HLD: Lustre Pool Quotas

Created by Nathan Rutman, last modified just a moment ago, viewed 196 times

Title	Pool Quotas
Summary	User and group per-pool capacity quotas
Author	@Nathan Rutman
Version	1.1
Date	05 Sep 2018
JIRA	LUS-5801 - Quotas for OST Pools IN PROGRESS

Inspectors

Inspector	Inspection Complete Date
@Alexey Lyashkov	
@Cory Spitz	
@Sergey Cheremencev	
@Andrew Perepechko	

Version	Date	Change
1.1	04 Dec 2018	Pool destroy should remove pool quotas

Introduction

This document specifies the High Level Design (HLD) for Lustre pool quotas

The main purposes of this document are:

- Define the requirements for pool-bassed quotas in Lustre
- Outline the strategy to implement pool quotas

The design use case for pool quotas will be for providing user and group usage restrictions on an SSD tier in a mixed SSD/HDD system.

Audience

This HLD provides guidance to developers who will implement the stated functionality. This document can also be used to communicate the high-level design and design considerations to other team members.

1. Definitions

2. Functional Requirements

2.1. Context

With heterogenous clusters consisting of some all-flash OSTs and some all-disk OSTs, it becomes desirable to limit an individual's rights to consume the higher-performance OSTs. The Lustre pools feature allows for the grouping of similar OSTs into performance tiers, and allocating files into tiers. However, it provides no methods to limit the usage of more desirable / expensive / smaller capacity tiers.

Quota controls are the natural solution to administrative limits on space resource. However, quotas in Lustre today are limited to filesystem-wide quota limits on a per-user, per-group, or per-project basis. This design proposes to extent Lustre's quotas capabilities to control allocations within pools.

2.2. User Requirements

ID	Description	Notes
R.poolquotas.perUser	Each user may have a unique pool quota	
R.poolquotas.perGroup	Each group may have a unique pool quota	
R.poolquotas.perProject	Each project may have a unique pool quota	(added rev 1.1)
R.poolquotas.perPoolQuotas	Each pool may have a unique set of user, group, and project quotas	
R.poolquotas.multiPool	Any OST may be part of multiple pools.	An OST may be part of multiple pools, each with different pool quotas

ID	Description	Notes
R.poolquotas.capacity	Pool quotas should limit space / capacity	We state no requirement for per-pool inode quotas (this is a moderately desirable stretch goal if simple)
R.poolqoutas.poolChange	Changes in pool definitions should dynamically affect remaining pool quotas	Both OST addition to and removal from a pool
R.poolquotas.poolDel	Removal of a pool should cause associated quotas limits to be removed	
R.poolquotas.disable	An administrator should be able to disable (and re-enable) quotas accounting for a particular pool	

R.poolquotas.multiPool notes the possibility that an OST may be part of multiple pools, each with different pool quotas. Logically, this is resolved by meeting all applicable quota limits, that is, **the allowable quota is the minimum of all applicable remaining quotas**.

2.3. Interface Requirements

Existing LFS controls for quota controls should be extended to cover pool quotas. LFS quota reporting must also be extended to support pool quotas. As "-p" is already used to specify project quotas, "-P" will be used for Pools (long option "–pool").

2.3.1. Usage

Standard LFS quota controls should be used to set and check pool quotas.

2.3.1.1. Create or change a pool quota for a user

```
# lfs setquota –u bob –P flash ––block–hardlimit 2g /mnt/lustre
```

This sets a 2GB limit for user Bob on the pool named "flash".

For a group:

```
# lfs setquota –g interns –P flash ––block–hardlimit 2g /mnt/lustre
```

For comparison, the command to set a user's total capacity limit:

```
# lfs setquota –u bob ––block–hardlimit 2g /mnt/lustre
```

2.3.1.2. Disable quota enforcement for a pool

```
# lfs quotaoff –P flash /mnt/lustre
```

Turn quotas enforcement off for pool "flash" only. Other pool quotas remain in effect. To re-enable enforcement on this pool:

```
# lfs quotaon –P flash /mnt/lustre
```

2.3.1.3. Report quotas for a user

List quotas of user Bob for pool "flash":

```
$ lfs quota –u bob –P flash /mnt/lustre
```

