OST Pools

J-Ch Lafoucrière

February 27, 2008

Version $Id: OstPools-DLD.lyx,v 1.9 2008/02/26 23:08:20 jcl Exp $

```
RCS file: OstPools-DLD.lyx,v
Working file: OstPools-DLD.lyx
head: 1.9
branch:
locks: strict
access list:
symbolic names:
keyword substitution: kv
total revisions: 9; selected revisions: 9
description:
Ost Pools DLD
----------------------------
revision 1.9
date: 2008/02/26 23:08:20;  author: jcl;  state: Exp;  lines: +8 -8
Update following Nathan comments
----------------------------
revision 1.8
date: 2008/02/09 14:41:10;  author: jcl;  state: Exp;  lines: +3 -3
Add changes for liblustre
----------------------------
revision 1.7
date: 2008/02/08 13:24:39;  author: jcl;  state: Exp;  lines: +39 -4
Add cleanup
----------------------------
revision 1.6
date: 2008/02/06 23:13:23;  author: jcl;  state: Exp;  lines: +7 -7
Typo
----------------------------
revision 1.5
date: 2008/02/06 21:46:14;  author: jcl;  state: Exp;  lines: +275 -124
Add head support and follow review comments
----------------------------
revision 1.4
date: 2007/12/21 08:11:11;  author: jcl;  state: Exp;  lines: +58 -12
Add compatibility description
----------------------------
revision 1.3
date: 2007/12/04 16:11:26;  author: jcl;  state: Exp;  lines: +164 -2
Add new lov EA description
----------------------------
revision 1.2
date: 2007/11/20 14:03:05;  author: jcl;  state: Exp;  lines: +204 -69
Document formatting
```

```
----------------------------
revision 1.1
date: 2007/11/20 13:48:53;  author: jcl;  state: Exp;
Initial revision
=============================================================================
```

Contents

0.1 Global view

To implement OST Pools, we need to :

1. Add pool knowledge to Lustre configuration
   1. Add commands to lctl tools to manage pools
   2. Add pool support to MGS configuration files
3. Add Pool support to MDS
   1. Add pool knowledge to lov (add, remove, ...)
   2. Add pool use to lov
3. Add user lever pool interface
   1. Add pool options to lfs setstripe command
   2. Add pool support to lfs getstripe command
   3. Add poollist command to lfs
4. Manage compatibility with previously created files/directories and between 1.6 clients and new servers

0.2 Lctl command

The goal is to implement

```
lctl pool_new <fs name>.<pool name>
lctl pool_add <fs name>.<pool name> <ost name index list>
lctl pool_remove <fs name>.<pool name> <ost name index list>
lctl pool_destroy <fs name>.<pool name>
Where <ost name index list> is <fs name>-OST[<first>-<last>/<step>]
```

Add 4 entries to command_t cmdlist[] array.

```
/* Pool commands */
{"===  Pools ==", jt_noop, 0, "pool management"},
{"pool_new", jt_pool_cmd, 0,
 "add a new pool\n"
 "usage pool_new <fsname>.<poolname>"},
{"pool_add", jt_pool_cmd, 0,
 "add the named OSTs to the pool\n"
```

```
 "usage pool_add <fsname>.<poolname> <ostname list> ..."},
{"pool_remove", jt_pool_cmd, 0,
 "remove the named OST from the pool\n"
 "usage pool_remove <fsname>.<poolname> <ostname list> ..."},
{"pool_destroy", jt_pool_cmd, 0,
 "destroy a pool\n"
 "usage pool_destroy <fsname>.<poolname>"},
```

0.2.1 Functional Specification

In enum lcfg_command_type, adds 4 new commands:

```
LCFG_POOL_NEW       = 0x00ce020,
LCFG_POOL_ADD       = 0x00ce021,
LCFG_POOL_REM       = 0x00ce022,
LCFG_POOL_DEL       = 0x00ce023,
```

The LCFG_POOL_XXX configuration records are optional configuration records.

