CMD3 Stability Fixing

Huang Hua

2007-04-02

1 Functional Specification

1.1 Introduction

This task is a post cmd3 stability fixing. Its main purpose is to to make CMD lustre stable, adding some features which are not implemented or not done well. Most of them are due to time constrains in last cycle.

1.2 req_capsule_shrink()

```
int req_capsule_shrink(const struct req_capsule *pill,
                       const struct req_msg_field *field,
                       const unsigned int newlen,
                       const int adjust,
                       const int move_data);
```

Parameters: const struct req_capsule *pill - request capsule pill which need shrink on the reply message;

const struct req_msg_field *field - indicate which reply segment (or called massage field) need shrink;

const unsigned int newlen - new length of the specified field;

const int adjust - add this number internally to the specified segment because some segments may be have been removed due to zero-length and move_data is not zero.

const int move_data - if this is not zero, and new length is zero, specified segment will be removed, and following segment(s) will be moved forward.

Return Value: Return 1 if segment is moved forward. Otherwise return zero.

Description: This function is used to shrink reply message buffer. This function should be called for segments, from lower to higher (that is from lower index to higher index). Some segments may be removed and higher one moved forward. So the @adjust is used to adjust the index (or called offset).

1.3 osd_object_create()

1.3.1 Data structures

In order to optimize the object allocation on dt device, we'd like to introduce a general purpose allocation hint. This hint is based on the parent object of the new object, the mode of the new object, and other parameters.

```
/*
 * This is a general purpose dt allocation hint.
 * It now contains the parent object and mode.
 * It can contain any allocation hint in the future.
 */
struct dt_allocation_hint {
        struct dt_object *dah_parent;
        __u32             dah_mode;
};
```

1.3.2 do_ah_init()

```
/*
 * Init allocation hint using parent object and child mode.
 * (1) The @parent might be NULL if this is a partial creation for
 *     remote object.
 * (2) The type of child is in @child_mode.
 * (3) The result hint is stored in @ah;
 */
int   (*do_ah_init)(const struct lu_env *env,
                    struct dt_allocation_hint *ah,
                    struct dt_object *parent,
                    umode_t child_mode);
```

Parameters: const struct lu_env *env - pointer to env;

struct dt_allocation_hint *ah - the result allocation hint is stored there. This pointer should be valid.

struct dt_object *parent - parent object.

umode_t child_mode - mode of new child object.

Return Value: Zero indicates success, other values indicate error.

Description: This function initialize the allocation hint for new object. It takes parent object and child mode as arguments. This result allocation hint should be passed to do_create().

1.3.3 do_create()

```
/*
 * Create new object on this device.
 *
 * precondition: !dt_object_exists(dt);
 * postcondition: ergo(result == 0, dt_object_exists(dt));
 */
int   (*do_create)(const struct lu_env *env,
                   struct dt_object *dt,
                   struct lu_attr *attr,
                   struct dt_allocation_hint *hint,
                   struct thandle *th);
```

Parameters: const struct lu_env *env - pointer to env;
struct dt_object *dt - new dt object;
struct lu_attr *attr - desired object attributes.
struct dt_allocation_hint *hint - allocation hint for this object;
struct thandle *th - transaction handle.

Return Value: Zero indicate success, and other values indicate error.

Description: An allocation hint is added to the function parameters. This helps to get a more intelligent and optimized algorithm.

1.4 mdd_lov_destroy()

```
int mdd_lov_destroy(const struct lu_env *env,
                    struct mdd_device *mdd,
                    struct mdd_object *obj,
                    struct lu_attr *la)
```

Parameters: const struct lu_env *env - pointer to env;
struct mdd_device *mdd - mdd device;
struct mdd_object *obj - mdd object to be destroyed;
struct lu_attr *la - attribute of this object.

Return Value: Zero indicate success, and other values indicate error.

Description: This function will destroy the corresponding lov object.

1.5 FIXME and TODO

1. name length checking. We shall use the actual name length instead of buffer length.

1.6 join file feature

The join file feature will be added again. This feature was not added due to time constrains in previous cycle. The design document can be found in ../DLD/LOVEA_DLD.lyx and ../DLD/MDSEA_DLD.lyx. This document will not include more description about join file feature.

1.7 remove unused Linux kernel patches

Unused kernel patches will be removed. Those patches are used for debugging, or performance, or other. They are not needed currently.

