Coordinatool

A copytool to rule them all

Dominique Martinet

dominique.martinet@codewreck.org

September 28-30, 2021

Thanks

Big thanks to CEA for organizing LAD and funding this development and talk

(And also to everyone involved in the lustre community for making this possible)

1. Introduction
   - What?
   - Why?
2. Internals
3. Evolutions
4. Demo!

What?

- A lustre copytool…
- … that acts as a userspace coordinator…
- … and gives work to real, existing copytools through LD_PRELOAD overloading

Overview

Two main components:

- Lustre copytool “server”:
  - accept requests from lustre
  - schedules requests to coordinatool clients
- LD_PRELOAD client library
  - overloads llapi HSM calls
  - allows using existing copytools as coordinatool clients
- administrative client
  - query status, lock states, requeue lost requests. . .

Still WIP (young “free time” project), but already works

Why?

- Reaching the limits of in-kernel coordinator
  - Set limits for each request type per agent
    - CEA agents allow more remove than the rest: lustre keeps banging its head on EAGAINs
  - Better request scheduling, retries…
- Stepping stone for the real userspace coordinator work (LU-13384)

1. Introduction
2. Internals
   - Lustre copytool “server”
   - Client protocol
   - LD_PRELOAD client
   - coordinatool_client
   - Tricky bits
3. Evolutions
4. Demo!

Server code flow

Single thread process:

- register llapi copytool
- bind/listen/accept TCP connections from clients
- epoll loop on `llapi_hsm_copytool_get_fd`/clients
  - Receive & process HSM requests
  - Reply to client requests
  - Hopefully won't block. . .

Server in depth

- What do we use to queue actions ?
  - performance bottleneck with catalogs on real coordinator
  - . . . might as well use it for recovery: work in memory

Quick reminder of lustre types: HAI

```
struct hsm_action_item {
  __u32      hai_len;    /* valid size of this struct */
  __u32      hai_action; /* hsm_copytool_action */
  struct lu_fid hai_fid; /* Lustre FID to operate on */
  struct lu_fid hai_dfid; /* fid used for data access */
  struct hsm_extent hai_extent;
                          /* byte range to operate on */
  __u64      hai_cookie;  /* action cookie */
  __u64      hai_gid;     /* grouplock id */
  char       hai_data[0]; /* variable length */
} __attribute__((packed));
```

Quick reminder of lustre types: HAL

```
struct hsm_action_list {
  __u32 hal_version;
  __u32 hal_count;       /* number of hais to follow */
  __u64 hal_compound_id; /* ignored */
  __u64 hal_flags;
  __u32 hal_archive_id; /* which archive backend */
  __u32 padding1;
  char  hal_fsname[0];   /* null-terminated */
  /* struct hsm_action_item[hal_count] follows, aligned
     on 8-byte boundaries. See hai_zero */
}
```

Queueing actions: requirements 1/2

- We need to send `hsm_action_list`s to clients
  - queues must group by identical hal flags/archive_id
  - for now (v0), single queue only checking identity on append
  - code is structured to allow multiple queues:
    - `hsm_action_queues_get(internal_state, archive_id, flags, fsname)`

Queueing actions: requirements 2/2

- We need to quickly find a request by cookie if required
  - for `HSMA_CANCEL` (TODO)
    - easy if not sent yet to a client
    - find which client if there is one
  - deduplication for recovery mechanism

Queueing actions: implementation

- liburcu’s double-linked list for queues
  - One queue for each action type (restore, archive, remove)
  - if sent, list head reused for in a list per client
- libc’s hashtable for lookup by cookie
  - also one per archive_id/flags. . .
- each action in memory also contains:
  - pointer to current client handling it if any
  - pointer to queues (archive_id/flags, requeue)
  - a copy of the `struct hsm_action_item`

Client protocol

- simple json serialization
- server only replies to clients
  - possibly not right away (e.g. no work to distribute)
- code is shared as much as possible
  - both clients and server
  - a new client would only need to implement a few callbacks

Commands:

- STATUS: query server info
  - pending, running and processed action counts
  - number of clients connected. . .
- RECV: request work.
  - Specify how many restore/archive/remove the client can process.
- DONE: report to the server that an action was processed
- QUEUE: push an `hsm_action_list` (more later)

LD_PRELOAD client

- Overload llapi hsm functions

LD_PRELOAD client: overriden functions 1/2

- `llapi_hsm_copytool_register`:
  - connect to server, open lustre root and .lustre/fid
  - alloc `hsm_copytool_private` with non-lustre MAGIC
- `llapi_hsm_copytool_unregister`: cleanup
- `llapi_hsm_copytool_recv`: send RECV and wait

LD_PRELOAD client: overriden functions 2/2

- `llapi_hsm_action_begin`
  - call real `llapi_hsm_action_begin`
  - extend `copyaction_private` with some fields
- `llapi_hsm_action_end`
  - call real `llapi_hsm_action_end`
  - send DONE
- `llapi_hsm_copytool_get_fd`
  - just return TCP fd (TODO)

That's it!

LD_PRELOAD client: why it works

- calling real `llapi_hsm_action_begin` works on different client
  - Lustre does not care if a client different from the one which received request processes it
  - Don't need to do it on coordinatool/send temporary file fid...
- similarily, progress and other calls mostly work
- borderline bug: `llapi_hsm_action_begin` does not check copytool_private MAGIC
  - need `mnt_fd` and `open_by_fid_fd` at correct offset

coordinatool_client

- Intended for administrative tasks (stats. . .)
- Or debugging/tests
  - Request work and dump it on stdout
  - No done: server requeues work on client disconnect
- Very simple and unpolished (~250 lines of code)
- Mostly just reuse common init and protocol code

Tricky bits

- Server restarting
- Client disconnecting
- Lustre tunings

Server restarting

- Action queue only in memory
- lustre doesn’t handle a copytool disappearing really well
  - actions that had been sent are never resent
- mdt.fsname-MDT0xyz.hsm.active_requests to the rescue
  - Parse the file and send it with “queue”
- Last problem: actions currently being handled by clients
  - (TODO) client writes in filesystem on action_begin
  - read files on startup, give related actions a grace period

Client disconnecting

- currently requeues its processing actions immediately
- (TODO) give a grace period to reconnect ?
  - reclaim running actions on connect
  - trust service manager to not have duplicate on same host
  - later.

Lustre tunings

- don’t let requests expire
- send requests to coordinatool ASAP

```
lctl set_param -P
   mdt.lustre0-MDT*.hsm.active_request_timeout=31days
   mdt.lustre0-MDT*.hsm.loop_period=1
   mdt.lustre0-MDT*.hsm.max_requests=1000
```

1 Introduction

2 Internals

3 Evolutions

- What next?

4 Demo!

What next?

- Finish TODOs listed earlier
  - server restart and HSMA_CANCEL
- Waa-ay more tests
- Improve scheduling:
  - Got basic restore > rest like coordinator
  - Check file owners and don't let a user hog all agents
  - group requests by locality on tape?
- More commands? dump requests, lock/unlock...
- Lustre userspace coordinator API support
- The sky's the limit!

1 Introduction

2 Internals

3 Evolutions

4 Demo!

Thanks!

https://github.com/martinetd/coordinatool/

Tests, issues or PR welcome :D

Taking questions on the chat!