CRAY
THE SUPERCOMPUTER COMPANY

Automated Lustre Failover on the Cray XT

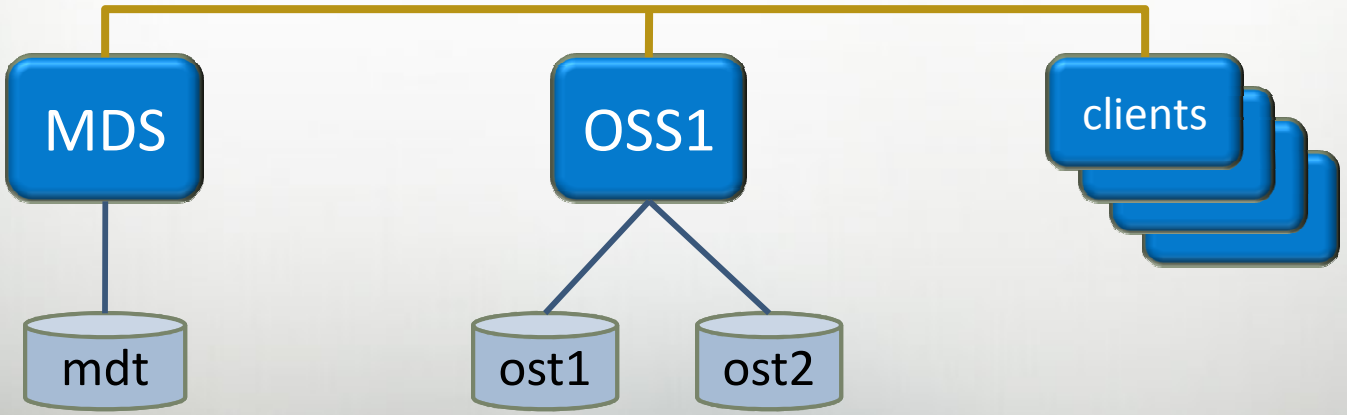
Nicholas Henke, Wally Wang and Ann Koehler
CUG 2009
Cray Inc.

Overview

- Lustre Background
- Why Lustre Failover ?
- How does Lustre Failover work ?
- Automation on the Cray XT
- System configuration requirements
- Software configuration for failover
- Current limitations
- Future work

Lustre Background

- Server Nodes and services
  - ***MDS*** with one ***mdt*** per file system
  - ***OSS*** with one or more ***ost*** per file system
- Clients maintain connections to each service
  - Failure detection is via network timeouts



Why Lustre Failover ?

- Loss of Lustre server currently requires machine reboot
  - Parallel file system is a critical resource for users
  - Decreases MTTI and increases downtime

- Interrupts impact Service Level Agreements and customer satisfaction
  - Cray ←→ Customer
  - Customer ←→ Users

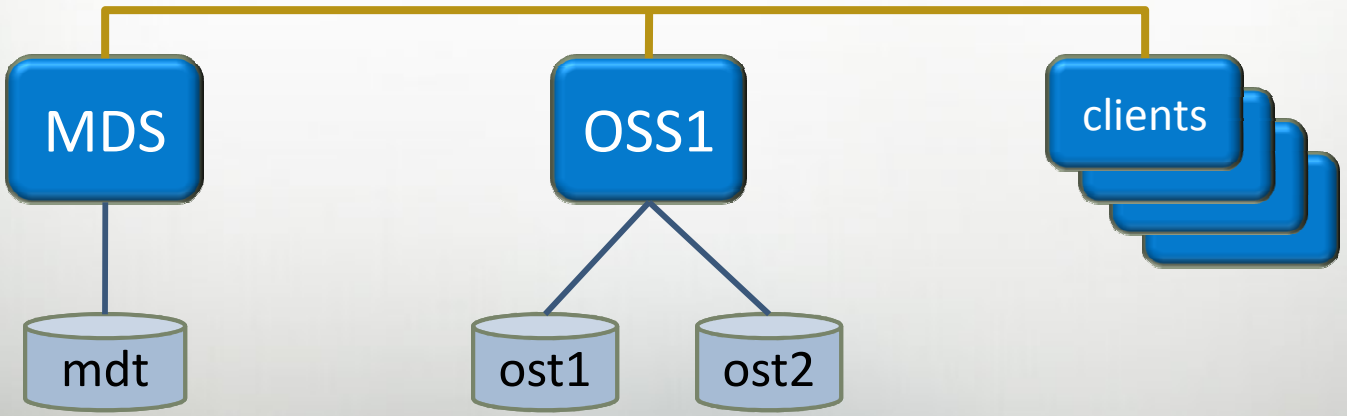
Why Lustre Failover, continued ?

- Objective is to keep the system functioning while minimizing job loss
- Regain machine functionality after Lustre server death
  - Access same data and files by connecting to backup server
- Primarily handles Lustre server death
  - Some documented cases of successful failover due to link failure
  - Depends on nature of network failure
- Warm-boot of Lustre servers
  - Uses same recovery methods

What does Lustre Failover not do ?

