

# The Best of Both Worlds: **Native Object Access Meets Parallel File System Performance**

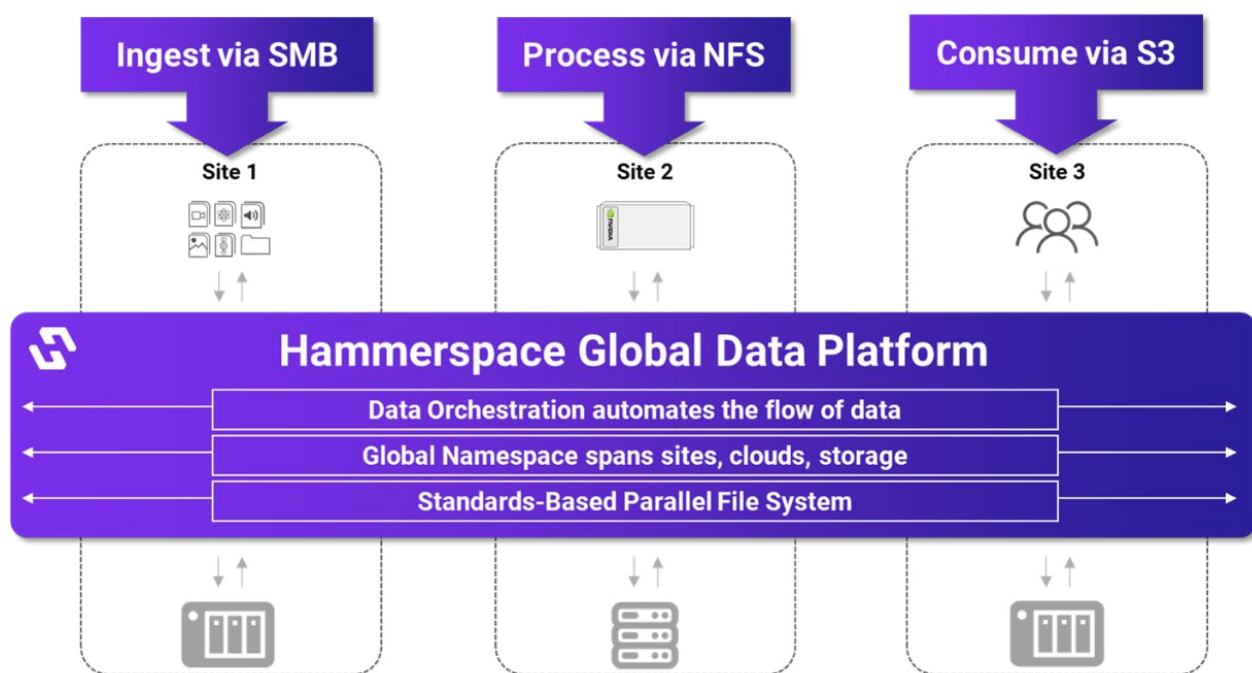
TECHNICAL BRIEF

## The Need for a Consolidated Unstructured Data Platform

Over the last few decades, file storage and object storage have evolved alongside each other, and they were largely deployed for different use cases. A local application requiring high throughput may use a parallel file system, and one running in the cloud would typically use object storage. Data to be used by humans is typically best presented as files, and data for algorithmic consumption is often better presented as objects. But especially with the rise of cloud computing and AI, things aren't so black and white anymore. Storing files and objects in separate silos creates barriers to the full use of an organization's data.

For one thing, applications don't exist in isolation. They get chained together in workflows, with storage requirements that vary at different stages. Mixed file and object workflows are becoming more common, where teams, tools, and data span sites and clouds. Data may originate on a Windows-based instrument and be accessed via SMB, get sent to a Linux cluster via NFS for processing, and the results may be shared or archived using the S3 protocol. Getting data through the workflow involves copying it from one storage silo to another, making it easy to lose track of where things are and eat up storage capacity with forgotten copies.

