

DELL EMC POWERSCALE ONEFS: A TECHNICAL OVERVIEW

Abstract

This white paper provides technical details on the key features and capabilities of the OneFS operating system that is used to power all Dell EMC PowerScale scale-out NAS storage solutions.

April 2021

Revisions

Version	Date	Comment
1.0	November 2013	Initial release for OneFS 7.1
2.0	June 2014	Updated for OneFS 7.1.1
3.0	November 2014	Updated for OneFS 7.2
4.0	June 2015	Updated for OneFS 7.2.1
5.0	November 2015	Updated for OneFS 8.0
6.0	September 2016	Updated for OneFS 8.0.1
7.0	April 2017	Updated for OneFS 8.1
8.0	November 2017	Updated for OneFS 8.1.1
9.0	February 2019	Updated for OneFS 8.1.3
10.0	April 2019	Updated for OneFS 8.2
11.0	August 2019	Updated for OneFS 8.2.1
12.0	December 2019	Updated for OneFS 8.2.2
13.0	June 2020	Updated for OneFS 9.0
14.0	September 2020	Updated for OneFS 9.1
15.0	April 2021	Updated for OneFS 9.2

Acknowledgements

This paper was produced by the following:

Author: Nick Trimbee

The information in this publication is provided “as is.” Dell Inc. makes no representations or warranties of any kind with respect to the information in this publication, and specifically disclaims implied warranties of merchantability or fitness for a particular purpose.

Use, copying, and distribution of any software described in this publication requires an applicable software license.

Copyright © Dell Inc. or its subsidiaries. All Rights Reserved. Dell, EMC, Dell EMC and other trademarks are trademarks of Dell Inc. or its subsidiaries. Other trademarks may be trademarks of their respective owners.

TABLE OF CONTENTS

- Introduction..... 4
- OneFS overview..... 4
 - Platform nodes 5
 - Network 6
 - OneFS software overview 7
 - File system structure 10
 - Data layout 11
 - File writes 12
 - OneFS caching..... 15
- OneFS cache coherency..... 15
 - Level 1 cache 15
 - Level 2 cache 16
 - Level 3 cache 17
 - File reads..... 20
 - Locks and concurrency 21
 - Multi-threaded IO 21
 - Data protection 22
 - Node compatibility 30
 - Supported protocols 31
 - Non-disruptive operations - protocol support..... 32
 - File filtering 32
 - Data deduplication - SmartDedupe..... 32
 - Small file storage efficiency..... 33
 - In-line Data Reduction..... 34
 - Interfaces..... 36
 - Authentication and access control 37
 - Active Directory 37
 - Access zones 38
 - Roles based administration 38
 - OneFS auditing 39
 - Software upgrade 40
- OneFS data protection and management software 41
- Conclusion..... 41
- TAKE THE NEXT STEP..... 42

Introduction

The three layers of the traditional storage model—file system, volume manager, and data protection—have evolved over time to suit the needs of small-scale storage architectures but introduce significant complexity and are not well adapted to petabyte-scale systems. The OneFS operating system replaces all of these, providing a unifying clustered file system with built-in scalable data protection, and obviating the need for volume management. OneFS is a fundamental building block for scale-out infrastructures, allowing for massive scale and tremendous efficiency, and is used to power all Dell EMC PowerScale and Isilon NAS storage solutions.

Crucially, OneFS is designed to scale not just in terms of machines, but also in human terms—allowing large-scale systems to be managed with a fraction of the personnel required for traditional storage systems. OneFS eliminates complexity and incorporates self-healing and self-managing functionality that dramatically reduces the burden of storage management. OneFS also incorporates parallelism at a very deep-level of the OS, such that virtually every key system service is distributed across multiple units of hardware. This allows OneFS to scale in virtually every dimension as the infrastructure is expanded, ensuring that what works today, will continue to work as the dataset grows.

OneFS is a fully symmetric file system with no single point of failure — taking advantage of clustering not just to scale performance and capacity, but also to allow for any-to-any failover and multiple levels of redundancy that go far beyond the capabilities of RAID. The trend for disk subsystems has been slowly-increasing performance while rapidly-increasing storage densities. OneFS responds to this reality by scaling the amount of redundancy as well as the speed of failure repair. This allows OneFS to grow to multi-petabyte scale while providing greater reliability than small, traditional storage systems.

PowerScale and Isilon hardware provides the appliance on which OneFS executes. Hardware components are best-of-breed, but commodity-based — ensuring the benefits of commodity hardware’s ever-improving cost and efficiency curves. OneFS allows hardware to be incorporated or removed from the cluster at will and at any time, abstracting the data and applications away from the hardware. Data is given infinite longevity, protected from the vicissitudes of evolving hardware generations. The cost and pain of data migrations and hardware refreshes are eliminated.

OneFS is ideally suited for file-based and unstructured “Big Data” applications in enterprise environments including large-scale home directories, file shares, archives, virtualization and business analytics. As such, OneFS is widely used in many data-intensive industries today, including energy, financial services, Internet and hosting services, business intelligence, engineering, manufacturing, media & entertainment, bioinformatics, scientific research and other high-performance computing environments.

Intended Audience

This paper presents information for deploying and managing Dell EMC PowerScale and Isilon clusters and provides a comprehensive background to the OneFS architecture.

The target audience for this white paper is anyone configuring and managing a PowerScale or Isilon clustered storage environment. It is assumed that the reader has a basic understanding of storage, networking, operating systems, and data management.

 More information on OneFS commands and feature configuration is available in the [OneFS Administration Guide](#).

OneFS overview

OneFS combines the three layers of traditional storage architectures—file system, volume manager, and data protection—into one unified software layer, creating a single intelligent distributed file system that runs on a OneFS powered storage cluster.

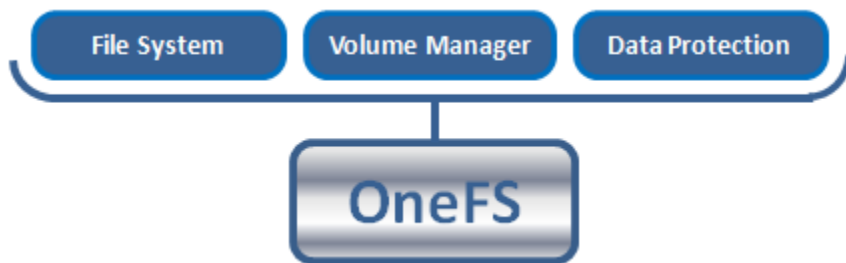


Figure 1: OneFS Combines File System, Volume Manager and Data Protection into One Single Intelligent, Distributed System.

This is the core innovation that directly enables enterprises to successfully utilize the scale-out NAS in their environments today. It adheres to the key principles of scale-out; intelligent software, commodity hardware and distributed architecture. OneFS is not only the operating system but also the underlying file system that drives and stores data in the cluster.

PowerScale and Isilon nodes

OneFS works exclusively with dedicated platform nodes, referred to as a “cluster”. A single cluster consists of multiple nodes, which are rack-mountable enterprise appliances containing: memory, CPU, networking, Ethernet or low-latency InfiniBand interconnects, disk controllers and storage media. As such, each node in the distributed cluster has compute as well as storage or capacity capabilities.

With the Isilon hardware (‘Gen6’), a single chassis of 4 nodes in a 4RU (rack units) form factor is required to create a cluster, which scales up to 252 nodes in OneFS 8.2 and later. Individual node platforms need a minimum of three nodes and 3RU of rack space to form a cluster. There are several different types of nodes, all of which can be incorporated into a single cluster, where different nodes provide varying ratios of capacity to throughput or input/output operations per second (IOPS). OneFS 9.0 also introduces support for the new 1RU all-flash F600 NVMe and F200 PowerScale nodes. Both the traditional Gen6 chassis and the PowerScale stand-alone nodes will happily co-exist within the same cluster.

Each node or chassis added to a cluster increases aggregate disk, cache, CPU, and network capacity. OneFS leverages each of the hardware building blocks, so that the whole becomes greater than the sum of the parts. The RAM is grouped together into a single coherent cache, allowing I/O on any part of the cluster to benefit from data cached anywhere. A file system journal ensures that writes that are safe across power failures. Spindles and CPU are combined to increase throughput, capacity and IOPS as the cluster grows, for access to one file or for multiple files. A cluster’s storage capacity can range from 10’s of TBs to 10’s of PBs. The maximum capacity will continue to increase as storage media and node chassis continue to get denser.

The OneFS powered platform nodes are broken into several classes, or tiers, according to their functionality:

Tier	I/O Profile	Drive Media	Node Type
Extreme Performance	High Perf, Low Latency	All-flash	F900 F800/F810 F600 F200
Performance	Transactional I/O	SAS & SSD	H600 H5600
Hybrid / Utility	Concurrency & Streaming Throughput	SATA/SAS & SSD	H500 H400
Archive	Nearline & Deep Archive	SATA	A200 A2000

Table 1: Hardware Tiers and Node Types

Network

There are two types of networks associated with a cluster: internal and external.

Back-end network

All intra-node communication in a cluster is performed across a dedicated backend network, comprising either 10, 40, or 100 Gb Ethernet, or low-latency QDR InfiniBand (IB). This back-end network, which is configured with redundant switches for high availability, acts as the backplane for the cluster. This enables each node to act as a contributor in the cluster and isolating node-to-node communication to a private, high-speed, low-latency network. This back-end network utilizes Internet Protocol (IP) for node-to-node communication.

Front-end network

Clients connect to the cluster using Ethernet connections (10GbE, 25GbE, 40GbE, or 100GbE) that are available on all nodes. Because each node provides its own Ethernet ports, the amount of network bandwidth available to the cluster scales linearly with performance and capacity. A cluster supports standard network communication protocols to a customer network, including NFS, SMB, HTTP, FTP, HDFS, and S3. Additionally, OneFS provides full integration with both IPv4 and IPv6 environments.

Complete cluster view

The complete cluster is combined with hardware, software, networks in the following view:

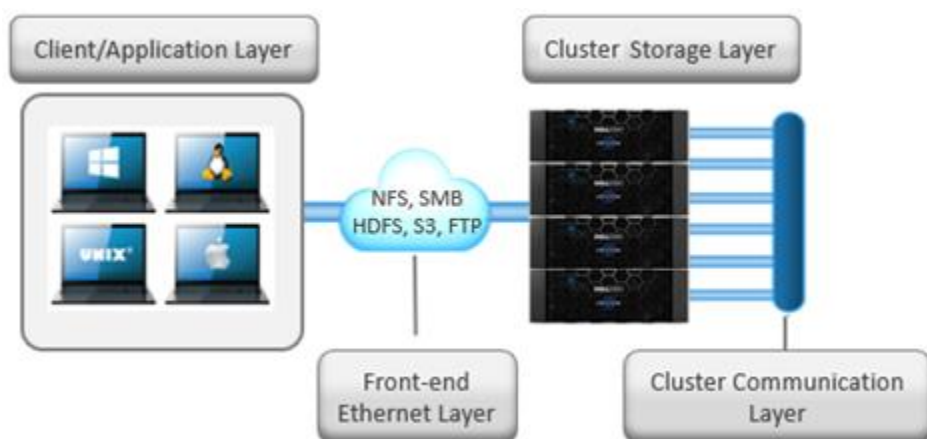


Figure 2: All Components of OneFS at Work

The diagram above depicts the complete architecture; software, hardware and network all working together in your environment with servers to provide a completely distributed single file system that can scale dynamically as workloads and capacity needs or throughput needs change in a scale-out environment.

OneFS SmartConnect is a load balancer that works at the front-end Ethernet layer to evenly distribute client connections across the cluster. SmartConnect supports dynamic NFS failover and failback for Linux and UNIX clients and SMB3 continuous availability for Windows clients. This ensures that when a node failure occurs, or preventative maintenance is performed, all in-flight reads and writes are handed off to another node in the cluster to finish its operation without any user or application interruption.

During failover, clients are evenly redistributed across all remaining nodes in the cluster, ensuring minimal performance impact. If a node is brought down for any reason, including a failure, the virtual IP addresses on that node is seamlessly migrated to another node in the cluster. When the offline node is brought back online, SmartConnect automatically rebalances the NFS and SMB3 clients across the entire cluster to ensure maximum storage and performance utilization. For periodic system maintenance and software updates, this functionality allows for per-node rolling upgrades affording full-availability throughout the duration of the maintenance window.

 Further information is available in the [OneFS SmartConnect](#) white paper.

OneFS software overview

Operating system

OneFS is built on a BSD-based UNIX Operating System (OS) foundation. It supports both Linux/UNIX and Windows semantics natively, including hard links, delete-on-close, atomic rename, ACLs, and extended attributes. It uses BSD as its base OS because it is a mature and proven Operating System and the open source community can be leveraged for innovation. From OneFS 8.0 onwards, the underlying OS version is FreeBSD 10.

Client services

The front-end protocols that the clients can use to interact with OneFS are referred to as client services. Please refer to the Supported Protocols section for a detailed list of supported protocols. In order to understand, how OneFS communicates with clients, we split the I/O subsystem into two halves: the top half or the 'initiator' and the bottom half or the 'participant'. Every node in the cluster is a participant for a particular I/O operation. The node that the client connects to is the initiator and that node acts as the 'captain' for the entire I/O operation. The read and write operation are detailed in later sections

Cluster operations

In a clustered architecture, there are cluster jobs that are responsible for taking care of the health and maintenance of the cluster itself—these jobs are all managed by the OneFS job engine. The job engine runs across the entire cluster and is responsible for dividing and conquering large storage management and protection tasks. To achieve this, it reduces a task into smaller work items and then allocates, or maps, these portions of the overall job to multiple worker threads on each node. Progress is tracked and reported on throughout job execution and a detailed report and status is presented upon completion or termination.

Job Engine includes a comprehensive check-pointing system which allows jobs to be paused and resumed, in addition to stopped and started. The Job Engine framework also includes an adaptive impact management system.

The Job Engine typically executes jobs as background tasks across the cluster, using spare or especially reserved capacity and resources. The jobs themselves can be categorized into three primary classes:

File system maintenance jobs

These jobs perform background file system maintenance, and typically require access to all nodes. These jobs are required to run in default configurations, and often in degraded cluster conditions. Examples include file system protection and drive rebuilds.

Feature support jobs

The feature support jobs perform work that facilitates some extended storage management function, and typically only run when the feature has been configured. Examples include deduplication and anti-virus scanning.

User action jobs

These jobs are run directly by the storage administrator to accomplish some data management goal. Examples include parallel tree deletes and permissions maintenance.

