Lustre Resiliency:

Dealing with Message Loss

Chris Horn, Cray Inc.

hornc@cray.com

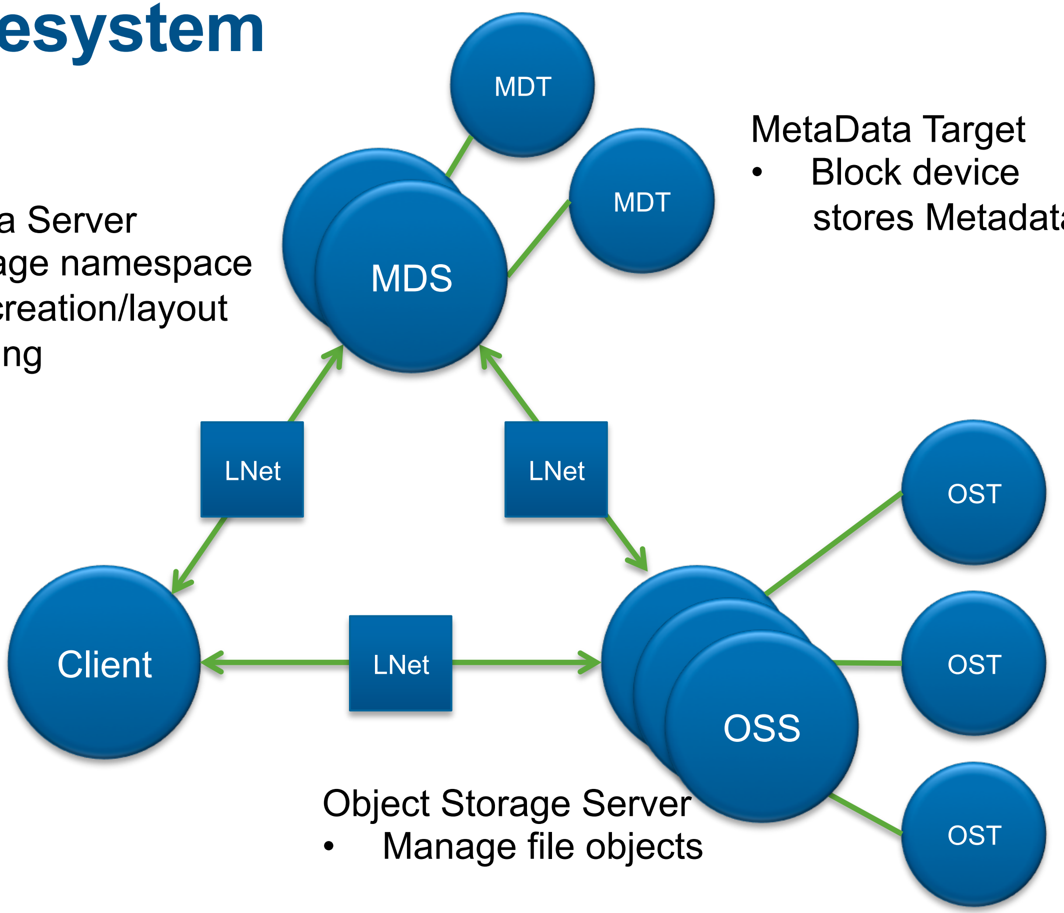
Safe Harbor Statement

This presentation may contain forward-looking statements that are based on our current expectations. Forward looking statements may include statements about our financial guidance and expected operating results, our opportunities and future potential, our product development and new product introduction plans, our ability to expand and penetrate our addressable markets and other statements that are not historical facts. These statements are only predictions and actual results may materially vary from those projected. Please refer to Cray's documents filed with the SEC from time to time concerning factors that could affect the Company and these forward-looking statements.

