Hot Pools

Alex Zhuravlev

Sebastien Buisson

2022-09-27

ddn

What are Hot Pools?

An automatic tiering system for Lustre

Managing data placement between standard and fast Lustre components

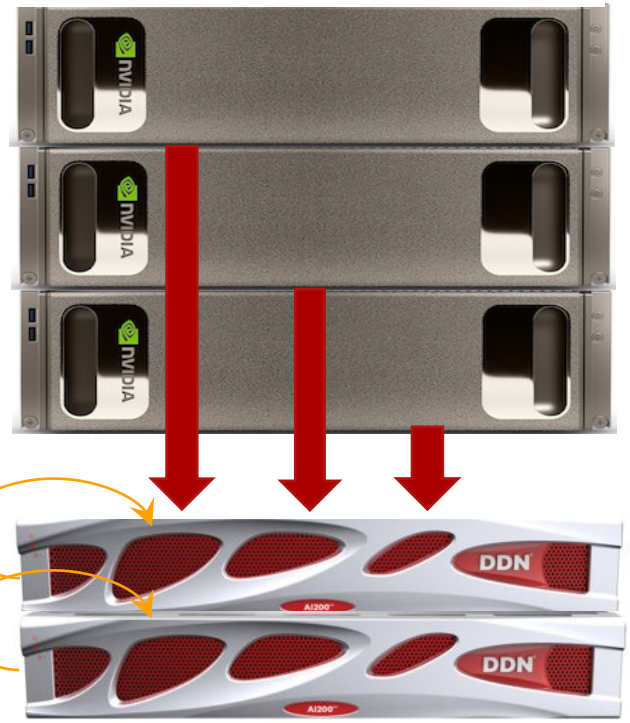
Specific to DDN EXAScaler for some parts

What are Hot Pools?

- Automatically manages fast Flash OST pools
- Allows workloads to automatically/transparently utilize Flash tiers
  - No special programming or functional calls needed by the application
- Reduces overall Disk storage system load
- Off-load high I/O demanding processes to Flash tiers
- Manages Flash tiers to avoid running out of space
- Automatically spills from Flash to Disk if needed

How Hot Pools work

Client writes/reads directly to/from flash



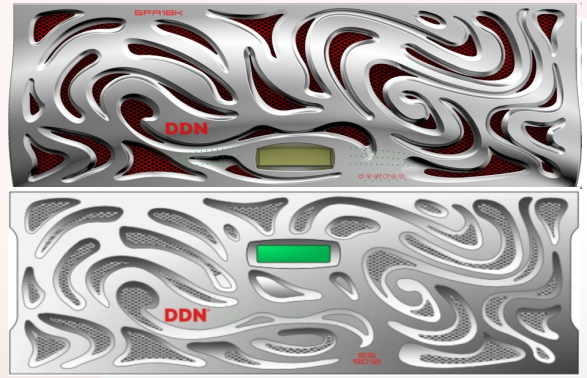
AI400 NVMe MDTs/OSTs
Parallel Scan to manage space

Stratagem Optimized Scan Engine
(one per MDT and flash OST)

