HLD of Network resource handling

Author Wangdi

2006/3/01

1 Introduction

According to the new MDS/OST layering architecture, exports and imports and network recovery strictly belong in the OSC/OST, MDC/MDT layers and the MDD and OSD should not contain per client data structures. So some network level resource handling should be moved from OSD/MDD to MDT/OST.

2 Requirements

Ensure network level resource handling is in the MDT/OST with support from MDD and OSD layer. This project includes:

1)Move grant handling to OST with the support of OSD, which is originally in obdfilter.

2)Move Network llog handling to MDT/OST with the support of MDD/OSD.

3 Functional specification

3.1 Grant handling

Clients need to cache write data to achieve maximum performance. So the OST need to grant each client some place to enable clients cache the write data and avoid NOSPC error when flushing these caching data to OST.

Current grant implementation include 2 parts, client and OST. Client will manage the grant cache and write back the cache data according to grant cache status, which is totally belong to OSC, so we do not need modify it according to the new layer. While grant cache in server is currently implemented in OSD(obdfilter), so it should be moved to OST, because server must maintain each client grant space, which is the function of OST.

3.2 Network llog handling

Network llog is used to help implement the cluster transaction, for example unlink and setattr. Currently, these network llog interface is implemented in MDS/obdfilter, we should move them to MDT/OST layer.

4 Use cases

4.1 Read

- OST got read req and check grant info.
- If the req include grant info, OST call grant API to update the OST grant info.

4.2 Write

- OST got write req and check grant info
- If the req include grant info, OST call grant API to do write grant check and update the OST grant info.
- OST will also increase or decrease the client grant space according to the grant info of the req.

4.3 OST setup

- Client send connect req to OST
- OST check whether the req include connect_grant flag, if it included, the OST will grant some cache to the client
- Client get the grant cache after connect, then use these cache in the following I/O.

4.4 Client evicted

- Client is evicted by the OST server
- Client cleanup grant cache space and reconnect to OST server to recovery.
- OST got connect request and grant cache space to client again.

4.5 unlink

- client send unlink req to MDS, MDS remove the metadata object and add unlink log with MDD llog support.
- MDS send unlink req to OST, OST will destroy the objects. then send cancel llog req to MDS by llog network API.
- MDS cancel the unlink llog with MDD llog support.

5 Logic specification

5.1 Grant implementation

Current grant mechanism is implemented in OSD(obdfilter). When moving them from OSD to OST, we just need to move those API to OST layer, but may need add some OSD support API to retrieve some filesystem attr to calculate the grant info. Furthermore, these three items fo_tot_dirty, fo_tot_granted and fo_tot_pending, which indicate the client cache status, should also be moved to OST layer. Since these API has been implemented in obdfilter, so in this HLD, we only describe the modification for these API. For those grant API:

1. ost_grant()

   It calculates amount of grant that we can give to this client, which is calculated based on amount of free space of the filesystem. So when this method is moved to OST, we need the OSD support API to get the free space of the filesystem, which can be done by osd_statfs.

2. ost_check_space()

   It checks the incoming write request pages if they are using the space from grant or not. Here we also need osd_statfs to check the free space of the filesystem, in case the page writing request is not taking the space from grant.

3. ost_grant_space_left()

   It calculates how much space we have left on the OST filesystem, so besides moving, we also need osd_statfs here too.

4. ost_grant_incoming()

   It is called on incoming write obdo. It checks for clients not using more cache than they were allowed to (warns if not). This API is totally independent on OSD, so we just moved the to OST.

Except these API, some related structure should also be moved from OSD to OST. The filter_export_data, which is originally in the export data of obdfilter, should be moved to OST layer.

5.2 network llog implementation

Current llog implementation include 2 parts, one is totally belong to filesystem, which will be moved to MDD/OSD, for example record unlink/setattr llog in MDS, which will be discussed in other HLD, another one is belong to network, we should care. In current implementation, we have llog network interface to handle this kind of llog. In this llog interface, the client part is totally belong to network, so this part should be moved to OST/MDT accordingly, while for server part, it should need OSD/MDD support when handling these llog req. the whole process should be

- client send llog req by llog network interface API.
- server accept the llog req and call OSD/MDD llog API to handle the req.

6 State management

6.1 API changes

The filter grant API should be moved from OSD to OST, but the logic should be the same as the current one. The API interface should be the same as the original. The situation is almost the same for network llog, the API is same but called from OST/MDT. Note: here we should care about some llog related callback, if in that callback, some network related stuff will be involved, for example filter_cancel_cookies_cb, then the callback should be put in from network layer instead of others. For example filter_cancel_cookie_cb should be put in from ost_destroy instead of filter_destroy.

7 Focus for inspections

1. Are there other network resources in the new layer?