OST pool based quota

Li Xi
DataDirect Networks
Background: What is OST pool?

• OST number of Lustre clusters is growing rapidly
• OST pool feature enables users to group OSTs together for more flexible and controllable striping
• OST pools follow these rules:
  – An OST can be a member of multiple pools
  – No ordering of OSTs in a pool is defined or implied
  – Stripe allocation within a pool follows the same rules as the normal stripe allocator
  – OST membership in a pool is flexible and can change over time
• OST pool based quota is not supported today
  – But luckily current quota framework is powerful and flexible which makes it easy to add new extension.
Why support quota on OST pools?

- Fine-grained quota control is important
  - user/group quota doesn't work in some use cases. (e.g. project based storage volume allocation)
  - Quota for small groups in a filesystem helps administrator to make a capacity plan of entire storage's volume
  - Pool separate the danger of disk space exhausting in the entire system
  - XFS supports per-directory or per-project quota and GPFS also supports fileset based quota which is conceptually similar
  - Patch which introduces subtree quota support for ext4 has existed for years

- Many use cases for directory-based or pool-based quotas
  - Directory-based quotas need support from lower level
  - Pool-based quotas are a much more straightforward to implement
  - Pool-based quotas can be used to set quota on a given directory

- Enhancement of user/group quota
  - Administrator can set quota limit for user/group to specific OST pools which means:
    - Alert before any partition becomes full
    - Most basic but useful storage management mechanism
Architecture of Quota

• Quota “master”
  – A centralized server hold the cluster wide limits
  – Guarantees that global quota limits are not exceeded and tracks quota usage on slaves
  – Stores the quota limits for each uid/gid
  – Accounts for how much quota space has been granted to slaves
  – Single quota master running on MDT0 currently

• Quota “slaves”
  – All the OSTs and MDT(s) are quota slaves
  – Manage local quota usage/hardlimit acquire/release quota space from the master
Architecture of Quota

- **Clients**
- **LOV**
- **MDS(QMT)**
- **OSS(QSD)**

- Bulk write request/reply
- DQACQ request/reply
- Local quota check
OST pool based quota:
Requirements

• Integrated in current quota framework
  – Ability to enforce both block and inode quotas
  – Support hard and soft limits
  – Support user/group (and maybe pool) accounting

• Full support of pool
  – Dynamic change of pool definition
  – Separate quotas of users/groups for each pool

• No significant performance impact
Design and implementation #1
Pool definition in LLOG

1. Modify pool

2. Find/create pool ID through name

3. Write and distribute LLOG

4. MDT(OSD)/MDT0(QMT)/LOD: update pool

4. OFD(OSD): update pool

4. LOV: update pool
Design and implementation #2
Quota changes for pool support

• The quota master keeps an hash table
  – One instance for each pool to hold the cluster wide limit
• All OSDs keep hash tables of QSD instances
  – One QSD instance for each pool
  – Corresponding QSD of a given pool is used when quota is acquired/released
• Objects on OSTs store their pool IDs as extended attributes
  – Pool ID is needed for QSD matching
  – Initialized before objects consume disk spaces
• Support of both LDISKFS and ZFS
  – Pool IDs of objects is cached for better performance
Design and implementation #3
Flow of a write request

Client
- Send RPC to write to an object of a file

OSS
- Lookup quota entry through pool ID from object xattr
- Check whether local quota is enough
  - Yes: Write data to object
  - No: Send RPC to acquire more quota from master
- Send reply to client

MDS
- Lookup quota entry through pool ID from RPC
- Grant space quota to the slave
- Write RPC completed
Status

• Main framework has been completed
• LU-4017 quota: Add pool support to quota
  – Main codes for pool support of quota
  – The patch is a big one which involves quite a lot of components
  – According to early test, the patch works well
  – Will be split into multiple parts for review
• User space command update
  – Use ‘-p pool_name’ argument to specify which pool to configure
• Test suits for pool based quota
  – Verify the correctness and efficiency of pool based quota
• LDISKFS support is ready, but ZFS support is not yet finished
UseCase #1

Quota of users/groups for directories

OST pool 1

OST pool 2

Real OSTs
UseCase #2

Quota for different kinds of OSTs

Pool for scratch files
Pool for premium users
Pool for normal users

Cheap OSTs
Faster OSTs
Normal OSTs
UseCase #3
Directory/Project based quota

- Directory/Project based quota will enable new Lustre use cases (e.g. collaboration, Cloud space, etc.)
  - Need space accounting of pool in total
How to use pool based quota

◆ Create and manage OST pools

# Normal utilities of pool management
# lctl pool_new fsname.pool1
# pool_add server1.pool_1 OST0000

◆ Set quotas of OST pools

# lfs setquota ... [-p <pool-name>] <filesystem>
# lfs setquota --block-hardlimit 2097152 -u user1 -p pool_1 /mnt/lustre
# lfs setquota --block-hardlimit 1048576 -u user1 /mnt/lustre

◆ Display quotas and disk usage of OST pools

# lfs quota ... [-p <pool-name>] <filesystem>
# lfs quota -u user1 -p pool_1 /mnt/lustre/
# lfs quota -u user1 /mnt/lustre/

◆ Associate directories/files with OST pools

# lfs setstripe <filename|dirname> --pool|-p pool-name
# lfs setstripe -p pool_1 /mnt/lustre/dir1

◆ Then the limits are enforced
Further work

• Compatibility with older versions
  – LLOG record format has changed
  – Disk format of quota files has changed
  – Quota control API has changed
  – Wire format has changed

• Space accounting of pools along with users/groups
  – Total quotas of a given pool
  – Enable directory/project based quota

• Clustered meta-data support
  – MDT pool support of quota

• Any advice will be welcome!
Thank you!