AI model training and other big data applications require access to enormous amounts of data, especially unstructured data. Simply getting access to all the right data can be a challenge when it's spread around the organization on different types of storage.

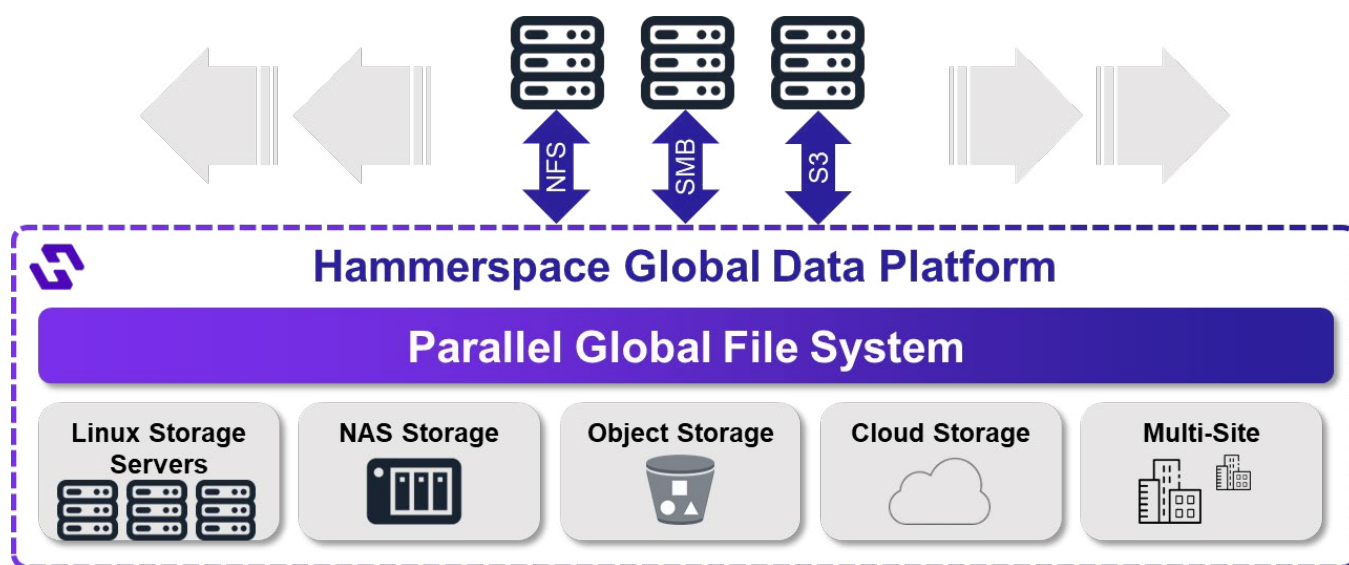


To eliminate this complexity doesn't require a new storage device (yet another silo), but rather a new approach – a data platform that brings together data wherever it lives. One that enables data to be presented as files, objects, or both, depending on the needs of the application and the moment. A data platform that provides parallel file system performance, deep archive economics, and everything in between. A data platform with orchestration capabilities to support workflows that span multi-vendor silos, sites, and clouds.

## Hammerspace Global Data Platform v5.1: The Best of Both

Hammerspace software has long provided this data platform for files, with the ability to orchestrate data to and from any file, object, and cloud storage from any vendor. Now with version 5.1, Hammerspace adds to the client side the ability for users/applications to read and write objects natively, becoming a consolidated unstructured data platform for both file and object data.

Data managed by Hammerspace may be stored and retrieved as files or objects, using NFS, SMB, and S3, regardless of which storage the data is on today, or moves to later. Unlike bolt-on or gateway solutions, in Hammerspace objects are truly equal peers with files. Any share or directory may be exposed as an S3 bucket, and buckets and objects created using S3 may be viewed as directories and files through standard NFS and SMB shares. Data is visible and accessible in a unified namespace via any protocol, reducing complexity instead of amplifying it. "File" and "Object" become different lenses through which data may be viewed and accessed.





















