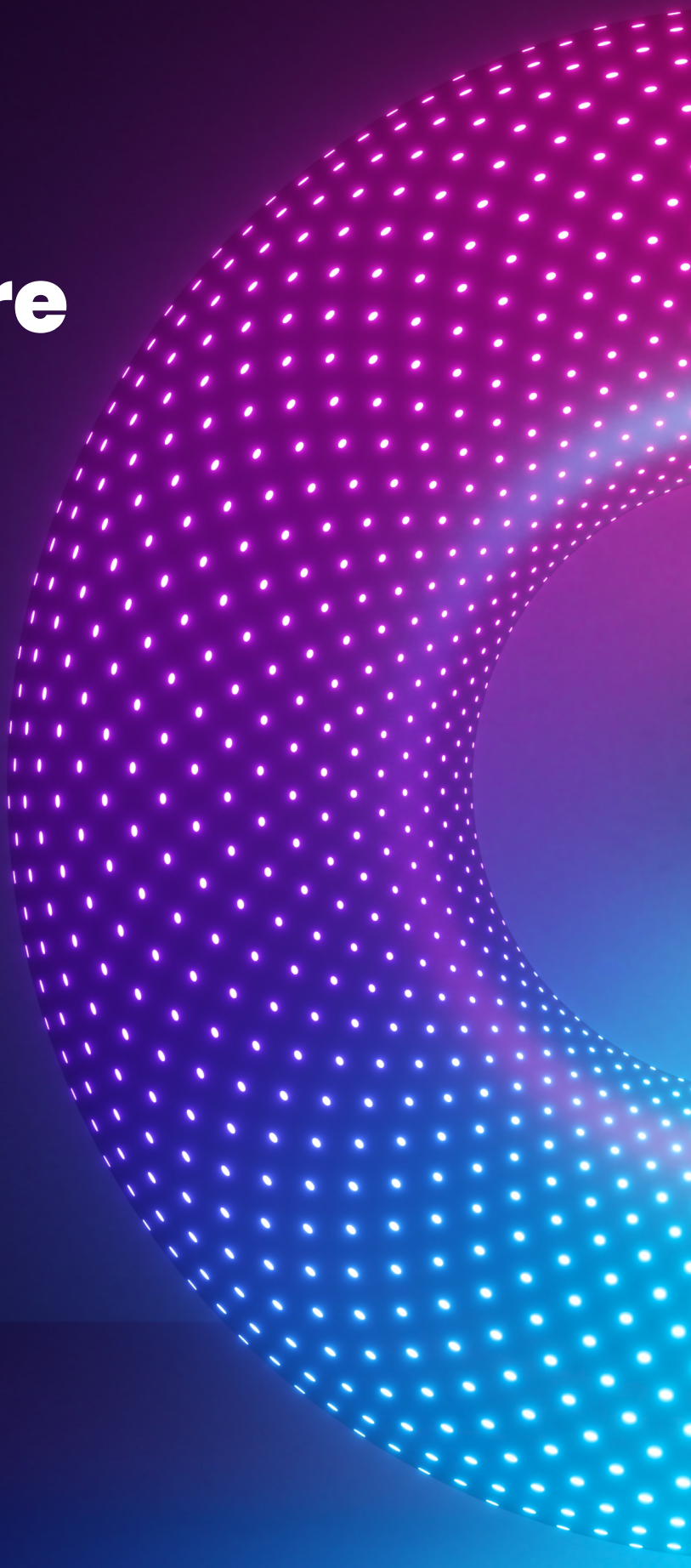
WEKA

WekaFS Architecture

March 2022

White Paper



Summary
and Revisions

This white paper provides a technical overview of the features and benefits of the WekaFS™ file system, including details on theory of operation, features and management, and independent performance validation.

Revisions

Date	Description
Jul. 2017	Initial Document, authored by Liran Zvibel and David Hiatt.
Jun. 2019	Feature updates, authored by Barbara Murphy.
Jul. 2020	Major update with expanded content to include new features, performance, sizing, best practices, authored by Barbara Murphy.
Mar. 2021	Feature updates, authored by Barbara Murphy.
Mar. 2022	New and updated features, updated performance, authored by Joel Kaufman.

Contents

Introduction

WekaIO™ (Weka) was founded on the idea that current storage solutions have only provided incremental improvements to legacy designs, allowing for a widening gap between compute performance and data storage performance. Storage remains a bottleneck to application performance, and with the continued densification of compute in areas such as GPU-based applications, has become even more problematic. In today's hyper-competitive market, organizations need flexible infrastructure; application workloads are becoming increasingly complex and data sets are continuing to grow unchecked, forcing enterprises to architect overly complicated and costly systems that reduce IT agility. As a result, important business insights remain locked away, out of reach of decision makers.

IT organizations are adopting cloud technology for its fluid, on-demand scalability that supports diverse workloads at scale. However, while network and compute can be virtualized to operate at scale very effectively, storage remains largely isolated in silos based on system performance profiles. Consequently, organizations are forced to architect a storage system that is highly customized for their environment and workloads from building blocks that do not scale. The result is a storage solution that is complex, temperamental, expensive, and slow.

Weka has built a software-only, high-performance file-based storage solution that is highly scalable and easy to deploy, configure, manage, and expand. The design philosophy behind the Weka file system (WekaFS™) 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.

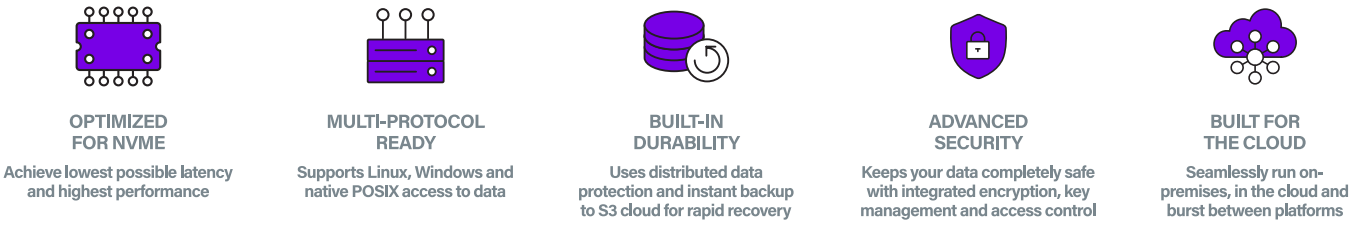


FIG. 1 WekaFS Benefits Summary

WekaFS Highlights (Fig. 1)

- World's fastest shared file system as validated on SPEC SFS 2014[1], SPEC Storage 2020[2], IO-500[3] and STAC[4] benchmarks
- Supports bare-metal, containerized, virtual, and cloud (on-prem, hybrid, public) environments
- Deployable as a converged platform, a dedicated storage appliance, or native in the cloud
- Flexible application storage access, including POSIX, NFS, SMB, S3, and NVIDIA® GPUDirect® storage
- Zero performance tuning to support small and large files simultaneously, with both mixed random and sequential I/O patterns
- Application-level 4K I/O, consistent sub-250 microsecond latency on high-speed networks, unlimited random IOPs performance – linear scaling in line with the size of the cluster

1 https://www.weka.io/press-releases/wekaio-matrix-cements-storage-leadership-groundbreaking-performance-latency-results-spec-sfs-2014/
2 https://www.spec.org/storage2020/results/
3 https://www.weka.io/press-releases/wekaio-places-first-on-io-500-challenge/
4 https://www.weka.io/blog/weka-hpe-and-kx-systems-set-17-new-stac-m3-records/

- Automated built-in hybrid storage to expand the namespace from fast flash to hard disk storage via tiering to on-premises or cloud object storage
- Strong security features including encryption (at-rest and in-flight), authentication, key management, LDAP
- Fully distributed data and metadata to ensure that there are no hotspots in the storage cluster
- Distributed resilience that eliminates the bottlenecks of traditional data protection
- Full cloud integration with cloud-bursting for a hybrid cloud model or 100% public cloud
- Snapshots and snapshot to the cloud for backup, archive and disaster recovery
- Filesystem cloning for rapid development and testing workflows
- Full GUI, CLI and API management

Common Storage Challenges In The Cloud Era

Modern day applications have a wide variety of storage performance requirements (IOPs, bandwidth, latency), and when combined with the diversity of application file formats, access protocols, and data structures, it can all lead to increased IT complexity.

Storage architects try to work around these limitations by employing multiple storage architectures for specific applications. All-flash Storage Area Networks (SAN) and all-flash arrays (AFAs) are optimized for performance but they provide neither the scale of the cloud nor the simplicity and shareability of network-attached storage (NAS). Furthermore, SANs provide block storage protocols that are not shareable across servers, thus they are unsuitable for shared storage use cases. The result of these workarounds has been a costly and endless cycle of rearchitecting infrastructure to keep pace with changing application requirements.

It is a daunting task to determine which storage solution is best suited to a particular environment or application workload because of the variety of options available. Some solutions are optimized for performance while others are optimized for scale. Workloads in the technical compute space, such as Artificial Intelligence (AI) and Machine Learning (ML), genomic research, and financial analytics, all of which generate both large file sequential access and small file random access on very large data sets, are especially problematic. No single traditional storage design has been able to address all these workload patterns. The workaround has always been to use multiple storage systems and complex data management platforms.

Designing A Unified, Cloud Era Storage Solution

When designing a modern storage solution, a key consideration is to account for the continuing evolution and improvement of technology. A truly software-defined storage solution should accommodate such changes, which means that it must be able to run on commodity server hardware, adapt to customer environments, and add cloud-like agility, scalability, and on-demand performance. It should also be simple to deploy and expand fluidly without incurring the typical procurement delays associated with traditional external storage appliances.

The limitations created by legacy design constraints led the founders of Weka to develop a brand new file system that delivers the performance of all-flash arrays, the simplicity of scale-out NAS, and the scalability of the cloud in a single architecture.

The Weka file system, WekaFS™, is a software-only storage solution with a clean sheet design that solves the problems associated with traditional storage systems. It runs on any standard AMD or Intel x86-based server hardware with commodity NVMe solid-state disks (SSDs), eliminating the need for custom specialized hardware. This approach allows you to take advantage of improvements in technology without the pain of complete forklift upgrades to next-generation architectures, including public cloud deployments.

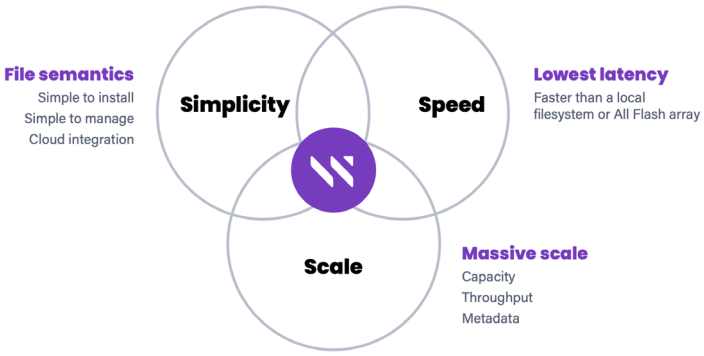


FIG. 2 The Weka File System Design Approach

WekaFS – Storage Without Compromise

WekaFS solves the common storage challenges previously mentioned by eliminating the chokepoints that impact application performance. It is well-suited for demanding environments that need shareable storage with low-latency, high-performance, and cloud scalability.

