Taming the Multi-Protocol Jungle

Unifying File and Object Access in Modern Data Workflows

Agenda

The obvious case for unified file and object storage

Traditional solutions have been limited

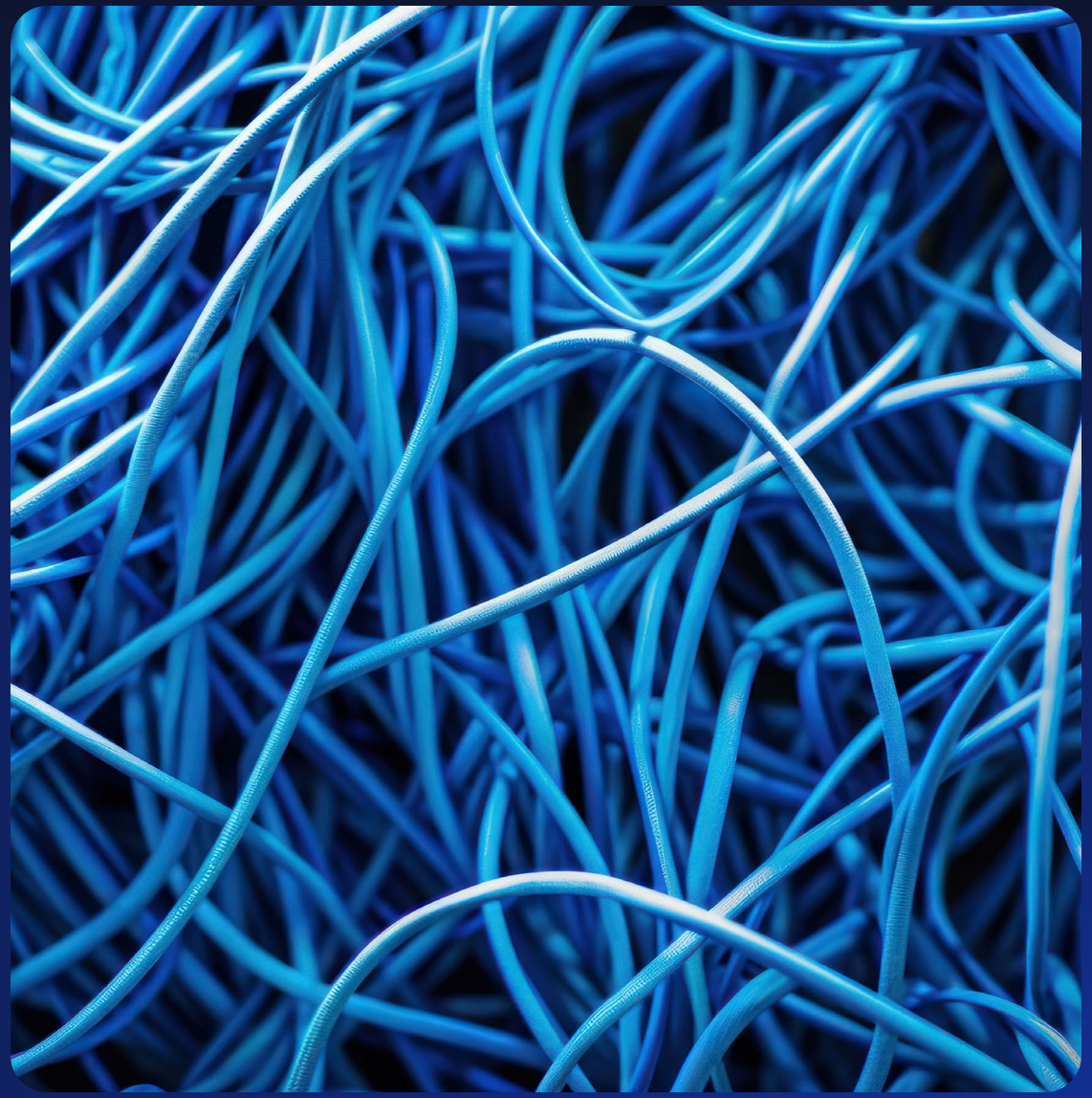
- Something is lost translating between metaphors
- And even worse when translating between security models

