OAK RIDGE National Laboratory

Data Management Practices on Large Multi-Tier Lustre Filesystems

LUG 2026 Apr 28–29 Indianapolis, IN

Ethan Frenza
HPC Storage Systems Engineer
frenzaej@ornl.gov

Rick Mohr
Senior HPC Systems Engineer
mohrrf@ornl.gov

Brad Gipson
HPC Storage Systems Engineer
gipsonbm@ornl.gov

Jesse Hanley
Group Leader, HPC Storage and Archive
hanleyja@ornl.gov

U.S. DEPARTMENT of ENERGY

ORNL IS MANAGED BY UT-BATTELLE LLC
FOR THE US DEPARTMENT OF ENERGY

Overview

- Summary of Orion and AFW filesystem architectures
- Lustre features used on production file systems
- Experiences using Progressive File Layouts (PFL)
- Impact of large numbers of small files and symlinks
- Orion purge policy and implementation
- Handling performance tier usage

OLCF and AFW Filesystem Architectures

	OLCF [1] (Orion)	AFW [2] (Blizzard / Typhoon)
Hardware Platform	HPE Cray ClusterStor E1000	HPE Cray ClusterStor E1000
Lustre Version	2.15.7 (ZFS backend)	2.15.2 (ZFS backend)
Performance Tier Size	10.1 PB	552.7 TB
Capacity Tier Size	593 PB	17.6 PB
# MDS / MDT	40 / 40	2 / 2
# OSS / OST	450 / 1350 (1 performance tier OST and 2 capacity tier OSTs per OSS)	10 / 30 (1 performance tier OST and 2 capacity tier OSTs per OSS)
Accessibility	Frontier, Andes, DTNs, and other smaller OLCF systems	AFW compute nodes, AFW DTNs, and other small AFW systems

[1] For more details, see https://wiki.lustre.org/images/9/96/LUG2023-Orion_Configuration_Performance-Mohr.pdf

[2] Blizzard and Typhoon are identical filesystems, and the information provided is for each one.

Lustre Features Used in Production

To capitalize on multi-tiered storage, OLCF uses several Lustre features:

- **OST Pools**
  - File systems contain a combination of hard drives and NVMe drives
  - Define two pools named "performance" and "capacity" to control where data resides
- **Distrubuted Namespaces (DNE)**
  - File systems use multiple MDS severs to minimize metadata bottlenecks
  - Project directories (`/lustre/orion/<proj_name>`) are automatically distributed across available MDTs
  - Project directories only reside on a single MDT for now, but we're evaluating striped directories (DNE3)
- **Data on Metadata (DoM)**
  - Used in conjunction with Progressive File Layouts (PFL) to keep small portion of each file on MDTs
  - Reduces contention on OSTs and aims to provide better performance for small file I/O

Lustre Features Used in Production (cont.)

- **Progressive File Layout (PFL)**
  - Attempt to define a default file layout that is suitable for a range of file types
  - Small files should reside on fast storage (MDT or performance pool)
  - Large files should be striped across multiple targets in the capacity pool to prevent filling up OSTs
  - Self Extending Layout (SEL) used to guard against filling up OSTs

Orion Default PFL (diverse file sizes)

```
lfs setstripe -E 256K -L mdt
              -E 8M -c 1 -S 1M -z 64M -p performance
              -E 128G -c 1 -S 1M -z 16G -p capacity
              -E -1 -c 8 -S 1M -z 256G -p capacity
```

Smallest files reside on MDT

Small-ish files stay on the performance tier

Files spill over to capacity tier and stripe over multiple OSTs as they get larger

AFW Default PFL (small file focused)

```
lfs setstripe -E 64K -L mdt
              -E 16M -c 1 -S 1M -z 128M -p performance
              -E 128G -c 1 -S 1M -z 16G -p capacity
              -E -1 -c 4 -S 1M -z 256G -p capacity
```

Very similar to Orion except for a few small differences

Realities of PFL and Multi-Tier Storage

Advantages

More versatile than default with fixed stripe count

- Small files end up on storage with higher IOPs
- Large files end up on multiple OSTs w/ more bandwidth

Layout "just works" for most users

- Not perfect, but does a reasonable job in most cases without intervention from user

Prevents headaches for sys admins

- Lazy instantiation prevents inodes from being allocated when not needed
- Helps avoid filling up OSTs from mis-striped files
- SEL providers additional protection as OSTs are full or unavailable

Disadvantages

There are still ways to "do the wrong thing"

- User can manually run `lfs setstripe` with suboptimal parameters
- Software libraries can create files with poor layout choices because code is not aware of how to use multiple storage tiers
- Parameter inheritance logic in `lfs setstripe` can cause files to default to the performance tier

Can cause more work for sys admins (in some cases)

- Determining if a file is mis-striped is harder (can't just compare file size and stripe count)
- High performance tier usage requires file restriping to capacity tier

Large Numbers of Small Files and Symlinks

Even with workload-specific optimizations, we have observed issues related to user behavior, particularly in environments with large numbers of small files and/or symlinks.