Agenda

- **Background Information**
  - Lustre Basics
  - The problem with dropped messages
  - Resiliency features, Lustre Locking, Lustre Evictions
- **Resiliency Enhancements**
- **Tuning Lustre for Resiliency**
- **Site-specific Tuning**
- **Future Work**
- **Questions**

A Lustre Filesystem



MetaData Server
- Manage namespace
- File creation/layout
- Locking

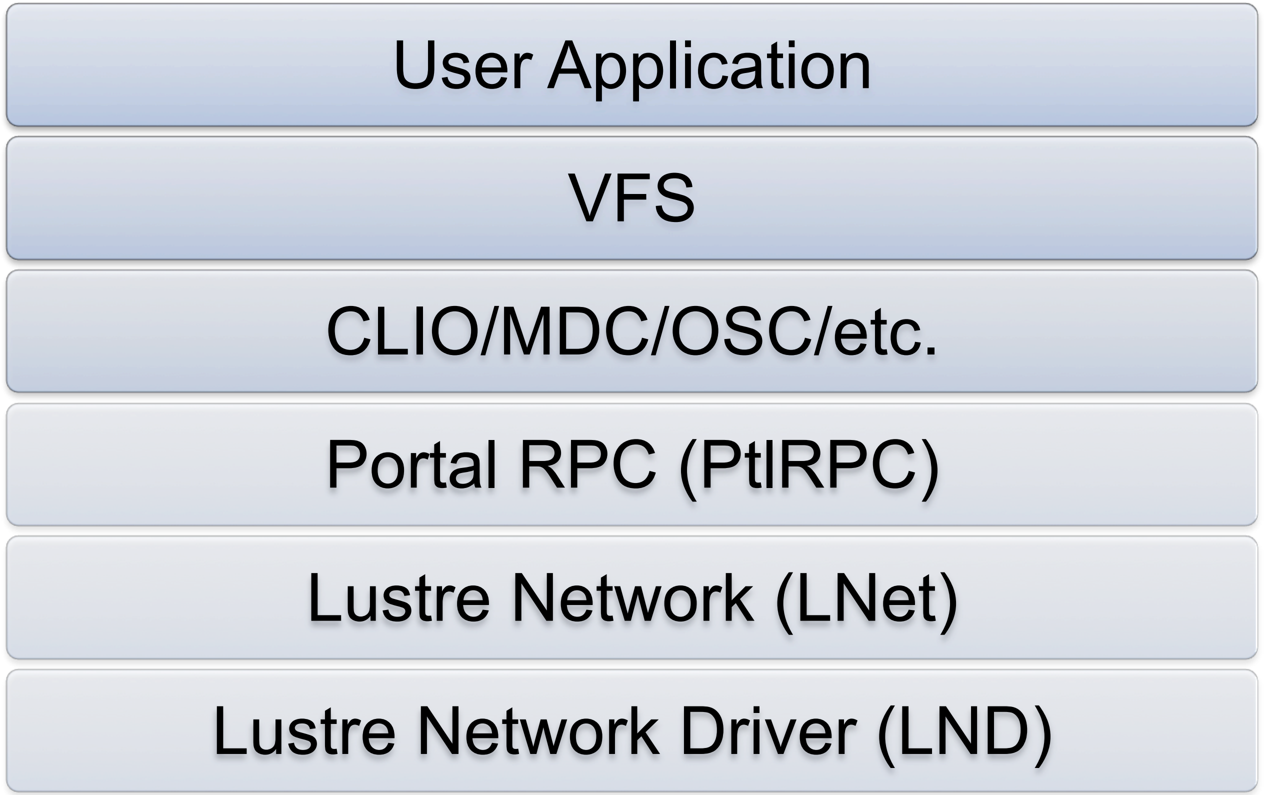
MetaData Target
- Block device stores Metadata

