Lustre Status and Roadmap

- Eric Barton
  CTO
  Whamcloud, Inc.
  eeb@whamcloud.com
Agenda

- **Whamcloud**
  - Who we are and what we do

- **Roadmap**
  - Release model
  - Upcoming releases
  - Development

- **Growing the market**
  - Making Lustre easier to use

- **Looking ahead**
  - Long term development direction
Whamcloud

- **Lustre for HPC on Linux**
  - Lustre development and support
  - World-leading Lustre engineering team

- **Multi-vendor**

- **Supporting the Lustre Community**
  - No copyright assignment on contributions
  - All Lustre releases: [http://downloads.whamcloud.com](http://downloads.whamcloud.com)
  - Jira bug tracker: [http://jira.whamcloud.com](http://jira.whamcloud.com)
  - Git repositories: [http://git.whamcloud.com](http://git.whamcloud.com)
  - Gerrit code review: [http://review.whamcloud.com](http://review.whamcloud.com)
  - Build: [http://build.whamcloud.com](http://build.whamcloud.com)
Lustre current status

- Lustre available from several sources
  - A common source tree
  - 1.8.* release from Oracle/Whamcloud
  - 2.1 community release from Whamcloud
  - Other parties doing their own releases

- Whamcloud release model
  - Maintenance releases
    - Currently 1.8.x
      - Expect switch to 2.x in 6-12 months
    - Proven reliability for production environments
    - Bugfixes only in maintenance updates
  - Feature Releases
    - Currently 2.1
    - Major Lustre releases containing new features
    - High level of focus on quality but production exposure more limited
**Lustre Roadmap** (wiki.whamcloud.com)

### Maintenance Release
- **1.8.6**
  - RHEL6 client support
  - 24TB LUNs

### Feature Release
- **2.1**
- **2.2**
- **2.3**
- **2.4**
  - Full RHEL6 support
  - Async journal commits on by default
  - ext4 by default
  - 128TB LUNs
  - LNET SMP scaling
  - Imperative recovery
  - Metadata performance

### Future features
- pNFS trial
- Job stats
- HSM
- Replication
- Dynamic NIDs
- Wide striping
- BTRFS OSD
- Patchless server

**Q2** 2011
**Q3** 2011
**Q4** 2011
**Q1** 2012
**Q2** 2012
**Q3** 2012
**Q4** 2012
**Q1** 2013
**Q2** 2013
Next Releases

• Maintenance: Lustre 1.8.6-wc
  - Available June 30th
  - RHEL6 client support
  - 24TB LUNs

• Feature: Lustre 2.1
  - Availability dependent on scale test resources
  - RHEL6 client and server
  - Async journal commits enabled by default
  - Ext4 Idiskfs enabled by default
  - Stability enhancements
  - 128TB LUNs
Lustre Roadmap (wiki.whamcloud.com)

Maintenance Release

1.8.6
* RHEL6 client support
* 24TB LUNs

Feature Release

2.1
* Full RHEL6 support
* Async journal commits on by default
* ext4 by default
* 128TB LUNs
* Stability enhancements

2.2
* LNET SMP scaling

2.2.1

2.2.2
* Imperative recovery
* Metadata performance

2.3
* OSD restructuring
* Server stack SMP scaling
* Distributed namespace
* Online check/scrub

2.4

Future features
* pNFS trial
* Job stats
* HSM
* Replication
* Dynamic NIDs
* Wide striping
* BTRFS OSD
* Patchless server

Q2 2011
Q3 2011
Q4 2011
Q1 2012
Q2 2012
Q3 2012
Q4 2012
Q1 2013
Q2 2013
Metadata performance: `ls -l, du, find`

- **Accelerate namespace traversal**
  - Common to `ls`, `du`, `find` etc
  - Scheduled for 2.3 – mid 2012

- **More efficient ldlm/object hash**
  - Reduce hash bucket depth from ~3K to < 50

- **Readdir**
  - 1 page per RPC in current releases
  - 1 Mbyte per RPC reduces overhead

- **Stat**
  - MDS attributes
    - Getattr RPC fetches & locks UID, GID, nlink etc
    - Statahead pipelines RPCs and populates dcache and inode cache
  - OST attributes
    - Glimpse RPC fetches & locks size, blocks, mtime, ctime etc
    - Asynchronous glimpse pipelines RPCs
Accelerated namespace traversal