The table below provides a comprehensive list of the exposed Job Engine jobs, the operations they perform, and their respective file system access methods:

Job Name	Job Description	Access Method
AutoBalance	Balances free space in the cluster.	Drive + LIN

Job Name	Job Description	Access Method
AutoBalanceLin	Balances free space in the cluster.	LIN
AVScan	Virus scanning job that antivirus server(s) run.	Tree
ChangelistCreate	Create a list of changes between two consecutive SyncIQ snapshots	Changelist
CloudPoolsLin	Archives data out to a cloud provider according to a file pool policy.	LIN
CloudPoolsTreewalk	Archives data out to a cloud provider according to a file pool policy.	Tree
Collect	Reclaims disk space that could not be freed due to a node or drive being unavailable while they suffer from various failure conditions.	Drive + LIN
ComplianceStoreDelete	SmartLock Compliance mode garbage collection job.	Tree
Dedupe	Deduplicates identical blocks in the file system.	Tree
DedupeAssessment	Dry run assessment of the benefits of deduplication.	Tree
DomainMark	Associates a path and its contents with a domain.	Tree
DomainTag	Associates a path and its contents with a domain.	Tree
EsrsMftDownload	ESRS managed file transfer job for license files.	
FilePolicy	Efficient SmartPools file pool policy job.	Changelist
FlexProtect	Rebuilds and re-protects the file system to recover from a failure scenario.	Drive + LIN
FlexProtectLin	Re-protects the file system.	LIN
FSAnalyze	Gathers file system analytics data that is used in conjunction with InsightIQ.	Changelist
IndexUpdate	Creates and updates an efficient file system index for FilePolicy and FSAnalyze jobs,	Changelist
IntegrityScan	Performs online verification and correction of any file system inconsistencies.	LIN
LinCount	Scans and counts the file system logical inodes (LINs).	LIN
MediaScan	Scans drives for media-level errors.	Drive + LIN
MultiScan	Runs Collect and AutoBalance jobs concurrently.	LIN
PermissionRepair	Correct permissions of files and directories.	Tree
QuotaScan	Updates quota accounting for domains created on an existing directory path.	Tree
SetProtectPlus	Applies the default file policy. This job is disabled if SmartPools is activated on the cluster.	LIN

Job Name	Job Description	Access Method
ShadowStoreDelete	Frees space associated with a shadow store.	LIN
ShadowStoreProtect	Protect shadow stores which are referenced by a LIN with higher requested protection.	LIN
ShadowStoreRepair	Repair shadow stores.	LIN
SmartPools	Job that runs and moves data between the tiers of nodes within the same cluster. Also executes the CloudPools functionality if licensed and configured.	LIN
SmartPoolsTree	Enforce SmartPools file policies on a subtree.	Tree
SnapRevert	Reverts an entire snapshot back to head.	LIN
SnapshotDelete	Frees disk space that is associated with deleted snapshots.	LIN
TreeDelete	Deletes a path in the file system directly from the cluster itself.	Tree
Undedupe	Removes deduplication of identical blocks in the file system.	Tree
Upgrade	Upgrades cluster on a later OneFS release.	Tree
WormQueue	Scan the SmartLock LIN queue	LIN

Figure 1: OneFS Job Engine Job Descriptions

Although the file system maintenance jobs are run by default, either on a schedule or in reaction to a particular file system event, any job engine job can be managed by configuring both its priority-level (in relation to other jobs) and its impact policy.

An impact policy can consist of one or many impact intervals, which are blocks of time within a given week. Each impact interval can be configured to use a single pre-defined impact-level which specifies the amount of cluster resources to use for a particular cluster operation. Available job engine impact-levels are:

- Paused
- Low
- Medium
- High

This degree of granularity allows impact intervals and levels to be configured per job, in order to ensure smooth cluster operation. And the resulting impact policies dictate when a job runs and the resources that a job can consume.

Additionally, job engine jobs are prioritized on a scale of one to ten, with a lower value signifying a higher priority. This is similar in concept to the UNIX scheduling utility, 'nice'.

The job engine allows up to three jobs to be run simultaneously. This concurrent job execution is governed by the following criteria:

- Job Priority
- Exclusion Sets - jobs which cannot run together (i.e., FlexProtect and AutoBalance)
- Cluster health - most jobs cannot run when the cluster is in a degraded state.

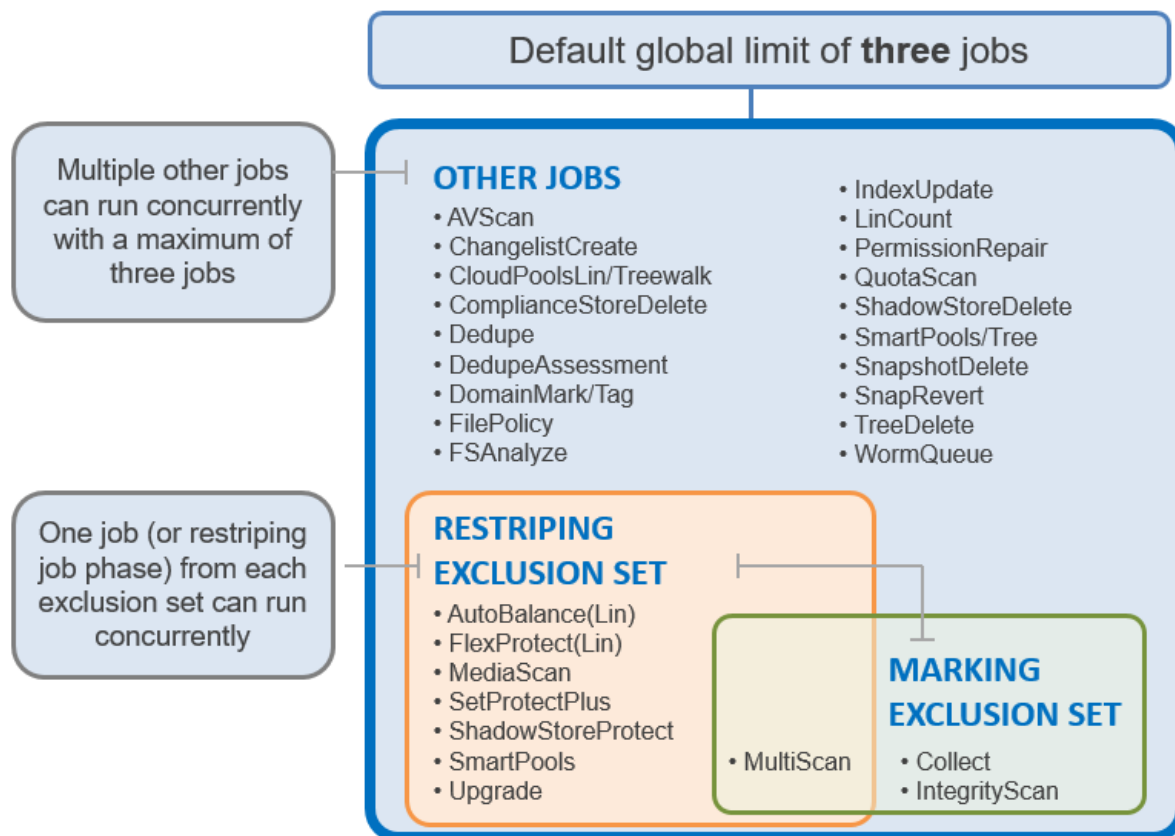


Figure 4: OneFS Job Engine Exclusion Sets

Further information is available in the [OneFS Job Engine](#) white paper.

File system structure

The OneFS file system is based on the UNIX file system (UFS) and, hence, is a very fast distributed file system. Each cluster creates a single namespace and file system. This means that the file system is distributed across all nodes in the cluster and is accessible by clients connecting to any node in the cluster. There is no partitioning, and no need for volume creation. Instead of limiting access to free space and to non-authorized files at the physical volume-level, OneFS provides for the same functionality in software via share and file permissions, and via the SmartQuotas service, which provides directory-level quota management.

Further information is available in the [OneFS SmartQuotas](#) white paper.

Because all information is shared among nodes across the internal network, data can be written to or read from any node, thus optimizing performance when multiple users are concurrently reading and writing to the same set of data.

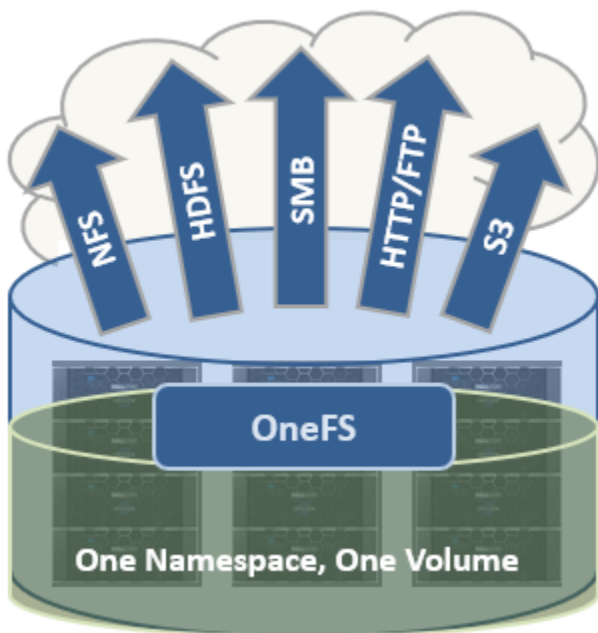


Figure 5: Single File System with Multiple Access Protocols

OneFS is truly a single file system with one namespace. Data and metadata are striped across the nodes for redundancy and availability. The storage has been completely virtualized for the users and administrator. The file tree can grow organically without requiring planning or oversight about how the tree grows or how users use it. No special thought has to be applied by the administrator about tiering files to the appropriate disk, because OneFS SmartPools will handle that automatically without disrupting the single tree. No special consideration needs to be given to how one might replicate such a large tree, because the OneFS SyncIQ service automatically parallelizes the transfer of the file tree to one or more alternate clusters, without regard to the shape or depth of the file tree.

This design should be compared with namespace aggregation, which is a commonly-used technology to make traditional NAS “appear” to have a single namespace. With namespace aggregation, files still have to be managed in separate volumes, but a simple “veneer” layer allows for individual directories in volumes to be “glued” to a “top-level” tree via symbolic links. In that model, LUNs and volumes, as well as volume limits, are still present. Files have to be manually moved from volume-to-volume in order to load-balance. The administrator has to be careful about how the tree is laid out. Tiering is far from seamless and requires significant and continual intervention. Failover requires mirroring files between volumes, driving down efficiency and ramping up purchase cost, power and cooling. Overall the administrator burden when using namespace aggregation is higher than it is for a simple traditional NAS device. This prevents such infrastructures from growing very large.

Data layout

OneFS uses physical pointers and extents for metadata and stores file and directory metadata in inodes. OneFS logical inodes (LINs) are typically 512 bytes in size, which allows them to fit into the native sectors which the majority of hard drives are formatted with. Support is also provided for 8KB inodes, in order to support the denser classes of hard drive which are now formatted with 4KB sectors.

B-trees are used extensively in the file system, allowing scalability to billions of objects and near-instant lookups of data or metadata. OneFS is a completely symmetric and highly distributed file system. Data and metadata are always redundant across multiple hardware devices. Data is protected using erasure coding across the nodes in the cluster, this creates a cluster that has high-efficiency, allowing 80% or better raw-to-usable on clusters of five nodes or more. Metadata (which makes up generally less than 1% of the system) is mirrored in the cluster for performance and availability. As OneFS is not reliant on RAID, the amount of redundancy is selectable by the administrator, at the file- or directory-level beyond the defaults of the cluster. Metadata access and locking tasks are managed by all nodes collectively and equally in a peer-to-peer architecture. This symmetry is key to the simplicity and resiliency of the architecture. There is no single metadata server, lock manager or gateway node.

Because OneFS must access blocks from several devices simultaneously, the addressing scheme used for data and metadata is indexed at the physical-level by a tuple of {node, drive, offset}. For example, if 12345 was a block address for a block that lived on disk 2 of node 3, then it would read, {3,2,12345}. All metadata within the cluster is multiply mirrored for data protection, at least to the level of redundancy of the associated file. For example, if a file were at an erasure-code protection of “+2n”, implying the file could withstand two simultaneous failures, then all metadata needed to access that file would be 3x mirrored, so it too could withstand two failures. The file system inherently allows for any structure to use any and all blocks on any nodes in the cluster.

Other storage systems send data through RAID and volume management layers, introducing inefficiencies in data layout and providing non-optimized block access. OneFS controls the placement of files directly, down to the sector-level on any drive anywhere in the cluster. This allows for optimized data placement and I/O patterns and avoids unnecessary read-modify-write operations. By laying data on disks in a file-by-file manner, OneFS is able to flexibly control the type of striping as well as the redundancy level of the storage system at the system, directory, and even file-levels. Traditional storage systems would require that an entire RAID volume be dedicated to a particular performance type and protection setting. For example, a set of disks might be arranged in a RAID 1+0 protection for a database. This makes it difficult to optimize spindle use over the entire storage estate (since idle spindles cannot be borrowed) and also leads to inflexible designs that do not adapt with the business requirement. OneFS allows for individual tuning and flexible changes at any time, fully online.

File writes

The OneFS software runs on all nodes equally - creating a single file system that runs across every node. No one node controls or “masters” the cluster; all nodes are true peers.

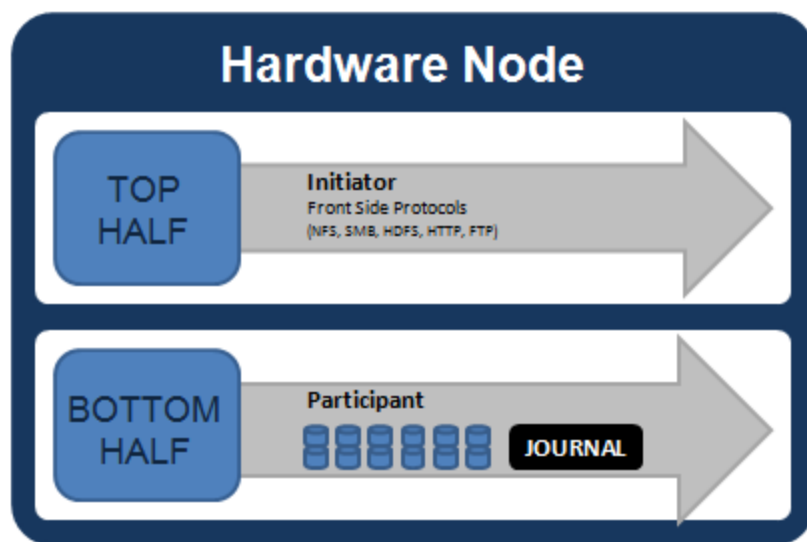


Figure 6: Model of Node Components Involved in I/O

If we were to look at all the components within every node of a cluster that are involved in I/O from a high-level, it would look like Figure 6 above. We have split the stack into a “top” layer, called the Initiator, and a “bottom” layer, called the Participant. This division is used as a “logical model” for the analysis of any one given read or write. At a physical-level, CPUs and RAM cache in the nodes are simultaneously handling Initiator and Participant tasks for I/O taking place throughout the cluster. There are caches and a distributed lock manager that are excluded from the diagram above to keep it simple. They will be covered in later sections of the paper.

When a client connects to a node to write a file, it is connecting to the top half or Initiator of that node. Files are broken into smaller logical chunks called stripes before being written to the bottom half or Participant of a node (disk). Failure-safe buffering using a write coalescer is used to ensure that writes are efficient and read-modify-write operations are avoided. The size of each file chunk is referred to as the stripe unit size.

OneFS stripes data across all nodes—and not simply across disks—and protects the files, directories and associated metadata via software erasure-code or mirroring technology. For data, OneFS can use (at the administrator's discretion) either the Reed-Solomon erasure coding system for data protection, or (less commonly) mirroring. Mirroring, when applied to user data, tends to be used more for high-transaction performance cases. The bulk of user data will generally use erasure coding, as it provides extremely high performance without sacrificing on-disk efficiency. Erasure coding can provide beyond 80% efficiency on raw disk with five nodes or more, and on large clusters can even do so while providing quadruple-level redundancy. The stripe width for any given file is the number of nodes (not drives) that a file is written across. It is determined by the number of nodes in the cluster, the size of the file, and the protection setting (for example, $+2n$).

OneFS uses advanced algorithms to determine data layout for maximum efficiency and performance. When a client connects to a node, that node's initiator acts as the "captain" for the write data layout of that file. Data, erasure code (ECC) protection, metadata and inodes are all distributed on multiple nodes within a cluster, and even across multiple drives within nodes.

Figure 7 below shows a file write happening across all nodes in a three-node cluster.