List quotas of user Bob, including all of Bob's pool quotas:

```
$ lfs quota –h –u bob /mnt/lustre
Disk quotas for user bob(uid 1579):
Filesystem    used    soft  hard   grace   files   soft    hard  grace
/mnt/lustre/  397     0     0      –       3       0       0     –
flash         100k    1M    ...
disk          510M    1G    ...
```

2.4. Operational Requirements

2.4.1. Pool changes

Lustre allows for the dynamic modification of pool definitions; an OST may be added or removed from a pool at any time. Consumed quota for a pool should be calculated based on the current pool definition[R.poolqoutas.poolChange]. If changes in the pool definition suddenly cause a pool quota limit to be exceeded, no new quota grants should be given to OSTs in the pool, but existing grants can continue to be used. Complete deletion of al OSTs in a pool should result in 0 pool quota used. Any quotas files previously associated with that pool should be removed[R.poolquotas.poolDel].

2.4.2. Security

Controls to set quotas should be restricted to the system administrator. (Presumably this is the case today.)

3. Design Highlights

Fundamental concepts: Files and objects do not belong to pools; OSTs belong to pools. Tracking of pool quotas is therefore based on **OST** tracking, not Lustre **file** tracking.

Each individual OST currently requests user (and group) quota *grants* for space to write objects on that OST. The aggregate space consumed by each user (and group) is already tracked by each OST (at the OSD layer), and feeds into the requests for more space from the quota master. The quota master determines subsequent grant sizes for individual

OSTs based on the sum of the current OST consumptions.

Grant sizes from the quota master are currently "filesystem wide"; that is, they consider space consumed on all OSTs. To implement pool-based quotas, the grants can be adjusted to only consider OSTs within a Lustre pool. That is, *any pool quotas decisions/impacts are hidden inside the quota master* as an impact to the amount of space granted to an OST. This results in a much simpler design than than previous proposals:

- No need to store pool information with each object/file.
- OSTs don't need to understand which pool(s) they may be part of.
- All knowledge of pools remains local to the quota master.
- Data written to an OST outside of a pool layout is still accounted for by the pool quota.

OSTs may be members of one or more pools, and we intend to add quota limit controls for any of these pools. If using additional space on an OST would cause a user to exceed *any* applicable quota limit, the user must be denied the ability to consume more space on that OST. Therefore, the quota master should grant the minimum of all applicable quota limits to any OST.

For changes in pool definition, pool quotas accounting must be adjusted for all users of the affected OST. Individual OSTs already track the total amount of space consumed for each quota user. When removing (or adding) an OST to a pool, the quota master need only add or subtract that OST's contribution to the user's pool quota.

Comparison with project quotas

Pool quotas are not like project quotas, which are file-based (files have an attribute describing which project they belong to, just as they have a user and group attribute). We do not need to track which files are part of a pool, which doesn't even make sense - files are not part of pools. Files may have objects that live on OSTs; some of those OSTs may belong to one or more pools. But the object space usage is already tracked by each OST, under the OSD quotas tracking. There is only one set of tracking needed by the OSTs, which is the space consumed by objects on the OST (totaled by user, group, or project). The quota master can add up those consumptions for each OST in various ways (global=everything, pool=subset) to determine grants.

Pool quotas are orthogonal to the user/group/project headings - each pool may have settings for user/group/project. You can have a project quota for the flash pool, just like you have a project quota for the global filesystem (which can be thought of, and indeed coded as, the pool of all OSTs).

Quotas are limits, not rights

Another confusing point is that with pool quotas, there may be multiple quotas in play. It may be initially confusing to be prevented from using "all of" one quota due to a different quota setting. In Lustre, a quota is a limit, not a right to use an amount. You don't always get to use your quota - an OST may be out of space, or some other quota is limiting. For example, if there is an inode quota and a space quota, and you hit your inode limit while you still have plenty of space, you can't use the space. For another example, quotas may easily be over-allocated: everyone gets 10PB of quota, in a 15PB system. That does not give them the right to use 10PB, it means they cannot use *more than* 10PB. They may very well get ENOSPC long before that - but they will not get EDQUOT.

This behavior already exists in Lustre today, but pool quotas increase the number of limits in play: user global space quota, user global inode quota, group global space quota, group global inode quota, project global space/inode quota, and now all of those limits can also be defined for each pool.

