Multiple Consumers Support for Changelog

Fan Yong

2010-11-01

0.1 Background

Lustre changelog supplies a set of interfaces for recording lustre namespace changes, reading such records, purging specified records, and so on. Currently, all such records will not be purged automatically, even if someone has read such records already. Means such records can be read many times before they are purged explicitly. On the other hand, the current implementation of changelog has no idea about "consumed", it neither knows nor cares about how many consumers want to access the records. It is the HSM engine's duty to decide (according to the so called "policy") which changes should be recorded and how long the records should be retained, just like lustre_rsync does. So even if some reader(s) had not accessed related changelog records yet, it can not prevent others to purge such records explicitly.

0.2 Requirements

- Keep track of multiple registered consumers and only purge records that have been marked consumed by all the registered consumers.
- Evict stale registered consumers after a period of time of inactivity from such registered consumers. (how to? need discussion)
- System administrator can purge specified changelog records explicitly by force, in spite of whether there are registered consumers or not. Such feature is not the same as current implementation which is "lfs" utils on client side, can be triggered by anyone.

0.3 Specification

0.3.1 Consumer type

There are three types changelog consumers: registered consumer, non-registered consumer, and system administrator.

- registered consumer

  It can read changelog records many times as long as such records are available. If someone wants to purge changelog records, and as long as there is at least one registered consumer claims that it has not read related records (to be purged), then such records can not be purged. Any client can register to any MDT as a registered consumer as following:

  lfs changelog_mark [-m mdtname] [-n nextrec#]
  On related MDT(s), assume ‘‘F’’ is the first changelog record index (can be read), ‘‘N’’ is the index to be used for next changelog record.

  - If no ‘‘-m mdtname’’ specified, then registers to all MDTs in the filesystem.
  - If no ‘‘-n nextrec#’’ specified, then tells related MDT(s) that ‘‘F’’ is the first record index preserved for this registered consumer.
  - If ‘‘-n -1’’ specified, then tells related MDT(s) that ‘‘N’’ is the first record index preserved for this registered consumer.
  - If ‘‘-n #’’ specified, if ‘‘# < F’’, then ‘‘F’’, if ‘‘# > N’’, then ‘‘N’’, otherwise ‘‘#’’ is the first record index preserved for this registered consumer.

Once succeed, related MDT(s) will return the real first record index preserved for this registered consumer. Such “lfs changelog_mark” utils is not only for register, but also for updating registered consumer status. When the registered consumer performs such command again (with new “nextrec#”), it tells related MDT(s) the new first record index preserved for this registered consumer, and those old records (before the new index) are meaningless for this registered consumer (can be purged from the view of it). To guarantee the old changelog records can be purged, periodically, such command should be triggered by some application (like lustre_rsync) explicitly after consuming some changelog records.

- non-registered consumer

  It can read changelog records many times as long as such records are available. But if someone wants to purge changelog records, such non-registered consumers will be ignored. Means as long as there is no registered consumer claims that it has not read related records (to be purged), then such records can be purged. Any client can be as a non-registered consumer, which is the default mode, mean if a client never to claim to be as registered, then it is a non-registered consumer. For the registered consumer, it can claim as a non-registered consumer as following:

  lfs changelog_mark [-m mdtname] <-d>
  - If no ‘‘-m mdtname’’ specified, then deregisters from all MDTs in the filesystem.

  To guarantee the old changelog records can be purged, such command should be performed by users or triggered by some application(s) explicitly when it does not want to trace the filesystem changes.

  One client can be either non-registered consumer or registered consumer, but not both at the same time. To make sure whether the client is registered consumer or not, it can be queried by:

  lfs changelog_mark [-m mdtname] <-q>
  - If no ‘‘-m mdtname’’ specified, then will query from all MDTs.

  - For registered consumer, returns the first record index preserved for this registered consumer.
  - For non-registered consumer, returns ‘‘NONE’’.

- system administrator

  1. It can query how many registered consumers on the specified MDT and which their first record index preserved are. As following:

     lctl --device mdtdev changelog_query [-n nid|-u uuid]
     - If no ‘‘-n nid|-u uuid’’ specified, then all registered consumers are queried, otherwise, only queries the specified.

