Storage Tiering with Dell PowerScale SmartPools

June 2025

H8321.24

White Paper

Abstract

This white paper provides a technical overview of Dell PowerScale SmartPools software and its native, policy-based tiering capability. This SmartPools capability enables enterprises to reduce storage costs and optimize their storage investment by automatically moving data to the most appropriate storage tier within a OneFS cluster.

Copyright

Contents

Executive summary

Overview

Dell PowerScale SmartPools software enables multiple levels of performance, protection, and storage density to co-exist within the same file system. SmartPools unlocks the ability to aggregate and consolidate a wide range of applications within a single extensible, ubiquitous storage resource pool. This architecture helps provide granular performance optimization, workflow isolation, higher utilization, and independent scalability—all with a single point of management.

SmartPools allows you to define the value of the data within your workflows based on policies, and automatically aligns data to the appropriate price/performance tier over time. Data movement is seamless. With file-level granularity and control through automated policies, manual control, or API interface, you can tune performance and layout, storage tier alignment, and protection settings. All this tuning has minimal impact to your end users.

Storage tiering has a convincing value proposition, namely separating data according to its business value, and aligning it with the appropriate class of storage and levels of performance and protection. Information Lifecycle Management techniques have been around for several years but have typically suffered from inefficiencies. They are complex to install and manage, they involve changes to the file system, they require the use of stub files, and so on.

SmartPools is a next-generation approach to tiering that facilitates the management of heterogeneous clusters. The SmartPools capability is native to the OneFS scale-out file system, which allows for unprecedented flexibility, granularity, and ease of management. SmartPools uses many of the components and attributes of OneFS, including data layout and mobility, protection, performance, scheduling, and impact management.

Audience and scope

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

This paper presents information for deploying and managing a heterogeneous Dell PowerScale OneFS cluster. This paper does not intend to provide a comprehensive background to the OneFS architecture.

For more information about the OneFS architecture, see the OneFS Technical Overview white paper.

For more information about OneFS commands and feature configuration, see the OneFS Administration Guide.

Revisions

Date	Part Number	Description
November 2013	H8321	Initial release for OneFS 7.1
June 2014	H8321.1	Updated for OneFS 7.1.1
November 2014	H8321.2	Updated for OneFS 7.2

Date	Part Number	Description
June 2015	H8321.3	Updated for OneFS 7.2.1
November 2015	H8321.4	Updated for OneFS 8.0
September 2016	H8321.5	Updated for OneFS 8.0.1
April 2017	H8321.6	Updated for OneFS 8.1
November 2017	H8321.7	Updated for OneFS 8.1.1
February 2019	H8321.8	Updated for OneFS 8.1.3
April 2019	H8321.9	Updated for OneFS 8.2
August 2019	H8321.10	Updated for OneFS 8.2.1
December 2019	H8321.11	Updated for OneFS 8.2.2
June 2020	H8321.12	Updated for OneFS 9.0
September 2020	H8321.13	Updated for OneFS 9.1
April 2021	H8321.14	Updated for OneFS 9.2
September 2021	H8321.15	Updated for OneFS 9.3
April 2022	H8321.16	Updated for OneFS 9.4
January 2023	H8321.17	Updated for OneFS 9.5
January 2024	H8321.18	Updated for OneFS 9.7
April 2024	H8321.19	Updated for OneFS 9.8
May 2024	H8321.20	Updated for PowerScale F9.9
August 2024	H8321.21	Updated for OneFS 9.9
December 2024	H8321.21	Updated for OneFS 9.10
April 2025	H8321.23	Updated for OneFS 9.11
June 2025	H8321.24	Updated for PowerScale H710/0 & A310/0

We value your feedback

Dell Technologies and the authors of this document welcome your feedback on this document. Contact the Dell Technologies team by email.

Author: Nick Trimbee

Note: For links to other documentation for this topic, see the PowerScale Info Hub.

Scale-out NAS architecture

Overview

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

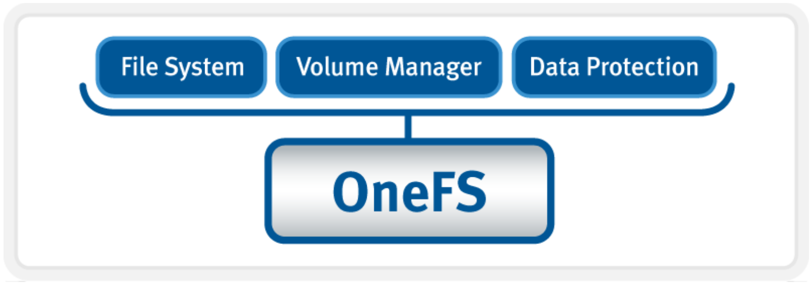


Figure 1. OneFS combines file system, volume manager, and data protection into one single intelligent distributed system.

This unified software layer is the core innovation that directly enables enterprises to successfully use 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 stores and manages data in a cluster.

Single file system

OneFS provides a single file system that operates outside the constraints of traditional scale up constructs such as RAID groups. OneFS allows data to be placed anywhere on the cluster, accommodating various levels of performance and protection.

Each cluster operates within a single volume and namespace with the file system distributed across all nodes. As such, there is no partitioning, and clients are provided a coherent view of their data from any node in the cluster.

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

From an application or user perspective, all capacity is available - the storage has been virtualized for the users and administrator. The file system can grow organically without requiring too much planning or oversight. The administrator does not have to be concerned about tiering files to the appropriate disk because SmartPools handles that automatically. Also, no special considerations need be given to how the administrator might replicate such a large file system. The OneFS SyncIQ service automatically parallelizes the transfer of the data to one or more alternate clusters.

Note: See the OneFS Technical Overview white paper for further details about the OneFS architecture.

Data layout and protection

OneFS is designed to withstand multiple simultaneous component failures (currently four per node pool) while still affording unfettered access to the entire file system and dataset. Data protection is implemented at the file level using Reed Solomon erasure coding, and, thus, is not dependent on any hardware RAID controllers. This implementation provides many benefits, including the ability to add new data protection schemes as market conditions or hardware attributes and characteristics evolve. Because protection is applied at the file level, a OneFS software upgrade is all that is required to make new protection and performance schemes available.

OneFS employs the popular Reed-Solomon erasure coding algorithm for its protection calculations. Protection is applied at the file level, enabling the cluster to recover data quickly and efficiently. Inodes, directories, and other metadata are protected at the same or higher level as the data blocks they reference. Because all data, metadata, and forward error correction (FEC) blocks are striped across multiple nodes, dedicated parity drives are not required. Striping across multiple nodes guards against single points of failure and bottlenecks, allows file reconstruction to be highly parallelized, and ensures that all hardware components in a cluster always do useful work.

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

Note: The best practice is to use the recommended protection level for a particular cluster configuration. The recommended level of protection is clearly marked as “suggested” in the OneFS WebUI storage pools configuration pages and is typically configured by default.

The hybrid protection schemes are particularly useful for high-density node configurations, such as the PowerScale A3000 chassis. In these configurations, the probability of multiple drives failing surpasses that of an entire node failure. One such example is if the file is beyond its protection level. In the unlikely event of multiple, simultaneous device failures, OneFS will reprotect everything possible and report errors on the individual files affected to the cluster’s logs.

OneFS also provides various mirroring options ranging from 2x to 8x, allowing from two to eight mirrors of the specified content. The mirroring method is used for protecting OneFS metadata. Metadata, for example, is mirrored at one level above FEC by default. For example, if a file is protected at +1n, its associated metadata object is 3x mirrored.

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

Table 1. OneFS FEC protection levels

Protection level	Description
+1n	Tolerate failure of one drive OR one node
+2d:1n	Tolerate failure of two drives OR one node
+2n	Tolerate failure of two drives OR two nodes
+3d:1n	Tolerate failure of three drives OR one node
+3d:1n1d	Tolerate failure of three drives OR one node AND one drive
+3n	Tolerate failure of three drives or three nodes
+4d:1n	Tolerate failure of four drives or one node
+4d:2n	Tolerate failure of four drives or two nodes
+4n	Tolerate failure of four nodes
2x to 8x	Mirrored over two through eight nodes, depending on configuration

Also, SmartPools can manage data and metadata objects separately due to a logical separation of data and metadata structures within the single OneFS file system.

OneFS stores file and directory metadata in inodes and B-trees, allowing the file system to scale to billions of objects and still provide fast lookups of data or metadata. OneFS is a symmetric and fully distributed file system with data and metadata spread across multiple hardware devices. Generally, data is protected using erasure coding for space-efficiency reasons, enabling utilization levels around 80 percent on clusters of five or more nodes. Metadata (which generally makes up around 2 percent of the system) is mirrored for performance and availability. Protection levels are dynamically configurable at a per-file or per-file system granularity, or anything in between. Data and metadata access and locking are coherent across the cluster, and this symmetry is fundamental to the simplicity and resiliency of the OneFS shared nothing architecture.

When a client connects to a OneFS-managed node and performs a write operation, files are broken into smaller logical chunks, or stripes units, before being written to disk. Then chunks are striped across the cluster's nodes and protected either through erasure-coding or mirroring. OneFS primarily uses the Reed-Solomon erasure coding system for data protection, and mirroring for metadata. OneFS file-level protection typically provides industry-leading levels of utilization. And, for nine node and larger clusters, OneFS can sustain up to four full node failures while still providing full access to data.

OneFS uses multiple data layout methods to optimize for maximum efficiency and performance according to the data's access pattern—for example, streaming, concurrency, random, and so on. As with protection, these performance attributes can also be applied per file or per file system.

Note: For more information about OneFS data protection levels, see the OneFS Technical Overview white paper.

Job Engine

The Job Engine is OneFS' parallel task scheduling framework, and is responsible for the distribution, execution, and impact management of critical jobs and operations across the entire cluster.

The OneFS Job Engine schedules and manages all the data protection and background cluster tasks: creating jobs for each task, prioritizing them and ensuring that internode communication and cluster-wide capacity utilization and performance are balanced and optimized. The Job Engine ensures that core cluster functions have priority over less important work. It gives applications integrated with OneFS the ability to control the priority of their various functions to ensure the best resource utilization. These applications can include add-on software or applications integrating with OneFS through the OneFS API.

Each job (such as the SmartPools job, for example) has an impact profile. An impact profile includes a configurable policy, a schedule that characterizes the amount of system resources the job will take, and an impact policy and impact schedule. The amount of work a job must perform is fixed. However, the resources dedicated to that work can be tuned to minimize the impact to other cluster functions, such as serving client data.

The SmartPools feature consists of the following jobs:

Table 2. SmartPools jobs

Job	Description
SmartPools	Job that runs and moves data between the tiers of nodes within the same cluster. Also manages the CloudPools functionality if CloudPools is licensed and configured.
SmartPoolsTree	Enforces SmartPools file policies on a subtree.
FilePolicy	Efficient changelist-based SmartPools file pool policy job.
IndexUpdate	Creates and updates an efficient file system index for the FilePolicy job.
SetProtectPlus	Applies the default file policy. This job is disabled if SmartPools is activated on the cluster.

Note: When a cluster running the FSAnalyze job is upgraded to OneFS 8.2 or later, the upgrade removes the legacy FSAnalyze index and snapshots. It replaces them with new snapshots the first time that IndexUpdate is run. The new index stores considerably more file and snapshot attributes than the old FSA index. Until the IndexUpdate job effects this change, FSA keeps running on the old index and snapshots.

Data rebalancing

Another key Job Engine task is AutoBalance. This task enables OneFS to reallocate and rebalance, or restripe, data across the nodes in a cluster, making storage space utilization more uniform and efficient.

OneFS manages protection of file data directly, and when a drive or entire node failure occurs, it rebuilds data in a parallel fashion. OneFS does not require either dedicated hot spare drives, or serial drive rebuilds. Instead, it borrows from the available free space in the file system to recover from failures. This technique is called Virtual Hot Spare. This approach allows the cluster to be self-healing, without human intervention, and with the advantages of fast, parallel data reconstruction. The administrator can create a virtual hot spare reserve, which prevents users from consuming capacity that is reserved for the virtual hot spare.

The process of choosing a new layout is called restriping, and this mechanism is identical for repair, rebalance, and tiering. Data is moved to the background, and the file is always available - a process that is transparent to end users and applications.

Note: Further information is available in the OneFS Job Engine white paper.

Caching

SmartCache is a globally coherent read and write caching infrastructure that provides low latency access to content. Like other resources in the cluster, as more nodes are added, the total cluster cache grows in size, enabling OneFS to deliver predictable, scalable performance within a single file system.

A OneFS cluster provides a high cache to disk ratio (multiple GB per node), which is dynamically allocated for read operations as required. This cache is unified and coherent across all nodes in the cluster, allowing a user on one node to benefit from I/O already transacted on another node. OneFS stores only *distinct* data on each node. The node's

RAM is used as a level 2 (L2) cache of such data. These distinct, cached blocks can be quickly accessed across the backplane, and, as the cluster grows, the cache benefit increases. For this reason, the amount of I/O to disk on a OneFS cluster is most often quite lower than on traditional platforms, allowing reduced latencies and a better user experience. For sequentially accessed data, OneFS SmartRead aggressively prefetches data, greatly improving read performance across all protocols.

