MINIO

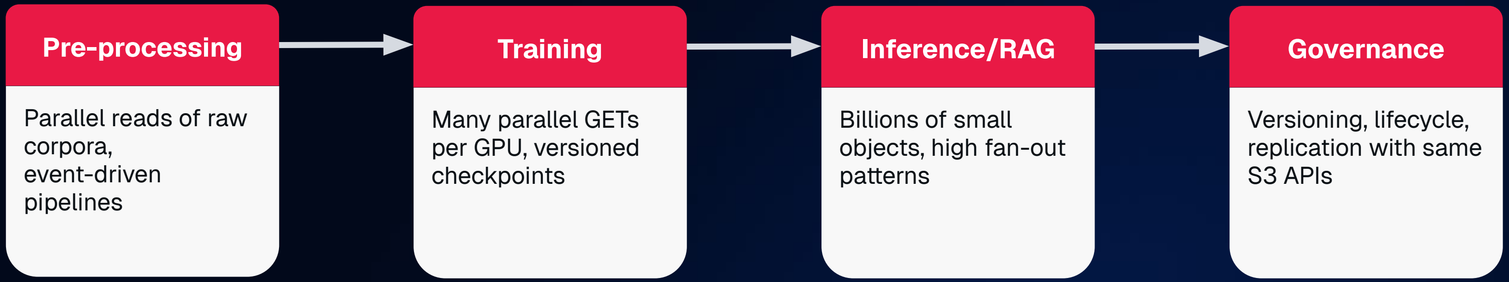
Object Storage Is the Backbone of AI

Murat Karslioglu

Head of Global Field Architects

murat@min.io

AI isn't just compute, it's a data movement, durability, and economics problem



Pre-processing

Parallel reads of raw corpora, event-driven pipelines

Training

Many parallel GETs per GPU, versioned checkpoints

Inference/RAG

Billions of small objects, high fan-out patterns

Governance

Versioning, lifecycle, replication with same S3 APIs

MINIO

Why Object-Native is Required at AI Scale

Single-Namespace Scalability

Store **10s to 1,000s of petabytes** of structured, semi- and unstructured analytical and AI data, all in a single global namespace.

Performance for Unstructured Data

Efficiently transform, analyze, and train using thousands of CPUs and GPUs with **massive parallel throughput across both large *and* small files** such as txt, JSON, parquet, audio, image, video etc.

Cost Effectiveness

Object-native storage persists data more efficiently than file storage due to its **flat namespace and software-defined** use of non-proprietary hardware.

Deep Ecosystem Support

All leading analytics/AI/ML tools such as **Hugging Face**, **PyTorch**, **TensorFlow**, **Ray**, **Kubeflow, Iceberg, Hudi, Hive, Spark, Presto, and Dremio** natively support S3/object storage APIs.