1.8 Reconnection refusal deadlock

class_export_rpc_put/get() can lead to 'deadlock' situation after disconnection:

1. client's request is in progress and server is waiting for the same client to reply AST;
2. Due to some network issue, the client disconnects from server;
3. Client tries to re-connect to the server; But the server knows that some requests from the same client is in progress, and refuse its connection.
4. The server is still waiting for this client to sand back AST reply, but timeout. So the server evicts this client, and client gets an application error.
5. This application error result maybe is not acceptable in some cases.

What we should do is to avoid this 'deadlock'.

1.9 Readdir()

We use hash value of sub-item as directory offset and as page index while doing readdir(). We may experience hash value collision in some cases, and need to handle this case correctly and efficiently. The problem arises when more than one pages have the same hash value, because page is indexed by hash value. The solution is to chain those pages with the same hash value into a list, and do not insert them into page cache.

Currently, we use TEA hash algorithm, and experience hash collision very rare. So this task has very low priority. The solution with more detail is described in the code.

1.10 llog in setattr for quota

llog in setattr for quota purpose is missing due to time constrains. This should be done just like unlink llog.

2 Use Cases

2.1 osd object creation with hint

Before calling underlying dt object creation function, mdd should initialize the allocation hint, and then pass to the dt object creation routine. The following code illustrate how to init allocation hint, and then call underlying object creation routine.

```
int
mdd_object_create_internal(const struct lu_env *env,
                           struct mdd_object *p,
                           struct mdd_object *c,
                           struct md_attr *ma,
                           struct thandle *handle)
{
        struct lu_attr *attr = &ma->ma_attr;
        struct dt_allocation_hint *hint = &mdd_env_info(env)->mti_hint;
        int rc;
        ENTRY;
        if (!mdd_object_exists(c)) {
                struct dt_object *next = mdd_object_child(p);
                rc = next->do_ops->do_ah_init(env, hint, next,
                                              attr->la_mode & S_IFMT);
                if (rc == 0) {
                        rc = mdo_create_obj(env, c, attr, hint, handle);
```

```
                        LASSERT(ergo(rc == 0, mdd_object_exists(c)));
                }
        } else
                rc = -EEXIST;
        RETURN(rc);
}
```

2.2 mdt shrink reply message

Calling to req_capsule_shrink() should be applied from lower segment to higher segment. No explicit buffer offset will be avoided in new implementation. Various situation should be tested.

```
void mdt_shrink_reply(struct mdt_thread_info *info)
{
        struct ptlrpc_request *req = mdt_info_req(info);
        struct req_capule     *pill = &info->mti_pill;
        struct mdt_body       *body;
        int acl_size, md_size, adjust = 0;
        body = req_capsule_server_get(pill, &RMF_MDT_BODY);
        LASSERT(body != NULL);
        if (body->valid & (OBD_MD_FLDIREA | OBD_MD_FLEASIZE |
                           OBD_MD_LINKNAME))
                md_size = body->eadatasize;
        else
                md_size = 0;
        acl_size = body->aclsize;
/*
 *  The most common reply message format is:
 *             &RMF_MDT_BODY,
 *             &RMF_MDT_MD,
 *             &RMF_ACL, or &RMF_LOGCOOKIES
 *(optional)  &RMF_CAPA1,
 *(optional)  &RMF_CAPA2,
 */

        adjust += req_capsule_shrink(pill, &RMF_MDT_MD,
                                     md_size, adjust, 1);
        if (req_capsule_has_field(pill, &RMF_ACL, RCL_SERVER))
                adjust += req_capsule_shrink(pill, &RMF_ACL,
                                             acl_size, adjust, 1);
        else if (req_capsule_has_field(pill, &RMF_LOGCOOKIES, RCL_SERVER))
                adjust += req_capsule_shrink(pill, &RMF_LOGCOOKIES,
                                             acl_size, adjust, 1);
```

```
        if ((req_capsule_has_field(pill, &RMF_CAPA1, RCL_SERVER) &&
               !(body->valid & OBD_MD_FLMDSCAPA)))
                adjust += req_capsule_shrink(pill, &RMF_CAPA1,
                                             0, adjust, 1);
        if ((req_capsule_has_field(pill, &RMF_CAPA2, RCL_SERVER) &&
                !(body->valid & OBD_MD_FLOSSCAPA)))
                adjust += req_capsule_shrink(pill, &RMF_CAPA2,
                                             0, adjust, 0);
}
```

2.3 join file

Join file use cases and test cases are all listed in sanity.sh shell script.
Please refer to that script for these information.