The command can be ignored, so an old client can mount the filesystem without error.

```
For debug: #define D_POOL D_OTHER
```

jt_pool_cmd()

```
int jt_pool_cmd(int argc, char **argv)
```

This new function checks we are on the MGS, parses the arguments and calls as many as necessary pool_cmd() with the right command (if OST name is a range, pool_cmd() is called for each OST).

get_array_idx()

```
static int get_array_idx(char *rule, char *format, int **array)
```

This new function generates an array of the indexes computed from the rule. On return, format is set to a printf format that can be used to generate the names from the index array.

pool_cmd()

```
static int pool_cmd(int mgs_device, enum lcfg_command_type cmd,
                    int argc, char **argv)
```

This new function set a struct lustre_cfg_bufs with the command and the arguments and calls l_ioctl() with the new ioctl OBD_IOC_POOL.

0.2.2 Use Cases

0.2.3 Logic Specification

jt_pool_cmd()

```
int jt_pool_cmd(int argc, char **argv)
```

This new function:

- checks lctl runs on the MGS and returns ENODEV if not.
- checks the number of arguments. If the number is 1 calls pool_cmd() with the right function depending of argv[0]. If the number is 2 or higher, checks argv[0] is pool_add, call get_array_idx() to generate the list of OST's, loop over each entry and calls pool_cmd.
- frees the allocated array.

In case of parsing error return CMD_HELP, return pool_cmd() rc otherwise.

get_array_idx()

```
static int get_array_idx(char *rule, char *format, int **array)
```

Extract the index rule (.....[first[-last[/step]]]). Compute the array size, allocate it (return -ENOMEM is failed) and fill it. Set the format from the rule. Return array size.

pool_cmd()

```
static int pool_cmd(int mgs_device, enum lcfg_command_type cmd,
                    int argc, char **argv)
```

This function:

- initializes a struct lustre_cfg_bufs lcfg
- fills buffer from args (lustre_cfg_bufs_set_string())
- sets cmd (lustre_cfg_new())
- copies data to buf (IOC_INIT(), IOC_PACK())

- calls l_ioctl(OBD_DEV_ID, OBD_IOC_POOL, buf);
- frees lcfg (lustre_cfg_free())

0.3 MGS

The goal is to write in the llog the pool cmd received and after to propagate the configuration to other lustre components.

We also have to modify llog_reader to add the new entries support.

0.3.1 Functional Specification

mgs_iocontrol()

```
int mgs_iocontrol(unsigned int cmd, struct obd_export *exp,
                  int len, void *karg, void *uarg)
```

Add support for OBD_IOC_POOL ioctl. First it extract fsname and poolname and check validity (mgs_extract_pool()). Depending of the lcfg_command command received, it calls mgs_pool_cmd(). After it revokes the lock on the filesystem configuration to start updates from everyone.

mgs_extract_fs_pool()

```
static int mgs_extract_fs_pool(char *arg, char *fsname, char *poolname)
```

It extracts the fsname and poolname from arg (<fsname>.<poolname>).

mgs_pool_cmd()

```
int mgs_pool_cmd(struct obd_device *obd, enum lcfg_command_type cmd,
                 char *fsname, char *poolname, char *ostname)
```

It checks if the filesystem exist, if not returns an error. If the ost is given, check it belongs to the filesystem. Than registers the entry in the MDT llog and in the client llog matching the filesystem.

mgs_write_log_pool()

```
static int mgs_write_log_pool(struct obd_device *obd, char *logname,
```

```
                                    struct fs_db *fsdb, char *lovname,
                                    enum lcfg_command_type cmd,
                                    char *poolname, char *fsname,
                                    char *ostname)
```

It records an entry to a llog, and add markers before and after the record.

void print_lustre_cfg(struct lustre_cfg *lcfg, int *skip)

Modified function. Add support for new LCFG_POOL records.