An optional third tier of read cache, called SmartFlash or level 3 (L3) cache, is also configurable on nodes that contain solid state drives (SSDs). SmartFlash is an eviction cache that L2 cache blocks populate as they are aged out from memory. Using SSDs for caching rather than as traditional file system storage devices offers several benefits. For example, when reserved for caching, the entire SSD is used, and writes occur in a linear and predictable way. Using SSDs for caching 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 SSDs for cache also makes sizing SSD capacity a more straightforward and less error-prone prospect compared to using SSDs as a storage tier.

OneFS write caching uses write buffering to aggregate, or coalesce, multiple write operations to the NVRAM file systems journals. The operations can be written to disk safely and more efficiently. This form of buffering reduces the disk write penalty that could result in multiple reads and writes for each write operation.

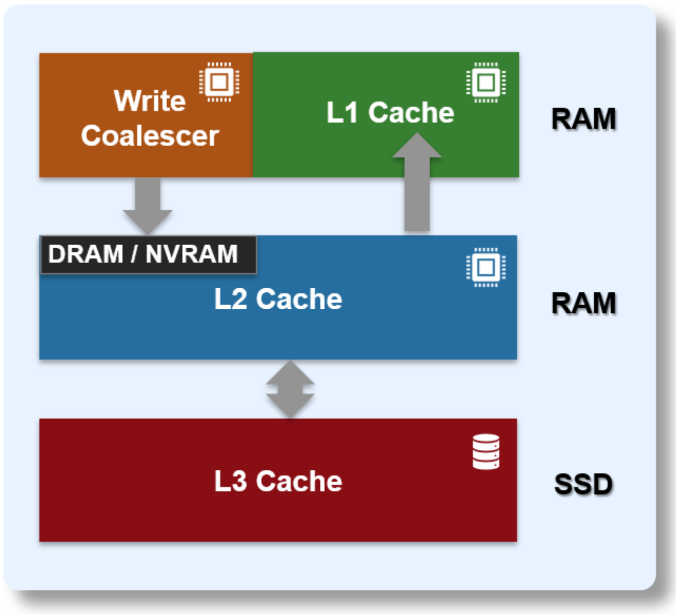


Figure 2. OneFS caching hierarchy

Note: For more information, see the OneFS SmartFlash white paper.

SmartPools

Overview

SmartPools is a next-generation data tiering product that builds directly on the core OneFS attributes. These attributes include the single file system, extensible data layout and protection framework, parallel job execution, and intelligent caching architecture.

SmartPools was originally envisioned as a combination of two fundamental notions:

- The ability to define subsets of hardware within a single, heterogeneous cluster.
- A method to associate logical groupings of files with the defined hardware subsets through simple, logical definitions or rules.

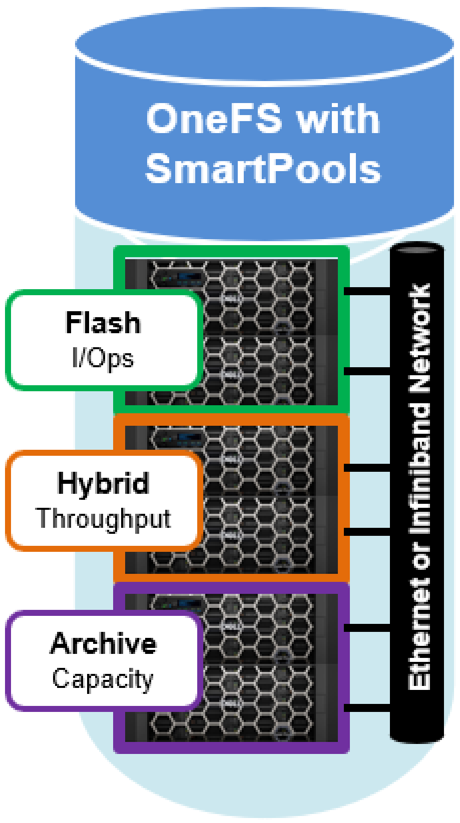
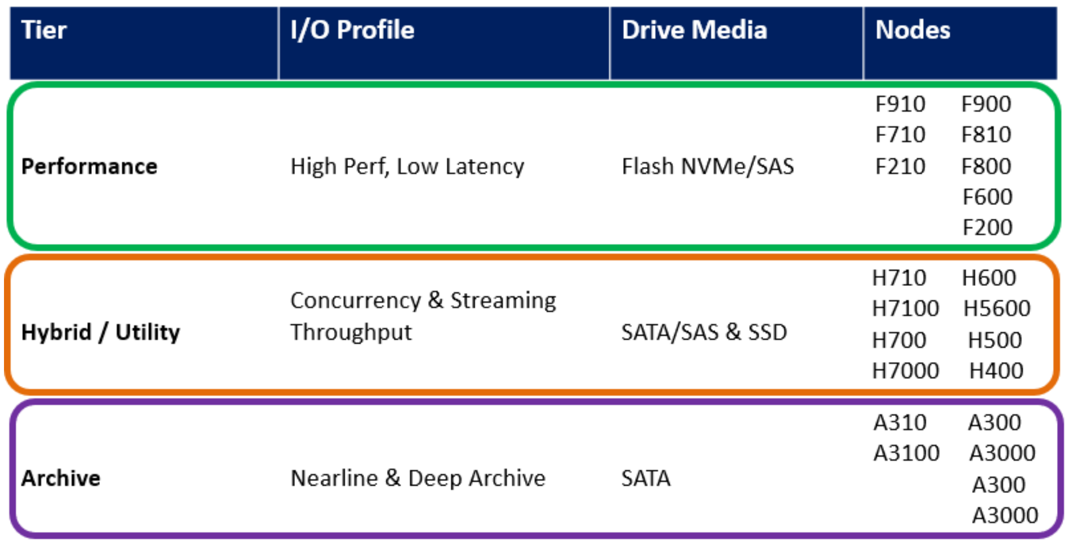


Figure 3. SmartPools tiering model

The current SmartPools implementation expands on these two core concepts in several additional ways, while maintaining simplicity and ease of use as primary goals.

Hardware tiers

Heterogeneous OneFS clusters can be architected with a wide variety of node styles and capacities to meet the needs of a varied dataset and wide spectrum of workloads. These node styles fall loosely into four main categories or tiers. The following figure illustrates these tiers, and the associated hardware models:



Tier	I/O Profile	Drive Media	Nodes
Performance	High Perf, Low Latency	Flash NVMe/SAS	F910 F900 F710 F810 F210 F800 F600 F200
Hybrid / Utility	Concurrency & Streaming Throughput	SATA/SAS & SSD	H710 H600 H7100 H5600 H700 H500 H7000 H400
Archive	Nearline & Deep Archive	SATA	A310 A300 A3100 A3000 A300 A3000

Figure 4. Hardware tiers and node types

Storage pools

Storage pools enable subsets of hardware to be defined within a single cluster. This ability allows file layout to be aligned with specific sets of nodes through the configuration of storage pool policies. The notion of storage pools is an abstraction that encompasses disk pools, node pools, and tiers, all described in the following sections.

Disk pools

Disk pools are the smallest unit within the storage pools hierarchy, as illustrated in Figure 5. 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 the failure of two disks or one entire node). The disk pools span from four to twenty nodes with modular, chassis-based platforms. Each chassis contains four compute modules (one per node), and five drive containers, or “sleds,” per node.

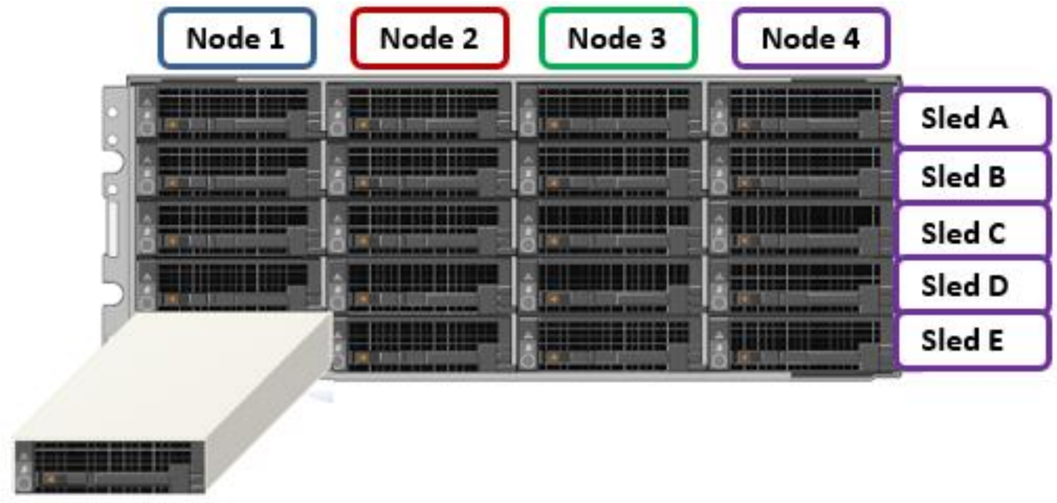


Figure 5. Modular chassis front view showing drive sleds

Each sled is a tray that slides into the front of the chassis and contains between three and six drives, depending on the configuration of a particular node chassis. Disk pools are laid out across all five sleds in each node. For example, a chassis-based node with three drives per sled would have the following disk pool configuration:

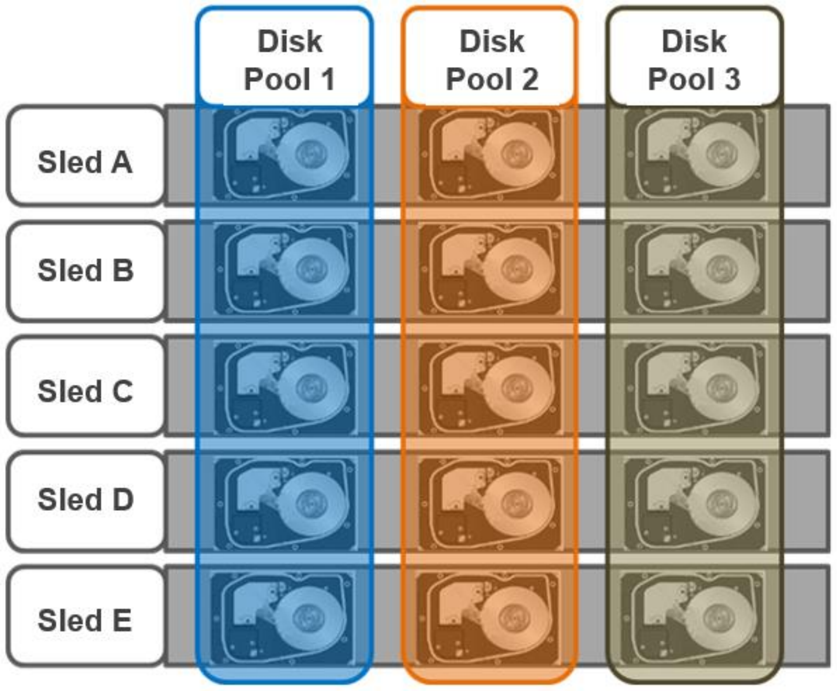


Figure 6. Modular chassis-based node disk pools

Note: Earlier generations of Isilon hardware and the current PowerScale nodes use six-drive node pools that span from three to forty nodes.

Each drive may belong to only one disk pool. Data protection stripes or mirrors do not extend across disk pools (the exception being a Global Namespace Acceleration extra mirror, as described in Global Namespace Acceleration). OneFS manages disk pools. Users cannot configure disk pools.

Node pools

Node pools are groups of disk pools, spread across similar storage nodes (equivalence classes), as illustrated in Figure 7. Multiple groups of different node types can work together in a single, heterogeneous cluster. For example:

- One node pool of all-flash F-Series nodes for IOPS-intensive applications
- One node pool of H-Series nodes, primarily used for high-concurrent and sequential workloads
- One node pool of A-Series nodes, primarily used for nearline and deep archive workloads

Node pools allow OneFS to present a single storage resource pool consisting of multiple drive media types: NVMe, SSD, high-speed SAS, and large capacity SATA. They provide 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 contains only disk pools from the same type of storage nodes, and a disk pool may only belong to one node pool. For example, all-flash F-Series nodes would be in one node pool, whereas A-Series nodes with high capacity SATA drives would be in another. Today, a minimum of four nodes, or one chassis, are required per node pool for Gen6 modular chassis-based hardware, or three previous generations of PowerScale nodes per node pool.

Nodes are not provisioned (not associated with each other and not writable) until at least three nodes from the same compatibility class are assigned in a node pool. If nodes are removed from a node pool, that pool becomes under-provisioned. In this situation, if two like nodes remain, they are still writable. If only one remains, it is automatically set to read-only.