2.4 llog for setattr

```
/* set attr and LOV EA at once, return updated attr */
static int mdd_attr_set(const struct lu_env *env, struct md_object *obj,
                        const struct md_attr *ma)
{
        struct mdd_object *mdd_obj = md2mdd_obj(obj);
        struct mdd_device *mdd = mdo2mdd(obj);
        struct thandle *handle;
        struct lov_mds_md *lmm = NULL;
        struct llog_cookie *logcookies = NULL;
        int  rc, lmm_size = 0, max_size = 0, cookie_size = 0;
        struct lu_attr *la_copy = &mdd_env_info(env)->mti_la_for_fix;
        ENTRY;
        mdd_txn_param_build(env, mdd, MDD_TXN_ATTR_SET_OP);
        handle = mdd_trans_start(env, mdd);
        if (IS_ERR(handle))
                RETURN(PTR_ERR(handle));
        /*
         * if this is a regular file and want to change uid or gid,
         * we will llog this operation, because this will affect quota
         * asynchronously on OST.
         */
        if (S_ISREG(mdd_object_type(mdd_obj)) &&
            ma->ma_attr.la_valid & (LA_UID | LA_GID)) {
                /* get lov ea for this regular file */
                max_size = mdd_lov_mdsize(env, mdd);
                OBD_ALLOC(lmm, max_size);
                lmm_size = max_size;
```

```
                if (lmm == NULL)
                        GOTO(cleanup, rc = -ENOMEM);
                rc = mdd_get_md_locked(env, mdd_obj, lmm, &lmm_size,
                                MDS_LOV_MD_NAME);
                if (rc < 0)
                        GOTO(cleanup, rc);
        }

        ...
        if (la_copy->la_valid & LA_FLAGS) {
                rc = mdd_attr_set_internal_locked(env, mdd_obj, la_copy,
                                                  handle, 1);
                if (rc == 0)
                        mdd_flags_xlate(mdd_obj, la_copy->la_flags);
        } else if (la_copy->la_valid) {            /* setattr */
                rc = mdd_attr_set_internal_locked(env, mdd_obj, la_copy,
                                                  handle, 1);
                /* journal chown/chgrp in llog, just like unlink */
                if (rc == 0 && lmm_size){
                        cookie_size = mdd_lov_cookiesize(env, mdd);
                        OBD_ALLOC(logcookies, cookie_size);
                        if (logcookies == NULL)
                                GOTO(cleanup, rc = -ENOMEM);
                        /* add setattr to llog */
                        if (mdd_setattr_log(env, mdd, ma, lmm, lmm_size,
                                            logcookies, cookie_size) <= 0) {
                                OBD_FREE(logcookies, cookie_size);
                                logcookies = NULL;
                        }
                }
        }
        if (rc == 0 && ma->ma_valid & MA_LOV) {
                umode_t mode;
                mode = mdd_object_type(mdd_obj);
                if (S_ISREG(mode) || S_ISDIR(mode)) {
                        /*TODO check permission*/
                        rc = mdd_lov_set_md(env, NULL, mdd_obj, ma->ma_lmm,
                                            ma->ma_lmm_size, handle, 1);
                }
        }
cleanup:
        mdd_trans_stop(env, mdd, rc, handle);
        if (rc == 0 && (lmm != NULL && lmm_size > 0 )) {
                /*
                 * set obd attr, if needed.
                 */
```

```
                rc = mdd_lov_setattr_async(env, mdd_obj, lmm, lmm_size, logc
        }
        if (lmm != NULL)
                OBD_FREE(lmm, max_size);
        if (logcookies)
                OBD_FREE(logcookies, cookie_size);
        RETURN(rc);
}
```

3 Logic Specification

3.1 req_capsule_shrink()

```
/*
 * Shrink the specified reply message buffer @field to a specified @newlen.
 * If @move_data is non-zero, then move the following buffer forward
 *    if @newlen is zero;
 * The internal offset should be adjusted by @adjust because buffer maybe ha
 *    been moved by previous call. (@adjust >= 0) is a must.
 * Return value: 1 if buffer has been moved, otherwise 0 is returned.
 */
int req_capsule_shrink(const struct req_capsule *pill,
                       const struct req_msg_field *field,
                       const unsigned int newlen,
                       const int adjust,
                       const int move_data)
{
        int offset;
        int moved;
        LASSERT(adjust >= 0);
        LASSERT(req_capsule_has_field(pill, field, RCL_SERVER));
        offset = __req_capsule_offset(pill, field, RCL_SERVER);
        offset -= adjust;
        LASSERT(offset >= 1);
        lustre_shrink_reply(pill->rc_req, offset, newlen, move_data);
        return (newlen == 0 && move_data) ? 1 : 0;
}
```

3.2 osd_object_create()

osd_object_create() is the dt implementation of do_create(). It receives a @dt_allocation_hint argument. It should use this hint to create new object.