0.3.2 Use cases

0.3.3 Logic Specification

mgs_iocontrol()

```
int mgs_iocontrol(unsigned int cmd, struct obd_export *exp,
                  int len, void *karg, void *uarg)
```

Add IOBD_IOC_POOL support in the big switch. In the case:

- allocates a struct lustre_cfg (size from llog_data_len(data->ioc_plen1)), return -ENOMEM if failed
- fills it from uarg and extract arguments with lustre_cfg_string() calls
- calls mgs_extract_fs_pool()
- calls mgs_pool_cmd(...,LCFG_POOL_XXX, ...) function
- calls mgs_get_cfg_lock() to revoke the lock, so everyone updates.

It returns -EINVAL if args count is wrong or lcfg_command is unknown, otherwise returns mgs_pool_cmd() rc.

mgs_extract_fs_pool()

```
static int mgs_extract_fs_pool(char *arg, char *fsname, char *poolname)
```

Parse arg following the syntax <fsname>.<pool> ans sets fsname, poolname.
Return 0 if success, -EINVAL otherwise.

mgs_pool_cmd()

```
int mgs_pool_cmd(struct obd_device *obd, enum lcfg_command_type cmd,
```

- calls mgs_find_or_make_fsdb() to check if filesystem exists (if not returns -EINVAL) and to open fsdb
- parse ostname to extract the filesystem name and check it is fsname. If not returns -EINVAL
- calls mgs_write_log_pool() on log fsname-MDT0 and on log fsname-client.

mgs_write_log_pool()

```
static int mgs_write_log_pool(struct obd_device *obd, char *logname,
                              struct fs_db *fsdb, char *lovname,
                              enum lcfg_command_type cmd,
                              char *poolname, char *fsname,
                              char *ostname)
```

- calls record_start_log() to get an llog_handle
- call record_marker(CM_START)
- call record_base(..., cmd, poolname, fsname, ostname, 0)
- call record_marker(CM_END), record_end_log()

print_lustre_cfg()

```
void print_lustre_cfg(struct lustre_cfg *lcfg, int *skip)
```

Add 4 new cases in the switch on lcfg_command. Print record name and calls print_1_cfg(lcfg) for each case.

0.4 lov

Add methods to manage pool in lov. Add pool use in ost allocations. The idea is to add a per pool new struct lov_tgt_desc ** that point to struct lov_tgt_desc ** present in lov. The index in the pool array is the same as in the lov array so the use of pool is transparent to the upper layers.

0.4.1 Functional Specification

Add 4 new methods to struct obd_ops (do obd_pool_xxx functions, and modify lprocfs_alloc_obd_stats())

```
int (*o_pool_new)(struct obd_export *exp, char *poolname);
int (*o_pool_del)(struct obd_export *exp, char *poolname);
int (*o_pool_add)(struct obd_export *exp, char *poolname, char *ostname);
int (*o_pool_rem)(struct obd_export *exp, char *poolname, char *ostname);
```

Add to lustre/include/obd.h

```
struct pool_desc {
        char                    pool_name[MAXPOOLNAME+1];
        struct lov_tgt_desc **pool_tgts;
        __u32                   pool_tgt_count; /* how many OBD's */
        __u32                   pool_tgt_size;
        struct hlist_node       pool_hash;
        struct list_head        pool_list;
        cfs_proc_dir_entry_t *pool_proc_entry;
        __u32                   pool_ref_count;
};
```

Add to struct lov_obd

```
struct lustre_class_hash_body *lov_pools_hash_body; /* used for access by poolname */
struct list_head              lov_pool_list; /* used for sequential access */
cfs_proc_dir_entry_t          *lov_pool_proc_entry; /* procfs pool directory */
```

lov_pools hash table is initialized in lov_setup().

```
static struct lustre_hash_operations pool_hash_operations = {
.lustre_hashfn                          = pool_hashfn;
.lustre_hash_key_compare                = pool_hashkey_compare;
.lustre_hash_object_ref_count_get       = pool_hashref_get;
.lustre_hash_object_ref_count_put       = pool_hashref_put;
}
```