### 1:1 Object to File Mapping

The key to this "dual citizenship" of data across both file and object realms is the principle that one object = one file and vice versa. Each object written via S3 becomes a file in the Hammerspace file system. This is true even for multipart uploads, where the individual object parts are written into a hidden temporary directory and then assembled into a complete file.

Similarly, when an existing share or directory containing files is exposed as an S3 bucket, the files appear as objects with names that reflect the file name, and if located in a subdirectory, the path. Buckets created via S3 become directories in the file system, located at the specified bucket root. Object names with embedded slashes will cause the corresponding directory structure to be created. The figure below illustrates these concepts.



File View		Object View
 Directory: \project1		 Bucket: project1
 File: sched.doc		 Object: sched.doc
 New directory: \project2		 CreateBucket project2
 Create subdir: \project1\user1  Create file: \project1\user1\data.mp4		 Bucket: project1  New object: user1\data.mp4
 New subdir: \project1\user2  New file: \project1\user2\output.ml		 PutObject user2\output.ml

Because objects are files, all the features and capabilities of the Hammerspace GDP now apply to objects, including data protection, auditing, objectives-based orchestration and more. Here are a few basic examples:

- Store objects at high speed onto NVMe flash, and automatically tier them to HDD for longer-term storage after processing is complete. When needed again, pre-stage back to NVMe flash.
- Store objects at site A, replicate to site B for DR protection.
- Support bursting workloads to cloud by orchestrating objects between on-premises storage and cloud storage. Note that all orchestration jobs are non-disruptive to users and applications, even on live files that are in use.

## Uniform Security Policy Across All Protocols

Dealing with security and access control in a multi-protocol environment can be a challenge. Hammerspace uses a uniform security policy across all protocols, including S3. S3 security is based on access keys, (analogous to user IDs), and secret keys (analogous to passwords). Optional object and bucket ACLs (Access Control Lists) may also be configured, and both v2 and v4 signatures are supported.

In Hammerspace, access key / secret key pairs are mapped 1:1 to users, who may be Active Directory users or locally defined. Each user who requires S3 access is configured with an access key and secret key in the S3 server configuration. Because of this 1:1 correspondence between users and access keys, all the access controls previously established on a directory will continue to apply if that directory is exposed as an S3 bucket. For example, if a user doesn't have access to a file or directory via NFS or SMB, they will be denied access to the corresponding bucket or object via S3.