Once node pools are created, they can be easily modified to adapt to changing requirements. Individual nodes can be reassigned from one node pool to another. Node pool associations can also be discarded, releasing member nodes so they can be added to new or existing pools. Node pools can also be renamed at any time without changing any other settings in the node pool configuration.

Any new node added to a cluster is automatically allocated to a node pool and then subdivided into disk pools without any additional configuration steps. The node inherits the SmartPools configuration properties of that node pool. Thus, the configuration of disk pool data protection, layout, and cache settings must be completed only once per node pool. Configuration can be done at the time the node pool is first created. The shared attributes of the new nodes with the closest matching node pool determine the automatic allocation. If the new node is not a close match to the nodes of any existing node pool, it remains unprovisioned until the minimum node pool node membership for like nodes is met. That minimum is three nodes of the same or similar storage and memory configuration.

When a new node pool is created, and nodes are added, SmartPools associates those nodes with an ID. This ID is also used in file pool policies and file attributes to dictate file placement within a specific disk pool.

By default, a file that a specific file pool policy does not cover goes to the default node pool or pools identified during setup. If no default is specified, SmartPools writes that data to the pool with the most available capacity.

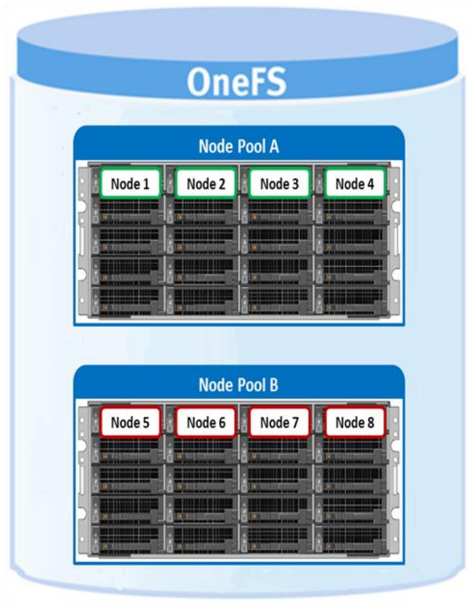


Figure 7. **Automatic provisioning into node pools**

Tiers

Tiers are groups of node pools combined into a logical superset to optimize data storage, according to OneFS platform type, as illustrated in Figure 8. For example, similar “archive” node pools are often consolidated into a single tier. This tier could incorporate different styles of archive node pools into a single, logical container. For example, PowerScale A300 with 12 TB SATA drives and PowerScale A3000 with 16 TB SATA drives. This capability is a significant benefit. It allows customers who consistently purchase the highest capacity nodes available to consolidate various node styles within a single group, or tier, and manage them as one logical group.

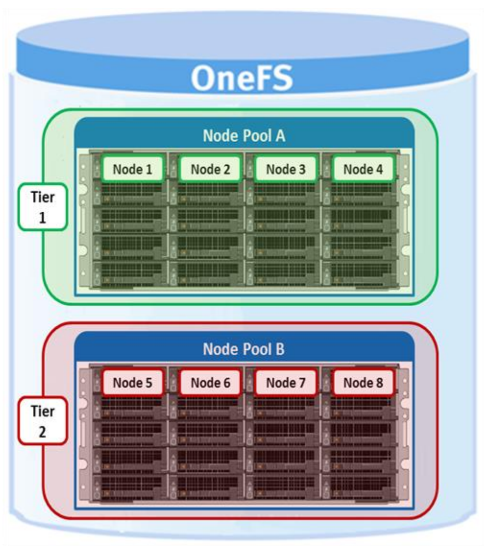
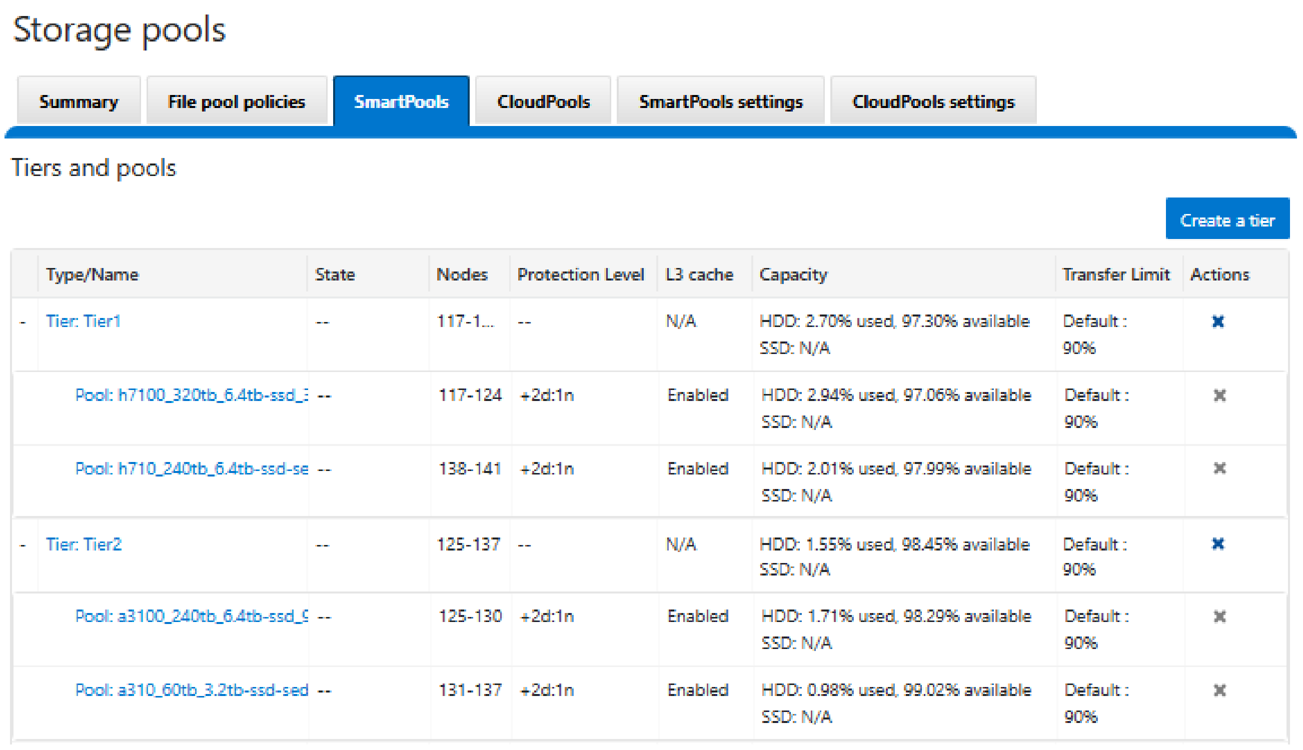


Figure 8. SmartPools tier configuration

SmartPools users typically deploy two to four tiers. The fastest tier often contains all-flash nodes for the most performance-demanding portions of a workflow, and the lowest, capacity-biased tier typically consists of high-capacity SATA drive nodes.

Figure 9. SmartPools WebUI view—tiers and node pools



Tiers and pools

	Type/Name	State	Nodes	Protection Level	L3 cache	Capacity	Transfer Limit	Actions
-	Tier: Tier1	--	117-1...	--	N/A	HDD: 2.70% used, 97.30% available SSD: N/A	Default : 90%	✖
	Pool: h7100_320tb_6.4tb-ssd_	--	117-124	+2d:1n	Enabled	HDD: 2.94% used, 97.06% available SSD: N/A	Default : 90%	✖
	Pool: h710_240tb_6.4tb-ssd-se	--	138-141	+2d:1n	Enabled	HDD: 2.01% used, 97.99% available SSD: N/A	Default : 90%	✖
-	Tier: Tier2	--	125-137	--	N/A	HDD: 1.55% used, 98.45% available SSD: N/A	Default : 90%	✖
	Pool: a3100_240tb_6.4tb-ssd_	--	125-130	+2d:1n	Enabled	HDD: 1.71% used, 98.29% available SSD: N/A	Default : 90%	✖
	Pool: a310_60tb_3.2tb-ssd-sed	--	131-137	+2d:1n	Enabled	HDD: 0.98% used, 99.02% available SSD: N/A	Default : 90%	✖

Node compatibility and equivalence

SmartPools allows nodes of any type supported by the particular OneFS version to be combined within the same cluster. The like nodes are provisioned into different node pools according to their physical attributes. The node equivalence classes are stringent to avoid disproportionate amounts of work being directed toward a subset of cluster resources. This issue is often referred to as “node bullying” and can manifest as severe overutilization of resources such as CPU, network, or disks.

However, administrators can safely target specific data to broader classes of storage by creating tiers. For example, if a cluster includes two different varieties of X nodes, the nodes are automatically provisioned into two different node pools. These two node pools can be logically combined into a tier, and file placement targeted to it, resulting in automatic balancing across the node pools.

Criteria governing the creation of a node pool include:

- A minimum of four nodes for chassis-based nodes, and three nodes for self-contained, of a particular type must be added to the cluster before a node pool is provisioned.
- If three or more nodes have been added, but fewer than three nodes are accessible, the cluster will be in a degraded state. The cluster may still be usable if cluster quorum (greater than half the total nodes available) still exists.
- All nodes in the cluster must have a current, valid support contract.

SmartPools separates hardware by node type and creates a separate node pool for each distinct hardware variant. To reside in the same node pool, nodes must have a set of core attributes in common:

- Family
- Chassis size
- Generation
- RAM capacity
- SSD or hard drive capacity and quantity

Node compatibilities can be defined to allow nodes with the same drive types, quantities and capacities, and compatible RAM configurations to be provisioned into the same pools.

Note: Due to significant architectural differences, there are no node compatibilities between the four-node modular chassis such as the PowerScale F810 platform and self-contained nodes such as the PowerScale F910.

SSD compatibility

OneFS also contains an SSD compatibility option, which allows nodes with dissimilar SSD capacity and count 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.

Note: When you create 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 are different, the OneFS WebUI prompts you to consolidate and align the settings.

Data spillover

If a node pool fills up, writes to that pool automatically spill over to the next pool. This default behavior ensures that work can continue even if one type of capacity is full. In some circumstances, spillover is undesirable. For example, when different business units within an organization purchase separate pools or when data location has security or protection implications. In these circumstances, spillover can be disabled. Disabling spillover ensures that a file exists in one pool and will not move to another. Keep in mind that reservations for virtual hot sparing affect spillover. For example, if VHS is configured to reserve 10 percent of a pool's capacity, spillover will occur at 90 percent full.

Protection settings can be configured outside SmartPools and managed at the cluster level, or within SmartPools at either the node pool or file pool level. Wherever protection levels exist, they are fully configurable and the default protection setting for a node pool is +2d:1n.

Automatic provisioning

The SmartPools framework handles data tiering and management in OneFS. 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 efficiently protected disk pools. For example, a 40-node nearline cluster with 3 TB SATA disks typically runs at a +4n protection level. However, partitioning it into two, 20-node disk pools would allow each pool to run at +2n protection. Doing so would lower the protection overhead and improve space utilization without any net increase in management overhead.

In keeping with the goal of storage management simplicity, OneFS automatically calculates and divides the cluster into pools of disks. The disks are optimized for both Mean Time to Data Loss (MTTDL) and efficient space utilization. Protection-level decisions, such as those described in the preceding 40-node cluster example, are not left to the customer.

With automatic provisioning, every set of equivalent node hardware is automatically divided into disk pools consisting of up to 40 nodes and six drives per node. These disk pools are protected against up to two drive failures per disk pool. Multiple similar disk pools are automatically combined into node pools. The node pools can be further aggregated into logical tiers and managed with SmartPools file pool policies. By subdividing a node's disks into multiple, separately protected disk pools, nodes are significantly more resilient to multiple disk failures than previously possible.

When initially configuring a cluster, OneFS automatically assigns nodes to node pools. Nodes are not pooled—not associated with each other—until at least three nodes are assigned to the node pool. A node pool with fewer than three nodes is considered an under-provisioned pool.

Manually managed node pools

If the automatically provisioned node pools that OneFS creates are not appropriate for an environment, they can be manually reconfigured. This reconfiguration is done by creating a manual node pool and moving nodes from an existing node pool to the newly created one.

Note: The recommendation is to use the default, automatically provisioned node pools. Manually assigned pools may not provide the same performance and storage efficiency levels as automatically assigned pools. In particular, if your changes result in fewer than 20 nodes in the manual node pool.

Global Namespace Acceleration

Global Namespace Acceleration (GNA) is an unlicensed, configurable component of SmartPools. GNA's principal goal is to help accelerate metadata read operations (for example, filename lookups) by keeping a copy of the cluster's metadata on high-performance, low-latency SSD media. See Figure 10. GNA allows customers to increase the performance of certain workloads across the whole file system without having to purchase or upgrade SSDs for every node in the cluster. With GNA, an extra mirror of metadata from storage pools that do not contain SSDs is stored on SSDs available anywhere else in the cluster, regardless of node pool boundaries. The extra mirror accelerates metadata read operations even for data on node pools that have no SSDs.

