WEKA®

NeuralMesh™ Architecture

White Paper

Summary

and Contents

Contents

Introduction

WEKA was founded on the idea that the data infrastructure which powered the internet era would not stand up to the demands of the AI era. A storage system that could grow and scale with the rapid advancements in GPUs, networking and storage media requires a radical new approach to how data is written, stored, protected and retrieved. Current file systems have only provided incremental improvements in design - largely relying on hardware advancements - but ultimately fail to meet the ever-increasing demands of data-intensive computing.

Within the Enterprise, storage solutions are highly fragmented according to the application needs, and companies have ended up with many different solutions to manage specific applications in their environment leading to management complexity. Additionally, the major innovations in performance have come with a hardware dependency to run as specified, making it impossible to experience comparable performance in the cloud as on premises.

In building a new storage system to service the AI era, the founders of WEKA deconstructed the entire data path all the way from user space down to the physical storage media and exposed all the major performance bottlenecks found in current architectures. In turn, they solved for them with new unique methods and algorithms that now compose our large and growing portfolio of patents – over 140 have been issued to date.

WEKA took on the most daunting aspects of the data path and engineered them from the ground up to deliver the lowest latency, most performant storage system at any scale. Other solutions have implemented incremental efficiencies that address some or part of the problem, however no other solution has implemented the level of innovation that WEKA can demonstrate. The engineering innovations that WEKA undertook when building the underlying data management system are foundational to the design and cannot be retrofit - or bolted on – to legacy architectures.

NeuralMesh™ by WEKA, is a software-only, high-performance, container-native storage system built for AI and data-intensive workloads at scale. It delivers low-latency, high-throughput data access with a microservices-based architecture that scales linearly and becomes more efficient as system size grows. NeuralMesh runs on standard x86 infrastructure across on-premises, public cloud, and hybrid cloud environments without the need for custom hardware configurations.

The design philosophy behind NeuralMesh was to create a single storage architecture that runs on-premises or in the public cloud with the performance of all-flash arrays, the simplicity and feature set of Network-Attached Storage (NAS), and the scalability and economics of the cloud in a single unified system.

NeuralMesh is built to address demanding environments that need shareable storage with low-latency, high-performance, and cloud scalability.

Example use cases include:

- Artificial Intelligence (AI) Inferencing
- Agentic AI
- AI Machine Learning (ML), including AIOps and MLOps
- Life sciences including genomics, Cryo-EM, pharmacometrics (NONMEM, PsN)
- Financial trading, including back-testing, time-series analysis, and risk management
- Engineering DevOps
- Electronic Design and Automation (EDA)
- Media rendering and visual effects (VFX)
- High-Performance Computing (HPC)
- GPU pipeline acceleration

By leveraging existing technologies in new ways and augmenting them with engineering innovations, NeuralMesh delivers a more powerful and simpler solution that would have traditionally required several disparate storage systems. The resulting software solution delivers high performance for all workloads (big and small files, reads and writes, random, sequential, and metadata heavy).

NeuralMesh is based on a fully distributed parallel file system that was written entirely from scratch to deliver the highest performance file services by leveraging NVMe (non-volatile memory express) flash. The software integrates with object storage, combining near memory-like flash performance with cost-effective economics across on-premises and cloud environments. This seamless integration transparently extends the namespace, enabling use cases like archiving and data protection—without requiring manual data migration, external tools, or complex scripting. An intuitive graphical user interface allows a single administrator to manage exabytes of data quickly and easily without any specialized storage training.

Key Design Decisions

When WEKA was founded, three major technological innovations were emerging. Containers, still in their infancy, enabled application sandboxing without the overhead of hardware virtualization. WEKA's founders recognized the potential of this deployment model—later known as a microservices architecture—which allowed flexible, distributed storage services to run seamlessly across on-premises and cloud infrastructure. This microservices approach adapted to a storage environment was revolutionary because it meant that storage features and functions could be deployed as microservices and run across any physical servers regardless of location.

The second major innovation was NVMe. Traditional interfaces like SAS (serial attached SCSI) and SATA (serial ATA) were optimized over decades for mechanical hard drives, but they introduced bottlenecks when paired with modern flash media. NVMe, by contrast, leverages high-speed PCIe (peripheral component interconnect express) lanes to provide a streamlined, parallelized path to flash storage, dramatically reducing latency and increasing throughput. WEKA identified the potential of NVMe early and built its platform from the ground up to take full advantage of it. By aligning its architecture to the 4K block size and direct access model of NVMe solid state disk (SSD), WEKA delivers flash-optimized performance on any standard x86 server—on-premises or in the cloud—without specialized hardware or added complexity.

The third enabler, following containerization and NVMe, was high-speed networking. At the time, 16 Gb Fibre Channel delivered 0.5–1 ms latency, while local SSDs offered 150–300 μs—making local storage the preferred option for low-latency workloads. But the introduction of 100 GbE, with just 1–5 μs latency, made network overhead negligible and data locality irrelevant; data could now be read faster over the network than from a local drive. WEKA's founders saw this shift and built a system that enables high-performance, low-latency data access across any network attached server. As Ethernet performance continues to outpace Fibre Channel—now accelerating toward 800 GbE—this architectural decision becomes even more relevant for AI-scale workloads that span hundreds to thousands of GPU servers.

WEKA's founders recognized that most storage performance gains over the past decades came from hardware advances layered onto legacy file systems. Yet the software itself remained constrained by fundamental architectural bottlenecks—especially evident at scale, with ever-growing I/O queues and massive metadata operations, performance ultimately degraded. NeuralMesh was purpose-built to eliminate these software bottlenecks while fully leveraging modern hardware to deliver consistent, high-speed performance at scale.

100% Software

Software-defined storage has often been associated with trade-offs in performance. WEKA took a different approach—intentionally designing a containerized software solution that allows the system to grow and evolve with technology advancements in compute, memory, networking, and storage media. By eliminating dependencies on specialty hardware, accelerators, offload engines, RAID engines, networking, or storage media to achieve its leading performance, WEKA engineers were driven to optimize code at all levels of the I/O stack – from user space to physical storage media – to ensure maximum performance and flexibility. NeuralMesh runs on standard server architectures from any original equipment manufacturer (OEM), public cloud vendor or Neoclouds. The exact same software will run on-premises and in the cloud without the need for re-engineering or software refactoring. From day one, WEKA is fully capable of working within any standard data center environment or cloud with full portability between environments. So, you get to choose where you want your data to reside, on-premises, co-located, public cloud, specialty GPU cloud, or a hybrid environment.

Designed for NVMe

AI accelerated compute requires storage that matches memory speed and latency. WEKA anticipated NVMe solid state disks would become the standard for large scale deployment and designed its storage system to take full advantage of the architecture. By aligning with NVMe's native 4KB block size and writing data in 4KB chunks WEKA eliminated translation overhead. In contrast, legacy systems, built for spinning disks with large block sizes, must repackage data into smaller chunks for every transaction, increasing CPU load and reducing performance—especially for small files and metadata, where they can experience 4x–16x higher read latency.

Lowest Latency for Highest Performance

Every I/O operation that involves the data path introduces latency to the overall system responsiveness. As systems scale, the cumulative impact of each microsecond latency hit has a significant impact as I/O queues grow and parallel I/O becomes serialized resulting in performance degradation. WEKA implemented several innovations to ensure lowest latency as the system scales.

Remove Kernel Overhead

Operating systems are excellent at maintaining compatibility across lots of different hardware choices, but that comes at the expense of performance. For applications that require significant interaction with the data, the Linux kernel is a major bottleneck. The kernel coordinates the movement of data between applications and the physical storage media, but since it is only allocated a small percentage of overall system resources, repeated requests to read or write data to and from storage begin to queue up. Highly parallelized requests become serialized as they are serviced by the kernel, resulting in latency spikes and inconsistent performance. NeuralMesh bypasses the operating system kernel completely resulting in a huge performance gain. WEKA wrote its own data-centric operating system that performs the data-specific activities that other solutions rely on the kernel to execute.

Memory management across CPU cores also places a significant load on the system kernel to ensure cache coherency. Weka manages cache coherency independent of the kernel by pinning device memory directly to user space using SPDK (storage performance development kit), an open-standard development kit for high-performance storage applications.

The kernel is also a latency inducing bottleneck for network traffic as it is interrupt driven. Applications can experience latency waiting for the Kernel to "wake up" and respond to a request. By contrast, WEKA processors run in polling mode, meaning they are constantly polling for status completions to see if there is work to be done, ensuring minimum latency. Additionally, WEKA uses DPDK (data plane development kit), an open standard, to bypass the kernel for network activity, thereby removing yet another performance roadblock.

Eliminate Software Layering

Many of the high-performance file systems, particularly the popular parallel file systems layer the file software over a block storage layer, and while they may be optimized for a global data view, they have no control over lower-level tasks such as data placement and data protection. These discrete entities require significant overhead to coordinate with the file system, impacting performance. In contrast, WEKA vertically integrates every aspect of the file system from the lowest level for maximum performance and scalability. Fewer software layers directly translate to lower latency and much greater overall system simplicity. This also allowed WEKA to innovate around data placement for maximum resiliency and lightning fast rebuilds. When drives and systems fail, WEKA's integrated design ensures rebuilds happen quickly, and performance remains consistently high throughout the degraded state.

Modern Data Structures

Computer systems utilize strategies to maintain an index of where data lies and allow for efficient access and retrieval. The predominant solution utilized by other file systems are B-tree structures utilizing linked lists for storing large data sets. The structure is a hierarchy with each layer pointing to sub structures underneath. To modify any part of the structure, the entire hierarchy has to be read into memory, modified and written back out in entirety. With traditional file systems the process cannot be distributed across compute resources and the simple process of deleting a file can bring the file system to its knees, and the problem grows worse with bigger data sets. WEKA took a unique approach to directory structures that allows for enormous directory sizes yet maintains very fast access, one can have billions of files in a single directory with no performance degradation. The solution acts more like a hash table, so we can take the address range of that structure and spread it across multiple cores allowing for far greater parallelism. If multiple users want to create a file at the same time inside a directory, then it does not require the entire structure to be locked, just the part of it that applies to the specific user.

Fully Distributed Data and Metadata

Most high-performance file systems isolate metadata services to dedicated metadata servers and data services (read/write) to dedicated data servers to ensure that I/O for metadata does not interrupt I/O for data. The theory behind this approach was that requests for metadata services tend to be unpredictable, while requests for data services are constant. However, it is difficult to keep the two systems balanced because the data servers and the metadata servers cannot share processing resources, and each system must be sized and scaled separately based on yet to be known data sets and access patterns. As data sets and I/O load grow, it becomes difficult to balance the two separate systems and performance suffers. WEKA fully distributes data and metadata services across every core in the system. Using our proprietary virtual metadata server, every compute resource in the system can handle every request for data or metadata. And when storage resources expand, metadata services expand in tandem to ensure that performance is maintained. WEKA preserves the balance of data layout across servers based on latency histograms to ensure there are no hot or idle servers. This is why WEKA can demonstrate perfectly linear performance scaling as the storage platform grows.

Containerized Deployment

WEKA anticipated that containers would become a foundational shift in virtualization and built a microservices orientation into the platform from the start. With NeuralMesh the software is now fully containerized from user space to the physical infrastructure and delivers its storage services through the Kubernetes Operator. Containers are particularly valuable because they provide a predictable, protected, sandboxed execution environment—exactly what is needed for consistent, reliable performance at scale. This architecture also enables fast, non-disruptive upgrades by restarting individual services with new code, keeping the system available and performant throughout. It also forms the basis of WEKA's Multitenancy model, providing strong tenant isolation and enhanced security across shared infrastructure.

How It Works

Unlike all other scale-out systems, NeuralMesh is a vertically integrated design that has been optimized for highest performance and lowest latency. The design goal is to consistently deliver higher performance than can be achieved by a local file system on local storage, yet allowing the platform to scale to Exabytes of data with no performance impact. The traditional layered approach of legacy architectures introduces latency, I/O coordination bottlenecks and scaling limits which cannot be solved with hardware. Instead, WEKA has architected its storage system as a series of data processes that coordinate and work together and in unison for optimal performance and scalability.

A key innovation was to containerize all the data processes and manage the system as a series of microservices based on Kubernetes. Capabilities such as access control, data reduction, snapshots, auditing, protocols, and more are deployed as independent containerized microservice communicating through well-defined, hardened APIs. This service-oriented design creates a clear boundary between components, enabling unprecedented flexibility in how storage resources are deployed, scaled, and managed. By leveraging the Kubernetes Operator, NeuralMesh abstracts away deployment complexity, allowing customers to declare their desired storage outcomes rather than managing individual containers. This approach delivers secure multitenancy and microsecond latency at exabyte scale.

