HAMMERSPACE

How to Adapt Legacy Infrastructure to Power GPU-Based AI/DL Workloads Using Parallel NFS v4.2

TECHNOLOGY OVERVIEW

By Floyd Christofferson

V. 1.0 - June 2024

Introduction

Organizations in every industry are grappling with how to realize the promise that artificial intelligence and deep learning (AI/DL) technologies can extract previously hidden value from unstructured data to improve their businesses. The key problem they face, however, is figuring out how to get the performance needed to feed GPU-based AI/DL workloads without copying their data to expensive new high-performance storage silos.

In this article you'll see how virtually every data center in the world today already contains the standards-based building blocks that are included within Linux to create a high-performance parallel file system with existing infrastructure using Parallel NFS v4.2 with Flex Files.

You'll see how pNFS v4.2 with Flex Files layouts provide a step change improvement for AI/DL workloads over legacy pNFS v4.1 with "files" layout implementations, as well as over other commercial and open source parallel file systems used in high-performance computing (HPC) data centers. And it does this based upon open standards in hybrid environments that may include multiple existing storage platforms from any vendor.

You'll learn how Hammerspace software uniquely leverages the parallelism of pNFS v4.2 to automate GPU data orchestration at extreme performance levels, achieving the linear scalability needed by Meta, for example, to feed 24,000 GPUs in its AI Research SuperCluster at eye-watering performance levels to power its Llama 2 & 3 large language models (LLMs). And how Hammerspace does this with standard pNFS v4.2 utilizing Meta's existing commodity storage/server infrastructure and standard networking, without the need for proprietary client software, or alterations to its storage.

What Changed? How pNFS v4.2 Has Become the Best Option for AI/DL Workloads

NFS (Network File System) is a standard that has become ubiquitous in the storage industry since it was first developed in the 1980s. Virtually every network attached storage (NAS) platform from every vendor has included support for NFS for decades. NFS client and server-side components are included in all Linux distributions, which results in it being one of the most reliable standards throughout the industry.

But despite the fact that it is everywhere, NFS has never been thought of as a high-performance file system. This perception is made worse by memories of the first parallel NFS implementation with pNFS v4.1, which dates back to 2011. Although it performed better than standard NFS, pNFS v4.1 suffered from numerous architectural flaws that severely throttled performance and prevented it from being adopted for high-performance use cases in HPC environments. Instead, parallel file systems such as Lustre, GPFS, and others were adopted for the extreme scale and performance needed in HPC use cases.

The problem for high-performance AI/DL use cases in the enterprise is that HPC file systems require specific hardware, expensive InfinBand networking infrastructure, and are not compatible with standard NFS and SMB file access unless funneled through a file gateway. In addition, these parallel file systems are extremely complex, require custom client software, and require specialized technical staff to maintain them with skill sets not typically found in enterprise IT departments.

This all changed in August 2019 with the introduction of pNFS v4.2 with Flex Files into RHEL v7.7, and in other standard Linux distributions, which dramatically altered the entire architecture of the protocol. For the first time, standard Linux included a parallel file system that met or exceeded the fastest HPC file systems in the world. And pNFS v4.2 with Flex Files achieves this extreme linear scalability with a standards-based architecture, meaning it is compatible with commodity servers and standard Ethernet, and can absorb any combination of commodity NFSv3 storage from any vendor into a high-performance parallel file system.

Just as important, since pNFS v4.2 client and server components are already included in every Linux distribution used in the enterprise, IT staff already have the skills needed to manage the file system, and do not need to install any proprietary client or server-side software on their legacy systems. Other than experiencing a dramatic increase in performance, users access the file system via standard file protocols as before, and may not even be aware that their IT environment has been fundamentally transformed with pNFS v4.2.

pNFS v4.2 with Flex Files: A Standards-Based Parallel File System

To achieve the extreme performance needed for GPU-based AI/DL workloads, pNFS v4.2 includes key features that are found in other HPC parallel file systems, in addition to numerous other capabilities that set it apart from those file systems. It is also a significant improvement over the pNFS v4.1 implementation from 2011, which some vendors still ship today in their products.

As you will see in the key capabilities noted below, the design goal for all the improvements to the pNFS v4.2 spec was to achieve the extreme performance of HPC parallel file systems, while still utilizing standards-based hardware and networking already ubiquitous in enterprise data centers.

In other words, the goal was to not create another vendor-locked storage silo that required exotic networking, specialized hardware, or proprietary client software. Instead, the objective was to enable both high-performance and standards-based compatibility within existing data center architectures, and to introduce into the NFS standard the key capabilities necessary to eliminate the performance bottlenecks that blocked it from being deployed in such high-performance use cases.

