Hewlett Packard
Enterprise

Experience Running DMF7 on Lustre

Olaf Weber
Master Technologist
HPC Data Management & Storage

Disclaimer

- The information contained in this presentation is proprietary to Hewlett Packard Enterprise (HPE) and may contain forward-looking information regarding products or services that are not yet available.

- Do not remove this slide from the presentation

- HPE does not warrant or represent that it will introduce any product to which the information relates

- The information contained herein is subject to change without notice

- HPE makes no warranties regarding the accuracy of this information

- The only warranties for HPE products and services are set forth in the express warranty statements accompanying such products and services

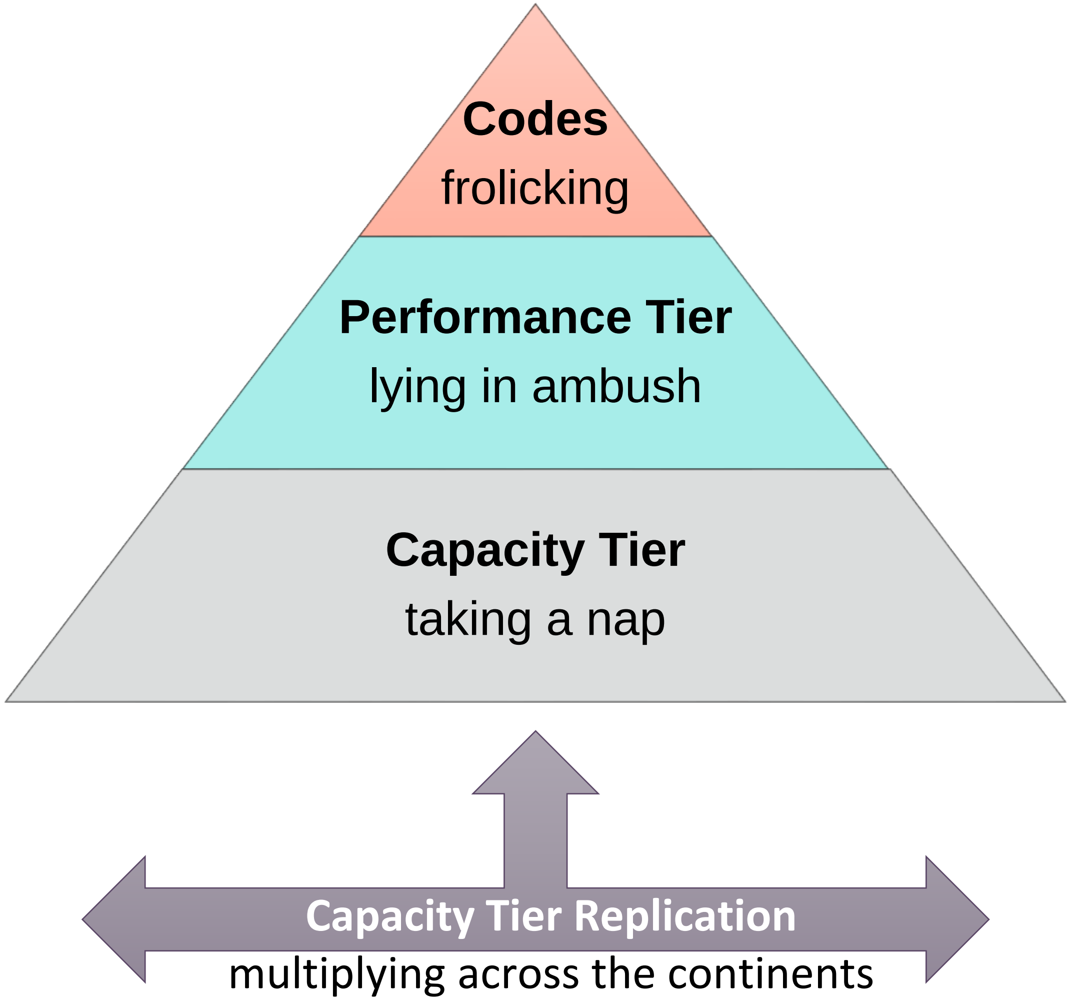
- Nothing herein should be construed as constituting an additional warranty

- HPE shall not be liable for technical or editorial errors or omissions contained herein