- Lustre Failover is not able to handle RAID subsystem failures
  - Storage controllers
  - Service node HBA
  - Connection from service node to storage array

- Solutions to these are being investigated



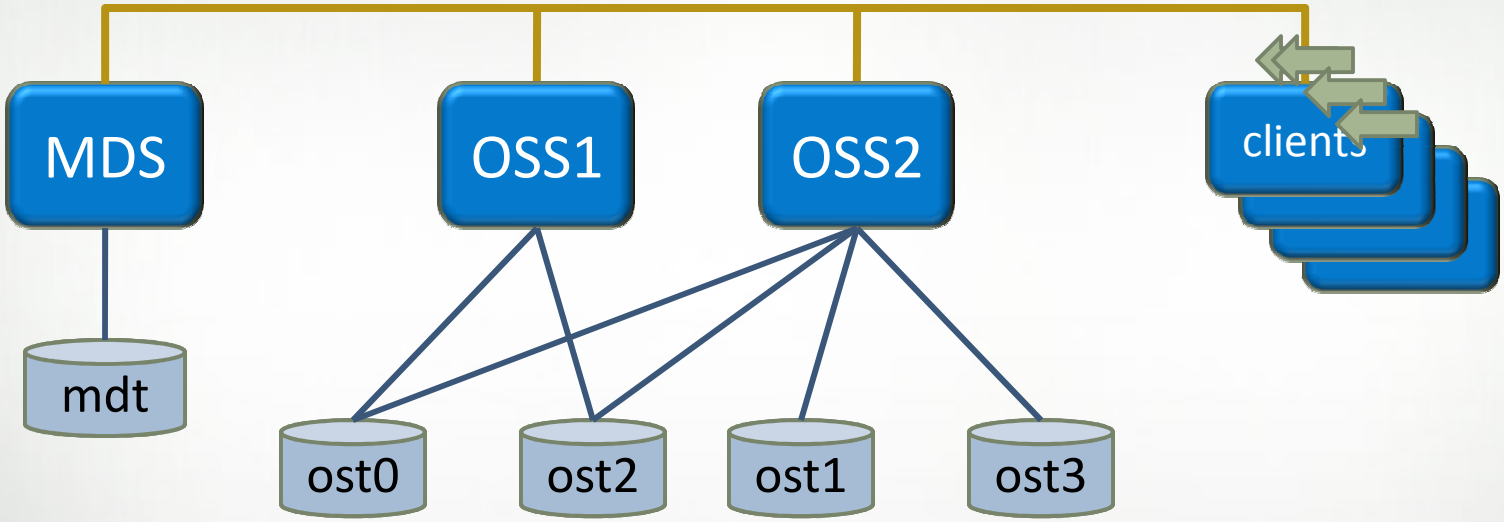
How does Lustre Failover work ?

- OSS1 dies
  - Was serving ost0, ost2
- ost0,ost2 started on OSS2
  - Waits for all clients to reconnect

- Client traffic to OSS1 times out
  - Clients try to reconnect to OSS1, this also times out
- Clients connect to OSS2

- Clients replay outstanding transactions
- Clients start sending new I/O requests

How does Lustre Failover work ?



Automation on the Cray XT

- Automation components
  - State Management
  - Health monitoring
  - Taking action

- XT automation achieved through Cray-developed xt-lustre-proxy
  - Runs as daemon on every Lustre server
  - CRMS Framework for heartbeat events
  - SDB for configuration and maintaining current state

Automation on the Cray XT: details

- CRMS heartbeat events
  - Existing node failed event
    - sent when node stops updating heartbeat
  - Added new Lustre service heartbeat
    - Use Lustre provided /proc health check
    - If health check fails, proxy stops updating Lustre service heartbeat
- Proxy and heartbeat
  - At startup, queries SDB for configuration
  - Registers for events for services it is backing up
- On server death, proxy takes action
  - “Shoots” node via CRMS event to ensure it stays dead
  - Start services on backup server

System Configuration Requirements

- OSS nodes are typically configured in active/active mode
  - Requires storage connectivity from both nodes
- MDS nodes configured in active/passive mode
  - Requires backup MDS on separate SIO node
  - If system has multiple file systems, MDS can be configured in active/active mode
- Hardware configuration changes
  - Cabling, zoning
  - Non-mirrored write cache turned off
- OSTs per OSS limits
  - Failover doubles, need to ensure survivability

Software configuration for failover

- Changes for FILESYSTEM.fs_defs
  - **OSTDEV[0]** =“nid00060:/dev/sda1 nid00063:/dev/sdb1”
  - **AUTO_FAILOVER**=yes
- lustre_control scripts
  - ‘*generate_config.sh*’ will generate CSV data files for proxy configuration
  - ‘*lustre_control.sh FILESYSTEM.fs_defs write_conf’* will push CSV tables into the SDB
- Manual configuration
  - xtfilesys2db, xtlustreserv2db, xtlustrefailover2db
  - xtlusfoadmin

Current limitations

- Failover duration is not optimal
  - Usually 10-15 minutes
  - Can take up to 30 minutes
- Quotas and MDS failover
  - Known issues in XT 2.2, working with Sun at high priority
- Some job loss is inevitable
  - Users with tight batch wall-clock limits
  - Client death during failover

Current limitations

- Manual failback
- Multiple filesystems and lustre_control configuration
  - Documented solutions
- Manual status monitoring
  - lctl get_param *.*.recovery_status

    status: RECOVERING
    recovery_start: 1236123918
    time_remaining: 886
    connected_clients: 1/178
    completed_clients: 1/178
    replayed_requests: 0/??
    queued_requests: 0
    next_transno: 1268285

Future work

- Imperative Recovery
  - Working with Sun to develop feature
    - Force client reconnect and stop server waiting on dead clients
  - Reduce failover times to under 5 minutes, ideally 1 to 3 minutes
- Version Based Recovery
  - Minimized evictions caused by unconnected clients
  - Only transactions requiring missing data will fail
  - Adaptive Timeouts
- Gemini Network
  - Allows shorter network timeouts and positive feedback on dead peers
- Targeted for Danube Release


CRAY
THE SUPERCOMPUTER COMPANY