     Once succeed, all the (specified) registered consumer status will be returned, including their nid, uuid and first record index preserved for this registered consumer.

  2. It can evict registered consumer as following:

     lctl --device mdtdev changelog_evict [-n nid|-u uuid]
     - If no ‘‘-n nid|-u uuid’’ specified, then all registered consumers are evicted, otherwise, only evicts the specified.

     It is used when such registered consumer is bad, like in inactive state for more than specified time. Once succeed, all the (specified) registered consumers will be converted to non-registered consumers. The system administrator can perform such command explicitly when needed, and it also can be triggered by the MDT automatically under some cases according to the configuration.

  3. It can purge specified changelog records explicitly by force, in spite of whether there are registered consumers or not. As following:

     lctl --device mdtdev changelog_purge endrec#
     - If ‘‘endrec#’’ is ‘‘-1’’, then all the changelog records are purged.

     Once succeed, all the registered consumers on this MDT will be updated, their first record index preserved will be changed as the specified “endrec#”. It is used when the changelog reaches some large size, and has to release some changelog records for free space. Under such case, all the registered consumers are ignored by force, maybe cause miss to consume related changelog records, so it is some dangerous. We can configure the MDT to trigger it automatically under some cases, but I prefer to performing it by system administrator explicitly.

0.3.2 Protocol changes

Introduce new get_info/set_info key "KEY_CHANGELOG_MARK" for processing "lfs changelog_mark" between client and MDT.

- MDS_SET_INFO

  For claiming to be registered consumer, or updating registered consumer status, or converting to non-registered consumer, uses "MDS_SET_INFO" RPC with such key "KEY_CHANGELOG_MARK". For old server, it does not recognize such key and returns "-EINVAL", which is normal, not an interoperability bug.

  On the other hand, the "MDS_SET_INFO" RPC reply should be expanded. The old format is as following:

```
struct req_format RQF_OBD_SET_INFO = DEFINE_REQ_FMT0
    ("OBD_SET_INFO", obd_set_info_client, empty);
static const struct req_msg_field *empty[] =
    { &RMF_PTLRPC_BODY };
```

  The expanded format is as following:

```
struct req_format RQF_OBD_SET_INFO = DEFINE_REQ_FMT0

    ("OBD_SET_INFO", obd_set_info_client, mds_getinfo_server);
static const struct req_msg_field *mds_getinfo_server[] =
    { &RMF_PTLRPC_BODY, &RMF_GETINFO_VAL, };
```

  For those non-changelog operations, they does not use the expanded field, so no interoperability issue introduced.

- MDS_GET_INFO

  For querying (non-)registered consumer status for the client, uses "MDS_GET_INFO" RPC with such key "KEY_CHANGELOG_MARK". For old server, it does not recognize such key and returns "-EINVAL", which is normal, not an interoperability bug.

0.3.3 MDS side logic

- When MDS receives "MDS_SET_INFO" RPC with key "KEY_CHANGELOG_MARK", it processes it as following:
  - For claiming to be registered consumer, adds it into some list, and records related information into "lsd_client_data()" on MDS disk for persistent store.

  - For updating registered consumer status, updates related list and the "lsd_client_data()" on MDS disk for persistent store.
  - For claiming to be non-registered consumer, removes it from related list, and updates related "lsd_client_data()" on MDS disk for persistent store.

- When MDS receives "MDS_GET_INFO" RPC with key "KEY_CHANGELOG_MARK", it searches related list, and returns the found information.
- When MDS recovers from crash, it rebuilds the registered consumers list according to the information stored in "lsd_client_data()" on MDS disk. To guarantee such information are available, using sync write when updates related information on MDS disk.
- All the registered consumers on the same MDT should be collated by their first record index preserved, then when someone wants to clear some changelog records, it can search whether there are some registered consumers that they have not consumed related records (to be purged). When registered consumers update their first record index preserved, the list should be reordered also.
- All the registered consumers on the same MDT should be hashed by their nid or uuid, then system administrator can find them easily.
- Once the client is evicted by MDT or disconnected, then it will be removed from above list(s), and related information recorded in "lsd_client_data()" on MDS disk is invalidate also, means non-registered consumer.
- If the client reconnects to MDT, then updates above list(s) and the information in related "lsd_client_data()" on MDS disk.
- When system administrator purges changelog records by force, then updates above list(s) and the information in related "lsd_client_data()" on MDS disk.
- Configure MDT to allow to purge changelog records by force (ignore registered consumer) automatically when changelog reaches some specified size, and to allow to evict stale registered consumer automatically. Will discuss those in "Options & discussion" section.

