CMD Code Walk through

Wang Di

Lustre Group Sun Microsystems

Current status and plan

- CMD status
  - 2.0 MDT stack is rebuilt for CMD, but there are still some problems in current implementation.
    - No recovery support.
      - Resolution: synchronize the operation between MDT.
    - Do not support rename in CMD environment.
      - Resolution: single global rename lock.
    - Dir split problem.
      - Resolution: static directory split.
    - IAM dir format is not compatible with previous version.
      - Resolution: put FID inside dir entry.
    - Stability problem.
    - Some features are not supported in CMD, for example quota, change log.
  - Release first CMD version on 2.2.
    - Resolve first 5 problems, but not sure for the last one.

CMD Infrastructure

- In CMD directory will be split
  - Dir stripe EA.
    - Two types of dir stripe.
      - Default dir stripe
      - Dir stripe
    - Three presentations of dir stripe
      - User dir stripe
      - Memory dir stripe
      - Disk stripe dir.

CMD Infrastructure

- User level

```
struct lmv_user_md_v1 {
        __u32       lum_magic;          /* must be the first field */
        __u32       lum_stripe_count;   /* dirstripe count */
        __u32       lum_stripe_offset;  /* MDT idx for default dirstripe */
        __u32       lum_hash_type;      /* Dir stripe hash type */
        __u32       lum_type;           /* Whether it is a lmv default (children's)
                                         * stripe or it is its own dirstripe */
        __u32       lum_padding1;
        __u32       lum_padding2;
        __u32       lum_padding3;
        char        lum_pool_name[LOV_MAXPOOLNAME];
        struct  lmv_user_mds_data  lum_objects[0];
};
```

CMD Infrastructure

- Memory level

```
struct lmv_mds_md {
    __u32       lmv_magic;
    __u32       lmv_count;
    __u32       lmv_master;
    __u32       lmv_hash_type;      /*dir stripe policy */
    __u32       lmv_layout_version;
    __u32       lmv_padding1;
    __u32       lmv_padding2;
    __u32       lmv_padding3;
    char        lmv_pool_name[LOV_MAXPOOLNAME];
    struct      lu_fid      lmv_ids[0];
};
struct lmv_oinfo {
            struct lu_fid   lmo_fid;
            unsigned long   lmo_size;
             mdsno_t        lmo_mds;
        };
```

CMD Infrastructure

- Disk level

```
struct lmv_stripe_md {
        __u32       mea_magic;
        __u32       mea_count;
        __u32       mea_master;
        __u32       mea_hash_type; /*dir stripe policy */
        __u32       mea_layout_version;
        __u32       mea_default_count;
        __u32       mea_default_index;
        char        mea_pool_name[LOV_MAXPOOLNAME];
         struct lmv_oinfo mea_oinfo[0];
};
```

CMD Infrastructure

- Metadata stack (Client side)
  - llite/lmv/mdc
    - No writeback cache, simpler than data stack.
    - Dir stripe information cache
      - Directory stripe EA is cached in the similar way as file stripe.
      - Stripe EA is passed in md_op_data on the client side metadata stack.

CMD Infrastructure

- Mkdir, open/create
  - lmv_locate_mdt
    - Choose MDT to send the create request.
      - Non-stripe dir: choose the same MDT with the parent.
      - Stripe-dir: choose MDT according to name_hash value(Default is TEA).
  - lmv_alloc_fid
    - Choose MDT to allocate the fid and create the directory.
      - Choose MDT if user specified that by lfs setdirstripe.
      - Choose MDT according to default stripe info.
      - Choose MDT according to global default stripe info.
      - Choose MDT by a simple name hash for directory. (QOS goes here)
      - Choose the same MDT with its parent for file.

CMD Infrastructure

- Lookup/Getattr
  - lmv_locate_mdt
    - Choose the same MDT with the parent for non-stripe dir
    - Choose MDT according to name_hash value(Default is TEA, and hash type is stored in disk)
  - md_intent_lock
    - Get FID from the MDT and lookup lock.
  - lmv_intent_remote
    - Check whether it is a cross-ref object. If it is, go to remote MDT to get the attribute of the object and update lock.
    - The update lock will be dropped after getting the attribute.
  - lmv_revalidate_slaves
    - If it is striped_dir, it needs go to slave MDTs and retrieve the attributes and update lock from those MDTs and fill the mea_oinfo. (only need this for getattr)
    - Update lock of these slave objects are also be dropped after that.

