whamcloud

Lustre User Group
Orlando Fl
April 2011

A Scalable Health Network For Lustre

- Eric Barton
  CTO
  Whamcloud, Inc

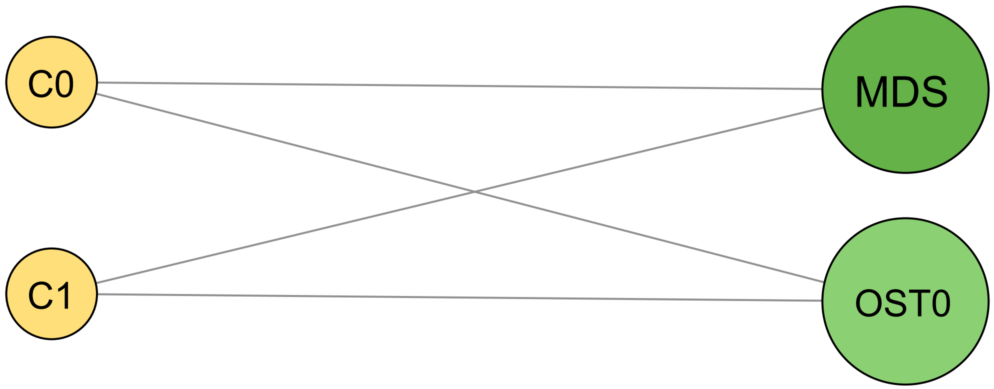
LNET Fault Detection Today

- Based on LND timeout
  - Independent of Lustre timeout
  - Token buildup if Lustre retries too eagerly
- Confused by congestion
  - Eager reader assumption
  - Requires long timeout

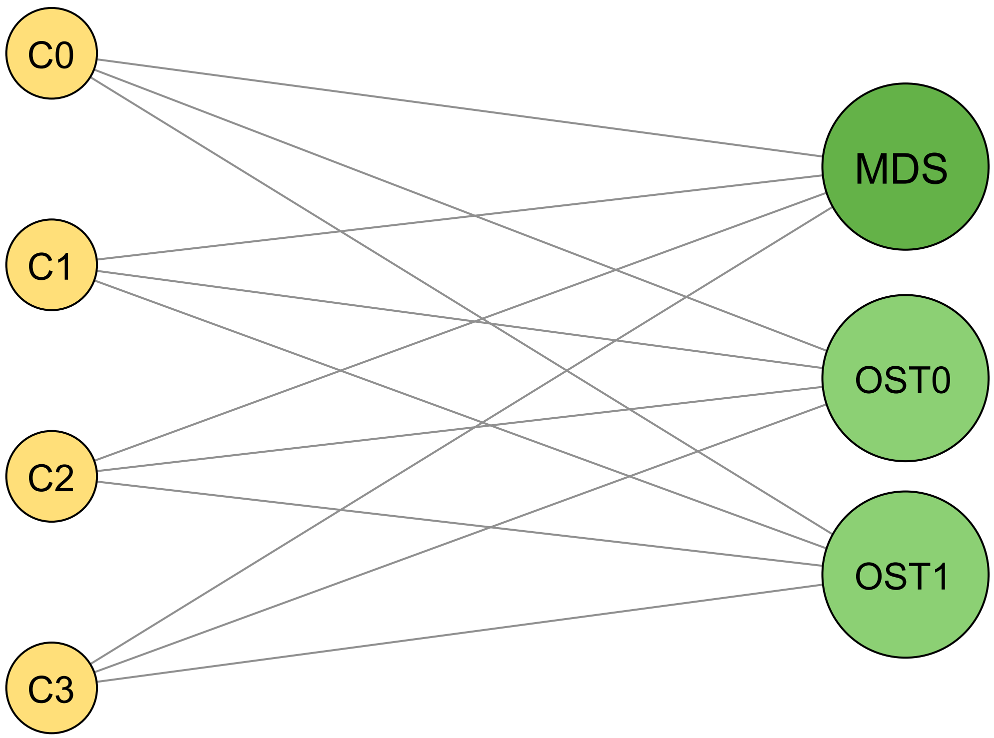
Lustre Pinger

- RPC timeout
  - Sole method of fault detection
- Dead client discovery
  - Delayed until DLM conflict
    - BAST timeout
  - Cascading timeouts
- Pinger
  - Keep-alive
  - Eager eviction on client death