'ls -l' with subdir items

- original
- improved

'ls -l' with 4-striped items

- original
- improved

<table>
<thead>
<tr>
<th>Item Count (100K)</th>
<th>Traversal Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>8</td>
<td>25</td>
</tr>
<tr>
<td>16</td>
<td>52</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item Count (100K)</th>
<th>Traversal Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>33</td>
</tr>
<tr>
<td>4</td>
<td>69</td>
</tr>
<tr>
<td>8</td>
<td>136</td>
</tr>
<tr>
<td>16</td>
<td>272</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item Count (100K)</th>
<th>Traversal Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>38</td>
</tr>
<tr>
<td>2</td>
<td>116</td>
</tr>
<tr>
<td>4</td>
<td>410</td>
</tr>
<tr>
<td>8</td>
<td>1692</td>
</tr>
<tr>
<td>16</td>
<td>7248</td>
</tr>
</tbody>
</table>
Imperative recovery: why

• Recovery “window”
  - Server must wait for all live clients to reconnect
  - Clients only start recovery after timeout with previous server instance
    • Then allow for two attempts to reconnect

• Fault detection with in-band RPC timeouts
  - Network and service latency depend on load and usage patterns
  - Long timeouts to distinguish congested v. dead peer
    • Service queue depth 100,000s
    • Peak request latency approaching 100s seconds
    • Recovery window 10s minutes

• Imperative recovery
  - Explicit server restart notification
  - Eliminates wait for clients to time out
  - Scheduled for 2.3 – mid 2012
Imperative recovery: how

- **Explicit notification**
  - Delivered to clients on server restart
  - Back off to timeout based methods when unavailable

- **Shared Target NID Table**
  - Target / network address binding maintained at MGS
  - Protected by DLM locks
    - Read by clients while holding shared DLM lock
    - Updated by MGS while holding exclusive DLM lock
  - Incremental updates by table version
Lustre Roadmap (wiki.whamcloud.com)

Maintenance Release
- 1.8.6
- Full RHEL6 support
- Async journal commits on by default
- ext4 by default
- 128TB LUNs
- Stability enhancements

Feature Release
- 2.1
- LNET SMP scaling
- Imperative recovery
- Metadata performance
- 2.2
- 2.2.1
- 2.2.2
- 2.3
- 2.4
- 2.4.1
- RHEL6 client support
- 24TB LUNs
- Async journal commits on by default
- ext4 by default
- 128TB LUNs
- Stability enhancements

Future features
- pNFS trial
- Job stats
- HSM
- Replication
- Dynamic NIDs
- Wide striping
- BTRFS OSD
- Patchless server

Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2
2011 2012 2013

Future features
- OSD restructuring
- Server stack SMP scaling
- Distributed namespace
- Online check/scrub
Metadata performance: vertical scale

- **MDS throughput CPU bound**
  - Poor affinity (cacheline pinging between CPUs)
  - Lock contention

- **Request affinity**
  - Define units of CPU affinity
    - Socket, core, hyper-thread
  - Separate RPC queue for each CPU unit
    - Reduced RPC queue lock contention
    - No data migration between CPU units

- **Shared directory locking**
  - LdIm
    - Parallel directory operations support implemented in Lustre 2.0
  - Backend filesystem
    - IAM (incompatible with Lustre 1.8)
    - Ldiskfs/ext4
      - Hierarchical lock with shared modes
      - Contend only on leaf nodes if split not required
Create in shared dir

Experimental results: 0-stripe files, 12-core MDS

files / sec

shared dir opencreate (improved)

files / sec

shared dir opencreate (original)

Logical client count

1 client  
2 clients  
4 clients  
8 clients

Logical client count

1 client  
2 clients  
4 clients  
8 clients
Metadata performance: horizontal scale

- **Distributed namespace**
  - Overcome vertical scaling limits
  - Inodes local to parent directory entry by default
    - Create scalable namespace using distributed (slower) operations
    - Use scalable namespace with non-distributed (fast) operations

- **DNE Phase 1 – remote directories**
  - Home/project dirs scattered over all MDTs
  - Home/project subdirs constrained to same MDT
  - Metadata performance isolation

- **DNE Phase 2 – striped directories**
  - Directory entries hashed over directory stripes
  - Files created on same MDT as directory entry
  - $O(n)$ speedup for shared dir ops (e.g. file-per-process create)
Online checker / scrubber

- Distributed namespace consistency
  - Bidirection references between directory entry and inode
- MDT/OST consistency
  - Bidirectional references between inode and stripe objects
- OSD internal consistency
  - Verify/rebuild OSD object index
  - *Not* checking underlying filesystem consistency
Growing the Lustre Market

