pNFS Lattice: From Concept to Lattice

pNFS Past, Present, and Future: What’s all the excitement about? The evolution of scale-out metadata and the next generation of open storage

Las Alamos National Laboratories and PEAK:AIO

pNFS Background: Building on NFS and pNFS, beginnings and improvements over decades

Original pNFS vision, University of Michigan, 2002 to 2009 (DOE/DOD/NSF funded)

- Eliminate FS client for the PFS's (Lustre, Panasas, GPFS, PVFS)
- Recallable File maps (where, when, replicated/erasured, locality etc.)
- Statefulness
- Compound RPC (less chat/more intents)
- Data servers can be just NFSV3 servers (standard part of Linux), as well as object and block
- PNFS metadata server becomes NFSv4.2-> metadata server to enable parallel file service/etc.

Some things we didn't think of have been added to NFS as well

- Re-export – nfs server can reexport files from another nfs server – cached copies/etc.
- Multi stream from one client node – eat all BW from client to server
- NFS RDMA using either OFED or TCP or other – efficient movement
- NFS Local-IO, Full Path-Direct-IO, etc.
- Referrals

pNFS Movers

PEAK AIO

PURESTORAGE

HAMMERSPACE

NetApp

VDURA

Los Alamos NATIONAL LABORATORY

UNIVERSITY OF MICHIGAN

Test and Evaluation Resources

- LANL and CMU/LANL testbeds available

Coming Soon-ish

- Multi-node/Multi-writers into one file
- Client Erasure (with N to 1 writers)

PNFS What Is Missing?

PNFS file solutions are open source/Linux distro provided for clients and data servers, but the metadata servers are/were all Proprietary (standards based)

- This limits community engagement and innovation
- Initial exploration of open-source BSD Kernel based non scalable mds activity by Peak-AIO
- PEAK:AIO/LANL (pNFS-Lattice) Launching effort to develop an example Linux Open-Source User Space metadata and data service, to provide to community to enable innovation, Public GIT/Web presence available now!

Enabling metadata scalability (a multi-decadal problem)

- Concept – LANL
- Separate MDS service, catalog, shared state mgmt., policy (layout/etc.)
- Enables ephemeral sizing and native failure mgmt. of service, catalog, etc.
- Core service enables plug in for catalog, policy, etc. enabling easy innovation
- Shared memory tech potential – SK hynix, Micron, LANL, Unifabrix, etc.
- Work in CXL and similar multi-node coherence (info in later session)
- Design, Initial Capability, Initial results – PEAK:AIO

pNFS Movers

pNFS Lattice:

An Open-Source, Scale-Out Metadata Server

Las Alamos National Laboratories and PEAK:AIO

A STANDARD CLIENT DEMANDS A STANDARD SERVER.

NFS 4.2 + FLEX FILES

Created a practical, vendor-usable framework.

