NeuralMesh Data Reduction

Technical Brief

WEKA NeuralMesh Data Reduction Architecture

AI and HPC workloads have made storage both a cost bottleneck and a performance dependency. Data growth from training pipelines, simulations, and multi-modal datasets is driving petabyte-scale demand, while NVMe costs continue to rise. Traditional data reduction techniques cannot be applied in these environments without introducing latency, throughput variability, and operational instability, making them unsuitable for GPU-based infrastructure.

NeuralMesh Data Reduction combines similarity-based compression with a background-first processing model that removes reduction from the write path. By capturing structural similarity rather than exact duplication and applying reduction asynchronously across the cluster, the system achieves significant capacity savings without compromising deterministic performance. The result is a storage platform that improves both cost efficiency and performance density, enabling higher GPU utilization, reduced data movement, and more scalable infrastructure.

Executive Overview

Data reduction in modern high-performance environments must be fundamentally rethought. Traditional approaches were designed for static, structured datasets and rely on inline compression or exact-match deduplication. These techniques break down in AI and HPC environments, where datasets are not only larger, but structurally different—composed of model checkpoints, multi-modal data, simulations, and continuously evolving datasets with high degrees of similarity rather than exact duplication.

Enterprises are experiencing rapid data growth driven by AI training and inference pipelines, large-scale simulation workloads such as EDA and CFD, genomics research, and multi-modal data processing. These workloads generate petabyte-scale datasets with extreme throughput requirements. Without a new approach to data reduction, storage infrastructure costs scale linearly with data growth, quickly becoming economically unsustainable.

At the same time, modern AI infrastructure operates at unprecedented scale. Clusters span thousands of GPUs and require multi-petabyte to exabyte-scale storage systems. In this

environment, data reduction must operate not just within a node or filesystem, but across the entire cluster, leveraging massive parallelism to identify similarity globally without introducing bottlenecks.

NeuralMesh Data Reduction is designed for this new reality. It combines similarity-based compression with cluster-wide reduction and a background-first architecture that removes reduction from the write path. By distributing reduction work across the cluster and avoiding centralized coordination, the system scales linearly with workload size while preserving deterministic performance.

In production environments, these benefits are already being realized. Field deployments have demonstrated multi-filesystem data reduction ratios of approximately 2.6:1, translating into hundreds of terabytes of saved capacity within a single environment. This reinforces that data reduction is not theoretical, but a proven mechanism for materially improving storage efficiency at scale.

Crucially, this efficiency is achieved without compromising performance. Because reduction processing is decoupled from the write path, write performance remains effectively unchanged, with observed overhead typically under ~5%. Sustained throughput is maintained, ensuring that GPUs remain fully utilized and infrastructure efficiency is preserved.

The Storage Constraint in AI Infrastructure

The economics of flash storage have shifted significantly. Enterprise NVMe prices have increased sharply since 2025, with supply constraints expected to persist into 2027. At the same time, AI infrastructure has expanded rapidly, with clusters scaling to tens of thousands of GPUs and requiring multiple petabytes of high performance storage.

At current pricing levels, storage is no longer a secondary component of infrastructure cost. A multi-petabyte flash deployment represents a multi-million dollar investment before accounting for compute, networking, and facility costs. As clusters scale, storage becomes a primary budget constraint.

Moreover, the nature of AI workloads amplifies storage demand. Model training pipelines generate large volumes of checkpoint data, intermediate outputs, and replicated datasets. These workloads are characterized by high throughput requirements and frequent write operations, which place sustained pressure on the storage system.

Therefore, any reduction in required physical capacity directly translates into meaningful savings. But, reduction must be achieved without compromising performance. Any degradation in throughput or latency reduces effective GPU utilization and increases overall training time.

Limitations of Traditional Approaches

Traditional storage platforms implement data reduction using inline compression, global deduplication, or a combination of both. These techniques are quite effective in environments such as backup or archival storage, where throughput requirements are moderate and latency variability is acceptable.

In performance sensitive environments, these approaches introduce several limitations.

