Lustre Locking overview

Oleg Drokin

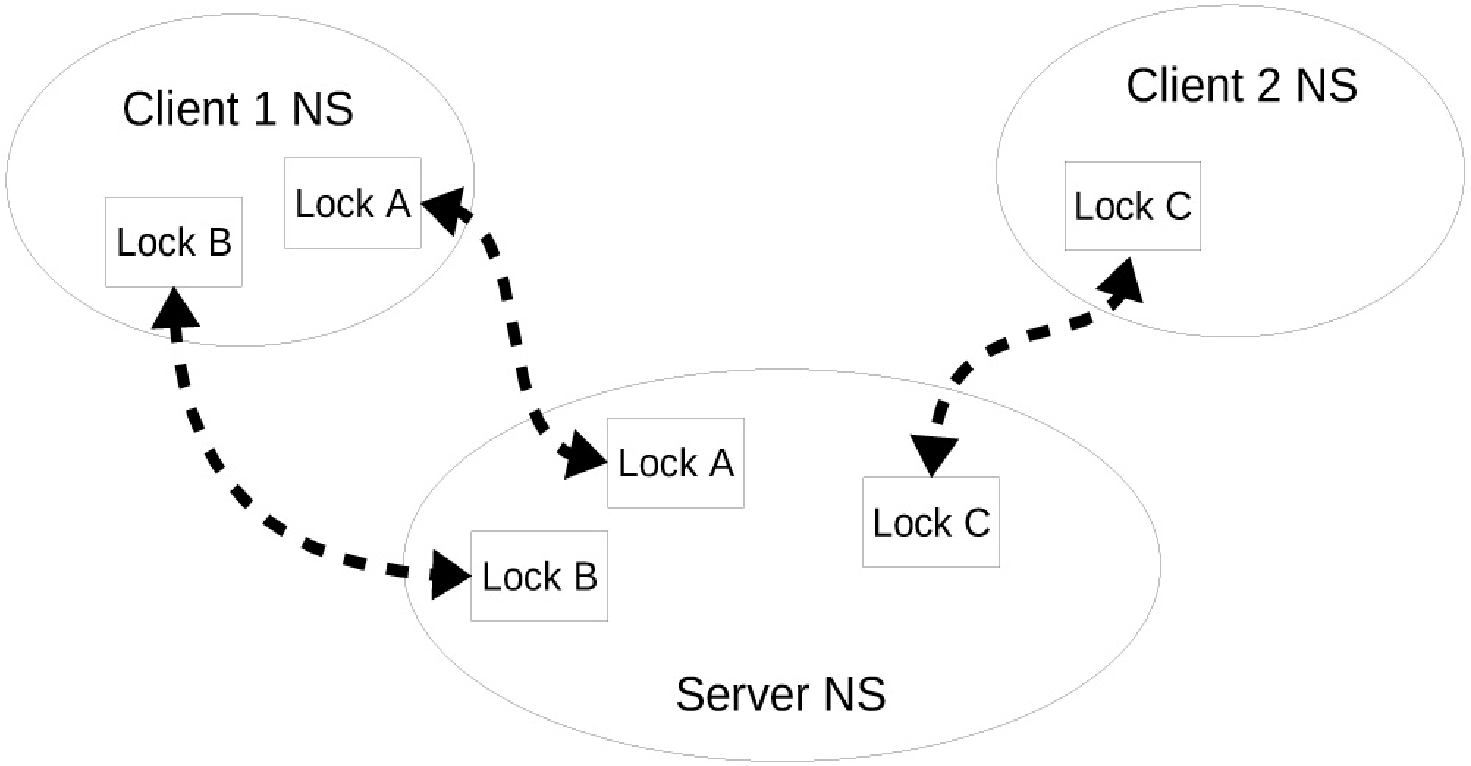
July 24, 2017

Lustre DLM from 10,000 ft

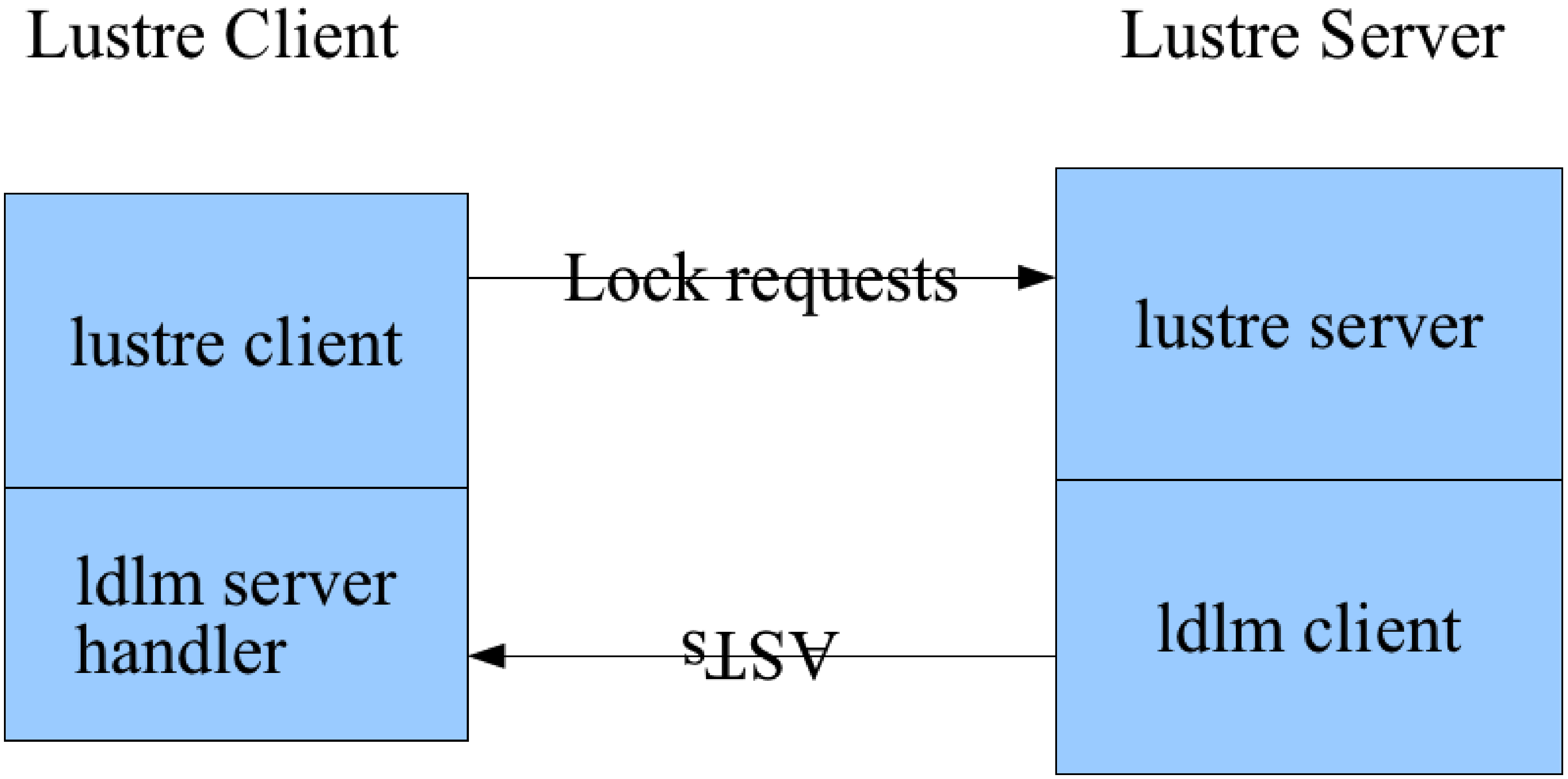
- Based on ideas from VMS distributed lock manager
  - Hence some confusing names like AST
- Every server has a namespace for objects it holds
  - Based on server type can be data or metadata
- Every server is the authority about its own namespace
  - No quorums.

Lustre DLM from 10,000 ft, cont'd

- Clients have limited visibility into server namespace
  - Only locks they have been granted

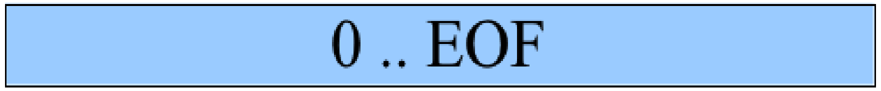


Lustre DLM from 10,000 ft, cont'd

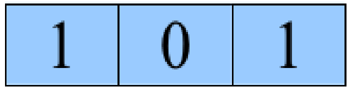


Lock modes, types

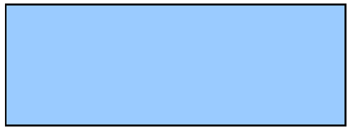
Extent – Data ranges



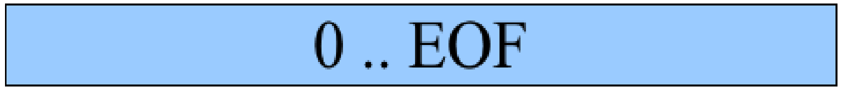
Inodebits - Metadata



Plain



Flock – data ranges



Why do we need the locks?

- Concurrency control
  - This is obvious
- Cache control
  - While a client holds some lock, corresponding object cannot change or cannot be touched at all.
  - This is how a lot of POSIX compliance is done while having client-side write cache

Special glimpse AST

- Write cache and file sizes don’t mix easily.
- Write from job nodes while another one has impatient user doing ls –l watching the size grow problem
  - We certainly don’t want to be flushing all dirty pages for this.
- Solution: Glimpse AST to ask the client “hey, what’s the highest offset in this file”
- Server only sends this message to the highest offset lock holder

Lock lifecycle on a client

- Ask for lock due to some operation being performed
- Server eventually grants the lock
- Client performs the operation it wanted the lock for
- Client retains the “unused” lock in local LRU
  - Next time we need this same lock, can just get it there
- Eventually lock is too stale and returned to the server
- Or there’s a conflict because another client wants to touch same object
  - Client receives a “blocking AST” and releases the lock.
  - Actual lock release is called lock cancel in Lustre.