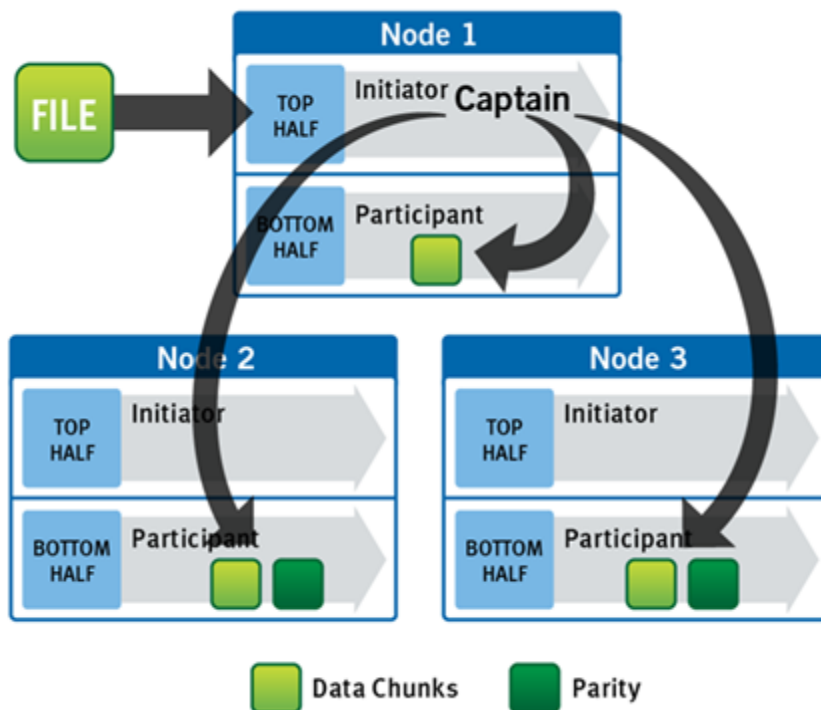


Figure 7: A File Write Operation on a 3-node Cluster

OneFS uses the back-end network to allocate and stripe data across all nodes in the cluster automatically, so no additional processing is required. As data is being written, it is being protected at the specified level. When writes take place, OneFS divides data out into atomic units called protection groups. Redundancy is built into protection groups, such that if every protection group is safe, then the entire file is safe. For files protected by erasure codes, a protection group consists of a series of data blocks as well as a set of erasure codes for those data blocks; for mirrored files, a protection group consists of all of the mirrors of a set of blocks. OneFS is capable of switching the type of protection group used in a file dynamically, as it is writing. This can allow many additional functionalities including, for example, allowing the system to continue without blocking in situations when temporary node failures in the cluster would prevent the desired number of erasure codes from being used. Mirroring can be used temporarily in these cases to allow writes to continue. When nodes are restored to the cluster, these mirrored protection groups are converted back seamlessly and automatically to erasure-code-protected, without administrator intervention.

The OneFS file system block size is 8KB. A file smaller than 8KB will use a full 8KB block. Depending on the data protection level, this 8KB file could end up using more than 8KB of data space. However, data protection settings are discussed in detail in a later section of this paper. OneFS can support file systems with billions of small files at very high performance, because all of the on-disk structures are designed to scale to such sizes and provide near-instantaneous access to any one object regardless of the total number of objects. For larger files, OneFS can take advantage of using multiple, contiguous 8KB blocks. In these cases, up to sixteen contiguous blocks can be striped onto a single node's disk. If a file is 32KB in size, then four contiguous 8KB blocks will be used.

For even larger files, OneFS can maximize sequential performance by taking advantage of a stripe unit consisting of 16 contiguous blocks, for a total of 128KB per stripe unit. During a write, data is broken into stripe units and these are spread across multiple nodes as a protection group. As data is being laid out across the cluster, erasure codes or mirrors, as required, are distributed within each protection group to ensure that files are protected at all times.

One of the key functions of the AutoBalance functionality of OneFS is to reallocate and rebalance data and make storage space more usable and efficient, when possible. In most cases, the stripe width of larger files can be increased to take advantage of new free space (as nodes are added) and to make the on-disk striping more efficient. AutoBalance maintains high on-disk efficiency and eliminates on-disk "hot spots" automatically.

The initiator top half of the "captain" node uses a modified two-phase commit transaction to safely distribute writes to multiple NVRAMs across the cluster, as shown in Figure 8 below.

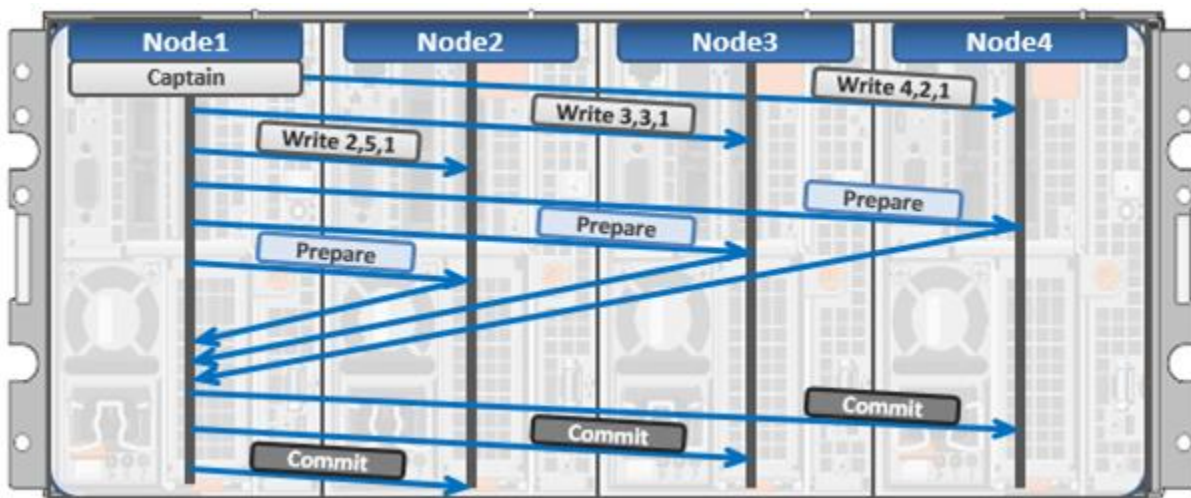


Figure 8: Distributed Transactions and Two-Phase Commit

Every node that owns blocks in a particular write is involved in a two-phase commit. The mechanism relies on NVRAM for journaling all the transactions that are occurring across every node in the storage cluster. Using multiple NVRAMs in parallel allows for high-throughput writes while maintaining data safety against all manner of failures, including power failures. In the event that a node should fail mid-transaction, the transaction is restarted instantly without that node involved. When the node returns, the only required actions are for the node to replay its journal from NVRAM—which takes seconds or minutes—and, occasionally, for AutoBalance to rebalance files that were involved in the transaction. No expensive 'fsck' or 'disk-check' processes are ever required. No drawn-out resynchronization ever needs to take place. Writes are never blocked due to a failure. This patented transaction system is one of the ways that OneFS eliminates single—and even multiple—points of failure.

In a write operation, the initiator "captains" or orchestrates the layout of data and metadata, the creation of erasure codes, and the normal operations of lock management and permissions control. An administrator from the web management or CLI interface at any point can optimize layout decisions made by OneFS to better suit the workflow. The administrator can choose from the access patterns below at a per-file or directory-level:

- **Concurrency:** Optimizes for current load on the cluster, featuring many simultaneous clients. This setting provides the best behavior for mixed workloads.
- **Streaming:** Optimizes for high-speed streaming of a single file, for example to enable very fast reading with a single client.
- **Random:** Optimizes for unpredictable access to the file, by adjusting striping and disabling the use of any prefetch cache.

OneFS also includes real-time adaptive prefetch, providing the optimal read performance for files with a recognizable access pattern, without any administrative intervention.

① The largest file size that OneFS currently supports is increased to 16TB in OneFS 8.2.2 and later, up from a maximum of 4TB in prior releases.

OneFS caching

The OneFS caching infrastructure design is predicated on aggregating the cache present on each node in a cluster into one globally accessible pool of memory. To do this, OneFS uses an efficient messaging system, similar to non-uniform memory access (NUMA). This allows all the nodes' memory cache to be available to each and every node in the cluster. Remote memory is accessed over an internal interconnect and has much lower latency than accessing hard disk drives.

For remote memory access, OneFS utilizes a redundant, under-subscribed flat Ethernet network, as, essentially, a distributed system bus. While not as fast as local memory, remote memory access is still very fast due to the low latency of 40Gb Ethernet.

The OneFS caching subsystem is coherent across the cluster. This means that if the same content exists in the private caches of multiple nodes, this cached data is consistent across all instances. OneFS utilizes the MESI Protocol to maintain cache coherency. This protocol implements an "invalidate-on-write" policy to ensure that all data is consistent across the entire shared cache.

OneFS uses up to three levels of read cache, plus an NVRAM-backed write cache, or coalescer. These, and their high-level interaction, are illustrated in the following diagram.

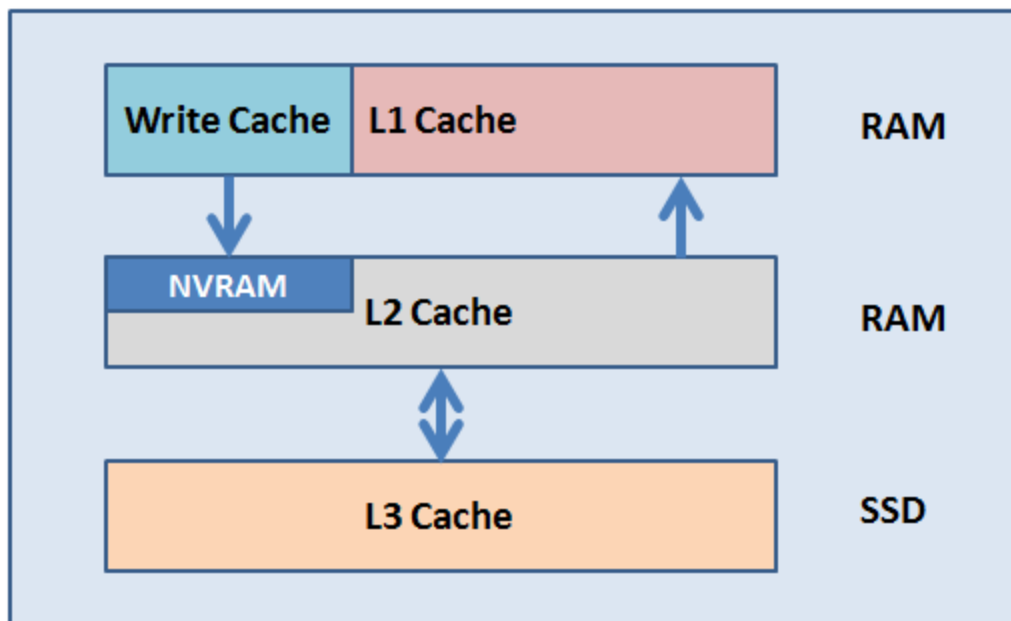


Figure 9: OneFS Caching Hierarchy

The first two types of read cache, level 1 (L1) and level 2 (L2), are memory (RAM) based, and analogous to the cache used in processors (CPUs). These two cache layers are present in all platform storage nodes.

Name	Type	Persistence	Description
L1 Cache	RAM	Volatile	Also called front-end cache, holds clean, cluster coherent copies of file system data and metadata blocks requested by clients via the front-end network
L2 Cache	RAM	Volatile	Back-end cache, containing clean copies of file system data and metadata on a local node
SmartCache / Write Coalescer	NVRAM	Non-volatile	Persistent, battery backed NVRAM journal cache which buffers any pending writes to front-end files that have not been committed to disk.
SmartFlash L3 Cache	SSD	Non-volatile	Contains file data and metadata blocks evicted from L2 cache, effectively increasing L2 cache capacity.

OneFS cache coherency

The OneFS caching subsystem is coherent across the cluster. This means that if the same content exists in the private caches of multiple nodes, this cached data is consistent across all instances. For example, consider the following initial state and sequence of events:

1. Node 1 and Node 5 each have a copy of data located at an address in shared cache.
2. Node 5, in response to a write request, invalidates node 1's copy.
3. Node 5 then updates the value. (See below).
4. Node 1 must re-read the data from shared cache to get the updated value.

OneFS utilizes the MESI Protocol to maintain cache coherency. This protocol implements an "invalidate-on-write" policy to ensure that all data is consistent across the entire shared cache. The following diagram illustrates the various states that in-cache data can take, and the transitions between them. The various states in the figure are:

- M – Modified: The data exists only in local cache and has been changed from the value in shared cache. Modified data is typically referred to as dirty.
- E – Exclusive: The data exists only in local cache but matches what is in shared cache. This data is often referred to as clean.
- S – Shared: The data in local cache may also be in other local caches in the cluster.
- I – Invalid: A lock (exclusive or shared) has been lost on the data.

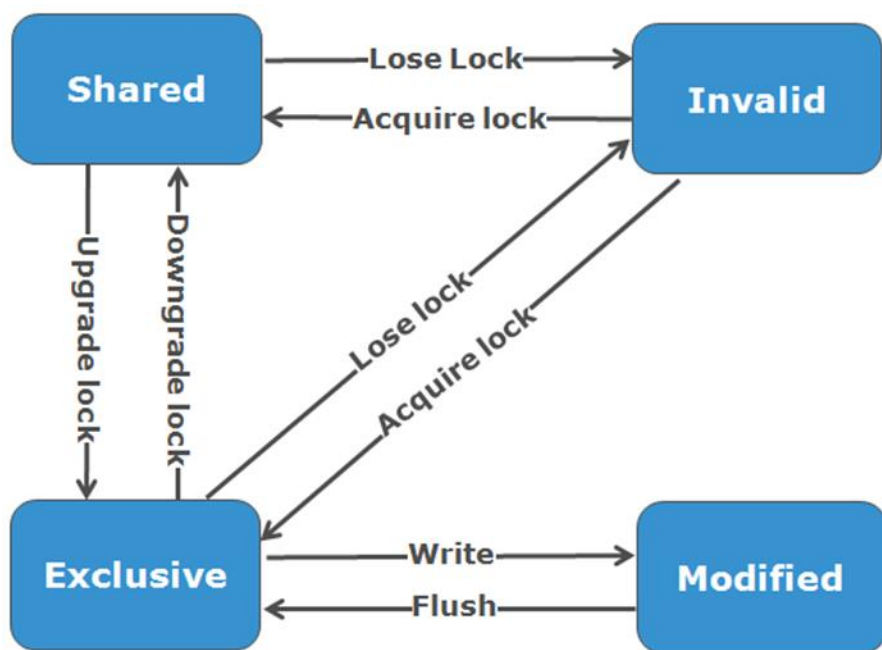


Figure 10: OneFS Cache Coherency State Diagram

Level 1 cache

The Level 1 cache (L1), or front-end cache, is memory that is nearest to the protocol layers (e.g. NFS, SMB, etc.) used by clients, or initiators, connected to that node. The primary purpose of L1 cache is to prefetch data from remote nodes. Data is prefetched per file, and this is optimized in order to reduce the latency associated with the nodes' back-end network. Since the backend interconnect latency is relatively small, the size of L1 cache, and the typical amount of data stored per request, is less than L2 cache.

L1 is also known as remote cache because it contains data retrieved from other nodes in the cluster. It is coherent across the cluster but is used only by the node on which it resides and is not accessible by other nodes. Data in L1 cache on storage nodes is aggressively discarded after it is used. L1 cache uses file-based addressing, in which data is accessed via an offset into a file object.

The L1 cache refers to memory on the same node as the initiator. It is only accessible to the local node, and typically the cache is not the master copy of the data. This is analogous to the L1 cache on a CPU core, which may be invalidated as other cores write to main memory.

L1 cache coherency is managed via a MESI-like protocol using distributed locks, as described above.

OneFS also uses a dedicated inode cache in which recently requested inodes are kept. The inode cache frequently has a large impact on performance, because clients often cache data, and many network I/O activities are primarily requests for file attributes and metadata, which can be quickly returned from the cached inode.

① L1 cache is utilized differently in cluster Accelerator nodes, which don't contain any disk drives. Instead, the entire read cache is L1 cache, since all the data is fetched from other storage nodes. Also, cache aging is based on a least recently used (LRU) eviction policy, as opposed to the drop-behind algorithm typically used in a storage node's L1 cache. Because an accelerator's L1 cache is large, and the data in it is much more likely to be requested again, so data blocks are not immediately removed from cache upon use. However, metadata & update heavy workloads don't benefit as much, and an accelerator's cache is only beneficial to clients directly connected to the node.