Versioning & Immutability

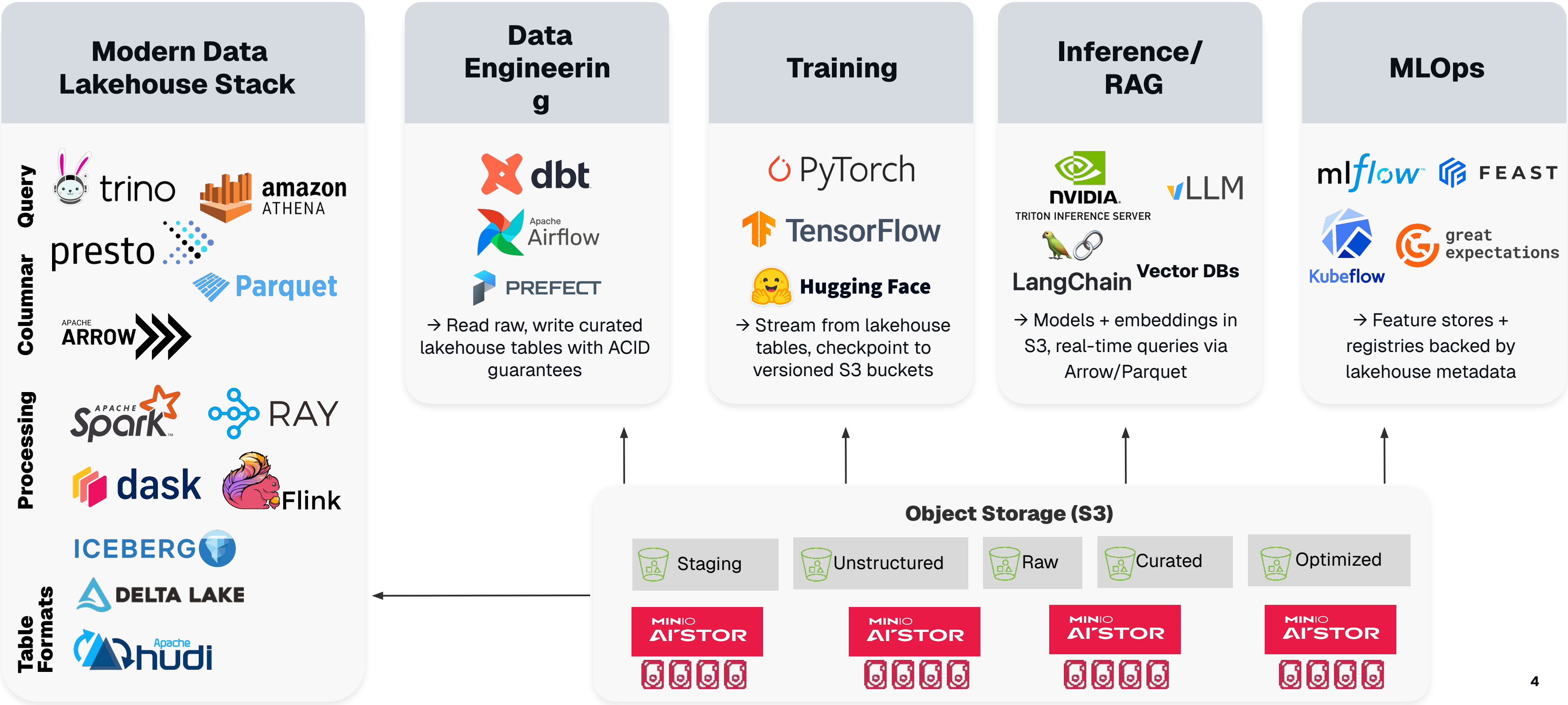
Easily **reproduce training runs and rollback models or data**, via safe, fine-grained continuous data protection powered by object immutability and versioning.

Operational Efficiency

Object-native storage has **no file system hierarchy or metadata bottlenecks** making it faster and easier to manage, automate, and monitor at AI scale.

The CSP Challenge: Cost & control barriers too high for most enterprises

AI Ecosystem on S3



Modern Data Lakehouse Stack

Query: trino, amazon ATHENA, presto

Columnar: Parquet, APACHE ARROW

Processing: APACHE Spark, RAY, dask, Flink

Table Formats: ICEBERG, DELTA LAKE, Apache hudi

Data Engineering

dbt, Apache Airflow, PREFECT

→ Read raw, write curated lakehouse tables with ACID guarantees

Training

PyTorch, TensorFlow, Hugging Face

→ Stream from lakehouse tables, checkpoint to versioned S3 buckets

Inference/ RAG

NVIDIA TRITON INFERENCE SERVER, vLLM, LangChain, Vector DBs

→ Models + embeddings in S3, real-time queries via Arrow/Parquet

MLOps

mlflow, FEAST, Kubeflow, great expectations

→ Feature stores + registries backed by lakehouse metadata

Object Storage (S3)

Staging, Unstructured, Raw, Curated, Optimized

RAG on Object Storage

Simple scale for ingest, retrieval, and provenance

Data Layout

- `raw/`: source docs (PDF, HTML, images)
- `chunks/`: normalized text, traceable to raw
- `embeddings/`: vectors as Parquet shards
- `policy/`: versioning, retention, tags

Event-Driven Build

- PUT on `raw/` triggers chunk→embed→write pipeline
- Batch into 64-256 MB shards for efficient retrieval
- Stamp lineage into object metadata

Retrieve & Serve

- Vector DB returns IDs → dereference chunks in S3
- Cache hot items locally, S3 stays golden copy

Training Patterns

Deterministic datasets, fat pipes, safe checkpoints

Data Layout

- 64-256 MB shards (WebDataset/Parquet)
- Balanced prefix sharding
- Manifest.json pins dataset version + hashes

Feeding GPUs

- 4-16 readers per GPU with async prefetch
- Multipart GET (8-64 MB parts)
- Local NVMe cache >80% hit rate