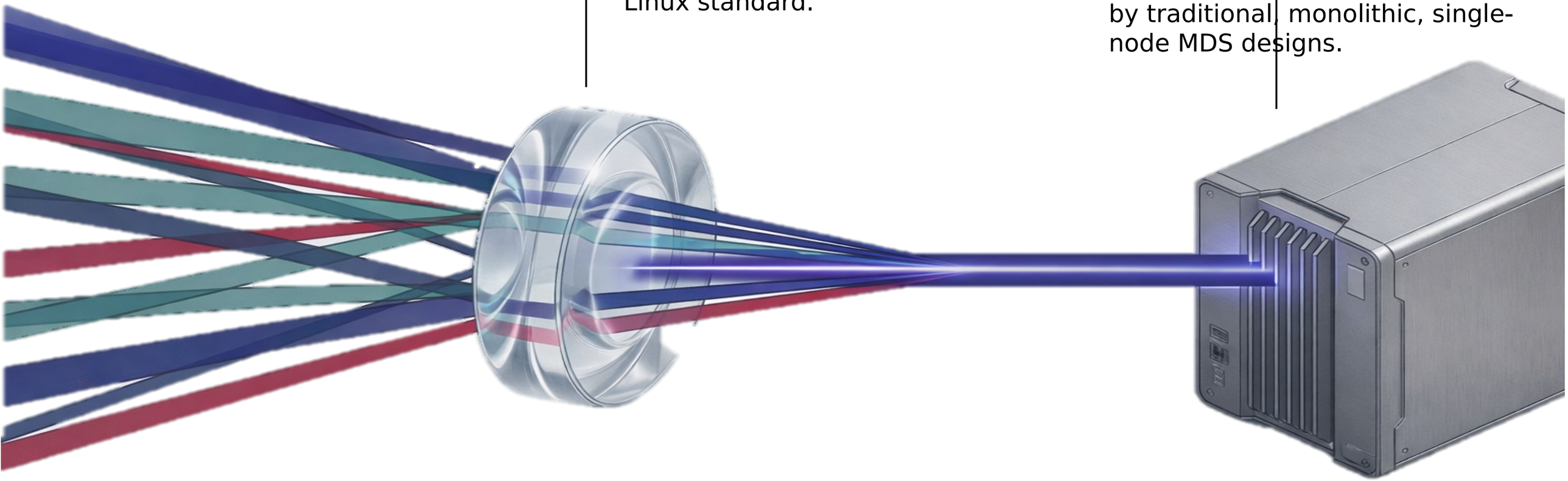
FLEX FILES

CLIENT-SIDE STABILIZATION

Reduced interoperability friction; created a de facto Linux standard.

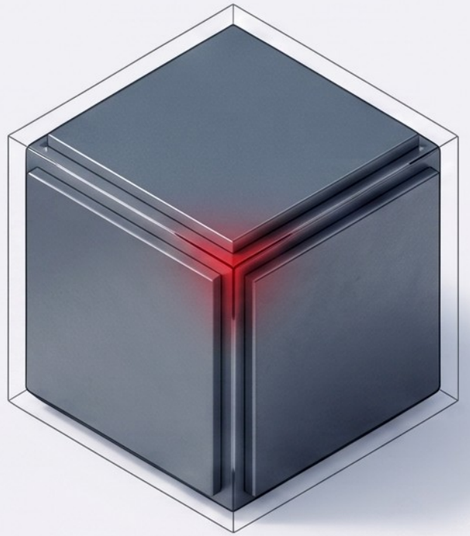
THE SERVER-SIDE GAP

While the client stabilized, the server-side remained constrained by traditional, monolithic, single-node MDS designs.



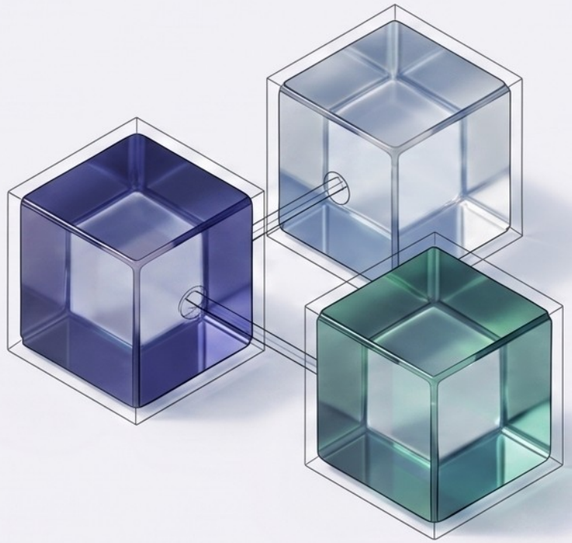
The Target: Ephemeral metadata fleets.

1. Traditional Monolith



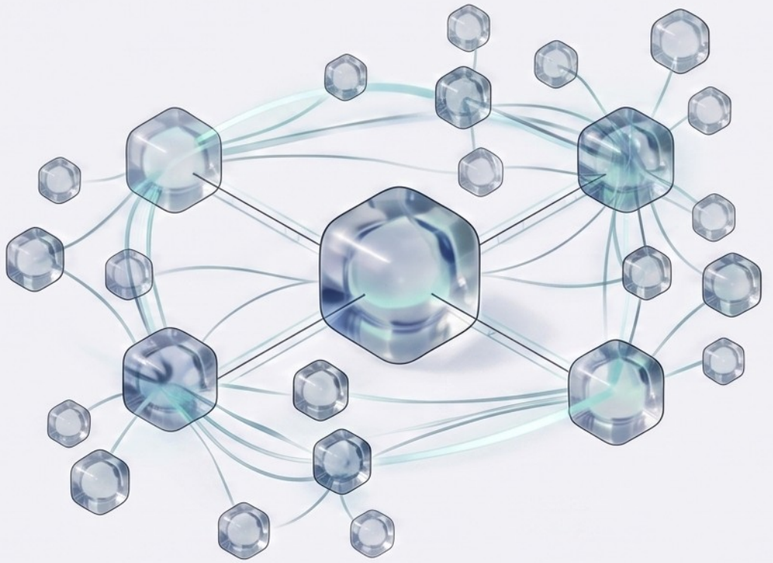
Fused state, execution, and authority. Static and fragile.