In all cases, the net effect is that the actual amount of space you can use is limited to the smallest (min) quota out of everything that is applicable. When reporting the quota limits for a user, group, or project, all of the pool quota limits set for that user/group/project should also be reported, so that a user can more easily see which of the limits has been hit.

4. Functional Specification

Data written by a user to an OST consumes space. If that OST is part of a pool, the consumed space must be accounted for in the pool quota consumed by that user. If the user attempts to write data beyond the quota limit, Lustre should return an EDQUOT error message to the user.

A system administrator must be able to set and check user and group quotas on any pool in the filesystem. Users must not be able to modify their own quotas.

5. Logical Specification

5.1. OST Changes

OST quota tracking remains unaffected. OSTs track user, group, and project quotas in the underlying local FS (ldiskfs or ZFS) as today. The OSTs continue to request new quota grants for a user or group from the quota master when OST objects are created or grow in size beyond the current granted space.

5.2. MDS changes

Handling of pool quotas is confined entirely to the quota master on the MDS. Per the Quota Protocol Design in LU-1300, the quota master maintains a per-slave index file that tracks quota grants for each identifier (uid/gid). In order to find the amount of quota granted for any identifier, the quota master dynamically sums the appropriate contributions from each OST. This is done for the full filesystem (all OSTs), but it can also be done for any subset of OSTs; that is, any pool. In fact, according to the above document, this use case was anticipated in the quota master file structure:

quota file structure

```
.
├── BATCHID
├── CATALOGS
├── changelog_catalog
├── ...
├── quota_master
│   ├── dt-0x0
│   │   ├── 0x1020000
│   │   ├── 0x1020000-OST0000_UUID
│   │   ├── 0x1020000-OST0001_UUID
│   │   ├── 0x20000
│   │   ├── 0x20000-OST0000_UUID
│   │   ├── 0x20000-OST0001_UUID
│   │   ├── 0x2020000
│   │   ├── 0x2020000-OST0000_UUID
│   │   └── 0x2020000-OST0001_UUID
│   └── md-0x0
│       ├── 0x10000
```

```
│       ├── 0x1010000
│       └── 0x2010000
├── quota_slave
│   ├── 0x10000
│   ├── 0x10000-MDT0000
│   ├── 0x1010000
│   ├── 0x1010000-MDT0000
│   ├── 0x2010000
│   └── 0x2010000-MDT0000
├── ...
```

Current code anticipated support for quota pools, but expected a scheme where the pool ID comes from the slaves. We therefore can't use the existing structure without breaking our requirement that OST-side quota tracking remains unaffected.

The first time a quota limit is set on a pool (via *lfs setquota*), the MDS will create a new file with the pool name under the "quota_master/pools" directory. The name of the file will be composed of a pool name and quota ID type ("usr", "grp" or "prj"). For example the name of the file that stores LQEs with type "user" for pool "flash" will be "flash_usr", full path - "quota_master/pools/flash_usr". The format of records in these files will be the same as that currently used for the global index:

```
/*
 * Global quota index support
 * Format of a global record, providing global quota settings for a given quota
 * identifier
 */
struct lquota_glb_rec { /* 32 bytes */
        __u64 qbr_hardlimit; /* quota hard limit, in #inodes or kbytes */
        __u64 qbr_softlimit; /* quota soft limit, in #inodes or kbytes */
        __u64 qbr_time;      /* grace time, in seconds */
        __u64 qbr_granted;   /* how much is granted to slaves, in #inodes or
                              * kbytes */
};
```

There will be no per-pool grant requests from the quota slaves; there will be a single general request for grant. When a quota slave requests additional quota grant, the MDT must check the quota limits for *each pool the OST is a member of*. For pool files that have a matching identifier entry, the quota master calculates the minimum amount of quota grant remaining for that identifier. This provides the upper bound to the quota granted to the slave.

```
E.g. for this case:
lfs setquota --block-hardlimit 2097152 -u user1 -P flash /mnt/lustre
lfs setquota --block-hardlimit 2097152 -u user2 -P flash /mnt/lustre
lfs setquota --block-hardlimit 2097152 -u user1 /mnt/lustre
```

there would be just one quotas file created (if it did not already exist): quota_master/pools/flash_usr. The global index file that scopes all slaves (quota_master/dt-0x0/0x20000) already exists since quota is On.
For a quota acq request for a user from an OST in the flash pool, the MDS would check both (0x20000 and flash_usr) files and return the minimal amount remaining. For an OST not in the flash pool, it would not check the quota_master/pools/flash_usr.