Level 2 cache

The Level 2 cache (L2), or back-end cache, refers to local memory on the node on which a particular block of data is stored. L2 cache is globally accessible from any node in the cluster and is used to reduce the latency of a read operation by not requiring a seek directly from the disk drives. As such, the amount of data prefetched into L2 cache for use by remote nodes is much greater than that in L1 cache.

L2 cache is also known as local cache because it contains data retrieved from disk drives located on that node and then made available for requests from remote nodes. Data in L2 cache is evicted according to a Least Recently Used (LRU) algorithm.

Data in L2 cache is addressed by the local node using an offset into a disk drive which is local to that node. Since the node knows where the data requested by the remote nodes is located on disk, this is a very fast way of retrieving data destined for remote nodes. A remote node accesses L2 cache by doing a lookup of the block address for a particular file object. As described above, there is no MESI invalidation necessary here and the cache is updated automatically during writes and kept coherent by the transaction system and NVRAM.

Level 3 cache

An optional third tier of read cache, called SmartFlash or Level 3 cache (L3), is also configurable on nodes that contain solid state drives (SSDs). SmartFlash (L3) is an eviction cache that is populated by L2 cache blocks as they are aged out from memory. There are several benefits to using SSDs for caching rather than as traditional file system storage devices. For example, when reserved for caching, the entire SSD will be used, and writes will occur in a very linear and predictable way. This provides far better utilization and also results in considerably reduced wear and increased durability over regular file system usage, particularly with random write workloads. Using SSD for cache also makes sizing SSD capacity a much more straightforward and less error prone prospect compared to using SSDs as a storage tier.

The following diagram illustrates how clients interact with the OneFS read cache infrastructure and the write coalescer. L1 cache still interacts with the L2 cache on any node it requires, and the L2 cache interacts with both the storage subsystem and L3 cache. L3 cache is stored on an SSD within the node and each node in the same node pool has L3 cache enabled.

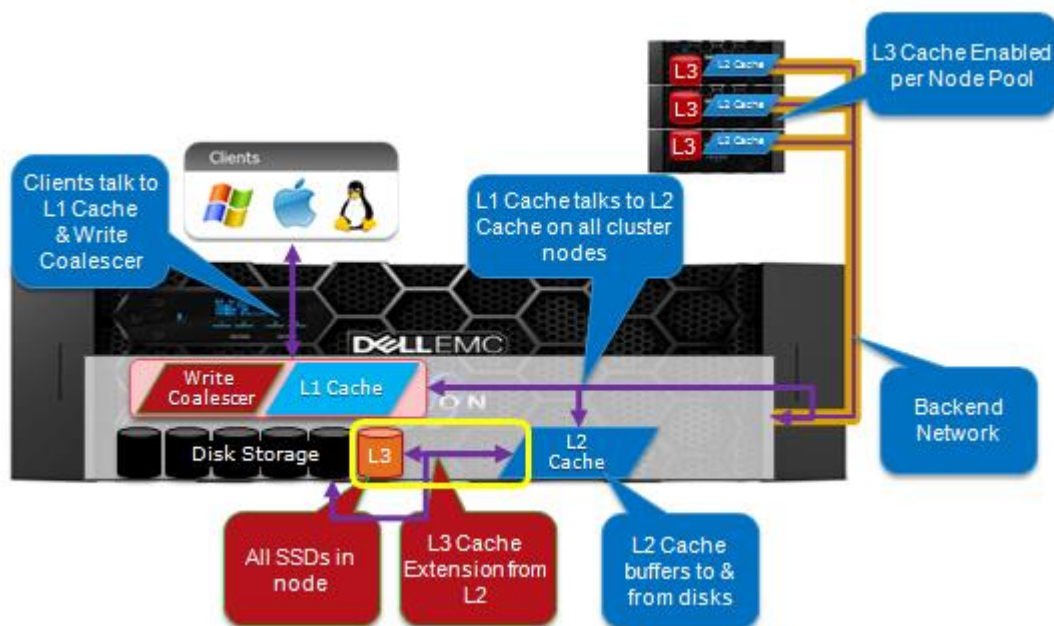


Figure 11: OneFS L1, L2 and L3 Caching Architecture

OneFS dictates that a file is written across multiple nodes in the cluster, and possibly multiple drives within a node, so all read requests involve reading remote (and possibly local) data. When a read request arrives from a client, OneFS determines whether the requested data is in local cache. Any data resident in local cache is read immediately. If data requested is not in local cache, it is read from disk. For data not on the local node, a request is made from the remote nodes on which it resides. On each of the other nodes, another cache lookup is performed. Any data in the cache is returned immediately, and any data not in the cache is retrieved from disk.

When the data has been retrieved from local and remote cache (and possibly disk), it is returned back to the client.

The high-level steps for fulfilling a read request on both a local and remote node are:

On local node (the node receiving the request):

1. Determine whether part of the requested data is in the local L1 cache. If so, return to client.
2. If not in the local cache, request data from the remote node(s).

On remote nodes:

1. Determine whether requested data is in the local L2 or L3 cache. If so, return to the requesting node.
2. If not in the local cache, read from disk and return to the requesting node.

Write caching accelerates the process of writing data to a cluster. This is achieved by batching up smaller write requests and sending them to disk in bigger chunks, removing a significant amount of disk writing latency. When clients write to the cluster, OneFS temporarily writes the data to an NVRAM-based journal cache on the initiator node, instead of immediately writing to disk. OneFS can then flush these cached writes to disk at a later, more convenient time. Additionally, these writes are also mirrored to participant nodes' NVRAM journals to satisfy the file's protection requirement. Therefore, in the event of a cluster split or unexpected node outage, uncommitted cached writes are fully protected.

The write cache operates as follows:

- An NFS client sends Node 1 a write request for a file with +2n protection.
- Node 1 accepts the writes into its NVRAM write cache (fast path) and then mirrors the writes to participant nodes' log files for protection.
- Write acknowledgements are returned to the NFS client immediately and as such, write to disk latency is avoided.
- As Node 1's write cache fills, it is periodically flushed, and writes are committed to disk via the two-phase commit process (described above) with the appropriate erasure code (ECC) protection applied (+2n).
- The write cache and participant node log files are cleared and available to accept new writes.

 Further information is available in the [OneFS SmartFlash](#) white paper.

File reads

Data, metadata and inodes are all distributed on multiple nodes within a cluster, and even across multiple drives within nodes. When reading or writing to the cluster, the node a client attaches to acts as the “captain” for the operation.

In a read operation, the “captain” node gathers all of the data from the various nodes in the cluster and presents it in a cohesive way to the requestor.

Due to the use of cost-optimized industry standard hardware, the cluster provides a high ratio of cache to disk (multiple GB per node) that is dynamically allocated for read and write operations as needed. This RAM-based cache is unified and coherent across all nodes in the cluster, allowing a client read request on one node to benefit from I/O already transacted on another node. These cached blocks can be quickly accessed from any node across the low-latency backplane, allowing for a large, efficient RAM cache, which greatly accelerates read performance.

As the cluster grows larger, the cache benefit increases. For this reason, the amount of I/O to disk on a cluster is generally substantially lower than it is on traditional platforms, allowing for reduced latencies and a better user experience.

For files marked with an access pattern of concurrent or streaming, OneFS can take advantage of pre-fetching of data based on heuristics used by the SmartRead component. SmartRead can create a data “pipeline” from L2 cache, prefetching into a local “L1” cache on the “captain” node. This greatly improves sequential-read performance across all protocols and means that reads come directly from RAM within milliseconds. For high-sequential cases, SmartRead can very aggressively prefetch ahead, allowing reads or writes of individual files at very high data rates.

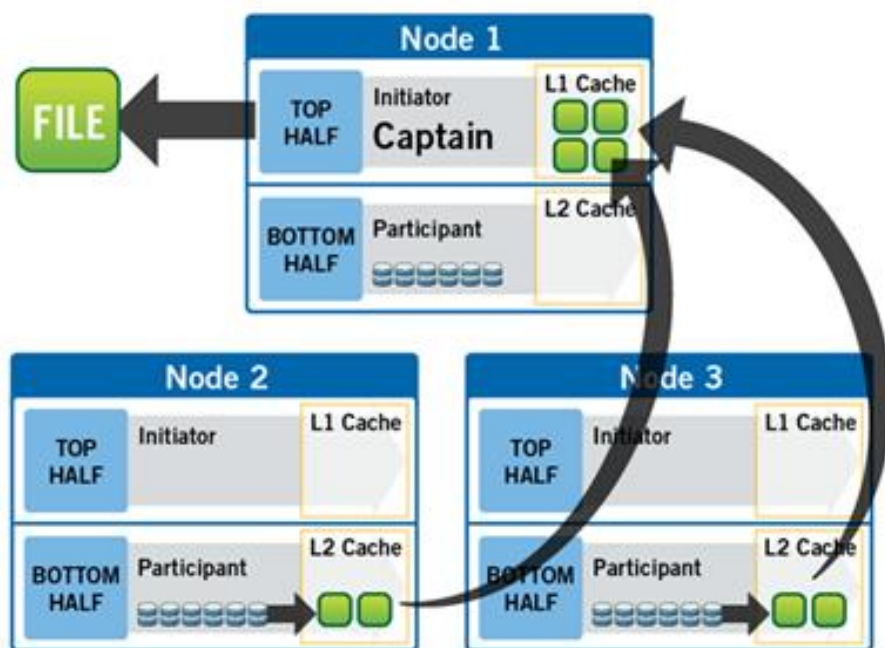


Figure 12: A File Read Operation on a 3-node Cluster

Figure 10 illustrates how SmartRead reads a sequentially-accessed, non-cached file that is requested by a client attached to Node1 in a 3-node cluster.

1. Node1 reads metadata to identify where all the blocks of file data exist.
2. Node1 also checks its L1 cache to see if it has the file data being requested.
3. Node1 builds a read pipeline, sending concurrent requests to all nodes that have a piece of file data to retrieve that file data from disk.
4. Each node pulls the blocks of file data from disk into their L2 cache (or L3 SmartFlash cache, when available), and transmits the file data to Node1.
5. Node1 records the incoming data to L1 cache, simultaneously serving the file to the client. Meanwhile, the pre-fetching process continues.
6. For highly sequential cases, data in L1 cache may be optionally “dropped behind” to free RAM for other L1 or L2 cache demands.

SmartRead’s intelligent caching allows for very high read performance with high levels of concurrent access. Importantly, it is faster for Node1 to get file data from the cache of Node2 (over the low-latency cluster interconnect) than to access its own local disk.

SmartRead’s algorithms control how aggressive the pre-fetching is (disabling pre-fetch for random-access cases) and how long data stays in the cache and optimizes where data is cached.

Locks and concurrency

OneFS has a fully distributed lock manager that marshals locks on data across all nodes in a storage cluster. The locking manager is highly extensible and allows for multiple lock “personalities” to support both file system locks as well as cluster-coherent protocol-level locks such as SMB share mode locks or NFS advisory-mode locks. OneFS also has support for delegated locks such as CIFS oplocks and NFSv4 delegations.

Every node in a cluster is a coordinator for locking resources and a coordinator is assigned to lockable resources based upon an advanced hashing algorithm. The way the algorithm is designed is that the coordinator almost always ends up on a different node than the initiator of the request. When a lock is requested for a file, it could be a shared lock (allowing multiple users to share the lock simultaneously, usually for reads) or an exclusive lock (allowing one user at any given moment, typically for writes).

Figure 13 below illustrates an example of how threads from different nodes could request a lock from the coordinator.

1. Node 2 is designated to be the coordinator of these resources.
2. Thread 1 from Node 4 and thread 2 from Node 3 request a shared lock on a file from Node 2 at the same time.
3. Node 2 checks if an exclusive lock exists for the requested file.
4. If no exclusive locks exist, Node 2 grants thread 1 from Node 4 and thread 2 from Node 3 shared locks on the requested file.
5. Node 3 and Node 4 are now performing a read on the requested file.
6. Thread 3 from Node 1 requests an exclusive lock for the same file as being read by Node 3 and Node 4.
7. Node 2 checks with Node 3 and Node 4 if the shared locks can be reclaimed.
8. Node 3 and Node 4 are still reading so Node 2 asks thread 3 from Node 1 to wait for a brief instant.
9. Thread 3 from Node 1 blocks until the exclusive lock is granted by Node 2 and then completes the write operation.

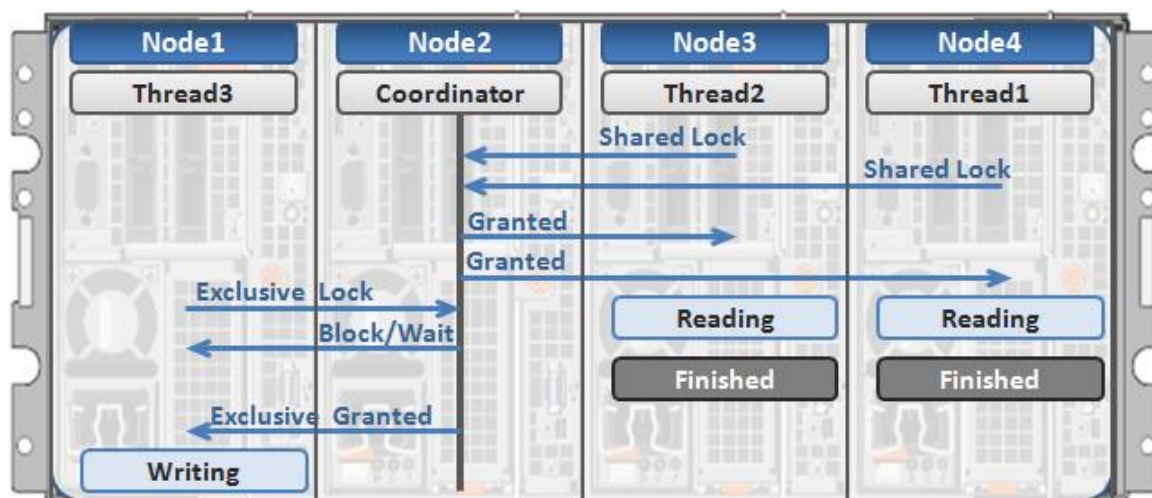


Figure 13: Distributed Lock Manager

Multi-threaded IO

With the growing use of large NFS datastores for server virtualization and enterprise application support comes the need for high throughput and low latency to large files. To accommodate this, OneFS Multi-writer supports multiple threads concurrently writing to individual files.

In the above example, concurrent write access to a large file can become limited by the exclusive locking mechanism, applied at the whole file level. In order to avoid this potential bottleneck, OneFS Multi-writer provides more granular write locking by sub-dividing the file into separate regions and granting exclusive write locks to individual regions, as opposed to the entire file. As such, multiple clients can simultaneously write to different portions of the same file.