Example use cases include:

- Artificial intelligence (AI) and machine learning (ML), including AIOps and MLOps
- Life sciences including genomics, Cryo-EM, pharmacometrics (NONMEM, PsN)
- Financial trading, including backtesting, 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, Weka's software 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). Furthermore, because it is designed to run on commodity server infrastructure, it does not rely on any specialized hardware.

WekaFS is a fully distributed parallel file system that was written entirely from scratch to deliver the highest-performance file services by leveraging NVMe flash. The software also includes integrated tiering that seamlessly expands the namespace to and from hard disk drive (HDD) object storage, without the need for special data migration software or complex scripts; all data resides in a single namespace for easy access and management. The intuitive graphical user interface allows a single administrator to quickly and easily manage hundreds of petabytes of data without any specialized storage training.

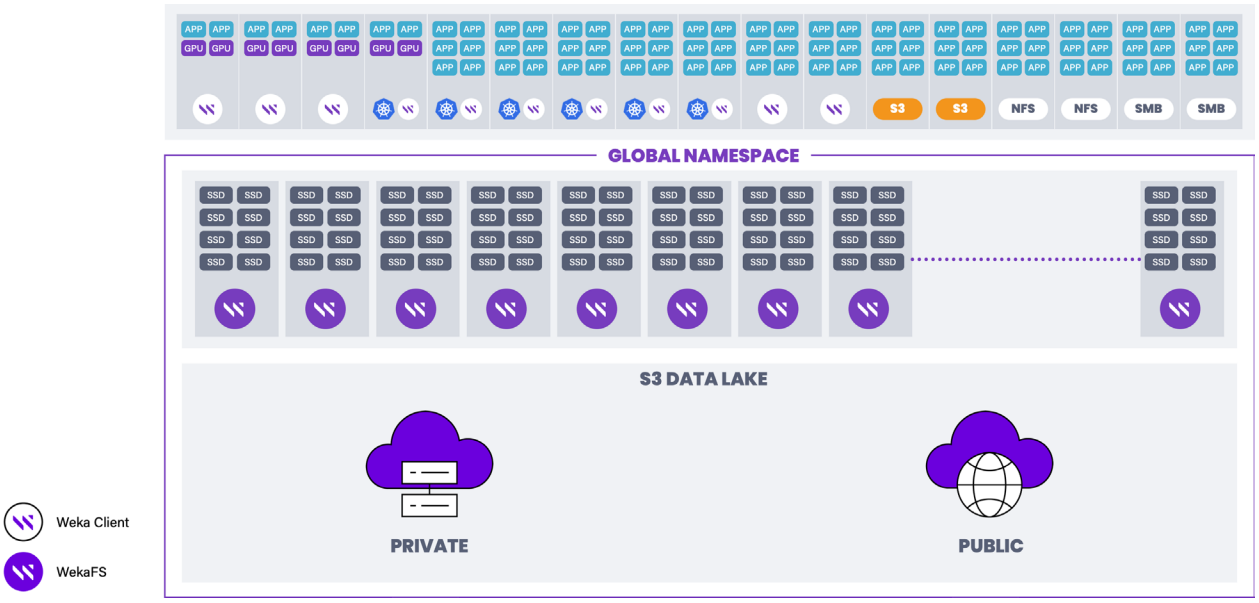


FIG. 3 WekaFS combines NVMe flash with cloud object storage in a single global namespace

WekaIO's unique architecture, as shown in FIG. 3, is radically different from legacy storage systems, appliances, and hypervisor-based software-defined storage solutions because it not only overcomes traditional storage scaling and file sharing limitations but also allows parallel file access via POSIX, NFS, SMB, S3 and GPUDirect Storage. It provides a rich enterprise feature set, including local snapshots and remote snapshots to the cloud, clones, automated tiering, cloud-bursting, dynamic cluster rebalancing, private cloud multi-tenancy, backup, encryption, authentication, key management, user groups, quotas with advisory, soft and hard parameters and much more.

WekaFS Benefits

- **Highest performance** – ideal for mixed small and large file workloads
- **Scalable capacity** – start as small as 30TB and scale to hundreds of petabytes in a single namespace
- **Strong security** – keep data safe from threat or rogue actors with encryption and authentication
- **Hybrid Cloud** – burst to AWS for compute agility or run natively in the cloud
- **Backup** – push backups straight to private or public cloud for long term retention
- **Best economics** – combine flash and disk for best cost at scale

WekaFS Architecture

Weka's parallel file system is designed to provide a cloud-like experience, whether you run your applications on-premises or plan to move them to the cloud. WekaFS provides a seamless transition to the cloud and back. Most legacy parallel file systems overlay file management software on top of block storage, creating a layered architecture that impacts performance. WekaFS is a distributed, parallel file system that eliminates the traditional block-volume layer managing underlying storage resources. This integrated architecture does not suffer the limitations of other shared storage solutions and delivers both scalability and performance effectively.

FIG. 4 below provides an overview of the software architecture from the application layer all the way to the physical persistent media layer. The Weka core components, including the WekaFS unified namespace and other functions such as virtual metadata servers (MDSs), execute in user space in a Linux container (LXC), effectively eliminating time-sharing and other kernel-specific dependencies. The notable exception is the Weka Virtual File System (VFS) kernel driver, which provides the POSIX filesystem 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 system.

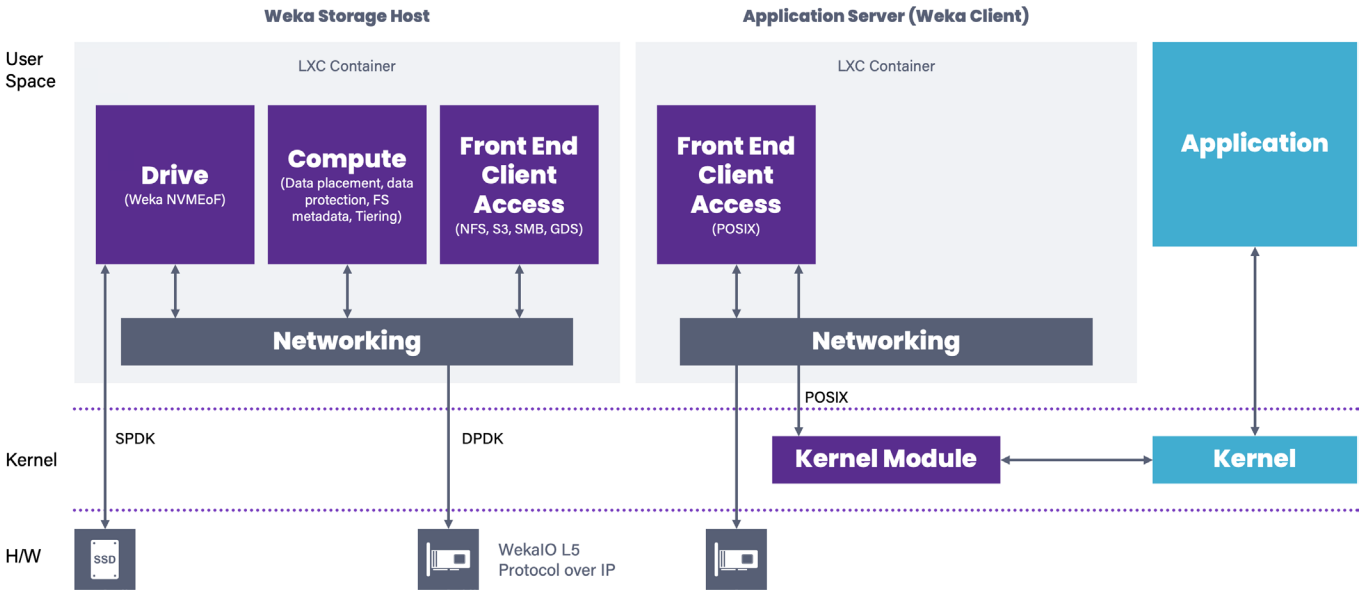


FIG. 4 WekaFS Software-Based Storage Architecture

Weka supports all major Linux distributions and leverages virtualization and low-level Linux container techniques to run its own RTOS (Real-Time Operating System) in user space, alongside the original Linux kernel. Weka 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. By not relying on the Linux kernel, WekaFS minimizes context switching, resulting in a shorter IO path and predictable low latencies. It also allows upgrading of the WekaFS backends independently of Linux OS and Weka client (front end) upgrades.

WekaFS functionality running in its RTOS (FIG. 4) is comprised of the following software components:

- File Services (Front End) – manages multi-protocol connectivity
- File System Clustering (Back End) – manages data distribution, data protection, and file system metadata servicesSSD Access Agent – transforms the
- SSD into an efficient networked deviceManagement Node – manages events, CLI, statistics, and call-home capabilityObject Connector – read and write to the object store
- Management Node – manages events, CLI, statistics, and call-home capability
- Object Connector – read and write to the object store

Weka core software in the RTOS runs inside LXC containers that has the benefit of improved isolation from other server processes. Weka software, when deployed, is containerized as microservices: Multiple containers for SMB, NFS, S3, and core WekaFS may exist per host. By spanning multiple LXC containers, Weka enables even greater parallelism and the ability to use more CPU cores and RAM than a single LXC container. A Weka VFS driver enables WekaFS to support full POSIX semantics and leverages lockless queues for I/O to achieve the best performance while enhancing interoperability. The WekaFS POSIX file system has the same runtime semantics as a local Linux file system (e.g., Ext4, XFS, and others), enabling applications that previously could not run on NFS shared storage because of POSIX locking requirements, MMAP files, performance limitations, or other reasons. These applications will enjoy massively improved performance compared to the local file system (FIG. 14).

Bypassing the kernel means that Weka's software stack is not only faster with lower latency, but is also portable across different bare-metal, VM, containerized, and cloud instance environments.

Resource consumption is often a problem with traditional software-based storage designs because these solutions either take over the entire server or share common resources with applications. This extra software overhead introduces latency and steals precious CPU cycles. By comparison, Weka only uses the resources that are allocated to it inside its LXC containers, which means it can consume as little as one server core and a small amount of RAM in a shared environment (converged architecture- application and storage software sharing the same server) or as much as all the resources of the server (a dedicated appliance). The same software stack is utilized in either case.

