High Level Design for bug 10707

Tian Zhiyong

2006.8.4–2006.8.6

1 Introduction

Because qd_count of struct `qunit_data` is 32 bit, when using > 4G limits for quotas the OSS nodes may die (never ending loop)

The faulty code is in `check_cur_qunit`():

```
if (limit <= usage + tune_sz) {
   while (qdata->qd_count + limit <= usage + tune_sz)
   qdata->qd_count += qunit_sz;
   ret = 1;
} else if (limit > usage + qunit_sz + tune_sz) {
   while (limit - qdata->qd_count > usage + qunit_sz + tune_sz)
   qdata->qd_count += qunit_sz;
   ret = 2;
}
```

2 Requirements

1. obey protocol compatibility policy.
2. after some time, qd_count of struct `qunit_data` can be translate into 64bit without making any trouble.
3. the size of structure qunit_data doesn't change, which can reduce some work.
4. flag in structure `qunit_data` uses binary operation.
5. try to reduce the time of RPCs in order to improve the performance.
6. easy to test.

3 Functional specification

First of all, I will make some definitions.

- **stale master** is a node holding the cluster wide limits for a uid or gid without the code of fixing bug10707.
- **stale slave** is a node which only considers hard quota and only has operational quota files without the code of fixing bug10707.
- **new master** is a node holding the cluster wide limits for a uid or gid with the code of fixing bug10707.
- **new slave** is a node which only considers hard quota and only has operational quota files with the code of fixing bug10707.

what I will do in order to fix this bug is:

- change structure `qunit_data:` makeing qd_count is 64bit; merging qd_type and qd_isblk into qd_flags which does binary operation;
- adding a structure naming `qunit_data_old`, which is just old sturcture `qunit_data`. That means that stale master/slave has:

```
struct qunit_data {
   __u32 qd_id; /* ID appiles to (uid, gid) */
   __u32 qd_type; /* Quota type (USRQUOTA, GRPQUOTA) */
   __u32 qd_count; /* acquire/release count (bytes for block quota) */
   __u32 qd_isblk; /* Block quota or file quota */
};
```

new master/slave has:

```
struct qunit_data {
   __u32 qd_id; /* ID appiles to (uid, gid) */
   __u32 qd_flags; /* Quota type (USRQUOTA, GRPQUOTA)occupy one bit; Block
   __u64 qd_count; /* acquire/release count (bytes for block quota) */
};
struct qunit_data_old {
   __u32 qd_id; /* ID appiles to (uid, gid) */
   __u32 qd_type; /* Quota type (USRQUOTA, GRPQUOTA) */
   __u32 qd_count; /* acquire/release count (bytes for block quota) */
   __u32 qd_isblk; /* Block quota or file quota */
};
```

- a new slave/master handles the quota using struct `qunit_data` internally; only when a new slave/master will send/receive the quota request/reply, it will translate struct `qunit_data` into struct `qunit_data_old` or adverse if necessary.
- a new master discerns whether the master is new or stale through **OBD_CONNECT_QUOTA64**; a new slave is same. More details will be within the DLD.
- handle qd_flags using binary operations.
- add proc entries so that we can easily change the new master/slave's connect_flags. In this way, we can make the new master/slave just acting like the old master/slave. Doing this is just for the test.

4 Use cases

Test environments:

- a stale master and a new slave
- a new master and a stale slave
- a new master and a new slave
- a new master , a new slave and a stale slave

Writing multiple big files to lustre(total size > 4G) and then deleting them in order that an ost will release >4G quota. When deleting the files, a corresponding ost will release >4G quota. it should send successfully and will never endless loop. This test will run successfully under the four environments above.

Certainly the script of tests/sanity-quota.sh will run under the four environments above. At last, when landing code this test will be added to tests/sanity-quota.sh.

5 Logic specification

5.1 Procedure of acquiring/releasing quota

- when a new slave sends a quota request, split the `qunit_data` struct into multiple `qunit_data_old` struct and send them if the master handling the quota request is a stale master; send the request directly if the master handling the quota request is a new master.
- when a new master receives a quota request, it will translate struct `qunit_data_old` into struct `qunit_data` and then handle it if the slave sending this request is a stale slave; it will hanle it directly if the slave sending this request is a new slave.