2. Phase 1 (Static Multi-Node)



Shared Catalog Authority (RonDB) + Sharded Local State Nodes are static but scale horizontally.

3 Future Lattice (Ephemeral Swarm)



Shared Catalog + Shared Protocol State Plane MDS instances become elastic, ephemeral participants that can safely join, fail, and takeover instantly.

Evolution

Lattice

Separating the Layers in User Space

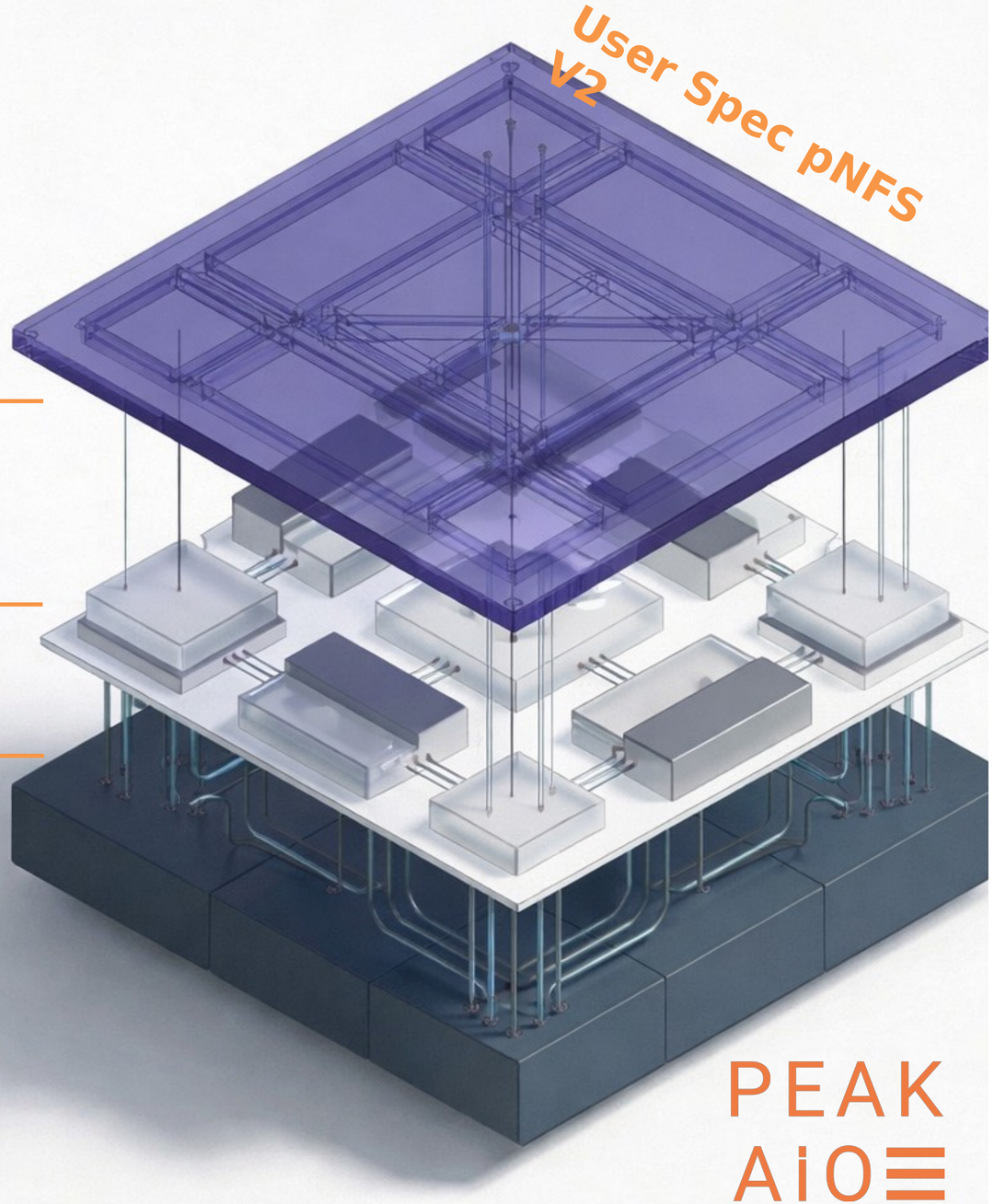
1. Protocol State Plane
Shared authority for NFSv4/pNFS client, session, replay, lease, stateid, layout, fencing, and failover state.