Client lock LRU

- Used to be 100*NUM_CPUS per client namespace by default
  - ldlm.$NAMESPACE.lru_size control
- Setting that to 0 (new default) enables “lru resize”
  - Client caches as many locks as it could, unless told by server not to.
  - This tends to use a lot of memory on servers sometimes starving caches – so something to look for.
  - Old locks “= older than 65 minutes” (used to be 10 hours) are automatically canceled

Client lock LRU

- Benefits:
  - Much faster to get a locally cached lock
- Drawbacks:
  - Much slower for a different client to get a conflicting lock due to all the RPCs.
  - More locks cached = more memory used
- Helps a lot on login nodes
- Computes between jobs may not benefit from stale LRUs
  - More people now opt to clear lock LRUs (and pagecache) between jobs

Useful server memory tunings

- Starting from 2.8.0 release you can set limits on ldlm memmory use on servers
  - ldlm.lock_limit_mb (in megabytes) – hard limit
    - Default 30Mb
  - ldlm.lock_reclaim_threshold_mb – start to ask clients to release locks.
    - Default 20Mb
- If you have a lot of RAM, it makes sense to increase these values

Blocked lock rpc flow

- Server sends Blocking AST
  - Waits for client reply for ~7 seconds. Nowadays also retries
  - If no confirmation -> client is evicted
- Once the confirmation is received lock is placed onto the waiting list
  - Client is expected to finish IO and cancel the lock in reasonable time
  - Every IO request under this lock prolongs the lock timeout
  - If timeout expires client is evicted.

Commonly seen errors

```
LustreError: 12408:0:(ldlm_lockd.c:687:ldlm_handle_ast_error()) ### client (nid 0@lo) failed to reply to
blocking AST (req@ffff880051aa6520 x1551685286400384 status 0 rc -5), evict it ns: mdt-lustre-
MDT0000_UUID
```

- Failure to reply to AST
  - Client dead or network partition most likely

```
LustreError: 0:0:(ldlm_lockd.c:358:waiting_locks_callback()) ### lock callback timer expired after 101s: evicting
client at 149.165.238.1@tcp ns: mdt-ffff881837d4e000 lock: ffff881d6af0c000/0x78c70fdb970d7e0 lrc: 3/0,0
mode: PR/PR res: 8589943942/77226 bits 0x3 rrc: 13 type: IBT flags: 0x4000020 remote: 0xe213f1ffc604946c
expref: 54330 pid: 11173 timeout: 4825691675
```

- Failure to cancel lock in time
- But why did it fail? Many possible reasons
  - Network slowdowns, packet loss, client busy or dead, …

Commonly seen errors

```
LustreError: 12408:0:(ldlm_lockd.c:687:ldlm_handle_ast_error()) ### client (nid 0@lo) failed to reply to
blocking AST (req@ffff880051aa6520 x1551685286400384 status 0 rc -5), evict it ns: mdt-lustre-
MDT0000_UUID
```

- Failure to reply to AST
  - Client dead or network partition most likely

```
LustreError: 0:0:(ldlm_lockd.c:358:waiting_locks_callback()) ### lock callback timer expired after 101s: evicting
client at 149.165.238.1@tcp ns: mdt-ffff881837d4e000 lock: ffff881d6af0c000/0x78c70fdb970d7e0 lrc: 3/0,0
mode: PR/PR res: 8589943942/77226 bits 0x3 rrc: 13 type: IBT flags: 0x4000020 remote: 0xe213f1ffc604946c
expref: 54330 pid: 11173 timeout: 4825691675
```

- Failure to cancel lock in time

```
LustreError: 20011:0:(ldlm_lockd.c:2074:ldlm_cancel_handler()) ldlm_cancel from 149.165.238.1@tcp arrived
at 1394488331 with bad export cookie 543933852487261410
```

- A clear sign there was some network or ingestion delay that prevented this lock from reaching server in time.

Commonly seen errors 2

```
LustreError: 12408:0:(ldlm_lockd.c:687:ldlm_handle_ast_error()) ### client (nid 0@lo) failed to reply to
blocking AST (req@ffff880051aa6520 x1551685286400384 status 0 rc -5), evict it ns: mdt-lustre-
MDT0000_UUID
```

- Failure to reply to AST
  - Client dead or network partition most likely

```
LustreError: 0:0:(ldlm_lockd.c:358:waiting_locks_callback()) ### lock callback timer expired after 101s: evicting
client at 149.165.238.1@tcp ns: mdt-ffff881837d4e000 lock: ffff881d6af0c000/0x78c70fdb970d7e0 lrc: 3/0,0
mode: PR/PR res: 8589943942/77226 bits 0x3 rrc: 13 type: IBT flags: 0x4000020 remote: 0xe213f1ffc604946c
expref: 54330 pid: 11173 timeout: 4825691675
```

- Failure to cancel lock in time
- But why did it fail? Many possible reasons
  - Network slowdowns, packet loss, client busy or dead, …

Questions?

Questions?