Key pNFS v4.2 Advantages and the Problems it Solves:

1. Problem: Client-Server I/O Bottlenecks

A fundamental problem that limits performance in traditional NAS architectures of all types is that both the data and the metadata must be funneled through the same servers and network paths.

This creates bottlenecks under high load conditions, which increase latency and degrade performance. In addition, this severely limits the number of storage nodes that can be added to a system.

Even the most advanced scale-out NAS architectures that claim to deliver the high performance needed for AI use cases suffer from the same fatal flaw. That is why performance of today's scale-out NAS platforms will all progressively degrade as the number of storage nodes increases beyond only a few dozen nodes.

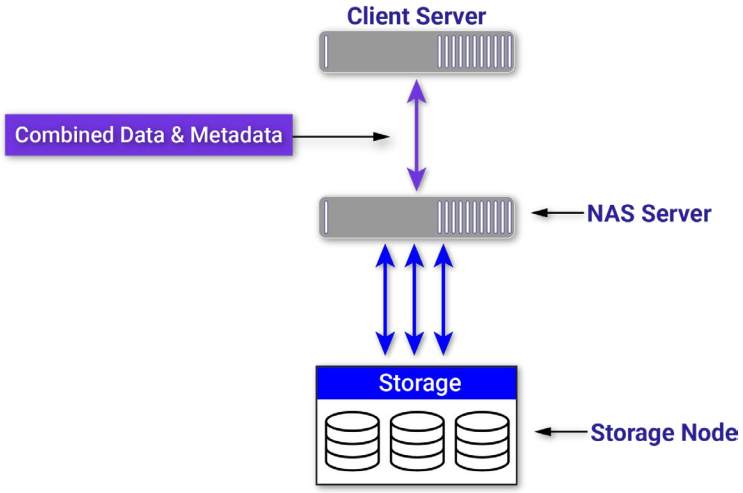

Fig. 1: Standard NAS architectures bottleneck metadata and data operations in the same server, which limits performance and scalability.

Solution » Separating the Metadata Control Plane from the Data Path

Parallel file systems of all types solve this performance bottleneck by creating a separate metadata control plane that is completely independent of the data path. By separating metadata operations from data I/O, parallel file systems allow clients to access data directly from the storage devices without passing through the metadata server, improving data access speeds and reducing resource contention.

This architecture also allows for improved scalability, enabling these systems to scale to even thousands of storage nodes with pNFS v4.2 without overwhelming the metadata server. You'll see examples below of how this and other improvements enable the extreme scalability for AI/DL workloads using pNFS v4.2 at Meta and elsewhere.

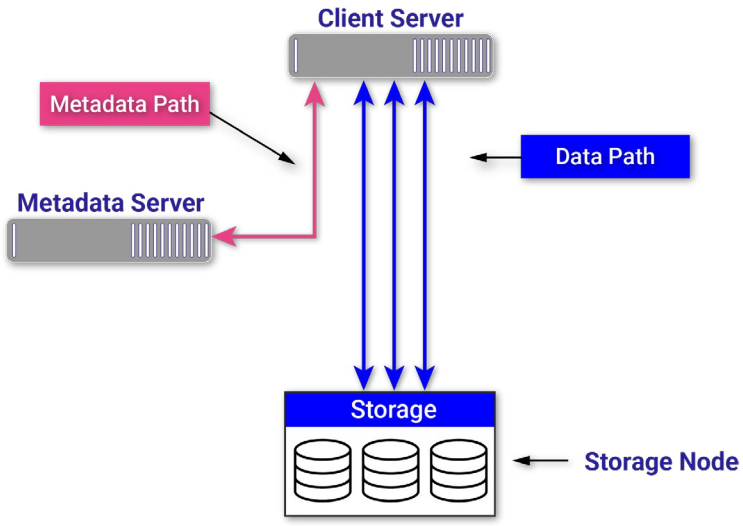

Fig. 2: PNFS v4.2 separates the metadata control plane from the data path, eliminating the I/O performance bottleneck and enabling parallel I/O and extreme linear scalability.

In fact, pNFSv4.2 can even be used to accelerate commercially available scale-out NAS platforms, overcoming the architectural limitations caused by the bottlenecks noted above. Hammerspace has customers doing this today, leveraging the parallelism within the protocol so all clients can talk to all storage nodes with a direct I/O path. Without altering the underlying legacy scale-out NAS, using pNFS v4.2 overcomes the cross-talk and cache contention between the NAS server and the storage nodes, as shown in Fig. 1, and enables even legacy NAS systems to increase performance as much as 3x over what they are capable of natively. See https://hammerspace.com/hyperscale-nas for more information on this.