2. Lattice Core
Lattice core functions, NFS / pNFS Core, XDR, targets, and composable stations. Namespace operations, layout decisions, placement logic.

198/198 pass rate on pynfs with flex files.

3. MD Catalog Authority
The authoritative, versioned truth for namespace, identity, attributes, placement, and workflow state

Fully in-mem and persistent KVS

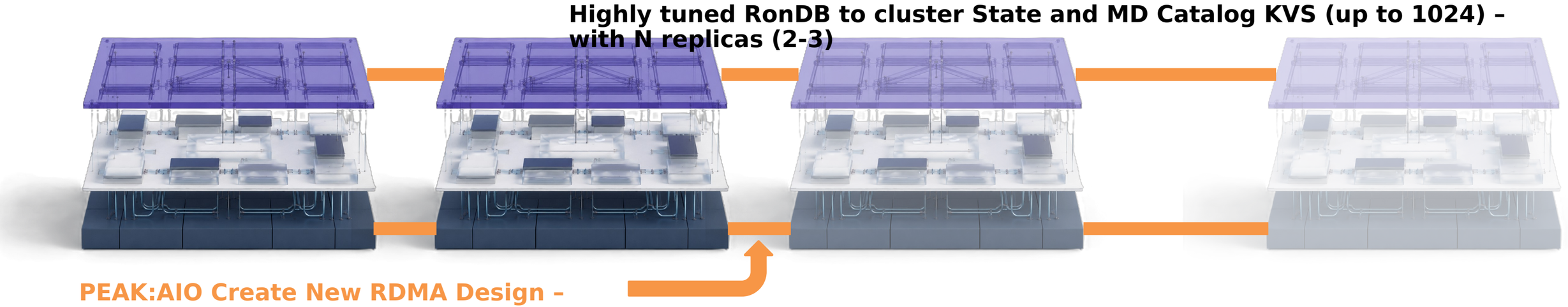


PEAK AIO

Lattice - Elastic Metadata for pNFS

PEAK AIO

The metadata server becomes a scalable service layer, not a fixed bottleneck



PEAK:AIO Create New RDMA Design – Open Source

Architectural shift

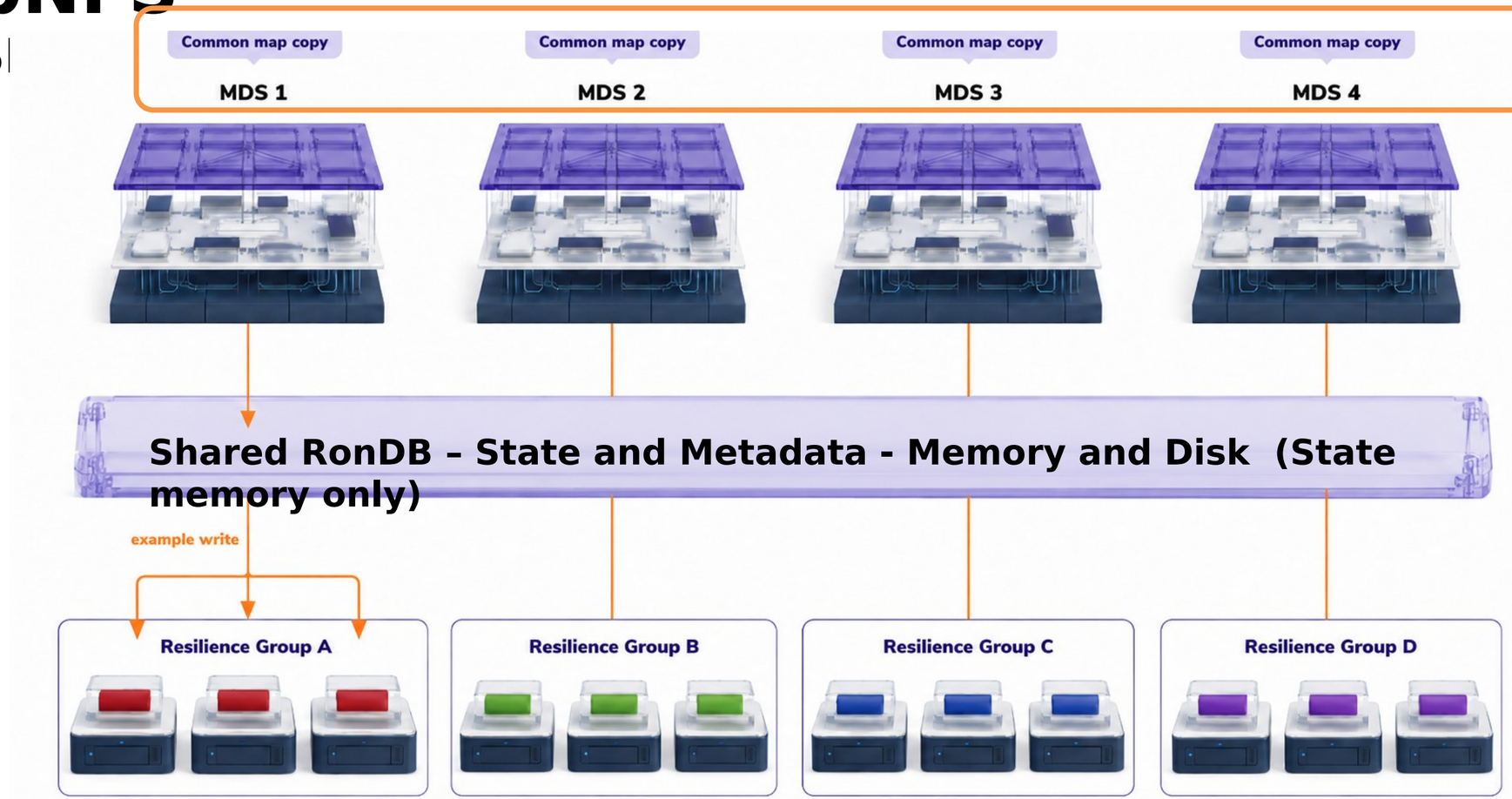
- State and catalog moved to a clustered KVS
- Focus on performance (could state be shared mem?)
- Reducing latency:
  - Standalone KVS ~30uS (**no replication**)
  - Lattice with PEAK:AIO RDMA ~50uS (**replicated**)

What does this mean?

- Mass scale MDS – On-demand
- Changes the concept of HA
- Range of sharding options
  - Standard Referral, Shared tree across MDSs,
  - eBGP round-robin

Lattice - Elastic Metadata for pNFS

PEAK AIO



Every MDS can resolve the global namespace through the cluster map

1. Write
Metadata/state is written once, then sharded and replicated across the resilience group.

2. Read
Any MDS can resolve the request using the shared map. Local or replicated memory serves the fast path.

3. Recovery
If an MDS fails, another MDS can continue from the shared catalogue and state layer.

State: memory resident for speed
Catalogue: memory resident with persistence for recovery