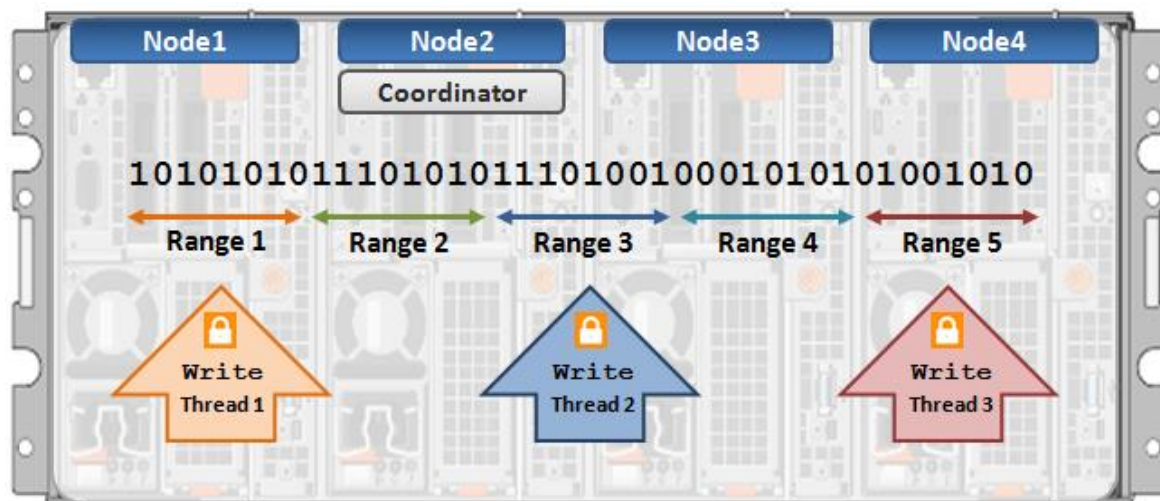


Figure 14: Multi-threaded IO Writer

Data protection

Power loss

A file system journal, which stores information about changes to the file system, is designed to enable fast, consistent recoveries after system failures or crashes, such as power loss. The file system replays the journal entries after a node or cluster recovers from a power loss or other outage. Without a journal, a file system would need to examine and review every potential change individually after a failure (an “fsck” or “chkdsk” operation); in a large file system, this operation can take a long time.

OneFS is a journaled file system in which each node contains a battery-backed NVRAM card used for protecting uncommitted writes to the file system. The NVRAM card battery charge lasts many days without requiring a recharge. When a node boots up, it checks its journal and selectively replays transactions to disk where the journaling system deems it necessary.

OneFS will mount only if it can guarantee that all transactions not already in the system have been recorded. For example, if proper shutdown procedures were not followed, and the NVRAM battery discharged, transactions might have been lost; to prevent any potential problems, the node will not mount the file system.

Hardware failures and quorum

In order for the cluster to properly function and accept data writes, a quorum of nodes must be active and responding. A quorum is defined as a simple majority: a cluster with x nodes must have $\lfloor x/2 \rfloor + 1$ nodes online in order to allow writes. For example, in a seven-node cluster, four nodes would be required for a quorum. If a node or group of nodes is up and responsive, but is not a member of a quorum, it runs in a read-only state.

OneFS uses a quorum to prevent “split-brain” conditions that can be introduced if the cluster should temporarily split into two clusters. By following the quorum rule, the architecture guarantees that regardless of how many nodes fail or come back online, if a write takes place, it can be made consistent with any previous writes that have ever taken place. The quorum also dictates the number of nodes required in order to move to a given data protection level. For an erasure-code-based protection-level of $N+M$, the cluster must contain at least $2M+1$ nodes. For example, a minimum of seven nodes is required for a +3n configuration; this allows for a simultaneous loss of three nodes while still maintaining a quorum of four nodes for the cluster to remain fully operational. If a cluster does drop below quorum, the file system will automatically be placed into a protected, read-only state, denying writes, but still allowing read access to the available data.

Hardware failures—add/remove nodes

A system called the group management protocol (GMP) enables global knowledge of the cluster state at all times and guarantees a consistent view across the entire cluster of the state of all other nodes. If one or more nodes become unreachable over the cluster interconnect, the group is “split” or removed from the cluster. All nodes resolve to a new consistent view of their cluster. (Think of this as if the cluster were splitting into two separate groups of nodes, though note that only one group can have quorum.) While in this split state, all data in the file system is reachable and, for the side maintaining quorum, modifiable. Any data stored on the “down” device is rebuilt using the redundancy stored in the cluster.

If the node becomes reachable again, a “merge” or add occurs, bringing node(s) back into the cluster. (The two groups merge back into one.) The node can rejoin the cluster without being rebuilt and reconfigured. This is unlike hardware RAID arrays, which require drives to be rebuilt. AutoBalance may restripe some files to increase efficiency, if some of their protection groups were overwritten and transformed to narrower stripes during the split.

The OneFS Job Engine also includes a process called Collect, which acts as an orphan collector. When a cluster splits during a write operation, some blocks that were allocated for the file may need to be re-allocated on the quorum side. This will “orphan” allocated blocks on the non-quorum side. When the cluster re-merges, the Collect job will locate these orphaned blocks through a parallelized mark-and-sweep scan and reclaim them as free space for the cluster.

Scalable rebuild

OneFS does not rely on hardware RAID either for data allocation, or for reconstruction of data after failures. Instead OneFS manages protection of file data directly, and when a failure occurs, it rebuilds data in a parallelized fashion. OneFS is able to determine which files are affected by a failure in constant time, by reading inode data in a linear manor, directly off disk. The set of affected files are assigned to a set of worker threads that are distributed among the cluster nodes by the job engine. The worker nodes repair the files in parallel. This implies that as cluster size increases, the time to rebuild from failures decreases. This has an enormous efficiency advantage in maintaining the resiliency of clusters as their size increases.

Virtual hot spare

Most traditional storage systems based on RAID require the provisioning of one or more “hot spare” drives to allow independent recovery of failed drives. The hot spare drive replaces the failed drive in a RAID set. If these hot spares are not themselves replaced before more failures appear, the system risks a catastrophic data loss. OneFS avoids the use of hot spare drives, and simply borrows from the available free space in the system in order to recover from failures; this technique is called virtual hot spare. In doing so, it allows the cluster to be fully self-healing, without human intervention. The administrator can create a virtual hot spare reserve, allowing the system to self-heal despite ongoing writes by users.

File level data protection with erasure coding

A cluster is designed to tolerate one or more simultaneous component failures, without preventing the cluster from serving data. To achieve this, OneFS protects files with either erasure code-based protection, via Reed-Solomon error correction ($N+M$ protection), or a mirroring system. Data protection is applied in software at the file-level, enabling the system to focus on recovering only those files that are compromised by a failure, rather than having to check and repair an entire file-set or volume. OneFS metadata and inodes are always protected by mirroring, rather than Reed-Solomon coding, and with at least the level of protection as the data they reference.

Because all data, metadata, and protection information are distributed across the nodes of the cluster, a cluster does not require a dedicated parity node or drive, or a dedicated device or set of devices to manage metadata. This ensures that no one node can become a single point of failure. All nodes share equally in the tasks to be performed, providing perfect symmetry and load-balancing in a peer-to-peer architecture.

OneFS provides several levels of configurable data protection settings, which you can modify at any time without needing to take the cluster or file system offline.

For a file protected with erasure codes, we say that each of its protection groups is protected at a level of $N+M/b$, where $N > M$ and $M \geq b$. The values N and M represent, respectively, the number of drives used for data and for erasure codes within the protection group. The value of b relates to the number of data stripes used to lay out that protection group and is covered below. A common and easily-understood case is where $b=1$, implying that a protection group incorporates: N drives worth of data; M drives worth of redundancy, stored in erasure codes; and that the protection group should be laid out over exactly one stripe across a set of nodes. This allows for M members of the protection group to fail simultaneously and still provide 100% data availability. The M erasure code members are computed from the N data members. Figure 13 below shows the case for a regular 4+2 protection group ($N=4$, $M=2$, $b=1$).

Because OneFS stripes files across nodes, this implies that files striped at $N+M$ can withstand M simultaneous node failures without loss of availability. OneFS therefore provides resiliency across any type of failure, whether it be to a drive, a node, or a component within a node (say, a card). Furthermore, a node counts as a single failure, regardless of the number or type of components that fail within it. Therefore, if five drives fail in a node, it only counts as a single failure for the purposes of $N+M$ protection.

OneFS can uniquely provide a variable level of M , up to four, providing for quadruple-failure protection. This goes far beyond the maximum level of RAID commonly in use today, which is the double-failure protection of RAID-6. Because the reliability of the storage increases geometrically with this amount of redundancy, $+4n$ protection can be orders of magnitude more reliable than traditional hardware RAID. This added protection means that large capacity SATA drives, such as 4 TB and 6 TB drives, can be added with confidence.

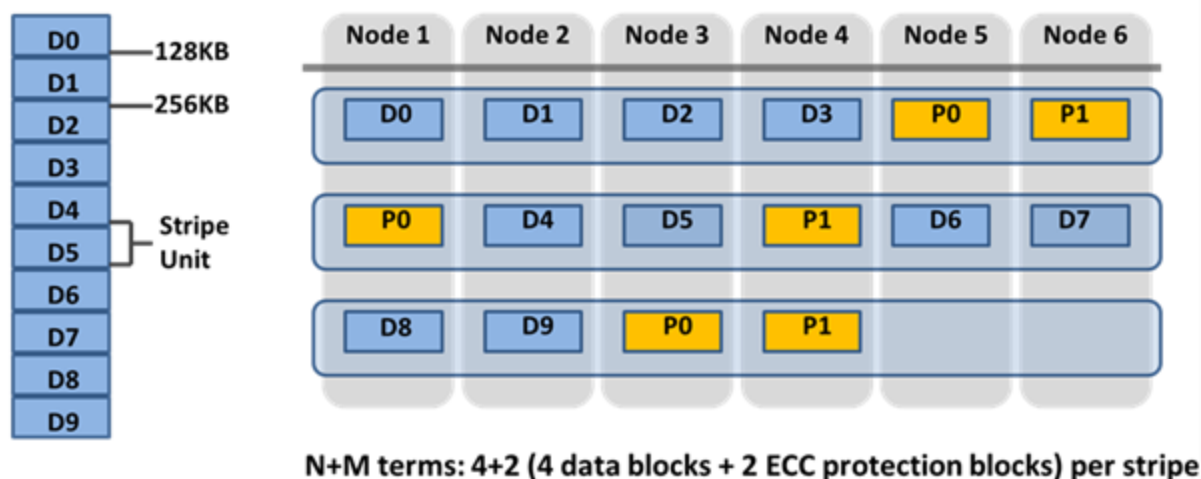


Figure 15: OneFS Redundancy – N+M Erasure Code Protection

Smaller clusters can be protected with $+1n$ protection, but this implies that while a single drive or node could be recovered, two drives in two different nodes could not. Drive failures are orders of magnitude more likely than node failures. For clusters with large drives, it is desirable to provide protection for multiple drive failures, though single-node recoverability is acceptable.

To provide for a situation where we wish to have double-disk redundancy and single-node redundancy, we can build up double or triple width protection groups of size. These double or triple width protection groups will “wrap” once or twice over the same set of nodes, as they are laid out. Since each protection group contains exactly two disks worth of redundancy, this mechanism will allow a cluster to sustain either a two or three drive failure or a full node failure, without any data unavailability.

Most important for small clusters, this method of striping is highly efficient, with an on-disk efficiency of $M/(N+M)$. For example, on a cluster of five nodes with double-failure protection, were we to use $N=3$, $M=2$, we would obtain a 3+2 protection group with an efficiency of $1-2/5$ or 60%. Using the same 5-node cluster but with each protection group laid out over 2 stripes, N would now be 8 and $M=2$, so we could obtain $1-2/(8+2)$ or 80% efficiency on disk, retaining our double-drive failure protection and sacrificing only double-node failure protection.

OneFS supports several protection schemes. These include the ubiquitous +2d:1n, which protects against two drive failures or one node failure.

① The best practice is to use the recommended protection level for a particular cluster configuration. This recommended level of protection is clearly marked as 'suggested' in the OneFS WebUI storage pools configuration pages and is typically configured by default. For all current Gen6 hardware configurations, the recommended protection level is "+2d:1n".

The hybrid protection schemes are particularly useful for Gen6 chassis high-density node configurations, where the probability of multiple drives failing far surpasses that of an entire node failure. In the unlikely event that multiple devices have simultaneously failed, such that the file is "beyond its protection level", OneFS will re-protect everything possible and report errors on the individual files affected to the cluster's logs.

OneFS also provides a variety of mirroring options ranging from 2x to 8x, allowing from two to eight mirrors of the specified content. Metadata, for example, is mirrored at one level above FEC by default. For example, if a file is protected at +2n, its associated metadata object will be 3x mirrored.

The full range of OneFS protection levels are summarized in the following table:

Protection Level	Description
+1n	Tolerate failure of 1 drive OR 1 node
+2d:1n	Tolerate failure of 2 drives OR 1 node
+2n	Tolerate failure of 2 drives OR 2 nodes
+3d:1n	Tolerate failure of 3 drives OR 1 node
+3d:1n1d	Tolerate failure of 3 drives OR 1 node AND 1 drive
+3n	Tolerate failure of 3 drives or 3 nodes
+4d:1n	Tolerate failure of 4 drives or 1 node
+4d:2n	Tolerate failure of 4 drives or 2 nodes
+4n	Tolerate failure of 4 nodes
2x to 8x	Mirrored over 2 to 8 nodes, depending on configuration

OneFS enables an administrator to modify the protection policy in real time, while clients are attached and are reading and writing data.

① Be aware that increasing a cluster's protection level may increase the amount of space consumed by the data on the cluster.

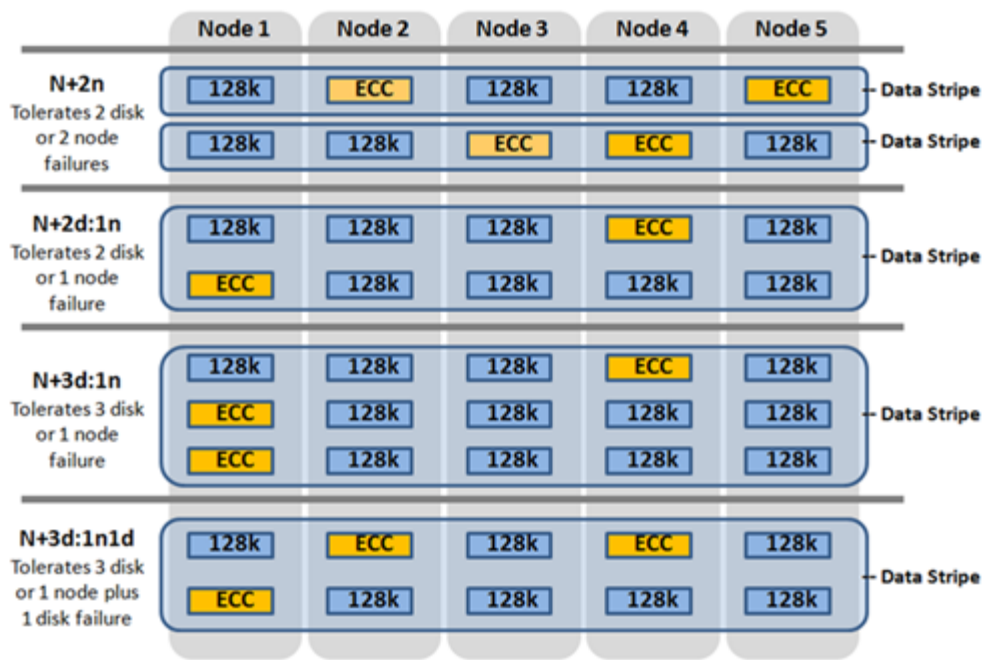


Figure 16: OneFS Hybrid Erasure Code Protection Schemes

① OneFS also provides under-protection alerting for new cluster installations. If the cluster is under-protected, the cluster event logging system (CELOG) will generate alerts, warning the administrator of the protection deficiency and recommending a change to the appropriate protection level for that particular cluster's configuration.

📖 Further information is available in the [OneFS high availability and data protection](#) white paper.

Automatic partitioning

Data tiering and management in OneFS is handled by the SmartPools framework. From a data protection and layout efficiency point of view, SmartPools facilitates the subdivision of large numbers of high-capacity, homogeneous nodes into smaller, more 'Mean Time to Data Loss' (MTTDL) friendly disk pools. For example, an 80-node H500 cluster would typically run at a +3d:1n1d protection level. However, partitioning it into four, twenty node disk pools would allow each pool to run at +2d:1n, thereby lowering the protection overhead and improving space utilization, without any net increase in management overhead.