Add a to struct lov_stripe_md{} a char lw_pool_name[MAXPOOLNAME] field and add

```
#define lsm_pool_name    lsm_wire.lw_pool_name
The lsm_magic field is used to know if the lsm_pool_name field is used.
```

Add a new structure for the new lov attribute (previous one with additionnal pool name field).

```
struct lov_mds_md_v3 {            /* LOV EA mds/wire data (little-endian) */
        __u32 lmm_magic;          /* magic number = LOV_MAGIC_V3 */
        __u32 lmm_pattern;        /* LOV_PATTERN_RAID0, LOV_PATTERN_RAID1 */
        __u64 lmm_object_id;      /* LOV object ID */
        __u64 lmm_object_gr;      /* LOV object group */
        __u32 lmm_stripe_size;    /* size of stripe in bytes */
        __u32 lmm_stripe_count;   /* num stripes in use for this object */
        char  lmm_pool_name[MAXPOOLNAME];
       struct lov_ost_data_v1 lmm_objects[0]; /* per-stripe data */
};
#define LOV_MAGIC    LOV_MAGIC_V1
#define LOV_MAGIC_V1 0x0BD10BD0
#define LOV_MAGIC_V3 0x0BD30BD0
```

lov_setup()

Modified function. Add lov->lov_pools_hash_body and lov->lov_pool_list initialization and create pools directory in /proc/fs/lustre/lov/fsname-lov..../.

lov_cleanup()

Modified function. Add pool structures deallocation.

pool_hashfn()

```
static __u32 pool_hashfn(struct lustre_class_hash_body *hash_body, void *key)
```

New function. Hash the poolname, which is the key, to an integer value comprise between 0 and 127.

pool_hashkey_compare()

```
static int pool_hashkey_compare(void *key, struct hlist_node *compared_hnode)
```

New function. Compare the key to the poolname extracted from compared_hnode.

pool_hashrefcount_get()

```
static void *pool_hashrefcount_get(struct hlist_node *actual_hnode)
```

New function. Return pool extracted from compared_hnode. Increment pool_ref_count;

pool_hashrefcount_put()

```
static void pool_hashrefcount_put(struct hlist_node *actual_hnode)
```

New function. Do nothing (what is this method for ?)

class_process_config()

```
int class_process_config(struct lustre_cfg *lcfg)
```

Modified function. Add support of LCFG_POOL_XXX commands in the switch of commands who require a device. The function calls the corresponding method.

obd_pool_new(), obd_pool_del(), obd_pool_add(), obd_pool_rem()

4 new functions used to access the corresponding methods from struct obd_ops.

lov_pool_new()

```
int lov_pool_new(struct obd_device *obd, char *poolname)
```

New function. Add a pool to the lov object.

lov_pool_del()

```
int lov_pool_del(struct obd_device *obd, char *poolname)
```

New function. Remove a pool from the lov object.

lov_pool_add()

```
int lov_pool_add(struct obd_device *obd, char *poolname, char *ostname)
```

New function. Add an OST to an existing pool.

lov_pool_rem()

```
int lov_pool_rem(struct obd_device *obd, char *poolname, char *ostname)
```

New function. Remove an OST from an existing pool.

pool_fops_read()

```
static ssize_t pool_fops_read(char *page, char **start, off_t off,
                              int count, int *eof, void *data)
```

New function. Method used to return OST members of a pool through a read in /proc.

find_pool_tgts()

```
static struct lov_tgt_desc **find_pool_tgts(struct lov_obd *lov,
                                            char *poolname,
                                            unsigned *ost_count)
```

New function. Search the pool and return the struct lov_tgt_desc subarray corresponding to the pool. If the pool does no exist, return the lov struct lov_tgt_desc array.

alloc_qos()

```
static int alloc_qos(struct obd_export *exp, int *idx_arr,
                     int *stripe_cnt, char *poolname, int flags)
```

Modified function. Add a a new arg: poolname. Call find_tgts() and use the subarray in place of lov array.