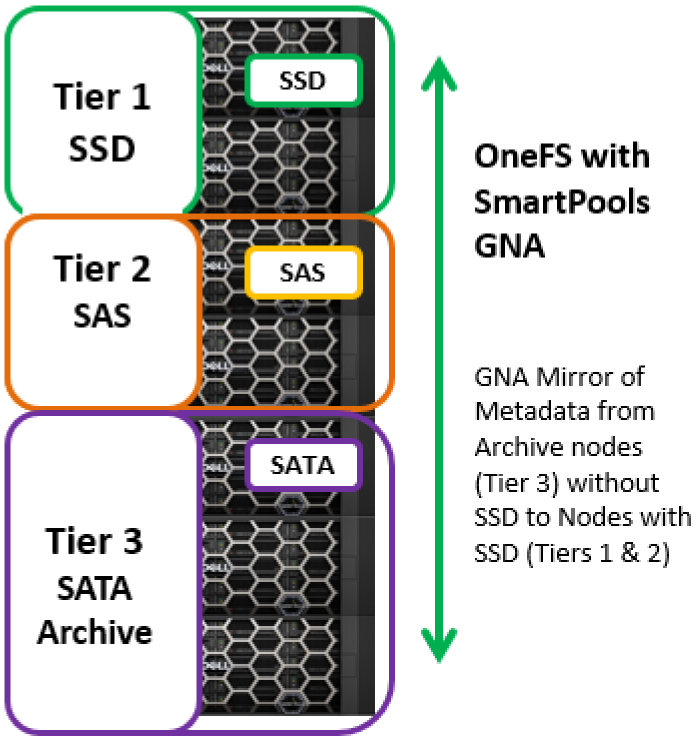

Tier 1
SSD
SSD
Tier 2
SAS
SAS
Tier 3
SATA
Archive
SATA
OneFS with SmartPools GNA
GNA Mirror of Metadata from Archive nodes (Tier 3) without SSD to Nodes with SSD (Tiers 1 & 2)

Figure 10. Global Name Space Acceleration

The purpose of GNA is to accelerate the performance of metadata-intensive applications and workloads such as home directories, workflows with heavy enumeration, and activities requiring many comparisons. Examples of metadata-read-heavy workflows exist across most of Dell's established and emerging scale-out NAS markets. In some, such as EDA, for example, such workloads are dominant, and the use of SSDs to provide the performance they require is ubiquitous.

Note: Since all the current PowerScale node platforms now contain some quantity of SSD, Global Namespace Acceleration is becoming more of a legacy feature.

Virtual Hot Spare

SmartPools Virtual Hot Spare (VHS) helps ensure that node pools maintain enough free space to successfully reprotect data if a drive fails. Though configured globally, VHS operates at the node pool level so that nodes with different size drives reserve the appropriate VHS space. This functionality helps ensure that, while data might move from one disk pool to another during repair, it remains on the same class of storage. VHS reservations are clusterwide and configurable as either a percentage of total storage (0 to

20 percent) or several virtual drives (one to four). The mechanism of this reservation is to allocate a fraction of the node pool's VHS space in each of the node pool's constituent disk pools.

No space is reserved for VHS on SSDs unless the entire node pool consists of SSDs, which means that a failed SSD could have data moved to HDDs during repair. However, without adding additional configuration settings, the alternative is reserving an unreasonable percentage of the SSD space in a node pool.

The default for new clusters is for VHS to have both **Reduce amount of available space** and **Deny new data writes** enabled with one virtual drive. On upgrade, existing settings are maintained. All customers are encouraged to enable VHS.

SmartDedupe and tiering

OneFS SmartDedupe 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 postprocess, 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 files themselves, and replaced with pointers to the shadow stores.

OneFS SmartDedupe does not deduplicate files that span SmartPools node pools or tiers, or that have different protection levels set. This feature avoids any performance or protection asymmetry that could occur if portions of a file live on different classes of storage. However, a deduplicated file that is moved to a pool with a different disk pool policy ID retains the shadow references to the shadow store on the original pool. Retaining these references breaks the rule for deduplication across different disk pool policies but is preferable to rehydrating files that are moved.

Further deduplication activity on that file can no longer reference any blocks in the original shadow store. The file must be deduplicated against other files in the same disk pool policy. If the file had not yet been deduplicated, the deduplication index might have knowledge about the file, and still think it is on the original pool. It will be discovered and corrected when a match is made against blocks in the file. Because the moved file has already been deduplicated, the deduplication index knows about the shadow store only. Because the shadow store has not moved, it will not cause problems for further matching. However, if the shadow store (but not both files) is moved as well, a similar situation occurs. The SmartDedupe job will discover it and purge knowledge of the shadow store from the deduplication index.

Note: Further information is available in the OneFS SmartDedupe white paper.

SmartPools licensing

The base SmartPools license enables:

- The ability to manually set individual files to a specific storage pool
- All aspects of file pool policy configuration
- Running the SmartPools job

The SmartPools job runs on a schedule and applies changes that have occurred through file pool policy configurations, file system activity, or the passage of time. The schedule is configurable through standard Job Engine means, defaulting to daily at 10 p.m.

The storage pool configuration of SmartPools requires no license. Drives are automatically provisioned into disk pools and node pools. Tiers can be created but have little utility because files are evenly allocated among the disk pools. The global SmartPools settings are all still available, with the caveat that spillover is forcibly enabled to `any` in the unlicensed case. Spillover has little meaning when any file can be stored anywhere.

The default file pool policy applies to all files. It can be used for protection policy and I/O optimization settings, but the disk pool policy cannot be changed. The SetProtectPlus job will run to enforce these settings when changes are made.

If a SmartPools license lapses, the disk pool policy set on files' inodes will be ignored and treated as the `ANY` disk pool policy. The disk pool policy is evaluated only when new disk pool targets are selected. So, writes to files with valid targets will continue to be directed to those targets based on the old disk pool policy. The writes will continue until a “reprotect” or higher level restripe causes the disk pool targets to be reevaluated. New files will inherit the disk pool policy of their parents (possibly through the New File Attributes mechanism). However, that disk pool policy will be ignored, and the disk pool targets will be selected according to the `ANY` policy.

When SmartPools is not licensed, any disk pool policy is ignored. Instead, the AutoBalance job spreads data evenly across node pools.

File pools

File pools make up the SmartPools logic layer. User-configurable file pool policies govern where data is placed, protected, and accessed, and how it moves among the node pools and tiers. Conceptually, file pools tiering is similar to storage information life cycle management (ILM) but do not involve file stubbing or other file system modifications. File pools allow data to be automatically moved from one type of storage to another within a single cluster to meet performance, space, cost, or other requirements. The data retains its data protection settings. For example, a file pool policy might dictate that anything written to path `/ifs/foo` goes to the H-Series nodes in node pool 1. Then, that data moves to the A-Series nodes in node pool 3 when the data is older than 30 days.

To simplify management, defaults are in place for node pool and file pool settings that handle basic data placement, movement, protection, and performance. All these settings can also be configured through the simple and intuitive WebUI, delivering deep granularity of control. Also provided are customizable template policies that are optimized for archiving, extra protection, performance, and VMware files.

When a SmartPools job runs, the data could be moved, undergo a protection or layout change, and so on. There are no stubs. Because the file system itself is doing the work, no transparency or data access risks apply.

Data movement is parallelized, using the resources of multiple nodes for speedy job completion. While a job is in progress, all data is available to users and applications.

The performance of different nodes can also be increased with the addition of system cache or solid state drives (SSDs). A OneFS cluster can use up to 181 TB of globally coherent cache. Within a file pool, SSD strategies can be configured to place a copy of that pool's metadata, or even some of its data, on SSDs in that pool.

Overall system performance impact can be configured to suit the peaks and lulls of an environment's workload. Change the time or frequency of any SmartPools job, and the quantity of resources allocated to SmartPools. For extremely high-utilization environments, a sample file pool policy can be used to match SmartPools run times to nonpeak computing hours. While resources required to run SmartPools jobs are low and the defaults work for most environments, that extra control can be beneficial when system resources are heavily used.

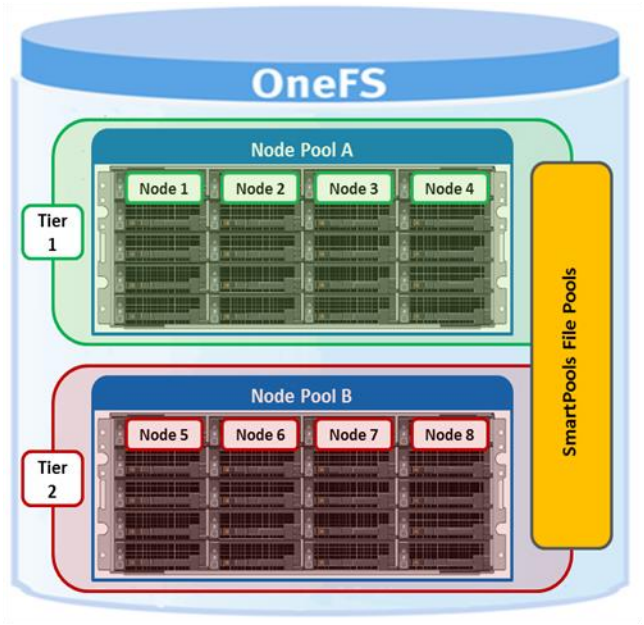


Figure 11. SmartPools file pool policy engine

File pool policies

SmartPools file pool policies can be used to broadly control the three principal attributes of a file, namely:

- Where a file resides
  - Tier
  - Node pool
- The file performance profile (I/O optimization setting)
  - Sequential
  - Concurrent
  - Random
  - SmartCache write caching
- The protection level of a file
  - Parity protected (+1n to +4n, +2d:1n, and so on)
  - Mirrored (2x – 8x)

A file pool policy is built on a file attribute the policy can match on. A file pool policy can use any of these attributes: File Name, Path, File Type, File Size, Modified Time, Create Time, Metadata Change Time, Access Time, or User Attributes.

Once the file attribute is set to select the appropriate files, the action to be taken on those files can be added. For example, if the attribute is **File Size**, additional settings are available to dictate thresholds (all files bigger than x, smaller than y). Next, actions are applied. Move to node pool x, set to y protection level, and lay out for z access setting.

Table 3. OneFS file pool file attributes

File attribute	Description
File Name	Specifies file criteria based on the file name
Path	Specifies file criteria based on where the file is stored
File Type	Specifies file criteria based on the file-system object type
File Size	Specifies file criteria based on the file size
Modified Time	Specifies file criteria based on when the file was last modified
Create Time	Specifies file criteria based on when the file was created
Metadata Change Time	Specifies file criteria based on when the file metadata was last modified
Access Time	Specifies file criteria based on when the file was last accessed
User Attributes	Specifies file criteria based on custom attributes (see Custom File Attributes)

AND and OR operators allow for the combination of criteria within a single policy for extremely data manipulation.

When a file is created in a directory with an associated path-based file pool policy, it is automatically written to the node pool specified in that policy. It does not require a SmartPools job to run. Files that match file pool policies based on attributes other than path are initially written to a default node pool. If necessary, the files are moved to the node pool specified in their matching file pool policy the next time a SmartPools job runs. This functionality ensures that write performance is not sacrificed for initial data placement.

As mentioned in Node pools, any data not covered by a file pool policy is moved to a tier that can be selected as a default for this purpose. If a disk pool has not been selected for this purpose, SmartPools defaults to the node pool with the most available capacity.

In practice, default file pool policies are almost always used because they can be so powerful. Most administrators do not want to set rules to govern all their data. They are mostly concerned about some or most of their data in terms of where it sits and how accessible it is. However, there is always data for which location in the cluster is going to be less important. For this data, there is the default policy, which is used for files for which none of the other policies in the list have applied. Typically, the default policy is set to optimize cost and to avoid using storage pools that are required for other data. For example, most default policies are at a lower protection level and use only the least expensive tier of storage.

When a file pool policy is created, SmartPools stores it in the OneFS configuration database with any other SmartPools policies. When a SmartPools job runs, it runs all the policies in order. If a file matches multiple policies, SmartPools applies only the first rule it fits.

For example, consider a scenario with the following two rules:

- One that moves all JPG files to an A-Series archive node pool
- Another that moves all files smaller than 2 MB to a performance tier

If the JPG rule appears first in the list, all JPG files smaller than 2 MB will go to nearline, *not* the performance tier. As mentioned, criteria can be combined within a single policy using AND or OR so that data can be classified granularly. To have all JPG files larger than 2 MB moved to the archive node pool, you can construct the file pool policy with an AND operator to cover that condition.

Policy order and policies themselves can be easily changed at any time. Policies can be added, deleted, edited, copied, and reordered.

