# MDD LLOG DLD

Author Name:WangDi

Date 08/15/2006

## 1 Functional Specification

Sometimes, when mdd will access the mdd, it will first record log locally, and then send req(with the logcookie) to ost, then ost will return these cookie back to ost after then these log cookies will be destoried. According to HLD, the mdd will still use a tmp MDS OBD to access data stack (lov/osc), so those llog ctxt and llog api can be keep untouched. So this DLD only care about how to use llog api in MDD layer. Actually, there are only 2 llog API in mdd, unlink and setattr.

int mdd_unlink_log(const struct lu_context *ctxt, struct mdd_device *mdd, struct mdd_object *mdd_cobj, struct md_attr *ma);

int mdd_setattr_log(const struct lu_context *ctxt, struct mdd_device *mdd, struct mdd_object *mdd_cobj, struct md_attr *ma);

Input:

ctxt: context of this thread.

mdd: mdd device name.

mdd_cobj: mdd child object.

Out:

ma: the md attr of the object, the returned logcookie will be filled here.

Return:

> 0 means it indeed record the log. ma->la_valid should be set to indicate this ma include the cookie.

<= 0 means did not record the log.

## 2 Use Cases

### 2.1 mdd unlink/setattr

When unlink, mdd need call llog api to record unlink log.

- mdd unlink the meta object, then it will check the nlink and open count of the object.
- If both of them are zero, then mdd will call mdd log api to record the log.

- After the objects in ost are cancelled, the correspondent llog will be cancelled.

Sometimes, setattr will also need record the log, for example set uid/gid.

- mdd set uid/gid the meta object.
- it will call mdd log api to record the log.
- After the objects in ost finish changing uid/gid, the correspondent llog will be cancelled.

# 3 Logic Specification

Since mdd will use a tmp mds to access the lov, and to keep implementation simple, we still use fsfilt to access the storge for llog. So here we only need care about the llog interface in the mdd layer.

```
int mdd_unlink_log(const struct lu_context *ctxt, struct mdd_device *mdd,
                                        struct mdd_object *mdd_cobj, struct md_attr *ma)
      {
                    struct obd_device *obd = mdd2_obd(mdd);
                    if (mds_log_op_unlink(obd, NULL, ma->ma_lmm, ma->ma_lmm_size,
                                                          ma->ma_cookie, ma->ma_cookie_s
                                    ma->ma_valid |= MA_COOKIE;
                    }
                    return 0;
       }
```

For setattr log, the process is same, but just call mds_log_op_setattr.

And also to keep llog working, we also need create some objects when that tmp mds setup.

```
static int mds_cmd_setup(struct obd_device *obd, struct lustre_cfg *lcfg)
{
        ..........................
         /* Create the objects dentry, in this dir to create anonymouns llog objects */
         dentry = simple_mkdir(current->fs->pwd, "OBJECTS", 0777, 1);
         mds->mds_objects_dir = dentry;
         /*create iopen to get the llog object from the storage according to fid(ino/ge
         dentry = lookup_one_len("__iopen__", current->fs->pwd,
                                                    strlen("__iopen__"));
         mds->mds_fid_de = dentry;
          ........................
}
```

# 4 State Specification

In mds recovery, mdd will reconnect llog to sync those llog context. Because mdd use MDS obd to access the llog, so those recovery stuff of original mds can be reused and keeping untouched.