Truly unified solutions abstract multiple metaphors

- Providing unified access
- Unified access controls
- Apply active data management from S3 to files
- Plus unified catalog and audit databases

It’s a jungle out there

- Automated jobs create data with NFS and S3.
- Users connect from their Windows machines and embedded devices to consume and create data over SMB.
- Other users and applications connect from a Linux machine and use NFS.
- Backup applications target S3.
- New applications are S3 only ‘cause “Cloud-Native” is fashionable
- Data gets siloed
  - Some on NAS (possibly even separate for NFS and SMB)
  - Other data on object stores including the Data Lake
  - AI and other pipelines require moving or duplicating data across silos



Files and Objects aren't quite the same

They're not just different protocols but different metaphors

- File systems have hierarchical folder structures
  - File applications write a limited number of files/folders
- The bucket structure is flat, with no directory hierarchy
  - Resulting in many objects per bucket
  - Object names may contain a delimiter ie "/"
- Need to map this to files and directories.
- S3 offers new methods to manage permissions.
- Objects are immutable, files allow in-place updates
- Objects offer extended metadata, notifications, active policies

Traditional Solutions Are Lacking

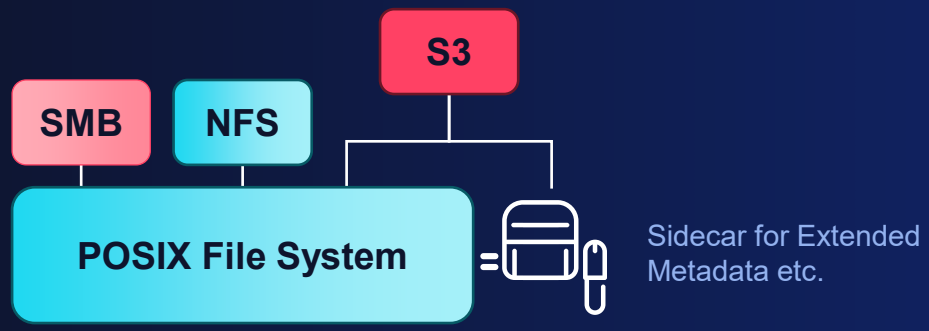
Something's always lost in translation

Object on file

Object protocol module (frequently 3rd party or open source) stores objects as files

Common Problems:

- Bad performance with millions of objects per bucket
- Limited support for object features like extended metadata, versioning, etc. require a sidecar

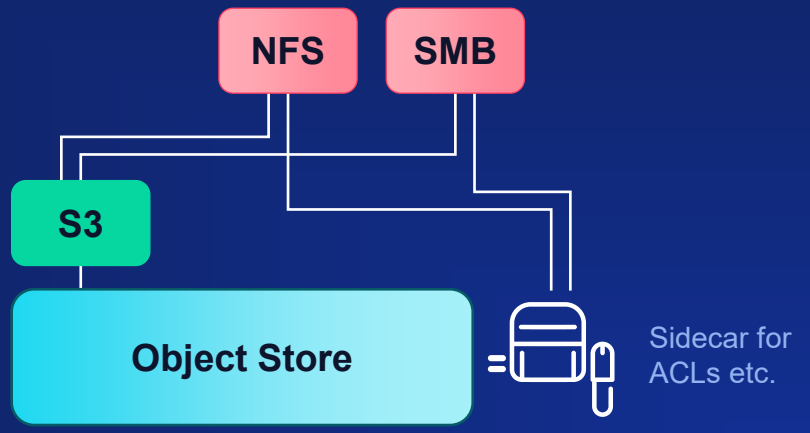


File on object

File protocol modules store files as objects

Common Problems:

- In-place and transactional updates are limited
- File protocol must create new object/version



“Unified” File and Object

File and Object as Independent Namespaces

- **Promises the best of both worlds**
- **Just delivers more silos with shared capacity**
- **Some cross-namespace access may be allowed**
  - But the limits of file-on-object and object-on-file apply
  - So users still have to move/copy between silos

