



OST Pools

nathan.rutman@sun.com

Sun Microsystems, Inc.



Outline

- Overview
 - > What are pools
 - > Why do we need them
- Design highlights
- Using pools
- Technical details
 - > interop
 - > new code
 - > gotchas

OST Pools

- bz 14836
- written by CEA / Jacques-Charles
- http://arch.lustre.org/index.php?title=Pools_of_targets

Overview

- Motivation: delineate groups of OSTs for enhanced striping control
- Objectives
 - > define and modify arbitrary groups of OSTs
 - > use pools by name for file or directory striping
 - > use existing QOS and RR methods within a pool

Use Cases

- only local OSTs for faster access
- high-performance OSTs for premium users
- cheap JBODs for scratch files
- only InfiniBand connected OSSes
- groups owning their own storage

Pool Characteristics

- An OST can be in multiple pools
- No ordering of OSTs is implied or defined within a pool
- Membership of a pool can change over time
- As with all striping params, once objects are allocated, poolname is unused. Existing files are never restriped.
- --pool overrules --index (next index in pool is used)

Design Highlights

- Pool definitions are stored in mdt and client config llogs
- Pool usage is specified and stored along with other striping information (e.g. stripe count, stripe size) for directories or individual files.
- Non-pool case is (mostly) treated as the “pool of all OSTs”
- Unlimited numbers of pools, pool members

Pool definition

- lctl to define and modify pools and members
 - > pool_new, pool_add, pool_remove, pool_destroy, pool_list
 - > modifies config logs
 - > waits for changes to propagate
 - > beware -writeconf
 - > *lctl pool_add lustre.pool1 OST[0-10/2]*

Pool usage

- To specify a pool striping
 - > *ifs setstripe -p poolname*
- To list pools in a named filesystem:
 - > *ifs pool_list <fsname> | <pathname>*
- To list OSTs in a named pool:
 - > *ifs pool_list <fsname>.<poolname>*

Interop

- A pre-Lustre 1.8 client can access files created using a pool, but `getstripe` does not display the pool information.
- New files are not striped with pool defs.
- MDT downconverts `lov_user_md_v3->v1` for non-pools-aware clients.

Code

- Size: 53 files changed, 3302 insertions(+), 530 deletions(-)
- Almost entirely in LOV
- Plus a little lctl, lfs for the UI
- Some MGS for config llogs
- Major entry points
 - > jt_pool_cmd
 - > mgs_iocontrol_pool
 - > llapi_file_create_pool
 - > alloc_qos/alloc_rr now take pool ID

Code

- `lov_user_md_v3`
 - > `char Imm_pool_name[16]`
 - > larger ea size
- `struct ost_pool`
 - > per-pool index array into `lov_tgts`
 - > per-pool lock
 - > “all ost” pool included in `lov_obd`
- `struct pool_desc`
 - > linked list of pools
 - > name, hash, proc, etc.
 - > pool array, round-robin array

Gotchas

- pool membership can change on a live system – locking and refcounts
- v3 striping is wire protocol change
- writeconf erases pool defs (bz16825)
- Bugs mostly from:
 - > larger EA size
 - > endianness
 - > v1/v3 interop



OST Pools

nathan.rutman@sun.com