At the core is a sophisticated resource management system that divides server resources into "service slots" which can be dynamically allocated based on workload needs. The architecture distinguishes between mandatory services (drive, compute, telemetry) and optional services (protocol handlers, data services), with clear policies for scaling and resource requirements. This approach allows WEKA to add a new level of flexibility: clusters can now auto-scale individual services based on changing demands, resources can be rebalanced as clusters expand, and new services can be deployed without disrupting existing workloads.

A NeuralMesh storage system is composed of a series of 'failure domains". A failure domain is a software construct and is typically a single physical server, although it can be configured at the drive or rack level too. For the purposes of this document, we will consider a failure domain as a single server.

WEKA's Unique Data Path

Some of the core innovations of NeuralMesh are built around the data path from user space to the physical media where data persists. The following section describes the I/O flow for the NeuralMesh POSIX interface, which is the recommended operational mode for highest performance. Figure 1 below provides an overview of the data path from the application layer all the way to the physical persistent media layer. NeuralMesh leverages virtualization and other low-level Linux container techniques such as Linux cgroups to run its own RTOS (Real-Time Operating System) in user space. The NeuralMesh RTOS coexists with the server's Linux operating system and carves away cores, NVMe devices and network ports (or fractions thereof) out of the Linux operating system. NeuralMesh manages its assigned resources (CPU cores, memory regions, network interface cards, and SSDs) to provide process scheduling, memory management, and to control the I/O and networking stacks. Leveraging its container design, NeuralMesh takes physical servers and breaks them into hundreds or thousands of virtual servers, working in unison to process data and metadata.

By not relying on the server's Linux kernel, the system's core components execute in user space in a Linux container (LXC), effectively eliminating time-sharing and other kernel-specific dependencies.

The physical servers making up a NeuralMesh system are members of a cluster. Each server includes multiple containers running some or all the following processes:

- Frontend (FE) processes for POSIX client access, I/O communication to the compute and drive process.
- Frontend processes for non-POSIX client access such as NFS, SMB and S3
- Compute processes for I/O from the client, file system functions and clustering
- Drive processes for I/O to and from the physical drive
- Management processes for managing the overall cluster
- Telemetry processes for auditing and logging

Figure 1 below provides an overview of the I/O path from the application layer all the way to the physical persistent media layer.

While parallel file systems have separated data and metadata services, legacy scale-out NAS solutions maintain a 1:1 relationship between the number of physical servers and the number of metadata servers. This design can quickly develop hot spots as one server can be heavily loaded while all the other metadata servers in the system are idle. In the NeuralMesh design there is no hard 1:1 relationship between container types in a failure domain, and the number and types of containers can be configured as a best-fit to the hardware provided as well. For example, WEKA 10 physical servers could have 10, 20 or 50 virtual metadata servers to execute the work, allowing for huge parallelism - and it is entirely configurable.

The NeuralMesh Virtual File System (VFS) kernel driver provides the POSIX file system interface to applications. Using the kernel driver provides significantly higher performance than what can be achieved using a FUSE user-space driver, and it allows applications that require full POSIX compatibility to run on a shared storage.

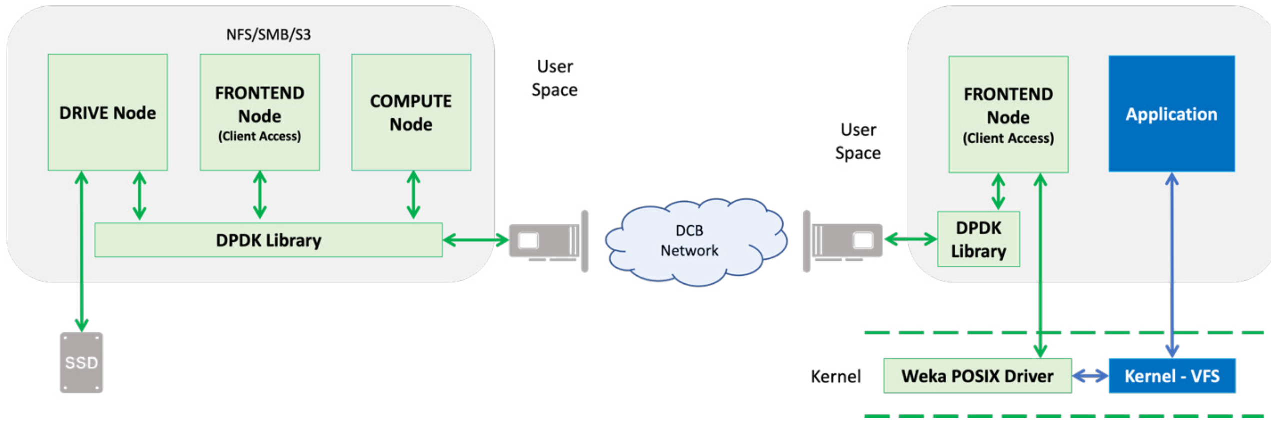

FIG. 1 The IO Data Path

POSIX Driver

Interaction with a WEKA system begins in the Linux Kernel via the kernel VFS. NeuralMesh implements its own kernel module which communicates with the application through the Frontend container. The POSIX driver intercepts POSIX system calls and forwards them to the Frontend container. When an application wants to perform an operation on a piece of data (open, write etc.) it makes a system call to the Linux VFS layer. The NeuralMesh POSIX driver consists of two kernel modules, a lightweight gateway driver that communicates to the VFS driver and a meaty file systems I/O driver that communicates with the Gateway driver and Frontend container. By separating the kernel functions into two modules NeuralMesh can perform upgrades to the driver in a non-disruptive way. The gateway driver allows WEKA to keep the file system mounted and maintains a heartbeat with the application server while the underlying driver module can be upgraded without having to unmount the application client.

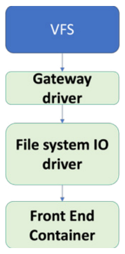

FIG. 2 NeuralMesh POSIX Driver

Frontend Container

The Frontend container is the entry point for application clients to access the distributed file system, providing the default POSIX protocol. It manages POSIX client access and coordinates I/O operations with the other containers (Compute, Drive, Management), and routes them to the appropriate container process within the cluster. Leveraging the NeuralMesh POSIX driver, the application sees what it thinks is a local file system such as ext4 or xfs. NeuralMesh has complete integration with the Linux Page Cache, so all the local Linux mechanisms will work seamlessly. VFS directs the request to the Frontend container (via the POSIX driver) which takes over responsibility for I/O from there on. The location of the metadata is entirely computationally derived via a hash function by the Frontend container vs storing in-memory. This is an important design consideration as it eliminates the need for memory-based look-up tables, reducing I/O overhead and boosting performance. As the overall system grows and scales it does not hit available memory limits that typically impact other systems.

Leveraging DPDK, a high-speed packet processing capability, the Frontend container communicates directly to the server NIC on its own "super-highway" network. In so doing, NeuralMesh avoids having to compete with other processes for resources from the network kernel. This kernel bypass eliminates any latency that can build up in the application server as user space resources fight for limited Kernel resources. All other storage systems are dependent on the system kernel to execute network traffic, and a heavily loaded client can cause significant tail latency.

The network protocol runs on either Infiniband with native verbs, or on Ethernet through DPDK and provides a complete zero-copy datagram protocol. Other protocols such as RoCE can provide similar capability, however this requires the network to be re-configured to support the settings RoCE requires (for example PFC). By contrast NeuralMesh can run on any standard Ethernet network – on premises or in the public cloud – without the need for special network settings, allowing it to be deployed in any standard environment while delivering consistent performance. For example, when NeuralMesh runs inside any of the public cloud environments, it delivers the same zero copy attributes as if running across RoCE, but without the network limitations.

Compute Container

The magic that gives NeuralMesh its incredible data distribution, scaling capability and rapid response to I/O requests happens to a great degree in the Compute container. The Frontend container hands off I/O and metadata requests to the Compute container, which is responsible for file system processes such as data distribution, data protection, clustering, tiering, rebuilds and metadata services. Additionally, the Compute container is responsible for implementing strategies for optimal read, write and metadata operations to ensure the system maintains highest performance and lowest possible latency. A key responsibility of the Compute container is managing load balancing across the cluster, redirecting I/O operations when it detects increasing latencies due to heavily utilized or failed hardware. This will be covered in more detail later.

The file system functions and data protection functions are vertically integrated and NeuralMesh uses the concept of a virtual metadata server to manage this relationship. The Compute container creates and manages several virtual metadata servers in a failure domain. Each virtual metadata server is a shard of the file system owning a portion of all the objects of every file system and snapshot. All the data and metadata of the storage system are equally divided among these logical entities, so any one virtual metadata server owns only a portion of the total cluster capacity. The virtual metadata server is one of the key building blocks of the distributed I/O processing and is responsible for uniform distribution of the data across the file system, data protection, snapshots, object storage tiering, and more. Each virtual metadata server contains a data protection agent that is responsible for calculating the data protection function of the data it is responsible for. Every metadata entity is associated with only one virtual metadata server, and it utilizes journaling to keep track of all the metadata operations executed on blocks of data. The journal enables WEKA to deliver very low latency failure resilience.

Additionally, the virtual metadata servers are not static entities limited to a single failure domain, they can travel and failover to any other Compute container for resiliency purposes. The virtual metadata servers are protected to a level 5 or level 9 across different failure domains in the cluster depending on the parity level (2 or 4) of the data protection stripe. A “council”, whose main purpose is to ensure redundancy, assigns each virtual metadata server five (or nine) compute cores on five (or nine) different failure domains. Leveraging a RAFT protocol, one of the council members is elected as a leader to run the virtual metadata server.

This capability is foundational to WEKA’s resiliency and allows NeuralMesh to rapidly respond to issues impacting performance, so it can take immediate remedial action if a virtual metadata server is acting up. This will be covered in additional detail in the Load Balancing section.

Drive Container

The Drive container is responsible for the "last mile" mapping between logical space and physical media as it manages the physical storage resources, specifically handling the NVMe drives and I/O operations to these drives. The Drive container communicates directly to the NVMe devices leveraging another standard called SPDK (storage performance development kit) a storage centric zero-copy module that runs in user space, again bypassing the Kernel for optimal performance and minimized latency. The communication to the drives provides similar functionality as NVMe over Fabrics (NVMeoF) however it does not require the network to be re-configured to support RoCE, allowing WEKA to run on standard Ethernet networks such as the public cloud while still delivering the highest performance possible. Typically, there is a one-to-one mapping of the Drive container to physical SSD, although for very dense systems the number of SSDs can be increased.

The Drive container also plays an important role in reporting the state of the physical drive by regularly polling the SMART data of the drive. It also monitors the PCIe status of the drive to ensure the controller is functioning.

Management Container

The Management container is a central tool for managing and monitoring the NeuralMesh storage system. It facilitates the management of the individual containers that make up the system including resource allocation and expansion. The Management container can be accessed via both a Graphical User Interface (GUI) or Command Line Interface (CLI) and is used to facilitate management of quotas, snapshots, protocol etc. It is also responsible for monitoring the health of the cluster, logging events, statistics, usage and analytics.

Telemetry Container

The Telemetry Container is a recent addition to NeuralMesh and highlights the power of the container approach. This container serves the function of logging events in the system and is an optional function that the user can set (for example if audit logging is required).