Object Storage Target
- Block device stores file objects

Object Storage Server
- Manage file objects

COMPUTE | STORE | ANALYZE

(Some of) The Software Stack



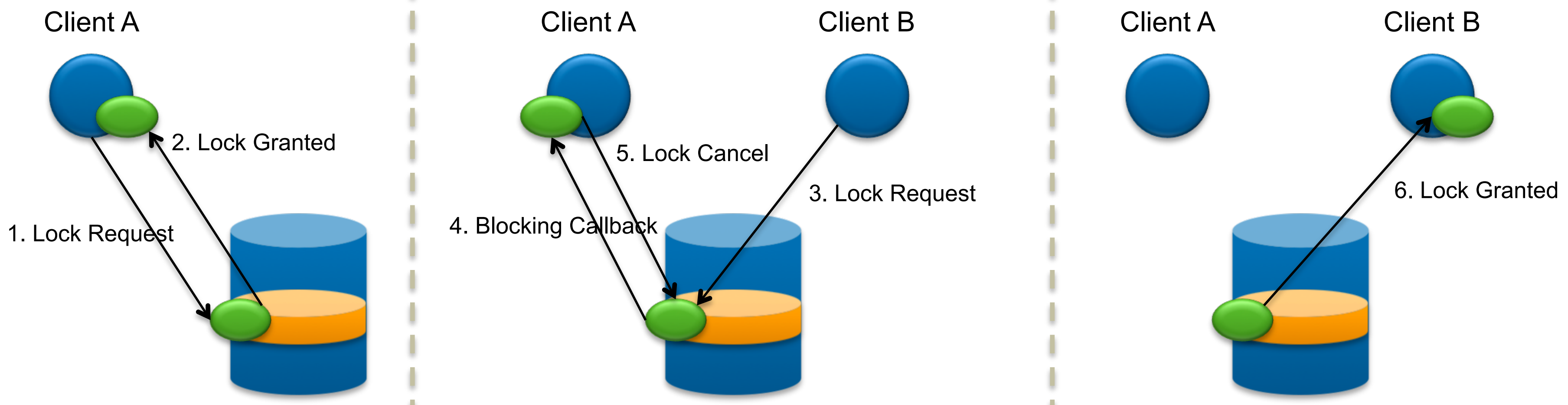
The Problem

- **What: Dropped messages result in performance degradation or application failure**
- **Why:**
  - Lost messages must be resent
    - RPC Timeouts (Fast or several minutes depending on load)
      - Adaptive Timeouts: Network latency + service estimate
    - Avoiding Bad Routes (Minutes)
      - LNet router pinger and asymmetric route failure detection
    - The Connect RPC (Minutes)
      - Sent on an interval
  - Important lock related message was not resent
    - Single point of failure in Lustre protocol
      - bugzilla.lustre.org bug 3622 opened June 2004
- **How: Client, Server, Router crash; Link Failure, etc.**
- **Our Goal: Survive finite network disruption without eviction and minimize performance impact**

Understanding Lustre Locking

- **A lock protects a resource (inode, file, etc.)**
- **MDS provides striping information (open())**
- **Client enqueues lock for each stripe to the respective OST**
- **Server revokes conflicting locks with blocking callback request**
  - Response to blocking RPC must, eventually, be a lock cancel
- **Completion RPC sent to client grants lock**
  - Client must acknowledge receipt of completion RPC