- ☐ Inline deduplication requires global coordination and metadata lookups during write operations. This increases latency and introduces variability in write performance. As systems scale, the metadata structures required to support deduplication grow significantly, creating additional pressure on memory and rebuild workflows.
- ☐ Fixed block compression operates on predefined chunk sizes and cannot effectively capture redundancy across block boundaries. While larger compression dictionaries can improve efficiency, they also increase decompression cost, which directly impacts read performance.
- ☐ Exact match deduplication relies on cryptographic hashes and is effective only when data is identical. Modern workloads such as AI training, simulations, and analytics produce data that is structurally similar but not identical. Small variations in offsets or values prevent traditional deduplication from identifying redundancy.
- ☐ Post-process reduction techniques avoid write path penalties but introduce unpredictable background activity that competes with foreground I O. This can result in performance variability under sustained load.

These limitations make traditional approaches poorly suited for AI and HPC environments where sustained throughput, low latency, and predictable performance are critical.

NeurealMesh Data Reduction

WEKA approaches data reduction as a performance safe optimization rather than a primary optimization target. The system is designed with the assumption that foreground I/O must remain deterministic regardless of reduction activity.

The architecture is guided by several principles:

- ☐ Reduction operations must not introduce global coordination on the write path.
- ☐ The system must scale linearly with cluster size and workload intensity.
- ☐ Metadata growth must remain bounded to avoid long term scalability issues.
- ☐ Background processing must be isolated in a way that prevents interference with application performance.

These principles lead to a separation between data ingestion and reduction processing. Writes are handled at full device speed without waiting for compression or deduplication decisions. Reduction work is performed asynchronously, allowing the system to amortize computational cost over time.

The result is an architecture that delivers capacity efficiency while preserving the performance characteristics required for demanding workloads.

Similarity-Based Compression and Data Layout

At the core of NeuralMeshData Reduction is a similarity-based compression model that extends beyond traditional deduplication techniques. Instead of relying on exact matches, the system analyzes structural characteristics of data blocks and identifies groups of blocks that can be compressed efficiently together.

Multiple similarity hashes are derived for each block. These hashes describe different aspects of the data and allow the system to identify relationships between blocks that are not byte identical. By grouping similar blocks, the system constructs shared compression dictionaries that capture common patterns across the dataset.

Compressed data is stored using a reference and delta model. Reference blocks serve as the basis for compression dictionaries, while delta blocks encode differences relative to these references. The system places references and associated deltas in close proximity to minimize read amplification and maintain high throughput during decompression.

During read operations, this model requires reconstruction of data by retrieving both reference blocks and associated delta blocks, followed by decompression. This can increase the number of I/O operations and CPU utilization compared to non-reduced reads.

However, this overhead is offset by the reduction in physical data that must be read from storage, often resulting in comparable or improved effective throughput depending on workload characteristics.

This approach enables significantly higher reduction ratios on machine generated and semi structured data compared to traditional methods, while keeping metadata overhead manageable.

Techniques to Eliminate Redundancy at Scale

The architectural choices described above are realized through a set of complementary techniques that operate across the cluster. NeuralMesh Data Reduction combines block-variable differential compression with cross-filesystem deduplication to identify and eliminate redundancy at scale. These techniques are not independent features, but direct consequences of the system's similarity-based and background-first design.

- Block-variable differential compression operates at a fine-grained level, analyzing data in approximately 4K segments while dynamically adjusting block boundaries. Rather than compressing fixed chunks in isolation, the system groups structurally similar blocks and applies shared compression dictionaries. This approach allows redundancy to be captured across block boundaries and across datasets, which is particularly effective for binary and semi-structured data common in AI and HPC environments.
- Cross-filesystem deduplication extends this model beyond individual datasets. During ingest, the system computes similarity fingerprints that describe the structure of each block. These fingerprints are later used to identify related data across all data reduction enabled filesystems in the cluster. As a result, redundancy is eliminated not only within a single workload, but across model checkpoints, replicated training datasets, container layers, and environment snapshots.

Together, these techniques enable the system to capture similarity rather than exact duplication, which is essential for achieving high reduction efficiency on modern machine-generated data.

Separation of Control Plane and Data Plane

WEKA maintains a clear separation between control plane and data plane responsibilities within the data reduction architecture.

- The data plane is responsible for performing inline lightweight operations such as zero block detection and ensuring that read and write paths remain deterministic. It also handles decompression during read operations, maintaining high throughput through parallel processing across drive cores.
- The control plane manages capacity accounting, snapshot policies, and observability. It tracks logical and physical usage independently and provides accurate reporting for operational planning. Importantly, control plane activities do not interfere with data path operations, ensuring that management functions do not impact application performance.