Replicated to i.e 3 Separate MDS Nodes

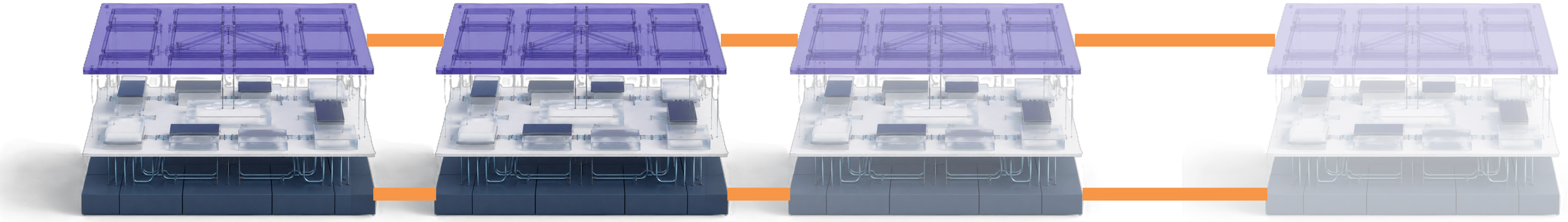
Lattice - Flexible Metadata

PEAK AIO

Sharding

N

A. Network Distribution
Traffic spread across MDS nodes using standard eBGP or load balancing



(B) Namespace Referral Sharding

- /foo → MDS1
- /project_2 → MDS2
- /project_2/images → MDS3

Clear ownership by path or project

(C) Shared Tree Sharding

- / can be served by multiple MDS nodes
- Clients see one namespace
- Different client groups can enter through different MDS IPs
- Load is spread without changing the visible tree

Lattice .

Performance

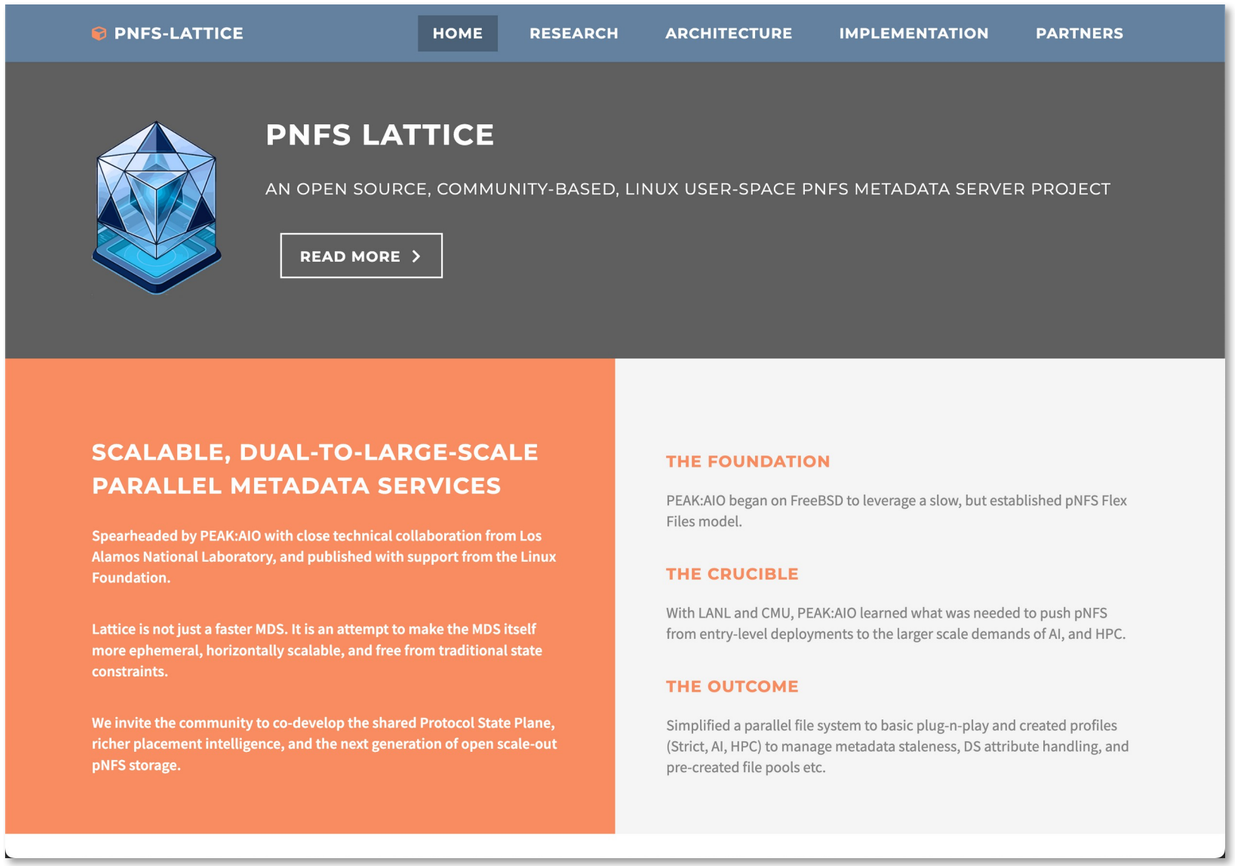
MDtest (Shared Directory)		LANL Ref	Linux-NFSD	Lattice
mdtest -F -C -T -r -n 16384 -d /mnt/mdtest/test_dir -N 16 -S	File creation	6,899	1,668	21,286
	File stat	26,175		118,992
	File removal	7,454		24,374
(testing and tuning in progress)				