File System Design

From the outset, WekaFS was designed to solve many of the problems inherent with legacy scale-out NAS solutions. One of the key design considerations was to build a software platform that could address the requirements of different user groups within an organization at scale, or a multi-tenant environment. The most popular scale-out NAS file systems support a construct of a single file system and a single namespace, utilizing directories and quota systems to allocate resources and manage permissions. While this solution worked at smaller scale, it has made management complex when the number of users and/or directories scale. Full isolation of user groups requires the creation of new file systems and namespaces, which then creates islands of physical storage to manage. Additionally, directory scaling is a problem and typically requires creating multiple directories to maintain performance, further exacerbating the complexity. As such WekaFS differs from other scale-out NAS solutions in that it embraces the concept of many file systems within the global namespace that share the same physical resources. Each file system has its own "persona" and can be configured to provide its own snapshot policies, tiering to object store, organizations, role-based access control (RBAC), quotas and much more. A Weka file system is a logical construct and unlike other solutions, the file system capacity can be changed on the fly. Clients that are mounted can observe the change in file system 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 on the fly. A file system can be split into multiple organizations managed by their own administrator.

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 WekaFS supports up to 1024 files systems, and up to 4096 snapshots in a single global namespace.

WekaFS Limits:

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

Supported Protocols

Clients with the appropriate credentials and privileges can create, modify, and read data using one of the following protocols:

- POSIX
- NVIDIA® GPUDirect® Storage (GDS)[5]
- NFS (Network File System) v3
- SMB (Server Message Block) v2 and v3
- S3 (Simple Storage Service)

NOTE: Many non-traditional applications and data systems can take advantage of the POSIX capabilities that WekaFS provides as it appears as a local mount. One example of this is HDFS (Hadoop Distributed File System); Weka's POSIX connector can directly mount to Hadoop nodes to provide very high performance.

Data written to the file system from one protocol can be read via another one, so the data is fully shareable among applications.

Weka Storage Hosts

Weka storage hosts are created by installing WekaFS on any standard AMD EPYCTM or Intel XeonTM Scalable Processor-based server with the appropriate memory, CPU processor, networking, and NVMe solid-state drives. A configuration of 6 storage hosts is required to create a cluster that can survive a two-host failure. To create an appliance-like experience, Weka has worked with leading server vendors to create a single part number that can be used to order a complete storage system inclusive of a software license. More details on specific configurations from Cisco, Dell, HPE, Hitachi Vantara, Lenovo, Penguin, and Supermicro as well as AWS are available on those vendors websites.

For all other platforms, each host must conform to the following minimum specification outlined below in Tables 1 and 2.

5. NVIDIA GPUDirect Storage is a protocol developed by NVIDIA to improve bandwidth and reduce latency between the NIC and GPU memory. It is currently available on select NVIDIA GPU-based systems.

Table 1: Minimum Hardware Specification for Intel Scalable Processor-Based Server

Component	Description	Minimum Quantity
Processor	AMD EPYC 7402P processor	1
Memory	DDR4-2666, 16GB DIMM	8
SSD (for boot device)	M.2 or U.2 SATA, 480GB or larger	1
SSD (for WekaFS)	U.2 NVMe, 960GB or larger	4
Network Interface Card (Ethernet or InfiniBand)	10Gb or above with DPDK support	1

* WekaFS requires an x86-64 architecture, with a minimum of one processor with 2.2GHz and 12 cores.

Table 2: Minimum Hardware Specification for AMD EPYCTM Scalable Processor-Based Server

Component	Description	Minimum Quantity
Processor	AMD EPYC 7402P processor	1
Memory	DDR4-2666, 16GB DIMM	8
SSD (for boot device)	M.2 or U.2 SATA, 480GB or larger	1
SSD (for WekaFS)	U.2 NVMe, 960GB or larger	4
Network Interface Card (Ethernet or InfiniBand)	10Gb or above with DPDK support	1

NOTE: In order to achieve the Performance referenced in later sections of this paper, 100Gb or above NICs will be required.

Integrated Flash and Disk Layers for Hybrid Storage

The WekaFS storage design consists of two separate layers, an NVMe SSD-based flash layer that provides high-performance file services to the applications, and an optional S3-compatible hard disk-based object storage layer that manages the long-term data lake (outlined in FIG. 3). The two layers can be physically separate, but logically serve as one extended namespace to the applications. WekaFS expands the namespace from the NVMe flash layer to the object store, presenting a single global namespace that scales to exabytes. The object store can be from any S3-compliant vendor for either on-premises or public cloud deployment. WekaFS only requires the presence of an S3 bucket, so an existing object store can be shared with WekaFS for the namespace extension, while still supporting other applications in separate buckets. As we will see later, WekaFS leverages components of the object store capability to enable cloud bursting, backup to the cloud, DR to another WekaFS cluster, or file system cloning.

Networking

The Weka system supports the following types of networking technologies:

- InfiniBand (IB) HDR and EDR
- Ethernet – 10Gbit minimum, 100Gbit and above recommended

The customer's available networking infrastructure dictates the choice between the two, as WekaFS delivers comparable performance on either one. For networking, the Weka system does not use standard kernel-based TCP/IP services, but a proprietary networking stack based on the following:

- Use of DPDK to map the network device in the user space and make use of the network device without any context switches and without copying data between kernels. This bypassing of the kernel stack eliminates the consumption of kernel resources for networking operations and can be scaled to run on multiple hosts. It applies to both backend and client hosts and enables the Weka system to fully saturate up to multiple 200Gbit Ethernet or InfiniBand links.
- Implementation of a proprietary Weka protocol over UDP, i.e., the underlying network may involve routing between subnets or any other networking infrastructure that supports UDP. Clients can be on different subnets, as long as they are routable to reach the storage nodes.

The use of DPDK delivers operations with high throughput and extremely low latency. Low latency is achieved by bypassing the kernel and sending and receiving packages directly from the NIC. High throughput is achieved because multiple cores in the same host can work in parallel, eliminating any common bottleneck.

For legacy systems that lack support for SR-IOV (Single Root I/O Virtualization) and DPDK, WekaFS 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 hardware such as the Mellanox CX3 family of NICs.
In addition to being compatible with older platforms, the UDP mode does not dedicate CPU resources, but will yield CPU resources to other applications. This can be useful when the extra CPU cores are needed for other purposes.

For RDMA-enabled environments, common in GPU accelerated computing, WekaFS supports RDMA for InfiniBand and Ethernet to supply high performance without the need to dedicate cores to the Weka front-end processes.

Application clients connect to the Weka storage cluster via Ethernet or InfiniBand connections. The Weka software supports 10GbE, 25GbE, 40GbE, 50GbE, 100GbE, 200GbE Ethernet networks, and EDR and 200Gb HDR 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. WekaFS 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.

WekaFS supports VMXNET3 networking from VMware. When the Weka client 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 client will continue to remain connected to WekaFS storage.

A list of supported NICs that work with Weka is available at https://docs.weka.io/v/3.14/support/prerequisites-and-compatibility#networking

Network High Availability (HA)

WekaFS 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 Weka system must be configured with no single component representing a single point of failure. Multiple switches are required, and hosts must have a connection to each switch. HA for clients is achieved through the implementation of two network interfaces on the same client. WekaFS 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, WekaFS supports failover of Infiniband to Ethernet within the storage cluster to maintain high availability in case the Infiniband network fails. This failover does not apply to clients which much be on one type of network or the other.

WekaFS can easily saturate the bandwidth of a single network interface card (NIC). For higher throughput, it is possible to leverage multiple NICs. Using a non-LACP approach sets a redundancy that enables the Weka software to utilize two interfaces for HA and bandwidth, respectively.

When working with HA networking, it is useful to hint the system to send data between hosts through the same switch rather than using the switch interconnect (ISL). The Weka system achieves this through network port labeling, which also ensures ease of use. This can reduce the overall traffic in the network.

NOTE: Unlike RoCE implementations that require Priority-based Flow Control (PFC) to be configured in the switch fabric, Weka does not require a lossless network setting to support its NVMe-over-fabrics implementation and can even deliver this level of low latency performance in public cloud networks.

Protocols

WekaFS 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. Unlike other parallel file systems, WekaFS does not require additional management server infrastructure to deliver this capability. The following list includes all currently supported protocols:

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

POSIX

The Weka system client is a standard, POSIX-compliant filesystem driver installed on application servers, that enables file access to Weka filesystems. Like any other filesystem driver, the Weka system client intercepts and executes all filesystem operations. This enables the Weka system to provide applications with local filesystem semantics and performance, while providing a centrally managed, sharable, and resilient storage platform. WekaFS 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. WekaFS has full support for GDS and has been validated by NVIDIA including a reference architecture at https://www.weka.io/promo/nvidia-ai-reference-architecture/

NFS

The NFS protocol allows remote systems to access the Weka file system from a Linux client without the WekaFS client. While this implementation will not deliver the performance of Weka's POSIX client, it provides a simple way to deploy and share data from the Weka storage cluster. WekaFS currently supports NFS v3.

SMB

The SMB protocol allows remote systems to connect to shared file services from a Windows or macOS client. The protocol provides a scalable, resilient and distributed implementation of SMB, supporting a broad range of SMB capabilities including:

- User authentication via Active Directory (Native and mixed mode)

- 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

NOTE: WekaFS currently supports SMB v2.x and v3.x

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 an S3 front-end support on its performance file system to accelerate S3 storage I/O. In particular, WekaFS delivers huge performance gains for small file I/O accessed via S3. The S3 API on WekaFS 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 Weka S3 implementation supports multiprotocol access, TLS, has full S3 audit logs, and has bucket level features such as policies, quotas-per-bucket, and Expiry rules for information lifecycle management. For more information, see https://docs.weka.io/additional-protocols/s3

Management GUI

Weka provides three quick and easy ways to manage the Weka file system, either through a Graphical User Interface (GUI), or a Command Line Interface (CLI), or 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 (see FIG. 5).

Point-and-click simplicity allows users to rapidly provision new storage; create and expand file systems within a global 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. 6).

