MINIO

Stateless Object Storage for the AI Era

“MinIO AIStor seamlessly integrated with our existing infrastructure, improved performance matrix and gave us greater flexibility in our AI/ML environments.”

Anas AlMarabha, Ministry of Education SA

Legacy architectures hit walls. AIStor removes them.

- **High-performance S3 data store:** Purpose-built for analytics and AI
- **Linear scaling:** Terabytes to exabytes, no performance cliffs
- **Fully stateless:** No central metadata database to bottleneck or fail
- **Deterministic hashing:** Object placement computed, never coordinated
- **Inline metadata:** Stored alongside object data across erasure sets
- **No single point of failure:** Eliminates the centralized metadata service entirely
- **Consistent at any scale:** Same performance at thousands or billions of objects

The Challenge: Data Grows. Legacy Storage Architectures Don't

Most object storage platforms weren't built for billions of objects across petabytes of capacity. They layer S3 gateways on NAS or SAN backends, or rely on centralized metadata databases that hit hard limits under load and become single points of failure. When these architectures bottleneck, the symptoms are immediate. Write throttling. Stalled LIST operations. Cluster wide halts. Scaling requires database sharding, rebalancing, and migration windows. Day to day operations require backups, schema migrations, performance tuning, and DBA overhead for infrastructure that shouldn't need it. For AI and analytics workloads, the impact is severe. Up to 70% of model training time is lost to storage I/O constraints. GPU utilization dragged below 40%. The underlying storage architecture becomes the ceiling on growth and productivity.

The AIStor Solution: Stateless Object Storage

AIStor treats everything the same way. Data, metadata, policies, configurations, internal state. All stored as objects with co-located metadata, distributed across erasure sets using deterministic hashing. No external database. No staging layer. No background processes. Every object gets the same data

services applied inline: erasure coding, encryption, checksums. The architecture is fully symmetrical and stateless, which is why it's so resilient. Erasure sets span racks, so in a 10 rack deployment, up to 4 full racks can fail and the cluster keeps serving requests. Any four. The sections that follow detail how this works in practice: how AIStor's S3 native design avoids gateway performance penalties, how deterministic hashing eliminates coordination overhead, and how erasure coding enables self-healing without manual intervention.

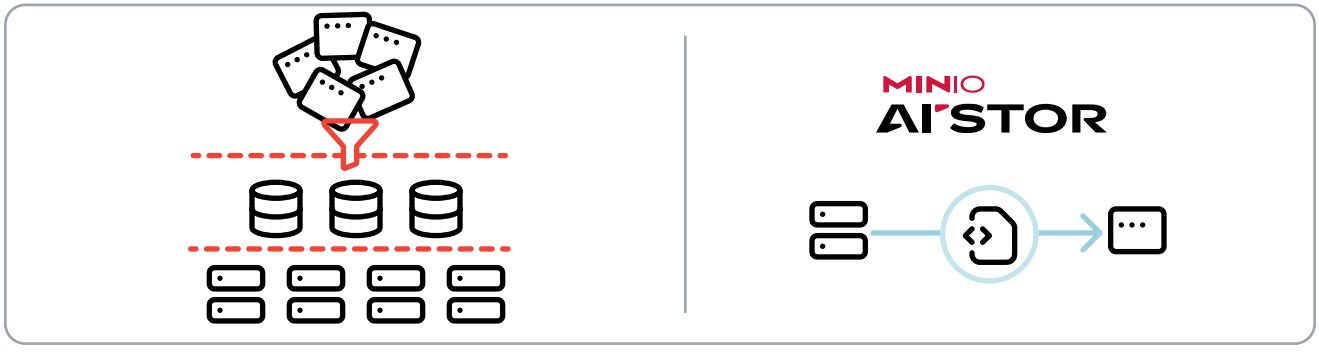
Object-Native by Design: Why S3-Native Architecture Matters

Object-Native means the system treats data as objects (data plus metadata plus unique ID) as its fundamental unit, not files in folders or storage blocks. Everything in AIStor is designed around this model. Every S3 operation executes directly against the object storage layer with no intermediate translation. A PUT writes erasure coded shards directly to drives across the cluster in a single operation. A GET retrieves shards and reconstructs the object without file system overhead. LIST operations query distributed metadata inline with object data rather than walking directory trees. AIStor implements the full S3 API natively: versioning stores each version as a discrete object with its own erasure coded shards, object lock enforces retention at the object level without external lock managers, lifecycle policies execute transitions and expirations without external schedulers. Code written for AWS S3 runs against AIStor without modification.