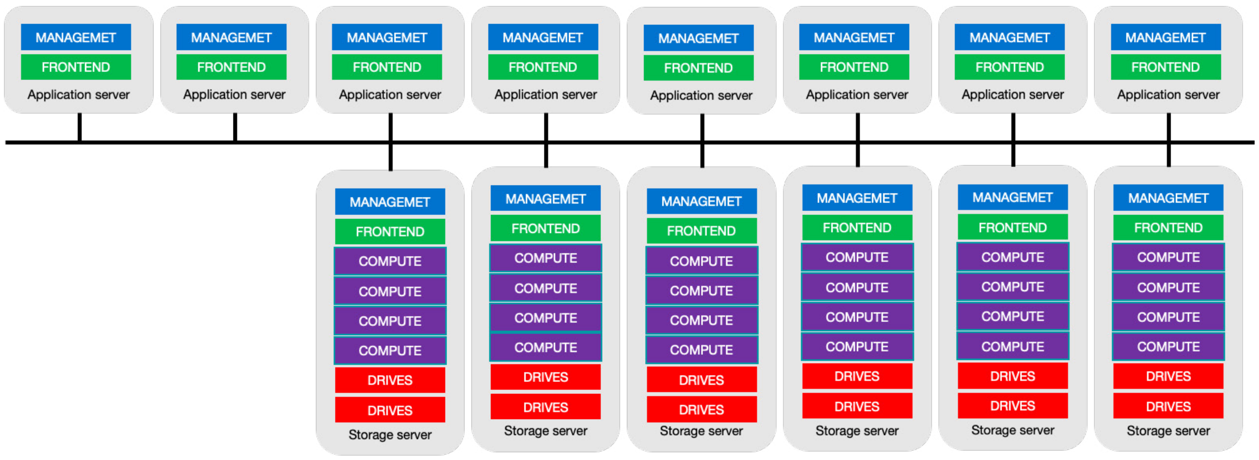

FIG. 3 Typical representation of a NeuralMesh Cluster

Inter-Container Communication

At its foundation, NeuralMesh is a series of containerized processes that communicate with each other to deliver the storage system capabilities. All communication happens over the network even if the containers are on the same physical server; everything is networked and distributed, and no inter-process communication happens locally. This allows the containers complete flexibility to reside anywhere on the cluster and they can be moved within the cluster as hardware resources fail or get overloaded. All communication is done using Remote Procedure Call (RPC), a protocol that allows functions to execute on a remote computer as if it were a local call.

NeuralMesh Architecture

NeuralMesh reinvents the data architecture stack with a fully composable, container-native design that delivers extreme performance, intelligent scalability, and built-in resiliency for AI workloads—across on-premises, cloud, and hybrid environments.

Its capabilities are organized into five key components—Core, Accelerate, Deploy, Enterprise Services, and Observe, and—that work in unison to power the most demanding AI pipelines at scale as shown in Figure 4 below.

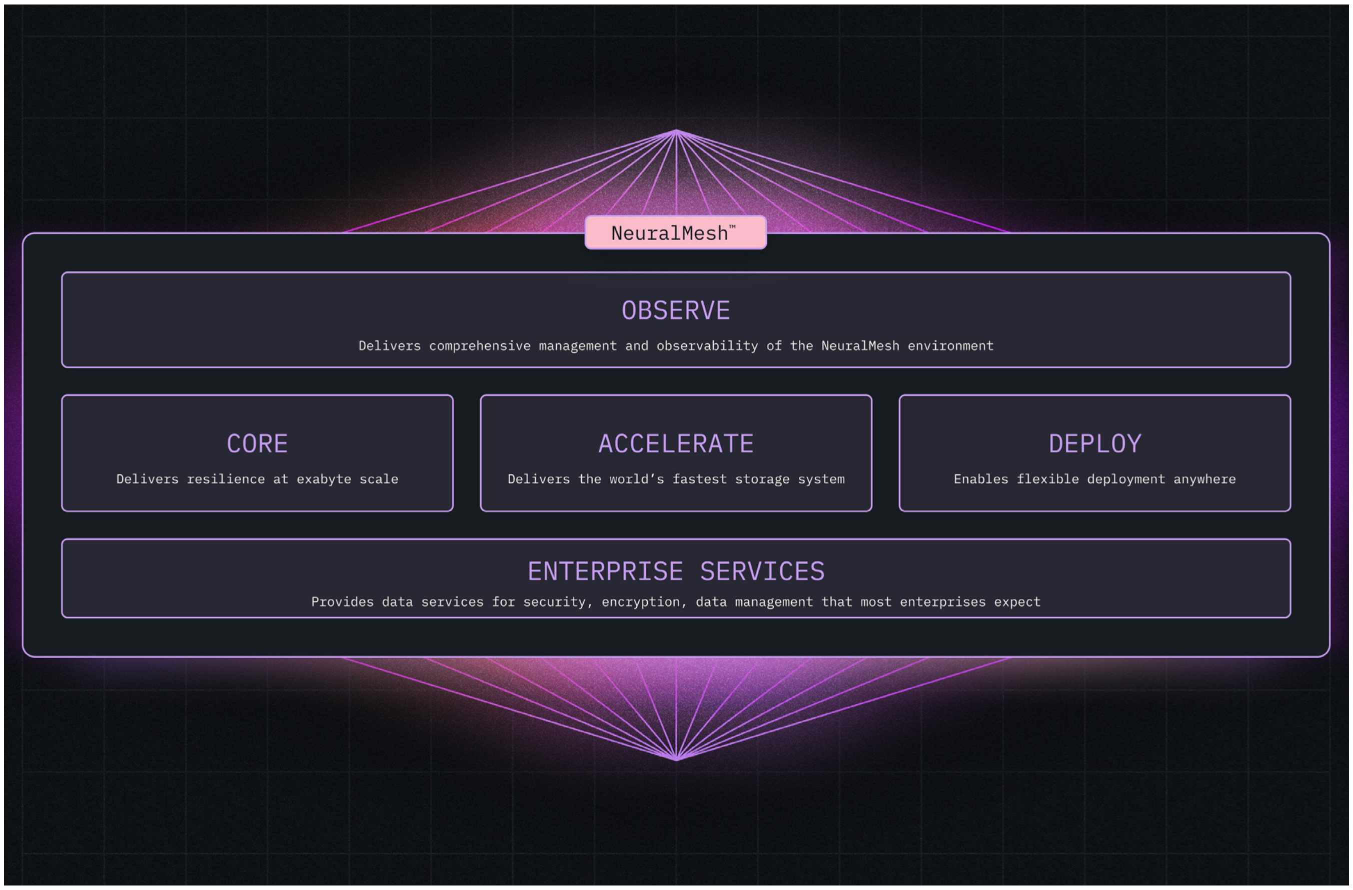
NeuralMesh™
OBSERVE
Delivers comprehensive management and observability of the NeuralMesh environment
CORE
Delivers resilience at exabyte scale
ACCELERATE
Delivers the world's fastest storage system
DEPLOY
Enables flexible deployment anywhere
ENTERPRISE SERVICES
Provides data services for security, encryption, data management that most enterprises expect

FIG. 4 The NeuralMesh Architectural Components

Core – Distributed, Resilient, and Self-Optimizing

The Core component is the foundation of NeuralMesh, designed to become stronger and more resilient as it scales. It intelligently distributes data and metadata across the system, automatically balancing I/O to prevent hotspots and idle resources. With built-in auto-healing, auto-scaling, and rapid rebuild capabilities, Core ensures high availability and durability— at petabyte scale and beyond - making it ideal for mission-critical AI infrastructure.

Data Protection

Data protection is a critical function of any storage system, and the challenges are amplified at scale. Without an appropriate internal data protection schema, file systems need to be limited in size to accommodate the effects of disk or server rebuild time windows and minimize the risk of data exposure. Legacy data protection schemes such as RAID and Reed Solomon based erasure coding (EC) force limitations on the file system to overcome the challenge of a long rebuild that increases the risk of catastrophic failure. The design philosophy of WEKA was to create a software solution that has the performance of an all-flash array with the scalability and economics of the cloud. This required devising a new coding schema that would solve the problems inherent in RAID, EC, and replication because any one of these schemes would violate this design philosophy. Traditional RAID cannot scale, EC has poor small file I/O performance, and replication is both expensive at scale and has poor throughput. The resulting solution from WEKA is a patented distributed data protection scheme that is faster than replication, has minimal capacity overhead, and can scale to billions of files.

NeuralMesh Data Stripes

The NeuralMesh Compute container manages all data protection across the cluster. As noted earlier, the file system and data protection functions are vertically integrated and managed at the virtual metadata server level. WEKA implements a distributed data protection system that uses a D+P model, where D represents the number of Data failure domains and P the number of parity failure domains. NeuralMesh supports data failure domain ranges from 5 to 16, while parity failure domains can be +2 or +4

D+2 – Standard fault tolerance recommended for most environments

D+4 – Used in large-scale clusters and converged environments for maximum redundancy

The minimum supported cluster size is 8 which allows for 1 full virtual spare failure domain for a rebuild from a 5+2 configuration. In practicality most configurations are much larger, and a 16+2 configuration will deliver 83% utilization of the NVMe SSD capacity with 1 virtual hot spare of rebuild capacity. Additionally, the larger stripe improves performance as all data chunks are written and read back in parallel.

Data Distribution

NeuralMesh protects data at the failure domain level, a failure domain is typically a single server, however it can be defined at the rack level or even an Availability Zone or Region if running in the cloud. Data is broken into 4KB chunks to align with the NVMe SSD and spread across the number of failure domains in the data stripe. For example, a 16+2 stripe will be distributed across 18 unique failure domains.

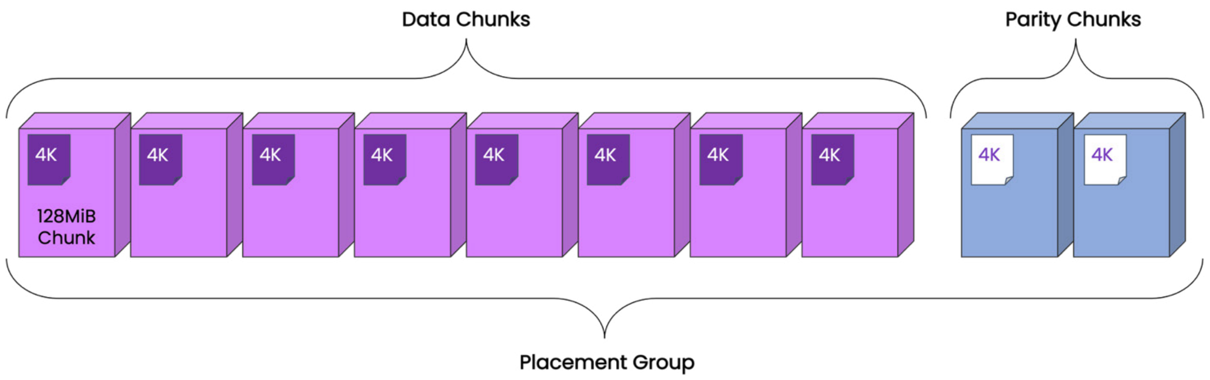

FIG. 5 Data Placement Across Failure Domains

With NeuralMesh, there is no concept of data or metadata locality, as all data and metadata are distributed evenly across the storage servers, which improves scalability, aggregate performance, and resiliency. With the advent of high-speed networks, data locality contributes to performance and reliability issues by creating data hot spots and system scalability issues. By directly managing data placement on the SSD layer, WEKA can distribute the data across the storage cluster for optimal placement.

As clusters grow in node count, the probability of a hardware or drive failure goes up proportionally. However, because of the massively distributed data writes, NeuralMesh becomes more resilient as the number of failure domains grows. Utilizing a highly randomized data placement methodology managed by the virtual metadata servers, the larger cluster becomes more resilient than a smaller cluster because the probability of any two failure domains sharing a chunk of the same data stripe goes down exponentially.

Example: for a stripe size of 18 (16+2) and a cluster size of 20 the number of possible stripe combinations is 190, adding one more server to bring the cluster size to 21 increases the possible number of combinations to 1330. As the cluster size grows to 25, the number of possible stripe combinations is now 480,700. The addition of just 5 more servers has had a dramatic reduction in the probability of two servers sharing the same stripe chunk. So while the probability of hardware failure goes up as more servers are added to the cluster, the probability of data loss goes down exponentially. In this way, NeuralMesh provides a more robust solution for large scale environments than traditional EC and RAID.

NOTE: *The number of possible stripe combinations is based on the following formula where C is the number of servers in a cluster, and S is the stripe size: C!/(S!*(C-S)!).*

Virtual Spare

A virtual (hot) spare is reserved capacity across every failure domain so that if a failure domain goes down, the system will always have sufficient reserved capacity for a complete rebuild of the failed data. All failure domains participate in storing the data, and the virtual spare capacity is evenly spread within all failure domains. The virtual spare capacity is defined during the cluster formation and can be reconfigured at any time. The default number of virtual spares is one.

Fast Data-Only Rebuilds

Failures are inevitable, and the bigger the system, the higher the probability that hardware failures will occur. NeuralMesh was designed to ensure that the system is restored to a more protected level as quickly as possible. WEKA uses several innovative strategies to quickly restore the system to a fully protected state and be ready to handle a subsequent failure.

First, NeuralMesh protects data at the file level, so it only needs to rebuild the specific parts of the file data related to the server hardware failure. Additionally, any data that has already been tiered off to object store is never impacted by a server failure because it is protected on the object store. The cached data (data that was tiered to object but remains on the flash tier until invalidated) does not need to be rebuilt either, limiting the rebuild priority to data that only resides on the flash tier.