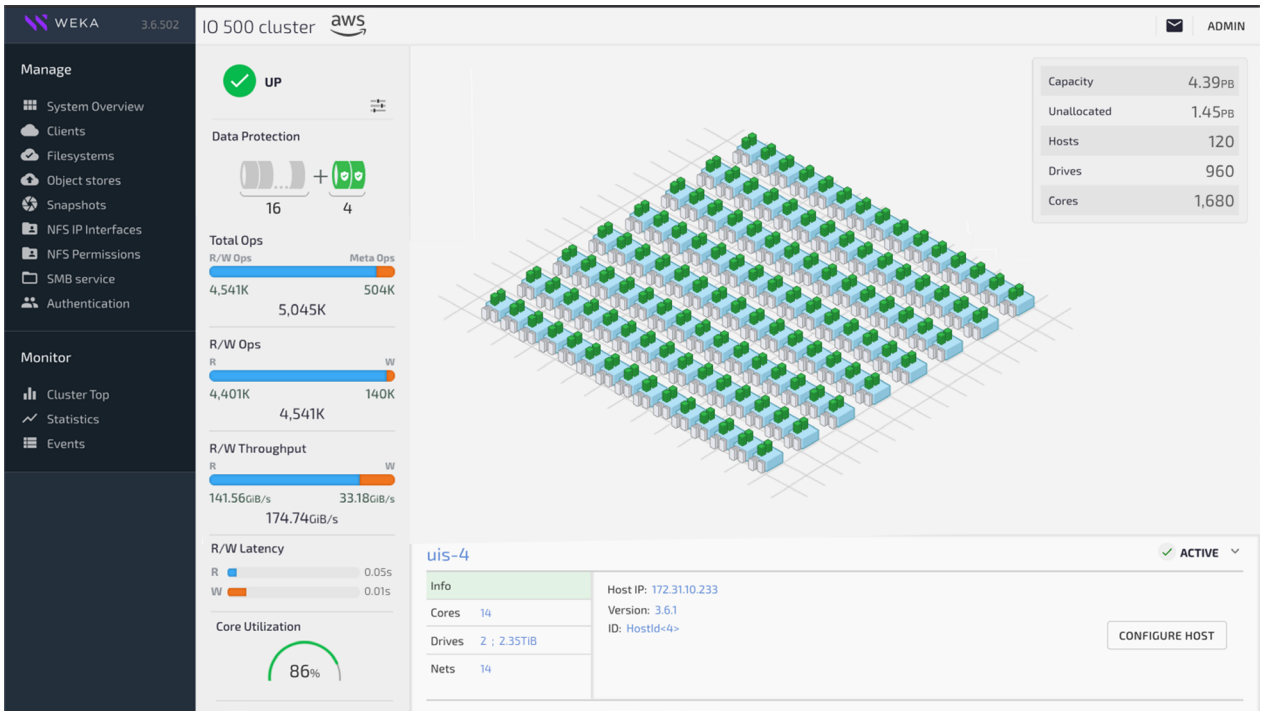


FIG. 5 WekaFS Management Software User Interface

A System Events tab lists the events that have occurred in a Weka environment. The events displayed in this window are also transmitted to the Weka Support Cloud (Weka Home; https://home.weka.io), so that they can be used by Weka support to actively assist you when necessary or to proactively notify you when action needs to be taken. The WekaFS GUI is entirely web-based and self contained, eliminating the need to physically install and maintain any software resources, and you always have access to the latest management console features.

FIG. 6 Time-Series Charts for Event Monitoring

Command Line Interface (CLI)

All WekaFS system functions and services can be executed via a CLI command. Most Weka 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 WekaFS system functions and services can be executed via a web service API, which adheres to the RESTful API architecture. Similar to the CLI, most WekaFS 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 a example of the Swagger interface at https://api.docs.weka.io/

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. The Weka file system 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. WekaFS supports leveraging Linux page cache — typically reserved for direct attached storage (DAS) or file services run 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. WekaFS will now 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 WekaFS dynamically manages caching, making the provisioning of the Weka environment very simple to manage with no danger of an administrative error causing data corruption.

WekaFS 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, Weka 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, no other file system provides the adaptive caching capability of WekaFS. Caching 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 administrator intervention as Weka does not depend on battery protection to protect acknowledged writes. As a result, the same software that runs on-premises can be seamlessly deployed in the public cloud with no software changes. 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.

Global Namespace and Expansion

WekaFS manages all data within the system as part of a global namespace and supports two persistent storage tiers in a single hybrid architecture — NVMe SSD for active data and HDD/Hybrid flash-based object storage for a data lake. Expanding the namespace to object store is an optional addition to the global namespace and can be configured with a few clicks from the management console. 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. When a file is tiered to the object store, the original file is kept on the flash layer until the physical space is required by new data, and hence acts as a cached file until overwritten. When file data is demoted to the object store tier, 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[5]. 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 administrator intervention is required. The flash tier and the object tier can scale independently depending on the required usage capacities.

The global namespace can be sub-divided into 1024 file systems and file system capacity can be expanded at any time on-the-fly without the need to unmount and mount the file system, simply by allocating more space to it. By segmenting the namespace, storage capacity can be allocated to individual users, projects, customers, or any other parameter, yet be easily and centrally managed. Data within a file system is fully isolated from every other file system to prevent noisy neighbor issues.

NOTE: An on-premises flash tier, with a cloud-based object tier would suffer the performance penalty of a WAN connection and the ingress/egress costs of cloud.

Thin Provisioning

WekaFS allows thin provisioning of filesystems within the global namespace. When additional Hosts 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 hosts or drives are added, but also in managing space if hosts or drives are removed from the cluster. Available capacity remaining after removal of hosts or drives must be enough to support the amount of data stored in the flash tier for all the filesystems. This feature also enables seamless integration with EC2 auto scaling groups in AWS.

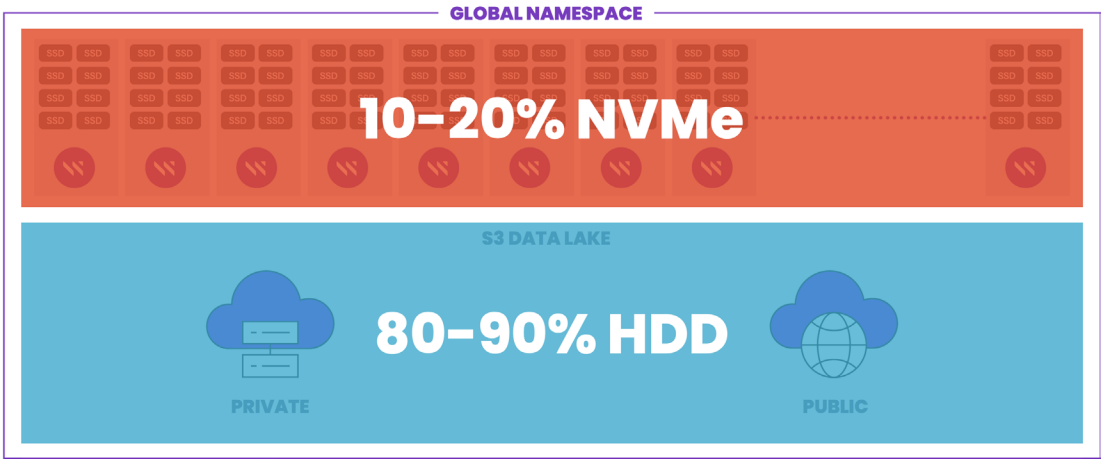


FIG. 7 Integrated Tiering to any S3 or Swift-Compatible Object Store

Integrated Tiered Data Management

WekaFS has a built-in, policy-based automated data management feature, that transparently moves data across storage types according to the data temperature. Weka supports moving data from the NVMe flash storage tier to on-premises or cloud-based object storage (FIG. 7). Data movement is set at the per-file 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 a given file system in the location they were written, regardless of their tiering status, thus no change in application is needed to leverage cost-optimized object storage-based solutions.

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. A list of certified object stores can be found at https://docs.weka.io/support/prerequisites-and-compatibility#object-store

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 FIG. 7). 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 Migration to WekaFS

The object storage tier is an ideal infrastructure choice for building out a large data warehouse or data lake for ongoing data analytics and data insights. WekaFS has the ability to consolidate multiple data lakes into a single massively scalable data warehouse allowing for a single exascale namespace with high performance data access and data processing.

Migrating from end-of-life hardware platforms or remote buckets can be challenging for administrators as data ingest will flood the file system, crushing the performance for any other application running normal I/O on the system. WekaFS allows a mount option direct to the object store, that passes the ingested data directly to the object store without consuming the flash tier capacity. This allows for a "lazy migration" of the Weka storage system, while maintaining normal operations for other applications.

Snapshots and Clones

Weka supports user-definable snapshots for routine data protection including backup as well as for cloud data migration and cloud bursting. For example, WekaFS 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, Weka 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, WekaFS can create full clones of the filesystem (read-only snapshots that can be converted into writable snapshots) with pointers back to the originating data. WekaFS 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. WekaFS support:

- Read-only snapshots
- Read/write snapshots
- Delete primary snapshot, keeping all other versions
- Delete any snapshot, keeping previous and later versions
- Convert read-only to read/write snapshots
- Snap-to-object (see next section)

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 filesystem.

Snap-to-Object

Once Tiering is enabled in a file system, WekaFS 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 snapshot-to-object has been completed, subsequent snapshots are stored in a incremental manner so backup time is limited to just the changes and is very fast. WekaFS 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.

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 Weka cluster or onto another Weka cluster. The secondary cluster that mounts the WekaFS snap-to- object snapshot does not need to be a mirror of the primary system. In fact, the primary system could have 20 storage hosts in the cluster, while the second system could have 6 or 10, or 100. Any cluster size will work. This makes it ideal for cloud bursting. Consequently, the snap-to-object feature is useful for a range of use cases, as follows:

1. **Generic Use Cases (on-premises and cloud)**
    - Backup of data to an on-premises or cloud-based object store: If too many hardware components in a Weka 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 Weka cluster, or re-hydrate onto the original cluster.
    - 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.
    - Asynchronous mirroring of data: Combining a Weka cluster with a replicated object store in another data center will create a mirror of the data that can be mounted on a second Weka cluster.
2. **Cloud-Only Use Cases**
    - Public Cloud pause and restart: In the AWS cloud, Weka utilizes 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 on the AWS S3 object store and re-hydrated when needed again at a later time.
    - AWS protection against single availability zone failure: Utilizing the snap-to-object feature allows users in AWS to recover from an availability zone (AZ) failure. Should the first AZ fail, if the Weka snapshot was replicated to a second AZ, it can be re-hydrated in minutes by a Weka cluster in the secondary AZ.
3. **Hybrid Cloud Use Case**
    - Cloud bursting: An on-premises customer can benefit from cloud elasticity by using additional computational power for short periods. By uploading a snapshot to S3, 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.

Data Protection

Data protection is a critical function of any storage system, and challenges are amplified at scale. Without an appropriate internal data protection schema, file systems would need to be limited in size to accommodate the effects of disk or host rebuild time windows and minimize the risk of data exposure. Popular data protection schemes such as RAID[6], internal replication (copies of blocks/files), and erasure coding are a compromise between scalability, protection, capacity, and performance.

With Weka, there is no concept of data or metadata locality, as all data and metadata are distributed evenly across the storage hosts, which improves the scalability, aggregate performance and resiliency. With the advent of high-speed networks, data locality actually contributes to performance and reliability issues by creating data hot spots and system scalability issues. By directly managing data placement on the SSD layer, WekaFS can shard the data and distribute it across the storage cluster for optimal placement based on user-configurable stripe sizes. WekaFS uses advanced algorithms to determine data layout; sharded data perfectly matches the block sizes used by the underlying flash memory to improve performance and extend SSD service life. Stripe sizes can be set to any value from 4 to 16, while parity can be set to either +2 or +4. FIG. 8 illustrates data placement across SSDs in a 6+2 configuration. The minimum supported cluster size is 6, which allows for two full virtual spares for a rebuild from a 4+2 configuration. The bigger the Weka cluster, the bigger the stripe size that it can support, and the greater the storage efficiency and write performance.

