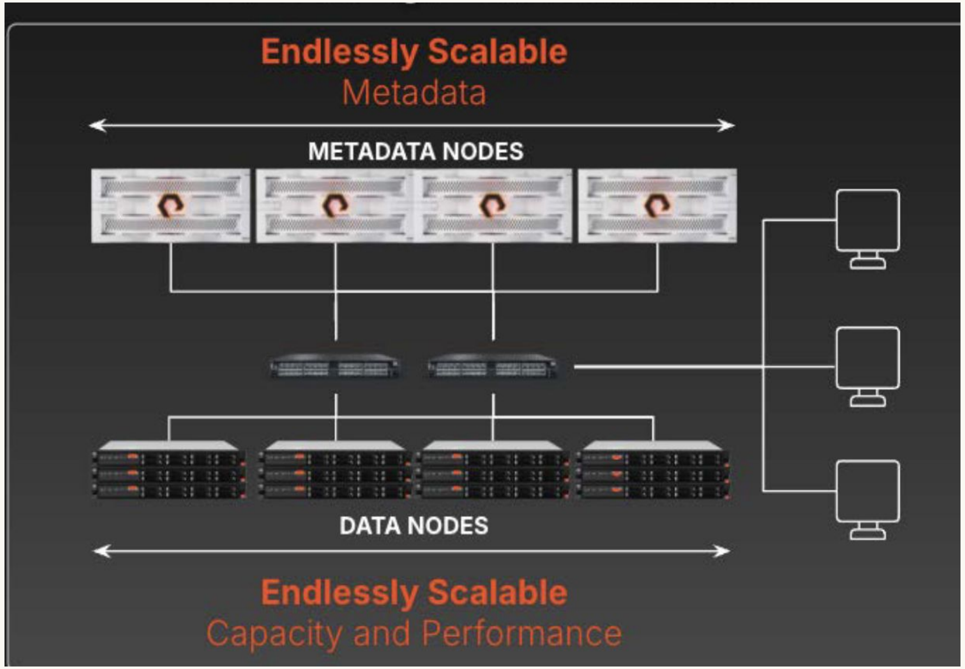
pNFS and FlashBlade//EXA™

Jonathan Curley (on behalf of many!)
Pure Storage
September 2025

©2025 Pure Storage

Introduction

- pNFS has renewed focus due to AI.
- Split metadata from data. Scale read & write performance as high as you want.
- Seems like a good fit for AI training.
  - Reads of individual files.
  - Write large checkpoints.
  - Take advantage of the scale?



©2025 Pure Storage

Journey Overview

- Kicking the tires with File Layouts.
- Switch to Flexible File Layouts.
- Debugging live locks and EAGAIN.
- Implementing Flexible File Layout striping.
- More EAGAIN fixes.
- pNFS & HA networking observations.

©2025 Pure Storage

Working with File Layouts

- Striping patterns are devices. Representing arbitrary erasure coding layouts is not feasible.
- Requires tight coupling. Must run software on devices. Complicates white box deployment model.
- What about Flex File Layouts?

©2025 Pure Storage

Working with Flexible File Layouts

- Layout error reporting is very useful.
- Mirroring is interesting.
  - Kicked the tires a bit.
  - Big write perf hit.
  - Failure scenarios require substantial implementation investment. Resilvering, consistency, etc.
- Live locks are a nightmare.
  - Stateid seqid to the rescue?
  - Client improvements?

©2025 Pure Storage

Error reporting to the Metadata Server (MDS)

- Data node setup to randomly return a fatal EIO error.
- Client is expected to return the layout to the MDS and report the error.
- Client issues get layout to refresh a new layout.

Super useful workflow for detecting errors at the client and reporting them to the MDS for handling. Ideally this allows the MDS the opportunity to correct issues before responding to the client with a new layout.

©2025 Pure Storage

Layout return livelock

The trouble starts with ff_layout_async_handle_error...

- All roads lead to pnfs_mark_matching_lsegs_return
- Write → pnfs_update_layout → pnfs_mark_matching_lsegs_return → EAGAIN → ...
- Write → pnfs_update_layout → nfs4_proc_layoutget → gets invalidated → EAGAIN → ...

Saw more than 15 similar backtraces with bpf trace...

©2025 Pure Storage

Livelock fix?

