uMDS Transaction API

Amit Sharma

24 Jan 2008

1 Introduction

This document outlines the detailed design for the uMDS transaction api task, which aims to replace the old transaction api with the new transaction api. The high level design has been outlined in the document umds-txn-api-hld.lyx.

2 Functional Specification

The transaction model currently in use provides the following api's :

```
dt_trans_start()
dt_trans_stop()
```

But, with the new transaction api's, a little more work is needed as in the transaction also has steps to make declaration for the action being performed, which is part of the transaction model. So, the complete list of api's in the new transaction model to be used is as follows :

```
Interfaces to manipulate transactions :
dt_trans_create()
dt_trans_start()
dt_trans_stop()
To declare intention of object manipulation :
do_declare_create()
do_declare_ref_add()
do_declare_ref_del()
do_declare_attr_set()
To declare intent of data modification
dbo_declare_write()
To declare index manipulation
dio_declare_insert()
dio_declare_delete()
```

In the current model only two steps are done as part of using the transactions, as follows:

```
Step 1. dt_trans_start()
Step 2. Perform the changes (write/create/delete.. etc.)
Step 3. dt_trans_stop()
```

With the new transaction model the above changes and would look something like the following:

```
Step 1. dt_trans_create()
Step 2. declare the changes
Step 3. dt_trans_start()
Step 4. perform the changes (write/create/delete.. etc.)
Step 5. dt_trans_stop()
```

3 Use Cases

- mdd_create
- mdd_unlink
- mdd_link
- mdd_name_insert
- mdd_name_remove
- mdd_rename_tgt
- mdd_create_data
- mdd_rename
- mdd_object_create
- mdd_object_delete
- mdd_attr_set
- mdd_xattr_set
- mdd_xattr_del
- dd_ref_del
- mdd_ref_add
- mdd_close

4 Logic Specification

The pseudo code implementation of some of the above are shown below.

4.1 mdd_create

The basic outline of the code would be as follows :

```
th = mdd_trans_create()
->do_declare_create(th)
->dio_declare_insert(th, parent, name)
mdd_trans_start(th)
mdd_pdo_write_lock()
mdd_write_lock()
->do_create()
mdd_write_unlock()
__mdd_index_insert()
mdd_lov_set_md()
mdd_pdo_write_unlock()
mdd_trans_stop(th)
```

New code would look somewhat like this :

```
static int mdd_create(const struct lu_env *env, struct md_object *pobj,
    const struct lu_name *lname, struct md_object *child, struct md_op_spec *spec,
    struct md_attr* ma)
{
    ...
    ...
    mdd_txn_param_build(env, mdd, MDD_TXN_MKDIR_OP);
    handle = mdd_trans_create(env, mdd, mdd_env_info(env)->mti_param);

    rc = mdd_declare_create(env, mdd, handle);
    if (IS_ERR(rc) {
        GOTO(out_trans, rc);
    }
    rc = mdd_trans_declare_index_insert(env, mdd, valsize, dt_key, thandle);
    if (IS_ERR(rc)) {
        GOTO(out_trans, rc);
    }

    rc = mdd_trans_start(env);
    if (IS_ERR(rc)) {
        GOTO(out_trans, rc);
    }
    dlh = mdd_pdo_write_lock(env, mdd_pobj, name);
    ...
    ...
out_trans:
    mdd_trans_stop(env, thandle)
```

```
    ...
}
```

4.2 mdd_unlink

```
th = mdd_trans_create()
->do_declare_ref_del(th)
->dio_declare_delete(th, parent, name)
llog_declare_new_record(th, lmm)
mdd_trans_start(th)
mdd_pdo_write_lock()
mdd_write_lock()
__mdd_index_delete()
mdo_ref_del()
mds_log_op_unlink()
mdd_write_unlock()
mdd_pdo_write_unlock()
mdd_trans_stop(th)
```

New code would look somewhat like this :

```
static int mdd_unlink(const struct lu_env *env, struct md_object *pobj,
    struct md_object *cobj, const struct lu_name *lname,
    struct md_attr *ma) {
...
...
rc = mdd_log_txn_param_build(env, cobj, ma, MDD_TXN_UNLINK_OP);
if (IS_ERR(rc))
    GOTO(out_trans, rc);
handle = mdd_trans_create(env, cobj, mdd_env_info(env)->mti_param);
do_declare_ref_del(env, cobj, handle);
rc = dio_declare_delete(env, mdd, key, handle);
if (IS_ERR(rc))
    GOTO(out_trans, rc);
rc = mdd_declare_ref_del(env, cobj, handle);
if (IS_ERR(rc))
    GOTO(out_trans, rc);
rc = mdd_trans_start(env);
if (IS_ERR(rc)) {
    GOTO(out_trans, rc);
}
dlh = mdd_pdo_write_lock(env, mdd_pobj, name);
...
...
out_trans:
```

```
        mdd_trans_stop(env, thandle)
        ...
    }
```