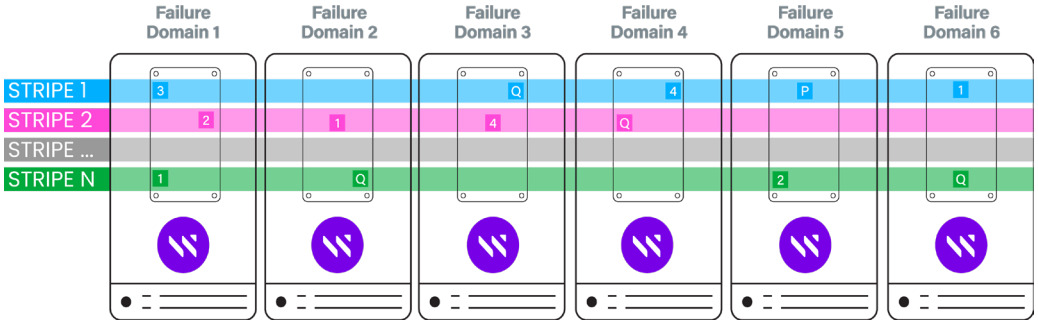


FIG. 8 WekaFS data distribution

WekaFS Data Protection Schema

WekaFS manages protection so data is always safe and accessible:

- Configurable data protection levels from 4+2 to 16+4
- Patented distributed data protection schema ▪ Configurable failure domains
- End-to-end checksums for data integrity
- Metadata journaling
- Local snapshots and clones
- Snapshot-to-object for backup and DR

6 RAID = Redundant Array of Independent Disks

Weka uses failure domains to define data protection levels. Failure domains are fully configurable starting at the server host level, which provides single or multiple SSD level granularity. Data protection levels are flexible depending on the size and scale of the server cluster— the larger the cluster, the larger the recommended data stripe size for the best utilization of SSD capacity, improved performance, and higher resiliency. For granular protection, the data protection level is set at the cluster level and parity can be set to two or four, meaning that the system can survive up to two or four simultaneous host failures without impacting data availability.

Weka's data protection scheme follows the convention of data (N) + parity (2 to 4). The N + 4 data protection level is unique from a cloud storage perspective. Many cloud based storage services use a triple replication scheme to protect data. Weka's data protection scheme delivers significantly better resiliency than triple replication – which only protects to 2 failures – without the expensive storage and throughput impact. An N + 2 protection level is sufficient for most production environments, whether with converged (application and storage sharing the same host) clusters or dedicated appliances. An N+4 protection level is recommended for clusters with a large number (hundreds) of converged cluster nodes because application failures or lockups can impact server availability.

In addition to core data protection, Weka recommends availability best practices such as hosts with redundant power supplies, multiple NICs and switches for network redundancy, etc.

Virtual (hot) Spare

A virtual (hot) spare is reserved capacity so that if a failure domain has failed, the system can undergo a complete rebuild of data, and still maintain the same net capacity. All failure domains are always participating in storing the data, and the virtual spare capacity is evenly spread within all failure domains.

The higher the virtual spare count, the more hardware that is required to obtain the same net capacity. Conversely, the higher the hot spare count, the more relaxed the IT maintenance schedule for replacements. The virtual spare is defined during the cluster formation and can be re-configured at any time. The default number of virtual spares is one.

Data Distribution

WekaIO utilizes a patented[7] distributed data protection coding scheme that increases resiliency as the number of the nodes in the cluster scale. It delivers the scalability and durability of erasure coding but without the performance penalty. Unlike legacy hardware and software RAID and other data protection schemes, Weka's rebuild time gets faster and more resilient as the system scales because every host in the cluster participates in the rebuild process.

WekaFS equally distributes data and metadata across logical nodes that span failure domains (FD). A failure domain can be a server host, a rack, or even a data center. In cloud environments, WekaFS can span availability zones (AZ).

Unlike traditional hardware and software data protection schemes, Weka only places a single segment of a given data stripe inside any one server (or FD), so in the event of multiple drive failures within a single server it will still be considered a single failure of the domain. The data distribution mechanism always stripes across failure domains. The protection level is defined at the FD level. WekaFS handles failures at the FD level, so individual or multiple failures within the FD is treated as a single failure. Data stripes are always spread across different server hosts, racks, or AZs

7 United States Patent 9448887 – http://www.freepatentsonline.com/9448887.html to learn more

depending on the resiliency chosen. Weka's resiliency is user-configurable to define the number of failures to tolerate within a Weka cluster to meet the application workload service level requirements. When a failure occurs, the system considers the FD a single failure, regardless of how large the domain is defined. In addition to distributing stripes at the FD level, Weka also ensures a highly randomized data placement for improved performance and resiliency. As the cluster size grows the probability of a hardware failure goes up proportionally, but WekaFS overcomes this issue by distributing the stripes in a randomized manner. The more nodes the higher the amount of random stripe combinations, making the probability of a double failure lower. Example: for a stripe size of 18 (16+2) and a cluster size of 20 the number of possible stripe combinations is 190, however as the cluster size grows to 25, the number of possible stripe combinations is now 480,700. The number of possible stripe combinations is based on the following formula where C is the number of hosts in a cluster, and S is the stripe size: C!/(S!*(C-S)!).

WekaFS Rebuilds

Weka uses several innovative strategies to return the system to a fully protected state as quickly as possible and be ready to handle a subsequent failure. This ensures that applications are not impacted by long data rebuild processes.

Weka protects data at the file level, so it only needs to rebuild the data that is actively stored on the failed server or SSD. This means, that the rebuild times are faster compared to a traditional RAID solution or file server that protects data at the block layer. RAID controller based systems typically rebuild all blocks on an affected storage device (SSD/HDD), including empty blocks, prolonging rebuilds and the time of exposure. Weka only needs to rebuild the specific file data that has been affected by the failure. When a tiering-to-object policy is in place with a filesystem, a further benefit is that data that has already been tiered off to the object store is never impacted by a node failure because it is protected on the object store. In addition, any cached data (data that was tiered to object but still 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.

Weka stripes are comprised of 4k blocks. 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. All the remaining FDs in the cluster will participate in the rebuilding of any missing blocks in the stripe. Examples of this include singular disk failures, host failures or entire failure domain failures. WEkaFS will rebuild data from that drive(s) or FD using a parity calculation and write that data across all remaining healthy FDs. This means that the larger the cluster size, the faster the rebuild and the more reliable the system becomes because more compute resources are available to participate in the rebuild process and the stripes become more randomized across the hosts. In the event of multiple failures, the system prioritizes data rebuilds starting with data stripes that are in the least protected state. Weka looks for data stripes, that are common to the failed hosts and rebuilds 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, in a replicated system only the mirrored servers participate in the recovery process, impacting performance significantly. Erasure coding suffers from a similar problem, where only a small subset of the servers participates in the recovery. With WekaFS, the recovery rate is user-configurable and the amount of network traffic dedicated to rebuild can be changed at any time, so administrators have complete control to determine the best tradeoff between continued application performance and time-to-recovery.

Power-Fail and End-to-End Data Protection

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

WekaFS 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 a 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 (FSCK). For most other file systems, the FSCK process recovery time is proportional to the size of the recovered file system. In large scale deployments, this recovery can take days or weeks.

Automated Data Rebalancing

WekaFS proactively monitors and manages the performance, resiliency, and capacity health status of a Weka cluster. This allows the system to calculate the utilization levels (performance and capacity) of hosts to redistribute data automatically and transparently across the cluster to prevent hot spots.

The benefit is that Weka can maintain well-balanced cluster performance and data protection as capacity and usage change. Another advantage is that as additional SSDs are added to existing server hosts or the cluster is expanded with more hosts, Weka automatically rebalances to enhance performance, resiliency, and capacity without requiring costly downtime for data migration. Matched capacity SSDs are not required, which allows you to leverage new technology and save money as SSD prices decline.

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 Weka 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.

Multi-Tenant Organizations

WekaFS supports the construct of organizations, an element of multi-tenancy that offers hierarchical access controls. Access can be separated such that data is managed at the organizational level and only visible to that organizational group's members. WekaFS can support up to 64 organizations, and within an organization, logical entities participating in obtaining control of that data are managed by the Organization Administrator, not the Cluster Administrator. The Cluster Admin can create organizations, define the Organizational Admin, delete organizations, and monitor capacity usage by the organization file system.

Capacity Quotas

As noted, there are many ways that WekaFS manages capacity utilization across organization.

- Organizational level quotas allow groups to manage their own file systems and capacity. WekaFS supports up to 64 organizations.
- File system level capacity allows different projects or departments to have their own allocated capacity. The Weka file system supports up to 1024 file systems on a single storage namespace.
- Directory level quotas provide a quota per project directory, useful when there are many projects withing a single file system. Quotas can be set at an advisory level, as hard quotas or as soft quotas.

Quality of Service

The Weka client also has additional performance management capabilities in the form of QoS functionality. This is a limiting function where you can set both a preferred throughput, and a maximum throughput. 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 WekaFS. When combined with quotas and organizational controls, this allows fine-grained resource management within the Weka system.

Authentication and Access Control

WekaFS provides authentication services at the user level and the client-server level to validate that the user or client has the ability to view and access data. WekaFS 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 are able to 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. WekaFS 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.
- WekaFS also offers 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 Weka 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 Weka 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. 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 Weka 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 filesystems:

- 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.

Encryption In-Flight and At-Rest

WekaFS provides full end-to-end encryption from the client all the way to the object storage solution, making it the most robust encrypted file system commercially available. Encryption is set at the file system level 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 a object store tier, the data will remain encrypted. Weka's 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 negligible impact on application performance when using the Weka client.

Key Rotation and Key Management

WekaFS supports any key management system (KMS) that is compliant with KMIP (the Key Management Interoperability Protocol) 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 WekaFS pushes a snapshot to an object store, the data is fully protected and can be authenticated only through the KMS system.

Auto Scaling Groups

For cloud-native applications auto scaling delivers the full flexibility of cloud elasticity by allowing resources to dynamically grow or shrink based on performance or user demand requirements. WekaFS now supports EC2 auto scaling groups in AWS to allow auto scaling up of the cluster for peak demand periods and auto scaling down when not needed. In addition to this, thin provisioning of filesystems along with auto scaling groups allows for available capacity to be automatically increased in the filesystems when the cluster is expanded, and automatically shrunk as needed when the cluster is scaled down.