alloc_rr()

```
static int alloc_rr(struct lov_obd *lov, int *idx_arr, int *stripe_cnt,
                    char *poolname, int flags)
```

Modified function. Add a a new arg: poolname. Call find_tgts() and use the subarray in place of lov array.

alloc_specific()

```
static int alloc_specific(struct lov_obd *lov,
                          struct lov_stripe_md *lsm, int *idx_arr)
```

Modified function. Call find_tgts() and use the subarray in place of lov array.

lov_dump_lmm_v3()

```
void lov_dump_lmm_v3(int level, struct lov_mds_md_v3 *lmm)
```

New function. Dump the lov_mds_md_v3 structure in the message log.

lov_packmd()

Modified function. Add support of the new LOV MAGIC version.

lov_verify_lmm()

Modified function. replace the dump v1 by a raw hex dump..

lov_alloc_memmd()

Modified function. Add support of the new LOV MAGIC version.

__lov_setstripe()

Modified function. Add support of the new LOV MAGIC version.

lsm_lmm_verify_v3()

Same as lsm_lmm_verify_v1() but use a struct lov_mds_md_v3 in place of a struct lov_mds_md_v1.

lsm_unpackmd_v3()

Same as lsm_unpackmd_v1() but use a struct lov_mds_md_v3 in place of a struct lov_mds_md_v1.

0.4.2 Use cases

0.4.3 Logic Specification

lov_setup()

At the end of lov_setup:

- calls lustre_hash_init()
- calls INIT_LIST_HEAD()
- set lov->lov_pool_count to 0
- calls lprocfs_register("pools", ...). Keep the return value in lov->lov_pool_proc_entry

lov_cleanup()

Go through the list of pools (lov->lov_pool_list) to get all pool names and call lov_pool_del() for each found pool.

pool_hashfn()

```
static __u32 pool_hashfn(struct lustre_class_hash_body *hash_body,
                         void *key)
```

The algorithm is the following :

```
result = 0;
for (i = 0 ; i < strlen(key) ; i++) {
     result = (result << 4)^(result >> 28) ^  key[i];
}
return (unsigned char)(result % 127);
```

pool_hashkey_compare()

```
static int pool_hashkey_compare(void *key,
                                struct hlist_node *compared_hnode)
```

Compare the key to the poolname extracted from compared_hnode (strncmp).

pool_hashrefcount_get()

```
static void *pool_hashrefcount_get(struct hlist_node *actual_hnode)
```

Return pool extracted from compared_hnode and increment pool_ref_count.

pool_hashrefcount_put()

```
static void pool_hashrefcount_put(struct hlist_node *actual_hnode)
```

Decrement pool_ref_count;.

class_process_config()

```
int class_process_config(struct lustre_cfg *lcfg)
```

Add support of LCFG_POOL_XXX commands in the switch of commands who require a device. Depending of the command calls the obd_pool_XXX() method. Return the rc of the method.

obd_pool_new(), obd_pool_del(), obd_pool_add(), obd_pool_rem()

```
static inline int obd_pool_new(struct obd_device *obd, char *poolname)
static inline int obd_pool_del(struct obd_device *obd, char *poolname)
static inline int obd_pool_add(struct obd_device *obd, char *poolname, char *ostname)
static inline int obd_pool_rem(struct obd_device *obd, char *poolname, char *ostname)
```

Same as other obd_XXX functions.

lov_pool_new()

```
int lov_pool_new(struct obd_device *obd, char *poolname)
```

- get lov from obd.
- call lustre_hash_get_object_by_key() to see if the pool already exists. If true return -EEXIST
- if not calls lustre_hash_additem() and list_add_tail()
- increase lov->lov_pool_count
- create the pool entry in /proc/fs/.... (call lprocfs_add_simple()) and set read method to pool_fops_read()

lov_pool_del()

```
int lov_pool_del(struct obd_device *obd, char *poolname)
```

- get lov from obd

- call lustre_hash_get_object_by_key() to get pool, return -ENOENT if not found
- decrease lov->lov_pool_count
- remove /proc/fs/... entry (remove_proc_entry(), with ifdef protection for liblustre support)
- call list_del()
- calls lustre_hash_delitem_by_key()
- free pool->pool_tgts and pool.

