DLD for Adding MDT with Mountconf

Huang Hua

2006-03-31

1 Requirements

- Addition of MDT with mountconf in CMD
- Consideration of new layering model

2 Functional specification

Because this task is based on the current "mountconf" implementation, I will not describe what and how mountconf does in Lustre mount. I will focus on the change that is brought into Lustre due to CMD.

2.1 Adding mdt

Prototype:

```
static int mgs_write_log_mdt(struct obd_device *obd, struct fs_db *fsdb, struct mgs
```

Parameters:

struct obd_device *obd - pointer to the MGS obd;

struct fs_db *fsdb - pointer to which fsdb of this file system;

struct mgs_target_info *mti - remote MGC passed target info, also returned back to remote MGC.

Return Value:

0 - success

non-zero - failure

Description: Adding this newly registered MDT to startup log of clients and other MDTes. Also, lov & osc information and cmm & osc information will be duplicated for it.

2.2 Adding ost

Prototype:

```
static int mgs_write_log_ost(struct obd_device *obd, struct fs_db *fsdb, struct mgs
```

Parameters:

struct obd_device *obd - pointer to the MGS obd;

struct fs_db *fsdb - pointer to which fsdb;

struct mgs_target_info *mti - remote MGC passed target info, also returned back to remote MGC.

Return Value:

0 - success

non-zero - failure

Description: Adding this newly registered OST to startup log of clients and other MDTes.

2.3 Adding lmv

Prototype:

```
static int mgs_write_log_lmv(struct obd_device *obd, struct fs_db *fsdb,
                             struct mgs_target_info *mti,
                             char *logname, char *lmvname)
```

Parameters:

struct obd_device *obd - pointer to the MGS obd;

struct fs_db *fsdb - pointer to which fsdb;

struct mgs_target_info *mti - remote MGC passed target info, also returned back to remote MGC;

char *logname - log name;

char *lmvname - lmv name;

Return Value:

0 - success

non-zero - failure

Description: Adding lmv startup log of clients.

2.4 Adding cmm

Prototype:

```
static int mgs_write_log_cmm(struct obd_device *obd, struct fs_db *fsdb,
                             struct mgs_target_info *mti,
                             char *logname, char *cmmname)
```

Parameters:

struct obd_device *obd - pointer to the MGS obd;

struct fs_db *fsdb - pointer to which fsdb (mdt or ost) ;

struct mgs_target_info *mti - remote MGC passed target info, also returned back to remote MGC;

char *logname - log name;

char *cmmname - cmm name;

Return Value:

0 - success

non-zero - failure

Description: Adding cmm startup log of clients.

3 Use cases

3.1 Lustre register & startup

- MGS should be started first.
- OSTes startup. MGS add these OSTes to clients' lov startup config log. MGS also add these OSTes to MDTes' startup config log;
- MDTes startup. MGS add these MDTes to client's lmv startup config log. MGS also add these MDTes to other MDTes' cmm startup config log; Other already registered MDTes' information is duplicated to this MDT from clients'.
- Clients need no registeration. MGS adds lov & osc, lmv & mdc config log into clients' startup config log when mdt and ost registers to MGS.

3.2 MDT/OST dynamic addition

- Lustre is running; MGCes on clients/servers enqueue LDLM, waiting for configuration change;
- some MDT/OST added into system dynamically;
- MGS constructs startup config log for it; then this MDT/OST starts up by processing config logs from MGS;
- MGS revokes all MGC LDLM; MGC starts to get config from MGS;
- Clients/servers process logs from MGS;
- MDT/OST addition completes. Newly added MDT/OST is working now.

4 Logic specification

4.1 Adding MDT

This is the modified add mdt routine:

```
static int mgs_write_log_mdt(struct obd_device *obd, struct fs_db *fsdb,
                             struct mgs_target_info *mti)
{
        if (this is a newly registered MDT -- no log for it)
        {
                mgs_write_log_lov(...);  //add lov for this mdt;
                mgs_write_log_cmm(...);  //add cmm for this mdt;
        }
        duplicate other osc's information from clients' log;
        duplicate other mdc's information from clients' log;
        add this mdt to this mdt's startup config log.
        add this mdt's corresponding mdc to all other mds's cmm;
        if (clients's log is empty)
        {
                mgs_write_log_lov(...);  //add lov for this mdt;
                mgs_write_log_lmv(...);  //add lmv for this mdt;
        }
        add this mdt's corresponding mdc to clients's lmv;
}
```

4.2 Adding OST

This is the modified add ost routine:

```
static int mgs_write_log_ost(struct obd_device *obd, struct fs_db *fsdb,
                             struct mgs_target_info *mti)
{
         add this ost's startup config log for this oss;
         add this ost's corresponding osc to all mdt's lov;
         add this ost's corresponding osc to clients' lov;
}
```

4.3 Adding lmv

This is the new adding lmv routine, similar to adding lov:

```
static int mgs_write_log_lmv(struct obd_device *obd, struct fs_db *fsdb,
                             struct mgs_target_info *mti,
                             char *logname, char *lmvname)
{
        prepare lmv description data structure;
        write lmv startup config log (similar as lov);
}
```

4.4 Adding cmm

This is the new adding cmm routine, similar to adding lov:

```
static int mgs_write_log_cmm(struct obd_device *obd, struct fs_db *fsdb,
                             struct mgs_target_info *mti,
                             char *logname, char *lmvname)
{
        prepare cmm description data structure;
        write cmm startup config log (similar as lov);
}
```

5 State Management

5.1 Recovery changes

MGS should handle server's registeration atomically. That is to create and construct all kinds of config log atomically. But mountconf now has no such functionalities. We should utilize these features when they are available.

6 Focus for inspections