Old and unused files mirrored to HDD OST pool

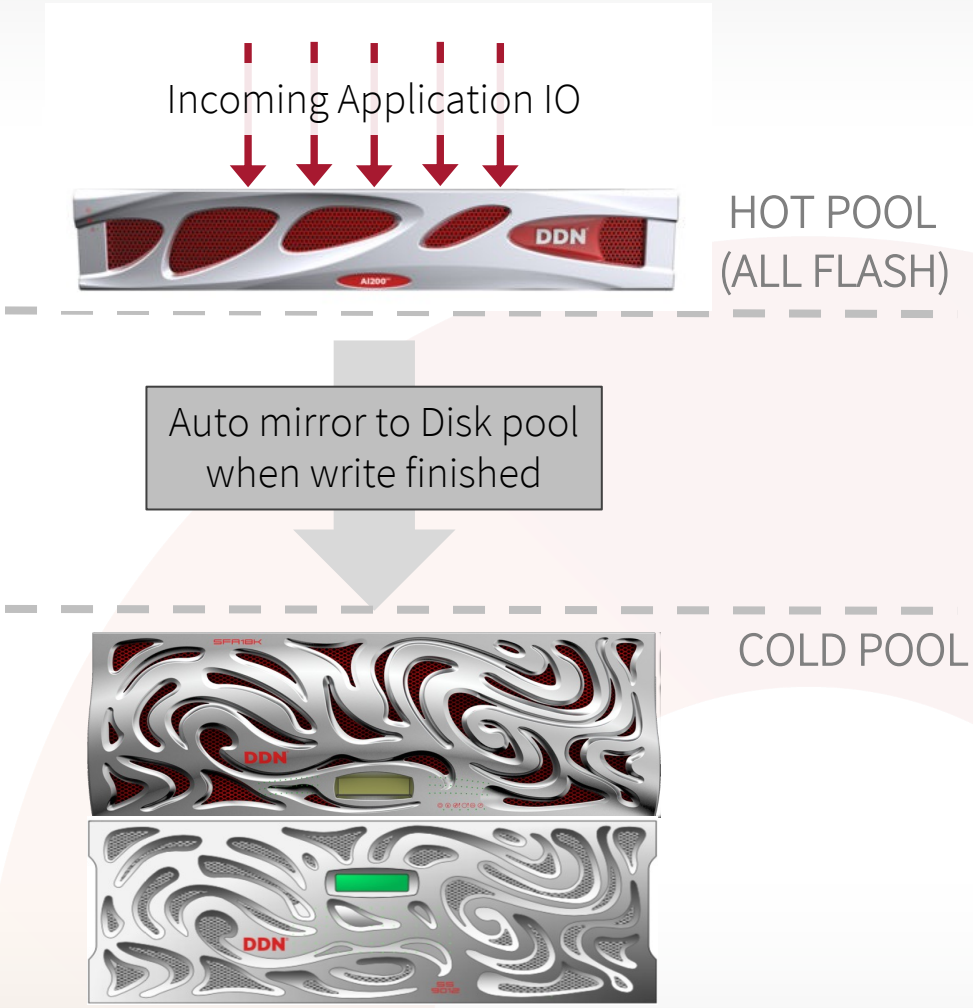
Mirror back to NVMe pool by job/request/policy

ES18K HDD Lustre OSTs



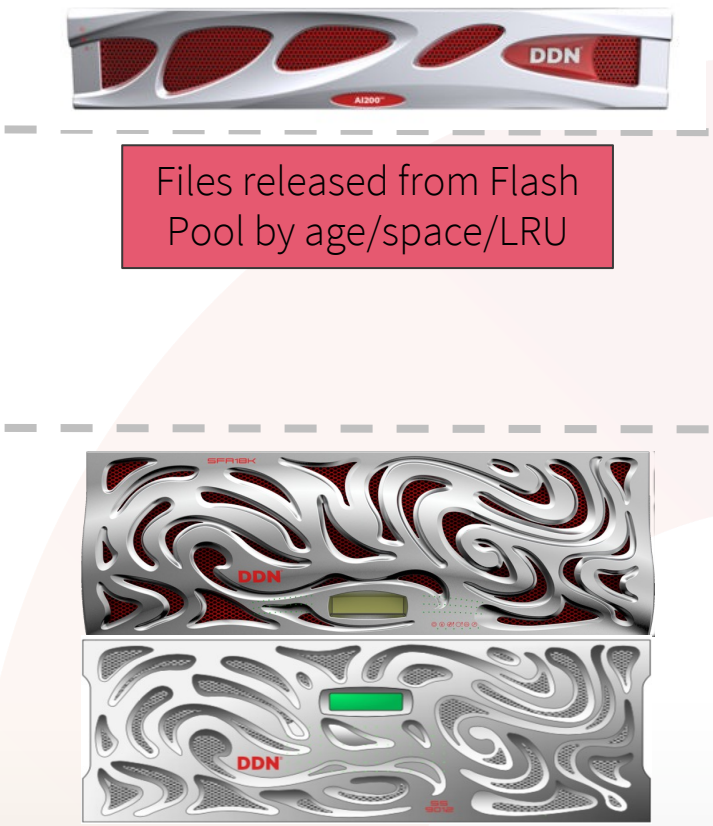
At Scale Auto-Tiering

- PFL layout writes files to Flash pool
  - Large files written directly to Disk pool
- **`lamigo` actively *mirrors* files from Flash to Disk pool**
  - One **`lamigo`** daemon per MDT
  - Start with full MDT scan for existing files
  - Follow Changelog for new/modified files
  - Wait until new files are inactive
  - Uses FLR mirroring to OSTs in Disk pool