2. Problem: The Extreme Chattiness of NFS and pNFS v4.1

Another huge limiter to performance of both standard NFS and pNFS v4.1 is the heavy overhead caused by the number of **'getattr'** (get attribute) and several other metadata requests needed to synchronize state between the metadata server and the data server (storage node) for even routine operations.

Even a simple file create and write operation in NFS takes at least three round trip metadata requests to the storage node, and then another two round trips between the client and server to read the file. What's worse, these five round trips are serialized, which means the system has to wait for each operation to acknowledge completion before the next can begin.

In the attempt to be stateless, the NFS client cannot know the state of the files or take action without this constant back-and-forth traffic between the client and server to repeatedly get those state attributes. It has no intelligence, or continuity.

This NFS chattiness problem was not fixed in the pNFS v4.1 with "files" layout implementation from 2011. In that implementation, when a client is given a "files" layout, the metadata server must perform multiple back-and-forth updates to the data server, and then again back and forth to the client to ensure state information such as lock **'stateids'**, **'openstateids'**, and/or delegations are available to validate the traffic between client and server. So even for simple I/O operations, this communication overhead causes pNFS v4.1 implementations to be severely constrained in both scalability and performance.

When there are small numbers of very large files this problem is tolerable to a point. But with large numbers of files, small file sizes, and the mixed I/O that is typical with AI/DL workloads the problem becomes a fatal flaw. This means that a significant percentage of network traffic in such systems is consumed with this back-and-forth communication, putting a serious handbrake on any attempts at achieving high performance or scalability.

Solution » pNFS v4.2 Adds Intelligence to the Client to Eliminate Client-Server Chattiness

With the improvements introduced to the standard in 2019 with the pNFS v4.2 architecture, this problem is completely eliminated. Firstly, the metadata operations in pNFS v4.2 are compounded, so multiple requests can run in parallel. Each request can even refer to the results of earlier requests to automatically build an intelligent command chain without going back and forth between client and server.

Additionally, rather than requiring the client to continually query the metadata server for file state information, with pNFS v4.2 this state information is cached on the client. In effect, the client becomes intelligent, maintaining ongoing knowledge of the state of files without having to be continually reminded.

This fundamental change eliminates 80% of the traffic overhead between client and server that NFS and pNFS v4.1 systems suffer from, reducing metadata server bottlenecks and latency, and dramatically improving performance.

Caching this state information in the client is not like a simple page cache, however. In fact the v4.2 spec is written so that the metadata server actually delegates this intelligence to the client, fully empowering it to act on its own. This enables the pNFS 4.2 client to do its job immediately, without having to go to the metadata server to ask permission, and enables it to have the intelligence to deal with error states and other potential problems, as will be shown below.

This client-side intelligence is a game changer, especially in the extremely high-performance AI/DL environments that require the scalability of a parallel file system to keep GPU clusters continually fed with data at high speed.

3. Problem: The TCP/IP Bottleneck

Another problem with trying to adapt traditional IT infrastructures to very high-performance I/O is the single connection limitations that are part of the TCP/IP suite of communication protocols. Standard TCP/IP uses a single connection between the client and the server, which means data between them can only go as fast as the bandwidth of a single network interface.

This limitation effectively chokes the scalability of pNFS v4.1 or any other system trying to parallelize I/O to achieve greater performance in standard IT environments. In addition, this also prevents load balancing across nodes in scale-out systems, and creates greater risk of interruption, since there is no built-in redundancy when any kind of interruption to the network path blocks I/O.

This is why HPC environments use InfiniBand, or adapt Ethernet using RDMA, with the special NICs and switches that are required for this, both of which can support the multiple connections needed for a parallel architecture. But these upgrades to the infrastructure are expensive, and not typically found in standard enterprise data centers.

Solution » N-connect with pNFS v4.2 Enables Parallelism Across Standard TCP/IP

With pNFS v4.2 the n-connect feature was introduced into the standard, allowing multiple network connections to a single storage node. This capability enables the distribution of I/O loads across multiple standard network interfaces, optimizing the use of available network bandwidth, and reducing congestion on any single connection.

This capability dramatically improves performance of the system without needing to alter NICs, switches, or other standard network infrastructure already in place in virtually every enterprise data center.

More importantly, when combined with the pNFS v4.2 client-side intelligence noted in the previous section, the n-connect capability enables clients to dynamically scale out to even extreme levels without running into bottlenecks.

The n-connect capability is further enhanced with the introduction in pNFS v4.2 of the ability to write to multiple storage nodes simultaneously in a striped or mirrored fashion to spread I/O across them. With this type of synchronous mirroring, you can create redundancy even across storage nodes that are not highly available on their own. Additional enhancements to the pNFS specification are currently being proposed to the community that would include client-side erasure coding as another option for this feature.