5.2.1. Pool changes

When a pool definition is changed (OST add or remove), the calculation of granted quota for that pool must be invalidated. At the next request for quota grant, the total already granted for the current pool definition will be recalculated. Note this may result in some identifiers suddenly being over quota; future grants to slaves will be 0, and subsequent write requests should get EDQUOT.

When a pool is deleted (*lctl pool destroy*), the associated quota files and subdirectories should be deleted. If the pool is re-created then new quotas limits must be re-established if desired.

5.2.2. Qunit changes

Lustre quotas are allocated in fixed sizes called *qunit* to avoid many small requests. Qunit is a part of lqe and is applied on all slaves from quota ID scope. It is calculated by master every time the global grants are modified. In the common case when quota soft limit is not set qunit is computed as follows*: limit / (2 * slv_cnt).* Then 75% of the quota space can be granted with current qunit value. The remaining 25% are then used with reduced qunit size (by a factor of 4) which is then divided in a similar manner.

The requirement that any OST may be part of multiple pools may cause several performance problems.
Let's consider the most obvious case when pools are overlapped:

Pool1: ost1, ost2, ost3, ost4; Limit 1G
Pool2: ost4, ost5, ost6; Limit 2G

User Joe has quota limits on pool1 and pool2.
The first question is how qunit should be calculated in this case. Choosing the formula: min(pool_limit/(2 * pool_slv_cnt)).
For our example qunit is 128 MB for pool1 and 341 MB for pool2, so min is 128 MB.
After granting over 750 MB in a pool1 qunit will be recalculated as 64 MB (poo1_limt/(4* pool1_slv_cnt)).
Later qunit will become smaller and smaller until it reaches *qpi_least_qunit* (1MB by default).
Since the qunit is common for both pools it affects pool2 write performance even if pool2 has a file on OSTs 5 and 6.
It looks unfair because pool2 has 1.25 GB remaining space and ideally should use much higher qunit value.
Furthermore the qunit recalculation causes sending glimpse callbacks on per-ID quota locks.

The same problem could be faced with PFL.

Solutions:

1. Calculate qunit per pool. In case of overlapping pools use minimum qunit.
   // TODO: write example with details.
   // btw, it is not clear how calculate qunit in case of complicated overlapping
2. Don't use overlapping quota pools. Possibly we can forbid such pools at creation time. However it brakes R.poolquotas.multiPool requirement.

3. Set the minimum qunit size(qpi_least_qunit) to some larger value - 4 or 16 MB. It doesn't solve the problem fully but could help.

5.2.3. Quota pools and LOD pools

We distinguish here between "LOD pools", grouping OSTs for file layout purposes, and "quota pools", which groups OST for quota accounting. Unfortunately, quota code does not have direct access to the LOD pool definitions due to layering restrictions between the LOV and the QMT.

The simplest way to mirror these "LOD" pools is to add the same configuration handlers to QMT and call them together with default "LOD" pools handlers. For example LCFG_POOL_NEW command now means to call obd_pool_new for 2 obd devices - lustre-MDT0000-mdtlov and lustre-QMT0000. This will create pools on both quota and lod layers in memory. OSTs will be added (and removed) in these pools by command LCFG_POOL_ADD.
Note, handling LCFG_POOL_xxx commands in quota will not cause any on-disk changes by itself. The creation of the "pool_name_quota_type" file in quota_master/pools will instead be caused by the first call of

```
"lfs setquota ... -P "pool_name"".
```

Another possible approach is to access pools in LOD layer directly from QMT. The benefit is that we don't need to add code for quota pools that has the same functionality with the code from lod_pool.c.
However, access between different layers (QMT and LOD) needs additional locking. Also the LOD layer has to notify QMT about all pool changes. This requires changes not only in QMT but in the LOD as well. We decided on the former approach in order to restrict the impact of the quota pools feature to the quota code only.

5.2.4. Quota pool list

All quota pool entries will be linked into a global quota_pool_list. Thus when master has a request from OST N it has to go through quota_pool_list and find all quota pool's that include OST N.
Thus we will need just N compares and one pass through the list. So no reasons to store quota pools entries in something more complex.