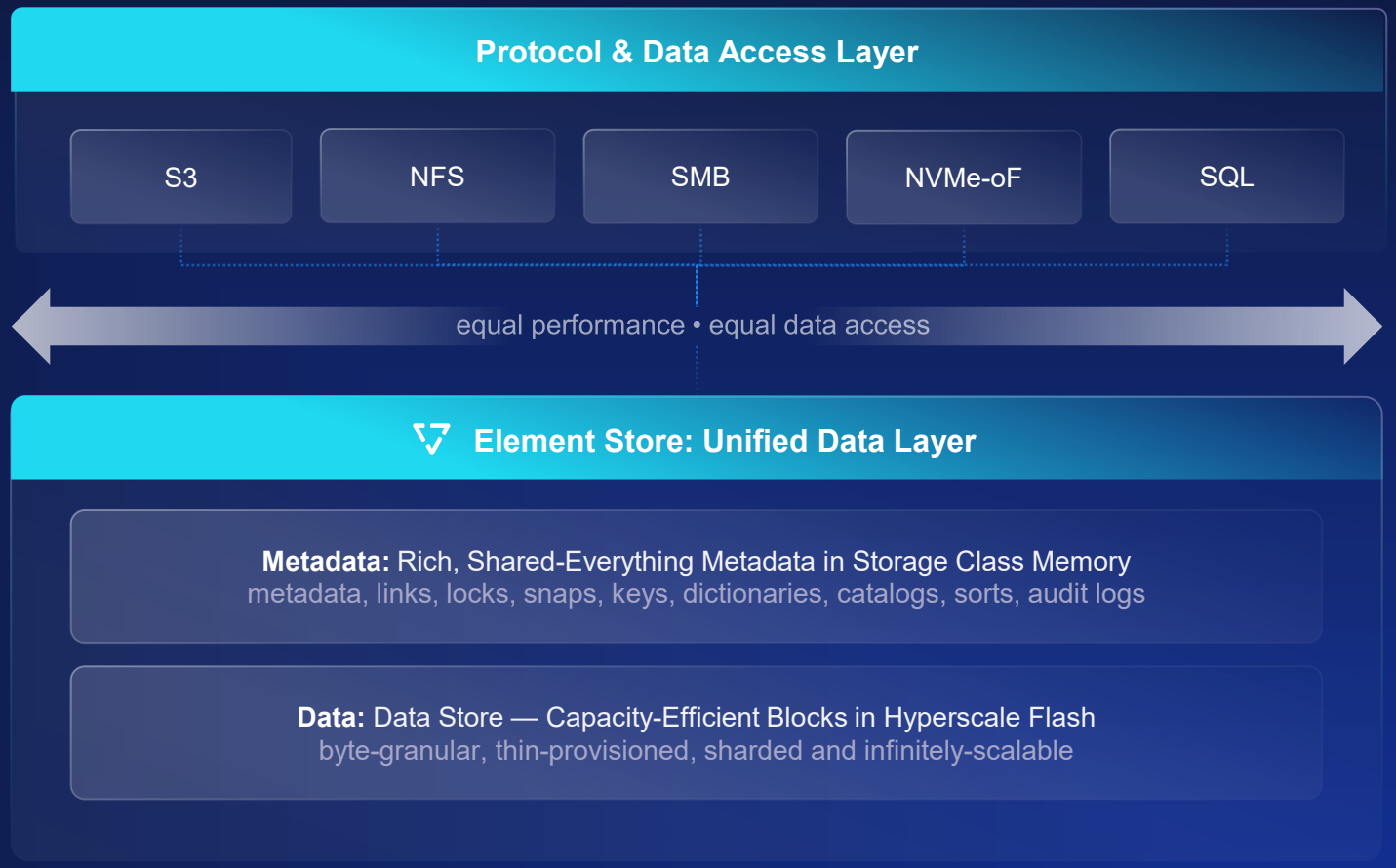
Permission mapping nightmares

- The security team insists on controlling access regardless of protocol.
- POSIX mode bits are simplistic
- SMB/NTFS permissions are much more complex/flexible
  - Inheritance
  - Explicit Deny
- Admins are comfortable with their tools
  - But tools are protocol specific
    - You can’t view/set NTFS ACLs with CHMOD or SETFACL
  - Setting mode bits with CHMOD can overwrite 100s of SMB ACEs
- File systems don’t support the more modern bucket policies vs. ACLs