Data Management Framework 7

Data Management Framework 7

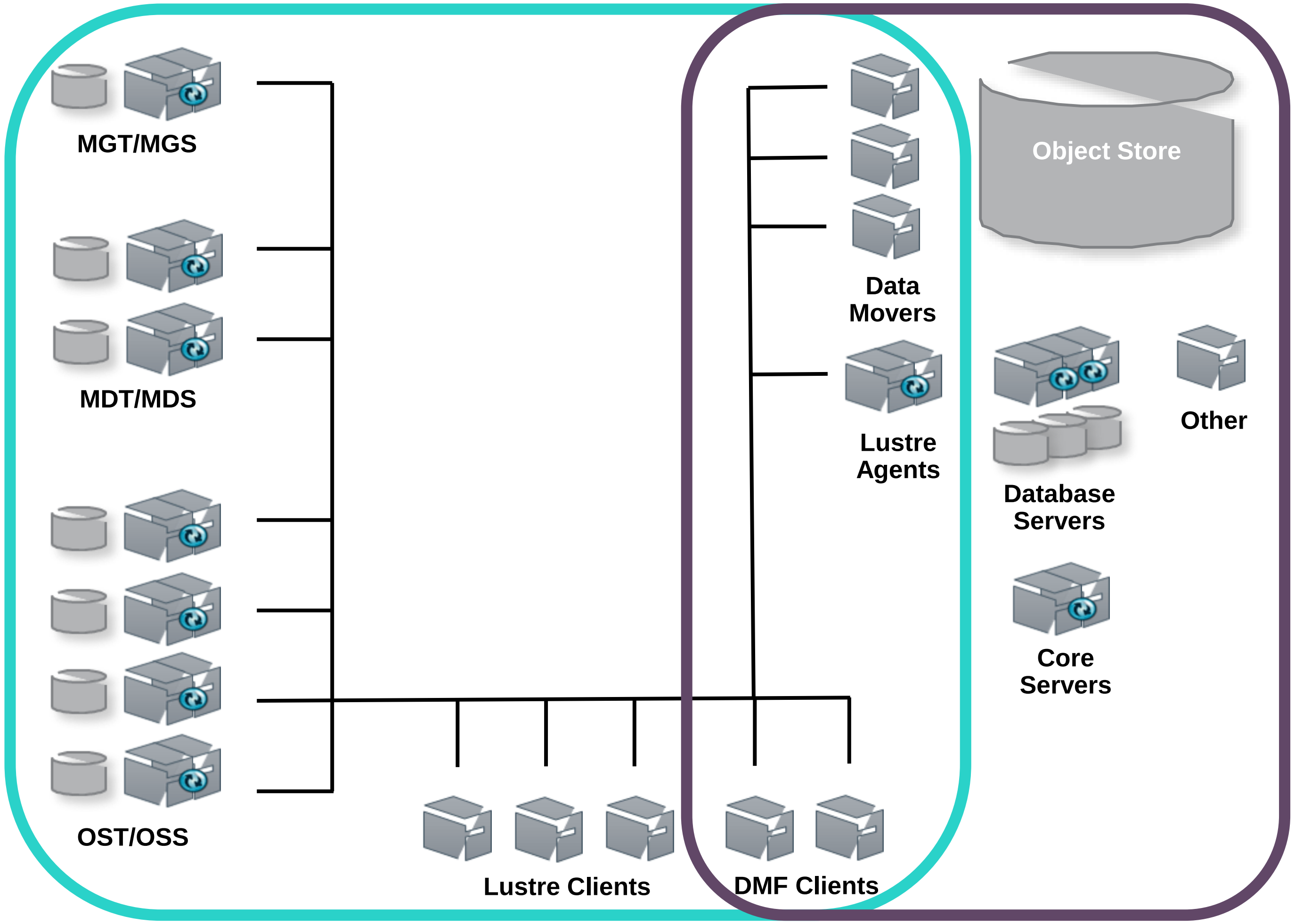
Overview



- Designed for Tiered Data Management
- Redesigned from the ground up
  - Build on lessons from DMF 6
- Designed for horizontal scaling
  - Scale by adding more servers
  - Distributed NoSQL database
- Many single-purpose components working together
- Most components are filesystem-agnostic
- Multiple supported filesystem types
  1. HPE XFS
  2. Lustre
  3. IBM Spectrum Scale (“GPFS”) in development