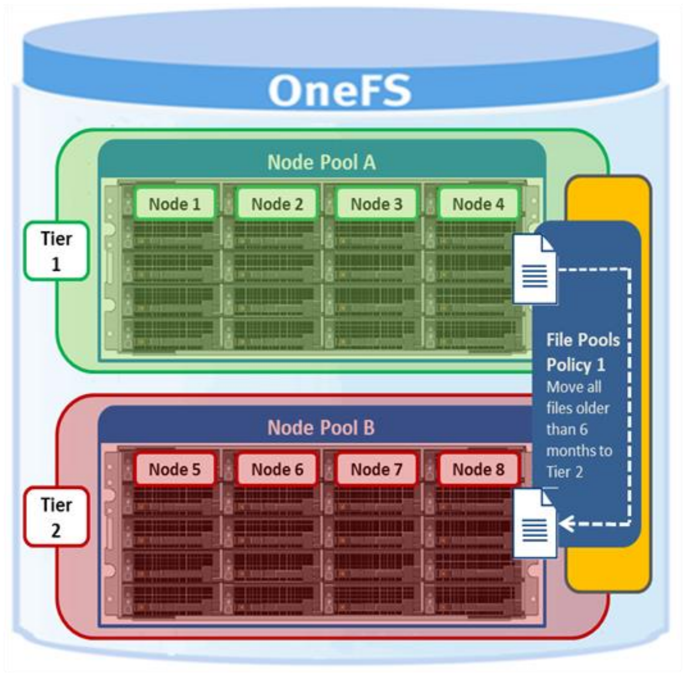
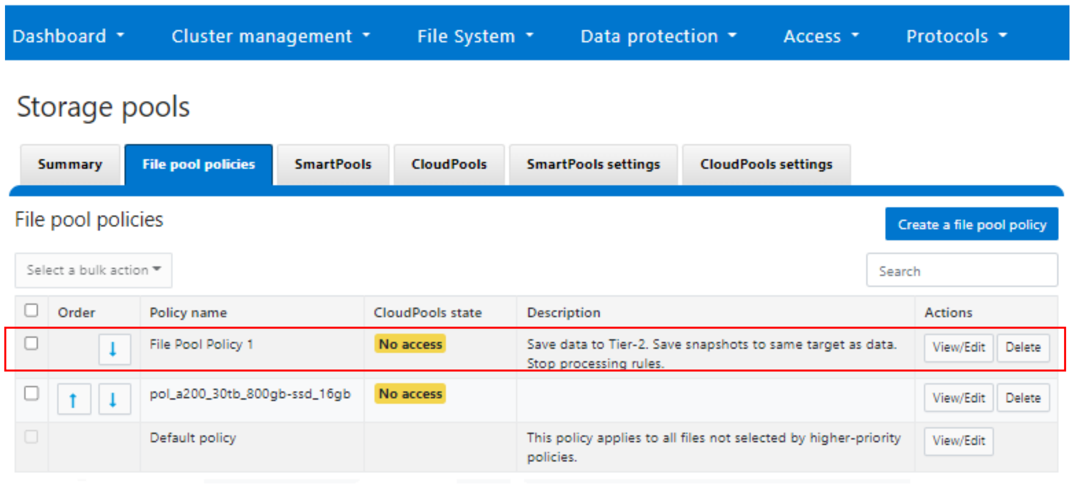


Figure 12. SmartPools file pool policy governing data movement



☐	Order	Policy name	CloudPools state	Description	Actions
☐	↓	File Pool Policy 1	No access	Save data to Tier-2. Save snapshots to same target as data. Stop processing rules.	View/Edit Delete
☐	↑ ↓	pol_a200_30tb_800gb-ssd_16gb	No access		View/Edit Delete
☐		Default policy		This policy applies to all files not selected by higher-priority policies.	View/Edit

Figure 13. File pool policy example

Figure 12 and Figure 13 illustrate a common file pools use case. An organization wants:

- Active data saved on their performance nodes in Tier 1 (SAS + SSD)
- Any data not accessed for 6 months moved to the cost optimized archive Tier 2

As the list of file pool policies grows, it becomes less practical to manually go through all of them to see how a file will behave when policies are applied. SmartPools supports up to 128 policies. SmartPools also has some advanced options that are available on the command line for this kind of scenario testing and troubleshooting.

File pool policies can be created, copied, modified, prioritized, or removed at any time. Also provided are sample policies that can be used as is or as templates for customization.

Custom File Attributes

Custom File Attributes, or user attributes, can be used when more granular control is required than can be achieved using the standard file attributes options. Those options include File Name, Path, File Type, File Size, Modified Time, Create Time, Metadata Change Time, and Access Time. User Attributes use key value pairs to tag files with additional identifying criteria that SmartPools can then use to apply file pool policies. You can set file attributes by running the `setextattr` command.

Custom File Attributes are most often used to designate ownership or create project affinities. Consider a biosciences user expecting to access many genomic sequencing files when personnel arrive at the lab in the morning. That user might employ Custom File Attributes to ensure that the files are migrated to the fastest available storage.

Once Custom File Attributes are set, SmartPools uses them to specify location, protection, and performance access for a matching group of files. SmartPools uses Custom File Attributes exactly as File Name, File Type, or any other file attributes are used. Unlike other SmartPools file pool policies, Custom File Attributes can be set from the command line or platform API.

Anatomy of a SmartPools job

When a SmartPools job runs, SmartPools examines all file attributes and checks them against the list of SmartPools policies. To maximize efficiency, wherever possible, the

SmartPools job uses an efficient, parallel metadata scan (logical inode, or LIN, tree scan) instead of a more expensive directory and file tree walk. This scan is even more efficient when the SmartPools SSD metadata acceleration strategy is deployed.

A SmartPools LIN tree scan breaks up the metadata into ranges for each node to work on in parallel. Each node can then dedicate a single, or multiple, threads to perform the scan on their assigned range. A LIN tree walk also ensures each file is opened only once. This approach is more efficient when compared to a directory walk where hard links and other constructs can result in single threading, multiple opens, and so on.

When the SmartPools file pool policy engine finds a match between a file and a policy, it stops processing policies for that file. The first policy match determines what will happen to that file. Next, SmartPools checks the file’s current settings against those settings that the policy would assign to identify the settings that do not match. Once SmartPools has the complete list of settings to be applied to that file, it sets them all simultaneously. It then moves to restripe that file to reflect changes to node pool, protection, SmartCache use, layout, and so on.

File pool policy engine

The Job Engine controls and manages the SmartPools file pool policy engine. The default schedule for the file pool policy engine process is every day at 10 p.m., and with a low impact policy. The schedule, priority, and impact policy can be manually configured and tailored to a particular environment and workload.

You can also run the engine on-demand using a separate invocation to apply the appropriate file-pool membership settings to an individual file or subdirectory. This approach avoids waiting for the background scan to do it.

To test how a new policy will affect file dispositions, you can run a SmartPools job on a subset of the data. That data can be either a single file or directory, or a group of files or directories. The job can either be run live, to make the policy changes, or in a Dry Run mode to estimate the scope and effect of a policy. Thus, you can simulate the end state to see how the set of file pool policies in place affect each file.

Running a SmartPools job against a directory or group of directories is available as a command-line option. The following CLI syntax runs the engine, on demand, for specific files or subtrees:

```
isi filepool apply [-r] <filename>
```

For a dry-run assessment that calculates and reports without making changes, run:

```
isi filepool apply –nv [PATH]
```

For a particular file pool policy, the command returns the following information:

- Policy number
- Files matched
- Directories matched
- ADS containers matched
- ADS streams matched
- Access changes skipped

- Protection changes skipped
- File creation templates matched
- File data placed on HDDs
- File data placed on SSDs

Note: When you use the `isi filepool apply` CLI utility, SmartPools traverses the specified directory tree instead of performing a LIN scan. It does so because the operation has been specified at the directory level. Also, the command is synchronous, and so will wait for completion before returning the command prompt.

FilePolicy job

Traditionally, OneFS has used the SmartPools jobs to apply its file pool policies. To apply the policies, the SmartPools job goes to every file, and the SmartPoolsTree job goes to a tree of files. However, the scanning portion of these jobs can result in significant random impact to the cluster and lengthy execution times, particularly in a SmartPools job.

To address this issue, the FilePolicy job provides a faster, lower-impact method for applying file pool policies than the full-blown SmartPools job. Along with the IndexUpdate job, FilePolicy improves job scan performance by using a file system index, or changelist—rather than a full tree scan—to find files requiring policy changes.

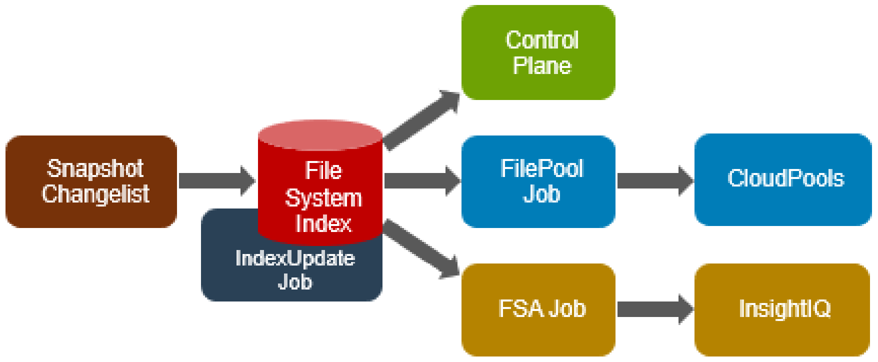


Figure 14. File system index, jobs, and consumers

Avoiding a full tree walk dramatically decreases the amount of locking and metadata scanning work the job is required to perform. This approach reduces impact on CPU and disk, at the expense of not doing everything that SmartPools does. The FilePolicy job enforces only the SmartPools file pool policies, as opposed to the storage pool settings. For example, FilePolicy does not deal with changes to storage pools or storage pool settings, such as:

- Restriping activity due to adding, removing, or reorganizing node pools
- Changes to storage pool settings or defaults, including protection

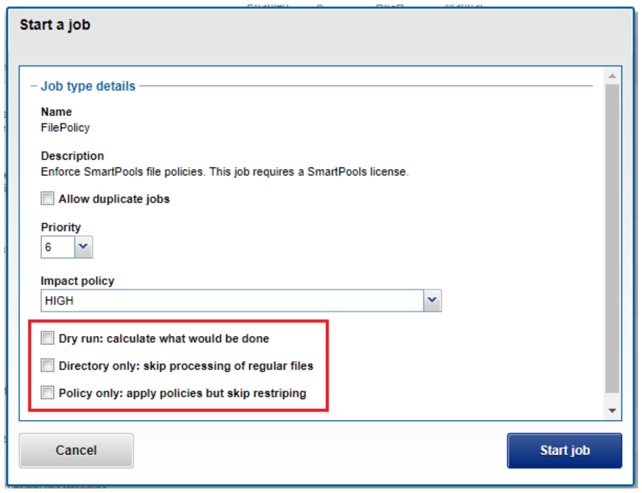
However, SmartPools and FilePolicy usually perform the same work. You can run the FilePolicy job in one of four modes:

Table 4. FilePolicy job execution modes

FilePolicy mode	Description
Default	Enacts full policy changes on file and directories

FilePolicy mode	Description
Dry Run	Calculates and reports on the policy changes that would be made
Directory Only	Process policies for directories but skips the files themselves
Policy Only	Applies the matching FilePool policies but skips the data restriping

Select the FilePolicy job mode in the WebUI by going to **Cluster Management** > **Job Operations** > **File Policy** > **View/Edit**.



Name
FilePolicy

Description
Enforce SmartPools file policies. This job requires a SmartPools license.