- Misuse of the DoM tier can negatively impact overall MDT inode capacity. Issues observed include:
  - Symlinks pointing to deleted or nonexistent targets, which consume DoM space and inodes unnecessarily
  - Elevated DoM utilization caused by small directories and empty files accumulating multiple extended attributes (LU-20010)
- These issues can be amplified in DNEv1 configurations and may result in unbalanced MDT consumption.
- Appropriate ZFS tunables such as recordsize, dnodesize, and redundant_metadata can mitigate the impact, but remediation may still require file migration.

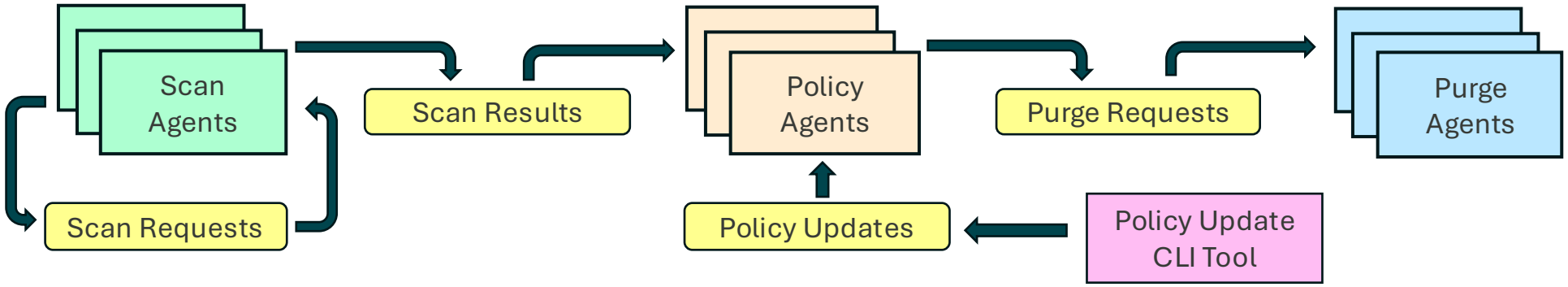
Orion Purge Policy

To keep filesystem usage at a maintainable level, Orion has a 90-day purge policy for “unused” files.

- In this context, unused files are those that have atime, mtime, and ctime values older than 90 days
- Purge policy only affects files. Empty directories are left intact since they might be part of a hierarchy used by a future job’s workflow.

Purging is handled using the PoliMOR policy engine software

- Multiple agents running across six Orion utility nodes share information using NATS message queues
- See previous LUG talks for more info about PoliMOR



PoliMOR in Practice

PoliMOR scan agents use “lazy stat” when gathering file metadata to reduce the impact on the filesystem, but the purge agents perform a full stat on the file before purging as a safety check.

- If the timestamps from the lazy stat and full stat do not match, the file is not purged out of caution
- Timestamp mismatches were assumed to be rare, but in practice they happen more often than expected
- File restriping is one of the biggest causes since it creates a mismatch in ctime info
- But we have also seen old files (not restriped) that have mismatches in atime and mtime as well

Timestamp mismatches prevent purging of files that are older than the purge policy threshold.

- Some of these files get removed on subsequent purge passes as a result of cached stat metadata
- A second PoliMOR instance is run on repurposed Orion routers to perform multiple sweeps on a project in an effort to purge the files with timestamp mismatches
- Looking into ways to minimize these mismatches
- File restriping is a key tool in reducing performance tier usage, but it also could delay the purging of files