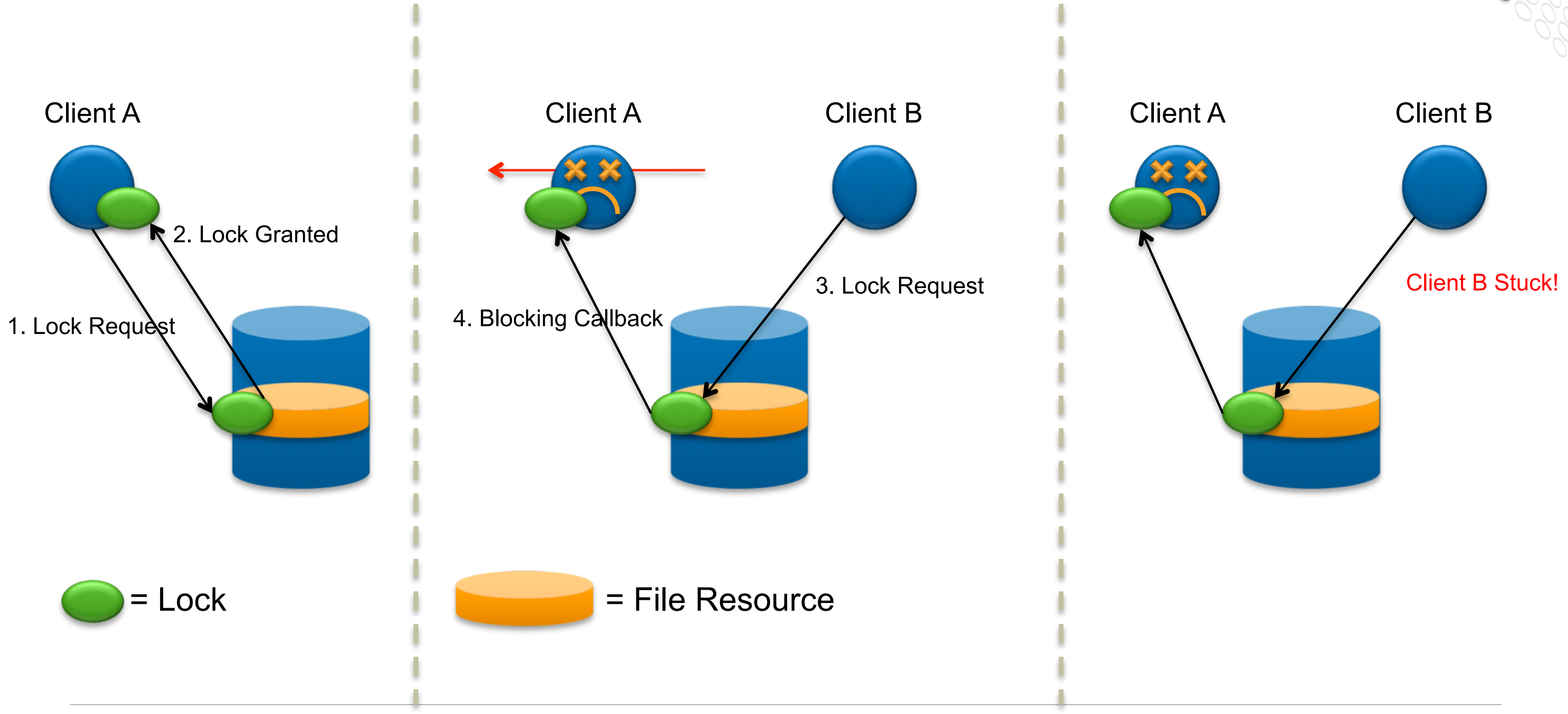
Locking in Lustre



● = Lock

▬ = File Resource

Client Crash While Holding Lock



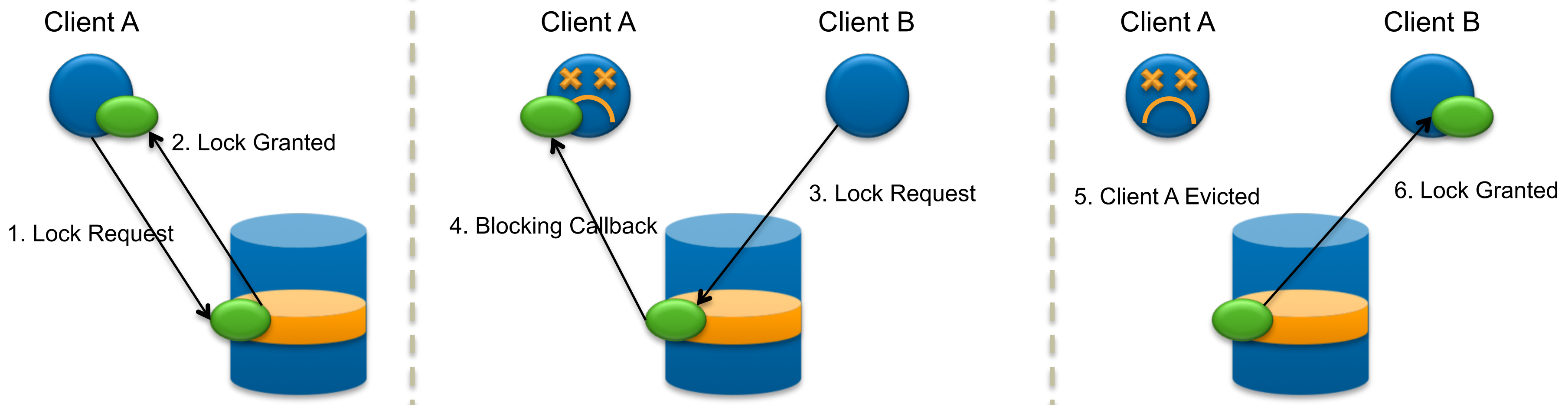
Understanding Lustre Evictions

- **What: Server's way of reclaiming resources held by a client and preventing further participation in file system operations**
- **Why: Server perceives client as misbehaving**
  - Failure to respond to certain requests or communicate regularly
  - Client side bugs, kernel panic or OOPs, heavy load, LBUGs, network errors
- **How: Server revokes all locks held by client**
  - Invalidates client's cached inodes and dirty pages
- **Things to know:**
  - Servers decide for themselves whether and when to evict a client
  - Clients don't learn that they are evicted until they reconnect to server
  - Clients then drop all locks and all dirty pages
  - Clients typically return -EIO (-5) up to user from syscalls
  - User programs typically exit on -EIO, but check your return codes!
  - Clients could be unaware if no outstanding user request (buffered I/O)
  - This is just POSIX semantics; check return codes and/or use fsync(2)
  - Unaware users call this silent data corruption

What is the Lock Callback Timer?