4.3 mdd_link

The basic outline of the code would be as follows :

```
th = mdd_trans_create()
->do_declare_ref_add(th, target)
->dio_declare_insert(th, parent, name)
mdd_trans_start(th)
mdd_pdo_write_lock()
mdd_write_lock()
__mdd_index_insert_only()
mdo_ref_add()
mdd_write_unlock()
mdd_pdo_write_unlock()
mdd_trans_stop(th)
```

New code would look somewhat like this :

```
static int mdd_link(const struct lu_env *env, struct md_object *tgt_obj,
    struct md_object *src_obj, const struct lu_name *lname,
    struct md_attr *ma)
{
    ...
    mdd_txn_param_build(env, mdd, MDD_TXN_LINK_OP);
    handle = mdd_trans_create(env, mdd, mdd_env_info(env)->mti_param);

    rc = do_declare_ref_add(env, mdd, handle);
    if (IS_ERR(rc)) {
        GOTO(out_trans, rc);
    }
    rc = dio_declare_insert(env, mdd, valsize, dt_key, handle);
    if (IS_ERR(rc)) {
        GOTO(out_trans, rc);
    }

    rc = mdd_trans_start(env);
    if (IS_ERR(rc)) {
        GOTO(out_trans, rc);
    }
    ...
    ...
out_trans:
```

```
        mdd_trans_stop(env, mdd, rc, handle);
    }
```

4.4 mdd_rename

The basic outline of the code would be as follows :

```
th = mdd_trans_create()
->do_declare_ref_del(th, target)
->dio_declare_delete(th, sourcedir, name)
->dio_declare_insert(th, parentdir, name)
mdd_trans_start(th)
mdd_pdo_write_lock(sourcedir)
mdd_pdo_write_lock(targetdir)
mdd_write_lock(source)
__mdd_index_delete(th, sourcedir, name)
__mdd_index_insert(th, targetdir, name)
mdd_write_unlock(source)
mdd_pdo_write_unlock(targetdir)
mdd_pdo_write_unlock(sourcedir)
mdd_trans_stop(th)
```

New code would look somewhat like this :

```
static int mdd_rename(const struct lu_env *env,
    struct md_object *src_pobj, struct md_object *tgt_pobj,
    const struct lu_fid *lf, const struct lu_name *lsname,
    struct md_object *tobj, const struct lu_name *ltname,
    struct md_attr *ma)
{
    ...
    mdd_txn_param_build(env, mdd, MDD_TXN_RENAME_OP);
    handle = mdd_trans_create(env, mdd, mdd_env_info(env)->mti_param);

    do_declare_ref_del(env, mdd, handle);
    rc = dio_declare_delete(env, mdd, key, handle);
    if (IS_ERR(rc)) {
        GOTO(cleanup_unlocked, rc);
    }
    rc = dio_declare_delete(env, mdd, valsize, key, handle);
    if (IS_ERR(rc)) {
        GOTO(cleanup_unlocked, rc);
    }
    rc = dio_declare_insert(env, mdd, valsize, dt_key, handle);
    if (IS_ERR(rc)) {
        GOTO(cleanup_unlocked, rc);
    }
```

```
        rc = mdd_trans_start(env);
        if (IS_ERR(rc)) {
            GOTO(cleanup_unlocked, rc);
        }
        ...
        ...
    cleanup_unlocked:
        mdd_trans_stop(env, mdd, rc, handle);
        ...
    }
```

4.5 mdd_attr_set

The basic outline of the code would be as follows :

```
th = mdd_trans_create()
->do_declare_attr_set(th, target)
mdd_trans_start(th)
mdd_write_lock(source)
mdd_attr_set_internal(source, attr)
mdd_write_unlock(source);
mdd_trans_stop(th)
```

New code would look somewhat like this :

```
static int mdd_attr_set(const struct lu_env *env, struct md_object *obj,
    const struct md_attr *ma)
{
    ...
    mdd_txn_param_build(env, mdd, MDD_TXN_ATTR_SET_OP);
    handle = mdd_trans_create(env, mdd, mdd_env_info(env)->mti_param);

    rc = do_declare_attr_set(env, mdd, handle);
    if (IS_ERR(rc)) {
        GOTO(cleanup, rc);
    }

    rc = mdd_trans_start(env);
    if (IS_ERR(rc)) {
        GOTO(cleanup, rc);
    }
    ...
    ...
cleanup:
    mdd_trans_stop(env, mdd, rc, handle);
```

```
    ...
}
```