Flexible Deployment Options (On-Premises and Cloud)

Whether your applications run on bare metal for performance, in a virtual or containerized environment for ease of deployment and resiliency, or entirely in the public cloud for on-demand scalability and elasticity, Weka is a single, no compromise, storage solution providing the freedom to choose the environment best suited for your application based on performance, scale, and economics. Weka clients can run in bare metal, virtualized, containerized, and cloud environments (FIG. 9).

Weka provides industry-leading flexibility by supporting an extremely broad array of operating environments and deployment models. It runs on standard x86-based servers using Ethernet or InfiniBand network adapters and off-the-shelf NVMe SSDs. Weka Storage hosts can run on bare metal as well as in approved cloud environments.

Starting at only six servers, Weka can scale to many thousands of servers. As infrastructure matures, Weka enables expanding the cluster with new servers, then retiring older generation servers, thus enabling lifecycle management of hardware refresh without the need to perform complete lift-and-shift data migration to different storage.

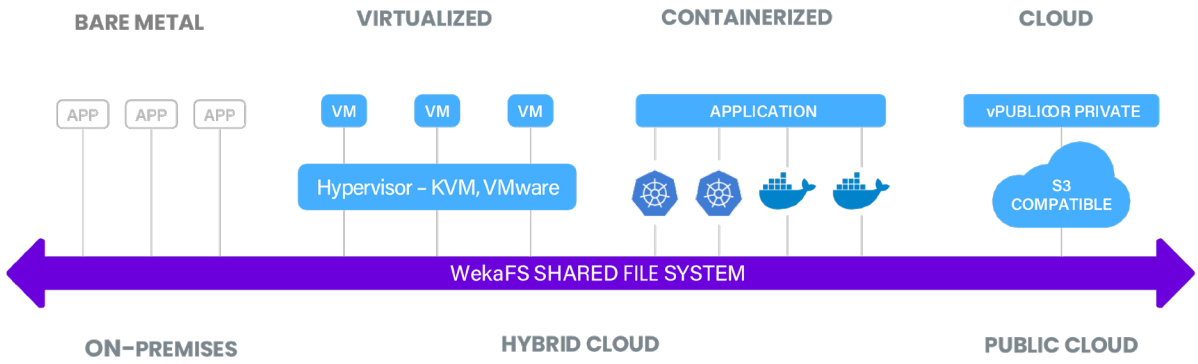


FIG. 9 Industry-Leading Flexibility to Support Your Operating Environment

Converged

In a converged deployment (FIG. 10), Weka is integrated into standard AMD or Intel x86-based application servers, or public cloud instances. Combining storage and compute resources with applications and data into a single building block delivers a highly optimized data center solution across a mix of workloads. This deployment model 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. WekaFS carves out storage resources and only utilizes the resources in its container, with the remaining resources available to the applications. This deployment model is popular for clusters of GPU servers which have local NVMe devices.

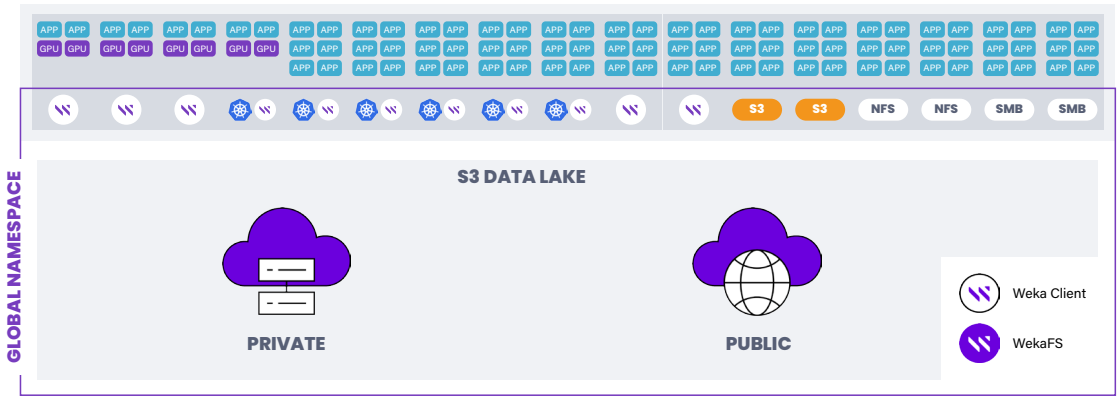


FIG. 10 WekaFS in Converged Mode with Compute and Storage on the Same Infrastructure

Dedicated Storage Server

Weka can be configured as a dedicated storage server (FIG. 11) when all the resources of the system are dedicated to storage services while applications run on a separate compute infrastructure. This mode is the most popular among customers as it ensures that application disruptions do not impact storage services.

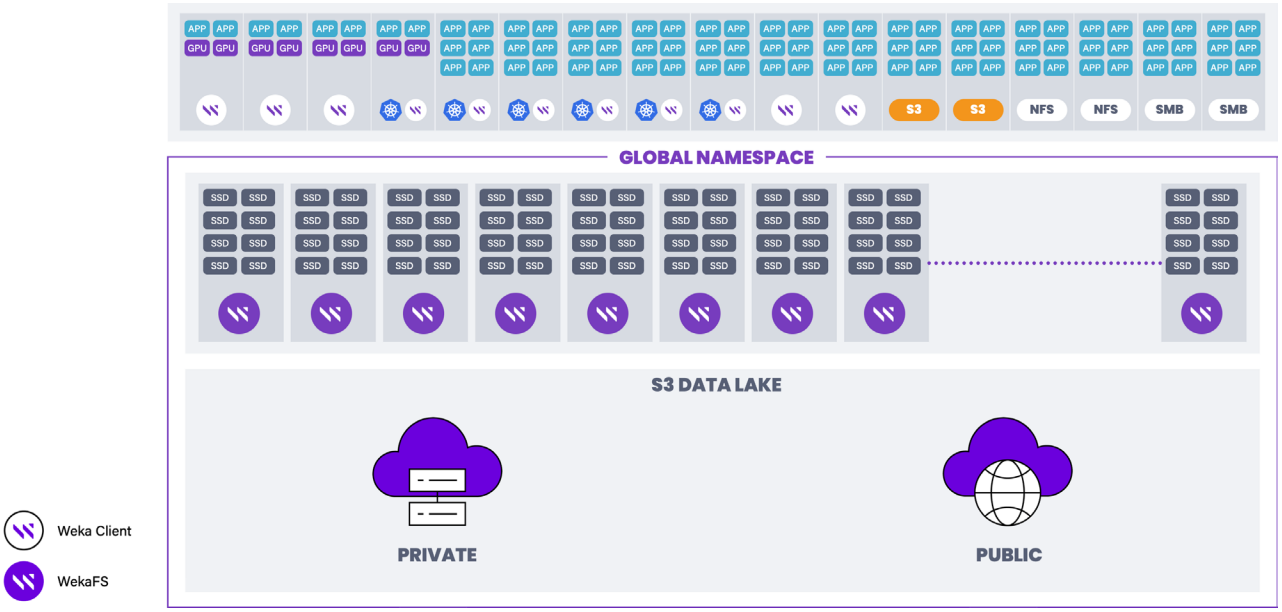


FIG. 11 Dedicated Storage Server with Separate Compute and Storage Infrastructure

Cloud

You can deploy Weka on Amazon Web Services (AWS) just as you would on-premises using EC2 instances with locally attached SSDs. Weka supports both converged and dedicated storage deployment models in the cloud on a selection of AWS EC2 instance types. For compute-intensive workloads, or to take advantage of GPU-based instances, it is best to use the dedicated storage server mode. If you need additional low-cost, high-capacity storage in AWS, Weka automated tiering to Amazon S3 is supported. If you want to burst or migrate data to the cloud, you can leverage Weka snapshot-to-object functionality for data movement. For improved resiliency, Weka also supports the spanning of AZs in the public cloud. FIG. 12 shows a typical deployment of WekaFS in the AWS public cloud.

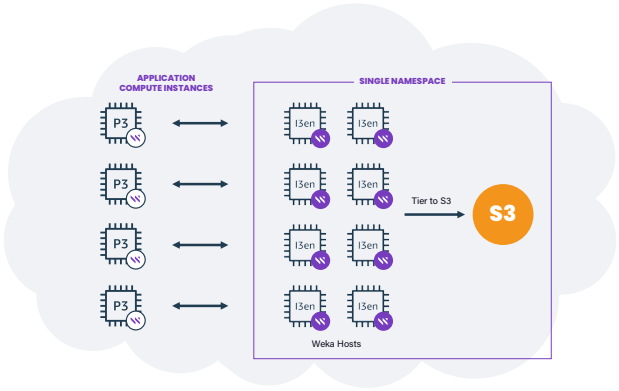


FIG. 12 WekaFS in the AWS Public Cloud

Performance in AWS is coupled to the number of NVMe devices in a Weka cluster as well as network speeds. For best performance Weka recommends using i3en instances as they have good options for fast networking and NVMe capacities.

WekaFS Performance Proof Points

Until now, IT organizations could only achieve their performance goals for file-based applications by running them on a local file system— also known as Direct Attach Storage (DAS) — or provisioning them over an All-Flash Array (AFA) volume. If shared access to the data set is required by multiple clients, performance is severely compromised because of the performance limitations of Linux NFS or Windows SMB. Weka delivers important performance benefits over this approach. Our shared POSIX-compliant file system leverages an innovative protocol to deliver file-based semantics that have all the local caching advantages of direct-attach storage and the shareability advantages of NFS or SMB. As a result, Weka allows you to run any workload on a networked shared file system with as much or more performance than DAS can offer. This, coupled with extremely low latency 4K I/O operations to the application, make the need to use complex AFA volumes for a local file system obsolete. The following chart (FIG. 13) outlines the actual performance of WekaFS in a production environment with 100Gbit networking, versus All-flash NAS and direct-attached local storage. WekaFS was 3x faster than local storage and 10x faster than NFS-based all-flash NAS. Doubling the network bandwidth to 200Gbit would result in 2x the client throughput, delivering 6x improvement in local file system performance and 20x the performance of NFS.

The Weka performance has been documented publicly on several industry-validated benchmarks. WekaFS set the record for the highest performing file system at Supercomputing 2019, winning both the number 1 overall position as well as the number 1 best metadata performance. WekaFS has set an industry record on SPEC 2014, and SPEC Storage 2020 across every major benchmark in the suite and WekaFS has set multiple records on the STAC M3 benchmark for financial analytics. Further proof points are outlined at the end of this document.

This document has outlined the architectural capabilities of WekaFS and the following section provides proof points to demonstrate the performance, scaling, and latency delivered with WekaFS.