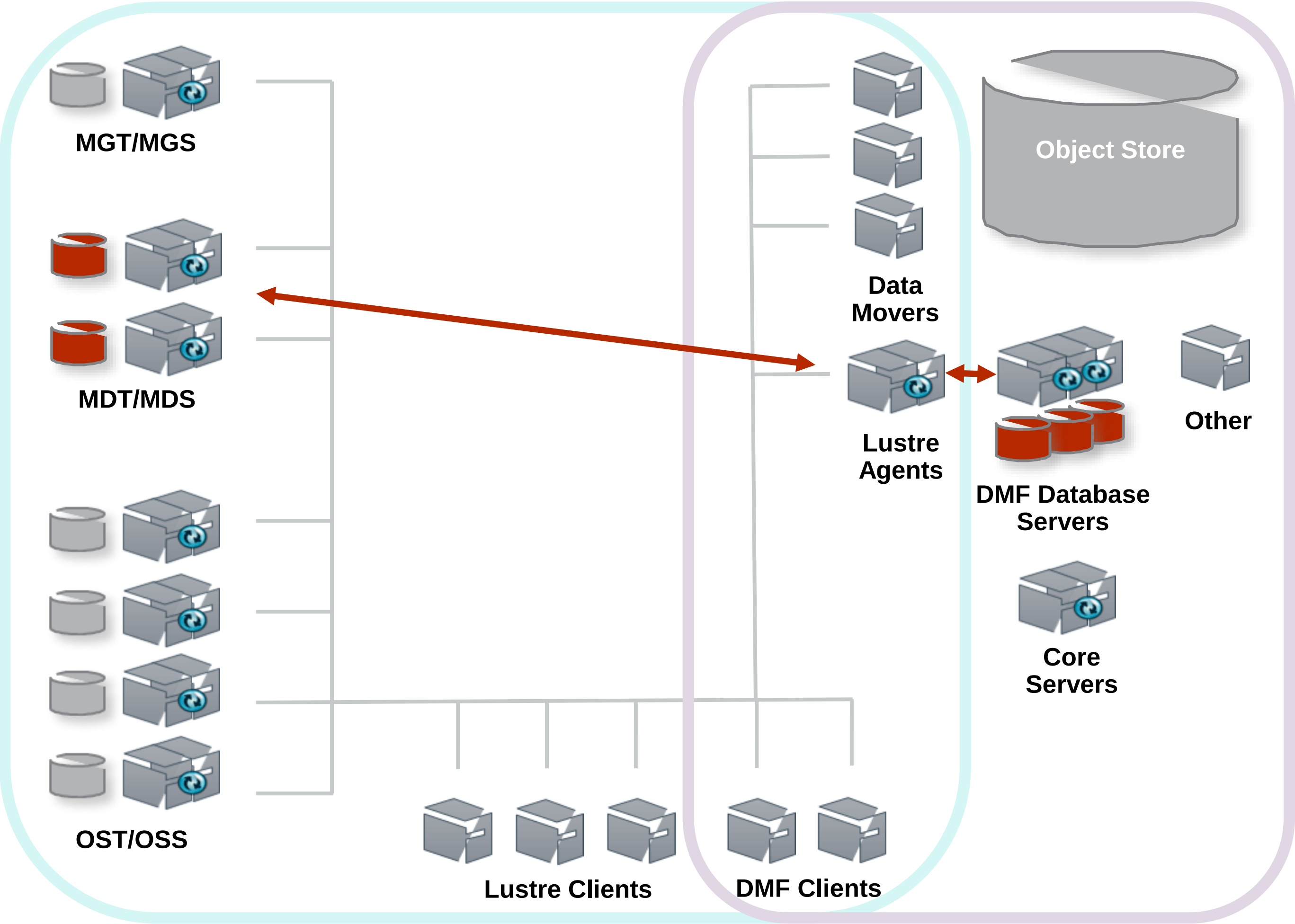
Data Management Framework 7 on Lustre

Roles of DMF7 nodes



- DMF Core Servers
  - Manage the other nodes
  - Provide the registry
  - Manage namespaces / filesystems
- DMF Database Servers
  - Manage the DMF database
  - Policy Agent
- DMF Data Movers
  - Move data between filesystem and backend
- DMF Lustre Agents
  - Changelog processor
  - Filesystem scanner
  - Database scrubber
- DMF Clients
  - DMF CLI available

Filesystem Reflection



- Synchronized copy of filesystem metadata
  - Inode metadata
  - Directory tree
  - Extended attributes
  - HSM state
- Maps Lustre FIDs to Object Store
- Cassandra database
- Maintained by the Lustre Agents
  - Filesystem scanner
  - Changelog processor
  - Database scrubber
- Used by policy engine
  - Parallel data mover framework
  - Copytool interfaces with Lustre HSM

Changelog

Lustre Changelog

```
# mkdir tmp

2112648 02MKDIR 09:30:25.501859712 2018.08.24 0x0
t=[0x200019271:0x2:0x0] ef=0xf u=0:0 nid=192.168.131.17@tcp1
p=[0x200000007:0x1:0x0] tmp

# chmod a+rwxt tmp

2112649 14SATTR 09:30:28.739566509 2018.08.24 0x14
t=[0x200019271:0x2:0x0] ef=0xf u=0:0 nid=192.168.131.17@tcp1

xfs_mkfile 1m file.1m

2112650 01CREAT 09:31:11.661327380 2018.08.24 0x0
t=[0x200019271:0x3:0x0] ef=0xf u=0:0 nid=192.168.131.17@tcp1
p=[0x200019271:0x2:0x0] file.1m

2112651 13TRUNC 09:31:11.741270796 2018.08.24 0xe
t=[0x200019271:0x3:0x0] ef=0xf u=0:0 nid=192.168.131.17@tcp1

2112652 11CLOSE 09:31:11.747861801 2018.08.24 0x243
t=[0x200019271:0x3:0x0] ef=0xf u=0:0 nid=192.168.131.17@tcp1

# touch file.1m

2112653 11CLOSE 09:36:18.556856115 2018.08.24 0x42
t=[0x200019271:0x3:0x0] ef=0xf u=0:0 nid=192.168.131.17@tcp1
```

- Tracks Metadata changes
  - Updated by MDS
  - Stored on MDT
  - Part of filesystem transactions
- Can only be read on Lustre client nodes
  - Must be root or equivalent
- Three types of metadata changes
  - Namespace
  - Side effects
  - Audit trail
- Controlled by per-MDT event mask
- Not a full log of the filesystem actions
  - Tracks that something changed…
  - …but not necessarily what changed