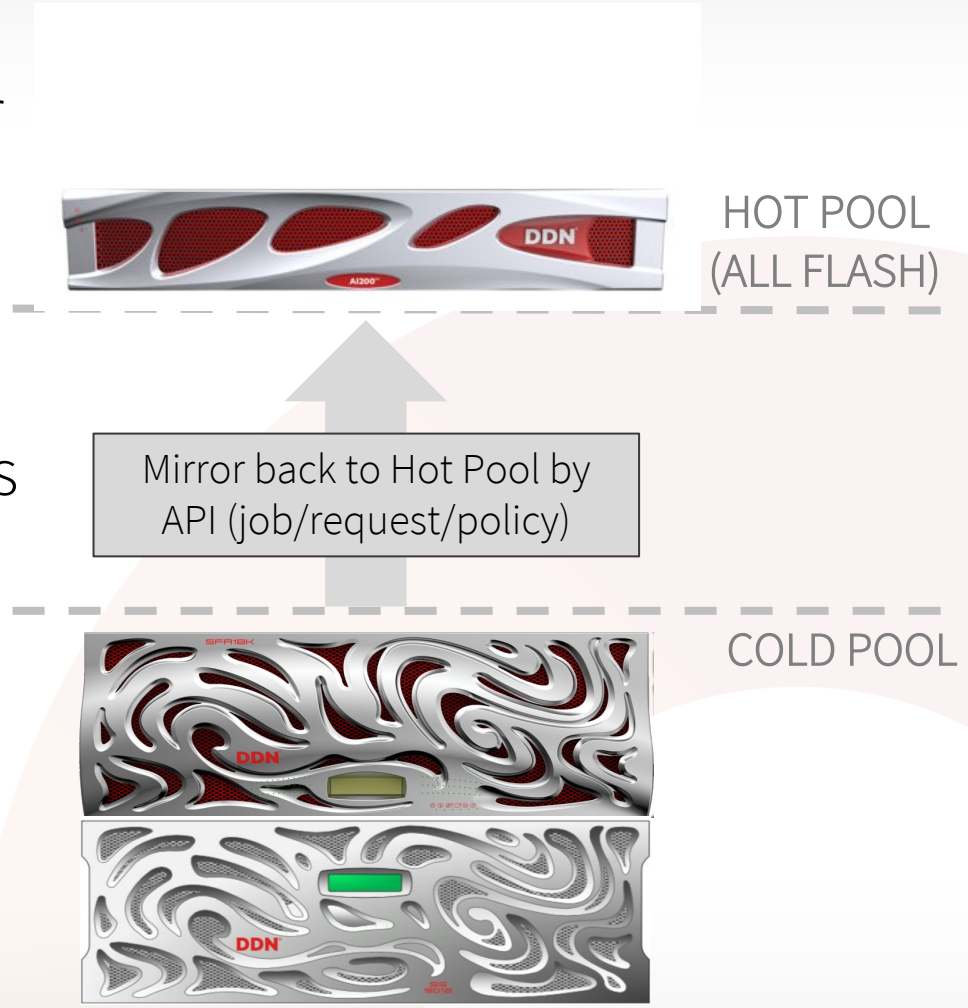
At Scale Auto-Tiering

- **`lpurge` monitors flash OSTs to purge Flash mirrors of old, least accessed files**
  - One **`lpurge`** daemon per Flash OST
  - Periodic full OST scans for unused files
  - Maintains list of oldest files in memory
  - Purges files from list only when space needed
  - Maximizes use of Flash OSTs, minimizes copies
  - Releases Flash space quickly if needed