CMD Infrastructure

- Readdir (for striped directory)
  - Dir entries is split into multiple MDT by hash_value, for example there are N servers and hash range is 0 to MAX_HASH. first server will keep records with hashes [0 ... MAX_HASH / N - 1], and second one with hashes [MAX_HASH / N ... 2 * MAX_HASH / N].
  - lmv_readpage
    - lmv_readdir_tgt
      - Locate the MDT by the offset (hash value offset)
      - Some tricky stuff here to avoid many clients do the ls -l at the same time in the same MDT.
    - md_readpage
    - lmv_hash_adjustment
      - Adjust the hash_value when the entries are finished by one MDT.
    - In md_blocking_ast, only the entries in correspondent hash_extent will be truncated.

CMD Infrastructure

- Metadata stack (server side)
  - General stack
    - MDT/CMM/MDD/OSD
    - md_object(lu_object)
      - md_operations/md_obj_operations
        - md_object_ops. (mdt/cmm/mdd)
        - md_dir_ops. (cmm/mdd)
    - On CMM layer, check whether the fid is belonged to this MDT to assign different md_dir_ops.
    - Stack parameters
      - md_op_spec (open/create parameters)
      - md_attr (attributes, lsm, lmv)

CMD Infrastructure

- Metadata stack (server side)
  - Metadata-path between MDTs
    - CMM module is used to communicating between MDT, and the partial operation is also done by the whole metadata stack.
    - Most of the remote operation will be synchronize.
      - RPC first or local operation first depend on the operation.
        - mkdir: remote RPC first, then create obj. Note: when create the remote obj, default acl/default lmv/default lov/stripe EA are both needed on the remote oject.
        - Unlink: unlink obj first, then remote RPC.
          - Get Update lock from the slave objects for check stripe dir empty.
      - Multiple slots for one mdc/mds connection.
        - Resent/replay xid check need to be changed.(need more thinking)

CMD Infrastructure

- Metadata stack (server side)
  - The RPC will be handled on the remote MDT by the whole stack.
    - The RPC will be packed with a special flag to tell whether the remote MDT whether it is RPC from another MDT for cross-ref ops.
      - MDT know this by a special flag mti-special-flag
      - CMM/MDD use special object API here.
  - Set stripe dir
    - Create slave objects(cmm_split_create_slave) .Set_attr of the stripe EA
    - Check default stripe EA in cml_create/cml_object_create

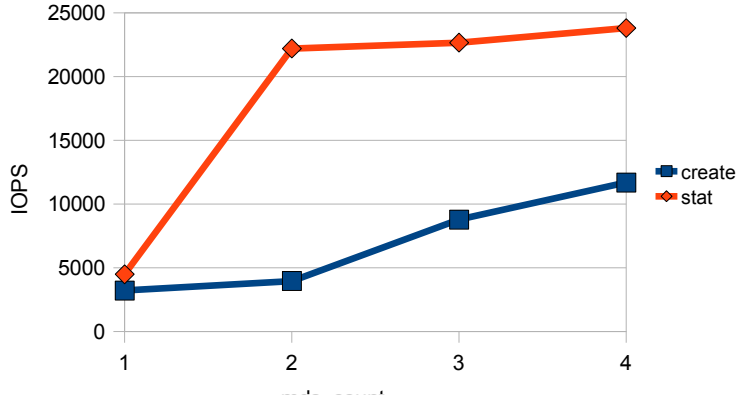
CMD Infrastructure

- Metadata stack (server side)
  - Lock issue
    - update/lookup lock are controlled by different MDTs.
      - Lookup lock is controlled by Master MDT (entry MDT).
      - Update lock is controlled by remote MDT.
        - Some attributes(default stripe EA and permission) changing also needs to revoke the lookup lock.
        - It can also check the update lock during lookup process.
      - MDT will remove the update lock if it found the object is cross-ref object.