- Implement stateids + stateid seqnum to match RFC & client expectations.
- Carefully inspect layout return response and make sure you're handling stateids properly, especially this behavior:

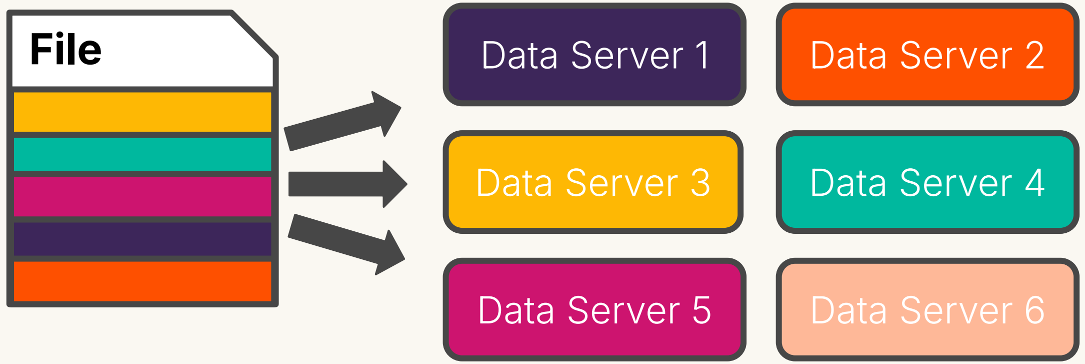
```
If the last byte of any layout for the current file, client ID, and layout type
is being returned and there are no remaining pending CB_LAYOUTRECALL operations
for which a LAYOUTRETURN operation must be done, lrs_present MUST be FALSE, and
no stateid will be returned.
```

Once it's all fixed, the client should establish a layout return barrier that invalidates only the layouts from before the layout return event.

Lingering livelock issues?

- Some client versions still tend to livelock in this scenario.
- Reasons for the continued issues are unclear.
- Git log shows several fixes. Bisecting points to diffs that shouldn’t make a difference.
- More on this later...

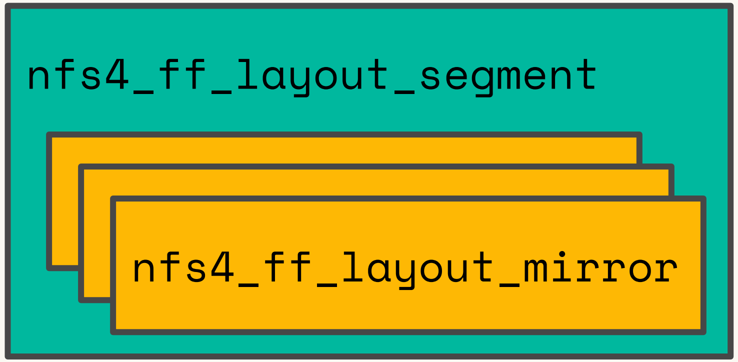
Announcing pNFS Flexfile striping support!



- Linux kernel patches posted to LKML for upstream merge.
- Follows patterns of file layout striping support.
- Multiplicative interaction with mirror support.

Flexfile striping support - Before

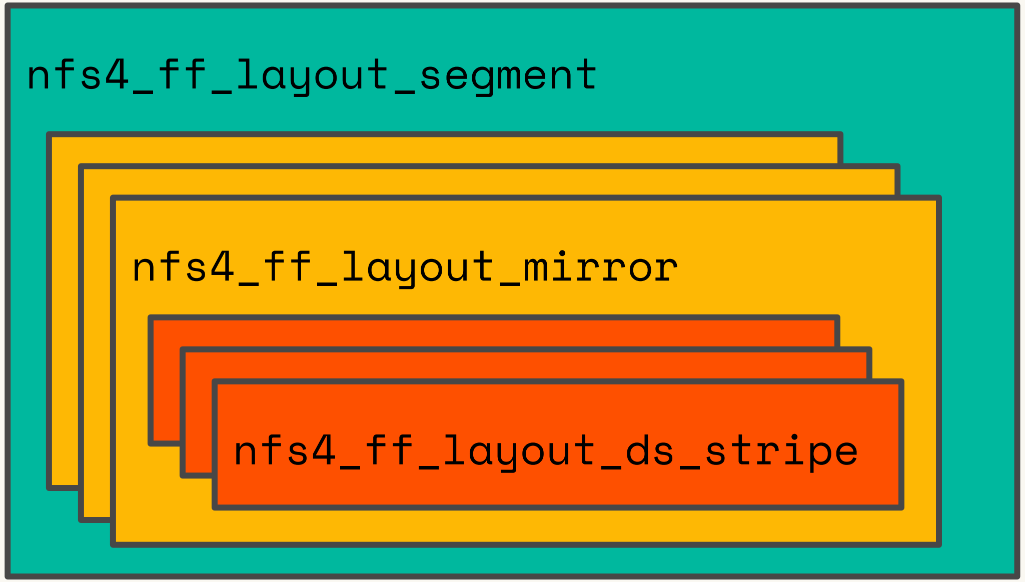
- Layout segments consist of a list of mirrors.
- Mirrors contain information about the layout segment from the RFC - FH, creds, DS device link.
- The Flexfile read, write and commit paths use this data structure to lookup information required to form and send RPCs to the data nodes.