Handling Performance Tier Usage on Orion

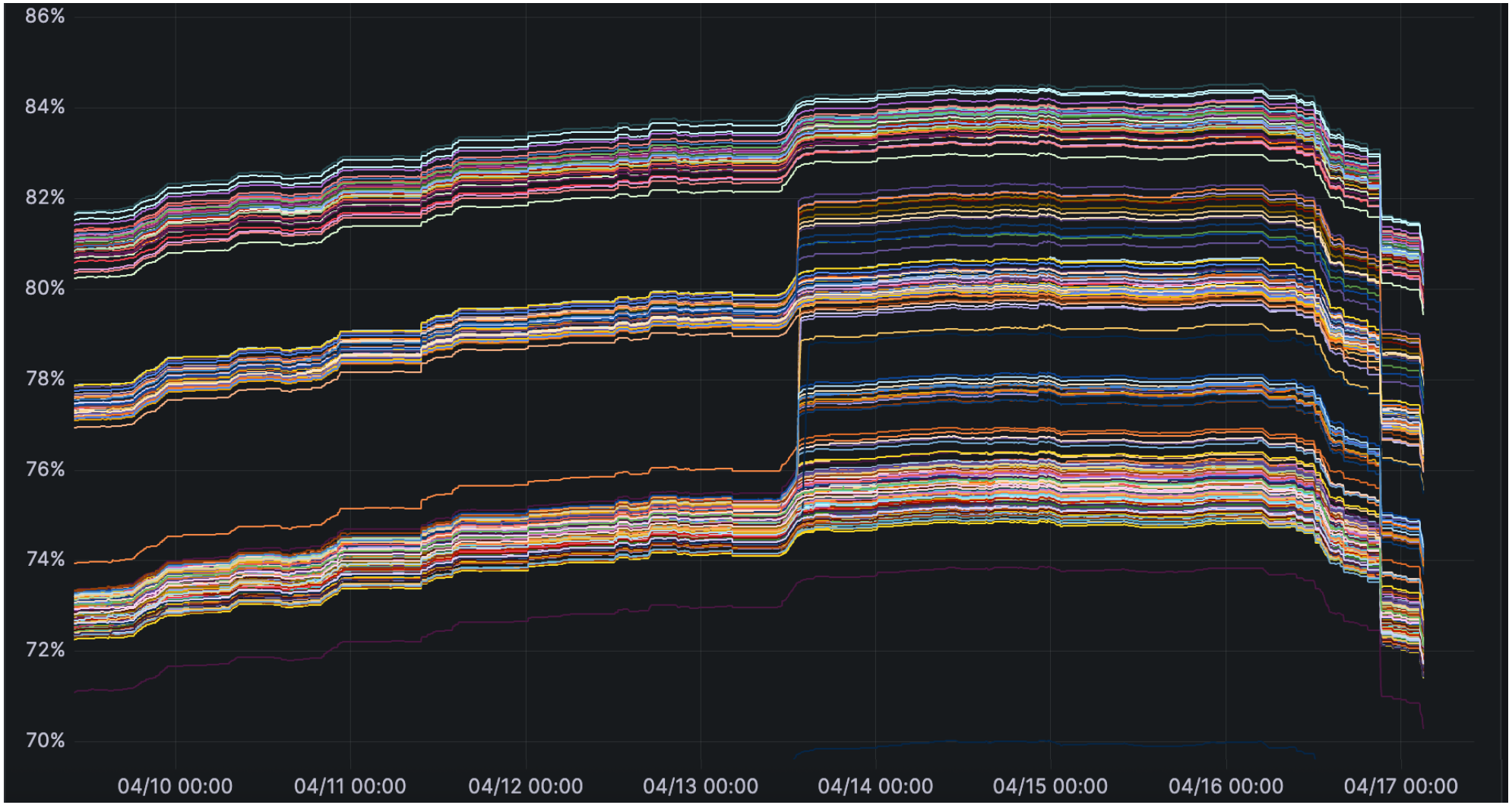
Current Process

1. Orion Performance OST Utilization >85% Nagios alert
2. Identify likely project with Perf OST project utilization script (generally look at average file size)
3. Check project quota history in RATS for the past 90 days and greater
    1. If project is less than 90 days old, then run candidate restripe script (accounts for SSF) and directory stripe default script
4. If project is greater than 90 days old check purge exemptions
5. If project is not purge exempt, then kick off manual PoliMOR purge on separate instance until most old files are cleared out
6. Once project is manually purged, check for restripe candidate files and directory stripes

Improvements

1. Switch PoliMOR over to a standard stat
2. Update restripe script to force old file times possibly with LL_IOC_FUTIMES_3

Handling Performance Tier Usage



Summary

- Multi-tier storage has been beneficial to our users, but there has been a learning curve associated with it (both for users and admins)
- OST pools, DNE, PFL, etc. are important tools for managing data placement and controlling Lustre usage
    - They might not solve every problem, but they have made life easier for admins in many respects
- Purging large filesystems is a time-consuming task and we are always looking for ways to improve it
    - PoliMOR is useful but not perfect
- Even with purge policies in place, keeping performance tier usage in check is an on-going battle

Acknowledgments

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Program. This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725.

Thank You

Questions?