
LUG 2019 Developer
Summit
nrutman@cray.com
CRAY

Fast ‘ls –l’

Get all the metadata for every file in this directory

- ls –l
- Slurp MD into external DB (every policy engine in the world)
- Generally want layouts also
- In almost all use cases lazy stats are fine
  - Polling for size/mtime would be a single file, not a full dir

Problems

- Non-lazy size, mtime, atime
- Serialized operations

```
opendir
readdir
        foreach file in dir
                stat path
                list xattr
                        foreach xattr
                                getxattr
closedir
```

What can be done?

- In the past: POSIX. Statahead. Keep syscall pipeline.
- Continuing down that path:
  - Readdir+? xgetdents?
  - Pre-fetch all getdents()+stat()[+getxattr()] into the client cache
- Fuck that. Let's just optimize our 1 use case.
  - Special ioctl(dir, flags) to read all the dirents, all the stats, and (optionally) all the xattrs, and return a big bulk blast from 1 RPC
  - Need to page through a bulk buffer - ioctl(GETNEXTPAGE)? read a virtual file?
  - In parallel to all DNE dir shards (don't care about ordering)
  - Write a "lfs ls". PEs can call the ioctl directly.

Lustre for Long-Term Storage

Feasibility and Interest

Lustre design point

- Current:
  - Custom, unusual hardware $$$
  - Focus on performance use cases

- Future?
  - Fastest tiers will NOT be Lustre (pmem)
  - Still need spinning / streaming bulk transport
  - Even larger scales – but without top performance requirements
  - Or cede the market to Ceph…

Lustre for bulk: hardware

- Disks / SSDs getting larger
  - RAID sets are too big
    - Too costly for entry point
    - Too slow to rebuild
    - Too large for ldiskfs
- Dual-ported disks are restrictive
  - Limited production $$
  - HA PITA
- Custom enclosures
  - $$
  - Long development cycles

Requirements

- No dependence on resilient HW
  - Lustre live mirror/EC
  - Continuous availability (not blocked by failover)
- Tunable (per file) availability / durability
- More, smaller OSTs (increase OST count limits)
- Fail-out instead of failover – dynamically add and remove OSTs

Possible Features

- Lustre live mirroring
- Lustre EC
  - Verify on read, DI checks
- CRUSH layouts for fail-out?
- Algorithmic or bitmap layouts to specify larger OST lists
- Larger configs
- File version access, snapshots with ZFS
- Encryption at rest
- API for data management
  - migrate, mirror, retire, replace
- Relaxed POSIX – just an archive (but fast, unlike cloud)
- New allocators – fullest-first + spindown
- Break namespace function from layout / accounting / quotas