In keeping with the goal of storage management simplicity, OneFS will automatically calculate and partition the cluster into pools of disks, or 'node pools', which are optimized for both MTTDL and efficient space utilization. This means that protection level decisions, such as the eighty-node cluster example above, are not left to the customer.

With Automatic Provisioning, every set of compatible node hardware is automatically divided into disk pools comprising up to forty nodes and six drives per node. These node pools are protected by default at +2d:1n, and multiple pools can then be combined into logical tiers and managed with SmartPools file pool policies. By subdividing a node's disks into multiple, separately protected pools, nodes are significantly more resilient to multiple disk failures than previously possible.

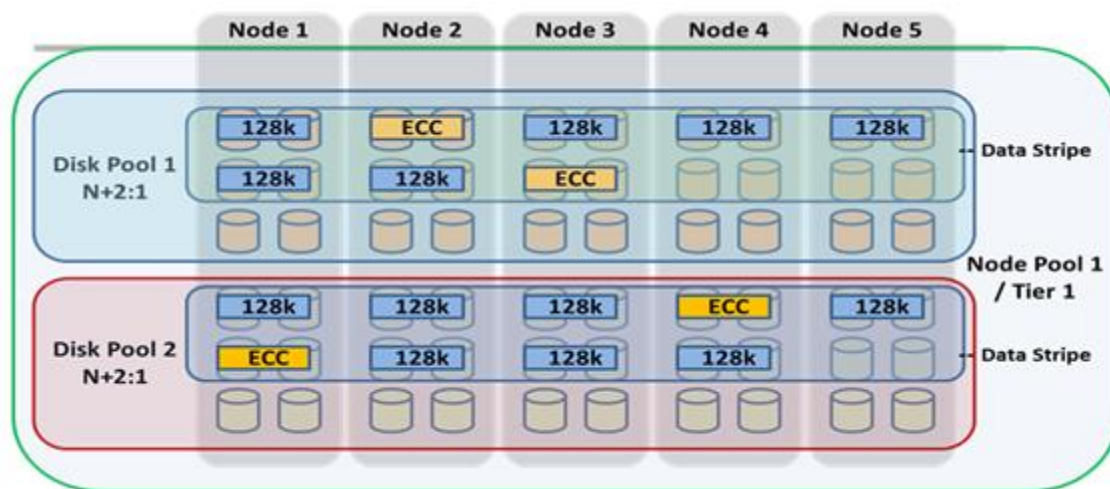


Figure 17: Automatic Partitioning with SmartPools

More information is available in the [SmartPools white paper](#).

Isilon Gen6 modular hardware platforms feature a highly dense, modular design in which four nodes are contained in a single 4RU chassis. This approach enhances the concept of disk pools, node pools, and 'neighborhoods' - which adds another level of resilience into the OneFS failure domain concept. Each Gen6 chassis contains four compute modules (one per node), and five drive containers, or sleds, per node.

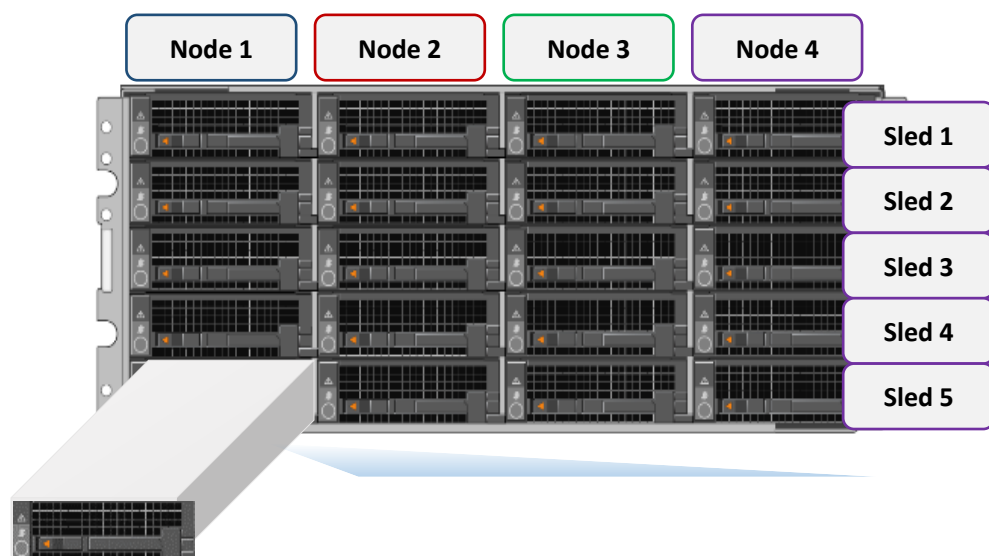


Figure 18. Gen6 Platform Chassis Front View Showing Drive Sleds

Each sled is a tray which slides into the front of the chassis and contains between three and six drives, depending on the configuration of a particular chassis. Disk Pools are the smallest unit within the Storage Pools hierarchy. OneFS provisioning works on the premise of dividing similar nodes' drives into sets, or disk pools, with each pool representing a separate failure domain. These disk pools are protected by default at +2d:1n (or the ability to withstand two drives or one entire node failure).

Disk pools are laid out across all five sleds in each Gen6 node. For example, a node with three drives per sled will have the following disk pool configuration:

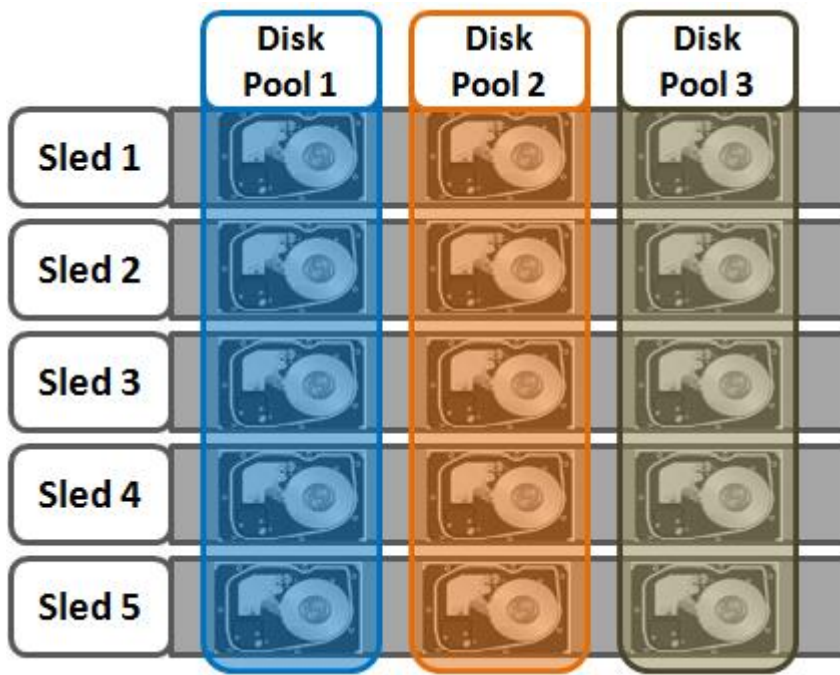


Figure 19. OneFS Disk Pools

Node Pools are groups of Disk Pools, spread across similar storage nodes (compatibility classes). This is illustrated in figure 20, below. Multiple groups of different node types can work together in a single, heterogeneous cluster. For example: one Node Pool of F-Series nodes for I/Ops-intensive applications, one Node Pool of H-Series nodes, primarily used for high-concurrent and sequential workloads, and one Node Pool of A-series nodes, primarily used for nearline and/or deep archive workloads.

This allows OneFS to present a single storage resource pool comprising multiple drive media types – SSD, high speed SAS, large capacity SATA, etc. - providing a range of different performance, protection and capacity characteristics. This heterogeneous storage pool in turn can support a diverse range of applications and workload requirements with a single, unified point of management. It also facilitates the mixing of older and newer hardware, allowing for simple investment protection even across product generations, and seamless hardware refreshes.

Each Node Pool only contains disk pools from the same type of storage nodes and a disk pool may belong to exactly one node pool. For example, F-Series nodes with 1.6TB SSD drives would be in one node pool, whereas A-Series nodes with 10 TB SATA Drives would be in another. Today, a minimum of 4 nodes (one chassis) are required per node pool for Gen6 hardware, such as the Isilon H600, or three nodes per pool for self-contained nodes like the PowerScale F600.

OneFS 'neighborhoods' are fault domains within a node pool, and their purpose is to improve reliability in general, and guard against data unavailability from the accidental removal of drive sleds. For self-contained nodes like the PowerScale F200 and F600, OneFS has an ideal size of 20 nodes per node pool, and a maximum size of 39 nodes. On the addition of the 40th node, the nodes split into two neighborhoods of twenty nodes.

With the Gen6 platform, the ideal size of a neighborhood changes from 20 to 10 nodes. This protects against simultaneous node-pair journal failures and full chassis failures.

Partner nodes are nodes whose journals are mirrored. With the Gen6 platform, rather than each node storing its journal in NVRAM as in previous platforms, the nodes' journals are stored on SSDs - and every journal has a mirror copy on another node. The node that contains the mirrored journal is referred to as the partner node. There are several reliability benefits gained from the changes to the journal. For example, SSDs are more persistent and reliable than NVRAM, which requires a charged battery to retain state. Also, with the mirrored journal, both journal drives have to die before a journal is considered lost. As such, unless both of the mirrored journal drives fail, both of the partner nodes can function as normal.

With partner node protection, where possible, nodes will be placed in different neighborhoods - and hence different failure domains. Partner node protection is possible once the cluster reaches five full chassis (20 nodes) when, after the first neighborhood split, OneFS places partner nodes in different neighborhoods:

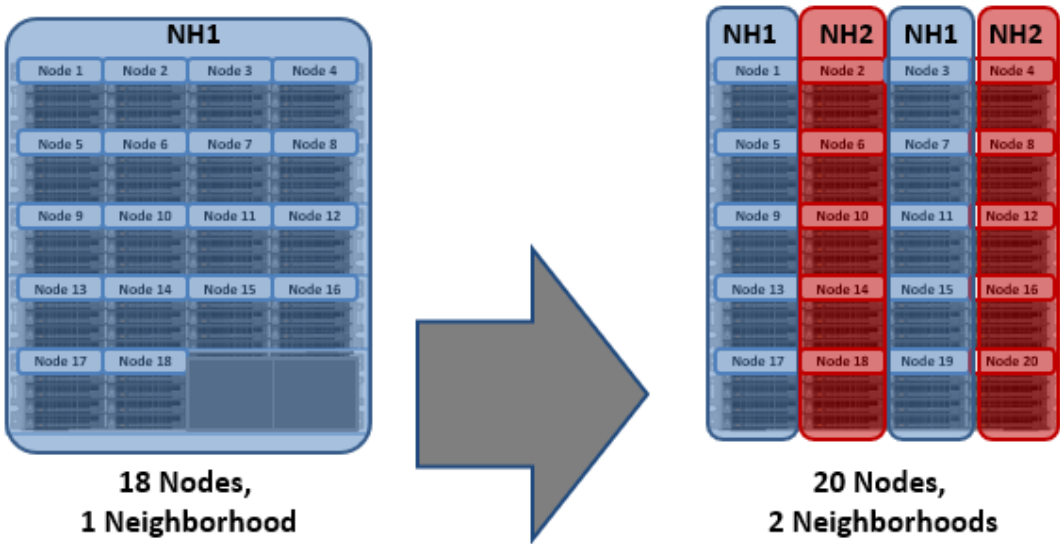


Figure 20. Split to Two Neighborhoods at Twenty Nodes

Partner node protection increases reliability because if both nodes go down, they are in different failure domains, so their failure domains only suffer the loss of a single node.

With chassis protection, when possible, each of the four nodes within a chassis will be placed in a separate neighborhood. Chassis protection becomes possible at 40 nodes, as the neighborhood split at 40 nodes enables every node in a chassis to be placed in a different neighborhood. As such, when a 38 node Gen6 cluster is expanded to 40 nodes, the two existing neighborhoods will be split into four 10-node neighborhoods:

Chassis protection ensures that if an entire chassis failed, each failure domain would only lose one node.

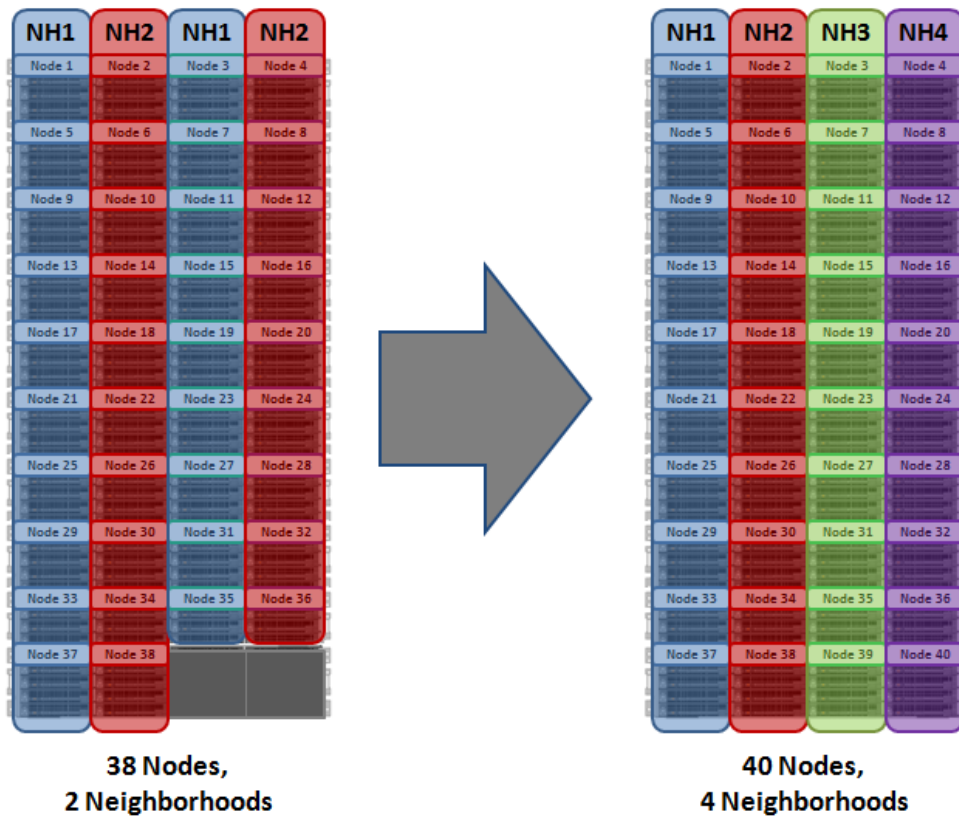


Figure 21. OneFS Neighborhoods – Four Neighborhood Split

① A 40 node or larger cluster with four neighborhoods, protected at the default level of +2d:1n can sustain a single node failure per neighborhood. This protects the cluster against a single Gen6 chassis failure.

Overall, a Gen6 platform cluster will have a reliability of at least one order of magnitude greater than previous generation clusters of a similar capacity as a direct result of the following enhancements:

- Mirrored Journals
- Smaller Neighborhoods
- Mirrored Boot Drives

Compatibility


Certain similar, but non-identical, node types can be provisioned to an existing node pool by node compatibility. OneFS requires that a node pool must contain a minimum of three nodes.

① Due to significant architectural differences, there are no node compatibilities between the Gen6 platform, previous hardware generations, or the PowerScale nodes.

OneFS also contains an SSD compatibility option, which allows nodes with dissimilar capacity SSDs to be provisioned to a single node pool.

The SSD compatibility is created and described in the OneFS WebUI SmartPools Compatibilities list and is also displayed in the Tiers & Node Pools list.

① When creating this SSD compatibility, OneFS automatically checks that the two pools to be merged have the same number of SSDs, tier, requested protection, and L3 cache settings. If these settings differ, the OneFS WebUI will prompt for consolidation and alignment of these settings.