This separation contributes to system stability and simplifies operational management at scale.

Performance Characteristics

NeuralMesh Data Reduction is designed to operate without introducing measurable impact on foreground write performance. Because compression and deduplication decisions are deferred to background processes, writes proceed at full NVMe speed. In practice, write performance impact is negligible, typically observed at less than ~5% compared to non–data-reduced operation, and often effectively identical from an application perspective.

Background reduction activity consumes CPU resources, but it is scheduled with lower priority relative to user I/O. This allows the system to adapt to available compute capacity without affecting

application workloads. In environments operating near peak utilization, careful consideration of CPU headroom is recommended to ensure that background processing does not contend with foreground tasks.

Decompression performance is optimized for parallel execution. Each drive core is capable of delivering high throughput during decompression, enabling the system to sustain aggregate read bandwidth at cluster scale. The placement of reference and delta blocks minimizes cross node dependencies and ensures efficient data access.

Read performance reflects a balance between reduced physical data transfer and additional reconstruction work. In worst-case scenarios, per-drive throughput is approximately 1.6–1.8 GB/s, while typical real-world workloads often achieve closer to ~3 GB/s depending on data characteristics and reduction ratios.

NeuralMesh Data Reduction Processing Pipeline

Data reduction is a post-process, background activity. This is an intentional architectural choice that preserves write performance:

Ingest (real-time)

Data enters the system through a high performance ingest path. During this phase, writes are stored uncompressed to maintain maximum throughput at full NVMe speed. At the same time, the system computes similarity hashes for each block. These fingerprints capture structural characteristics that will later be used to identify related data.

Clusterization (background)

Once sufficient data has accumulated, the system performs background clustering. The system groups similar 4K blocks across all DR-enabled filesystems once sufficient data accumulates, running at lower priority than user I/O.

Compression & write-back

Grouped blocks are compressed, and the results written back to the drive tier. Original uncompressed blocks are marked for deletion.

Defragmentation

Invalidated uncompressed blocks are permanently reclaimed when sufficient quantities accumulate, returning physical capacity.

Decompression on read

Drive containers decompress data inline, transparently, before serving it to clients. Applications require no modification.

Applications are not aware of the reduction process and do not require modification. This pipeline ensures that write performance remains unaffected while enabling continuous optimization of stored data.

Data Reduction in AI Workflows

AI training environments present a workload profile that is particularly favorable for data reduction. By reducing the physical footprint of these workloads, data reduction allows more data to reside within the high performance storage tier. This supports faster iteration cycles, improved collaboration across teams, and more efficient use of infrastructure resources.

NeuralMesh Data Reduction for AI Training	
Model checkpointing cadence	Frequent checkpointing produces multiple versions of model weights with minimal variation. Shared base models and datasets are replicated across teams and experiments, creating additional redundancy. Training runs save checkpoints every N steps; adjacent checkpoints differ by only a small fraction of total weights. A 70B parameter model checkpoint at BF16 precision occupies ~140 GB — and consecutive saves may differ by less than 5%.
Shared base models	Multiple teams fine-tuning from the same foundation model (Llama, Mistral, Falcon) store near-identical base weight tensors. Cross-filesystem deduplication eliminates this redundancy automatically.
Dataset repetition	Training datasets are frequently duplicated across experiments, versions, and preprocessing pipelines. Deduplication captures these savings transparently.
Container and environment layers	Containerized environments introduce further duplication through repeated storage of software layers and dependencies. These patterns create a high degree of similarity across the dataset, enabling efficient compression and deduplication. Docker images, Python environments, and CUDA toolkit files stored on the cluster are identical across nodes and jobs.