Checkpointing

- Atomic tarballs → single object PUT
- Versioning + Object Lock (WORM) for safety
- Resume via last_good.txt pointer

Hardware Trends

Dense NVMe

- E3.L/E3.S/U.2 QLC (122.88-245.76+ TB) > multi-PB per rack
- Front-service, better thermals, fewer chassis

Network First

- 2×100-400 GbE per node standard
- Non-blocking leaf-spine, jumbo frames

Right-Sizing

- Spend on NICs/NVMe before extra CPUs
- EC 12:4/16:4 vs 3× replication = 2× capacity

What to Measure

- $/TiB-usable, watts/TiB, throughput per rack
- GPU util >90%, p99 GET latency

Design Playbook

Field-proven checklist for 30-min deployments

Network First

- 2×100-400 GbE per node, non-blocking fabric
- Jumbo frames, tuned RSS/multi-queue

Feed GPUs

- 8 readers/GPU (tune 4-16), async prefetch
- Local NVMe cache >80% hit rate

Shape the Data

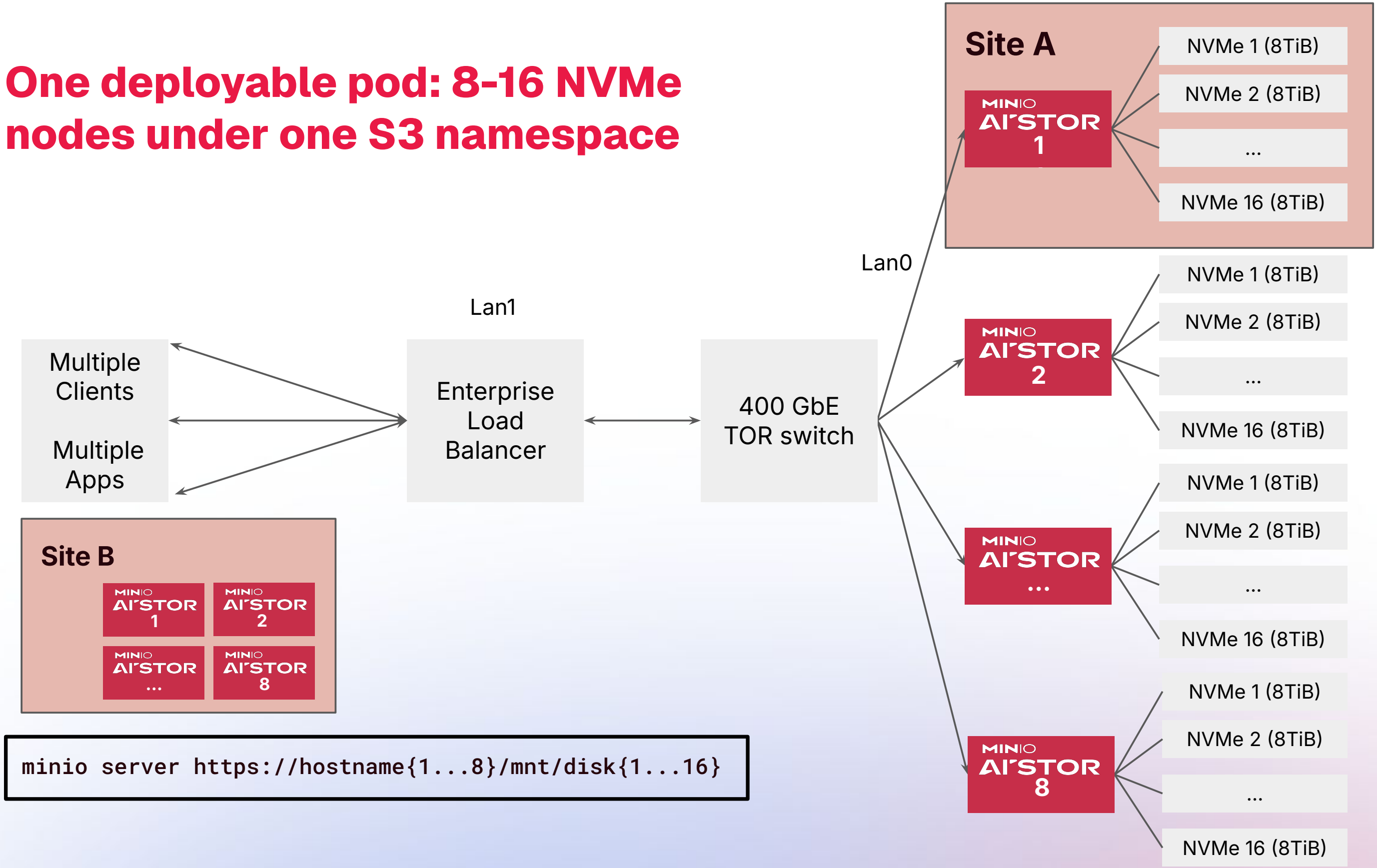
- 64-256 MB objects, balanced prefix sharding
- Layout: s3://lake/{domain}/{dataset}/{version}/dt=YYYY-MM-DD/