©2025 Pure Storage

Flexfile striping support - After

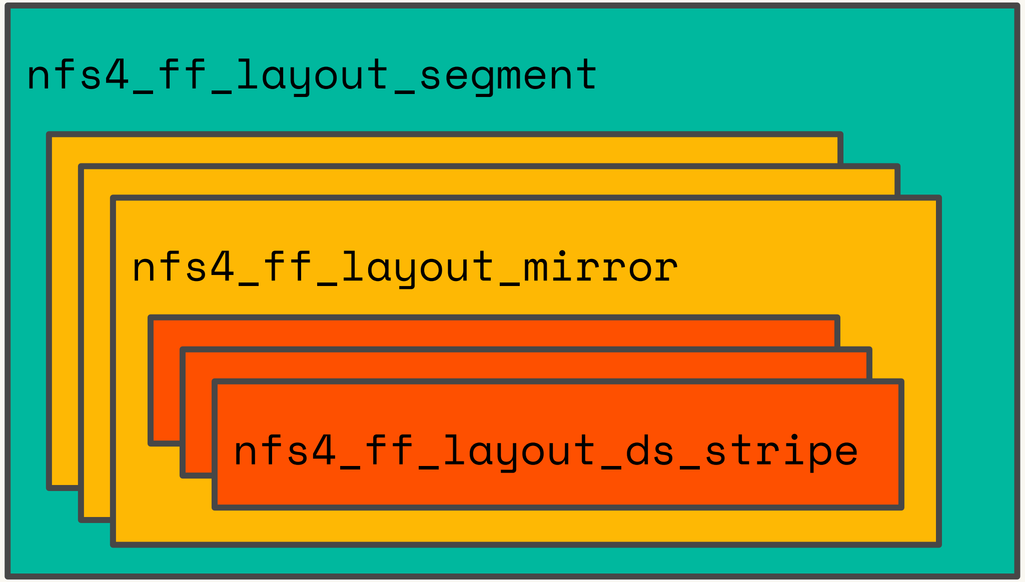
- Mirror has an additional array within it to represent the stripes.
- Most code simply works by transforming the reference from one dimension to 2.
- Some paths need to loop over the stripes.



©2025 Pure Storage

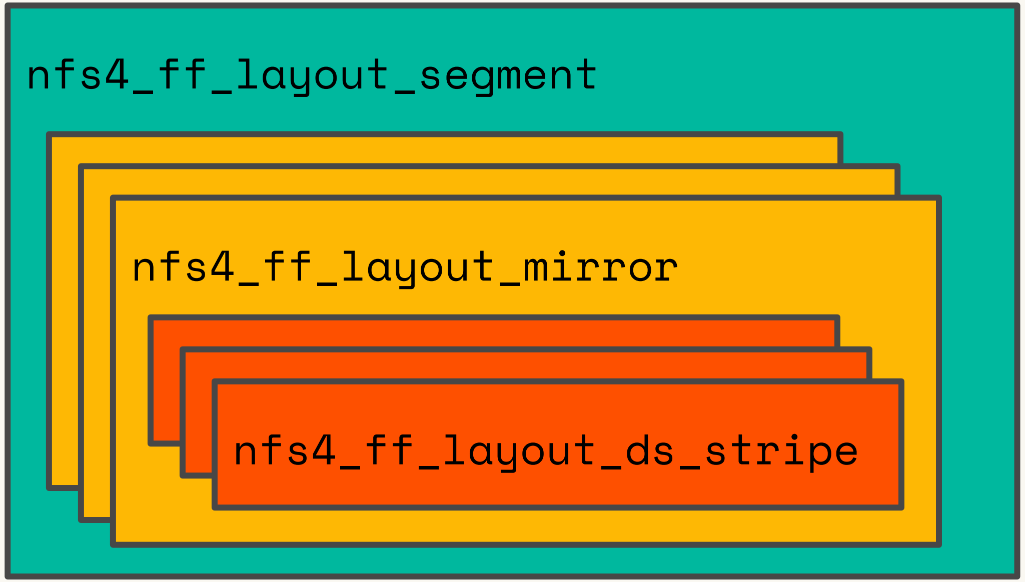
Flexfile striping support - Challenges

- Layoutstats reporting goes multiplicative (mirrors * stripes).
- Could end up being quite a bit of overhead to track all that for each file.