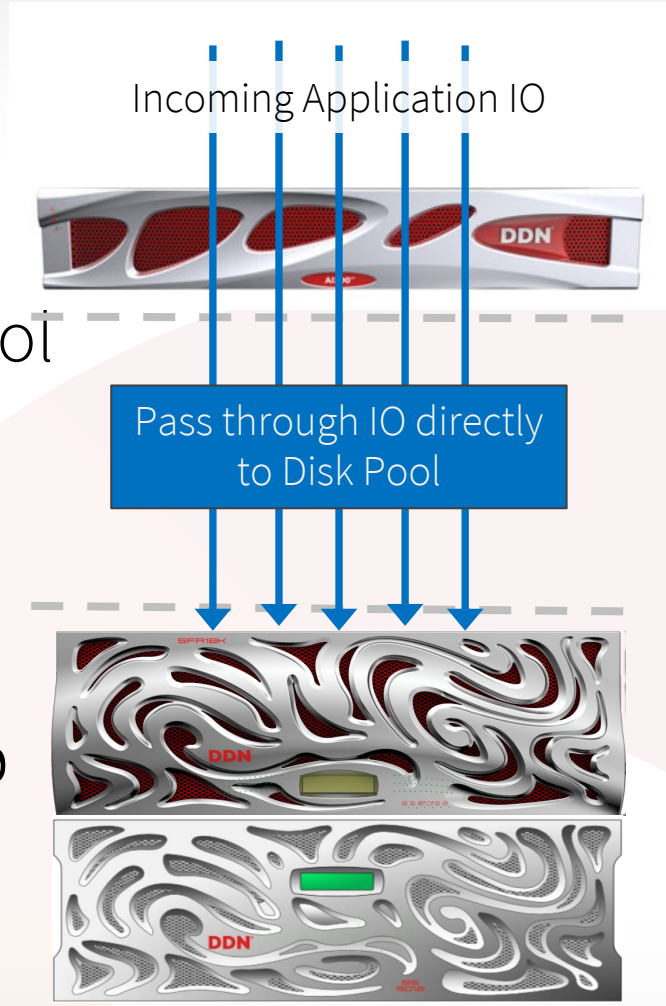
Active I/O Tracking

- OSTs logs all I/O into a circular in-memory buffer
  - Low overhead, lockless from kernel to userspace
- **`lamigo`** collects and aggregates I/O logs per file
  - Only fraction of data is read by server
  - Only most frequently accessed files are updated
- History table maintained to find “hot”/“cold” files
  - File “heat” increases if file is repeatedly accessed
  - If file not accessed in interval, then heat goes down
- “hot” files not subject to mirror to “cold” pool
  - This saves I/O, network and CPU resources
- “hot” files can be rehydrated back to “hot” pool
  - Newer feature, disabled by default
  - Need to collect data from field to improve policy



Pool Spilling to Avoid Full Hot Pool

- For any OST pool a “spill” pool can be configured
  - If Flash pool continually filled faster than it drains
- If *new* file layout on full pool, objects created on spill pool
  - Direct writes to spill pool avoids pressure on full pool
  - Inherently throttles writes rather than running out of space
- Each pool has “full” threshold and target spill pool
  - `lctl set_param lod.*.pool.pool1.spill_target=pool2`
  - `lctl set_param lod.*.pool.pool1.spill_threshold_pct=90`
- Maximum 10 levels of spill pools following this logic
  - pool1 -> pool2 -> pool3 -> … pool10, to avoid infinite loops



Policies and Rules

- **`lamigo`** supports "rules"
- Using rules user can control what files to replicate/migrate/resync
- Syntax is shared with **`lipe_find`**:

```
"expression": "projid > 1000000 && projid < 2000000", "action": "skip",
"expression": "blocks < 1M || group == vipusers", "action": "skip"
```

- Attributes supported:
  - Regular POSIX attributes: size, blocks, [amc]time, uid, gid, filename/extension
  - Lustre attributes: HSM, LSoM
  - Custom xattrs: **`name: value`** interpreted as text

Changes to Improve File Mirroring

- **`lamigo`** can do full filesystem scan
  - Useful in case Hot Pools is enabled on a pre-existing filesystem
    - Changelogs have no records for pre-existing files
  - Can be used if Changelogs are lost (MDT space, age, errors)
  - Named Changelog users, per-user Changelog mask
    - Simplify automatic (re-)registration and minimize overhead

- Clients prefer file replica on non-rotational storage
  - OST reports itself as rotational (HDD) or non-rotational (NVMe)
  - Clients prefer to read/write replica on non-rotational OSTs
  - No need to mark preferred replica on every replicated file
  - If present, **`prefer`** per-file layout flag takes precedence