☐ Allow duplicate jobs

Priority
6

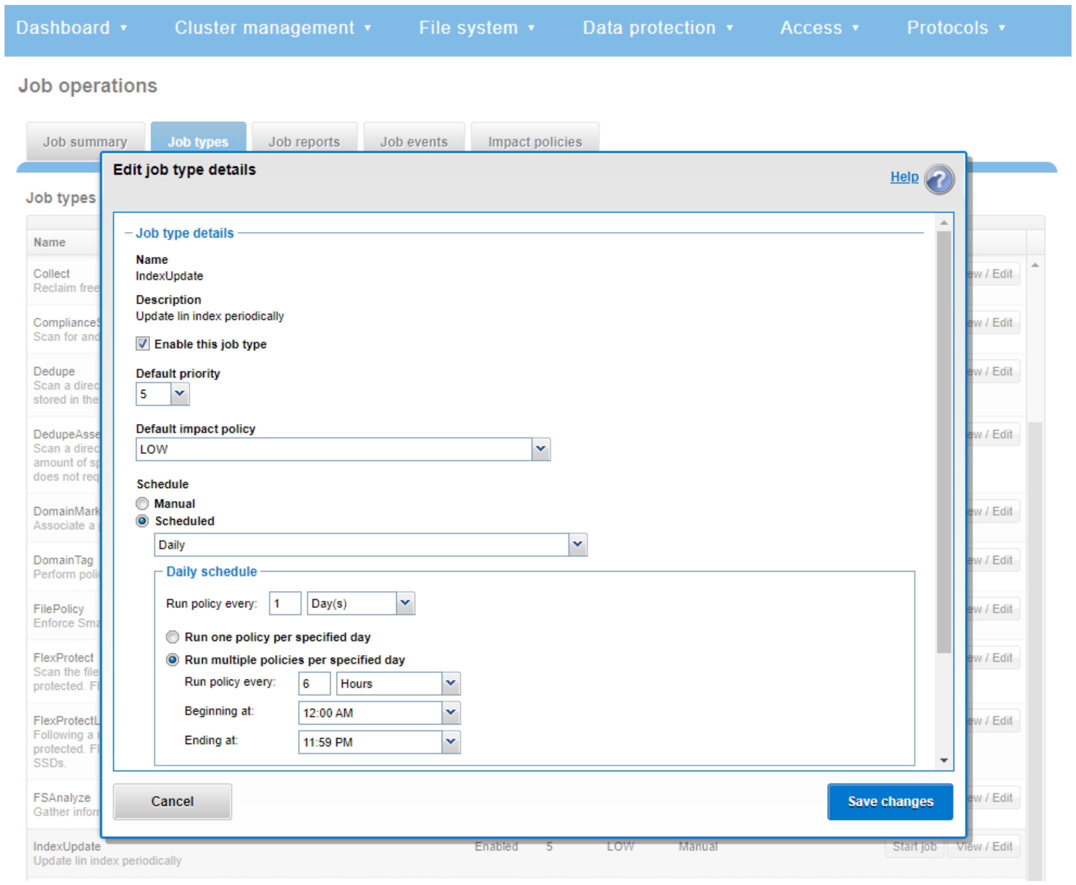
Impact policy
HIGH

☐ Dry run: calculate what would be done
☐ Directory only: skip processing of regular files
☐ Policy only: apply policies but skip restriping

Figure 15. FilePolicy job mode selection

Disabled by default, FilePolicy supports the full range of file pool policy features, reports the same information, and provides the same configuration options as the SmartPools job. Because FilePolicy is a changelist-based job, it performs best when run frequently. For example, once or multiple times a day, depending on the configured file pool policies, data size, and rate of change.

Configure job schedules in the WebUI by going to **Cluster Management** > **Job Operations**, highlighting the job, and selecting **View\Edit**. The following example illustrates configuring the IndexUpdate job to run every 6 hours at a LOW impact level with a priority value of 5:



Edit job type details

Job type details

Name: IndexUpdate

Description: Update lin index periodically

[✓] Enable this job type

Default priority: 5

Default impact policy: LOW

Schedule: Manual [✓ Scheduled]

Daily

Daily schedule

Run policy every: 1 Day(s)

Run one policy per specified day [✓ Run multiple policies per specified day]

Run policy every: 6 Hours

Beginning at: 12:00 AM

Ending at: 11:59 PM

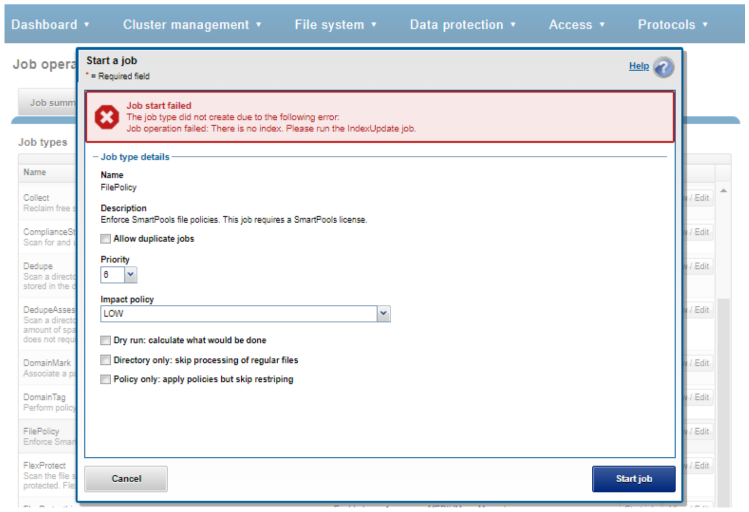
IndexUpdate — Update lin index periodically — Enabled — 5 — LOW — Manual — Start job — View / Edit

Figure 16. IndexUpdate job schedule configuration through the OneFS WebUI

When you are enabling and using the FilePolicy and IndexUpdate jobs, the recommendation is to continue running the SmartPools job as well, but at a reduced frequency (monthly).

Note: In addition to running on a configured schedule, the FilePolicy job can also be performed manually.

FilePolicy requires access to a current index. If the IndexUpdate job has not yet been run, an attempt to start the FilePolicy job will fail with an error, as shown in Figure 17. The error message will provide an instruction to run the IndexUpdate job first. Once the index has been created, the FilePolicy job will run successfully. The IndexUpdate job can be run several times daily (for example, every 6 hours) to keep the index current and prevent the snapshots from getting large.



Start a job
* = Required field

Job start failed
The job type did not create due to the following error:
Job operation failed: There is no index. Please run the IndexUpdate job.

Job type details

Name
FilePolicy

Description
Enforce SmartPools file policies. This job requires a SmartPools license.

☐ Allow duplicate jobs

Priority
6

Impact policy
LOW

☐ Dry run: calculate what would be done
☐ Directory only: skip processing of regular files
☐ Policy only: apply policies but skip restriping

Figure 17. FilePolicy job error message with prompt to run IndexUpdate first

Consider using the FilePolicy job with the job schedules shown in Table 5 for workflows and datasets with the following characteristics:

- Data has long retention times.
- There are many small files.
- Path-based file pool filters are configured
- FSAnalyze job is already running on the cluster (InsightIQ monitored clusters).
- A SnapshotIQ schedule is already configured.
- The SmartPools job typically takes a day or more to run to completion at LOW impact.

Note: For clusters without these characteristics, the recommendation is to continue running the SmartPools job as usual and to not activate the FilePolicy job.

The following table provides a suggested job schedule when deploying FilePolicy:

Table 5. Suggested job schedule, impact, and priority configuration when activating the FilePolicy job

Job	Schedule	Impact	Priority
FilePolicy	Every day at 22:00	LOW	6
IndexUpdate	Every 6 hours, every day	LOW	5
SmartPools	Monthly – Sunday at 23:00	LOW	6

Note: Because no two clusters are the same, this suggested job schedule might require additional tuning to meet the needs of a specific environment.

Note: For more information about job configuration, see the OneFS Job Engine white paper.

Data location

The file system explorer provides a detailed view of where SmartPools managed data is at any time. This view includes both the node pool location and the file pool policy-dictated location. The policy-dictated location is where that file will move after the next successful completion of the SmartPools job.

When data is written to the cluster, SmartPools writes it to a single node pool only. In almost all cases, a file exists in its entirety within a node pool, and not across node pools. SmartPools determines which pool to write to, based on one of two scenarios: If a file matches a file pool policy based on directory path, that file is immediately written into the node pool dictated by the file pool policy. If that file matches a file pool policy that is based on any other criteria besides path name, it will be moved when the next scheduled SmartPools job runs. If the file does not match a specific file pool policy, it is written to a pool based on the default policy.

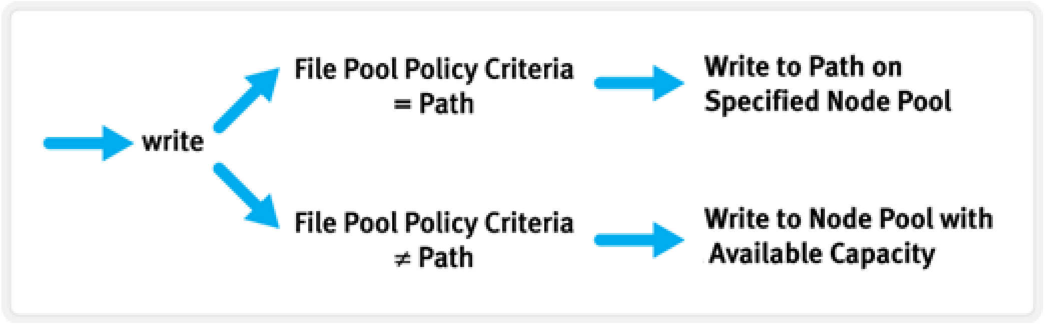

write
File Pool Policy Criteria = Path
Write to Path on Specified Node Pool
File Pool Policy Criteria ≠ Path
Write to Node Pool with Available Capacity

Figure 18. File pools and node pools

For performance, charge back, ownership, or security purposes, it is sometimes important to know exactly where a specific file or group of files is on disk at any given time. Any file in a SmartPools environment typically exists entirely in one storage pool. There are exceptions when a single file might be split, usually on a temporary basis, across two or more node pools at one time.

Node pool affinity

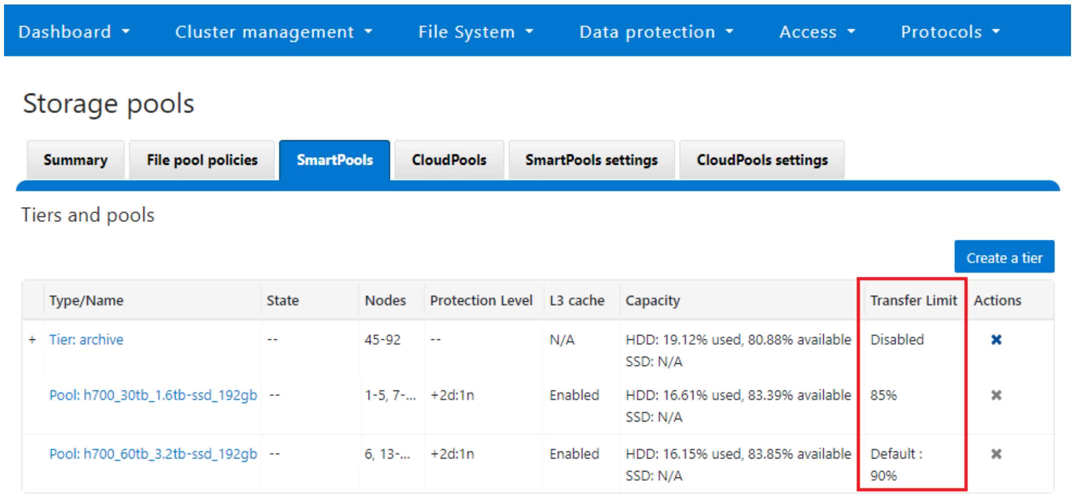
SmartPools generally allows a file to reside in only one node pool, but a file might temporarily span several node pools in some situations. A file pool policy may dictate that a file move from one node pool to another. In that case the file will exist partially on the source node pool and partially on the destination node pool until the move is complete. If the node pool configuration is changed, a file might be split across those two new pools until the next scheduled SmartPools job runs. For example, when splitting a node pool into two node pools. Consider a situation where a node pool fills up, and data spills over to another node pool so the cluster can continue accepting writes. In that case a file might be spilled over the intended node pool and the default spillover node pool. The last circumstance under which a file might span more than one node pool is for typical restriping activities such as cross-node pool rebalances or rebuilds.

Transfer Limits

SmartPools transfer limits, introduced in OneFS 9.5, govern storage pool usage beyond which file pool policies will not attempt to move files to a target. The principal benefits of transfer limits include reliability, by preventing undesirable spillover actions, and performance, by avoiding unnecessary work.

The default transfer limit in OneFS 9.5 and later releases is 90%, and it applies to any of a cluster's storage pools where a limit is not explicitly set. However, this limit only applies when SmartPools is licensed on a cluster and jobs are run that apply file pool policies, namely SmartPools, SmartPoolsTree, and FilePolicy.

Transfer limits can be specified as a percentage per tier, per node pool, or disabled. You can configure transfer limits from the OneFS WebUI by going to **File system > Storage pools > SmartPools** and selecting the resource you want.



	Type/Name	State	Nodes	Protection Level	L3 cache	Capacity	Transfer Limit	Actions
+	Tier: archive	--	45-92	--	N/A	HDD: 19.12% used, 80.88% available SSD: N/A	Disabled	✖
	Pool: h700_30tb_1.6tb-ssd_192gb	--	1-5, 7-...	+2d:1n	Enabled	HDD: 16.61% used, 83.39% available SSD: N/A	85%	✖
	Pool: h700_60tb_3.2tb-ssd_192gb	--	6, 13-...	+2d:1n	Enabled	HDD: 16.15% used, 83.85% available SSD: N/A	Default : 90%	✖

Figure 19. Tier and Node Pools Transfer Limits Configuration.

Note: If transfer limits are configured per-tier, all the constituent node pools within that tier will share the same limit value.

In OneFS 9.5 and later releases, the Storage Pools Summary WebUI also displays a histogram which includes the transfer limit thresholds and indicates any limit violations.