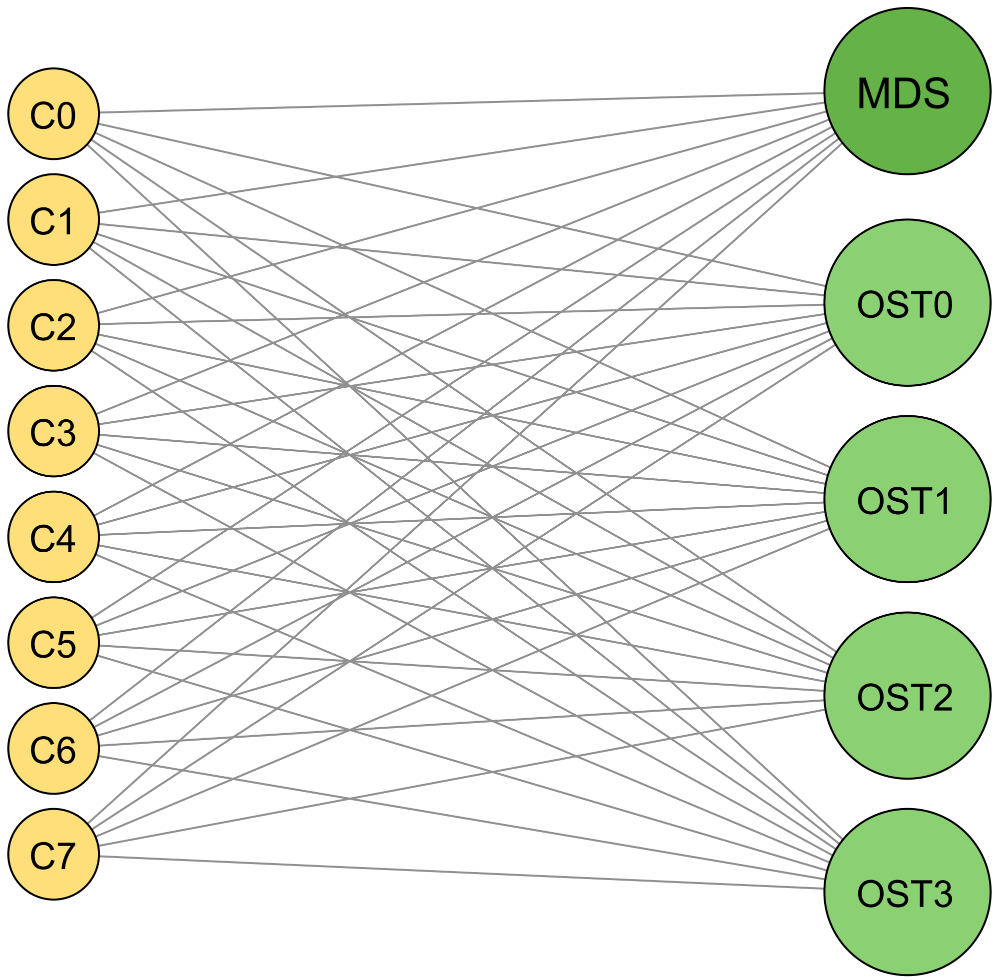
Ping Overhead



Ping Overhead



Ping Overhead



Lustre Fault Detection Today

- "Every man for himself"
  - No non-local fault notification
  - Inherently non-scalable
    - O(n**2) pings for constant ping interval
    - Compromise on O(n) ping interval
- Exclusive reliance on in-band RPC timeouts
  - Network and service latency highly variable
    - Depends on load and usage patterns
  - Must distinguish congested v. dead peer
    - False error detection compounds load
  - Timeouts are long to include disk latency and congestion
    - Adaptive timeouts can't alter the worst case
- O(n) fault detection latency
  - With a large multiplier

Server Recovery

- Recovery "Window"
  - Server must wait for all live clients to reconnect
  - Late replay risky
  - Ensure dependent transactions replay in correct order
    - Commit-on-share avoids need but penalizes normal operation
- Conservative window duration
  - Clients must first timeout the previous server instance
  - Then allow for two attempts to reconnect
    - First attempt retries same NID
      in case of transient communications failure
  - Required if imperative recovery not available