True Multiprotocol Requires a New Namespace Construct

Introducing the Element Store

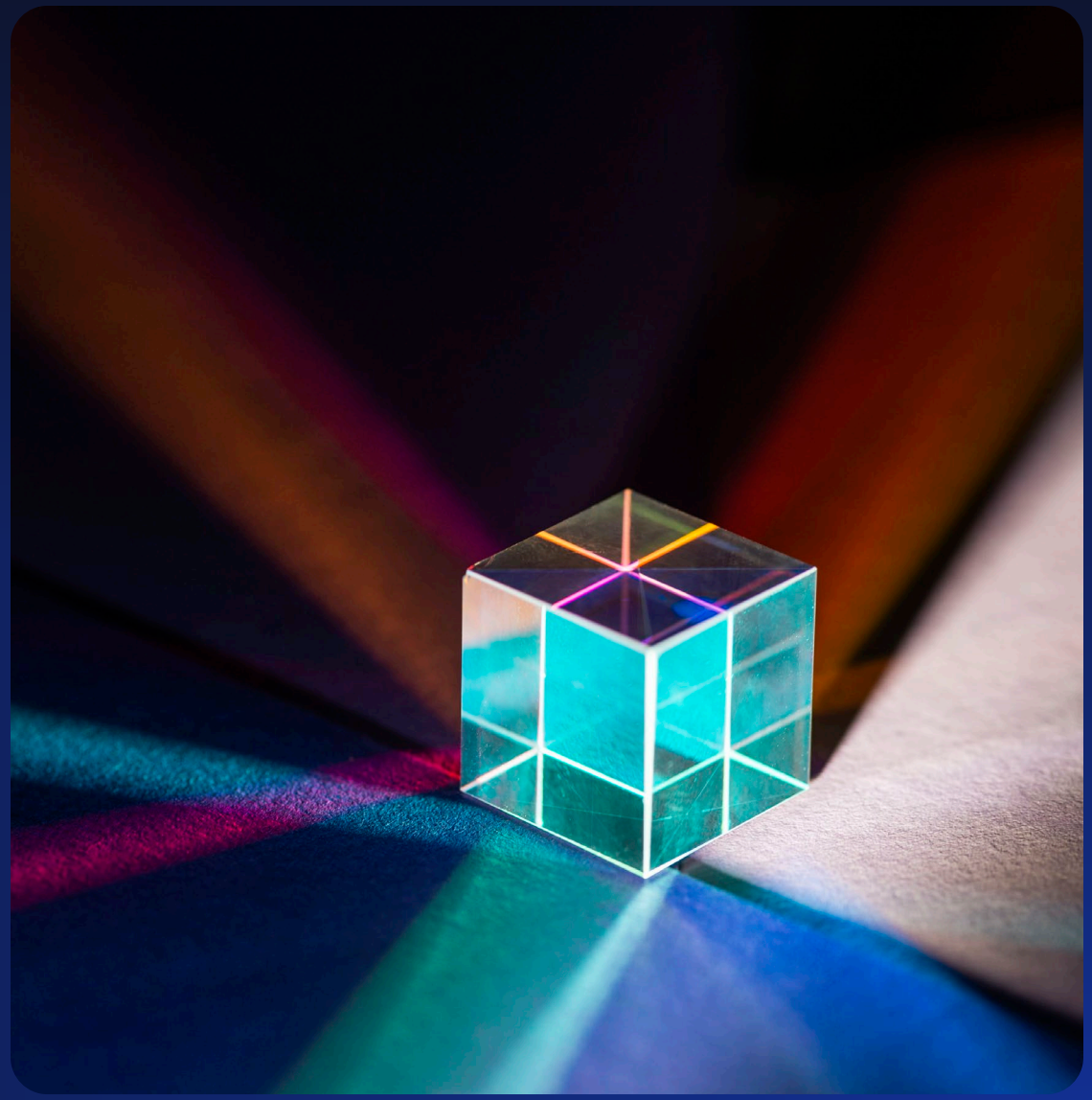
- One namespace to rule them all
  - Support for flat and hierarchical organization
  - Superset of file & object attributes
  - Abstracted ACLs
- Universal metadata constructs
  - Extends to volumes and tables
- Protocols interface directly as peers