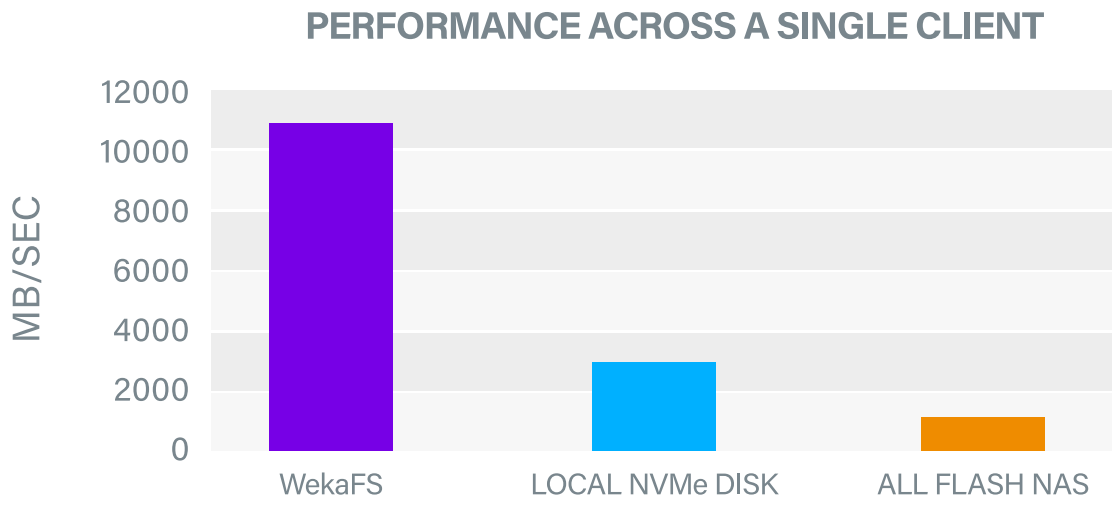


FIG. 13 WekaFS Performance vs. Local NVMe and All-Flash NAS

Standard Performance Evaluation Corporation: SPEC.org

The SPEC SFS® 2014 benchmark is a version of the Standard Performance Evaluation Corporation benchmark suite measuring file server throughput and response time, providing a standardized method for comparing performance across different vendor platforms.

WekaFS holds the leadership position for databases, Virtual Desktop Infrastructure (VDI), Electronic Design Automation (EDA), and Video Data Acquisition (VDA), with a close second for software builds.

The following figure (FIG. 14) demonstrates Weka's performance for databases. WekaFS supported 4480 databases, 2x more than the next closest submission, at 340-microsecond overall response time. https://www.spec.org/sfs2014/results/sfs2014database.html

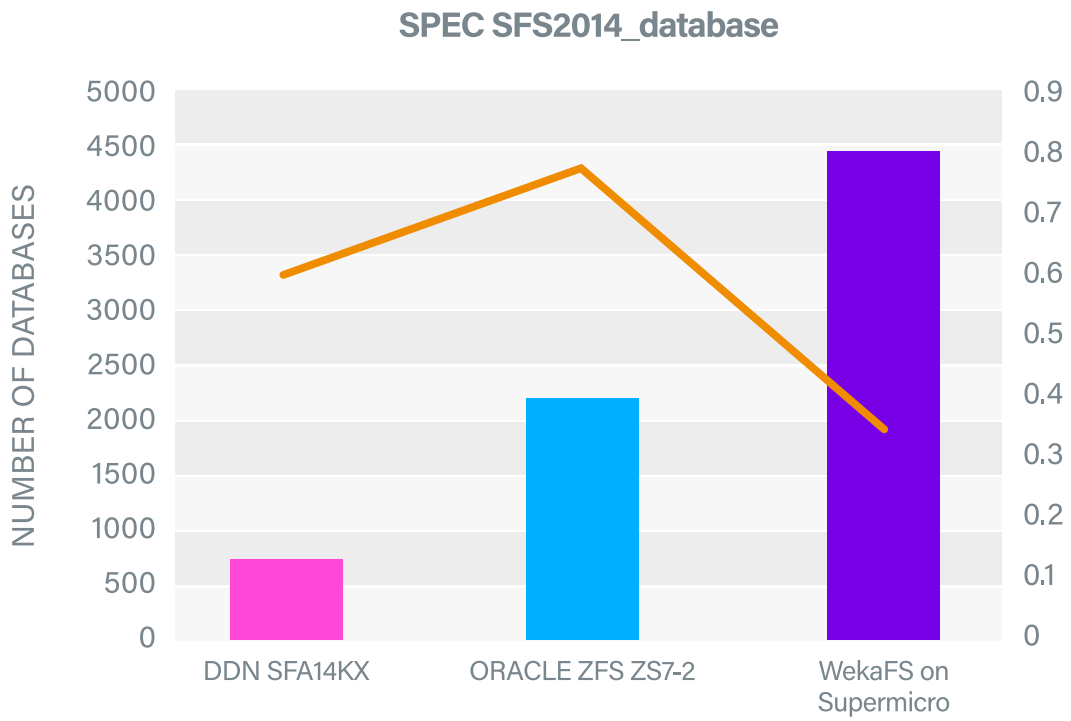


FIG. 14 WekaFS Performance for Databases

The next figure (FIG. 15) demonstrates the ultra-low latency of WekaFS. While the NetApp AFF A800 supported 8% more builds than Weka, the latency for WekaFS was 260 microseconds compared to 830 microseconds for NetApp. In other words, WekaFS can complete over 3x as many software builds as NetApp in the same time. https://www.spec.org/sfs2014/results/sfs2014swbuild.html

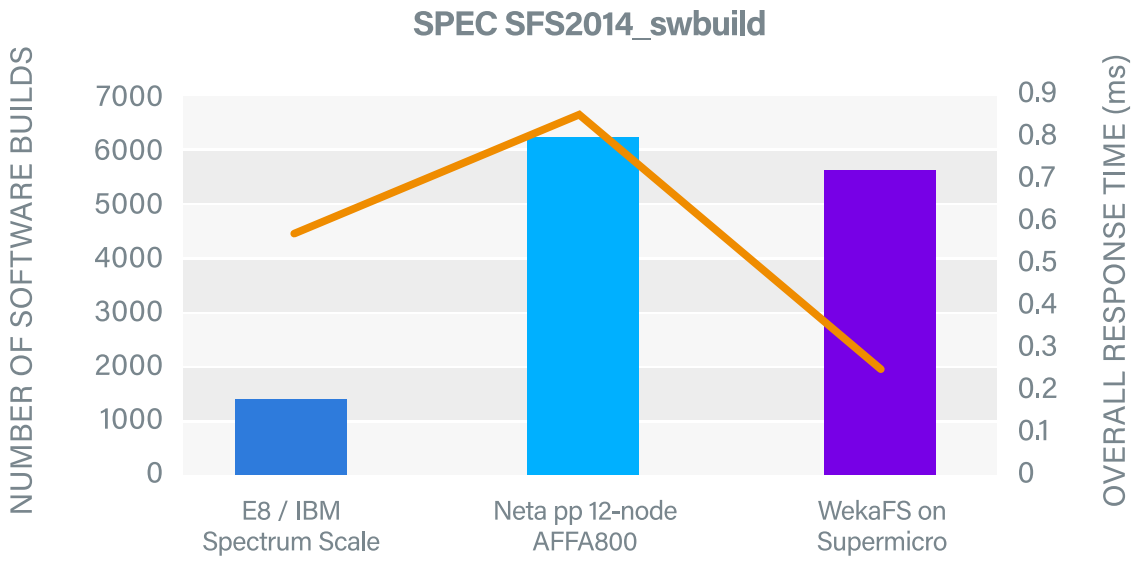


FIG. 15 WekaFS supports 5700 builds at 260-microsecond latency

Performance Scaling

To demonstrate the linear performance scaling of WekaFS, we executed testing on Amazon Web Services to show how WekaFS scales from a minimum configuration of 6 nodes to hundreds of storage nodes. The following graph (FIG. 16 on the next page) demonstrates the perfect linear scaling measured on WekaFS. The performance was measured over 10GB network links so it is not representative of the performance that WekaFS can deliver on 100Gbit or 200Gbit networks. Rather, these numbers should be looked at for the linear scalability of the file system.

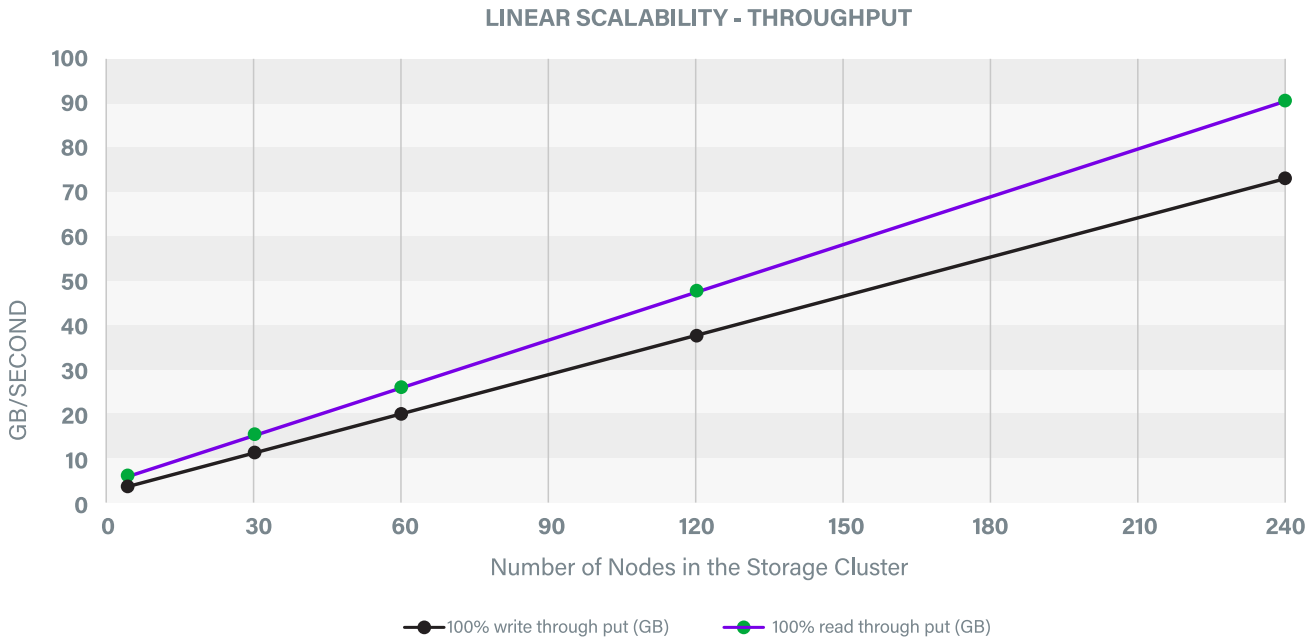


FIG. 16 Perfect Linear Performance Scaling as a Storage Cluster Grows

To understand the performance capability of WekaFS in the AWS public cloud, the following graphs (FIG. 17) demonstrate the 4K IOPs and bandwidth performance of WekaFS on i3 instances with 100Gbit networking. WekaFS achieved over 100GB/sec and 5 million IOPs at 250-microsecond latency with only 16 storage nodes. The flat latency graph shows that there is no penalty for scaling and doubling the storage server resources results in twice the I/O performance.