Server Recovery

Example scenario

- Configuration
  - File-per-process, 4 stripes/file
  - 20,000 clients, 12 processes/client
  - 8 x 1MByte RPCs in flight per client * OST
  - 100 OSS nodes
  - OSS bandwidth 2.4GB/sec
- Average OSS request queue depth: ~75,000
- Average I/O RPC latency: ~30s
- Minimum safe timeout: ~300s
- Recovery window: ~1000s

Client Eviction

- No non-local fault notifications
  - Servers evict clients independently
- Clients may write OST objects after MDS eviction
  - Problem for…
  - Create-on-write
    - Must guarantee client cannot re-create destroyed object
  - OST-derived attribute caching on MDS
    - Size (SOM), Dirty flag (HSM)
    - Must invalidate MDS cache on OST update

Moore's Law

- Relentlessly increasing scale
  - Today
    - 100s of server nodes, 100,000s of client nodes
    - MTTF of 100s of hours
  - Anticipated
    - 1000s of server nodes, 1,000,000s of client nodes
    - MTTF of 100s of minutes
- Prompt fault handling mandatory
  - Avoidance
  - Recovery

Health Network Requirements

- Low latency fault detection
  - Servers and clients
  - Reliable
- Low latency global notification
  - Reliable to servers, best efforts to clients
- Server collectives
  - Close-coupled state shared between servers
- Scalable
  - 1,000s servers, 1,000,000s clients
- Minimal administration / configuration
- Low overhead
  - Server CPU & Networking

Health Network Assumptions

- Servers and LNET routers
  - Not malicious
    - Try to participate constructively in HN protocols
    - May be buggy ("flapping")
  - Many (all) may crash/restart together
    - Cluster reboot / power fail
  - Normally don't crash/restart
    - Population stable for at least 10s of minutes at a time
    - Easily long enough for collectives to succeed
- Clients
  - Can't be relied upon
  - Population may never reach stability
- (Re)connection is O(n) overhead
  - Normal operation is lower overhead

LNET

- Additional uncongested virtual network
  - Hi-priority messages
    - Extension of LND RDMA setup / zero-copy completion
  - No routing
    - Guaranteed eager reader
  - Rate limit ingest
    - Discard when per-peer message rate exceeds agreed threshold
    - Underutilization provides latency guarantee
- Peer death detection
  - Prompt fault detection while utilized
    - Message timeout scaled to link latency
    - no networks with “beer” timeouts
  - Not fooled by congestion
    - Hi-priority keepalives on backpressure
  - Dead peer == /dev/null

Health Network Construction

- Spanning tree over servers and LNET routers
  - Paxos root
    - Highly available
  - Wide / shallow
    - Branching ratio O(forwarding_latency * send_rate)
  - Clients balanced across tree nodes/routers in same LNET network
- Parent node selection
  - Root maintains tree topology
    - Detects “flapping” nodes
  - Root LNET network nodes
    - Query root directly
  - Non-root LNET network nodes
    - Proxy query via any local router

Tree communications

- Tree version
  - Increment on server/router attach/death
- Requests
  - Forwarded to root and transformed into a notification
    - Rate limit for congestion avoidance
  - Combine compatible requests from self/children
    - Collective requests block for all children
  - Destroy collective requests on tree version change
- Notifications
  - Forward/broadcast down tree towards leaves
  - Destroy duplicate notifications
  - Requestors retry on version change

whamcloud

Peer Liveness

- Servers/Routers
  - Sustain minimum message rate to parent and children
    - Send keepalives while idle
  - Regard immediate peers as dead on
    - Sufficient interval of silence
    - LNET notification
  - On parent death, rejoin tree retaining existing children
  - On child death, send notification request
    - Root discards if stale
- Clients
  - Sustain minimum message rate to monitoring tree node
    - Scale to reflect increased branching ratio

Benefits

- Scalable server collectives
  - Single system image tables
  - Gang-scheduling for true QoS
  - Scalable distributed transactions (epochs)
- Scalable, reliable server restart notifications
  - Reduced reliance on congestion-based timeouts
  - Collectives distribute Imperative Recovery target status table
    - No need to back off to timeout based recovery
- Scalable, reliable global client connection/eviction
  - Clients need not connect to all server nodes immediately on startup
  - Lock callbacks can “try harder”
  - No O(n**2) pinger overhead
  - Safeguards create-on-write, SOM, HSM “dirty” flag

Thank You

- Eric Barton

  CTO

  Whamcloud, Inc.