The main change happens in osd_mkfile():

```
static int osd_mkfile(struct osd_thread_info *info,
                      struct osd_object *obj,
                      umode_t mode,
                      struct dt_allocation_hint *hint,
                      struct thandle *th)
{
        int result;
        struct osd_device  *osd = osd_obj2dev(obj);
        struct osd_thandle *oth;
        struct inode       *parent;
        struct inode       *inode;

        LASSERT(osd_invariant(obj));
        LASSERT(obj->oo_inode == NULL);
        LASSERT(osd->od_obj_area != NULL);
        oth = container_of(th, struct osd_thandle, ot_super);
        LASSERT(oth->ot_handle->h_transaction != NULL);

        /* use this hint if it is valid. */
        if (hint && hint->dah_parent)
                parent = osd_dt_obj(hint->dah_parent)->oo_inode;
        else
                parent = osd->od_obj_area->d_inode;
        LASSERT(parent->i_op != NULL);
        inode = ldiskfs_create_inode(oth->ot_handle, parent, mode);
        if (!IS_ERR(inode)) {
                obj->oo_inode = inode;
                result = 0;
        } else
                result = PTR_ERR(inode);
        LASSERT(osd_invariant(obj));
        return result;
}
```

3.3 Remove unused kernel patches

We will remove those patches which are used only for debugging usage purposes. We will also add some patches which are added in b1_6, such as tcp-zerocopy, and quotas.

```
Index: lustre/kernel_patches/series/2.6-rhel4.series
===================================================================
RCS file: /cvsroot/cfs/lustre-core/kernel_patches/series/2.6-rhel4.series,v
```

```
retrieving revision 1.1.2.3.8.1.4.3.2.30.2.1
diff -u -p -r1.1.2.3.8.1.4.3.2.30.2.1 2.6-rhel4.series
--- lustre/kernel_patches/series/2.6-rhel4.series       28 Mar 2007 08:26:03
+++ lustre/kernel_patches/series/2.6-rhel4.series       9 Apr 2007 02:49:06
@@ -8,17 +8,18 @@ iopen-misc-2.6-suse.patch
 export-truncate-2.6-suse.patch
 export_symbols-2.6-rhel4.patch
 dev_read_only-2.6-suse.patch
 lookup_bdev_init_intent.patch
 remove-suid-2.6-suse.patch
 export-show_task-2.6-vanilla.patch
 sd_iostats-2.6-rhel4.patch
+blkdev_tunables-2.6-suse.patch
 fsprivate-2.6.patch
 export_symbol_numa.patch
 qsnet-rhel4-2.6.patch
 linux-2.6-binutils-2.16.patch
 vm-tunables-rhel4.patch
+tcp-rto_proc-2.6.9.patch
 iallocsem_consistency.patch
 raid5-stats.patch
 raid5-configurable-cachesize.patch
@@ -28,14 +29,9 @@ raid5-merge-ios.patch
 raid5-serialize-ovelapping-reqs.patch
 jbd-stats-2.6.9.patch
 bitops_ext2_find_next_le_bit-2.6.patch
+quota-deadlock-on-pagelock-core.patch
+quota-umount-race-fix.patch
+quota-deadlock-on-pagelock-ext3.patch
+new-tcp-zero-copy-2.6.9-41.2chaos.patch
 dynamic-locks-2.6.9.patch
 export-nr_free_buffer_pages.patch
-debugging-fields-in-current.patch
-increase-BH_LRU_SIZE.patch
-linux-2.6.9-network_driver-for-sk98.patch
-ipoib_tcpdump.patch
```

3.4 Reconnection refusal in recovery

This is done by Wang Di, but in a tricky way. The current solution is:

1. add “atomic_t exp_waiting_count” to “struct obd_export” to indicate that the server is waiting for some AST reply;
2. before waiting for AST reply, server increase this count; And when got reply, decrease this count;

3. When server gets re-connect request from client, check if the server is waiting for this client to send back reply. If so, even if there are some requests from the same client are in progress, re-connection is permitted.

4. The disadvantage of this solution is that: the server may process the same requests twice.

New solution is under investigation. More details will be added here if new solution appears.

3.5 llog in setattr for quota

This is a port from HEAD branch (it comes from b1_5 and will become b1_6 soon). This is missed because of time constrains. We need to add it back just like unlink llog. Refer to use cases for more information.