With these capabilities, the intelligent pNFS v4.2 client in coordination with the metadata server can direct files to be written redundantly to multiple targets. This means even an array of commodity servers can be highly reliable without any additional sophistication needed in the client.

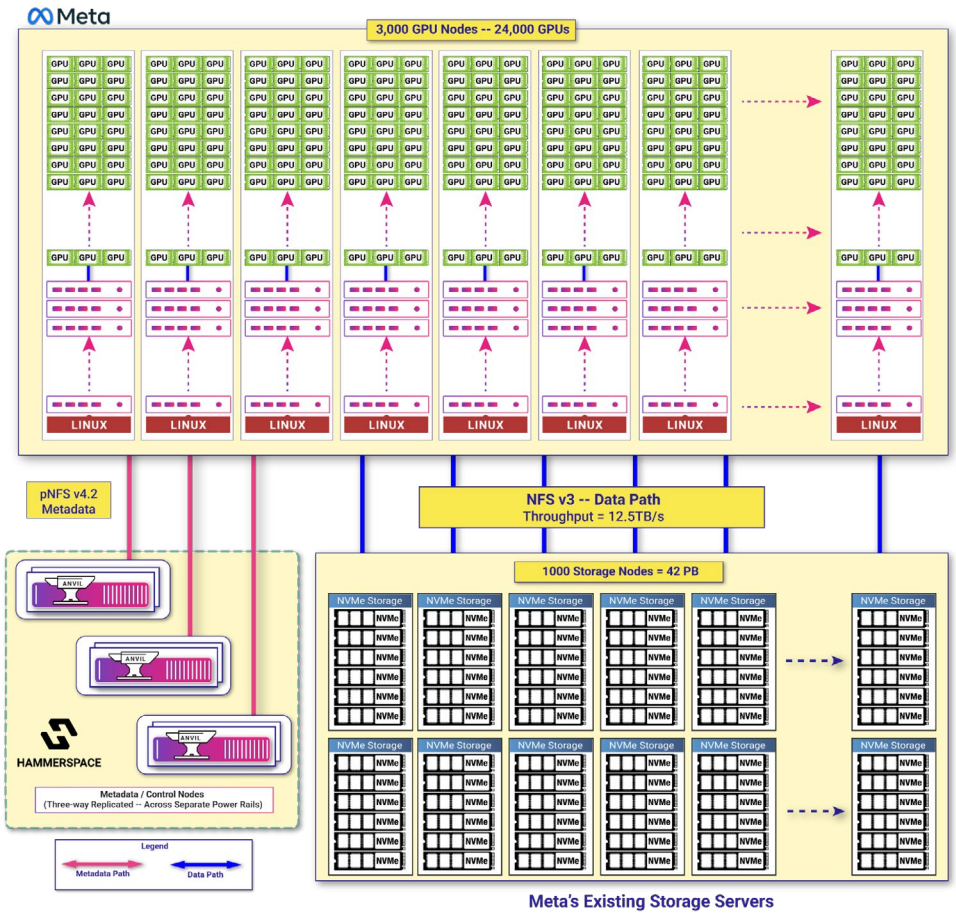

Fig. 3 - Hammerspace utilizes the pNFS v4.2 client already in the Linux kernel to power Meta's AI Research SuperCluster at extreme performance levels on commodity hardware and standard Ethernet, without alteration to existing infrastructure or the need for proprietary client software..

This is one of the ways in which Hammerspace was able to linearly scale across 1,000 storage nodes in Meta's AI Research SuperCluster, parallelizing I/O to achieve throughputs as high as 12.5TB per second over standard Ethernet, enabling the Meta team to keep 24,000 GPUs continually fed with data. Hammerspace leveraged standard pNFS v4.2 that is already in the Linux kernel to do this on Meta's existing servers and storage nodes without modification, and without the need to install any proprietary client software.

4. Enabling Parallel Performance With Any Existing NFSv3 Storage Platform.

To meet the design goal of maximum compatibility with existing storage from any vendor, pNFS v4.2 requires only a standard NFSv3 connection on the storage node, which is included in all NAS platforms today.

This means that just like in Meta's huge AI Research SuperCluster, enterprises can use existing storage from any vendor in a high-performance parallel file system architecture.

Other parallel file systems do not have this cross-platform capability to use existing storage, and usually require specialized servers and back-end storage.