Protect & Govern

- EC 12:4/16:4, versioning + Object Lock
- Lifecycle hot>warm>cold from day 0

Reference Architecture

One deployable pod: 8-16 NVMe nodes under one S3 namespace



Multiple Clients

Multiple Apps

Lan1

Enterprise Load Balancer

400 GbE TOR switch

Lan0

Site A

MINIO AISTOR 1

NVMe 1 (8TiB)

NVMe 2 (8TiB)

...

NVMe 16 (8TiB)

MINIO AISTOR 2

NVMe 1 (8TiB)

NVMe 2 (8TiB)

...

NVMe 16 (8TiB)

MINIO AISTOR ...

NVMe 1 (8TiB)

NVMe 2 (8TiB)

...

NVMe 16 (8TiB)

MINIO AISTOR 8

NVMe 1 (8TiB)

NVMe 2 (8TiB)

...

NVMe 16 (8TiB)

Site B

MINIO AISTOR 1

MINIO AISTOR 2

MINIO AISTOR ...

MINIO AISTOR 8

```
minio server https://hostname{1...8}/mnt/disk{1...16}
```

A 8 node 16 drive cluster

Will survive the loss of any 48 drives or 3 servers

Total usable capacity will be ~ 63% of raw storage deployed

Bare Metal Deployment Recommendation

Storage Server Reference

	Dell	HPE	Supermicro
1U	 PowerEdge R6715	 ProLiant DL325	 ASG-1115S-NE316R
2U	 PowerEdge R7715	 ProLiant DL345	 ASG-2115S-NE332R

Single Node Configuration:

- **CPU** - Single AMD EPYC™ 9655P 96 cores or higher
- **RAM** - DDR5 RDIMM-6400
  394GB or higher (12×32 GB) (1U Medium) or
  786GB or higher (12×64 GB) (2U Large)
- **NIC** - 2×400GbE or 2×100GbE
- **STORAGE** - E3.S NVMe SSD
  16x E3.S NVMe SSD 15.36TiB (1U Medium) or
  32x E3.S NVMe SSD 61.44TiB (2U Large)

Example: 2U Large Config: (~9.6 PiB useable)

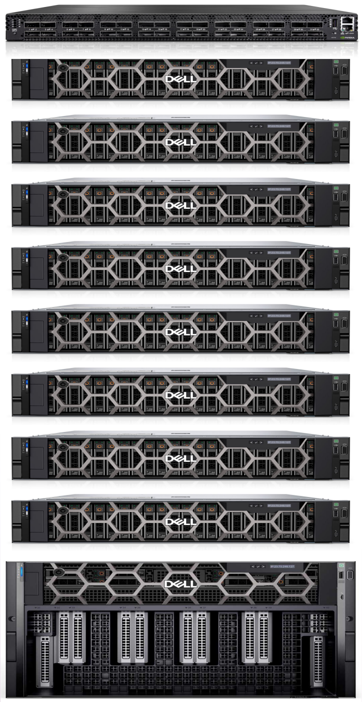
- **8×2U nodes (16U)**
- **Erasure Code Stripe Size (K+M) - 8**
- **Erasure Code Parity (M) - 3**
  - Drive Failure Tolerance: 96
  - Server Failure(s) Tolerance: 3