NeuralMesh stripes are comprised of 4k blocks and a stripe is distributed across all available failure domains. No two blocks belonging to the same stripe will be written to the same failure domain. Therefore, losing a failure domain results in only losing a single block from a stripe, and the system will continue operating with no performance degradation. All the remaining failure domains in the cluster will participate in rebuilding any missing blocks in the stripe. NeuralMesh will rebuild data using a parity calculation and write that data across all remaining healthy failure domains. The larger the cluster size, the faster the rebuild and the more reliable the system becomes, because stripes are more randomized and there are more compute resources available to participate in the rebuild process. In the event of two failures, the system prioritizes data rebuilds starting with data stripes that are common to both failed units. NeuralMesh rebuilds and redistributes these data stripes first so the system can be returned to the next level higher of protection as fast as possible. This prioritized rebuild process continues until the system is returned to full redundancy. By contrast, RAID and erasure coded systems suffer a performance hit as only a small subset of the servers that are responsible for the data participate in the recovery. With WEKA, the recovery rate is user-configurable and the amount of network traffic dedicated to rebuilding can be changed at any time, so administrators have complete control to determine the best tradeoff between continued application performance and time-to-recovery. This ensures that applications are not impacted by long data rebuild processes.

Drive Silent Corruption Protection

Drive failure is relatively straightforward to recover from; however, drive corruption can be the silent and deadly enemy of storage systems. The problem with other RAID solutions is that it rebuilds a missing block by recomputing from its peers and parities. However, if one of the drives has corruption (for example a bit flip), it will report bogus data, and the reconstructed data will now be corrupt. WEKA's data protection can handle drive level data corruption – up to 4 in a single drive – and still repair the data.

Power-Fail and End-to-End Data Protection

Using a checksum process to ensure data consistency, NeuralMesh provides end-to-end data protection for both reads and writes. Checksums are created on a write and validated on a read. NeuralMesh always stores data and checksum information separately from each other on different failure domains for improved protection.

NeuralMesh provides additional data integrity capabilities by protecting against data loss due to power failures. When a write is acknowledged back to the client, it is safely protected from server failures or data-center-wide power failure through the virtual metadata journaling process. WEKA's innovative data layout and algorithms enable it to recover from a data-center-wide power failure in minutes because there is no need to do a complete file system consistency check. Instead, NeuralMesh just verifies the last journal entry for every virtual metadata server in the system. For other storage systems, the recovery time is proportional to the size of the recovered file system. In large scale deployments, recovery can take days or weeks.

Multitenancy

WEKA's Multitenancy capabilities provide secure and scalable hierarchical access controls through organizations, ensuring both physical and logical isolation for tenants. Leveraging its container architecture, NeuralMesh offers a robust, high-performance solution for multitenancy by creating secure spaces that ensure efficient resource allocation while keeping tenants isolated. This is particularly beneficial for GPU-as-a-Service providers who require precise resource management, as well as enterprise AI clients and healthcare organizations that handle sensitive data and must adhere to strict security and compliance standards. By delivering job-level security and ensuring data integrity and privacy, WEKA helps these organizations maintain reliable operations without compromise.

Legacy systems were never designed to be operated in a multitenant environment and largely rely on network segregation and shared architectures that may expose vulnerabilities. By contrast, NeuralMesh implements its multitenant environment leveraging its container architecture, which inherently creates tenant isolation. NeuralMesh provides full isolation across storage, compute, and network resources. This approach significantly enhances security, mitigating risks associated with older systems and ensuring a more robust, isolated environment. The clusters can be deployed in minutes, operating independently to avoid the pitfalls of noisy neighbor issues. They integrate seamlessly with the NCP (Network Control Protocol) common network reference architecture, thereby streamlining operations and boosting overall efficiency.

Security is further strengthened through comprehensive encryption at the cluster level. When enabled, all data is automatically secured with multiple layers of encryption applied to the file system, file data, and even sensitive metadata such as file names and timestamps—only the file size remains unencrypted. Clients retain control over encryption management, while NeuralMesh safeguards wrapped key material, ensuring a level of protection that surpasses traditional models.

All multitenant environments are thinly provisioned to ensure the best utilization of the underlying infrastructure. Tenant capacity is set at the file system level and can be dynamically expanded without any need to unmount the file system.

Secure Tenant Isolation

WEKA's multitenancy architecture ensures both physical and logical isolation for tenants, addressing challenges like the "noisy neighbor" effect and performance interference. Each tenant is allocated dedicated resources, such as drives, memory, and CPU cores, delivering secure and independent operations. Each tenant has unique encryption keys and authentication credentials, ensuring that data is secure and isolated. This approach not only enhances security but also delivers predictable performance for demanding workloads, even as tenant numbers scale. By leveraging WEKA's composable architecture, enterprises and cloud providers can dynamically adjust resource allocation in real-time, maintaining efficiency without compromising tenant isolation or performance.

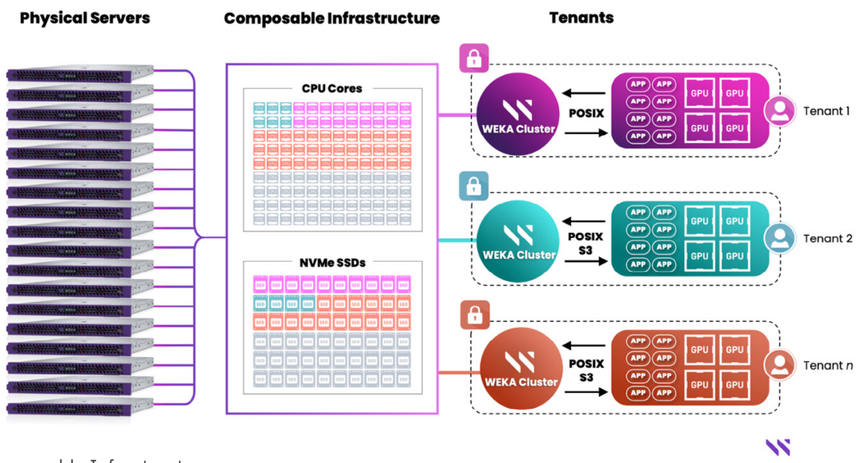

FIG. 6 The NeuralMesh Composable Infrastructure

The NeuralMesh Operator

WEKA leverages the capabilities of Kubernetes to deploy and manage its multi-tenant environment. The NeuralMesh Operator simplifies the deployment and management of a cluster using Kubernetes. By automating routine storage operations and enhancing cluster resilience through custom resources, the Operator not only accelerates AI, ML, and HPC workloads but also supports large datasets with a high-performance data layer. As business needs evolve, the platform allows for seamless expansion of storage and tenant resources, supported by real-time monitoring tools that provide critical insights into usage and performance for proactive management.

The Operator contains all the information related to a cluster configuration and monitors the current state of the Kubernetes cluster. When it detects changes to the Customer Resource Definitions (CRD), the Operator applies the changes to the running cluster.

Tenant Management

The Operator simplifies the Day 2 operations of managing a shared tenant storage cluster providing a simplified way to scale, upgrade and provision the environment.

Hardware maintenance

- Hardware component replacement
- Node management
- Decommissioning
- Reallocating Container to new hardware

Cluster scaling

- Expand a cluster
- Expand cores
- Shrink a cluster
- Resource allocation and optimization

Upgrades

- Update software
- Update drivers

WEKA's solution is designed for private cloud and specialty cloud providers – such as GPU-as-a-service providers - to enable them to easily configure and manage tenants. The solution maximizes hardware utilization, reduces infrastructure complexity, streamlines configuration, and reduces administrative overhead. With advanced data protection that ensures high availability and compliance, NeuralMesh creates a secure and efficient environment for mission-critical workloads, meeting the demands of modern enterprises in an ever-evolving technological landscape.

Secure Multitenancy with VLAN Tagging

The NeuralMesh storage system leverages VLAN tagging to ensure secure, scalable, and isolated network communication in multi-tenant environments. This feature allows for precise network traffic segregation by associating specific containers with VLANs, with each container supporting one VLAN per NIC. This granular control enhances security and optimizes network configurations while remaining compatible with existing workflows.

NeuralMesh supports IEEE 802.1Q VLAN encapsulation, enabling isolation and segregation of network traffic while granting connectivity between clients and backend servers. A WEKA tenant cluster operates in a single VLAN ID, where all the containers within that cluster must use the same VLAN ID. All clients connecting to the tenant cluster must use that cluster's assigned VLAN ID as well. When used in conjunction with the NeuralMesh Operator the VLAN ID consistency between the tenant and its clients is automatically maintained.

In a multi-cluster environment, each tenant cluster can operate in a different VLAN ID. For example, Tenant Cluster A can have VLAN ID 100, while Tenant Cluster B has VLAN ID 200, providing full network isolation between clusters.

By isolating traffic at the network layer, WEKA ensures consistent performance and secure communication, even in shared environments. For advanced setups, VLAN tagging accommodates extended configurations, including gateways, IPs, and netmasks, providing flexibility to address unique network requirements.

Quality of Service

Traditional IT environments struggle to balance I/O between performance hungry applications with other applications on a shared system, leading to the 'noisy neighbor" problem. It has too frequently resulted in isolating applications on dedicating infrastructure leading to management issues. WEKA provides the ability to prioritize network bandwidth to application clients, setting a preferred and maximum throughput level. The client will attempt to limit as close to the value of the preferred performance as possible but allow bursting up to the maximum amount if resources are available. This enables per-application performance management when accessing a NeuralMesh storage system. When combined with quotas and organizational controls, this allows fine-grained resource management within the overall system.

Coherent Adaptive Caching

Applications, particularly those with small files and lots of metadata calls, benefit greatly from local caches. The data is available with very low latency, and it reduces the load on the shared network as well as on the back-end storage itself. NeuralMesh provides a unique advanced caching capability, called adaptive caching, that allows users to fully leverage the performance advantages of Linux data caching (page cache) and metadata caching (dentry cache) while ensuring full coherency across the shared storage cluster. NFS v3 does not support coherency, so utilizing Linux caching can lead to data inconsistency for read cache and potential data corruption for write cache. NeuralMesh supports leveraging Linux page cache — typically reserved for direct attached storage (DAS) or file services over block storage — on a shared networked file system, while maintaining full data consistency. The intelligent adaptive caching feature will proactively inform any client that was an exclusive user of a file (and hence running in local cache mode), that another client now has access to the data set. Once this flag is set, the client can continue running in local cache mode until the file is modified by another client. NeuralMesh will then invalidate the local cache ensuring that both clients are only accessing the most recent iteration of the data. This ensures the highest performance from local cache when appropriate and always ensures full coherency on data. This functionality does not require specific mount options to leverage local page cache as NeuralMesh dynamically manages caching, making the provisioning of the WEKA environment very simple to manage with no danger of an administrative error causing data corruption.

NeuralMesh provides the same capability for metadata caching, also known as Linux dentry cache. A client can leverage local metadata cache for a directory, reducing latency significantly. However, once another client has access to the same directory, NeuralMesh will ensure that any directory changes from one client will invalidate the cached metadata for all other clients accessing that directory. The caching capability also includes extended attributes and access control lists (ACLs).

While some shared file storage vendors allow local caching, it is typically disabled by default and requires an administrator to change the mount option. That is because write coherency typically depends on some form of battery backup protection on the client to ensure data consistency on a committed write. WEKA's caching implementation will work out-of-the-box without any administrative intervention as NeuralMesh does not depend on battery protection to protect acknowledged writes. This feature is ideal for use cases such as file "Untar", which will run significantly faster as a local process vs. across a shared file system.

Accelerate – Consistent High-Performance Data Access

The NeuralMesh Accelerate components establish ultra-low latency, direct paths between compute and data by distributing both data and metadata across the entire system. This eliminates performance bottlenecks, ensures linear scalability, and maximizes GPU utilization—making it a critical enabler of high-throughput training and low-latency inference in AI environments.

Highly Efficient Data Path

NeuralMesh is a highly distributed system that delivers exceptional I/O performance at scale. As noted earlier, the foundation of the design is a series of containers that work both independently and in unison to deliver the most efficient and scalable performance available. To understand how WEKA achieves its linear scaling and performance, this section will go under the hood to explain how NeuralMesh builds parallelism and scalability into the solution while keeping the entire system balanced.

