uMDS Transaction API

Amit Sharma

17 Jan 2008

1 Introduction

With the back end moving to ZFS/DMU, the transaction model will have to change to adapt to the two phase transaction model of the DMU. The new api's have already been designed as part of an other task (osd_new_trans_api, bug id : 14311), as part of this task we just need to make sure that at all the places appropriate changes to the code are made to start using the new api's.

2 Requirements

Use the newly defined transaction api in the code by replacing the currently used api's.

3 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()
```

4 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
- mdd_ref_del
- mdd_ref_add
- mdd_close

5 Logic specification

All the functions mentioned in the use case would need modification to add additional code to make calls to the mdd_trans_create function and also to the respective declare functions to complete the transition to the new transaction apis.

We see a few example below to illustrate the changes that would be needed to achieve this.

5.1 mdd_create

```
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)
```

5.2 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)
```

5.3 mdd_link

```
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)
```

5.4 mdd_rename

```
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)
```

5.5 mdd_setattr

```
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)
```