0.3.4 Client side logic

- Keep current changelog code on client side, just implement the new interfaces: "lfs changelog_mark", no need special description for this, it is relative straight and not complex.
- When client restarts or disconnects from MDS, the registered consumer role will be canceled automatically.
- When server restarts from crash, the registered consumer role for the client(s) will be recovered automatically.

Other three things should be mentioned.

1. changelog record purged

   If system administrator performs "lctl changelog_purge", it maybe cause the registered consumer missing some changelog record(s) when "next_block" read, and then return error to application (HSM engine). Under such case, it is the application's duty to determine whether continue to consume the remaining changelog records or exit with failure.

2. changelog evicted

   If system administrator performs "lctl changelog_evict", it will cause the registered consumer converted to be non-registered one. The client will be notified that when "next_block" read. Under such case, the client can claim as registered consumer again by triggering "changelog_mark" as described above if it should not be evicted, it also can just run as non-registered consumer. If someone purged some changelog records after such changelog eviction, then the client should process similarly as described in "changelog record purged".

3. registered consumer ping

   To guarantee the registered consumer will not be evicted for long time of inactivity, will discuss it in "Options & discussion" section.

0.3.5 Resend & replay

MDS can process the resend "MDS_SET_INFO" RPC with key "KEY_CHANGELOG_MARK" just as the normal one (first send) without any harm. On the other hand, registered consumer status is stored as part of "lsd_client_data()" on MDS disk, which is persistent. If MDS starts up from crash, it will load all these information from MDS disk, then the former registered consumer status can be recovered automatically. So no special replay process from client side for the changelog enhancement.

0.3.6 Scalability & performance

Usually, the registered consumer count is not too large (not more than tens), so related list described above is not long. Purging changelog record operation is seldom compared with other metadata operations. So the overhead introduced by purging changelog record operation can be ignored. On the other hand, the RPC count for "KEY_CHANGELOG_MARK" depends on the application (HSM's engine) on client side, if the registered consumer status update period is not so short, like one RPC per 1000 records or per minute, then the overhead introduced by "KEY_CHANGELOG_MARK" RPC is relative small also.

0.3.7 Options & discussion

1. Clever MDT can automatically catch the information of how many changelog records have been read by the client(s), and which is the next changelog record to be read, so it seems that the registered consumers need not to update their information periodically, then some "MDS_SET_INFO" RPCs are saved. But in fact, it is not true. Because we do not know whether the changelog records read by the client are consumed successfully by the application on client side or not, and we also should not restrict how many times the same changelog record can be read by the same client. It is the application's duty to determine when to discard the old changelog records explicitly.

2. Configure MDT to allow to purge changelog records by force (ignore registered consumer) automatically when changelog reaches some specified size. It is some dangerous to do that. Instead, if changelog is so large as to can not contain any more records, we can prevent further lustre namespace changes, just like disk full case. I am not sure it is reasonable enough. Need more consideration and discussion for that.

3. Configure MDT to allow to evict stale registered consumer automatically. Then how to define the "stale"? It is different from "stale" client by checking whether there is RPC from the client in a specified period. Maybe the client alive (has heartbeat), but has not read changelog records (or not update the registered consumer status) for a long time, then we think such inactive registered consumer is "stale". But in fact, it is application's duty to decide how to consume the changelog record with how speed. On the other hand, when replicates lustre system to remote, it is possibly blocked by remote system, the application can not control that also. We can introduce some thread (or let the application) on client side which will update the registered consumer status automatically periodically, just like "ping" operation. But it can not guarantee the first record index preserved for the registered consumer can move forward. So under such case, whether it is meaningful to do that or not? whether should allow MDT to evict stale registered consumer automatically or not? Or just allow system administrator to do that by hand?