Lustre administration

- De-mystified procedures
  - Provisioning
  - Maintenance
  - HA Setup
  - Fault diagnosis

- Management information
  - Performance
  - Utilization
  - Alerts

- Intuitive interfaces
  - GUI (single pane of glass)
  - Scriptable CLI (automation)

- System integration
  - Multi-vendor storage management
  - Multi-vendor cluster/site management
Growing the Lustre Market

Stability

- Release model
  - Feature
  - Maintenance

- Development process
  - Long term roadmap
  - Execute incrementally
  - Peer review

- Test
  - Early and often
  - Scale testing essential
  - Limited availability
  - Easy test framework deployment
  - Easy test execution
  - Easy test results gathering/sharing
Exascale challenges

• Scale
  – ~10,000 server nodes
  – ~1,000,000 client nodes

• Fault tolerance
  – MTTF of 100s of minutes

• Performance
  – Amdahl’s law: speedup = \( 1/(S + (1-S)/n) \)
    • 50% utilization exceeded only when \( S < 1/n \)
  – Punitive load-balancing requirements
    • Barrier “raggedness” \(<<\) Barrier interval

• Data Management
  – Multi-Pbyte
    • Simulation models
    • Unstructured data
  – Billions of entities
  – Ad-hoc queries
Scalable fault detection

- Current model inherently non-scalable
  - All-to-all $O(n^2)$ pings for constant ping interval
  - Compromise on $O(n)$ ping interval / detection latency
- Constraints
  - Scalable solution can’t monitor full $O(n^2)$ connectivity
  - Can’t detect all possible error conditions promptly
- Requirements
  - Must not depend on clients
  - Must not be affected by congestion (endpoint or network)
  - Must provide $O(0)$ detection latency for routine faults
  - Must detect all possible error conditions eventually
- Health Network Outline
  - Uncongested virtual network using hi-priority LNET messages
    - Extension of LND RDMA setup / zero-copy completion
  - Spanning tree over servers and LNET routers
    - Paxos root: highly available
    - Leaf clients balanced across servers/routers in same LNET network
  - Scalable communications
    - Message combining on path to root
    - Message replication on path to leaves
Health network benefits

- **Scalable server collectives**
  - Single system image tables
  - Gang-scheduling for true QoS
  - Scalable distributed transactions (epochs)

- **Scalable, reliable global client connection/eviction**
  - Lazy connection establishment
  - Client callbacks can “try harder”
  - Safeguards create-on-write, SOM, HSM “dirty” flag

- **Scalable, reliable server status notifications**
  - Much reduced reliance on service latency based timeouts
  - Prompt server death notification
    - Minimize interruption of service accessing replicated data
  - Prompt server restart notification
    - Collectives distribute Imperative Recovery target NID table
    - No need to back off to timeout based recovery
Exascale I/O model

- **Storage Pools**
  - Staging via storage tiers
  - Performance isolation
  - Space management

- **Conventional namespace**
  - Works at human scale
  - Administration, security, accounting
  - Legacy data / apps

- **Object collections**
  - Object I/O
    - Mixed application data and metadata
    - 100,000,000s of objects distributed over 1,000s of OSTs
  - ACID transactions on objects and collections
    - Safeguard precious data
  - Multiple application APIs
    - OODB, HDF5, NetCDF, Hadoop...
  - Application-specific resilience strategies
    - N-way mirroring, RAID6

- MapReduce data
  - Blocksequence
  - data
  - data
  - data

- Simulation data
  - OODB metadata
  - data
  - data
  - data

- Posix striped file
  - a
  - b
  - c
  - a
  - b
  - c
  - a

- Lustre Status and Roadmap

© 2011 Whamcloud, Inc.
Layering

- **Application Libraries**
  - Application/Query/Tool I/O API
    - Collective I/O
    - Placement/replication/migration
  - Many:1 user -> F/S object mapping
  - Transactional updates

- **Object collection API**
  - Middleware API
    - Divides responsibilities between layers
  - Collective create/open/close support
  - Transaction support
    - Object create/destroy/update and I/O barriers
    - Collection merge/split
  - Resilience
    - Raided/replicated objects
  - Object migration
    - Between OSTs
    - Between collections
Thank You

• Eric Barton
  CTO
  Whamcloud, Inc.
  eeb@whamcloud.com