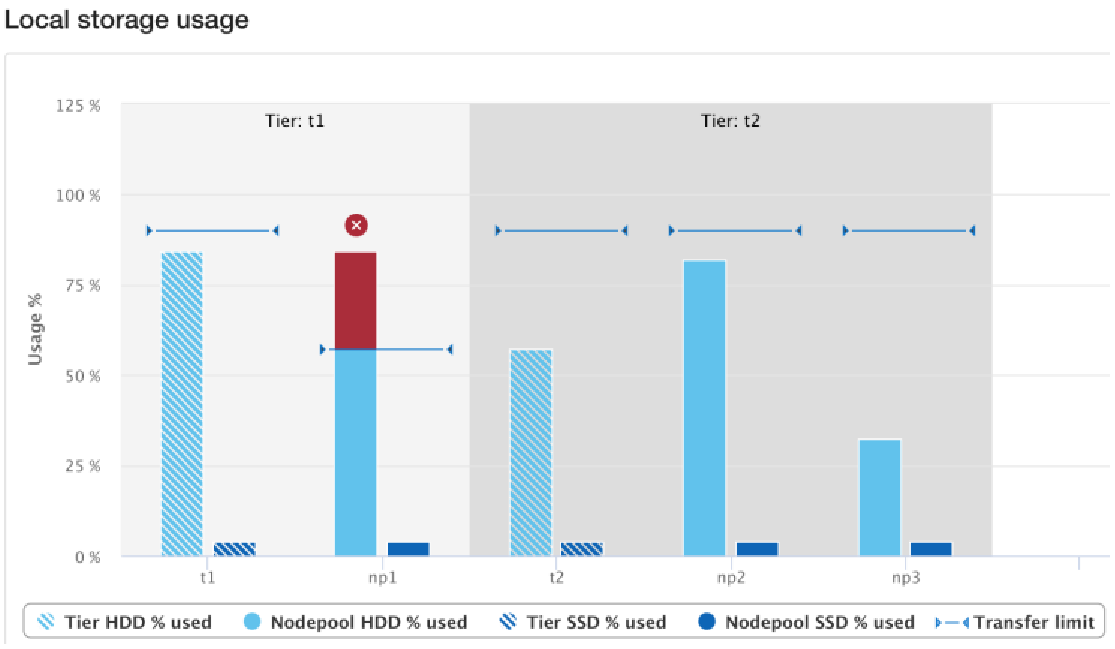


Figure 20. WebUI Storage Usage View Showing Transfer Limits.

Using SmartPools to improve performance

One of the principal goals of storage tiering is to reduce data storage costs without compromising data protection or access. SmartPools can be used to improve performance in several ways:

- Location-based performance improvements
- Performance settings
- SSD strategies
- Performance isolation

Location-based performance uses SmartPools file pool policies to classify and direct data to the most appropriate media (SSD, SAS, or SATA) for its performance requirements.

In addition, SmartPools file pool rules also allow data to be optimized for both performance and protection.

As we have seen, SSDs can also be employed in various ways, accelerating combinations of data, metadata read and write, and metadata read performance on other tiers.

Another application of location-based performance is for performance isolation goals. Using SmartPools, a specific node pool can be isolated from all but the highest performance data. File pool policies can be used to direct all but the most critical data away from this node pool. This approach is sometimes used to isolate only a few nodes of a certain type out of the cluster for intense work. Because node pools are easily configurable, a larger node pool can be split, and one of the resulting node pools isolated and used to meet a temporary requirement. Then, split pools can be reconfigured back into a larger node pool.

For example, the administrators of this cluster are meeting a temporary need to provide higher performance support for a specific application for a limited time period. They have split their highest performance tier and set policies to migrate all data not related to their project to other node pools. The servers running their critical application have direct access to the isolated node pool. A default policy has been set to ensure all other data in the environment is not placed on the isolated node pool. In this way, the new node pool is completely available to the critical application.

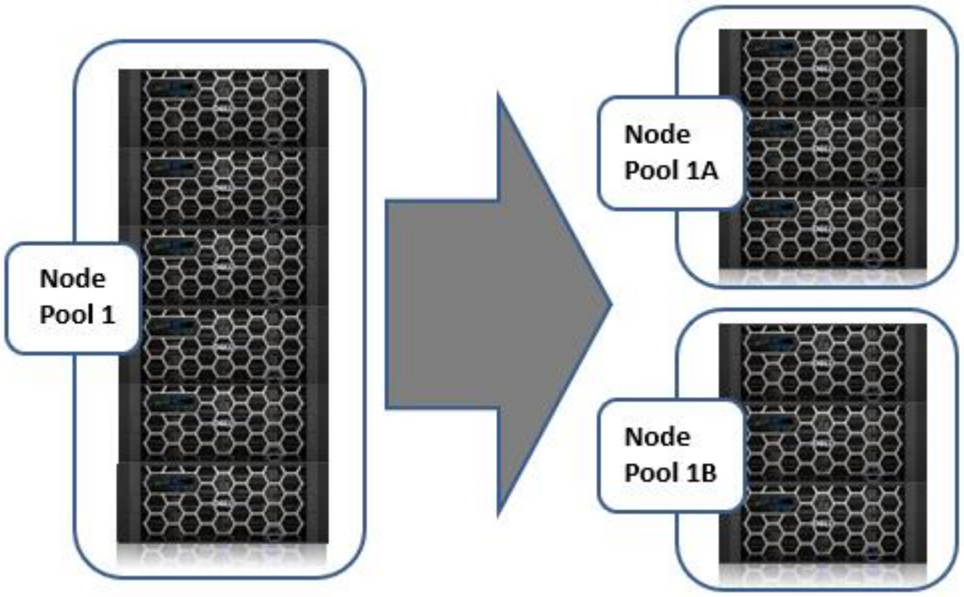


Figure 21. Performance isolation example

Data access settings

At the file pool (or even the single file) level, data access settings can be configured to optimize data access for the type of application accessing it. Data can be optimized for concurrent, streaming, or random access. Each of these settings changes how data is laid out on disk and how it is cached.

Table 6. SmartPools data access settings

Data access setting	Description	On disk layout	Caching
Concurrency	Optimizes for current load on the cluster, featuring many simultaneous clients. This setting provides the best behavior for mixed workloads.	Stripes data across the minimum number of drives required to achieve the data protection setting configured for the file.	Moderate prefetching
Streaming	Optimizes for high-speed streaming of a single file. For example, to enable rapid reading with a single client.	Stripes data across a larger number of drives.	Aggressive prefetching
Random	Optimizes for unpredictable access to the file by performing almost no cache prefetching.	Stripes data across the minimum number of drives required to achieve the data protection setting configured for the file.	Little to no prefetching

As the settings indicate, the **Random** data access setting performs little to no read-cache prefetching to avoid wasted disk access. This setting works best for small files (< 128 KB) and large files with random small block accesses. Streaming access works best for sequentially read medium to large files. This access pattern uses aggressive prefetching to improve overall read throughput, and on disk layout it spreads the file across many disks to optimize access. Concurrency (the default setting for all file data) access is the middle ground with moderate prefetching. Use this setting for file sets with both random and sequential access.

Leveraging SSDs for metadata and data performance

Adding SSDs within a node pool can boost performance significantly for many workloads. In the OneFS architecture, SSDs can be used to accelerate performance across the entire cluster using SSD strategies for data or metadata acceleration.

SmartPools offers several SSD strategies to choose from, including:

- **Metadata read acceleration**—Creates a preferred mirror of file metadata on SSDs and writes the rest of the metadata, plus all the file data, to HDDs.
- **Metadata read/write acceleration**—Creates all the mirrors of a file's metadata on SSDs.
- **Avoid SSDs**—Never uses SSDs; writes all associated file data and metadata to HDDs only. This strategy is used when insufficient SSD storage capacity exists and you want to prioritize its utilization.
- **Data on SSDs**—A node pool's entire data and metadata resides on SSDs.
- **SSDs for L3 cache**—A node pool's entire set of SSDs are used for L3, or SmartFlash, read caching.

To configure a different SSD strategy on any node pools that are not specified as a storage target in the default policy, create a user-defined policy for each pool.

Note: SSD strategies and GNA work in concert. For example, if a cluster's entire data is set to **Avoid SSDs** strategy, enabling GNA has no effect, and the SSDs will not be used. If the files are all set to **Metadata read** with GNA disabled, only the files on node pools with SSDs will get read acceleration. If GNA is enabled, all files will get read acceleration.

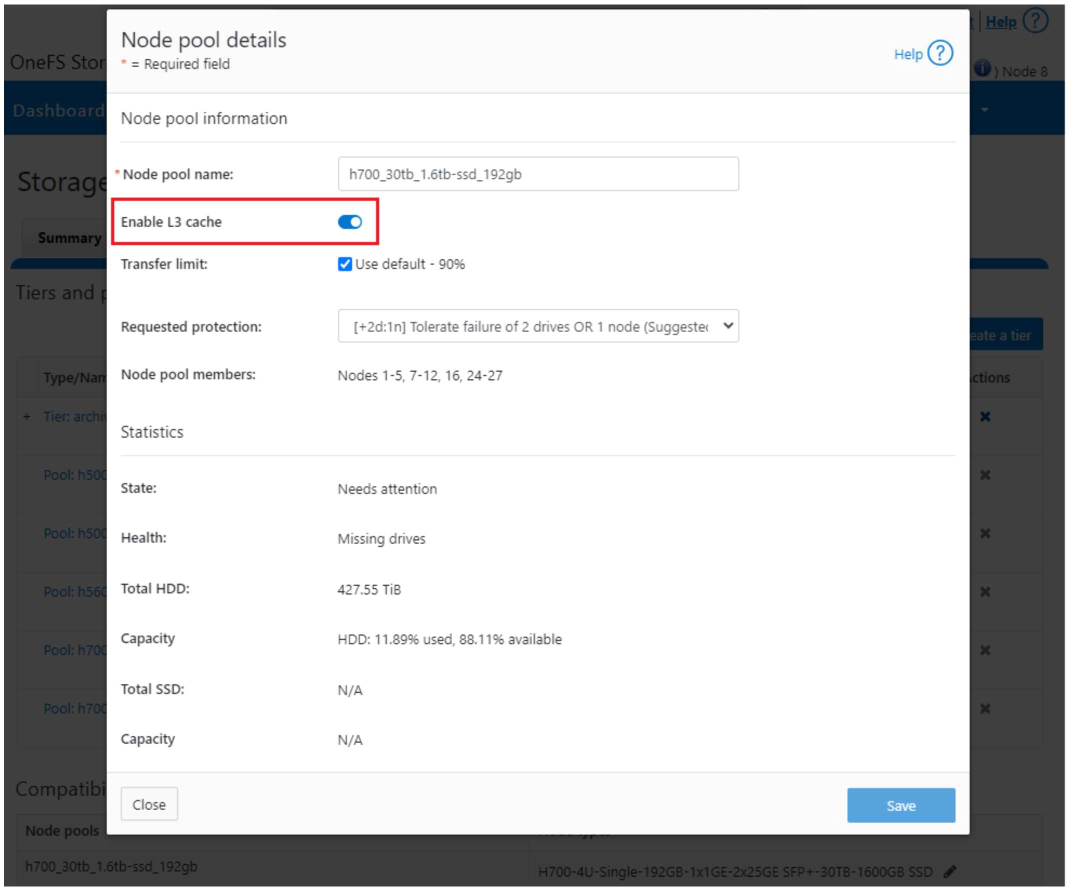
A common setting for mixed clusters, where not all nodes contain SSDs, is often the **Metadata read** SSD strategy for *all* files and GNA enabled. A SmartPools license is required to be able to direct different files to different tiers of storage. The SSD strategy is globally set because it is set to the same value on all files even though the storage pools for different file pools are set differently.

Note: Ensure that all F-Series node pools are configured with the SSD strategy of **Meta +Data**. If they are configured for another SSD strategy, OneFS will not automatically use them.

Enabling L3 cache (SmartFlash)

L3 cache is enabled per node pool through a simple on or off configuration setting. When enabled, L3 cache consumes all the SSDs in a node pool. Also, L3 cache cannot co-exist with other SSD strategies, except for Global Namespace Acceleration. L3 cache node pool SSDs cannot participate in GNA.

Note: L3 cache is enabled by default on any new node pool containing SSDs.



