Ptlrpc and related modules cleanup

The purpose of cleanup is to simplify client side code, make Lustre client be easier to be accepted/merged to upstream Linux kernel. By the cleanup, kinds of server side specific handlings are removed, kernel modules image size and memory footprint are reduced, some server side kernel modules or kernel threads need not be loaded/started at client.

This document is the design of ptlrpc and related modules cleanup. Section 1 illustrates the cleanup works in detail. The cleanup introduces some differences between Lustre's client side code and Lustre main code tree, and section 2 will discuss about the side effects and code maintenance.

1 Client side cleanup

1.1 Remove server side connection/disconnection handling

Several server side connection/disconnection functions can be removed from client.

Impacted files: ldlm/ldlm_lib.c, ldlm/ldlm_lockd.c.

These functions can be removed (grey color is for internal static function): server_disconnect_export, target_handle_reconnect, target_client_add_cb, target_handle_connect, target_handle_disconnect, and target_destroy_export.

Those functions are only used at server side. But as both obdecho client and server are encapsulated in a single kernel module, we need to split it as next subsection described.

1.2 Split obdecho client and server

Obdecho is a testing/benchmarking tool for Lustre, it is not in the critical path but we need to split the obdecho client and server to make the cleanup in section 1.1 being safe.

Impated files: obdecho/echo.c, obdecho/echo_client.c, and building process.

Main changes are obdecho_init/obdecho_exit and building process. Client side built obdecho.ko only has client side handling, while obdecho.ko at server side has both client and server handling. So users can use it same as before. We can also build kernel module with different name (obdecho_cli.ko and obdecho_srv.ko) as discussed in section 2.1.

1.3 Remove server side recovery handling

Sever side recovery handling can be removed from client, this is quite a big stuff for cleanup.

Impacted files: ldlm/ldlm_lib.c, ldlm/ldlm_lockd.c, ptlrpc/p tlrpc_module.c, ptlrpc/recov_thread.c, lustre/include/lustre_log.h.

These functions can be removed: target_request_copy_get, target_request_copy_put, target_exp_enqueue_req_replay, target_finish_recovery, abort_req_replay_queue, abort_lock_replay_queue, target_cleanup_recovery, target_cancel_recovery_timer, target_start_recovery_timer, extend_recovery_timer, check_and_start_recovery_timer, exp_connect_healthy, exp_req_replay_healthy, exp_lock_replay_healthy, exp_vbr_healthy, exp_finished, check_for_clients, check_for_next_transno, check_for_next_lock, target_recovery_overseer, target_next_replay_req, target_next_replay_lock, target_next_final_ping, handle_recovery_req, target_recovery_thread, target_start_recovery_thread, target_stop_recovery_thread, target_recovery_fini, target_recovery_expired, target_recovery_init, target_process_req_flags, target_queue_recovery_request.

Client side ptlrpc_init/ptlrpc_exit can be changed to not call llog_recov_init/ llog_recov_fini. The main functions in prltpc/recov_thead.c can be removed, except that llog_obd_repl_cancel (and its internal static functions) would remain to be used by client side OSC module.

1.4 *Remove server side bulk I/O*

For bulk I/O, client side use ptlrpc_register_bulk/ptlrpc_unregister_bulk to register bulk buffer. Server side issued bulk I/O related functions can be removed from client.

Impacted files: ldlm/ldlm_lib.c, ptlrpc/sec.c, ptlrpc/niobuf.c, ptlrpc/ptlrpc_module.c.

These functions can be removed:
Ldlm/ldlm_lib.c’s target_bulk_io,
ptlrpc/sec.c’s sptlrpc_svc_wrap_bulk, sptlrpc_svc_unwrap_bulk, sptlrpc_svc_prep_bulk,
ptlrpc/niobuf.c’s ptlrpc_start_bulk_transfer, ptlrpc_abort_bulk.

1.5 *Remove server side specific lock handling*

Some server side specific lock handling can be removed from client side.

1.5.1 Remove server side issued ASTs

Blocking/completion/glimpse ASTs are sent from server to client, related codes can be removed from client.

Impacted files: ldlm/ldlm_lockd.c.

These functions can be removed: ldlm_server_blocking_ast, ldlm_server_completion_ast, ldlm_server_glimpse_ast, and their internal static functions ldlm_failed_ast, ldlm_handle_ast_error, ldlm_cb_interpret, ldlm_bl_and_cp_ast_tail, ldlm_lock_reorder_req.

1.5.2 Need not start ldlm_cancel_service

ldlm_cancel_service need not be started at client side.

Impacted files: ldlm/ldlm_lockd.c

Change ldlm/ldlm_lockd.c’s ldlm_setup/ldlm_cleanup, need not start/stop ldlm_cancel_service. ldlm_cn_xx kernel threads will not be started at client side.
These functions can be removed: ldlm_cancel_handler, ldlm_handle_cancel, ldlm_request_cancel.

1.5.3 Remove ldlm policy functions

Ldlm policy functions are only used by server, we can remove it from client. However, we need some other code changes to remove some code dependencies.

Impacted files: ldlm/ldlm_lock.c, ldlm/ldlm_lockd.c, ldlm/ldlm_plain.c, ldlm/extend.c, ldlm/ldlm_flock.c, ldlm/inodebits.c.