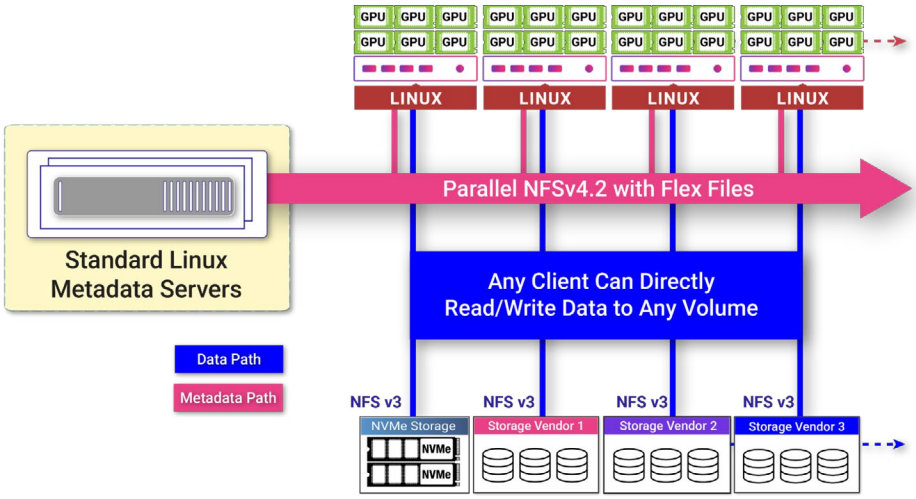

Fig. 4 - pNFSv4.2 with Flex Files can connect parallel file system architecture to any NFSv3 backend storage. Hammerspace extends this to include other storage types on the backend, such as object and tape.

When combined with the other capabilities noted above that eliminate the many bottlenecks found in NFS and pNFS v4.1, this compatibility standard in the pNFS v4.2 specification ensures that enterprises can adapt their existing infrastructure to feed AI/DL pipelines without the need to purchase redundant new storage silos.

Especially in the early stages of an enterprise's AI journey, where unstructured data may be scattered across any number of existing storage resources, it means that data can be put into motion for such workflows from the storage it is on today, without needing the added expense of copying it into a net new repository.

5. Flexible Files Layouts – A Key Differentiator for Parallel NFS v4.2

A significant capability within pNFS v4.2 that enables its tremendous flexibility and scalability is the introduction of Flex Files layouts into the v4.2 specification. This capability leverages the client-side intelligence that was introduced into pNFS v4.2 as noted in the item No. 2 above.

Although extremely powerful, the Flex Files capability is actually quite simple in practice. When a client needs a subset of data for a job, say to run analytics, to feed into a GPU cluster, or for another purpose, it requests a layout, or map, of the file locations from the metadata server to tell it exactly where the data can be found on whichever storage it is on. Often these subsets of data are scattered across multiple different otherwise siloed storage systems, but the layout provides precise information so the pNFS v4.2 client can directly access them at a file-granular level across any or all of them.

This layout is provided to the client instantaneously, after which they directly communicate with the storage servers to perform read or write operations on their own. Since the client is able to intelligently cache that layout information and the back-and-forth **'getattr'** and other state information traffic that pNFS v4.1 suffers from has been eliminated, this means that with pNFS v4.2 the direct I/O access between client and storage entirely bypasses the metadata server. This reduces latency, and dramatically increases performance compared with v4.1 and other file systems.

For AI/DL use cases, where keeping expensive GPU clusters continually fed with data is essential, this capability eliminates delays and ensures high utilization rates.

That is, the completion of one job and the beginning of another that needs a different subset of data across different storage can transition almost instantaneously. Upon completion of the first job, the pNFS v4.2 clients ask the metadata server for a new layout for the next job, which immediately gives them the real-time location of the data sets that they need. No lags in staging data or other interruptions that would otherwise keep the GPUs idle and waiting.

Robust error handling and fault tolerance

The beauty of this architecture is that it is tremendously fault tolerant, with real-time telemetry and layout stats being fed back to the metadata server from both the server side and the client side. This enables the system to adapt to network congestion or other factors, and dynamically update the layouts to the clients to reroute I/O, or to further parallelize I/O on the fly to maintain the desired performance objectives.

This self-healing capability enables the system to intelligently adapt to real-world conditions during a job and to maintain the desired I/O by overcoming network or storage server faults. This even enables the system to automatically compensate on the fly when individual nodes may become sluggish or are underperforming for whatever reason.

In legacy pNFS v4.1 implementations, this is not the case. Yes, failures in a storage node are communicated back to the metadata server in v4.1, but that is as far as it goes. The problem is that any external errors seen by the client side, such as network conditions or other server issues, are lost and not sent back to the metadata server.

In other words, there is no way for a pNFS v4.1 client to communicate fault conditions it runs into back to the metadata server so the system can adapt. Instead, I/O with the client will simply fail and result in a temporary outage that requires IT staff to manually track down the problem and fix it.

In pNFS v4.2, with its client-server feedback loop intact at all times, when the client detects issues the metadata server will automatically update the client with a new layout based upon the combined telemetry from both the client and server, and point it to a different instance of the file, or otherwise reroute the I/O around the fault.