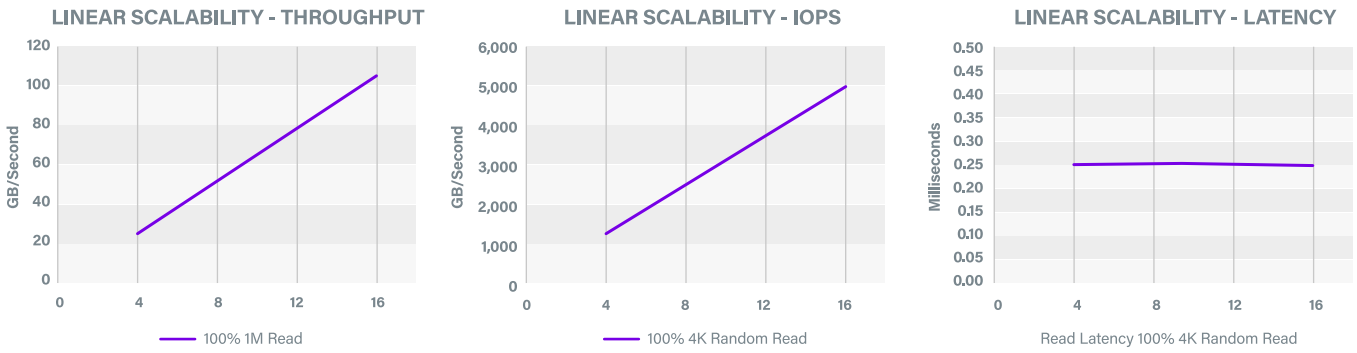


FIG. 17 WekaFS Performance on AWS

Performance to GPU Storage

WekaFS is an ideal file system for GPU-intensive workloads. Weka developed a reference architecture for the NVIDIA® DGX-1™ GPU System. FIO testing provides a baseline measure of the I/O capability of the reference architecture. The performance test was conducted with a single DGX-1 system to establish the performance that WekaFS could deliver with the minimum hardware configuration on a single 100-Gbit InfiniBand link to the host. FIG. 18 shows that WekaFS is capable of fully saturating a 100-Gbit link, delivering a peak read performance of 10.8 GBytes/second to a single DGX-1 system. The IOPs performance measurement shows that WekaFS delivered over 250,000 IOPs to a single DGX-1 system on one 100-Gbit network link.

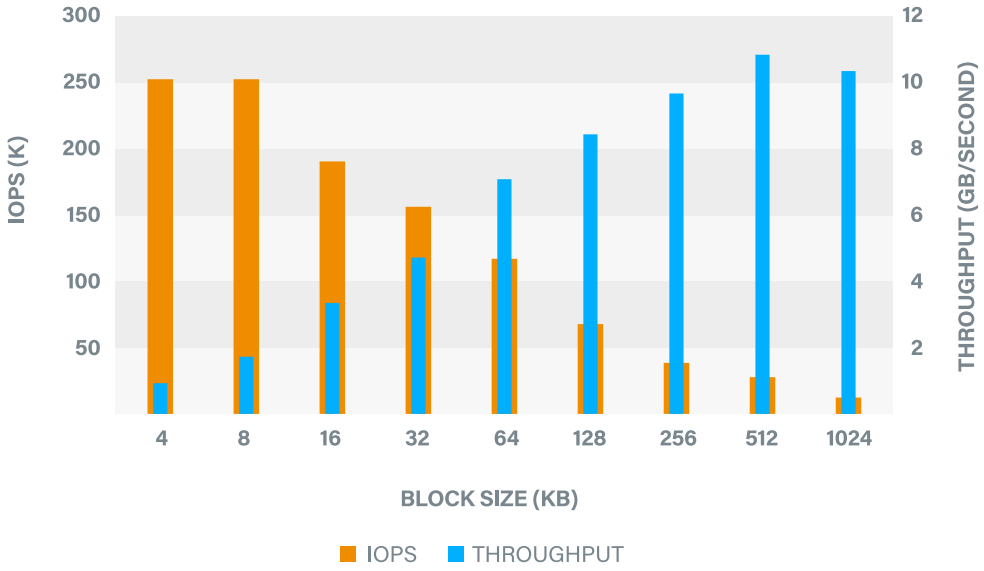


FIG. 18 Performance to a Single NVIDIA DGX-1 GPU System over a 100Gbit Network Link

The next set of performance testing measures Weka's ability to scale from a single GPU server to multiple GPU servers. The following figure (FIG. 19) demonstrates how WekaFS maintained perfect linear scaling from 1 NVIDIA DGX-1 to 9 NVIDIA DGX-1 systems. Each NVIDIA DGX-1 had 8 GPUs, for a total scaling to 72 Tesla V100 GPUs.

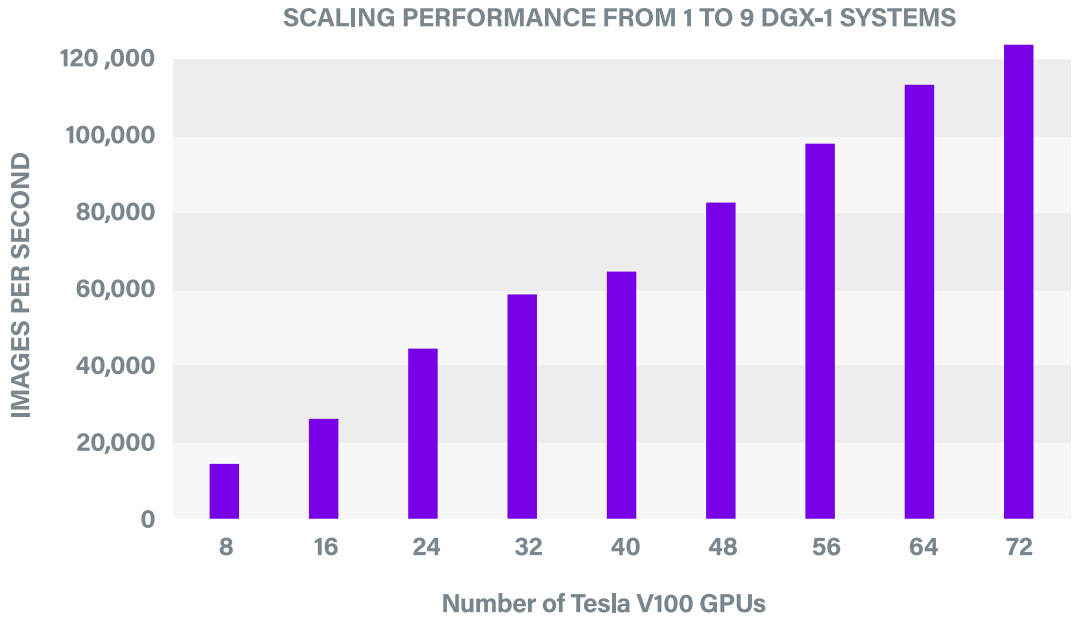


FIG. 19 Performance Scaling from a Single GPU System to 9 GPU Systems

The final set of performance tests (FIG. 20) measures Weka's ability to deliver performance to a single GPU server utilizing GPUDirect Storage. WekaFS scaled performance inside a single NVIDIA DGX-2™ GPU server from one 100Gbit EDR link to 8 EDR links with perfect linear scaling, saturating the network bandwidth.

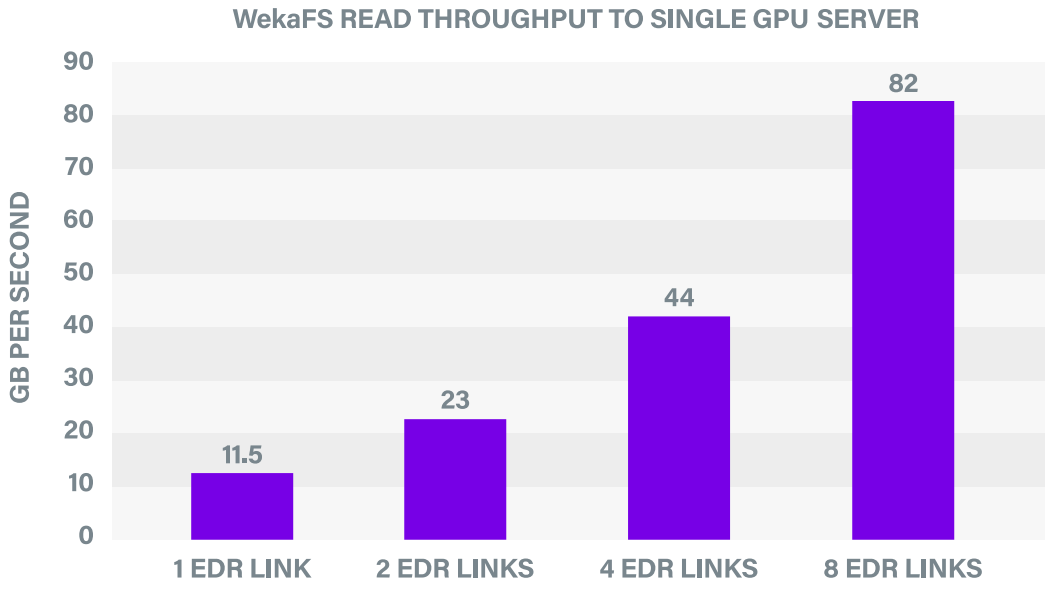


FIG. 20 Performance Scaling Inside a Single NVIDIA DGX-2 GPU Server with GPUDirect Storage

Summary

Weka addresses common IT storage problems by providing a fast, efficient, and resilient distributed parallel file system that is cloud- native and delivers the performance of All-Flash Arrays, the simplicity of file storage, and the scalability of the cloud. Part of Weka's ease of use and cloud-like experience includes rapid provisioning to reduce time to get new workloads deployed, along with elasticity scaling, resiliency, performance, and cost-effectiveness.

WekaFS is a POSIX-compliant high-performance clustered, parallel file system that has been built from the ground up to run natively on NVMe based storage. It leverages high-performance networking – either Ethernet or InfiniBand – to fully saturate the network links for maximum performance. It is an ideal solution for performance-intensive applications that demand high I/O and high concurrency to multiple clients. It is in widespread use across areas such as Life Sciences, Financial Analytics, GPU-based ML,DL and AI applications, EDA, and HPC applications. It excels in large bandwidth and small I/O-intensive applications that have relied on parallel file systems for best performance. Weka reduces the cost and complexity of storage, requiring fewer hardware resources compared to traditional solutions. It also fully supports legacy protocols such as NFS and SMB and has a rich set of enterprise-class features.

WKA314-03 03/2022