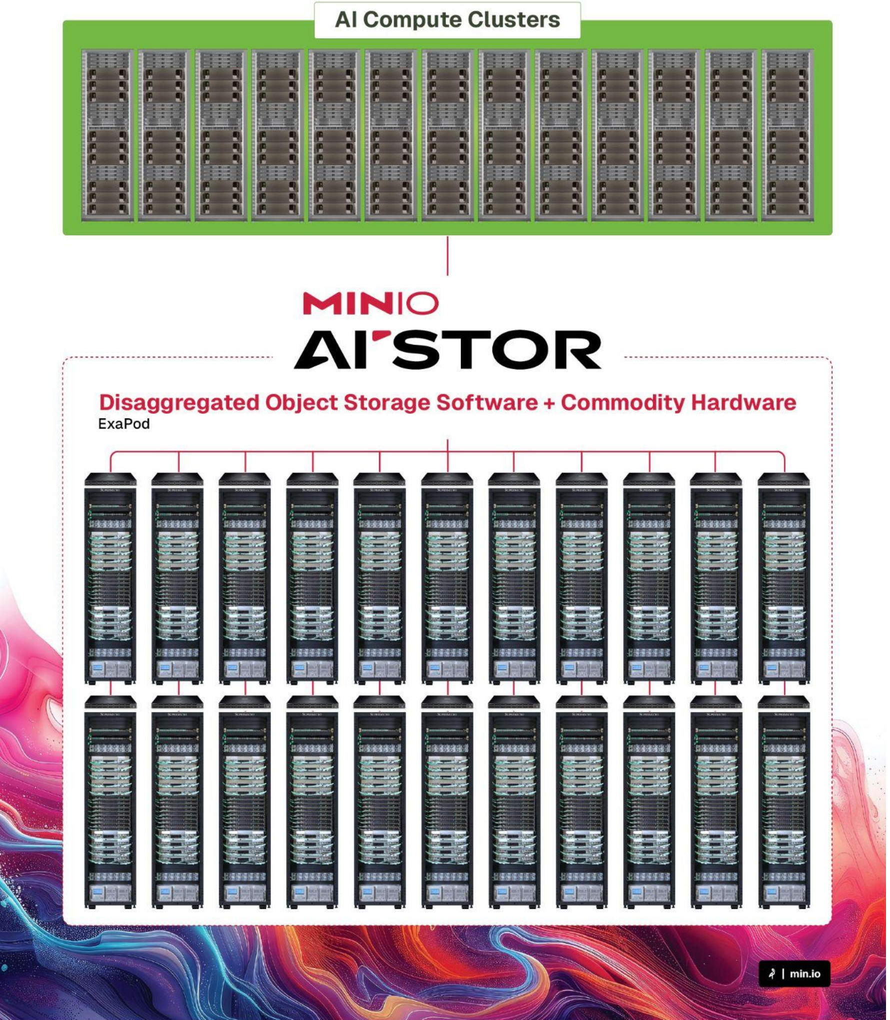
NOTE: For sizing and throughput estimates, please see the MinIO sizing calculator

AI DataPOD Reference Design

- 8 x **PowerEdge R7715** (2U Chassis)
  - AMD EPYC™ 9755 (128 cores)
  - 32 x E3.S NVMe PCIe 5.0 drives (61.44 TB)
  - 400GbE **NVIDIA ConnectX-7** NIC (x16 PCIe Gen 5.0)
- 1 x **NVIDIA Spectrum 3** - 400GbE Ethernet (or PowerSwitch Z Series)
- 1 x **PowerEdge XE9680** 6U Chassis /w 8 x **NVIDIA H200** GPUs
  - 2 x 400GbE **NVIDIA ConnectX-7** NICs for storage networking
- **Dell Rack** and **PDU**