Reduction Efficiency Across Workloads

Achievable ratios depend heavily on workload characteristics: repetition, similarity, and data type. It is critical to understand that not all datasets benefit from data reduction. Workloads consisting of already-compressed data, encrypted content, or inherently random data may see minimal to no reduction. In some environments, existing datasets may yield little benefit, making upfront workload qualification essential. The effectiveness of data reduction depends on the characteristics of the workload:

- ☐ Data that exhibits repetition, structural similarity, or incremental changes over time is particularly well suited for similarity-based compression.
- ☐ AI and machine learning workloads often achieve high reduction ratios due to the nature of model checkpointing (snapshots of weights saved frequently during training runs) and dataset reuse. Consecutive checkpoints contain only small differences relative to previous versions, exhibiting high cross-version similarity, and making them excellent candidates for deduplication. A training run saving checkpoints every 100 steps may write dozens of nearly-identical weight tensors. Across teams sharing a cluster, the same base models, datasets, and Docker layers are duplicated further.
- ☐ Electronic design automation (EDA) workloads also benefit from high levels of reduction due to repetitive simulation outputs and structured data patterns.
- ☐ Other domains such as genomics, analytics, and time series data exhibit moderate to strong reduction depending on data structure and processing pipelines.

Conversely, data that is already compressed, encrypted, or inherently random does not benefit significantly from reduction and is better suited for separate storage configurations.

Field-Proven Results

Deployments of NeuralMesh Data Reduction have demonstrated measurable results across real-world environments. In one representative production deployment spanning multiple filesystems, the system achieved approximately 2.6:1 data reduction, resulting in roughly 765 TB of saved physical capacity.

These results highlight the practical impact of similarity-based compression in enterprise environments. However, outcomes vary significantly based on workload characteristics, reinforcing the importance of dataset qualification and pre-deployment estimation.

Data Reduction Ratios by Workload

The table below reflects WEKA's published benchmarks and field experience across workload categories:

Workload Type	Typical Reduction Ratio	Notes
AI / ML model training data	**~6X**	Checkpoints, intermediate activations, embeddings
EDA (electronic design auto.)	**~8X**	Highly repetitive simulation outputs & netlists
Genomics & life sciences	**2–3X**	FASTQ, VCF, BAM files; redundancy across samples
Media & entertainment	**~2X**	Image sequences; video frames with scene similarity
Databases & code repositories	**3–5X**	Structured data, source trees, build artifacts
Sensor data / IoT / telemetry	**4–6X**	Time-series logs with high temporal correlation

Achievable ratios depend heavily on workload characteristics — repetition, similarity, and data type.

Sources: WEKA Blog — Data Reduction Efficiency; WEKA Documentation

Economic Impact and Infrastructure Efficiency

Data reduction directly influences the economics of high performance storage deployments. By reducing the physical capacity required to store a given dataset, the system lowers capital expenditure on storage infrastructure like flash.

Economic Impact and Infrastructure Efficiency of NeuralMesh Data Reduction	
`Significant cost savings`	In large scale deployments, even modest reduction ratios translate into substantial savings. A reduction factor of three effectively reduces required raw capacity to one third of the original footprint. At current NVMe pricing levels, this represents significant cost avoidance in multi-petabyte environments.
`Lower power consumption`	Beyond direct cost savings, reduced physical capacity leads to lower power consumption, decreased rack space requirements, and simplified operational management. Fewer drives reduce the

	number of potential failure domains and improve overall system reliability.
Faster job completion times and improved utilization	Data reduction also contributes to performance efficiency. Smaller physical datasets reduce the volume of data movement during checkpointing, replication, and rebuild operations. This results in faster job completion times and improved utilization of compute resources.

Beyond Data Reduction: Holistic Data Efficiency

NeuralMesh Data Reduction is one component of a broader, integrated data efficiency model within the WEKA platform. In modern AI and HPC environments, capacity efficiency is not achieved through a single technique, but through the coordinated use of multiple mechanisms that operate across the cluster and share a common data architecture.

Thin provisioning in WEKA is implemented at the filesystem level, allowing logical capacity to significantly exceed the underlying physical allocation. Capacity is consumed only as data is written, while the system continuously tracks both logical and physical usage independently. This enables organizations to oversubscribe storage safely based on expected data reduction ratios and workload behavior, maximizing utilization of expensive NVMe resources without requiring upfront overprovisioning.

Snapshots in WEKA are inherently space-efficient, leveraging the platform's shared block architecture to avoid duplicating unchanged data. Snapshots create point-in-time views by preserving references to existing data blocks, while only incremental changes consume additional capacity. Because snapshots operate on the same underlying data structures used for reduction, they compound efficiency gains—multiple versions of datasets, model checkpoints, or environments can be retained with minimal incremental overhead.