Changes ldlm/ldlm_lock.c’s ldlm_lock_enqueue, removes ldlm_processing_policy_table calling as actually it is not called at client (if (local) GOTO out;).
Changes ldlm/ldlm_lock.c’s ldlm_reprocess_all, removes calling of ldlm_reprocess_queue (inside which calls ldlm_processing_policy_table) as actually it is not called at client (if (ns_is_client) return;).
Changes ldlm/ldlm_lock.c’s ldlm_lock_convert, removes calling of ldlm_processing_policy_table as actually it is not called at client (if ns_is_client else).

By the above changes, we can remove the client side dependency of ldlm_processing_policy_table. Then we can remove related policy functions at client side: ldlm_process_plain_lock, ldlm_process_extent_lock, ldlm_process_flock_lock, ldlm_process_inodebits_lock. And many internal static functions inside ldlm/ldlm_plain.c, ldlm/extend.c, ldlm/ldlm_flock.c and ldlm/inodebits.c can be removed from client.

1.5.4 Remove ldlm_handle_enqueue /ldlm_handle_convert etc.

Impacted files: ldlm/ldlm_lockd.c, ldlm/ldlm_lock.c.

These functions are only used by server side and can be removed from client: ldlm/ldlm_lockd.c's ldlm_svc_get_eopc, ldlm_handle_enqueue0, ldlm_handle_enqueue, ldlm_handle_convert0, ldlm_handle_convert, ldlm_revoke_export_locks, ldlm_revoke_lock_cb; ldlm/ldlm_lock.c's ldlm_lock_downgrade, ldlm_cancel_locks_for_export, ldlm_cancel_locks_for_export_cb.

1.6 Remove lquota module from client

Now lquota kernel module has no dependencies at client, but will be compiled and loaded at client side by default.

There are some other quota related codes bracketed by "#ifdef HAVE_QUOTA_SUPPORT … #endif" macro. These codes need not to be compiled for client.

We can change the building process – only compile lquota and define HAVE_QUOTA_SUPPORT as 1 when Lustre is configured without "--disable-server" option **and** without "--enable-quota=no" option.

1.7 Others

There are some Obsolete functions in the code and without any callers, such as ptlrpc_prep_req, ptlrpc_prep_req_pool, llog_handle_connect, ldlm_cli_convert etc. We can remove them in the case that it need not be remained for future's reference.

There are also some only server side needed functions, such as target_handle_ping, target_committed_to_req etc.

However, it is difficult and un-necessary to make client side code to be "fully clean". Currently we only need to remove main server side handling mentioned from section 1.1 to section 1.6.

2 Side effects and code maintenance

Because the code changes introduced in section 1, there will be some code differences between client side and server side. We need to consider some side effects and the code maintenance.

2.1 Use same or different kernel module name

For example, the kernel module ptlrpc and obdecho will be different in client and server.

Option 1: use same kernel module name.
Don't change the kernel module names, client and server side use the same kernel module name, but possibly with different content. Server side kernel modules will have all content including client side code as before, while client side kernel modules will be a reduced version – some server specific handling code is removed.

Option 2: use different kernel module name.
For example, client side loads ptlrpc_cli.ko, while server side loads ptlrpc_srv.ko. In the case that Lustre client will be loaded in server, server side will load both ptlrpc_cli.ko and ptlrpc_srv.ko. Here we may need to separate the common parts(ptlrpc_comm.ko) which ptlrpc_cli.ko and ptlrpc_srv.ko shares.

Option 2 will introduce significant change to the code tree (need to split the client part, server part and common part) and building process, will affect some users' conventional usage, also need to change some testing tools/scripts.

2.2 Code maintenance

How to maintain client side code (which needs to be pushed to upstream Linux kernel) together with Lustre main code tree?

Option 1: use macro to comment out the cleaned-up codes for client
Use a special macro to comment out the cleaned-up codes. Here we have two methods for the macro:

a) use pre-defined HAVE_SERVER_SUPPORT macro
Users will get different kernel modules when compiling Lustre with or without "--disable-server" configuration option. Some kernel modules inside lustre-client-modules-xxx.rpm and lustre-modules-xxx.rpm may have same name and different content.
b) use other specific new macro such as CLIENT_SIDE_CLEANUP
This will not affect common usage.

We can use a tool to fork a client-side code tree from Lustre main code tree. This tool can remove server side specific handling (by recognizing "#ifdef HAVE_QUOTA_SUPPORT … #endif" or "#ifdef CLIENT_SIDE_CLEANUP … #endif") while coping out source files from Lustre main code tree. The tool can be a separate script or just using automake framework (for example: make cleanclient).

After Lustre client being merged to upstream Linux kernel, in the case that somebody finds bugs in kernel and submits patch for that, that patch can also be patched to Lustre main code tree (possibly with some warnings as file offset difference).

Option 2: Fully split client part, server part and common part codes
Split the client part, server part and common part codes. This will introduce many significant code changes to Lustre main code tree, also will change many source code files' name (such as xxx_cli.c, xxx_srv.c, xxx_comm.c). The building process also need changes. It is difficult to make the split in the case we change common functions, for example modue_init or ldlm_lock_enqueue etc.