This enables a level of resilience and self-healing in a pNFS v4.2 system that is simply not possible otherwise, and significantly reduces downtime that would otherwise interrupt operations, and reduce I/O to the GPUs.

6. pNFS v4.2 Enables Live Data Mobility – Move Data Transparently Even While it is in Use

Another key feature included in the pNFS v4.2 spec, and which is made possible by the enhancements noted above, is the ability to create new instantiations of a file in real time on another storage node, even while the file is actively in use being read or written, and without interruption to users or applications. This is not the creation of a file copy, but another instantiation of the same file that shares the same file metadata, making it imperceptible to the user.

The importance of this capability in enterprise IT environments is that now data can be moved across storage tiers as a transparent background operation, or parallelized to feed into high-performance clusters for AI/DL workflows without interrupting user access or creating the proliferation of file copies that must be reconciled. Users would never even know that the file was moved from one storage type to another, even if they were actively using it when the move occurred.

In addition to enabling the parallelization to fan data out across multiple nodes for performance purposes as noted in the section on n-connect in item No. 3 above, live data mobility means that for any kind of data operation such as tiering, DR, or even file migrations from old storage to new, it is completely transparent and non-disruptive to users and applications. With pNFS v4.2 disruption from these kinds of actions is a thing of the past.

The elimination of copy sprawl is a critical benefit to IT organizations struggling to contain costs. It also reduces OPEX, since IT teams don't have to wrangle copies, or worry about symbolic links, stubs, or other fragile hooks used by hierarchical storage management point solutions, or other similar technologies.

Live data mobility is a game changer in environments where the same data is needed by different users and applications, especially when data sets are scattered across multiple otherwise incompatible storage silos. It enables IT admins and data owners to place data where it is needed, when it needs to be there, without worrying about interrupting users or applications.

How Hammerspace utilizes pNFS v4.2 to Create a Global Data Platform

Hammerspace is in a unique position to leverage many advantages of pNFS v4.2, which is a key component of its software-defined Global Data Platform. It enables Hammerspace to bring HPC-class performance to existing IT environments that can span vendor silos, multiple sites, and clouds.

One advantage Hammerspace enjoys is its deep knowledge and experience within the Linux community, since the key capabilities of pNFS v4.2 specification noted above were developed over many years by Hammerspace engineers. These and other improvements going back nearly 10 years have been continually pushed upstream into the Linux kernel by Hammerspace, beginning with the ratification by the Linux community of the pNFS v4.2 specification in 2019.

Tight integration with the Linux community is a key component of Hammerspace DNA. Trond Myklebust, the Hammerspace CTO, has also been the Linux NFS Kernel maintainer for more than 20 years, and in that role has ensured that the work Hammerspace does adheres to the standards maintained in the community.

Hammerspace Senior Principal Software Engineer Mike Schnitzer is also a Linux kernel maintainer, responsible for the upstream Linux kernel's Device Mapper (DM) subsystem. And Tom Haynes, another Hammerspace software engineer, is a co-author of RFC 8435, which defined the pNFS Flexible File Layout in 2018, the key feature of pNFS that was adopted into the v4.2 spec the following year.

As a result of this focus on standards, a significant portion of the Hammerspace near-term roadmap includes even more enhancements to the pNFS specification, with numerous features that will be submitted to the Internet Engineering Task Force (IETF) this summer and over the next year for consideration to be included in future pNFS releases.

The advantage of this is that Hammerspace is able to seamlessly build on the capabilities in the pNFS spec to add game-changing features to its Global Data Platform while still maintaining tight compatibility with open standards. This enables Hammerspace to not only bridge storage silos of any type from any vendor, but also leverages pNFS v4.2 to power automated high-performance data orchestration across multiple data centers and cloud storage to create a true high-performance Parallel Global File System.

For AI/DL workloads, this enables Hammerspace software to leverage data from multiple otherwise incompatible storage locations to feed even the largest GPU clusters at maximum performance, without requiring customers to migrate data copies or purchase vendor-locked new repositories.

Below is a short summary of the key features Hammerspace provides that build on the pNFS v4.2 spec to create its Global Data Platform software solution. For more detailed discussion of these features, please refer to Hammerspace resource materials at Hammerspace.com:

1. Data-in-place metadata assimilation.

This capability means that Hammerspace can rapidly pull in the file system metadata from existing storage platforms without the need to migrate data to a new repository.

This powerful feature is extremely fast, such that even in very large environments such as Meta's 42PB, 1,000-node storage cluster, Hammerspace was able to rapidly assimilate metadata and begin file operations immediately.