5.3. Interoperability

Because there are no OST code changes and quotas grant is effective a quota master policy, the MDS code can be updated independently. Similarly, downgrading should be just as simple: quota pools file's in directory quota_master/pools will be ignored.

6. Use Cases

6.1. Typical use

For design purposes, we should concentrate on one general use case for pool quotas. In this use case, an administrator would set up a global filesystem quota for each user, say 10TB in a 1PB system. The system also has 15TB of flash, and the administrator wants to prevent any one user from using more than 1TB of flash. He creates a pool of the flash OSTs, and assigns a pool quota of 1TB each to the users. The quota consumed by the user depends on the location where he writes files; writes to the flash pool consume quota against both the global and the flash limits.

6.2. Quotas changes

6.2.1. New quotas on a pool

A pool named "flash" is set up with OST10 through OST20.

This sets a 2GB limit for user Bob on the pool named "flash":

```
# lfs setquota -u bob -P flash --block-hardlimit 2g /mnt/lustre
```

If Bob's capacity used on the OSTs in pool "flash" exceeds 2GB, further writes will fail with EDQUOT. Bob can continue to write to any other OSTs NOT in "flash" without restriction.

6.2.2. Multiple quotas

A second pool is set up called "site1", comprised of OST5 through OST15. Bob is given a quota on site1 of 1GB. Bob currently has:

OSTs	Used space (GB)	Pools
OST0-4	6.0	-
OST5-9	0.5	site1
OST10-15	0.4	site1, flash
OST16-20	1.0	flash

The associated quota limits are then:

Pool	Limit (GB)	Used	Remaining grantable quota
-	no limit	7.0	no limit
site1 (OST5-15)	1.0	0.9	0.1
flash (OST10-20)	2.0	1.4	0.6

The amount of additional grant space that can be given to Bob on OST11 is therefore 0.1GB (min of site1 and flash). If Bob uses this 0.1GB, he will not be granted any more space on OST5-15, but he may be granted up to 0.5GB more on OST16-20.

6.2.3. Remove pool quotas

Turn quotas enforcement off for pool "flash" only. Other pool quotas remain in effect.

```
# lfs quotaoff -P flash /mnt/lustre
```

To re-enable enforcement on this pool:

```
# lfs quotaon -P flash /mnt/lustre
```

6.2.4. Change pool quotas

Change to a 1GB limit for user Bob on the pool named "flash":

```
# lfs setquota -u bob -P flash --block-hardlimit 1g /mnt/lustre
```

Bob is now over limit in pool "flash"

Pool	Limit (GB)	Used	Remaining grantable quota
-	no limit	7	no limit
site1 (OST5-15)	1	0.9	0.1
flash (OST10-20)	1	1.4	-0.4

Bob can write nothing to OST10-20, 0.1GB to OST5-9, and unlimited to OST0-4.

6.3. Pool changes

6.3.1. New pool

New pools should not generate any pool quota activity.

6.3.2. Destroy pool

Any existing pool quota definitions for a deleted pool are removed. Future re-definition of the pool would have no quotas set.

6.3.3. Add OST to pool

OST16 with 0.2GB of Bob's data on it is added to pool site1. Bob is now over quota for site1. He can write nothing in either pool, but can still use non-pool OSTs.

Pool	Limit (GB)	Used	Remaining grantable quota
-	no limit	7	no limit
site1 (OST5-16)	1	1.1	-0.1
flash (OST10-20)	1	1.4	-0.4

6.3.4. Remove OST from pool

7. Performance Analysis

If no pool quotas are set, the quota master has no additional work to do when administering quota grants. If a pool quota is set for a user, then the quota master may have to check more than one quota file to determine the correct minimum grant; this may have some slight performance impact on a non-cached LQE's. As the number of pools with quotas increases, this load will also increase.

Quota slaves are not impacted by pool quotas; slave code is not touched in this design. However, since additional quota limits may be imposed by pool quotas, the size of the quota unit granted for a user (quota id) will be affected by the "closest" limit applicable to that id; see 5.2.2.

8. Deployment

8.1. Compatibility

While new quotas files will be created when pool quotas are configured, no existing on-disk formats are changed. Previous version of Lustre will simply ignore the new quotas files. We anticipate no backwards compatibility issues.