lov_pool_add()

```
int lov_pool_add(struct obd_device *obd, char *poolname, char *ostname)
```

- get lov from obd
- call lustre_hash_get_object_by_key() to get pool, return -ENOENT if not found
- call obd_str2uuid() to get OST uuid
- search it in lov->lov_tgts and add found entry pointer to pool->pool_tgts (extend the array if needed)
- increase pool->pool_tgt_count.

lov_pool_rem()

```
int lov_pool_rem(struct obd_device *obd, char *poolname, char *ostname)
```

- get lov from obd
- call lustre_hash_get_object_by_key() to get pool, return -ENOENT if not found
- call obd_str2uuid() to get OST uuid
- search it in pool->pool_tgts and set the entry to NULL. Return -EINVAL if not found.
- decrease pool->pool_tgt_count.

pool_fops_read()

```
static ssize_t pool_fops_read(char *page, char **start, off_t off,
                              int count, int *eof, void *data)
```

- get pool from data

- go through the pool_tgts array and fill page with ost uuid

find_pool_tgts()

```
static struct lov_tgt_desc **find_pool_tgts(struct lov_obd *lov,
                                            char *poolname,
                                            unsigned *ost_count)
```

- if poolname is an empty string, return lov->lov_tgts.
- if not call lustre_hash_get_object_by_key()
- if pool found return pool->pool_tgts, if not return lov->lov_tgts

alloc_qos()

```
static int alloc_qos(struct obd_export *exp, int *idx_arr,
                     int *stripe_cnt, char *poolname, int flags)
```

Call ost_tgts = find_pool_tgts() and replace use of lov->lov_tgts by ost_tgts when testing ost presence (is the array field NULL) and ost status (active or not).

alloc_rr()

```
static int alloc_rr(struct lov_obd *lov, int *idx_arr, int *stripe_cnt,
                    char *poolname, int flags)
```

Call ost_tgts = find_pool_tgts() and replace use of lov->lov_tgts by ost_tgts when testing ost presence (is the array field NULL) and ost status (active or not).

alloc_specific()

```
static int alloc_specific(struct lov_obd *lov,
                          struct lov_stripe_md *lsm, int *idx_arr)
```

Call ost_tgts = find_pool_tgts() and replace use of lov->lov_tgts by ost_tgts when testing ost presence (is the array field NULL) and ost status (active or not).

lov_dump_lmm_v3()

```
void lov_dump_lmm_v3(int level, struct lov_mds_md_v3 *lmm)
```

Go through the different fields and call CDEBUG() to print them.

lov_packmd()

Check LOV MAGIC version from lsm or lmmp and set lmm_size to the right value (got from lov_mds_mdv{1,3}_size()). Also set lmv fields depending of the version.

lov_verify_lmm()

Add hex dump when LOV MAGIC is unkown.

lov_alloc_memmd()

Initialized lsm_pool_name to an empty string.

__lov_setstripe()

Determine magic version and use the corresponding struct.

lsm_lmm_verify_v3()

Same as lsm_lmm_verify_v1() but replace use of v1 by v3.

lsm_unpackmd_v3()

Same as lsm_unpackmd_v1() but replace use of v1 by v3.

0.5 lfs command

The changes to the lfs command are:

- add -p <poolname> option to setstripe command
- add pool support to getstripe command
- add a poollist command

We need to change command_t cmdlist[].