Hammerspace is compatible with any storage platform, from any vendor. Meaning that assimilation of file or object metadata from multiple otherwise incompatible data silos enables them to all become part of the same global data environment, effectively bridging the gaps between the different storage types.

2. Parallel Global File System spanning silos, sites and clouds.

As existing metadata is assimilated from the storage, Hammerspace builds a cross-platform high-performance Parallel Global File System. The global file system unifies not only different vendor silos in a single data center, but can be extended to span multiple data centers, as well as one or more cloud vendors and cloud regions.

Any storage type can become part of the Hammerspace global data environment. This includes block, NVMe, NAS, object, cloud, and tape storage from any vendor.

All silos, sites, and clouds are now presented as a unified cross-platform global namespace. And just as noted in item No. 6 above, Hammerspace provides live data mobility across all silos, sites, and clouds so file movement and other tasks are non-disruptive to users and applications.

3. Multi-protocol Global File System Access.

The global reach of the Hammerspace Parallel Global File System is immediately accessible to users via standard NFS, SMB, and S3 protocols. As discussed above, no client software is required, since the system is built to be compatible with the existing file access protocols.

From the perspective of a user or application, data is accessible exactly as it was before Hammerspace was installed. Files and folder structures are intact, as before, and users do not need training on a new application or user interface, and in fact may not even be aware that Hammerspace is there behind the scenes.

But what's different is that instead of needing to access their data across multiple file shares or buckets for each different storage silo, now users see a unified global namespace that bridges them all, exactly as if all data in all on-prem and cloud locations had been consolidated into a single giant global NAS.

4. Automated Data Orchestration

With the global reach and the ability to transparently place data anywhere without interruption with live data mobility, this provides the foundation for Hammerspace's powerful data orchestration capabilities at a file-granular level.

Leveraging the ability to move data anywhere in the background, plus the extreme performance capabilities of pNFS v4.2 noted throughout this paper, Hammerspace automates data orchestration to solve a myriad of requirements within the data center, as well as between multiple sites and clouds.

For example, feeding an AI/DL workload into GPU clusters that may be on-premises or at a remote GPU-as-a-Service provider can be automated in the background, even when that data is distributed across multiple silos and locations.

In addition, protecting data with replication to remote sites or clouds is performed with the same non-disruptive data orchestration capabilities. And unlike backup applications or manual processes such as Robocopy or Rsync, these replication file copies are not isolated in separate silos, but instead may be accessed and managed by automated policies globally.

5. Programmable Metadata

In addition to the file system metadata that is assimilated from existing storage, the Hammerspace Global Data Platform supports multiple types of custom metadata. These can be tags, labels, or other references that can help data owners categorize and classify their data.

Leveraging the cross-platform reach of the Parallel Global File System, such custom metadata tags are persistent as data moves over time to different locations, enabling precise tracking and policy-based actions.

In addition, such custom tags can be automatically applied based upon controlled vocabularies, eliminating the risk that users forget to tag something. A folder can be given a custom tag referencing a project ID, cost center, or other variable, and then any data that lands in that folder will automatically inherit that tag.

This enables data orchestration workflows to be constructed that can reference such custom metadata tags to determine data protection rules for certain classes of data, track information for cost reporting, or in the case of an AI/DL workflow, pull only the data that has been cleansed and properly labeled into the high-performance storage tiers to feed the GPU cluster.

Since like everything in Hammerspace this is done at a file granular level, it means individual subsets of data buried deep within otherwise siloed file system hierarchies can be treated globally, without wholesale data migrations that would have been necessary in other solutions.

6. Cross-Platform Data Services.

A key benefit of non-disruptive data orchestration across silos, sites, and clouds is that Hammerspace can automate enterprise-class data services globally across them all. These include the kinds of services that are standard in enterprise-class NAS platforms, but mostly absent from parallel file systems.

Data protection services, such as global snapshots and clones, file versioning, undelete, WORM, global audit, anti-virus, and more can be automated at a file-granular level. Data placement services such as tiering, data migration and DR can all be automated as well, driven by any combination of metadata variables, including custom metadata tags.

Hammerspace includes an objectives-based policy engine that enables administrators to apply plain-language business rules for how different classes of data should be managed. Everything from the data protection level for different data types, the cost of underlying storage to determine data placement policies, and much more may be globally automated with Hammerspace across the entire global data environment.