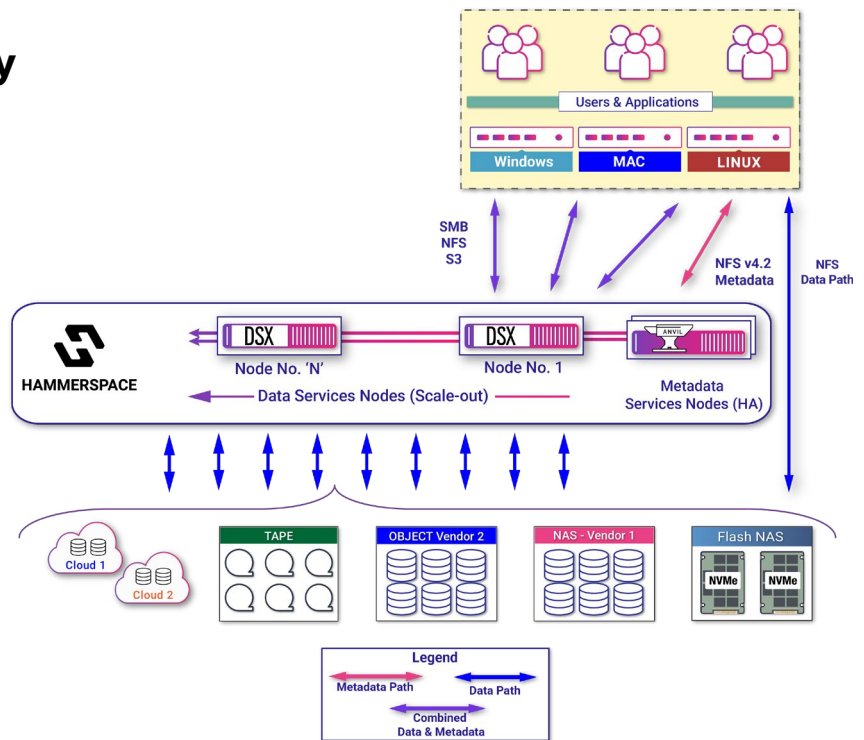


# Architecture and Scalability

For a thorough overview of the Hammerspace architecture, refer to the [Technology White Paper](#).

Hammerspace uses its Anvil metadata services nodes to handle all metadata operations, and DSX (Data Services eXtension) nodes as portals to storage, hosts for storage, and engines for data orchestration. Multiple DSX nodes may be deployed as required, scaling out to as many as 60 nodes per Hammerspace cluster to enable very high-performance, high-volume use cases.

S3 services may be enabled on any or all DSX nodes. When using S3, enabling the services on all DSX nodes is typical. This allows the use of floating IPs instead of DSX-specific IPs, enabling clients to continue working even if a network interface or entire DSX node should fail.



Cluster Name: HS-LAStatus: ✔ Standalone

System

Services

Network

SNMP

Software Update

Support

Active Directory

Users

Licenses

System Backup

Enable Service ▼

Disable Service ▼

Search

<input type="checkbox"/>	Name	IP Address	SMB	NFS v3	NFS v4.1	S3	Portal	Mover	Store
<input type="checkbox"/>	la-dsx-1.hammer.local	10.200.76.101/22	<div>Enabled</div>	<div>Enabled</div>	<div>Disabled</div>	<div>Enabled</div>	<div>✔ Healthy</div>	<div>✔ Healthy</div>	<div>✔ Healthy</div>

Once S3 services are enabled, one or more S3 servers may be created and configured. Like Hammerspace shares, S3 servers are logical entities that are not tied to any one DSX node. Every S3 server is identified with one or more unique host names and has access to the entire global file system.

### Add S3 server configuration

1 Details2 Bucket containers & Buckets3 Access keys

#### S3 Server Details

*i* Provide S3 Server details.

\* Name:

S3Server1 *i*

☒ Default Server *i*

Hostnames:

S3Server1.hammer.local *i*

+

\* Ports:

☐ http:80 ☒ https:443 *i*

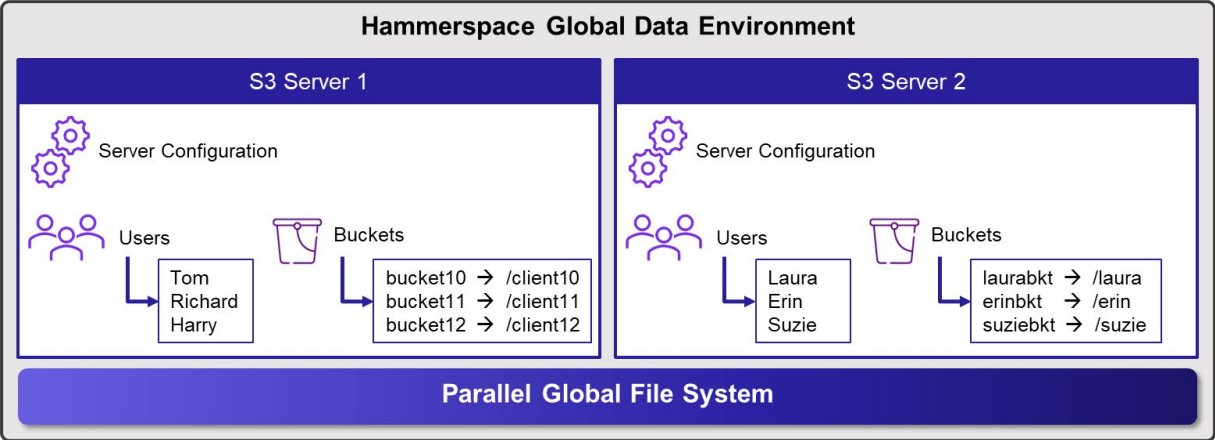
\* Identity Provider:

☐ Local ☐ Active Directory *i*

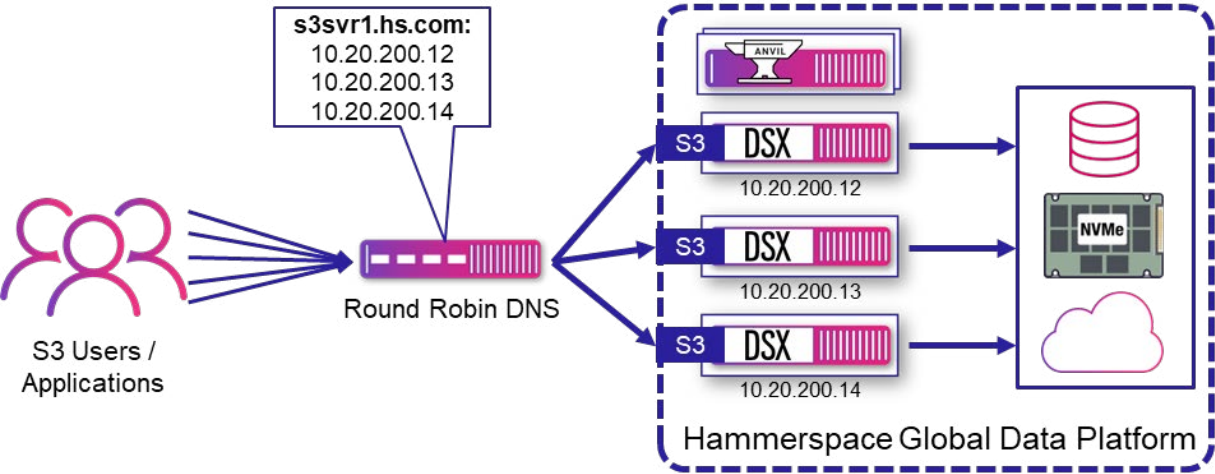




Again, as with shares, multiple S3 servers may each have their own configuration. This provides the ability to isolate multiple tenants from each other. For S3 servers, the configuration includes both the defined users and buckets as well as bucket configuration parameters, such as whether users may create and/or delete buckets. Multiple S3 servers can also share the same configuration, providing scalability and redundancy.



When additional I/O bandwidth is required beyond what a single DSX node can supply, or even simply for redundancy, a round-robin DNS or load balancer may be deployed upstream of Hammerspace to spread S3 requests across multiple DSX nodes.



## Simpler Multi-site Configurations

When a Hammerspace environment spans sites, an S3 bucket is used as a connection point to allow all sites to communicate with each other. This bucket is often deployed in a public cloud service, or sometimes using an object storage system accessible to all sites. Now with Hammerspace’s ability to store and serve object data via S3, the shared bucket may be natively hosted within Hammerspace. This makes it easier to set up multi-site configurations as they can be self-sufficient.



## Supported APIs

As with any third-party (non-AWS) implementation of S3, not every S3 API call is supported. The list of supported S3 API calls is continually growing, in part based on customer feedback. For the current list please refer to the latest product release notes.

## Conclusion

With software version 5.1, Hammerspace becomes a consolidated unstructured data platform that provides both parallel file system performance and native object access. This is ideal for mixed file and object workflows and projects such as AI initiatives that require access to data stored in many different repositories. The elegant and scalable implementation enables access to any data via NFS, SMB, and S3, with a uniform security policy. Key Hammerspace features such as Objectives-based policies, Data Orchestration and automated tiering work as they always have, but now support direct client-side access for objects in addition to files.

[Contact Hammerspace](#) to discuss your requirements and schedule a demonstration.