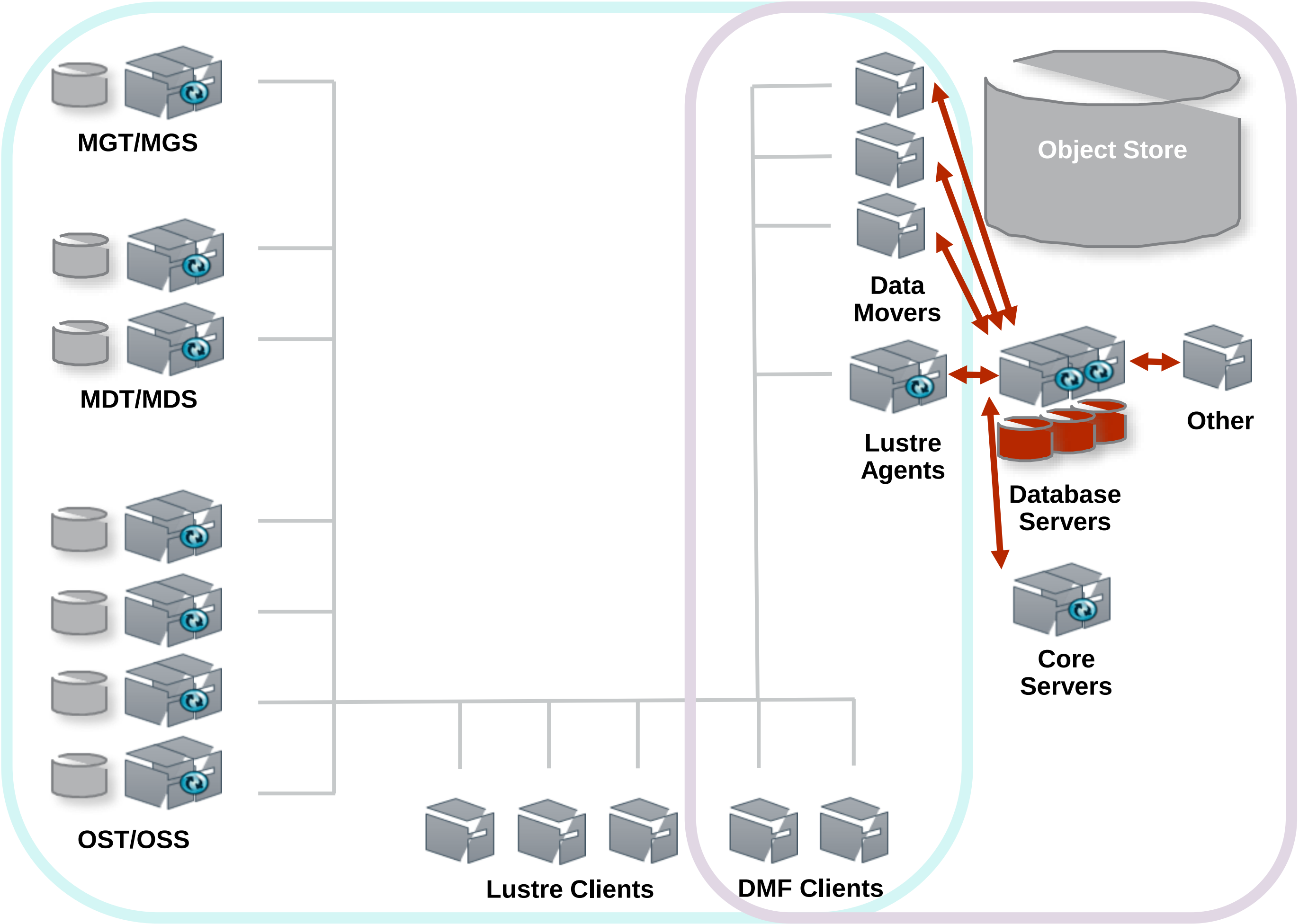
Reading the Wrong Changelog

LU-12650: get_root_path() filesystem name compare error that leads to fid2path fail

```
# lfs changelog lustre-MDT0000

2112649 14SATTR 09:30:28.739566509 2018.08.24 0x14
t=[0x200019271:0x2:0x0] ef=0xf u=0:0 nid=192.168.131.17@tcp1

2112650 01CREAT 09:31:11.661327380 2018.08.24 0x0
t=[0x200019271:0x3:0x0] ef=0xf u=0:0 nid=192.168.131.17@tcp1
p=[0x200019271:0x2:0x0] file.1m

2112651 13TRUNC 09:31:11.741270796 2018.08.24 0xe
t=[0x200019271:0x3:0x0] ef=0xf u=0:0 nid=192.168.131.17@tcp1

# lfs changelog lustre1-MDT0000

2112649 14SATTR 09:30:28.739566509 2018.08.24 0x14
t=[0x200019271:0x2:0x0] ef=0xf u=0:0 nid=192.168.131.17@tcp1

2112650 01CREAT 09:31:11.661327380 2018.08.24 0x0
t=[0x200019271:0x3:0x0] ef=0xf u=0:0 nid=192.168.131.17@tcp1
p=[0x200019271:0x2:0x0] file.1m

2112651 13TRUNC 09:31:11.741270796 2018.08.24 0xe
t=[0x200019271:0x3:0x0] ef=0xf u=0:0 nid=192.168.131.17@tcp1
```

