota_finvalidate
        }
};
static const struct md_device_operations mdd_md_ops = {
        ...
    .mdo_quota      = {
            .mqo_notify     = mdd_quota_notify,
                ...
            .mqo_finvalidate = mdd_quota_finvalidate
        }
};
```

Lquota master stack



Security process

- Lustre root privilege (CAP_SYS_RESOURCE)

Root user is not restricted by quota. Client's duty to declare to server whether it is root.

- Rogue client (remote client, untrusted) maybe declare as root user to turn off/reset quota.
- Rogue client maybe declare as root user to break through quota limitation.
- MDS can map root user of client as normal user by ID mapping or root_squash. Administrator can configure "CAP_SYS_RESOURCE" for specified client.

  mdt_identity_do_upcall(), /etc/lustre/perm.conf
- OSS has too little knowledge to process as MDS does, a simple way is to ignore root user declaration.

Security process (cont'd)

- Lustre pre-create

For create, MDS checks whether the OSS objects have been pre-created: if yes, only creates MDS object; otherwise, MDS requires OSS to create the OSS objects and pre-creates some other objects for later creation. At that time, the OSS objects have "0" uid/gid with SUID/SGID mode. When client writes such OSS objects for the first time, it sends file owner information to OSS, then OSS set the real file owner which should be the same as MDS object.

> Rogue client maybe send false owner information to OSS for cheating quota verification.

> MDS can pack related owner information into the unmodified OSS capability.

Lquota in future - CMD support

How many quota master?

- Single
  - > Simple but does not match CMD idea well
  - > Potential bottleneck for large scaled quota users
  - > Suggested to be the first version for CMD quota
- Multiple
  - > Need suitable algorithm to distribute quota related requirement to proper quota master
  - > More complex error process, status recovery and concurrency control. (some idea)

Quota on HEAD TOI

yong.fan@sun.com