One view to rule them all

The logical endpoint for all protocols

- NFS exports, SMB shares and S3 buckets are all logical objects that point to a folder in the namespace.
- View combines these abstractions into a single point of administration
  - A View can expose any or all protocols
  - QoS, filters, etc.
  - Protocol Specifics
    - Root squash
    - SMB share ACLs
- Folder provides data/storage management equivalent
  - Snapshots, clones, replication. Global namespace



VAST permissions

A protocol-agnostic approach

Internal VAST permissions are a superset of all protocol permissions

- Protocol permissions translated to VAST permissions on write.

External IDs (UID, GID, SID, S3 access keys) are translated to VAST IDs.

If you examine files and objects stored on VAST, you can't tell which protocol created them.

- Common logic to make access check decisions for all protocols
- No need to translate from the permission set of the protocol that created the file to the protocol that is accessing the file.
- Apply fine-grained controls to NFS v3 access

When a client sends a GET ACL or GETATTR RPC, translate the VAST representation to the native protocol language.

Enforce access controls even when the client is blind to the specific ACE details

Security flavors

Manage via admins preferred tools, secure access by all

- Most admins would like to manage permissions with a single permission model:
  - POSIX
  - Windows ACLs
  - NFSv4 ACLs
  - S3 identity and bucket policies
- But they can’t agree on which ones
- A security flavor per view allows permission setting via a single preferred protocol
  - e.g. SMB: View, set ACLs via Windows Explorer/PowerShell
    - CHMOD via NFS fails
  - Files/objects that are created by other protocols will get default or inherited permissions.
- Free for all – Last Writer wins flavor is a last resort

Unleash the power of S3 policies

- S3 ACLs is a legacy feature and is disabled by default in AWS S3 buckets.
- Identity policies are a set of access rules that you can attach to users and groups.
  - It applies to all objects that have a specific prefix, so it applies to a “sub tree”.
  - No need to recursively fix ACLs anymore!
- A bucket policy also defines a set of access rules.
  - It is associated with a single bucket.
- Why not use this also for NFS and SMB?
  - Map NFS/SMB RPCs to S3 actions (e.g. NFS READ to S3 GetObject)
  - A prefix is equivalent to a sub directory.
  - Permissions for all protocols on the same dataset are managed by policies only.

Lifecycle rules for the masses

Lifecycle rules is an S3 feature that deletes objects in the background

Objects are deleted based on creation time, size etc

Why not use this also for files created by NFS/SMB?

Automatically enforce data retention policies

Bucket notifications, not just for buckets

Moving from passive storage to orchestrating pipelines

S3 bucket notifications allow the invocation of lambda functions when an object is uploaded, deleted etc.

Real-time AI pipelines can ingest data on event.