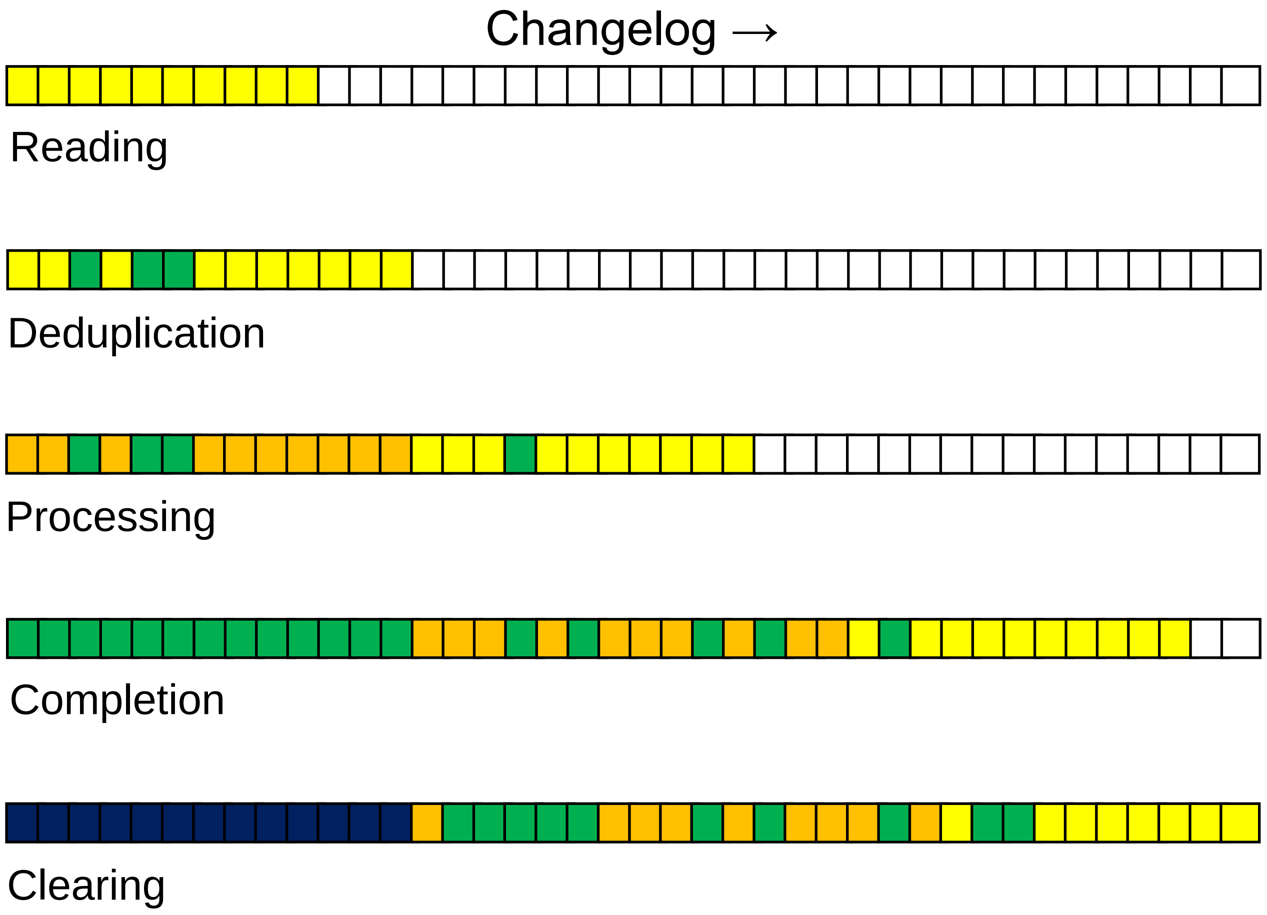
- We saw the same stream of records for:
  - lustre-MDT0000
  - lustre1-MDT0000
- Did not happen on all nodes.
- Cause:
  - “lustre” filesystem name is a prefix of “lustre1”
  - Code matches prefix instead of full string
  - This is a side effect of LU-12650
- Workaround:
  - No filesystem name that is a prefix of another

Processing Changelog Records



- Processing a changelog record is work
  - Avoid doing duplicate work
  - Read ahead and drop duplicate updates
- DMF7 coordinates through the reflection:
  - Reflection updates cannot be postponed long
  - The maximum deduplication window is small
- Deduplication is *required* for correctness:
  - Database timestamps have limited resolution
  - Limit to one update to a row within a window
  - Deduplication window has a minimum size
- Deduplication adds performance:
  - Reduce frequency of filesystem access
  - Reduce number of database updates
  - Gains are limited due to small window