9. References

- lfs man page
- LU-11023
- Lustre / Quota overview

hld

20 Comments

Patrick Farrell

General comment:
No mention of project quotas, but they would work just like user and group, right? So, user, group, project...? (I'm not completely certain I understand project quotas, so please correct me if needed.)

Nathan Rutman

need to add lfs quota -p, like lfs df -p, and display all pool quotas without -p.

John Fragalla

lfs quota -p is already used today for "project quota", from what I see from the command.

Sergey Cheremencev

```
└─/
  ├─ROOT
  ├─last_rcvd
  ├─...
  └─quota
      └─0 /* pool ID, 0 is the default and only one supported for now */
        ├─0 /* USRQUOTA */
        | ├─admin_quotafile.usr /* global index for user quota */
        | ├─data
        | |   ├─0000 /* slave index for OST0000 */
        | |   ├─0001 /* slave index for OST0001 */
        |
        └─1 /* GRPQUOTA */
          ├─admin_quotafile.grp /* global index for group quota */
          ├─data
          |   ├─0000 /* slave index for OST0000 */
          |   ├─0001 /* slave index for OST0001 */
```

In fact above structure is not used now.
MDS stores 2 directories for that - quota_master and quota_slave.
The first one is for whole cluster quota accounting. The second is to track quota per target(need to dive here deeper).

```
[root@dhcppc4 mds]# find quota_master/
quota_master/
quota_master/md-0x0
quota_master/md-0x0/0x2010000
quota_master/md-0x0/0x10000
quota_master/md-0x0/0x1010000
quota_master/dt-0x0
quota_master/dt-0x0/0x2020000-OST0000_UUID
quota_master/dt-0x0/0x20000-OST0000_UUID
quota_master/dt-0x0/0x1020000-OST0000_UUID
quota_master/dt-0x0/0x1020000-OST0001_UUID
quota_master/dt-0x0/0x2020000
quota_master/dt-0x0/0x20000
quota_master/dt-0x0/0x2020000-OST0001_UUID
quota_master/dt-0x0/0x1020000
quota_master/dt-0x0/0x20000-OST0001_UUID
[root@dhcppc4 mds]# find quota_slave
quota_slave
quota_slave/0x2010000
quota_slave/0x10000-MDT0000
quota_slave/0x1010000-MDT0000
quota_slave/0x2010000-MDT0000
quota_slave/0x10000
quota_slave/0x1010000
```

Let's look into quota_master. md-0x0 and dt-0x0 are quota metadata and data directories.
The pool ID number is stored in lower 2 bytes of the fid if it has sequence 0x200000006(FID_SEQ_QUOTA_GLB).

```
[root@dhcppc4 quota_master]# lfs path2fid */* | grep 0x200000006
dt-0x0/0x1020000: [0x200000006:0x1020000:0x0]
dt-0x0/0x20000: [0x200000006:0x20000:0x0]
dt-0x0/0x2020000: [0x200000006:0x2020000:0x0]
md-0x0/0x10000: [0x200000006:0x10000:0x0]
md-0x0/0x1010000: [0x200000006:0x1010000:0x0]
md-0x0/0x2010000: [0x200000006:0x2010000:0x0]
```

As can be seen object ID of FID is equivalent to the name of the file.
FID id generation formula - fid->f_oid = (lqtype << 24) | (pool_type << 16) | (__u16)pool_id; (see lquota_generate_fid).

Nathan Rutman

Still, if there is a pool ID, then we can add new quota files for new pools, yes? I don't really care about the directory layout; I just want to make sure the basic idea still works.

Sergey Cheremencev

Yes, the basic idea should work. Just collect the details gathered through investigation.
However want to point there are a lot of comments in a code that several quota pools are not supported.
Want to check this.

Nathan Rutman

I just updated this HLD on the behavior of pool destroy. I think we should retain the pool definitions files instead of deleting them, so that if the pool is re-created in the future the old definitions are again used.

Nathan Rutman

@Sergey Cheremencev can you please update section 5.2 of this HLD to reflect the actual directory structure?

Also, does the "0x0" mean anything here: `quota_master/md-0x0`

Does it make sense to use this field as the pool id, i.e. `quota_master/md-0x02` for pool 02?

Patrick Farrell

Do pools have ids? Or do you mean the pool name?

If pools have ids, they are basically never used - everything is all strings in the code I've looked at. (It actually seems a little weird that there isn't a string-id translation rather than just strings everywhere... but that's how it seems to work.)