Extend the process to event triggers for events via file protocols as well

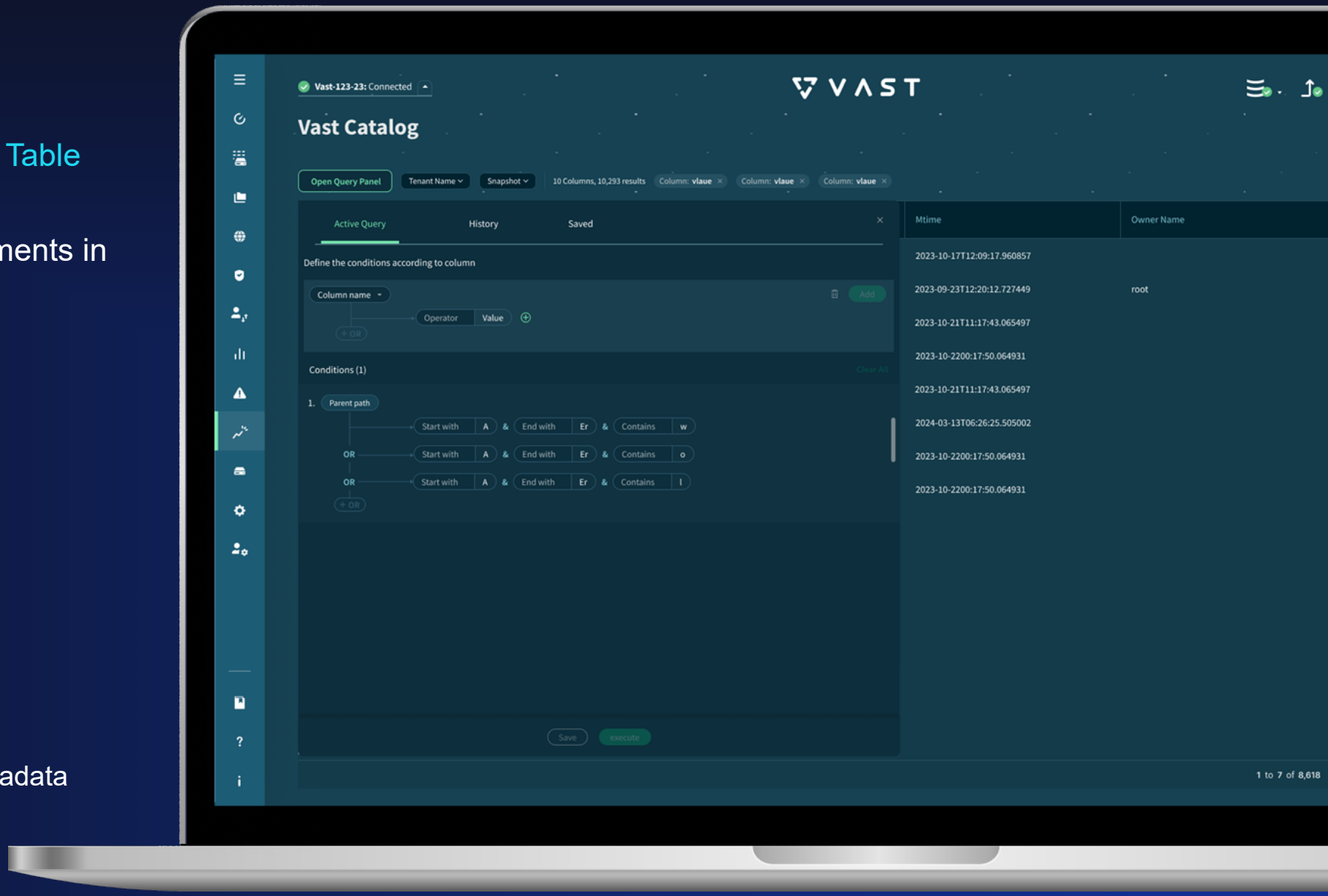
The bucket notification configuration can define triggers also for NFS and SMB.

Automate processing on DNA scans, microscope images on creation

Unified Catalog

System Metadata as Database Table

- Query across billions of elements in seconds
- Full VAST Metadata
  - S3 tags for files
- Snapshot table for
  - Point in time
  - Change/Delta queries
- Join to additional tables for additional metadata
  - Expose embedded metadata
  - Vector embeddings