Stepping through the Changelog



- Asynchronous processing of records:
  - White: unread records
  - Yellow: records read
    - Deduplicate as records are read
  - Orange: records being processed
    - Update of database requested
  - Green: completed records
    - Deduplicated records
    - Update of database confirmed
  - Dark Blue: cleared records
    - Clear contiguous blocks of completed records
- Recordkeeping tracks *out of order* records
  - Changelog processing got stuck

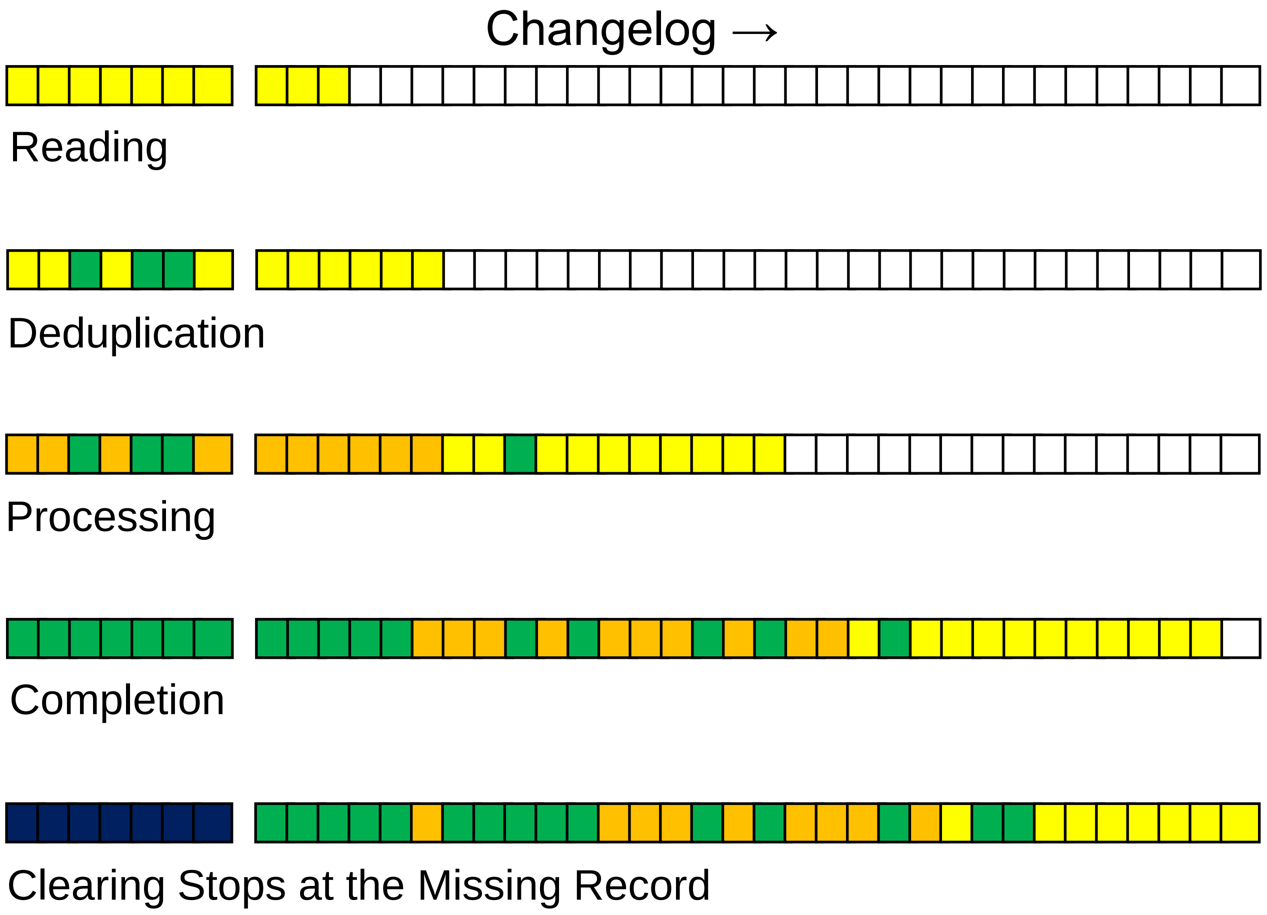
Out of Order Changelog Records

LU-11426: 2/2 Olafs agree: changelog entries are emitted out of order

- Sometimes the indices of subsequent entries are like this:
  - `2112648`
  - `2112649`
  - `2112651`
  - `2112650`
  - `2112652`
  - `2112653`
- Happens across all Lustre versions
  - Only requirement is sufficient concurrent activity on the filesystem
  - Easy to reproduce by running a cluster in VMs on a laptop
- Implicated in a number of issues:
  - LU-11426: 2/2 Olafs agree: changelog entries are emitted out of order
    - LU-11205: Failure to clear the changelog for user 1 on MDT
    - LU-11581: Not all changelog entries are returned to userspace