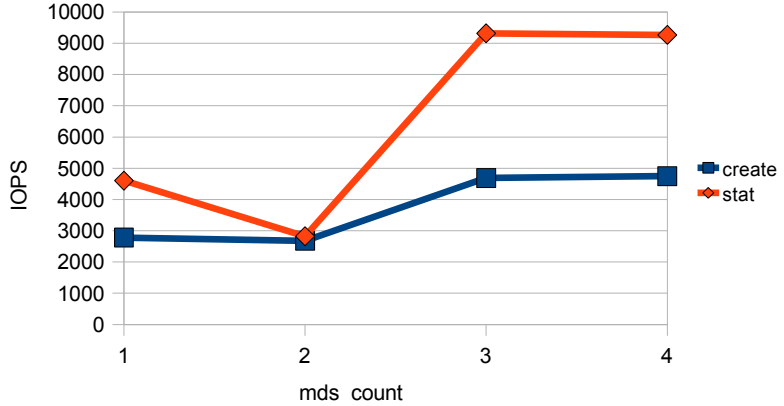
CMD Performance result

- Performance (8 clients 2 threads/per client, 4 OST/MDT)

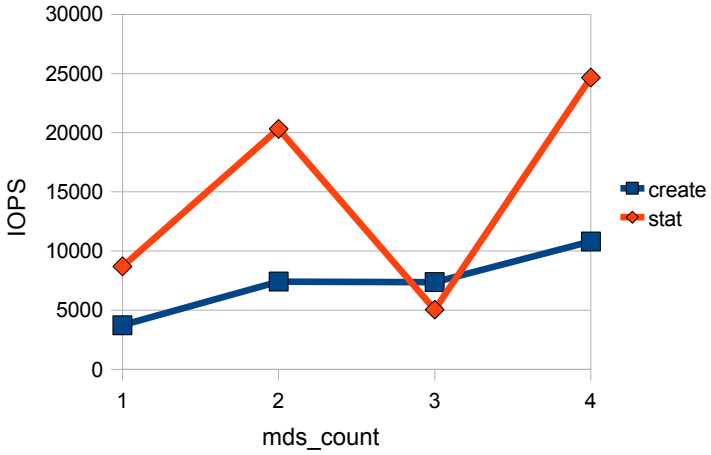
Metabench: different directory (w/o setdirstripe)



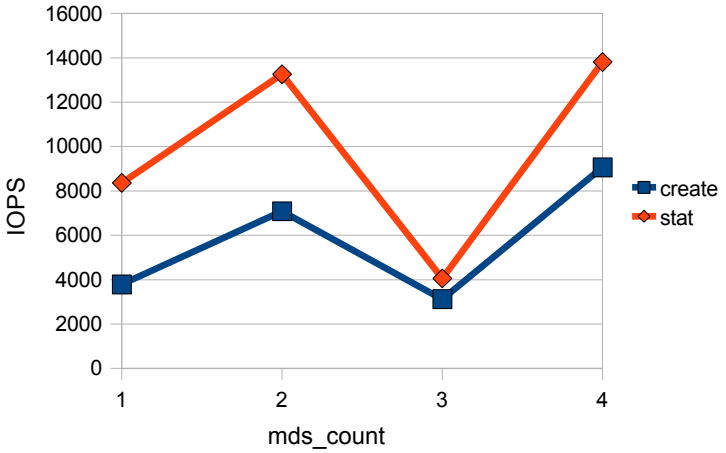
metabench: Same directory (w/o setdirstripe)



Metabench: different directory (setdirstripe)



Metabench: (same directory) (setdirstripe)



Some other issues

- Some other issues
  - CMD and single MDT compatibility
    - Multi-slot entires in last_rcvd.
  - Cross-ref lock use case still need further investigate
  - Group number will be replaced by FID sequence
  - Rename
    - Rename lock will be a global lock
    - Error handler for rename
  - Performance of IAM
  - How dirstripe information will be fit into the MDS stack(alex proposed), above MDD or below MDD?
  - MDT pools (Group MDTs in different locations)
  - Restructure based on lu_dev (CLMD)

THANK YOU

Wang Di
Lustre Group Sun Microsystems