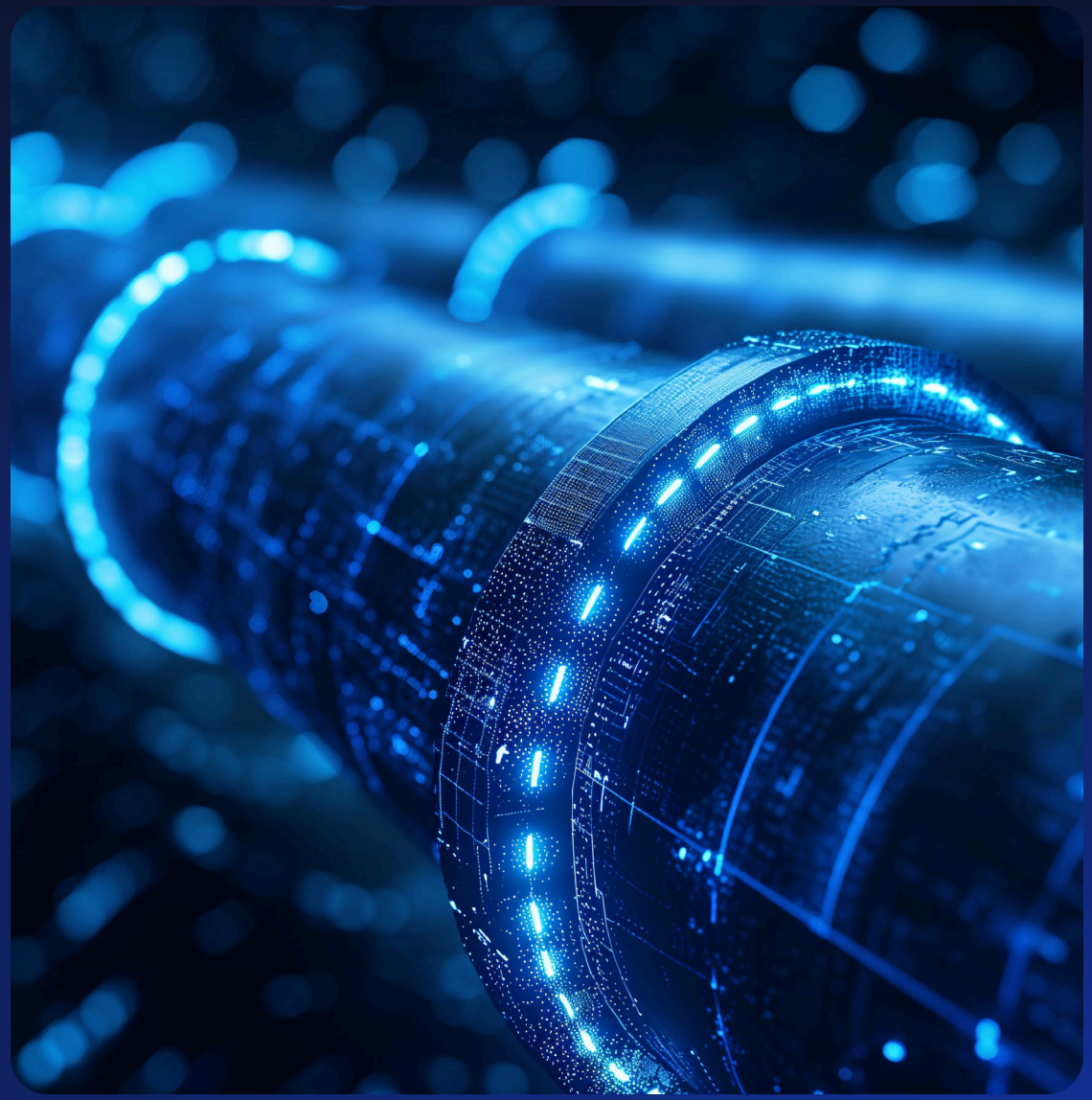
Unified Audit

Protocol gateways generate multiple audit logs

“Who read the customer file around 3AM” requires correlation across multiple independent logs

Data valuable for more than Audit
Performance monitoring

Summary

Unified file and object enables AI pipelines

- Traditional overlays and gateways are no longer sufficient
  - They typically don’t scale
  - Translation always reduces fidelity
- True unification requires a clean slate
  - Metadata including all protocol features
  - Abstracted superset access controls
  - Flexible administration
  - Unified catalog, audit, etc.
- Fully equal access regardless of protocol
  - Makes application deployment simple