- **Lock Callback Timer: Client must fulfill callback request before the timer expires or the client will be evicted**
  - Blocking callbacks are subject to a lock callback timer
    - A blocking callback can be embedded in a completion callback request
  - Started on the server when callback is sent
  - Extended when client performs I/O under the lock

Eviction to Reclaim Lock



🟢 = Lock

🟧 = File Resource

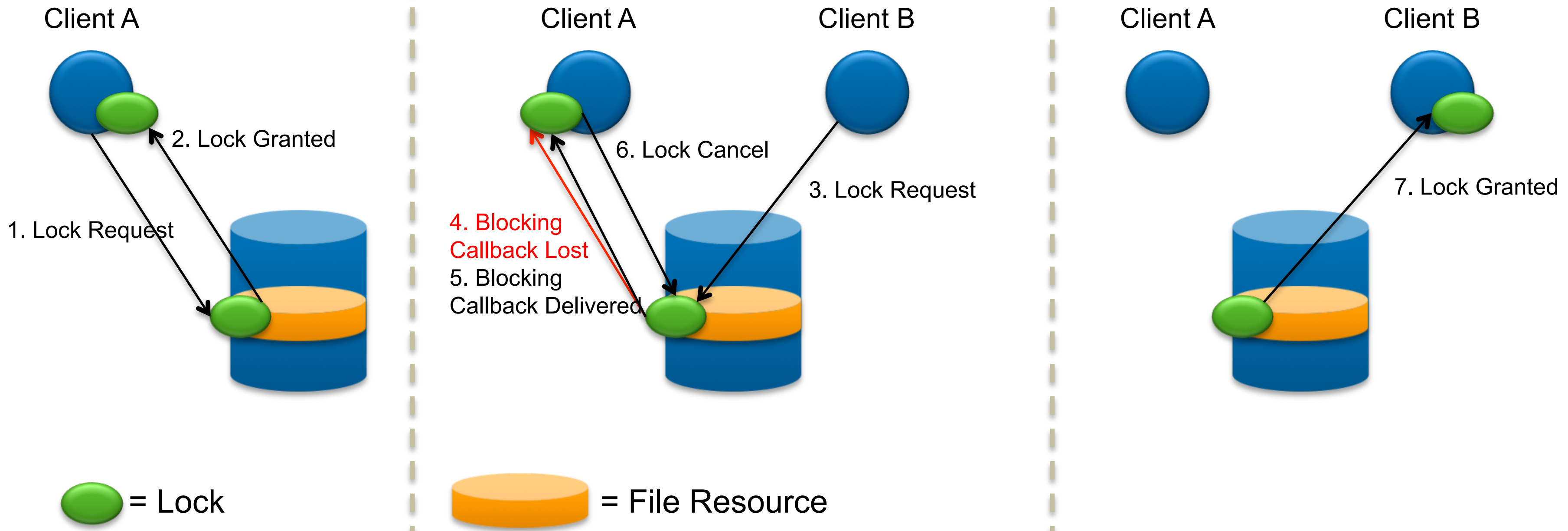
What Scenarios Result in Message Loss

- **Route and Router Death**
  - Rely on router pinger and asymmetric route failure detection
- **Client Death**
  - Dead clients evicted by lock callback timer or ping evictor
- **Server Death**
  - Lustre recovery should ensure filesystem returns to useable state
    - See Lustre Operations Manual
- **Link Resiliency**
  - HSN quiesce causes headaches for Lustre
  - Servers have no knowledge of quiesce

Enhancement: Lock Callback Resend

- **Reduces occurrence of evictions by resending lock callbacks**
- **Lock callbacks are resent over the duration of the lock callback timer**
  - RPC timeout: Lock callback timer expired?
    - Yes: Evict client
    - No: Resend RPC
- **Feature landed for Lustre 2.7.0**
  - CLE 5.2 clients mounting Sonexion w/NEO 1.3.1SU14