Missing Changelog Records

LU-11581: Not all changelog entries are returned to userspace



Changelog →

Reading

Deduplication

Processing

Completion

Clearing Stops at the Missing Record

- Sometimes a changelog record is missing:
  - Generated for an operation
  - Present in the on-disk log
  - But not returned to userspace
  - This is a side effect of LU-11426
- When a record does not appear at first:
  - It may be out of order and appear later
  - It may be missing and never appear
- Major impact on DMF:
  - Changelog processing got stuck
    - Hole in contiguous block of records
  - Heuristic enables progress
    - Assume loss after reading *N* more records
  - Filesystem reflection misses an update
    - Filesystem scan required

Also Affects RobinHood

LU-11205: Failure to clear the changelog for user 1 on MDT

- RobinHood is also affected
  - RobinHood processes the changelog in a different way
  - Thus very different symptoms
- Syslog messages:

```
… kernel: Lustre: 11137:0:(mdd_device.c:1577:mdd_changelog_clear()) fs-MDD0000:
  Failure to clear the changelog for user 1: -22
```

- RobinHood log messages:

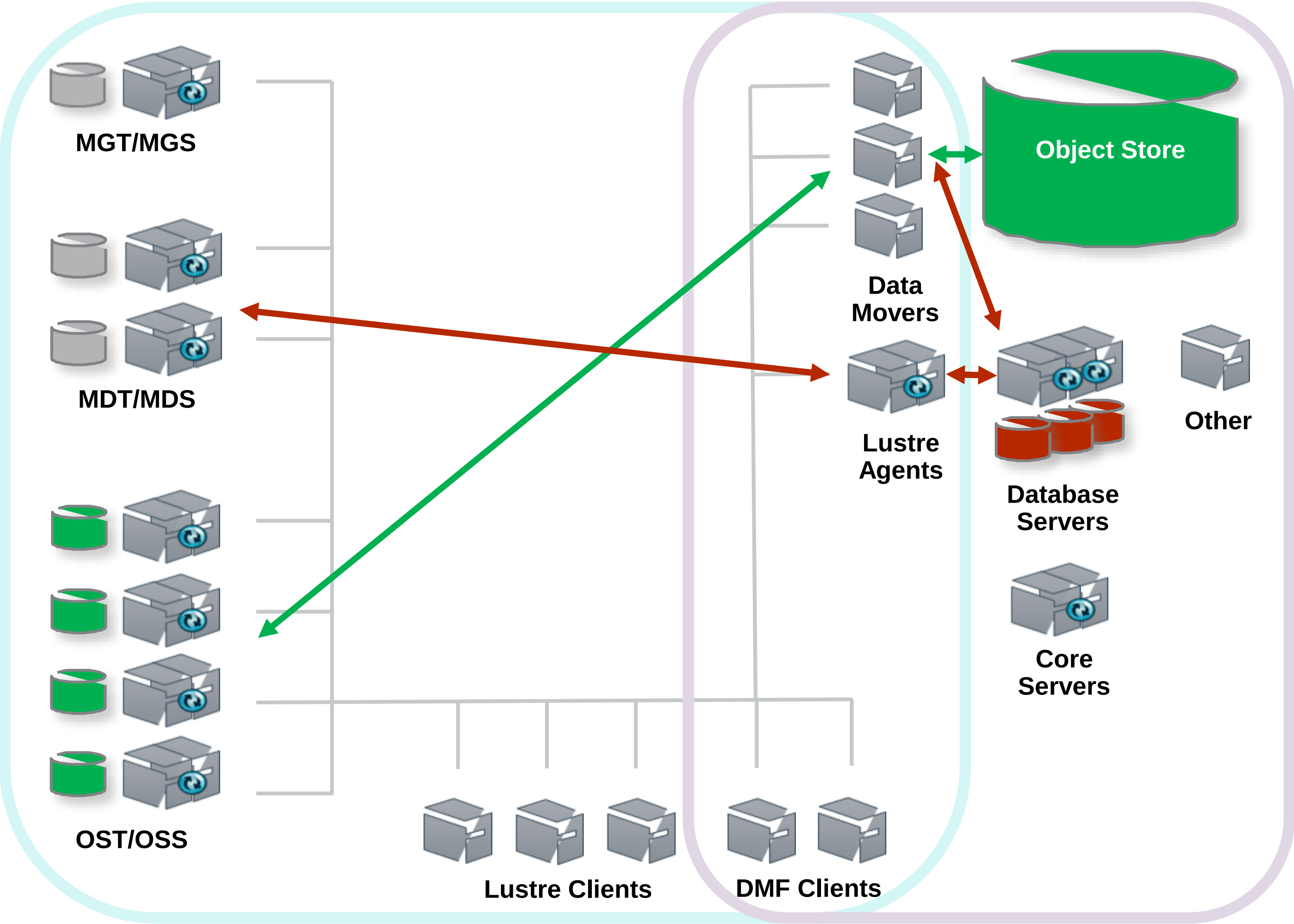
```
… [13766/22] ChangeLog | ERROR: llapi_changelog_clear(“fs-MDT0000", "cl1", 13975842301) returned -22
… [13766/22] EntryProc | Error -22 performing callback at stage STAGE_CHGLOG_CLR.
… [13766/16] llapi | cannot purge records for 'cl1`
```