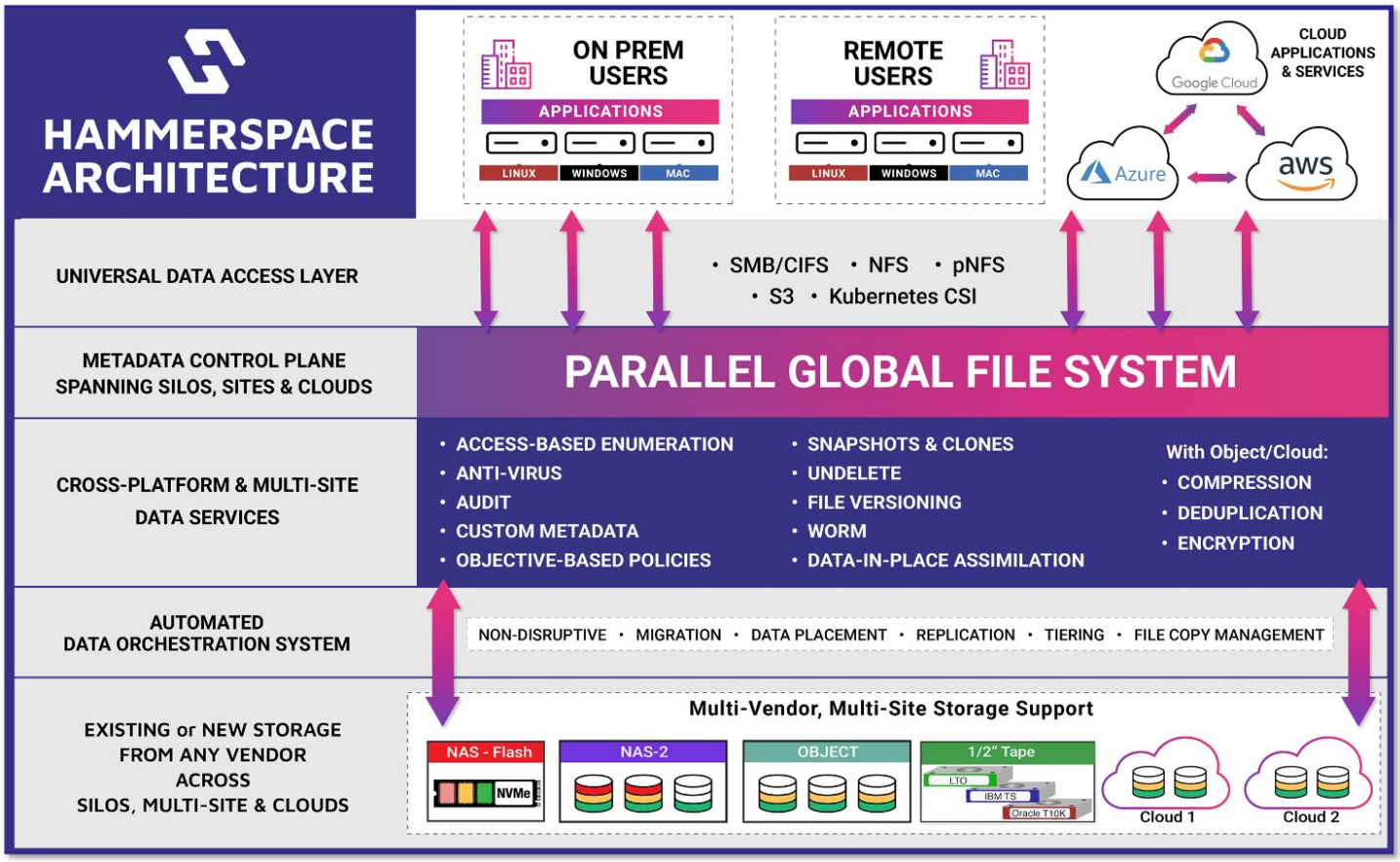

Fig 5 - A logical view of the functional layers of Hammerspace Global Data Platform software. Hammerspace is a single software package that may be installed on commodity bare metal servers, Type-2 hypervisors, or cloud instances.

Summary

Until the recent AI/DL boom, most enterprise use cases worked just fine with lower performing NFS, SMB, and later with S3 protocols. But the revolution caused by AI/DL and the need to feed data into GPU clusters at HPC-level performance changed everything.

The powerful features of pNFS v4.2 enable organizations of all types to take advantage of GPU clusters immediately. It enables them to begin consolidating files to feed AI/DL workloads from data in place on existing systems. Instead of wasting budget on building bespoke new storage platforms to copy their data into, organizations can leverage the systems they have. Budgets can be focused on outcomes, and not on duplication of resources.

Building on this standards-based foundation, Hammerspace extends the power of pNFS v4.2 with data orchestration and other features to enable AI/DL workloads to bridge multiple silos and sites seamlessly, including the ability to burst workloads into remote or cloud-based GPU clusters.

The foundation of pNFS v4.2 which is already in the Linux kernel in every data center on earth is a game changer to provide parallel file system performance within existing data center architectures to accelerate the rapid adoption of GPU-based computing everywhere.