AI DataPOD Rack Unit





AI Compute Clusters

MINIO AISTOR

Disaggregated Object Storage Software + Commodity Hardware

ExaPod

min.io

The Only Exabyte-Scale AI Storage

1.0 EiB

Single namespace capacity

484

Storage servers

21.8TiB/s

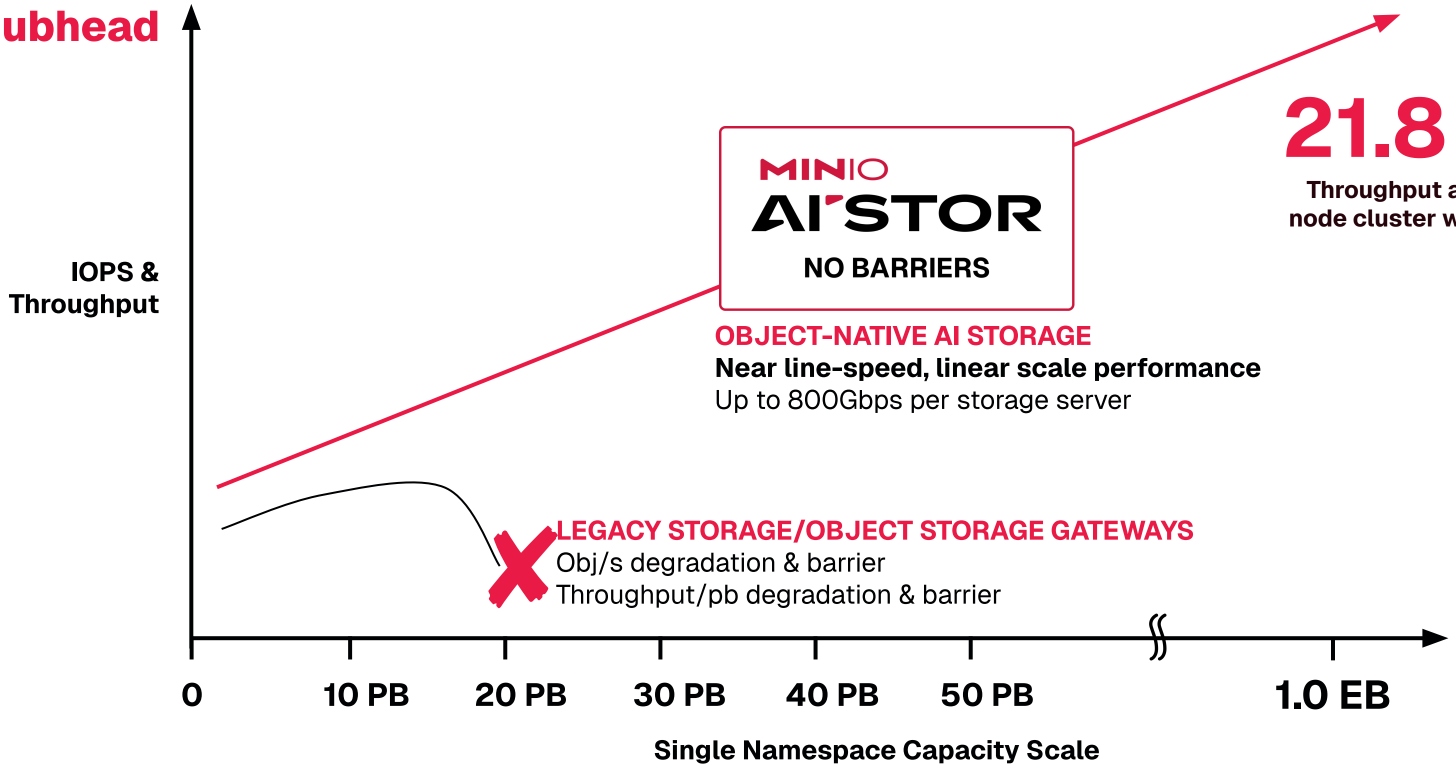
Read throughput

$3.0/TiB

Per month, S/W & H/W, all features included

Why Object Storage for AI: PERFORMANCE

Subhead



21.8 TiB/s

Throughput at 1.0 EiB, via 484 node cluster with 400Gbps NICs

OBJECT-NATIVE AI STORAGE

Near line-speed, linear scale performance

Up to 800Gbps per storage server

LEGACY STORAGE/OBJECT STORAGE GATEWAYS

Obj/s degradation & barrier

Throughput/pb degradation & barrier

Call to Action

Make GPUs happy. Make auditors happy. Cut racks.

Network First

- 2×100-400 GbE per node, non-blocking fabric
- Jumbo frames, RSS tuning, NUMA-aware IRQ pinning

Protect with Policy

- EC 12:4/16:4 over 3× replication (2× capacity gain)
- Versioning ON + Object Lock for checkpoints
- Lifecycle automation: hot > warm > cold from day 0

Shape Data for Scale

- 64-256 MB objects with balanced prefix sharding
- 4-16 readers/GPU with async prefetch
- Measure: GPU util >90%, p99 GET latency

Validate with Real Workloads

- Test with actual I/O patterns, not just synthetic
- Lock in SLOs: bytes/GPU-hour, cache hit %, time-to-resume

Visit:

min.io

Thank you

@minio
https://github.com/minio/minio
https://slack.min.io
https://min.io