Accelerated Writes

WEKA's parallelism begins with the Frontend container which intercepts all POSIX system calls and directs them to the Compute container. NeuralMesh has implemented single hop write for large file writes, instead of executing data blocks in two network hops - from Frontend container to Compute container to Drive container - NeuralMesh will pass the data blocks directly in a single network hop from the Frontend Container to the Drive container. For example, with a data protection policy of 16+2, if the Frontend container receives a request to write a 1MB file. It first breaks the files into 16x64K chunks plus 2 parity chunks, and sends a request to the Compute container for a range of addresses to write the data and parity. The Compute container calculates the block location for the Write requests, including the associated data protection, and returns the locations to the Frontend container. The data is then sent in 18 parallel streams of 64K chunks directly to the Drive container, which commits the data to physical media. Single hop writes occur when certain conditions are met that allow the write to be executed in a safe manner. If the conditions are not met then the system will automatically revert to a two hop write from Frontend to Compute to Drive containers. This capability does not require any API or user adjustments; it just works intelligently based on the write conditions.

To modify existing data NeuralMesh also implements strategies to minimize the number of operations which result in a performance impact. Other storage systems use a multi-step "read-modify-write" operation on existing files, however NeuralMesh always performs a write to a new location negating the need for this additional step. For a write operation on existing data an application sends a write request to the Frontend via the NeuralMesh POSIX driver. The Frontend coordinates data and metadata operations within the system and calculates which metadata server owns the operation by checking the file or block range within the incoming file. On receipt of incoming data, the virtual metadata server calculates new data placement blocks from the data protection process within the Compute container. These blocks will be written out to new locations distributed across different failure domains based on the data protection setup plus parity. The Drive process is responsible for the "last mile" of committing the data to NVMe. When data is successfully committed, the Drive process sends an acknowledgement to the Compute virtual metadata server, and the journal is updated with the new location of the data blocks. The Compute container acknowledges the write to the Frontend container, which in turn acknowledges to the NeuralMesh POSIX driver and finally back to the application.

Accelerated Reads

Read operations are performed in a similar manner; on receipt of an application request for data the Frontend container calculates what virtual metadata servers own the data and sends a request to the Compute containers for the file ranges. The Compute containers communicate with the Drive containers which fetch the data blocks from the associated NVMe drives and return the data to the application. Because the data is highly distributed, multiple metadata servers are working in unison to return the file data to the application in a highly parallel manner.

Load Balancing to Maintain Peak Performance

A key component of NeuralMesh is the ability to maintain a highly balanced system. WEKA implements a load balancing strategy for all reads, writes and metadata operations with a primary focus on minimizing latency impact of the operation.

Load Balancing Writes

The Compute container plays a critical role in maintaining the linear performance scaling of the NeuralMesh storage system. The file system and data protection are vertically integrated, unlike other systems where the two maintain independent journals, WEKA maintains a single journal to manage all of the writes. As a result, WEKA does not use the normal "read-modify-write" strategy for writes, which introduces a lot of latency into each operation. Every write is independent of the previous write to ensure optimal load balancing. A write is striped across the number of failure domains defined in the data protection stripe, and the write location is chosen based on the latency profile of the servers within the cluster. Using the server latency histograms, it will look for less busy servers to distribute the write. Data and parity are written directly from the Frontend to the NVMe drives in parallel for maximum performance.

Load Balancing Reads

The process of load balancing reads is significantly more complex than writes because once written, a read is dependent on the performance of the underlying hardware infrastructure. In legacy systems, a read is a single operation dependent on the drive performance. If one of the drives is misbehaving, then the read will suffer a significant latency hit. However, WEKA maintains a histogram of the latency across all the drives and if it detects a latency spike on a single misbehaving drive, it will treat it as a failed drive and resort to building the stripe from the remaining drives including the parity drives. In this way WEKA maintains very low latency even in the presence of a hardware failure and the system will stay performant and well balanced.

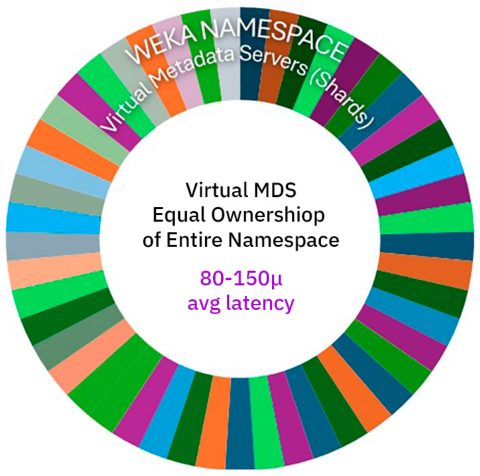

FIG. 7 Data Distribution across Virtual Metadata Servers

Load Balancing Metadata

One of the key reasons behind WEKA's incredible speed is this unique distributed metadata management system described earlier. Unlike traditional systems that rely on a single metadata server or centralized databases, WEKA takes a different approach that scales out horizontally without bottlenecks. In traditional systems, metadata often becomes a performance choke point, limiting scalability and creating "hotspots" where too much traffic hits a single metadata server.

NeuralMesh is the only system that has a dynamic method of load balancing metadata to ensure consistent low latency. In building the metadata load balancing strategy, WEKA leveraged some of the theoretical concepts of Object Storage and expanded them significantly.

NeuralMesh uses a pie concept to manage metadata operations. Every 1MB chunk of a file has a unique position in a different slice of the pie, so within a single file different metadata operations will hit different virtual metadata servers resulting in a highly distributed workload across the entire system. Each metadata server only handles a small portion of the total namespace, so the workload is automatically distributed. As clients access different files or parts of files they interact with different metadata servers. Even if multiple clients access the same file but in different block ranges, the load is split across different virtual metadata servers. This means no single metadata server becomes a bottleneck. Additionally, the virtual metadata servers do not interact with each other, therefore, there is no wasted east-west communication that degrades performance. The load-balancing mechanism further ensures that no "hot spots" form, preventing performance degradation and maintaining the efficiency of the entire platform.

As mentioned earlier, the virtual metadata servers are protected by assigning each one 5 (or 9) possible cores that can manage that unit. The Compute container system chooses where to execute the virtual metadata servers based on how busy the compute cores are. If the system detects that the latency from one virtual metadata server is not good enough due to a misbehaving core, it will move the virtual metadata server to run on another one of the four or eight remaining CPU cores in that council. Likewise, if a failure domain goes down, the virtual metadata server will move to a CPU core on another failure domain as well as assign a new fifth server core to that virtual metadata server. Failover occurs in split second time with no perceptible impact on performance.

Because of the massive distribution of the metadata over so many virtual metadata servers, NeuralMesh does not suffer the latency hits that other systems experience when the metadata load is high, as is common in AI workloads.

WEKA implements a similar principle for directories, so if a single directory ends up with thousands or millions of files, NeuralMesh splits the directory into more directories to maintain fast response to requests.

Augmented Memory Grid to Extend GPU Memory Footprint

Augmented Memory Grid is a distributed, persistent storage layer designed to extend the memory capacity of GPU-based inference workloads. By integrating with common inference optimization frameworks, it enables the Key-Value (KV) cache to be efficiently offloaded from GPU high bandwidth memory (HBM) to WEKA's NVMe-backed storage pool. Leveraging GPUDirect Storage (GDS), Augmented Memory Grid bypasses the CPU to enable direct memory access and zero-copy data transfers between WEKA and GPU HBM. This allows NeuralMesh to function as a high-speed, low-latency token warehouse, delivering near-DRAM performance while significantly expanding available memory capacity to petabyte levels. The result is a scalable, high-performance solution for large language model inference that eliminates traditional memory bottlenecks.

High Performance Protocols

WEKA supports full multi-protocol and data-sharing capability across a variety of protocols allowing diverse application types and users to share a single pool of data. Data can be written in one protocol and read via another protocol, allowing users to share a single data set across a diverse group of application types. NeuralMesh has full cross protocol locking so users can simultaneously access the same data without any corruption and manages all security and access. The following lists the protocols supported:

- Full POSIX for local file system support
- NVIDIA GPUDirect Storage (GDS) for GPU acceleration
- NFS
- SMB for Windows
- S3 for Object access

POSIX

The NeuralMesh client is a standard, POSIX-compliant file system driver installed on application servers that enables file access to the storage system. Like any other file system driver, the NeuralMesh POSIX client intercepts and executes all file system operations. This enables WEKA to provide applications with local file system semantics and performance, while providing a centrally managed, sharable, and resilient storage platform. WEKA provides advanced capability such as byte-range locks and is tightly integrated with the

Linux operating system page cache, covered later in the caching section.

The WEKA POSIX client provides the highest performance for IOPS, bandwidth, and metadata at the lowest latency.

NVIDIA GDS

GPUDirect Storage is a protocol developed by NVIDIA to improve bandwidth and reduce latency between the server NIC and GPU memory, leveraging RDMA. WEKA has full support for GDS and has been validated by NVIDIA including a reference architecture.

NFS

The NFS protocol allows remote systems to access NeuralMesh from a Linux client without the POSIX client. It provides a simple way to deploy and share data from the storage cluster. WEKA currently supports NFS v3, and NFS v4.1.

SMB

The SMB protocol allows remote systems to connect to shared file services from a Windows or macOS client. WEKA has support for SMB v2.x and v3.x. and supports clients configured with Multichannel to deliver high performance.

The protocol provides a highly performant, scalable, resilient and distributed implementation of SMB, supporting a broad range of SMB capabilities including:

- User authentication via Active Directory (Native and mixed mode)
- High availability and client failover
- POSIX mapping (uid, gid, rid)
- UNIX extension
- SHA 256 signing
- Expanded identifier space
- Dynamic crediting
- Durable opens for handling disconnects
- Symbolic link support
- Trusted domains
- Encryption
- Guest access
- Hidden shares
- SMB ACLs
- Conversion from Windows to POSIX ACLs
- SMB security related share options
- SMB over RDMA

S3

Many web-based applications now support the S3 protocol, however S3 was designed for scalability at the expense of performance. Applications, such as real-time analytics on IoT data can benefit from high performance S3 access. WEKA has implemented S3 front-end support for its performance file system to accelerate S3 storage I/O. NeuralMesh delivers huge performance gains for small file I/O accessed via S3. The NeuralMesh S3 API supports the following calls:

Buckets (HEAD/GET/PUT/DEL)

Bucket Lifecycle (GET/PUT/DEL)

Bucket Policy (GET/PUT/DEL)

Bucket Tagging (GET/PUT/DEL)

Object (GET/PUT/DEL)

Object Tagging (GET/PUT/DEL)

Object Multiparts (POST Create/Complete, GET/PUT/DEL, GET Parts)

In addition, the S3 implementation supports multiprotocol access, Transport Layer Security (TLS), has full S3 audit logs, and bucket level features such as policies, quotas-per-bucket, and Expiry rules for information lifecycle management.

S3 Fast Data Store

Leveraging the S3 protocol for data ingestion combined with POSIX-native data processing, NeuralMesh now delivers an ultra-high performance S3 data store. Once data is ingested into the storage system, it benefits from all the data optimizations that WEKA has implemented at the data layer. For AI workloads that require S3 semantics, the NeuralMesh S3 Data Store delivers exceptional performance when deployed with DPDK. WEKA's S3 has been optimized to handle millions of small objects and files with the high throughput and low latency that AI applications require. The Zero-copy design improves GET requests by up to 70% compared to legacy deployment - the larger the object size, the bigger the performance gain.

Deploy – Run Anywhere, Scale Everywhere

Modern AI infrastructure is not confined to a single location—it spans on-premises data centers, public clouds, and hybrid environments. NeuralMesh™ ensures that your data layer can go wherever your AI runs. Whether you're building AI Factories on bare metal, deploying across multicloud environments, or scaling inference at the edge, NeuralMesh delivers seamless, high-performance data services with full architectural consistency. This flexibility enables faster time to value, lower TCO, and future-proof adaptability as customer environments evolve.

Deploying NeuralMesh

A WEKA cluster is created by installing NeuralMesh on any standard x86 based server such as AMD EPYC™ or Intel Xeon™ Scalable Processor-based hardware with the appropriate memory, CPU processor, networking, and NVMe solid-state drives. The deployment can be on-premises or in the public cloud.

A minimum configuration of 8 servers is required to create a cluster that can survive a two-server failure (5+2) with a 1 server virtual hot spare. The software then clusters the servers into a single namespace leveraging the standard network. Note, NeuralMesh does not require a dedicated network fabric; it shares the same network as the application servers and runs its own networking protocol across the standard network.