ddn

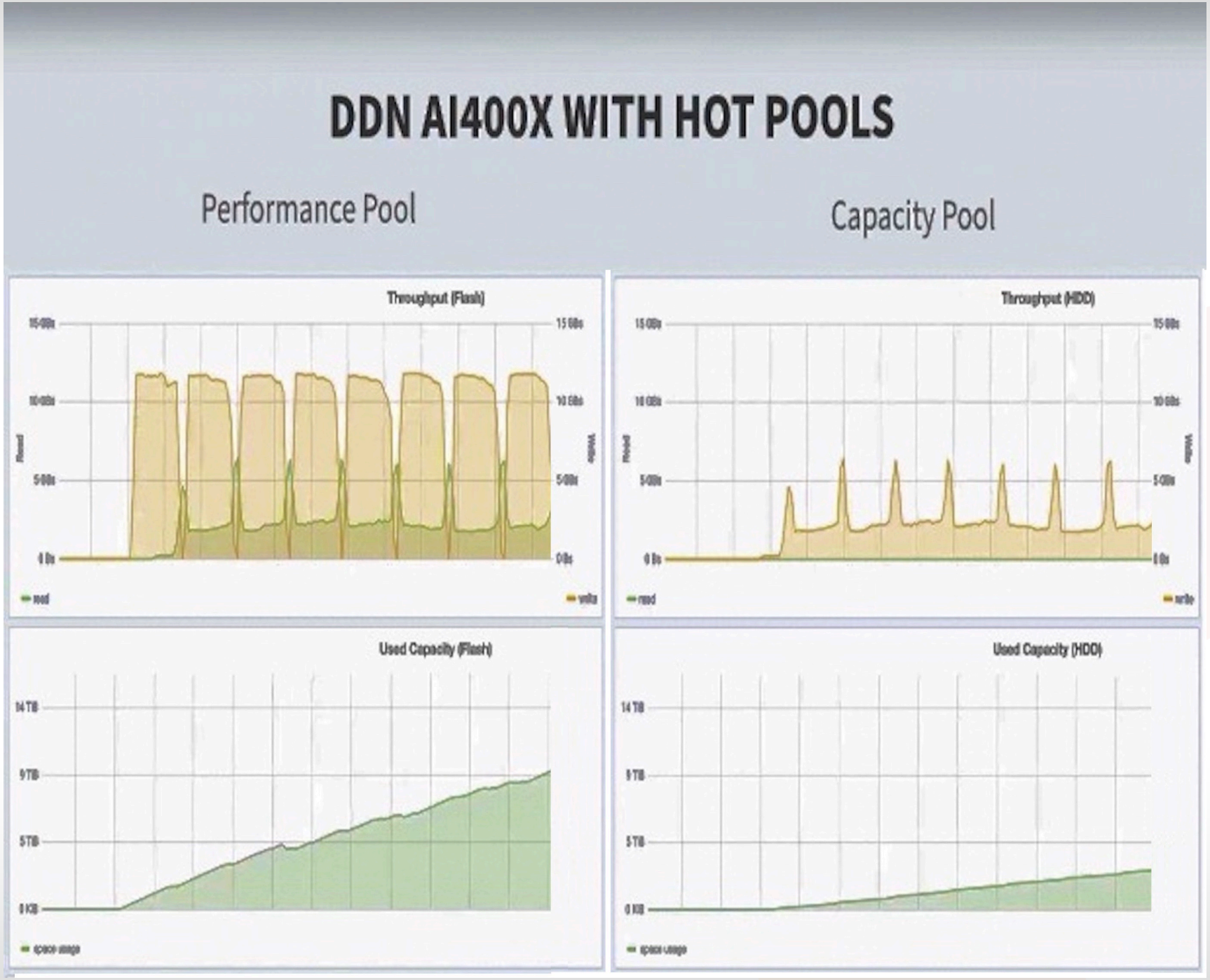
Testing

- 24x1.9TB - 1x5TB MDT, 4x8TB OSTs in **`nvme_pool`**
- 83x4TB 7.2k RPM HDD - 4x21TB OSTs in **`hdd_pool`**
- 2x R640 Dual 2.1GHz 16-core Cascade Lake clients

- Local IOR FPP to NVMe write ~15GB/s
- Local IOR FPP to HDD write ~10GB/s

- Application:
  - 2 client 64-thread IOR writes in loop

Testing



Use Cases

- Burst Buffer
  - HPC applications
  - Fast checkpoint to SSD storage, then back to computation
  - Hot Pools migrates checkpoint data to HDD in background
- Increased virtual Flash capacity
  - Applications mostly access flash
  - Older files often unused after days
- Backup
  - Important data can be backed up on more economical storage
  - Multiple copies supported

What is available in the Lustre Community release

- Pool spilling
  - When pool nearly full, new files redirect to other pool - LU-15011 – Lustre 2.15
- Named Changelog user and per-user mask
  - Simplify Changelog user management - LU-13055 – Lustre 2.15
- Rotational status
  - OSTs report rotational/non-rotational storage - LU-11963 – Lustre 2.13
  - Lustre client prefers non-rotational OSTs for I/O - LU-14996 – Lustre 2.15
- I/O access log
  - OSTs generate access log for I/O - LU-13238 – Lustre 2.14
  - External tool can use access logs to generate stats, analyze performance

ddn