- when a new master replies a quota request, it will translate struct `qunit_data` into struct `qunit_data_old` and send it if the slave sending this request is a stale slave; it will send the reply directly if the slave sending this request is a new slave.
- when a new slave receives a quota reply, it will translate struct `qunit_data` into struct `qunit_data_old` and handle it if the master handling the quota request is a stale master; it will handle the request directly if the master handling the quota request is a new master.

The key idea is that a new slave/master handles the quota using struct `qunit_data` internally; only when a new slave/master will send/receive the quota request/reply, it will translate struct `qunit_data` into struct `qunit_data_old` or adverse if necessary.

5.2 Functions manipulate the qunit_data structure

- When a slave sends a quota request to a master, it will call function schedule_dqacq to send it. This function will translate the format of qunit_data to qunit_data_old if necessary. There are three functions: dqacq_completion, qctxt_adjust_qunit, qslave_recovery_main. They will call schedule_dqacq and are charge of spliting the quota request if necessary.
- Then the master receives the request and sends it to function target_handle_dqacq_callback to handle it. After that, it will send a reply back to the slave. This function maybe change qd_count.
- The slave receives the reply and sends it to function dqacq_interpret to handle it.

For example, when a client writes a big file to lustre, the chain of functions call is:

1. slave(send a quota request to the master): ptlrpc_main->ptlrpc_server_handle_request->ost_handle->ost_brw_write->filter_commitrw->filter_commitrw_write->filter_quota_adjust->qctxt_adjust_qunit->schedule_dqacq;
2. master(receive a quota request and send the reply back): ptlrpc_main->ptlrpc_server_handle_request->ldlm_callback_handler->target_handle_dqcaq_callback;
3. slave(receive the reply): ptlrpcd->ptlrpcd_check->ptlrpc_check_set->dqacq_interpret

5.3 RPC which will be affected

It only changes the format of qunit_data based on capabilities of other peer before the RPC. The length of qunit_data and the order of RPC isn't changed. Details can be seen in "Procedure of acquiring/releasing quota".

5.4 Swabbing problem

There are two swabbing functions to deal with swabbing. lustre_swab_qdata deals with qunit_data; lustre_swab_qdata_old deals with qunit_data_old.

6 State management

A new master has two states, when the corresponding slave is a stale slave, it will do translation; when the corresponding slave is a new slave, it won't do translation.

The same to a new slave.

A new master/slave switches the states by the corresponding import or export(reference to exp->exp_connect_flags or imp->imp_connect_data->ocd_connect_flags)

6.1 Scalability & performance

This does nothing to scalability.

Only when a new master and a new slave handle quota request, it is efficient. Except that, when there are >4G quota to release/acquire, it is spilit into small quota requests(<4G). That is less efficient in that situation.

6.2 Wire format changes

Stale master/slave has:

```
struct qunit_data {
   __u32 qd_id; /* ID appiles to (uid, gid) */
   __u32 qd_type; /* Quota type (USRQUOTA, GRPQUOTA) */
   __u32 qd_count; /* acquire/release count (bytes for block quota) */
   __u32 qd_isblk; /* Block quota or file quota */
};
```

new master/slave has:

```
struct qunit_data {
   __u32 qd_id; /* ID appiles to (uid, gid) */
   __u32 qd_flags; /* Quota type (USRQUOTA, GRPQUOTA)occupy one bit; Block
   __u64 qd_count; /* acquire/release count (bytes for block quota) */
};
struct qunit_data_old {
   __u32 qd_id; /* ID appiles to (uid, gid) */
```

```
    __u32 qd_type; /* Quota type (USRQUOTA, GRPQUOTA) */
    __u32 qd_count; /* acquire/release count (bytes for block quota) */
    __u32 qd_isblk; /* Block quota or file quota */
};
```

structrue `qunit_data` is changed, but its size isn't changed. The code will deal with the difference.