This is a fundamentally different approach than gateway based architectures, which convert S3 API calls into POSIX file operations or block I/O on an underlying SAN or NAS backend. In those systems, a simple PUT becomes a file create, a write, and a close, each with its own system call latency. Listing objects requires traversing directory structures that weren't designed for flat namespaces with millions of keys. Rename operations that are atomic in S3 semantics become copy and delete sequences. Gateway architectures also force dual data protection: RAID or replication at the block or file layer, plus a separate scheme at the object layer. Each layer consumes capacity, compute, adds failure modes, and increases operational complexity. AIStor's single layer design eliminates all of this.

How AIStor Scales Without Slowing Down

When an object is written, AIStor computes a hash of the full object path (bucket, prefix, and object name) to determine which erasure set will store it. This deterministic hashing means that for any given object key, AIStor always selects the same erasure set for read and write operations. The object is then partitioned into data and parity shards using erasure coding and distributed across drives in that set. Metadata is written inline with the object data in a single atomic operation. There is no external catalog to update, no coordination service to consult, no two phase commit.



Every node in the cluster maintains a complete picture of the distributed topology in memory. Any node can receive a request, compute the hash, and route directly to the correct erasure set without querying a central authority. This is why performance remains consistent as the cluster grows. Adding capacity means adding server pools, which expands the number of available erasure sets. AIStor handles all routing within pools and erasure sets transparently, all while the application continues to use the same end point. There is no rebalancing, no migration jobs, no metadata resharding. A cluster with one pool and a cluster with ten pools operate the same way.

Self-Healing That Keeps You Online

AIStor protects every object using Reed Solomon erasure coding, partitioning data into shards distributed across drives in an erasure set. The number of parity shards determines fault tolerance. In a 16 drive erasure set configured with EC:4 parity, AIStor generates 12 data shards and 4 parity shards. Lose up to four drives in that set and the cluster keeps serving read and write requests without interruption. Configure EC:8 parity and fault tolerance doubles. Half the drives in an erasure set can fail and the data stays available.

When a drive fails, AIStor keeps serving requests. No degraded volume state. No I/O throttling while a RAID controller rebuilds. Healing happens in the background, one object at a time, using any combination of surviving data or parity shards to reconstruct what was lost. The cluster operates at full capacity while affected objects heal incrementally. Traditional RAID can't do this. A single drive failure triggers a volume wide rebuild that consumes I/O bandwidth for hours.

AIStor also catches what other systems miss: AIStor uses an optimized implementation of HighwayHash to detect and heal corrupted objects on the fly, computing a hash on READ and verifying on WRITE to ensure object integrity and protect against bitrot. If media degradation has corrupted a shard, AIStor reconstructs it from parity automatically and repairs the corruption before returning the object to the client. No operator intervention. No tickets. And because metadata uses the same erasure coding as object data, it heals the same way. No separate database to back up, restore, or reconcile.

For details on configuring parity levels and understanding read/write quorum, see the Erasure Coding documentation.

Traditional Object Storage vs MinIO AIStor

	Traditional Object Storage	MinIO AIStor
S3 Compatibility	Gateway translation, partial compatibility, semantic mismatches	Native S3 API, runs unmodified AWS S3 code
Performance at Scale	Degrades as metadata database becomes bottleneck	Consistent latency regardless of cluster size
Scaling	Database sharding, rebalancing, migration windows	Add server pools, no rebalancing, no downtime
Operational Overhead	Separate database to tune, back up, and maintain	Single system, no external dependencies
Data Durability	RAID rebuilds, separate metadata protection	Per object erasure coding, unified protection
Failure Recovery	Volume wide rebuilds, degraded I/O during recovery	Background healing, full performance maintained

Why MinIO AIStor

AIStor is object storage built differently. No external metadata database to become a bottleneck. No gateway translation layer adding latency. No RAID controllers triggering volume wide rebuilds. Every S3 operation executes natively. Every object placement is determined by deterministic hashing. Every failure heals automatically at the object level. The architecture that works at terabytes works at exabytes.

Ready to see it in action?

Visit min.io to learn more. Download AIStor and try it yourself. And request to talk to our team about your environment, and see a demo. We'll show you what's possible.