While WEKA is software only, a comprehensive list of pre-qualified platforms are available from all major original equipment suppliers (OEMs). NeuralMesh is also available on the major public clouds including Amazon Web Services (AWS), Google Cloud Platform (GCP), Oracle Cloud infrastructure (OCI) and Microsoft Azure, and delivers the same experience and functionality as the on-premises model.

For users who want a single appliance experience, WEKA also has fully integrated appliance configurations.

Dedicated Storage Server

NeuralMesh can be configured as a dedicated storage server where all the resources of the system are dedicated to storage services while applications run on a separate server infrastructure. This mode is the most popular among customers as it ensures that application disruptions do not impact storage services.

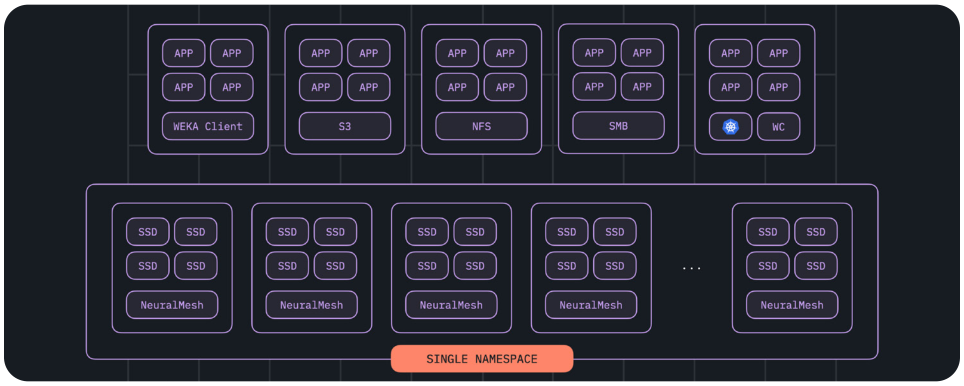

FIG. 8 Dedicated Storage Server with Separate Compute and Storage Infrastructure

NeuralMesh™ Axon™

When deploying NeuralMesh Axon, NeuralMesh is installed on the same servers as the applications. WEKA runs in its own Linux containers separated from the application environment and has a predefined amount of server resources (CPU, RAM, network and NVMe). NeuralMesh Axon is primarily targeted to extremely large scale GPU server deployments to avail of the hundreds of unused CPU cores and NVMe drives in the GPU servers. This offering provides the ability to create integrated application-based solutions with better economics, by eliminating the need for external storage, reducing hardware footprint and power while boosting performance, availability, and capacity. NeuralMesh Axon deployments require a minimum of 25 servers, and the data protection schema is always set to 4 parity failure domains.

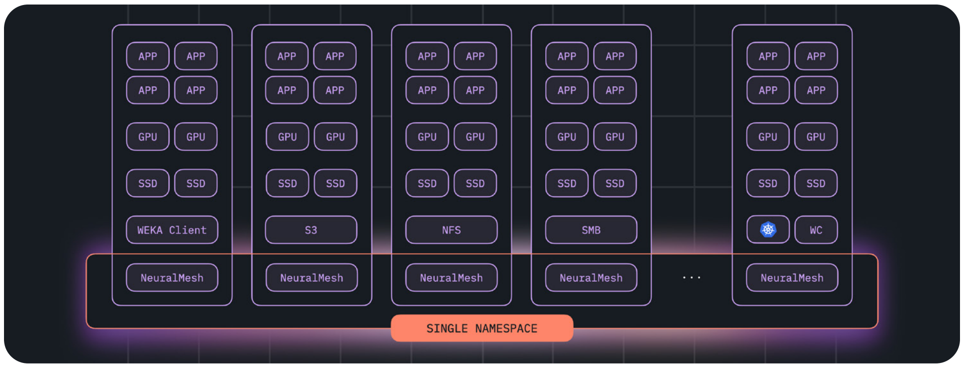

FIG. 9 NeuralMesh Axon deployments with Compute and Storage on the Same Infrastructure

Native Public Cloud

NeuralMesh is available in multiple public clouds including Amazon Web Services (AWS), Google Cloud Platform (GCP), Oracle Cloud infrastructure (OCI) and Microsoft Azure. WEKA provides simple deployment managers for each of the public clouds, and the procedure is comparable to on-prem deployment: You choose a server configuration

(AWS EC2, OCI Compute Shape, GCP Compute Engine, Etc.), make sure that NVMe devices are in or attached to the compute resource, assign networking, and build the cluster. For additional low-cost, high-capacity storage, WEKA automated tiering to S3-compatible object stores. This includes AWS S3, Google Cloud Storage, Oracle object storage and Azure blob. For improved resiliency, WEKA also supports the spanning of AZs in the public cloud. Figure 10 shows a typical deployment of NeuralMesh in the AWS public cloud. Performance in the cloud is coupled to the number of NVMe devices in a WEKA cluster as well as network speed provided by the cloud vendor.

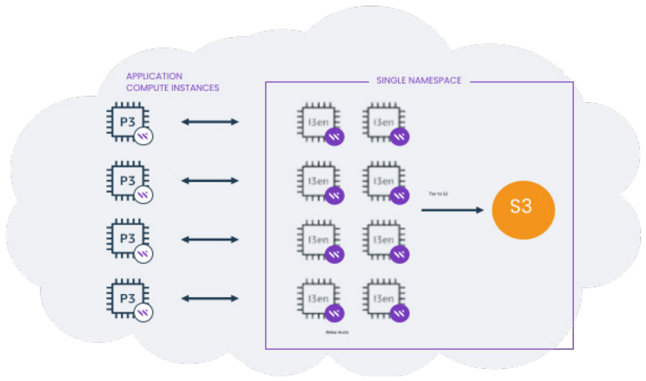

FIG. 10 AWS Public Cloud Deployment

Integrated Flash and Object

NeuralMesh manages all data within the system as part of a single namespace and supports two persistent storage tiers in a single hybrid architecture. The design consists of an NVMe SSD-based flash layer that provides high-performance file services to the applications and runs in dedicated mode or with NeuralMesh™ Axon™. Additionally, it supports an optional S3-compatible object storage layer that manages the long-term data lake (outlined in Figure 11 below). Expanding the namespace to object store is an optional addition to the single namespace and can be configured at any time once the NVMe tier has been configured. NeuralMesh expands the namespace from the NVMe flash layer to the object store, presenting a single namespace that scales to exabytes. The two layers can be physically separate but logically serve as one extended namespace to the applications. The object store can be from any S3-API compliant vendor and can reside on-premises or connect from on-premises to the public cloud. As we will see later, NeuralMesh leverages components of the object store capability to enable cloud bursting, backup to the cloud, DR to another WEKA cluster, or file system cloning.

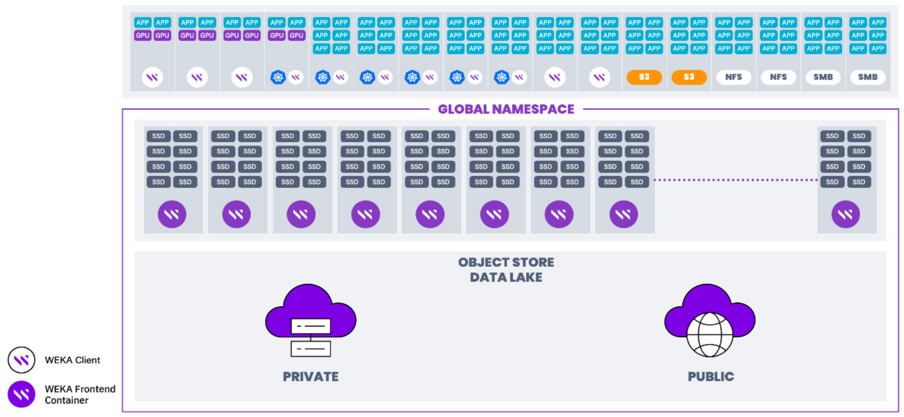

FIG. 11 NeuralMesh Single Namespace

Networking

NeuralMesh does not require its own dedicated network, rather it leverages the shared network to communicate between failure domains across the storage system and to the application servers. WEKA supports the following types of networking technologies:

- Ethernet – 100Gbit and above recommended (10Gbit is also supported)
- InfiniBand (IB) HDR and EDR

Application clients connect to the NeuralMesh storage system via Ethernet or InfiniBand connections. The software supports 10GbE, 25GbE, 40GbE, 50GbE, 100GbE, 200GbE, and 400GbE Ethernet networks, and EDR, HDR, and 400Gb NDR InfiniBand networks. For the best performance outlined in this document, WEKA recommends using at least 100Gbit network links.

Many enterprise environments have a mixed network topology composed of both InfiniBand and Ethernet to support both high performance computing application clients as well as more traditional enterprise application clients. WEKA allows InfiniBand clients and Ethernet clients to access the same cluster in these mixed networking environments, allowing all applications to leverage WEKA's high-performance storage.

For networking, the WEKA system does not use standard kernel-based TCP/IP services which introduce significant latency in the data path; instead, WEKA wrote its own networking stack leveraging DPDK. The use of DPDK delivers operations with high throughput and extremely low latency.

For legacy systems that lack support for SR-IOV (Single Root I/O Virtualization) and DPDK, NeuralMesh defaults to the in-kernel processing and UDP as the transport protocol. This mode of operation is commonly referred to as the 'UDP mode' and is typically used with older NICs. In addition to being compatible with older platforms, the UDP mode does not require dedicated compute resources which can instead be consumed by the applications.

For RDMA-enabled environments, common in GPU accelerated computing, NeuralMesh supports RDMA for InfiniBand and Ethernet to supply high performance without the need to dedicate cores to the Frontend container. RDMA is enabled by default for any interface that is RDMA capable. In mixed network environments NeuralMesh supports RDMA for both Ethernet and InfiniBand simultaneously to the clients. For example, if one client is writing over InfiniBand and another is reading over Ethernet, NeuralMesh will support RDMA to both so performance is not compromised.

WEKA supports VMXNET3 networking from VMware when the Frontend container is deployed inside a guest OS in a VMware ESX hypervisor. This ensures that when a vMotion from one ESX server to another occurs, the WEKA Frontend continues to remain connected to NeuralMesh storage.

Network High Availability (HA)

WEKA supports high availability (HA) networking to ensure continued operation should a network interface card (NIC) or network switch fail. HA performs failover and failback for reliability and load balancing on both interfaces and is operational for both Ethernet and InfiniBand. For HA support, the NeuralMesh storage system must be configured with no single component representing a single point of failure. Multiple switches are required, and servers must have a connection to each switch. HA for clients is achieved through the implementation of two network interfaces on the same client. WEKA also supports the Link Aggregation Control Protocol (LACP) on the compute clients on Ethernet (modes 1 and 4) for a single dual-ported NIC. Additionally, WEKA supports failover of InfiniBand to Ethernet within the storage cluster to maintain high availability in case the InfiniBand network fails.

NeuralMesh can easily saturate the bandwidth of a single network interface card (NIC). For higher throughput, it is possible to leverage multiple NICs in a single server.

Enterprise Services – Secure, Optimized, and Feature-Rich

The components within Enterprise Services deliver advanced security, encryption, data protection, and data management features required in enterprise environments. It includes performance-enhancing features like zero-copy and zero-tuning data access, eliminating the need for traditional tiering strategies and simplifying management across diverse AI workloads.

Single Namespace

WEKA's NeuralMesh differs from other scale-out solutions in that it embraces the concept of many file systems within the single namespace that share the same physical resources. A file system can be created instantly, and once configured is ready to accept data. Each file system has its own "persona" and can be configured to provide its own snapshot policies, tiering to object storage, organizations, role-based access control (RBAC), quotas and much more. A NeuralMesh file system is a logical construct and unlike other solutions, the capacity can be changed instantaneously on the fly. Clients that are mounted can observe the change in size right away without any need to pause I/O. As already mentioned, each file system has a choice to tier to an object store, and if it is a tiered file system, the ratio of hot (NVMe) tier and object (HDD) tier can also be changed instantaneously on the fly.

A single file system can support billions of directories and trillions of files, delivering a scalability model more akin to object stores than NAS systems, and directories scale with no loss in performance. Currently WEKA supports up to 1024 file systems, and up to 24,576 read-only snapshots and 14,336 writable snapshots in a single cluster.