Add a new lov stripe type and set it as the default:

```
#define LOV_USER_MAGIC_V3       0x0BD30BD0
#define lov_user_md lov_user_md_v3
struct lov_user_md_v3 {          /* LOV EA user data (host-endian) */
   __u32 lmm_magic;          /* magic number = LOV_USER_MAGIC_V3 */
   __u32 lmm_pattern;        /* LOV_PATTERN_RAID0, LOV_PATTERN_RAID1 */
   __u64 lmm_object_id;      /* LOV object ID */
   __u64 lmm_object_gr;      /* LOV object group */
   __u32 lmm_stripe_size;    /* size of stripe in bytes */
   __u16 lmm_stripe_count;   /* num stripes in use for this object */
   __u16 lmm_stripe_offset;  /* starting stripe offset in lmm_objects */ :q
   char  lmm_pool_name[MAXPOOLNAME]; /* pool name */
   struct lov_user_ost_data_v1 lmm_objects[0]; /* per-stripe data */
} __attribute__((packed));
#define lov_user_mds_data lov_user_mds_data_v3
struct lov_user_mds_data_v3 {
   lstat_t lmd_st;                 /* MDS stat struct */
   struct lov_user_md_v3 lmd_lmm;  /* LOV EA user data */
} __attribute__((packed));
```

0.5.1 Functional Specification

lfs_setstripe()

```
static int lfs_setstripe(int argc, char **argv)
```

Modified functions. Depending on the pool arg, call llapi_file_create() or llapi_file_create_pool() to set the stripe.

llapi_stripe_limit_check()

```
int llapi_stripe_limit_check(unsigned long stripe_size,
                             int stripe_offset, int stripe_count,
                             int stripe_pattern)
```

New function. Do the checks previously done in llapi_file_open() function.

llapi_file_open()

```
int llapi_file_open(const char *name, int flags, int mode,
                    unsigned long stripe_size, int stripe_offset,
                    int stripe_count, int stripe_pattern)
```

Modified function. Replace stripe info checks by a call to llapi_stripe_limit_check().

llapi_file_open_pool()

```
int llapi_file_open_pool(const char *name, int flags, int mode,
                         )
```

New function. Work as previous llapi_file_open() but use a struct lov_user_md * to carry stripe informations.

llapi_file_create_pool()

New function. Same as llapi_file_open() but call llapi_file_open_pool() in place of llapi_file_open_pool().

llapi_poollist()

New function. Return the list of pools or list of pool members. Use /proc/fs/lustre/lov/.... entries.

lfs_poollist()

New function. Call llapi_poollist().

ll_lov_setstripe()

Modified function to support v1 and v3 LOV_USER_MAGIC.

ll_dir_setstripe()

Modified function to support v1 and v3 LOV_USER_MAGIC.

ll_dir_getstripe()

Modified function. Add support of LOV_MAGIC_V3.

ll_dir_ioctl()

Modified function to support ioctl() with v1 and v3 LOV_USER_MAGIC.

0.5.2 Use cases

0.5.3 Logic Specification

lfs_setstripe()

```
static int lfs_setstripe(int argc, char **argv)
```

- add the new option in struct option long_opts[]
- add a new case in the switch over options
- call llapi_file_create() or lapi_file_create_pool() .

llapi_stripe_limit_check()

```
int llapi_stripe_limit_check(unsigned long stripe_size,
                             int stripe_offset, int stripe_count,
                             int stripe_pattern)
```

- do the checks previously done in llapi_file_open() function and return -EINVAL if error.

llapi_file_open()

```
int llapi_file_open(const char *name, int flags, int mode,
                    unsigned long stripe_size, int stripe_offset,
                    int stripe_count, int stripe_pattern)
```

- call llapi_stripe_limit_check()
- call to llapi_file_open()

llapi_file_open_pool()

```
int llapi_file_open_pool(const char *name, int flags, int mode,
                         unsigned long stripe_size, int stripe_offset,
                         int stripe_count, int stripe_pattern, char *pool_name)
```

- call open() with O_LOV_DELAY_CREATE flag added
- call llapi_stripe_limit_check()
- fill a struct lov_user_md_v3
- call ioctl(fd, LL_IOC_LOV_SETSTRIPE, lum) with same error checking as previous llapi_file_open().