 More information is available in the [SmartPools white paper](#).

Supported protocols

Clients with adequate credentials and privileges can create, modify, and read data using one of the standard supported methods for communicating with the cluster:

- NFS (Network File System)
- SMB/CIFS (Server Message Block/Common Internet File System)
- FTP (File Transfer Protocol)
- HTTP (Hypertext Transfer Protocol)
- HDFS (Hadoop Distributed File System)
- REST API (Representational State Transfer Application Programming Interface)
- S3 (Object Storage API)

For the NFS protocol, OneFS 9.2 supports both NFSv3 and NFSv4. Additionally, OneFS 9.2 includes support for NFSv3overRDMA.

On the Microsoft Windows side, the SMB protocol is supported up to version 3. As part of the SMB3 dialect, OneFS supports the following features:

- SMB3 Multi-path
- SMB3 Continuous Availability and Witness
- SMB3 Encryption

SMB3 encryption can be configured on a per-share, per-zone, or cluster-wide basis. Only operating systems that support SMB3 encryption can work with encrypted shares. These operating systems can also work with unencrypted shares if the cluster is configured to allow non-encrypted connections. Other operating systems can access non-encrypted shares only if the cluster is configured to allow non-encrypted connections.


The file system root for all data in the cluster is /ifs (the OneFS file system). This can be presented via SMB protocol as an 'ifs' share (\\<cluster_name>\ifs), and via the NFS protocol as a '/ifs' export (<cluster_name>:/ifs).

① Data is common between all protocols, so changes made to file content via one access protocol are instantly viewable from all others.

OneFS provides full support for both IPv4 and IPv6 environments across the front-end Ethernet network(s), SmartConnect, and the complete array of storage protocols and management tools.

Additionally, OneFS CloudPools supports the following cloud providers' storage APIs, allowing files to be stubbed out to a number of storage targets, including:

- Amazon Web Services S3
- Microsoft Azure
- Google Cloud Service
- Alibaba Cloud
- Dell EMC ECS
- OneFS RAN (RESTful Access to Namespace)

 More information is available in the [CloudPools administration guide](#).

Non-disruptive operations - protocol support

OneFS contributes to data availability by supporting dynamic NFSv3 and NFSv4 failover and failback for Linux and UNIX clients, and SMB3 continuous availability for Windows clients. This ensures that when a node failure occurs, or preventative maintenance is performed, all in-flight reads and writes are handed off to another node in the cluster to finish its operation without any user or application interruption.

During failover, clients are evenly redistributed across all remaining nodes in the cluster, ensuring minimal performance impact. If a node is brought down for any reason, including a failure, the virtual IP addresses on that node is seamlessly migrated to another node in the cluster.

When the offline node is brought back online, SmartConnect automatically rebalances the NFS and SMB3 clients across the entire cluster to ensure maximum storage and performance utilization. For periodic system maintenance and software updates, this functionality allows for per-node rolling upgrades, affording full-availability throughout the duration of the maintenance window.

File filtering

OneFS file filtering can be used across NFS and SMB clients to allow or disallow writes to an export, share, or access zone. This feature prevents certain types of file extensions to be blocked, for files which might cause security problems, productivity disruptions, throughput issues or storage clutter. Configuration can be either via an exclusion list, which blocks explicit file extensions, or an inclusion list, which explicitly allows writes of only certain file types.

Data deduplication - SmartDedupe

SmartDedupe product maximizes the storage efficiency of a cluster by decreasing the amount of physical storage required to house an organization's data. Efficiency is achieved by scanning the on-disk data for identical blocks and then eliminating the duplicates. This approach is commonly referred to as post-process, or asynchronous, deduplication.

After duplicate blocks are discovered, SmartDedupe moves a single copy of those blocks to a special set of files known as shadow stores. During this process, duplicate blocks are removed from the actual files and replaced with pointers to the shadow stores.

With post-process deduplication, new data is first stored on the storage device and then a subsequent process analyzes the data looking for commonality. This means that initial file write or modify performance is not impacted, since no additional computation is required in the write path.

SmartDedupe architecture

The OneFS SmartDedupe architecture is comprised of five principle modules:

- Deduplication Control Path
- Deduplication Job
- Deduplication Engine
- Shadow Store
- Deduplication Infrastructure

The SmartDedupe control path comprises the OneFS Web Management Interface (WebUI), command line interface (CLI) and RESTful platform API, and is responsible for managing the configuration, scheduling and control of the deduplication job. The job itself is a highly distributed background process that manages the orchestration of deduplication across all the nodes in the cluster. Job control encompasses file system scanning, detection and sharing of matching data blocks, in concert with the Deduplication Engine. The Deduplication Infrastructure layer is the kernel module that performs the consolidation of shared data blocks into shadow stores, the file system containers that hold both physical data blocks and references, or pointers, to shared blocks. These elements are described in more detail below.



Figure 22: OneFS SmartDedupe Modular Architecture

Further information is available in the [OneFS SmartDedupe](#) white paper.

Shadow stores

OneFS shadow stores are file system containers that allow data to be stored in a sharable manner. As such, files on OneFS can contain both physical data and pointers, or references, to shared blocks in shadow stores.

Shadow stores are similar to regular files, but typically don't contain all the metadata typically associated with regular file inodes. In particular, time-based attributes (creation time, modification time, etc.) are explicitly not maintained. Each shadow store can contain up to 256 blocks, with each block able to be referenced by 32,000 files. If this 32K reference limit is exceeded, a new shadow store is created. Additionally, shadow stores do not reference other shadow stores. And snapshots of shadow stores are not allowed, since shadow stores have no hard links.

① Shadow stores are also utilized for OneFS file clones and small file storage efficiency (SFSE), in addition to deduplication.

Small File Storage Efficiency

Another principle consumer of shadow stores is OneFS Small File Storage Efficiency. Introduced in OneFS 8.0.1, this feature maximizes the space utilization of a cluster by decreasing the amount of physical storage required to house the small files that often comprise an archive dataset, such as found in healthcare PACS workflows.

Efficiency is achieved by scanning the on-disk data for small files, which are protected by full copy mirrors, and packing them in shadow stores. These shadow stores are then parity protected, rather than mirrored, and typically provide storage efficiency of 80% or greater.

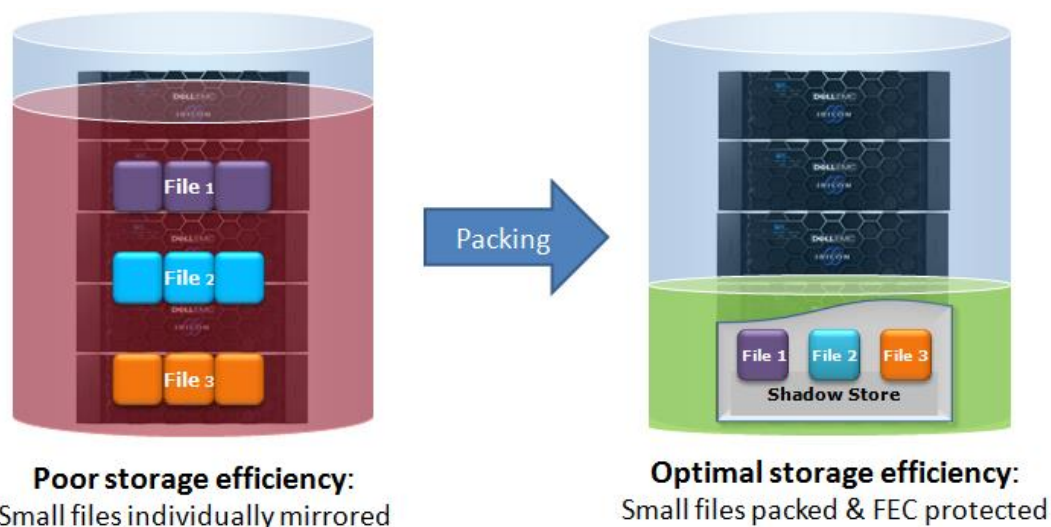


Figure 23: Small File Containerization

Small File Storage Efficiency trades a small read latency performance penalty for improved storage utilization. The archived files obviously remain writable, but when containerized files with shadow references are deleted, truncated or overwritten it can leave unreferenced blocks in shadow stores. These blocks are later freed and can result in holes which reduces the storage efficiency.

The actual efficiency loss depends on the protection level layout used by the shadow store. Smaller protection group sizes are more susceptible, as are containerized files, since all the blocks in containers have at most one referring file and the packed sizes (file size) are small.

A defragmenter is provided in OneFS 8.2 and later to reduce the fragmentation of files as a result of overwrites and deletes. This shadow store defragmenter is integrated into the ShadowStoreDelete job. The defragmentation process works by dividing each containerized file into logical chunks (~32MB each) and assess each chunk for fragmentation.

If the storage efficiency of a fragmented chunk is below target, that chunk is processed by evacuating the data to another location. The default target efficiency is 90% of the maximum storage efficiency available with the protection level used by the shadow store. Larger protection group sizes can tolerate a higher level of fragmentation before the storage efficiency drops below this threshold.

In-line Data Reduction

OneFS in-line data reduction is available exclusively on the Isilon F810, PowerScale F600, and PowerScale F200 all-flash nodes and Isilon H5600 deep-hybrid chassis. The OneFS architecture is comprised of the following principle components:

- Data Reduction Platform
- Compression Engine and Chunk Map
- Zero block removal phase
- Deduplication In-memory Index and Shadow Store Infrastructure
- Data Reduction Alerting and Reporting Framework
- Data Reduction Control Path

The in-line data reduction write path comprises three main phases:

- Zero Block Removal
- In-line Deduplication
- In-line Compression

If both in-line compression and deduplication are enabled on a cluster, zero block removal is performed first, followed by dedupe, and then compression. This order allows each phase to reduce the scope of work each subsequent phase.



Figure 24: In-line data reduction workflow

The Isilon F810 includes a hardware compression off-load capability, with each node in an F810 chassis containing a Mellanox InnoVa-2 Flex Adapter. This means that compression and decompression are transparently performed by the Mellanox adapter with minimal latency, thereby avoiding the need for consuming a node's expensive CPU and memory resources.

The OneFS hardware compression engine uses zlib, with a software implementation of igzip for the PowerScale F600 & F200, and Isilon H5600 nodes. Software compression is also used as fallback in the event of a compression hardware failure, and in a mixed cluster, for use in non-F810 nodes without a hardware compression capability, and as fallback in the event of a compression hardware failure. OneFS employs a compression chunk size of 128KB, with each chunk comprising sixteen 8KB data blocks. This is optimal since it is also the same size that OneFS uses for its data protection stripe units, providing simplicity and efficiency, by avoiding the overhead of additional chunk packing.

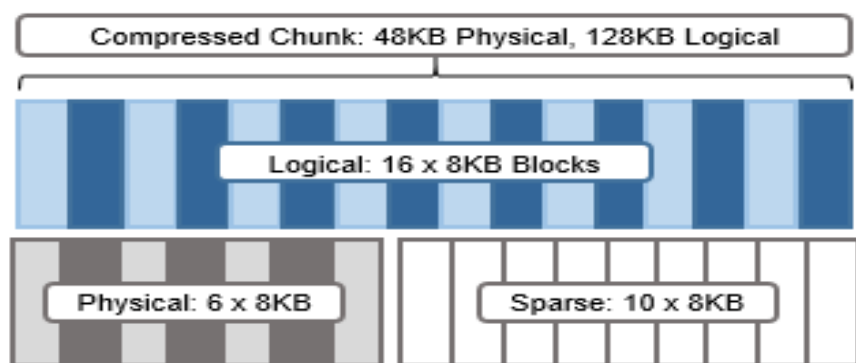


Figure 25: Compression chunks and OneFS transparent overlay

Consider the diagram above. After compression, this chunk is reduced from sixteen to six 8KB blocks in size. This means that this chunk is now physically 48KB in size. OneFS provides a transparent logical overlay to the physical attributes. This overlay describes whether the backing data is compressed or not and which blocks in the chunk are physical or sparse, such that file system consumers are unaffected by compression. As such, the compressed chunk is logically represented as 128KB in size, regardless of its actual physical size.

Efficiency savings must be at least 8KB (one block) in order for compression to occur, otherwise that chunk or file will be passed over and remain in its original, uncompressed state. For example, a file of 16KB that yields 8KB (one block) of savings would be compressed. Once a file has been compressed, it is then FEC protected.

Compression chunks will never cross node pools. This avoids the need to de-compress or recompress data to change protection levels, perform recovered writes, or otherwise shift protection-group boundaries.

Dynamic scale / scale on demand

Performance and capacity

In contrast to traditional storage systems that must “scale up” when additional performance or capacity is needed, OneFS enables a storage system to “scale out,” seamlessly increasing the existing file system or volume into petabytes of capacity while increasing performance in tandem in a linear fashion.

Adding capacity and performance capabilities to a cluster is significantly easier than with other storage systems—requiring only three simple steps for the storage administrator: adding another node into the rack, attaching the node to the back-end network, and instructing the cluster to add the additional node. The new node provides additional capacity and performance since each node includes CPU, memory, cache, network, NVRAM and I/O control pathways.

The AutoBalance feature of OneFS will automatically move data across the backend network in an automatic, coherent manner so existing data that resides on the cluster moves onto this new storage node. This automatic rebalancing ensures the new node will not become a hot spot for new data and that existing data is able to gain the benefits of a more powerful storage system. The AutoBalance feature of OneFS is also completely transparent to the end user and can be adjusted to minimize impact on high-performance workloads. This capability alone allows OneFS to scale transparently, on-the-fly, from TBs to PBs with no added management time for the administrator or increase in complexity within the storage system.

A large-scale storage system must provide the performance required for a variety of workflows, whether they be sequential, concurrent, or random. Different workflows will exist between applications and within individual applications. OneFS provides for all of these needs simultaneously with intelligent software. More importantly, with OneFS, throughput and IOPS scale linearly with the number of nodes present in a single system. Due to balanced data distribution, automatic rebalancing and distributed processing, OneFS is able to leverage additional CPUs, network ports, and memory as the system scales.

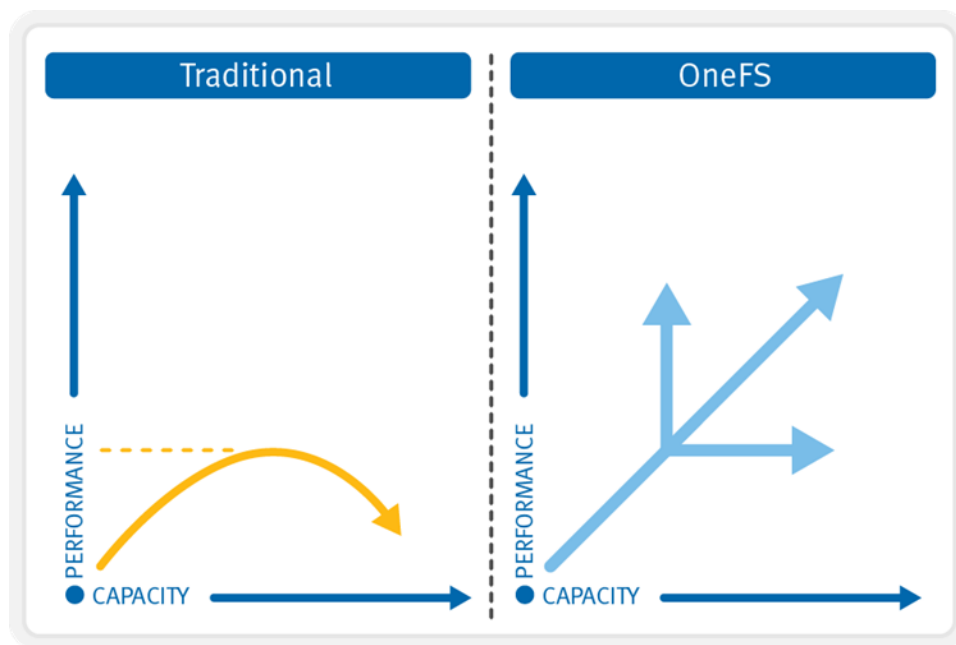


Figure 26: OneFS Linear Scalability

Interfaces

Administrators can use multiple interfaces to administer a storage cluster in their environments:

- Web Administration User Interface ("WebUI")
- Command Line Interface via SSH network access or RS232 serial connection
- LCD Panel on the nodes themselves for simple add/remove functions
- RESTful Platform API for programmatic control and automation of cluster configuration and management.