WEKA File System Limits:

- Up to 6.4 trillion files or directories
- Up to 14 Exabytes managed capacity in the single namespace
- Up to 6.4 billion files in a directory
- Up to 4 petabytes for a single file

Thin Provisioning the Namespace

WEKA allows thin provisioning of file systems within the single namespace. When additional servers are added into the cluster, any capacity that is available can be pooled and accessed as a thin provisioned resource. This feature is key in allowing both automatic capacity expansion when servers or drives are added, but also in managing space if servers or drives are removed from the cluster. Available capacity remaining after removal of servers or drives must be sufficient to support the amount of data stored in the flash tier for all the file systems. This feature also enables seamless integration with EC2 auto scaling groups in AWS, and auto scaling capabilities in other hyperscaler clouds such as GCP, OCI and Azure.

Capacity Quotas

WEKA supports multiple levels of Quota management to limit capacity usage.

As noted, there are many ways that NeuralMesh manages capacity utilization across organizations.

- Organizational level quotas allow groups to manage their own file systems and capacity. NeuralMesh supports up to 64 organizations.
- File system level capacity allows different projects or departments to have their own allocated capacity.
- Directory level quotas provide a quota per project directory, useful when there are many projects within a single file system. Quotas can be established within nested directories, supporting up to four levels of nested quotas, and can be over-provisioned under the same directory quota tree.
- Quotas can be set at an advisory level, as hard quotas, or as soft quotas.

Tiering to Object Store

NeuralMesh has a built-in, policy-based automated data management feature, that transparently moves data across storage types according to the data access. WEKA supports moving data from the NVMe flash storage tier to onpremises or cloud-based object storage. This feature is particularly useful in cloud deployments to optimize cost by allowing a significant part of the data to be stored in S3.

Data movement is set at the perfile system level and is an optional extension of the NVMe flash tier. For example, to always ensure the highest performance, users may want to keep certain file systems exclusively on NVMe SSD, while other file systems implement data movement to object storage for the best cost economics. Metadata is always stored on the flash tier, and a read-only or read-write snapshot of the entire file system, including its data structures, can be stored on the object storage tier to protect against a failure on the flash tier. The application clients see all the files in each file system in the location they were written, regardless of their tiering status. As a result, no change in application is needed to leverage cost-optimized object storage-based solutions.

A file resides on flash while it is active or until it is tiered off to object storage based on preset or user-defined policies. As noted earlier, files are broken up and distributed across all the failure domains in the cluster and partial file chunks can be pushed to the object store independently. When a file chunk is tiered to the object store, the original chunk is kept on the flash layer until the physical space is required by new data and hence acts as a cache until overwritten. The file metadata always remains locally on the flash tier, so all files are available to applications in the location they were written to, irrespective of tiering placement, even if the object store bucket was in the public cloud. As NVMe flash system capacity is consumed and usage reaches a high watermark, data is dynamically pushed to the object tier, which means you never have to worry about running out of capacity on the flash tier. This is particularly useful for write-intensive applications, as no administrative intervention is required. The flash tier and the object tier can scale independently depending on the required usage capacities.

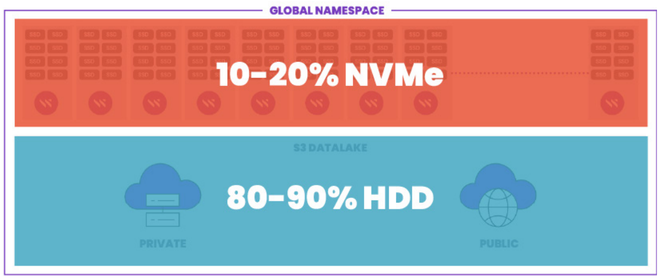

FIG. 12 Integrated Tiering to any S3 Object Store

This integrated capability eliminates the need for additional Hierarchical Storage Management (HSM) or data tiering software that adds complexity and cost. The by-product of the integrated data management features is an elastic unified namespace that can scale to the size of your cloud vendor's capacity limits.

There is no hard and fast rule on how much data should be on NVMe flash versus object store, but a survey of WEKA's customers shows that the typical distribution is now around 20% on flash (As seen in Figure 12). A good rule of thumb is to measure the flash tier such that it will hold the normal working data sets for current workloads, and enough capacity to pre-stage new workloads for maximum performance.

Data Reduction

NeuralMesh integrates advanced data reduction techniques to optimize storage efficiency and reduce costs in high-performance environments. By combining compression and similarity-based deduplication, the platform minimizes the data footprint without compromising performance or integrity. Intelligent algorithms, such as similarity hashing, identify and compress data efficiently, achieving significant reductions—up to 8x for workloads like electronic design automation (EDA) and 10x for AI Inferencing.

WEKA's real-time compression utilizes dynamic dictionaries and similarity hashes to deliver efficient, low-overhead data processing. During the data ingest phase the system computes fingerprints and compresses matching blocks, ensuring a balanced computational and storage load for seamless operations. Compressed blocks are stored sequentially within RAID structures to enhance read and write efficiency. Additionally, NeuralMesh employs a sophisticated garbage collection process to manage fragmented data by rewriting active segments, maintaining storage efficiency while safeguarding data integrity. The design approach is to make data reduction a background process so that it does not hurt write performance or metadata performance.

The benefits of WEKA's data reduction capabilities are substantial. Organizations can achieve significant space savings, reducing storage costs by enabling higher data density without degrading access speeds. Performance remains consistent, as decompression is handled at drive nodes to ensure predictable throughput with minimal latency. Advanced mechanisms, such as robust reference counting and garbage collection mitigate risks of data corruption or loss during operations.

Snapshots and Clones

Snapshots in the WEKA system are a native part of the NeuralMesh design, unlike many other solutions where snapshot is an afterthought or external service. This deep integration ensures efficiency, consistency, and speed. NeuralMesh uses copy-on-write semantics to create a snapshot to ensure the snap is lightweight and fast to create. WEKA leverages the virtual metadata server journals to execute its file system snapshots. The snapshots are instantaneous primarily because the snapshot itself is just a new metadata reference point to the existing data blocks, and it also benefits from the high degree of parallelism across all the virtual metadata servers. Creating a snapshot does not involve copying the actual data, it simply creates a pointer of the file system at a specific moment in time. Because no data blocks are copied during the snapshot creation, the actual time to create a snapshot is independent of the data set size. Whether the file system is 100 gigabytes or 10 petabytes, snapshot creation is instantaneous.

NeuralMesh supports user-definable snapshots for routine data protection including backup as well as for cloud data migration and cloud bursting. For example, snapshots can be used to back up files locally on the flash tier as well as making copies to cloud storage tiers for backup or disaster recovery. Also, snapshots can be saved to lower-cost cold storage such as public cloud and on-premises object storage. In addition to point-in-time snapshots, NeuralMesh can create full clones of the file system (read-only snapshots that can be converted into writable snapshots) with pointers back to the originating data. Snapshots and clones occur instantaneously and are incremental after the first instance, which dramatically reduces the time and storage required for protection. Furthermore, system performance is unaffected by the snapshot process or when writing to a clone. Snapshots can be created from the GUI, CLI or through a REST API call. Snapshots are exposed to clients via a /.snapshot directory. If tiering is enabled, snapshotted data will be moved to the object tier based on the same policies as the active file system.

NeuralMesh supports the following snapshot operations:

- Create a snapshot of an existing file system
- Delete a snapshot
- View snapshots
- Access a snapshot under a dedicated directory name
- Restore a file system from a snapshot
- Writable snapshots
- List snapshots and obtain their metadata
- Schedule automatic snapshots
- Snap-to-object (see next section)

Snap-to-Object

Once tiering is enabled in a file system, NeuralMesh supports a unique feature called Snap-to-Object. This feature enables the committing of all the data of a specific snapshot, including metadata, to an object store. Unlike data lifecycle management processes utilizing tiering, this feature involves copying all the contents of the snapshot, including all files and metadata to an object store. After the first snap-to-object has been completed, subsequent snapshots are stored in an incremental manner, so backup time is limited to just the changes and is extremely fast. WEKA also supports sending snapshots to a second object store using the Remote Backup feature. This leverages the incremental nature of snapshots by only sending the changes across the wire to the destination object store. The object store then only needs to store the incremental capacity of the snapshots at any given time instead of the complete capacity of each snapshot that is uploaded.

Snap-to-Object also has an incremental Snapshot Download capability embedded within it. When an initial snapshot from a source file system is restored into a new file system by downloading the snapshot from the object store, further snapshots of that source file system can then be incrementally restored into the destination file system. With this technology, any updates to the destination file system will be seen by closing the file or directory and reopening it when using the POSIX client, and by closing the file or directory, then invalidating any client caching when using other protocols. The clients do not have to unmount and remount the file system to see the changes that have occurred.

The outcome of using the snap-to-object feature is that the object store contains a full copy of the snapshot of the data, which can be used to restore the data on the original cluster or onto another cluster. The secondary cluster that mounts the snap-to-object snapshot does not need to be a mirror of the primary system. In fact, the primary system could have 20 storage servers in the cluster, while the second system could have 6 or 10, or 100. Consequently, the snap-to-object feature is useful for a range of use cases, as follows:

1. **General Use Cases**

    a. Backup of data to an on-premises or cloud-based object store: If too many hardware components in a NeuralMesh cluster fail beyond recovery because of a failure of the system or an external event such as a fire, earthquake, flood, etc., the snapshot saved to the object store can be used to re-create the same data on another NeuralMesh cluster, or re-hydrate onto the original cluster.

    b. Data archival: The periodic creation of data snapshots, followed by uploading the snapshot to an object store or the cloud to create an archive copy of the data. WEKA can also support "cheaper and deeper" Object store services such as AWS Glacier instant retrieval for this purpose.

c. Asynchronous mirroring of data: Combining a NeuralMesh cluster with a replicated object store in another data center will create a mirror of the data that can be mounted on a second cluster.

2. **Cloud-Only Use Cases**

a. Public Cloud pause and restart: In the various cloud providers, WEKA utilizes compute instances with local SSDs to create a cluster. For bursty project-specific work, users may want to shut down or hibernate the cluster to save costs. The snapshot can be saved to an object store and re-hydrated when needed again at a later time.

b. Protection against single availability zone failure: Utilizing the snap-to-object feature allows users to recover from an availability zone (AZ) failure. Should the first AZ fail, if the WEKA snapshot was replicated to a second AZ via the object store, it can be re-hydrated in minutes by a WEKA cluster in the secondary AZ.

3. **Hybrid Cloud Use Case**

a. Cloud bursting: An on-premises customer can benefit from cloud elasticity by using additional computational power for short periods. By uploading a snapshot to a cloud-based object store, the file system can be run in the cloud. After running in the cloud, the data can be deleted or archived and the compute instances shut down.

Snapshot Compare for Point-in-Time Changes

A WEKA user can compare snapshots from two points in time and list the changes that have occurred leveraging the NeuralMesh Snapshot DiffList function. While this is a common capability in systems, the WEKA approach is significantly more powerful and performant than others.

The common snapshot comparison technique requires comparing the most recent snapshot to the previous one, recording the changes and then comparing the previous one to the one prior to that again, and so on. A tree-walk through the file system can take significant time to execute, and for very large data sets found in areas like AI and life sciences, it would be unacceptable to users. However, NeuralMesh has the unique ability to compare any snapshot to another skipping over the intermediate snapshots without the need to tree-walk. This can speed up a scan by orders of magnitude compared to the traditional approach.

In fact, WEKA has the unique ability to compare any snapshot to the original data set prior to any snapshot being taken.

The DiffList can be consumed through APIs for application integration, and can be used to create incremental backups of the file system or cataloging.

Non-Disruptive Upgrades

WEKA has the ability to be upgraded without impacting clients. Because NeuralMesh uses containers and sets of processes inside these containers as its OS, it is able to upgrade containers in a rolling process without causing a remount of the clients. This capability, when combined with the resiliency of the NeuralMesh data protection schema, allows for WEKA to be upgraded on the fly with only a minimum of I/O pauses during the process. While the file system is mounted and applications are live, the kernel driver and Frontend container can be removed and replaced with a new container upgrade with no perceptible service interruption.

Maintenance of the storage containers can be scheduled to ensure data remains safe. Processes can be paused and resumed until the entire upgrade is complete. All this happens in the background while maintaining full performance to the cluster.

Container Storage Integration

Container Storage Interface (CSI) is a standard that has been developed to provision and manage shared file storage for containerized workloads. The NeuralMesh CSI Plugin for Kubernetes provides an interface between the logical volumes in a Kubernetes environment (Persistent Volumes (PVs)) and the storage, enabling customers to deploy stateless WEKA clients to connect storage to the appropriate container. The CSI plugin provisions a Kubernetes pod volume either via Persistent Volume (by an administrator) or it can be dynamically provisioned via a Persistent Volume Claim (PVC). This feature simplifies the process of moving containerized workloads to the cloud or sharing data across multiple Kubernetes clusters. The CSI plugin also supports using quotas to help manage space consumption for containerized applications.