llapi_file_create_pool()

```
int llapi_file_create_pool(const char *name, unsigned long stripe_size,
                           int stripe_offset, int stripe_count, int stripe_pattern,
                           pool_name)
```

- call llapi_file_open_pool() with O_CREAT | O_WRONLY flags

llapi_poollist()

```
int llapi_poollist(const char *name)
```

- extract fsname from name (name = fsname[.poolname])
- find lov directory from filesystem name (/proc/fs/lustre/lov/...)
- if no poolname, do readdir() to get list of pools and print the name of each pool
- if a poolname is specified, open/read the pool files to get the list of OST's and print

lfs_poollist()

```
static int lfs_poollist(int argc, char **argv)
```

- call llapi_poollist()

ll_lov_setstripe()

- copy the argument to a lov_user_md_v1
- if the magic is LOV_USER_MAGIC_V3, copy the arg to a lov_user_md_v3
- depending of the magic call ll_lov_setstripe_ea_info(), put_user() and obd_iocontrol() with the right arg (v1 or v3)

ll_dir_setstripe()

Depending of magic, call lustre_swab_lov_user_md_v1/3() and call md_setattr() with the right lum size.

ll_dir_getstripe()

Depending of magic, call lustre_swab_lov_user_md_v1/3() and lustre_swab_lov_user_md_v1/3_objects()

ll_dir_ioctl()

- copy the argument to a lov_user_md_v1
- if the magic is LOV_USER_MAGIC_V3, copy the arg to a lov_user_md_v3
- depending of the magic call ll_dir_setstripe() with the right args (v1 or v3)

0.6 Environment

The default lov EA is V1. The new lov EA (V3) is used only when explicity requested. So if pools are never used the ondisk structures stay the same as today. New EA is used only if -p option is used. If pool are used only directories default and created files have the new lov EA.

0.6.1 Binary compatibility - Old binary, New client

An old binary using the old ioctl() can call the new client.

0.6.2 Network Protocol Compatibility - New clients, Old Servers

Not supported, but if new client do not use pool options it works transparently.

0.6.3 Network Protocol Compatibility - Old clients, New Servers

To avoid unknown lmm_magic error we can add V3 knowledge to a pre-pool version but the client keeps using the V1 (same a server downgrade).

On the server side when a V1 attribute is received from a client, it used as usualy.

0.6.4 Disk Format Changes

We introduce a new lov EA, when an old EA if found, we do not change the EA format on disk.

We also modify lsm_op_find() to add support of the LOV_MAGIC_V1 and add a new struct lsm_operations for V1. The methods that have a struct lov_mds_md are specific to V1 (lsm_lmm_verify() lsm_unpackmd()), the others are the common to V1 and V3.

0.6.5 Server downgrade

An intermediary release must be used to support the new EA. When a V3 EA is found on disk, it is converted to a V1 memory format.

We can also choose to rewrite the EA to the old format.

We introduce only lov_mds_md_v3 and LOV_MAGIC_V3.

0.6.6 Documentation Changes

Add the new user command options (lctl, lfs).

0.7 Other design

The idea is to use a pool id (64 bits) to identify pools internally. This introduces the following changes :

- in the EA structure, replace the pool_name string by a _u64
- the MGS has to memorise a the last pool id in a on disk file (so a new pool is always assigned a new pool id, pool id are not reused)
- in the llog, the pool id is added to the pool_new record and the id is used in place of the name in the other records

Disadvantages:

- we have to manage a lastpoolid file per LOV
- limit the number of pool to 2^64

Advantages:

- structures are smaller
- lustre uses id internaly for every ressources, so it is homogeneous
- pool name size can be extended later without on disk and on the wire changes
- it is simpler to use a _u32 later if we had new features based on pools (e.g. policy rules to associate name file pattern to a pool)
- with poolnames, in case of reuse of the name we can have conflict with previous name use. With poolid it will note be the case (until a large number of pool)

In any case we can add a pool id in the structures and do not use it, so the structures are ready (my prefered solution).