Table 8.6 MDTEST Microbenchmark Crossroads (MB/s)

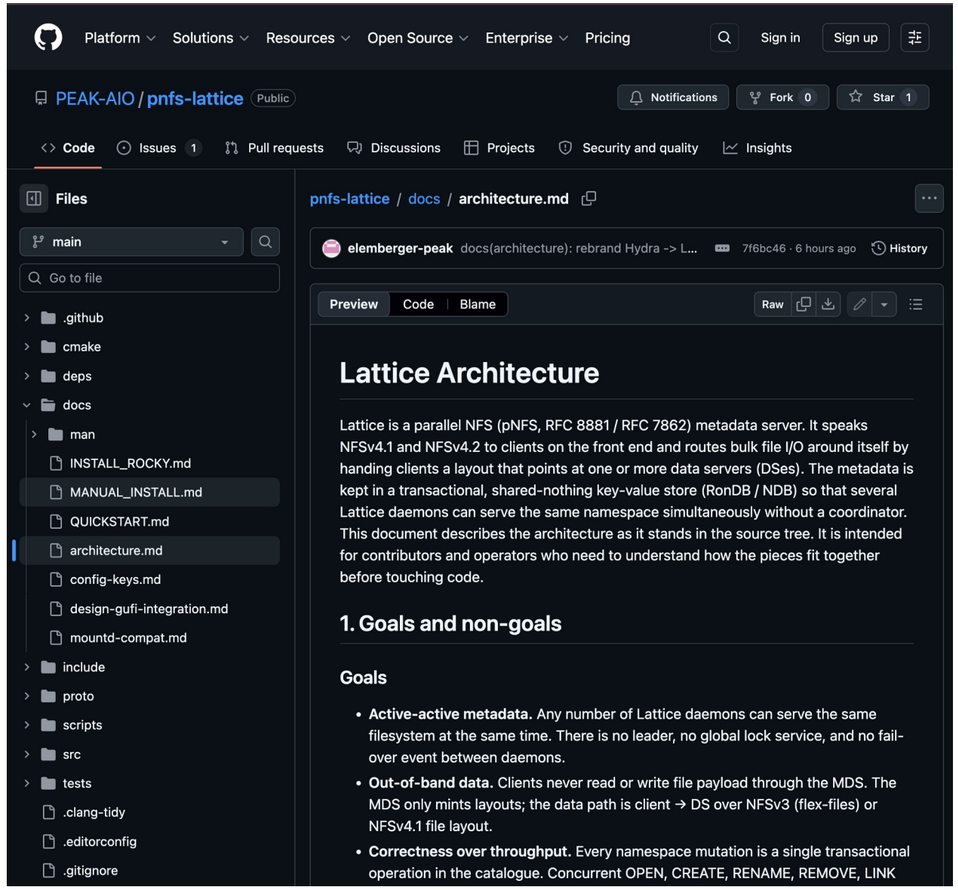
	Create	Stat	Remove
Single Process	5702	6773	8361
Optimal MPI processes single node	6899	26175	7454
Minimal MPI processes on multiple nodes w/peak results	79901	370968	79583
Extrapolated maximum possible MPI-level concurrency	79901	370968	79583

To Learn More …

https://pnfs-lattice.io/ or https://pnfs-lattice.org/



GitHub is Live!



PEAK
AIO

Thank You!