Dashboard

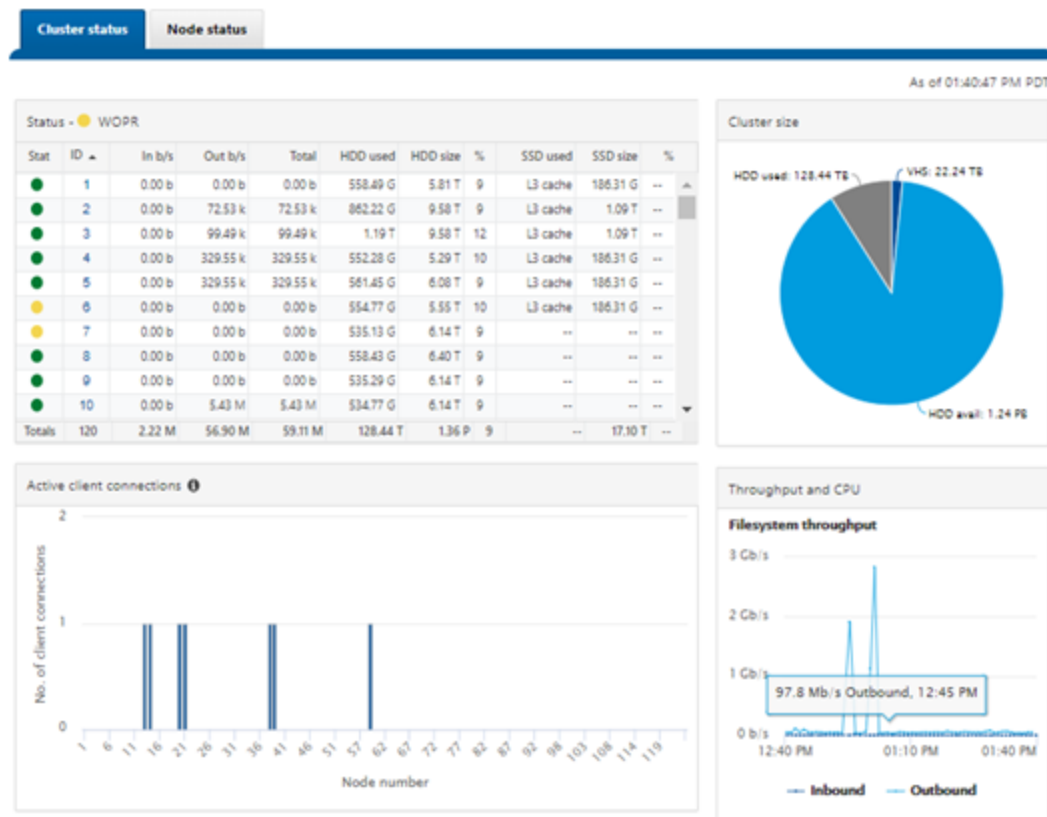


Figure 27: OneFS Web User Interface

More information on OneFS commands and feature configuration is available in the [OneFS Administration Guide](#).

Authentication and access control

Authentication services offer a layer of security by verifying users' credentials before allowing them to access and modify files. OneFS supports four methods for authenticating users:

- Active Directory (AD)
- LDAP (Lightweight Directory Access Protocol)
- NIS (Network Information Service)
- Local users & Groups

OneFS supports the use of more than one authentication type. However, it is recommended that you fully understand the interactions between authentication types before enabling multiple methods on the cluster. Refer to the product documentation for detailed information about how to properly configure multiple authentication modes.

Active Directory

Active Directory, a Microsoft implementation of LDAP, is a directory service that can store information about the network resources. While Active Directory can serve many functions, the primary reason for joining the cluster to the domain is to perform user and group authentication.

You can configure and manage a cluster's Active Directory settings from the Web Administration interface or the command-line interface; however, it is recommended that you use Web Administration whenever possible.

Each node in the cluster shares the same Active Directory machine account making it very easy to administer and manage.

LDAP

The Lightweight Directory Access Protocol (LDAP) is a networking protocol used for defining, querying, and modifying services and resources. A primary advantage of LDAP is the open nature of the directory services and the ability to use LDAP across many platforms. The clustered storage system can use LDAP to authenticate users and groups in order to grant them access to the cluster.

NIS

The Network Information Service (NIS), designed by Sun Microsystems, is a directory services protocol that can be used by OneFS to authenticate users and groups when accessing the cluster. NIS, sometimes referred to as Yellow Pages (YP), is different from NIS+, which OneFS does not support.

Local users

OneFS supports local user and group authentication. You can create local user and group accounts directly on the cluster, using the WebUI interface. Local authentication can be useful when directory services—Active Directory, LDAP, or NIS—are not used, or when a specific user or application needs to access the cluster.

Access zones

Access zones provide a method to logically partition cluster access and allocate resources to self-contained units, thereby providing a shared tenant, or multi-tenant, environment. To facilitate this, Access Zones tie together the three core external access components:

- Cluster network configuration
- File protocol access
- Authentication


As such, SmartConnect zones are associated with a set of SMB shares, NFS exports, HDFS racks, and one or more authentication providers per zone for access control. This provides the benefits of a centrally managed single file system, which can be provisioned and secured for multiple tenants. This is particularly useful for enterprise environments where multiple separate business units are served by a central IT department. Another example is during a server consolidation initiative, when merging multiple Windows file servers that are joined to separate, un-trusted, Active Directory forests.

With Access Zones, the built-in System access zone includes an instance of each supported authentication provider, all available SMB shares, and all available NFS exports by default.

These authentication providers can include multiple instances of Microsoft Active Directory, LDAP, NIS, and local user or group databases.

Roles based administration

Roles Based Administration is a cluster management roles-based access control system (RBAC) that divides up the powers of the “root” and “administrator” users into more granular privileges and allows assignment of these to specific roles. These roles can then be granted to other non-privileged users. For example, data center operations staff can be assigned read-only rights to the entire cluster, allowing full monitoring access but no configuration changes to be made. OneFS provides a collection of built-in roles, including audit, system & security administrator, plus the ability to create custom defined roles, either per access zone or across the cluster. Roles Based Administration is integrated with the OneFS command line interface, WebUI and Platform API.

 For more information on identity management, authentication, and access control in multi-protocol environments, please refer to the [OneFS Multiprotocol Security Guide](#).

OneFS auditing

OneFS provides the ability to audit system configuration and NFS, SMB, and HDFS protocol activity on a cluster. This allows organizations to satisfy various data governance and regulatory compliance mandates that they may be bound to.

All audit data is stored and protected within the cluster file system and is organized by audit topic. From here, audit data can be exported via the Dell EMC Common Event Enabler (CEE) framework to third party applications like Varonis DatAdvantage and Symantec Data Insight. OneFS Protocol auditing can be enabled per Access Zone, allowing granular control across the cluster.

With OneFS 8.0.1 and later, a cluster can write audit events across up to five CEE servers per node in a parallel, load-balanced configuration. This allows OneFS to deliver an end to end, enterprise grade audit solution.

 Further information is available in the [OneFS Audit](#) white paper.

Software upgrade

Upgrading to the latest version of OneFS allows you to take advantage of any new features, fixes and functionality. Clusters can be upgraded using two methods: Simultaneous or Rolling Upgrade

Simultaneous upgrade

A simultaneous upgrade installs the new operating system and reboots all nodes in the cluster at the same time. A simultaneous upgrade requires a temporary, sub-two-minute, interruption of service during the upgrade process while the nodes are restarted.

Rolling upgrade

A rolling upgrade individually upgrades and restarts each node in the cluster sequentially. During a rolling upgrade, the cluster remains online and continues serving data to clients with no interruption in service. Prior to OneFS 8.0, a rolling upgrade can only be performed within a OneFS code version family and not between OneFS major code version revisions. From OneFS 8.0 onwards, every new release will be rolling-upgradable from the prior version.

Non-disruptive upgrades

Non-disruptive upgrades (NDUs) allow a cluster administrator to upgrade the storage OS while their end users continue to access data without error or interruption. Updating the operating system on a cluster is a simple matter of a rolling upgrade. During this process, one node at a time is upgraded to the new code, and the active NFS and SMB3 clients attached to it are automatically migrated to other nodes in the cluster. Partial upgrade is also permitted, whereby a subset of cluster nodes can be upgraded. The subset of nodes may also be grown during the upgrade. In OneFS 8.2 and later, an upgrade can be paused and resumed allowing customers to span upgrades over multiple smaller Maintenance Windows. Additionally, OneFS 8.2.2 and later offer parallel upgrades, whereby clusters can upgrade an entire neighborhood, or fault domain, at a time, substantially reducing the duration of large cluster upgrades. OneFS 9.2 introduces drain based upgrades, whereby nodes are prevented from rebooting or restarting protocol services until all SMB clients have disconnected from the node.

Roll back capable

OneFS supports upgrade rollback, providing the ability to return a cluster with an uncommitted upgrade to its previous version of OneFS.

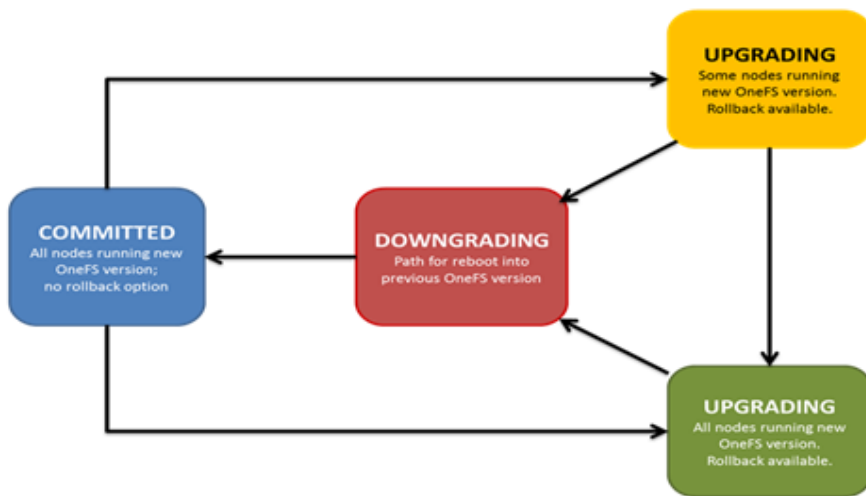


Figure 28: OneFS Non-Disruptive Upgrade States

Automatic firmware updates

OneFS powered clusters support automatic drive firmware updates for new and replacement drives, as part of the non-disruptive firmware update process. Firmware updates are delivered via drive support packages, which both simplify and streamline the management of existing and new drives across the cluster. This ensures that drive firmware is up to date and mitigates the likelihood of failures due to known drive issues. As such, automatic drive firmware updates are an important component of OneFS' high availability and non-disruptive operations strategy. Drive and node firmware can be applied as either a rolling upgrade or via a full cluster reboot.

Prior to OneFS 8.2, node firmware updates had to be installed one node at a time, which was a time-consuming operation especially in large clusters. Node firmware updates can now be choreographed across a cluster by providing a list of nodes to be simultaneously updated. The upgrade helper tool can be used to select a desired combination of nodes that can be updated simultaneously and an explicit list of nodes that should not be updated together (for example, nodes in a node-pair).

Performing the upgrade

As part of an upgrade, OneFS automatically runs a pre-install verification check. This verifies that the configuration in your current installation of OneFS is compatible with the version of OneFS that is intended for the upgrade. When an unsupported configuration is found, the upgrade is stopped and instructions on troubleshooting the issue are displayed. Proactively running the pre-installation upgrade check before starting an upgrade helps to avoid any interruption due to incompatible configuration.

OneFS data protection and management software

OneFS offers a comprehensive portfolio of data protection and management software to address your needs:

Software module	Function	Description
<u>CloudIQ™</u>	Cluster health monitoring	Implement intelligent and predictive analytics to proactively monitor the health of your cluster.
<u>InsightIQ™</u>	Performance Management	Maximize performance of your cluster with innovative performance monitoring and reporting tools
<u>DataIQ™</u>	Data Analysis and Management	Locate, access and manage data in seconds, no matter where it resides – across file and object storage, on-prem or in the cloud. Gain a holistic view across heterogeneous storage systems with a single pane of glass, effectively breaking down data trapped in siloes.
<u>SmartPools™</u>	Resource Management	Implement a highly efficient, automated tiered storage strategy to optimize storage performance and costs
<u>SmartQuotas™</u>	Data Management	Assign and manage quotas that seamlessly partition and thin provision storage into easily managed segments at the cluster, directory, sub-directory, user, and group levels
<u>SmartConnect™</u>	Data Access	Enable client connection load balancing and dynamic NFS failover and failback of client connections across storage nodes to optimize use of cluster resources
<u>SnapshotIQ™</u>	Data Protection	Protect data efficiently and reliably with secure, near instantaneous snapshots while incurring little to no performance overhead. Speed recovery of critical data with near-immediate on-demand snapshot restores.
<u>SyncIQ™</u>	Data Replication	Replicate and distribute large, mission-critical data sets asynchronously to multiple shared storage systems in multiple sites for reliable disaster recovery capability. Push-button failover and failback simplicity to increase availability of mission-critical data.
<u>SmartLock™</u>	Data Retention	Protect your critical data against accidental, premature or malicious alteration or deletion with our software-based approach to Write Once Read Many (WORM) and meet stringent compliance and governance needs such as SEC 17a-4 requirements.
<u>SmartDedupe™</u>	Data Deduplication	Maximize storage efficiency by scanning the cluster for identical blocks and then eliminating the duplicates, decreasing the amount of physical storage required.
<u>CloudPools™</u>	Cloud Tiering	CloudPools enables you to define which data on your cluster should be archived to cloud storage. Cloud providers include Microsoft Azure, Google Cloud, Amazon S3, Dell EMC ECS, and native OneFS.

Table 3: Dell EMC Power Scale Data Services Portfolio

Please refer to product documentation for details.

Conclusion

With Dell EMC scale-out NAS solutions powered by the OneFS operating system, organizations can scale from TBs to PBs within a single file system, single volume, with a single point of administration. OneFS delivers high-performance, high-throughput, or both, without adding management complexity.

Next-generation data centers must be built for sustainable scalability. They will harness the power of automation, leverage the commoditization of hardware, ensure the full consumption of the network fabric, and provide maximum flexibility for organizations intent on satisfying an ever-changing set of requirements.

OneFS is the next-generation file system designed to meet these challenges. OneFS provides:

- Fully distributed single file system
- High-performance, fully symmetric cluster
- File striping across all nodes in a cluster
- Automated software to eliminate complexity
- Dynamic content balancing
- Flexible data protection
- High availability
- Web-based and command-line administration

OneFS is ideally suited for file-based and unstructured “Big Data” applications in enterprise data lake environments – including large-scale home directories, file shares, archives, virtualization and business analytics – as well as a wide range of data-intensive, high performance computing environments including energy exploration, financial services, Internet and hosting services, business intelligence, engineering, manufacturing, media & entertainment, bioinformatics, and scientific research.

TAKE THE NEXT STEP

Contact your Dell EMC sales representative or authorized reseller to learn more about how PowerScale and Isilon NAS storage solutions can benefit your organization.

[Visit Dell EMC PowerScale](#) to compare features and get more information.



Learn more about Dell
EMC PowerScale
solutions



Contact a Dell EMC Expert



View more resources



Join the conversation
with #DellEMCStorage