Resend of Callback Avoids Eviction



Enhancement: Router Failure Detection

- **Decreases time to detect failed routers from ~110 seconds to ~30 seconds**
- **Cray's gnilnd subscribes to node failure events available on the HSN**
- **Used for LNet peer health feature on routers**
  - Routers immediately drop messages being sent to down peers
- **Clients now use this for router health**
  - LNet on a client is notified of down routers
  - Down routers are not used as next-hop on future sends
  - Faster than relying on router ping

Enhancement: Gnilnd Fast Reconnect

- **Quickly restore client <-> router connections following link resiliency event**
- **Historically, gnilnd only established connections when there was an outstanding transmit**
  - Connection between clients and routers often timeout during HSN quiesce
  - Router pings often timeout during HSN quiesce
  - No available routes means the only new transmits are router pings
- **With fast reconnect gnilnd is more aggressive with re-establishing connection**
  - Disabled on routers
  - LNet is notified upon reconnect

Tuning for Resiliency – Adaptive Timeouts

- **Upper and lower bounds are configurable**
  - Lower bound = at_min
  - Upper bound = at_max
- **Generally want lower timeouts so lost messages are detected quickly, but we don't want false positives**
- **Upper bound limits potential impact of lost RPCs**
  - Able to lower time to recovery from link resiliency from ~20 minutes to ~15 minutes simply by lowering at_max

Tuning for Resiliency – Lock Callback Timer

- Callback RPCs use same timeouts as other RPCs
- Lower bound of lock callback timer configured separately: ldlm_enqueue_min
- The lock callback timer should allow enough time for at least one resend
  - Tradeoff: Time to detect misbehaving client vs. Time for resend
- ldlm_enqueue_min = max(2*net_latency, net_latency + quiesce_time) + 2*service_time
- quiesce_time may vary on system size, number of clients, number of mounted filesystems, etc.

Tuning for Resiliency – LNet

- **Router Pinger**
  - Faster/more pings on server side
  - Slower/fewer pings on client side
- **Asymmetric Route Failure Detection**
  - Disabled on routers
  - Enabled everywhere else
- **Peer Health**
  - Enabled on routers
  - Disabled everywhere else
- **Lustre Network Driver**
  - ko2iblnd timeout default is too high

Site-specific Tuning

- **Try to measure quiesce time -> increase/decrease ldlm_enqueue_min appropriately**
  - 21:26:51.388273-05:00 c1-0c2s5n0 LNet: Quiesce start: hardware quiesce
  - 21:27:06.393195-05:00 c1-0c2s5n0 LNet: Quiesce complete: hardware quiesce
  - 21:27:13.429388-05:00 c1-0c2s5n0 LNet: Quiesce start: hardware quiesce
  - 21:27:23.435159-05:00 c1-0c2s5n0 LNet: Quiesce complete: hardware quiesce
  - 21:28:24.938501-05:00 c1-0c2s5n0 Lustre: snx11023-OST0009-osc-ffff880833997000: Connection restored to snx11023-OST0009 (at 10.149.4.7@o2ib)
  - 21:28:49.952123-05:00 c1-0c2s5n0 Lustre: snx11023-OST0002-osc-ffff880833997000: Connection restored to snx11023-OST0002 (at 10.149.4.5@o2ib)
  - 21:29:05.252357-05:00 c1-0c2s5n0 Lustre: snx11023-OST000c-osc-ffff880833997000: Connection restored to snx11023-OST000c (at 10.149.4.8@o2ib)
  - Time from first quiesce message to last “Connection restored” is 124 seconds
- **at_max adjusted based on server load/worst case timeouts**
  - Lustre: ost_io: This server is not able to keep up with request traffic (cpu-bound).

Future Work

- **NRS Delay**
  - Details in https://jira.hpdd.intel.com/browse/LU-6283
- **Imperative Eviction**
- **at_net_min, at_net_max**
- **Add resend for other request types**

Summary

- Lustre does not deal with dropped messages very well
- Hole in Lustre protocol has been fixed
- Occurrence of client eviction resulting from message loss reduced
- Performance impact from message loss reduced
- More information and detailed tuning advice in the paper
- Our team is committed to continued improvements

Q&A

Chris Horn
hornc@cray.com