- This is a side effect of LU-11426

HSM

Hewlett Packard
Enterprise

HSM Workflow



- Multiple ways to initiate data movement
  - Migration policy
  - Recall on access
  - Manual through CLI
- Data Mover nodes handle bulk traffic
  - Modified version of copytool on agent nodes
  - Mover nodes do actual data movement
- Coordination through the reflection
  - Reflection tracks HSM state
- Interacts with Lustre HSM coordinator
  - Part of Lustre kernel code
  - We encountered some issues

HSM Coordinator Restart Panics

LU-11675: Don't allow new HSM requests during CDT_INIT

```
# cd /sys/fs/lustre/mdt/lustre-MDT0002
# echo shutdown > hsm_control
# cat hsm_control
stopping
# cat hsm_control
stopped
# echo enabled > hsm_control
# cat hsm_control
init
# cat hsm_control
enabled
```

- Restart the HSM coordinator
- In *init* it looks for pending HSM requests
- In *init* phase it accepted new HSM requests
  - These could be given a duplicate ID
  - This triggered an assert
- Fixed under LU-11675
  - Fix is to not accept new requests during *init*

HSM Files Not Marked Dirty

LU-11369: hsm: files are not dirtied when modified by someone else than their owner

```
[alice] $ touch /mnt/lustre/alice/file
[alice] $ chmod o+w /mnt/lustre/alice/file
[alice] $ exit
logout
[root] # lfs hsm_archive /mnt/lustre/alice/file
[root] # lfs hsm_state /mnt/lustre/alice/file
/mnt/lustre/alice/file: (0x00000009) exists archived, archive_id:1
[root] # su - bob
[bob] $ echo "123" > /mnt/lustre/alice/file
[bob] $ exit
logout
[root] # lfs hsm_state /mnt/lustre/alice/file
/mnt/lustre/alice/file: (0x00000009) exists archived, archive_id:1
```

- An *archived* file has an identical copy stored
- A *dirty* file has an older copy stored
- Modifying an archived file marks it dirty
- This did not happen for files not owned by the modifying user
- Fixed under LU-11369
  - Code had permission to modify file
  - But not to modify file HSM state

Attributes Not Updated on Restored Files

LU-11925: Attributes not updated after open+append and write to archived,released file

```
# echo -n “123” > /mnt/lustre/file

# lfs hsm_archive /mnt/lustre/file

# lfs hsm_release /mnt/lustre/file

stat -c %s /mnt/lustre/file

3

# lfs hsm_restore /mnt/lustre/file

# echo -n “456” >> /mnt/lustre/file

# stat -c %s /mnt/lustre/file

3
```

- Query the size of a released file
- Append to file
- File size does not change
- Fixed under LU-11925
  - File size was obtained with an UPDATE lock
  - On restore any such lock must be canceled

Small Files

What is a Small File?

HPE Tfinity ExaScale Tape Library

- Tape is cheap bulk storage
  - If you can afford it
- Tape is slow
  - Physically moving a cartridge in a library
  - Mounting a tape
- Tape is fast
  - LTO-8 native transfer rates are > 300 MB/s
- Tape is big
  - LTO-8 cartridge holds 12 TB uncompressed
- Tape needs a continuous stream of data
  - A good I/O size is between 15 and 20 GB
- A file sized < 18 GB is a small file

Handling Small Files

Tape Zones

- DMF collects small files into *tape zones*
  - “Tape” for historical reasons
  - DMF does this for all files
    - Big files are split across zones
  - Large transfer units are good for S3 as well
    - Especially if you have to pay per transfer
- Zone is written and read as a single unit
  - Can be constructed as a temporary file
  - We prefer a “scatter-gather I/O” approach
- An 18 GB zone may hold many files
  - The files all migrate at the same time
  - All are open simultaneously
- At least one active zone per tape drive
  - 4 to 10 tape drives is typical

An Example for Context

- Take a fresh clone of the Lustre git repo
  - 160 MB
  - 1,923 files and directories
- In an 18GB zone
  - Lustre source fits 112 times
  - Zone then contains 215,376 files
- Assume 5 tape drives
  - 5 active zones
  - Lustre source fits 560 times
  - 1,076,880 files are migrating
- We have seen worse cases

The Small File Problem

- With small files, we can easily have several million files migrating at the same time
- The Lustre HSM coordinator thus needs to track several million active requests
- There is a maximum number of active HSM requests per MDT
- This limit is tunable
- The default is *3*
- The current HSM coordinator was not designed for large numbers of requests
- Direct migration to and from high-latency media requires rethinking the HSM coordinator
  - Handle large numbers of requests
  - Keep blocks of requests together
    - Submit all requests for a zone as a single logical block
    - The entire block is sent to a single copytool

Questioned Answers

Hewlett Packard
Enterprise

Thank you

Olaf Weber
olaf.weber@hpe.com