Flexfile striping support - EAGAIN Challenges

- When layouts need to be expanded to add an additional stripe the kernel returns EAGAIN to the application to handle the interrupt.
- Some applications don't handle EAGAIN...



©2025 Pure Storage

Merging layouts considered harmful?

The layout merge path has this block of code:

```
if (do_merge(new_lseg, old_lseg)) {
    mark_lseg_invalid(old_lseg, free_me);
    continue;
}
```

If we decide to merge a layout segment, we mark the currently cached segment invalid. This causes syscalls that already reference this segment to fail with EAGAIN, trigger disconnects, etc.

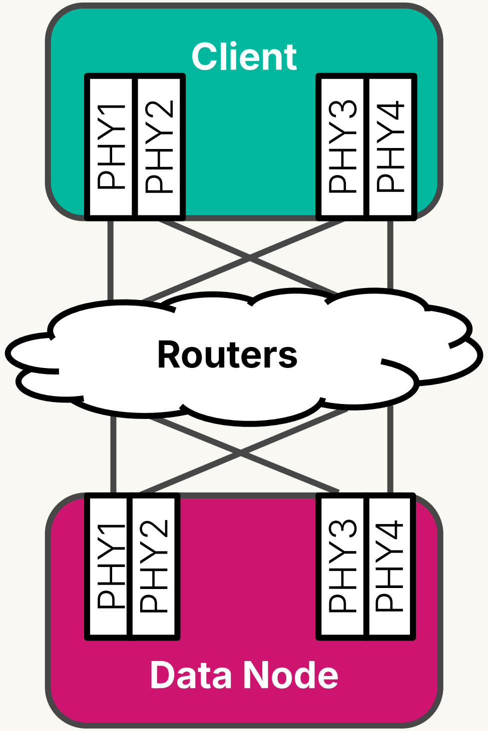
- A bug fix in the merge logic makes this case much less common when expanding the number of stripes in a layout.

Major source of EAGAIN & disconnects fixed!

©2025 Pure Storage

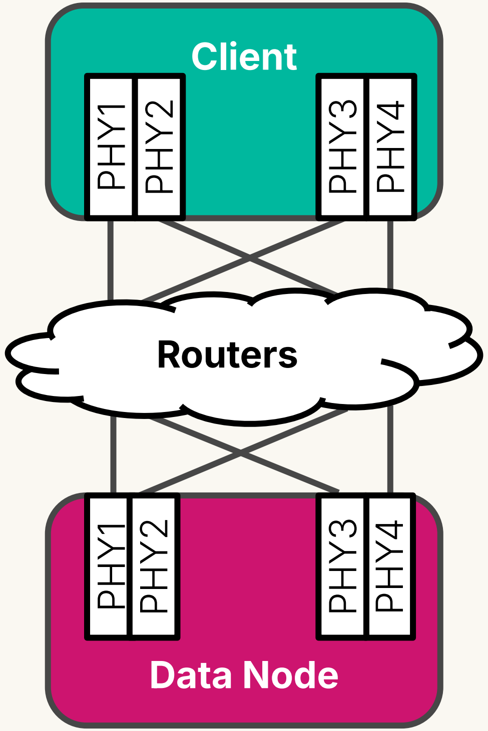
pNFS Networking Ideal Setup

1. Can't rely on bonding being there because it's often incompatible with ROCE.
2. Data node may have different IPs associated with each PHY.
3. Want to nconnect all client PHY to all data node PHY.
4. Router layer generally wants connection entropy to take advantage of redundant routes.
5. If we lose connectivity due between any particular set of PHYs we want to rebalance immediately.
6. And if connectivity gets fixed we want to detect that and rebalance connections.



pNFS HA challenges

- RDMA considerations
  - Generally can't use bonding except for some dual ported NICs.
- Protocol supports returning a list of different IPs to client (multipathing).
- ... but client only nconnects to one of the IPs.
  - V3 only connects first IP (per spec? reply cache?)
  - V4 nconnects first ip, one connection for rest.
- Balancing IO across network interfaces?
- Failover and failback, timeouts
  - Userspace management via /sys interface?



©2025 Pure Storage

Conclusion

- pNFS read & write performance scales fantastically as expected.
- Livelocks, EAGAIN and other stability issues continue to be a challenge with complex workloads.
- HA networking behavior is a solid baseline with room for improvement.

©2025 Pure Storage

Questions?

©2025 Pure Storage