Ah, yeah - there's some nascent pool_id support, but it's always zero.

Nathan Rutman

Hadn't really thought about that. But I don't immediately see a problem with making a directory like `quota_master/md-poolname`

Sergey Cheremencev

Yes, I will update 5.2

0x0 is a pool number.

Sergey Cheremencev

@Nathan Rutman, @Patrick Farrell, please see my comment LUS-6583 - Finish HLD for OST Pool Quotas TO DO in LUS-6583.

Pools have ids. By default only 0 is used. But as you can see from experiment pool numbers > 0 are also supported.

Nathan Rutman

Where is the mapping of pool name to pool ID? Is this persistent? If there's a persistent 1:1 poolname:poolID then fine, otherwise it seems like we can just use poolname in the quotas directories.

Sergey Cheremencev

Right now there is NO any relationships between OST pools and Quota pools.
OST pools have string names, Quota pools - numeric IDs.

The simplest way is to add a field to OST pool structure to store appropriate Quota Pool id.
Quota pool in turn should store appropriate OST pool name or a link to a record in config.
I will look into the code for details.

Nathan Rutman

What about changing quota pools to use string names also? I.e. /quota_master/md-poolname instead of /quota_master/md-0x1

Sergey Cheremencev

It requires a little bit more work.

Now quota pool id is used as a part of qpi_key - it is a key for hash table. gpi_key consists of pool_id and pool_type: pool_id + (pool_type << 16).
Yes, we can use ost_pool_name string like a hash key.
However qpi_key is also used to generate the FID associated with the global index storing quota settings for appropriate pool. The FID is also used as a name of such file.
So we need to change this part as well.

Sergey Cheremencev

Came back to the question to make quota pools independent from LOD pools.
I tried to make a simple code and realised that it is not pretty simple.
For example a command "lfs setquota -u bob -p flash --block-hardlimit 2g /mnt/lustre" is handled on MDS in osd layer (lustre-MDT0000-osd in my case). While all ost pools related data is stored in lustre-MDT0000-mdtlov. There is no simple way to interact between these 2 devices.

At the same time I reread https://jira.whamcloud.com/browse/LU-11023 again and paid attention to Andreas comment:

> I'm not fixed on linking this quota to OST pools since there are some complexities there, and we'd also want to have MDT pools for that to be useful for DoM, but I think some kind of limits are needed for these use cases.

Possibly we can create new quota pools directly from the quota device using smth like

```
lfs quota_pool_new -type OST flash1 /mnt/lustre
lfs quota_pool_add flash1 OST0001,OST0002
lfs quota_pool_new -type DOM dom1 /mnt/lustre
lfs quota_pool_add dom1 MDT0001,MDT0002 /mnt/lustre
```

On the other hand we still need to have access to LOD layer to avoid adding nonexistent OST to quota pool.
Possibly we can get this info from configs.

Any thoughts ?

Nathan Rutman

So are you saying the quota device can't have access to the mdtlov, and therefore doesn't/can't know about OST pool definitions?

Is that the only problem? Say we get this OST pool info to the quota code somehow, is that sufficient info to proceed with pool quotas? (I.e. quota code can add up individual OST quota usage and do the correct granting?)

Seems like it should be possible to send this info somehow between layers, worst case doing it via configs.

Sadly I am very unfamiliar with quota code - @Sergey Cheremencev I think you could ask for advice in the LU-11023 ticket (as well as our Lustre team).

Sergey Cheremencev

> So are you saying the quota device can't have access to the mdtlov, and therefore doesn't/can't know about OST pool definitions?

I looked at the code again and found at least one way:

```
obd = class_name2obd("lustre-MDT0000-mdtlov");
lod = lu2lod_dev(obd->obd_lu_dev);
```

> Sadly I am very unfamiliar with quota code - @Sergey Cheremencev I think you could ask for advice in the LU-11023 ticket (as well as our Lustre team).

Sure. I've asked in the ticket and Andreas answered that it is better to base quota pools on already existed LOV pools.

Alexey Lyashkov

You should don't use a code similar this.

I still don't understand why you try to do it is in the lustre server side, while user land implementation much simple.

We can ask a MDT to provide a pools list with additional info like 'pool ID', can quota code can be asked about quota info by pool ID.