Security

WEKA implements a comprehensive security framework designed to protect sensitive data, ensure secure communication, and prevent unauthorized access to a NeuralMesh storage system. By integrating robust authentication, encryption, access controls, and monitoring, WEKA provides enterprise-grade security for business-critical data.

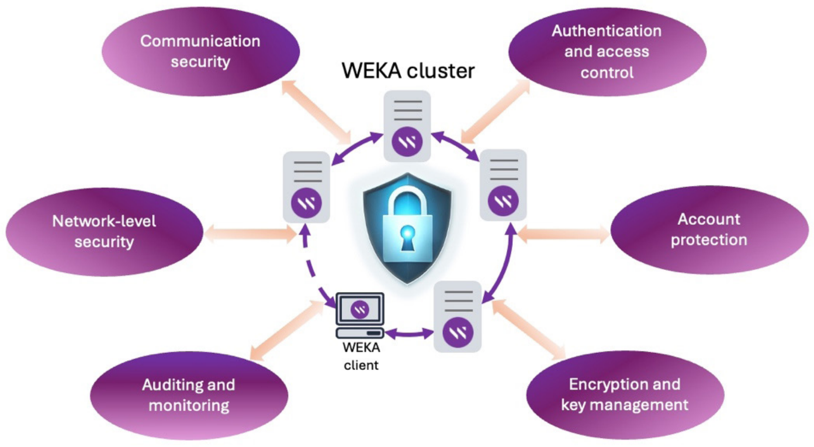

FIG. 13 Multiple Levels of Security

Encryption In-Flight and At-Rest

WEKA provides full end-to-end encryption from the client all the way to the object storage solution, making it the most robust encrypted storage system commercially available. Encryption is set upon creation of the file system, so some file systems that are deemed critical can be encrypted while others are not. When files are encrypted, it means that even if cold blocks underlying the file are sent to an object store tier, the data will remain encrypted. The NeuralMesh encryption solution protects against physical media theft, low-level firmware hacking on the SSD, and packet eavesdropping on the network. File data is encrypted with the FIPS 140-3 Level 1 compliant encryption key XTS-AES using a 512-bit key length.

WEKA has demonstrated that encrypted file systems have minimal impact on application performance when using the POSIX client.

Key Rotation and Key Management

WEKA supports any key management system (KMS) that is compliant with the Key Management Interoperability Protocol (KMIP) 1.2 and above, as well as Hashicorp Vault proprietary API. Cluster keys are rotated by the KMS, file system keys can be rotated via the KMS and re- encrypted with the new KMS master key, and file keys can be rotated by copying the file.

File data on the object store is also encrypted. When uploading a snap-to-object to the object store, among other file system parameters, the file system key is included and encrypted with a special "backup-specific" cluster key that is available via the KMS and is used for all snap-to-object backups and restores. When NeuralMesh pushes a snapshot to an object store, the data is fully protected and can be authenticated only through the KMS system.

Authentication and Access Control

NeuralMesh provides authentication services at the user and client-server level to validate that the user or client can view and access data. It allows different authenticated mount modes such as read-only or read-write and is defined at the file system level.

Authenticated mounts are defined on the Organizational level and are encrypted by an encryption key. Only clients with the proper key can access authenticated mount points. This methodology increases security by drastically limiting access to certain subsets of an organization and limiting access to clients with the proper encryption key. NeuralMesh supports the following:

LDAP (Lightweight Directory Access Protocol): a networking protocol that provides directory services across many different platforms.

Active Directory: a Microsoft implementation of LDAP, a directory service that can store information about the network resources. It is primarily used to authenticate users and groups who want to join the cluster.

Authentication tokens: NeuralMesh employs a dual-token system for secure access management

- Access Tokens – Short lived (5 minutes) for API access and file system mount
- Refresh Tokens – Long lived (configurable, default for 1 month) for obtaining new access tokens

Token expiration management: to enforce security compliance and reduce risk

Secure cluster membership with joint secret authentication: when enabled only storage Compute containers with the correct secret key can join, preventing accidental cross-cluster joins and unauthorized access in a multi-cluster environment.

Role-Based Access Control (RBAC): delivering different levels of privileges to users and administrators. Some users can be granted full access rights while others have read-only rights.

Every NeuralMesh storage system user has one of the following defined roles:

Cluster Admin: A user with additional privileges over regular users. These include the ability to

- Create new users
- Delete existing users
- Change user passwords
- Set user roles
- Manage LDAP configurations
- Manage organizations

Additionally, the following restrictions are implemented for Cluster Admin users to avoid situations where a Cluster Admin loses access to a NeuralMesh system cluster:

- Cluster Admins cannot delete themselves
- Cluster Admins cannot change their role to a regular user role

Organization Admin: A user who has similar privileges to cluster admins, except that these privileges are limited to the organization level within the same file system. They can perform the following within their organization:

- Create new users
- Delete existing users
- Change user passwords
- Set user roles
- Manage the organization LDAP configuration

Furthermore, to avoid situations where an organization admin loses access to a NeuralMesh system cluster, the following restrictions are implemented for organization admins:

- Organizational Admins cannot delete themselves
- Organizational Admins cannot change their role to a regular user role

Regular: A user with read and write privileges. A user that should only be able to mount file systems

- can log-in to obtain an access token
- can change their password
- cannot access the UI or run other CLI/API command

Read-only: A user with read-only privileges

S3: A user to run S3 commands and APIs. This user can operate within the limits of the S3 IAM policy attached to it.

Communication Security

Transport Layer Security (TLS) certificate validation is the process of verifying a website's digital certificate to ensure it is legitimate and trusted, establishing a secure, encrypted connection. This involves checking the certificate's validity, its issuer's trustworthiness, and verifying the digital signature.

TLS certificate management ensures secure HTTPS access (port 14000) with options for

- Default self-signed certificates for cluster communication
- Custom certificates for enhanced security

Cross-Origin Resource Sharing (CORS) implements secure cross-domain access controls to prevent unauthorized API access.

Network Level Security

CIDR based Security

WEKA allows users to restrict access to resources based on the IP address of the requesting client. Instead of specifying individual IP addresses, NeuralMesh supports Classless Inter-Domain Routing (CIDR) notation to define ranges of allowed IP addresses.

Network Segmentation and Firewalls

- Isolates critical systems from unauthorized access
- Supports Intrusion Detection and Prevention System (IDS/IPS)
- Firewalls and network ACLS to restrict communication

Observe – Intelligent, Scalable Observability

Observe introduces a powerful observability hub designed for petabyte-scale environments. It delivers deep, real-time insight into system performance, I/O behavior, and resource utilization—enabling proactive issue resolution, efficient optimization, and streamlined operations for AI and HPC teams.

Management

WEKA provides three quick and easy ways to manage a NeuralMesh storage system, either through a Graphical User Interface (GUI), a Command Line Interface (CLI), or a Representational State Transfer API (REST). Reporting, visualization, and overall system management functions are accessible using the REST API, CLI or the intuitive GUI-driven management console.

GUI

Point-and-click simplicity allows users to rapidly provision new storage: create and expand file systems within a single namespace, establish tiering policy, data protection, encryption, authentication, permissions, NFS, SMB and S3 configuration, read-only or read-write snapshots, snapshot-to-objects, and quality of service policies, as well as monitor overall system health. Detailed event logging provides users the ability to view system events and status over time or drill down into event details with point-in-time precision via the time-series graphing function.

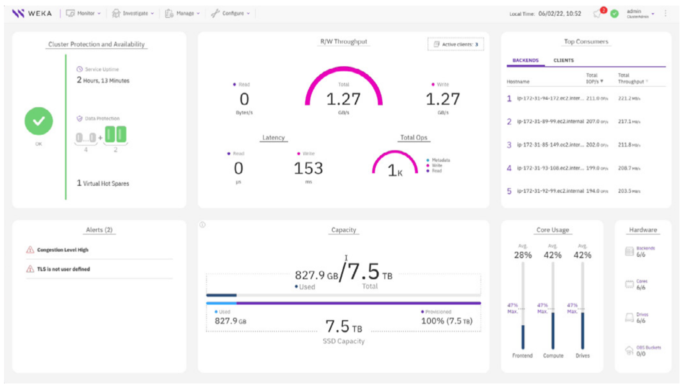

FIG. 14 NeuralMesh GUI

A System Events menu lists the events that have occurred in a NeuralMesh environment. The events displayed in this window are also transmitted to the WEKA Support Cloud so that they can be used by WEKA support to actively assist when necessary, or to proactively notify customers when action needs to be taken. The NeuralMesh GUI is entirely web-based and self-contained, eliminating the need to physically install and maintain any software resources, and ensures users always have access to the latest management console features.

Command Line Interface (CLI)

All NeuralMesh system functions and services can be executed via a CLI command. Most system commands are system wide and deliver the same result across all cluster nodes. Some commands are executed on specific nodes such as IP address management.

REST API

All NeuralMesh system functions and services can be executed via a web service API, which adheres to the RESTful API architecture. Like the CLI, most RESTful commands are system-wide and deliver the same results on all cluster nodes. Some commands are executed on specific nodes. The API is presented via a Swagger interface for ease of use and examples of API code in multiple programming languages. You can find an example of the Swagger interface at https://api.docs.weka.io

Auditing and Monitoring

A comprehensive and robust auditing capability is critical to meet requirements from regulatory standards (such as HIPAA or GINA), to respond to security incidents, and to ensure overall data integrity - this is the primary use case of auditing.

Most of the systems that offer audit logging do it in a way that has a significant impact on the performance of the data being audited, to the point where it is often disabled except when absolutely required. The design philosophy behind the NeuralMesh Auditing system was to ensure it could keep up with continuous streaming and be sufficiently lightweight such that it did not impact performance to the applications.

From the get-go WEKA has collected traces of every event that happens in the system implemented in a manner that has no identifiable impact on performance. When an I/O is sent to any container in the system, it can also send an event notification to a trace buffer. These traces are used in production environments to debug issues with the system. Leveraging this same system for operations that need to be audit logged, NeuralMesh sends the audit events from the Compute container to a separate trace buffer. Each Compute container has its own separate Telemetry buffer, and the contents are forwarded to the Telemetry container almost instantaneously. The Telemetry container has a process to enrich the events asynchronously with other information that the audit system wants to capture (for example adding the file system name or reconstructing the complete file path). This enrichment process allows WEKA to collect the minimum amount of event data, minimizing its buffer footprint and maintaining the system performance. Additionally, if for some reason event traces were lost in the buffer, NeuralMesh audit logging can report the point in time when the last issue happened and report it.

All the Audit messages are written in JSON format, and WEKA records the GUID of the cluster and the file system ID as well as the file system name such that the chain of custody is maintained.

The Telemetry container has an additional process that provides an external gateway and allows the data to be exported to an external entity such as Splunk, S3 or Kafka.

Performance Proof Points

WEKA believes that performance claims should be backed up with external validated proof points. The following links provide public proof of WEKA's performance in production environments. All of these public benchmarks have been independently validated by industry peers.

The following results have been audited on the SPECstorage Solution 2020 Benchmark https://www.spec.org/storage2020/results/ai_image/

https://www.spec.org/storage2020/results/eda_blended/

https://www.spec.org/storage2020/results/genomics/

https://www.spec.org/storage2020/results/swbuild/

https://www.spec.org/storage2020/results/vda/

The following results have been submitted by production customers to the IO500 High performance computing list.

Overall List ranking - https://io500.org/

Samsung - https://io500.org/submissions/view/748

Memorial Sloan Kettering Cancer Center - https://io500.org/submissions/configuration/719

Conclusion

The NeuralMesh storage system has been designed from the ground up to address the current and future challenges of data intensive workloads. Between 2025 and 2030 IDC estimates that data will triple from 175 Zettabytes (ZB) to 600 ZB largely driven by machine learning and inference growth. Data will be generated continuously by everything from cars to personal wear. Managing data efficiently will be as critical as generating or storing it. Current solutions are built on architectures designed in the last century and are struggling to manage today's volume of data. These systems will collapse under future projected growth. Only WEKA has been designed and built to solve tomorrow's challenges and beyond with a radical design built on modern container-oriented architecture, with the performance, resiliency and enterprise capability that is table stakes for any deployment.