These capabilities are tightly integrated with NeuralMesh Data Reduction. Thin provisioning amplifies the impact of reduction by allowing logical capacity expansion beyond physical limits, while snapshots exploit structural similarity across versions of data, further increasing effective capacity savings. Together, they form a unified efficiency model that improves power density, reduces physical footprint, and lowers total cost of ownership.

Importantly, these features operate within the same distributed, parallel architecture as the rest of the WEKA system. Efficiency mechanisms are applied cluster-wide, scale linearly with system size, and do not introduce centralized bottlenecks or performance penalties. This ensures that capacity optimization does not come at the expense of throughput, latency, or scalability.

Example Economic Case

Let's consider a cost savings example. If data reduction delivers 3X improvement, you need one-third the raw NVMe capacity to serve the same logical footprint. In an environment where enterprise NVMe drives cost $500–1,000/TB, this translates directly to capital expenditure reduction. This is NOT theoretical savings, but fewer drives purchased.

The table below models representative savings at 3X effective reduction (a conservative assumption for AI/ML workloads, which typically achieve 4–6X) at a baseline of $1,000/TB enterprise NVMe.

Scenario	Raw Capacity	Raw NVMe Cost[1]	With 3× DR	Savings
Single AI training pod (500 TB logical)	500 TB	~$500K	~167 TB raw	**~$330K**
Mid-size GPU cluster (2 PB logical)	2 PB	~$2.0M	~667 TB raw	**~$1.3M**
AI Gigafactory tier (10 PB logical)	10 PB	~$10.0M	~3.3 PB raw	**~$6.7M**

[1] *Based on ~$1,000/TB enterprise NVMe (current 2026 market rate for high-capacity data center NVMe). See: storagediskprices.com*

Beyond raw capital savings, data reduction delivers operational benefits that compound over time: fewer drives means fewer failure domains to manage, lower power draw, reduced rack space, and smaller physical footprint — all of which matter at the scale of a 100,000-GPU Gigafactory.

Deployment Considerations and Best Practices

Effective use of data reduction requires alignment between workload characteristics and system configuration.

- ☐ Filesystems intended for reduction should be designed with thin provisioning enabled (minimum and maximum capacity thresholds set)
- ☐ Avoid features such as encryption or object tiering that interfere with similarity analysis
- ☐ A valid Data Reduction license (DPO-based) applied to the cluster

These constraints are by design: thin provisioning is necessary to represent logical capacity above the physical pool; tiered filesystems rely on object-store capacity that is managed separately; and encryption precludes the block-similarity analysis that makes deduplication effective.

Validate Savings Before Committing

WEKA provides a Data Reduction Estimation Tool that analyzes a representative sample of your actual data and projects the expected savings ratio before you enable the feature in production.

The estimation workflow should be considered a required step prior to enabling data reduction in production environments. The tool enables what-if analysis on real datasets and provides the foundation for setting expectations, sizing infrastructure, and informing procurement decisions.

As the capability matures, estimation and validation are expected to evolve toward more formalized guarantees of reduction efficiency based on workload characteristics.

CPU and Resource Planning

Data reduction introduces additional CPU demand for both background processing and decompression during reads. As a general guideline, environments should provision approximately one CPU core per NVMe drive, with some configurations benefiting from up to a 2:1 CPU-to-drive ratio depending on workload intensity and performance requirements.

Careful evaluation of CPU headroom is recommended, particularly in environments operating near peak I/O throughput.

Design for Data Reduction from Day One

Retrofitting data reduction onto existing filesystems is possible but requires downtime for reconfiguration. The optimal approach is to design the filesystem layout for DR at cluster deployment:

- ☐ Enable thin provisioning during filesystem creation — not as an afterthought
- ☐ Separate DR-eligible data (checkpoints, datasets, logs) from data that should be encrypted at rest (secrets, keys, PII) into distinct filesystems
- ☐ Avoid tiering on DR-enabled filesystems — keep DR workloads on the all-flash tier
- ☐ Plan logical-to-physical oversubscription ratios based on your estimation tool output, with a conservative buffer (e.g. if the tool predicts 4×, plan for 3× to account for data evolution)

Monitor and Tune Ongoing Operations

Data reduction is self-managing, but production visibility is essential:

- ☐ Track CPU utilization in background processes — the DR pipeline shares compute with user I/O; monitor for contention during peak training runs

- ☐ Watch the ingest rate (blocks/second) and fingerprint computation metrics via Grafana dashboards — a sudden drop may indicate data patterns resistant to compression (e.g. already-compressed data like zip files, pre-compressed model weights from certain frameworks)
- ☐ Use weka fs status to check the current data_reduction_ratio and data_reduction_saved_bytes per filesystem
- ☐ The background DR process can be paused on-demand for latency-sensitive windows (e.g. final benchmark runs, acceptance tests) without affecting user data

Understand What Data Reduces Well — and What Doesn't

Not all data is suitable for compression. The following data types deliver minimal or no benefit and should be kept on non-DR filesystems to avoid wasting CPU on fingerprinting:

- ☐ Already-compressed files: JPEG, MP4, ZIP, gzip archives, compressed Parquet files
- ☐ Encrypted volumes and data at rest (encryption precludes DR by design)
- ☐ True random or cryptographic data
- ☐ Certain pre-compressed deep learning weight formats from specific training frameworks

Field Learnings

Deployments of NeuralMesh Data Reduction in production enterprise environments have surfaced several important operational learnings:

NeuralMesh Data Reduction: Field Report	
Performance Sensitivity at Limits	On clusters operating at or near peak NVMe I/O throughput, the background DR pipeline can become a performance variable. This is most relevant during sustained bulk-write phases (large checkpoint saves, dataset ingestion). The best-practice response is to characterize the cluster's CPU headroom under peak I/O before enabling DR, and optionally configure monitoring alerts for background process CPU utilization. On clusters with headroom, DR runs transparently without user impact.

License Entitlement and Capacity Planning	The data reduction license (DEO, now unified under DPO in WEKA 5.1.x) specifies the physical capacity entitled for DR. An important implementation detail: DEO, DPO, and DTO license fields all map to OBS (object-store equivalent) capacity in the license model, not to separate entitlement tracks. When planning cluster expansion driven by DR-enabled capacity needs, update the license entitlement to match the new physical capacity, not the logical provisioned size. Confusion between logical and physical capacity in license management has caused operational friction in field deployments.
CPU headroom	In high-performance environments, particularly those pushing the physical limits of NVMe drives, enabling data reduction increases background CPU load. On clusters running sustained peak I/O workloads, characterize CPU headroom carefully before enabling. The background process runs at lower priority but still competes for CPU cycles on heavily loaded nodes.
The Value of the Gate	Data reduction requires an explicit license and SE or CS involvement before a customer can enable it on a production cluster. While this may appear as friction, it functions as a valuable guardrail: it ensures that a WEKA engineer validates the workload, confirms filesystem design, and confirms expected ratios using the estimation tool before commitment. This prevents suboptimal deployments — for example, inadvertently enabling DR on a filesystem serving compressed media — and creates a natural customer success touchpoint.
Positioned as capacity multiplier	Data reduction should be positioned in the pre-sales conversation as a capacity multiplier, not a post-procurement optimization. Customers sizing storage for a GPU cluster should be shown — using the estimation tool on representative data — how data reduction changes the bill of materials. This positions WEKA competitively against all-flash incumbents that cannot offer equivalent space efficiency at the same performance level.

Summary

The rapid growth of AI infrastructure and the rising cost of high performance storage have made data reduction a critical capability. However, not all approaches are suitable for performance sensitive environments.

NeuralMesh Data Reduction addresses this challenge through an architecture that balances efficiency with deterministic performance. By combining similarity-based compression, cross-filesystem deduplication, and background processing, the system delivers meaningful capacity savings without compromising throughput or latency.

For organizations deploying large scale AI and HPC environments, data reduction is not simply an optimization. It is a mechanism for aligning storage economics with the demands of modern workloads, enabling sustainable growth and efficient use of infrastructure.

Take the Next Step with NeuralMesh

Dig further into NeuralMesh by checking out our Reference Architecture. If you are interested in discussing how NeuralMesh can help you, click here to contact us.

Resources:

- ☐ WEKA Documentation – Filesystems, Object Stores, and Data Reduction
- ☐ WEKA Blog – How Data Reduction Can Improve Operational Efficiency
- ☐ WEKA – What is Data Reduction?