Node pool details

* = Required field

Node pool information

*Node pool name: h700_30tb_1.6tb-ssd_192gb

Enable L3 cache: on

Transfer limit: ✓ Use default - 90%

Requested protection: [+2d:1n] Tolerate failure of 2 drives OR 1 node (Suggeste

Node pool members: Nodes 1-5, 7-12, 16, 24-27

Statistics

State: Needs attention

Health: Missing drives

Total HDD: 427.55 TiB

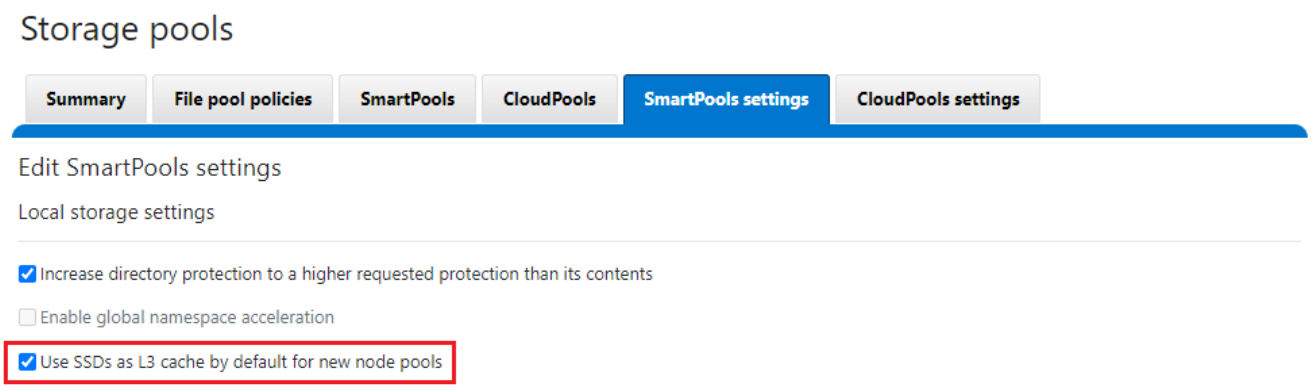
Capacity: HDD: 11.89% used, 88.11% available

Total SSD: N/A

Capacity: N/A

Figure 22. Enabling L3 caching on a node pool containing SSDs

The **SmartPools settings** tab, under **Storage pools** in the WebUI, also enables specifying whether L3 cache is the default configuration for any new node pools that contain SSDs.



Storage pools

Edit SmartPools settings

Local storage settings

✓ Increase directory protection to a higher requested protection than its contents

☐ Enable global namespace acceleration

✓ Use SSDs as L3 cache by default for new node pools

Figure 23. Configuring L3 caching by default on new node pools

When converting the SSDs in a particular node pool to use L3 cache rather than SmartPools, you can see conversion progress by periodically tracking SSD space usage (used capacity) during the process. The Flexprotect_Plus or SmartPools job is responsible for the L3 cache conversion. The job impact policy of the Flexprotect_Plus or SmartPools job can be reprioritized to run faster or slower. Doing so will increase or decrease the impact of the conversion process on cluster resources.

Note: For more information, see the OneFS SmartFlash white paper.

Minimizing the performance impact of tiered data

Despite the performance enhancements that SmartPools can provide, any tiering approach will have some cases where performance can suffer. These degradations are the result of data location, tiering data movement activity, and so on.

Previously we discussed data location and performance isolation as performance enhancement functions of SmartPools. As expected, if data is on slower drive media, it will typically be slower to access.

Another architectural attribute of scale-out NAS is that data might reside on one node but the network access point for clients accessing that data may be on, or load balanced to, another class of node. This access node might have different front-end IO capabilities and cache, but the class of storage on which the data resides primarily govern the performance characteristics.

While SAS is usually faster than SATA, spindle counts can have a significant impact as well. The most important performance impacts in a SmartPools environment are the characteristics of the nodes in the node pool the application connects to, rather than what media type the data physically resides on. For example, assume that no bottleneck exists on the network side. A streaming application will experience little difference in performance between data on a SATA node with less CPU power, and data on a SAS node with more CPU power. This scenario is true if the application’s connection into the cluster is through a node pool where the nodes have a great deal of cache. A similar scenario exists for applications with random access patterns. If the node pool they connect into has sufficient CPUs, data location by spinning media type does not usually make a significant performance difference.

Another important performance consideration in a tiered storage environment is the effect of data movement itself on overall system resources. The action of moving data from one node pool to another uses system resources. SmartPools mitigates the impact of data movement in multiple ways. Architecturally, through the highly efficient Job Engine, and through impact policies. Impact policies are configurable by the end user to control the amount of system resources allocated to data movement and when data movement takes place. The Job Engine impact policies are:

- **Paused**—Do nothing and wait.
- **Low**—Use 10 percent or less of Job Engine allocated resources.
- **Medium**—Use 30 percent or less of Job Engine allocated resources.
- **High**—Use maximum amount of Job Engine allocated resources.

Impact policy can be set once for SmartPools jobs. A file pool policy that can change the impact settings to match the typical peaks and lulls of the workload in a particular environment can also control impact policy.

SmartPools best practices

For optimal cluster performance, observe the following OneFS SmartPools best practices:

- Define a performance and protection profile for each tier and configure each tier accordingly.

- Enable SmartPools Virtual Hot Spare with a minimum of 10 percent space allocation.
- Avoid creating hard links to files, which will cause the file to match different file pool policies.
- If node pools are combined into tiers, target the file pool rules to the tiers rather than specific node pools within the tiers.
- Avoid creating tiers that combine node pools both with and without SSDs.
- Ensure that SSDs consist of a minimum of 2 percent of the total cluster usable capacity, spread across at least 20 percent of the nodes, before enabling GNA.
- Determine if metadata operations for a particular workload are biased towards reads, writes, or an even mix, and select the optimal SmartPools metadata strategy.
- Ensure that node pool and cluster capacity utilization (hard drive and SSD) remains below 90 percent.
- When enabling and using the FilePolicy job, continue running the SmartPools job at a reduced frequency.
- When employing a deep archiving strategy, ensure that the performance pool is optimized for all directories and metadata, and the archive tier for cold files as they age. You can perform this configuration by adding a TYPE=FILE statement to the aging file pool policy rules to only move files to the archive tier.
- If SmartPools takes more than a day to run, or the cluster is already running the FSAnalyze job, consider scheduling the FilePolicy (and corresponding IndexUpdate job) to run daily. Also reduce the frequency of the SmartPools job to monthly.

SmartPools use cases

OneFS SmartPools is typically used in the following ways:

Traditional file tiering

- Store "current" data on the faster nodes and "old" data on cost-optimized storage.
- More flexibility in choosing which files reside where.
- Faster movement of files.

Metadata acceleration

- Use SSD nodes to accelerate metadata operations (for example, searches and indexing) on archive data stored on archive nodes.

Application partitioning

- Store files from different users, groups, or projects on separate hardware.

Equivalent node support

- Add or replace a new style of node to a pool of near identical previous generation nodes.

Auto-provisioning and large cluster support

- Large clusters are automatically provisioned into disk pools to enhance reliability.

- Painless to add nodes.
- Appropriate protection is configured automatically.

SmartPools workflow examples

Example A: Storage cost efficiency in media post-production

In media post-production, each phase typically deals with massive quantities of large files. Editing and visual effects work is storage-resource-intensive and, when complete, the products are archived as reference work and retained for long time periods. In the last five years alone, with the move to 3D rendering and high definition formats, the amount of storage required for post-production has increased up to ten-fold. Finished projects have doubled or tripled in size. Post-production facilities are under increased pressure to provide faster turn times, richer effects, and an ever-greater number of post-release projects to market tie-ins and licensed products and derivatives.

Post-production facilities create a massive amount of data as their main product. This data can grow to hundreds of terabytes, and often multiple petabytes. Many people and processes need to access the data quickly while a project is active. Then it continues its rich commercial life within the company, being drawn on intermittently for follow-on projects over long time periods. Thus, post-production data is critical and timely, and then dormant until it is required again, and the process is repeated.

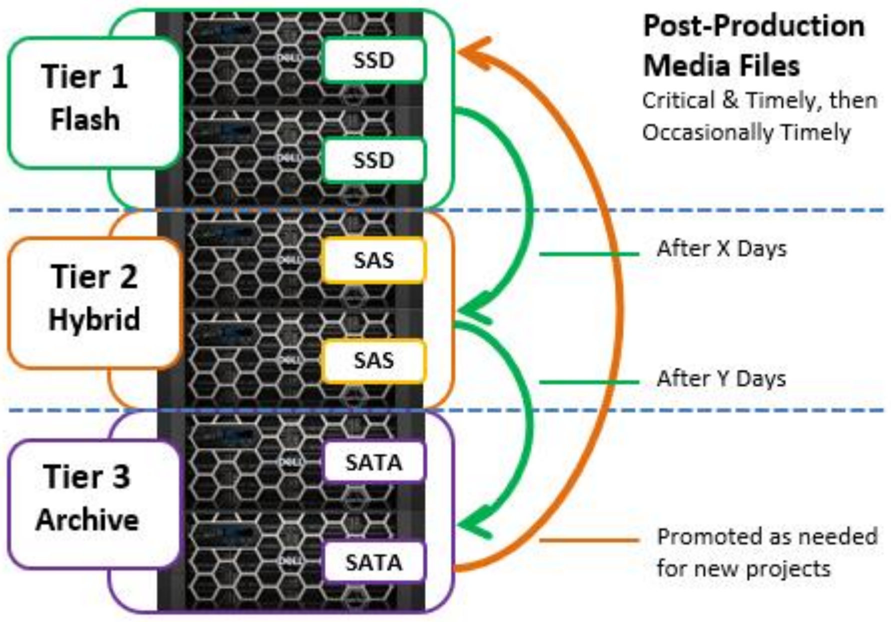


Figure 24. Media post-production example

It is increasingly popular to have a two or three-tiered solution where working data is on high-performance disk. All the rest is on lower performance storage. All the working data is set for streaming access and protected at a high level. This way, 60 to 95 percent of all data under management can reside on less expensive disk, but it is all accessible at reasonable performance rates. Furthermore, this archived data can be promoted to the fastest disk at any point and is protected for long time periods.

Example B: Data availability and protection in semiconductor design

Logical and physical design and verification are complex early stages in the electronic design automation (EDA) process for semiconductor production. This design data is critical. Time-to-market is paramount in this industry. The company is counting on the revenue from this product, and the workflow is high priority. Multiple engineers from around the globe will typically be collaborating on the project, and all require timely access to the data.

Design data and other intellectual property are often reused in subsequent projects, so archive and retention requirements spanning decades are commonplace. Moreover, it can remain critical for many years, even decades. But it is not timely—nobody has to access older designs at high-performance speeds. All instances in which historical design data is referenced are not time-critical enough to change the type of disk they are stored on. However, an argument could be made against deep archive in legal discovery timeframes. Older designs are critical, but not timely.

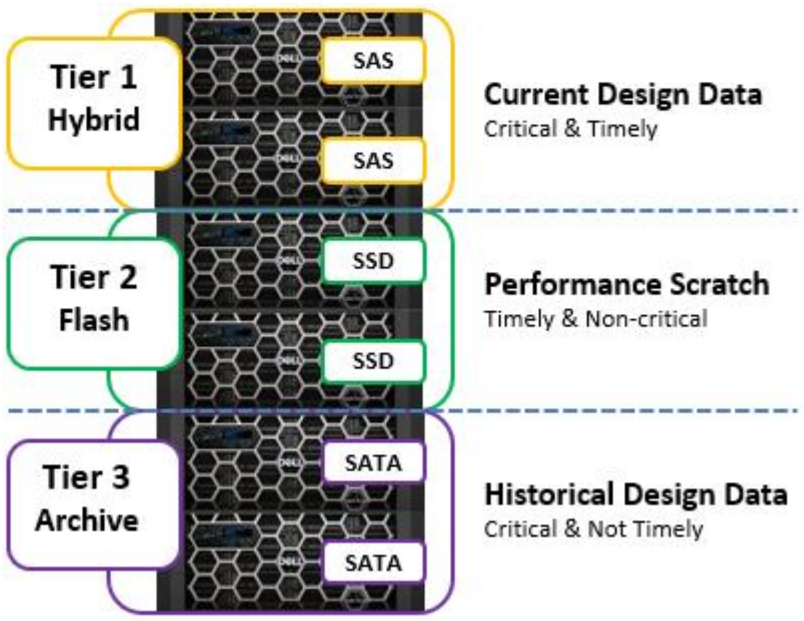


Figure 25. Data performance and protection in EDA

These engineering design data and historical design data examples are valuable. They illustrate the need for data protection for two sets of data that exist side by side in the same infrastructure. One dataset is timely and critical. The other is critical and not timely. The need is clearly to have a system that can serve up data quickly or less expensively but protect both datatypes equally.

EDA workflows often employ an additional tier of storage, or scratch space, for the transient processing requirements of HPC compute farms. Typically, this scratch space has high transactional performance requirements and often employs an SSD-based storage node. However, because scratch is temporary data, the protection and retention requirements are low.

Using SmartPools, these requirements can be met with a three-tier architecture using high-performance SAS and SSD nodes for both the performance and scratch tiers. The architecture will use high-capacity SATA for the high-capacity archive tier. One file pool policy would restrict historical design data to the high-capacity tier, protecting it at a high level. Another would restrict current design data to the fastest tier at the same protection level.

Example C: Investment protection for financial market data

An investment firm collects financial market data from multiple global sources around the clock. This real-time data must be captured, stored, and then analyzed to predict micro and macro market trends to justify purchase recommendations.

The data capture, plus subsequent analysis results, generates terabytes of data per day. Real-time capture is free, but if a time period is missed, historical data must be purchased from a service. Hence, performance and reliability are crucial. Most data analyzed is less than 48 hours old, but the data is also retained indefinitely. Such retention allows long-term analysis and review of model accuracy over time to be performed in any review timeframe.

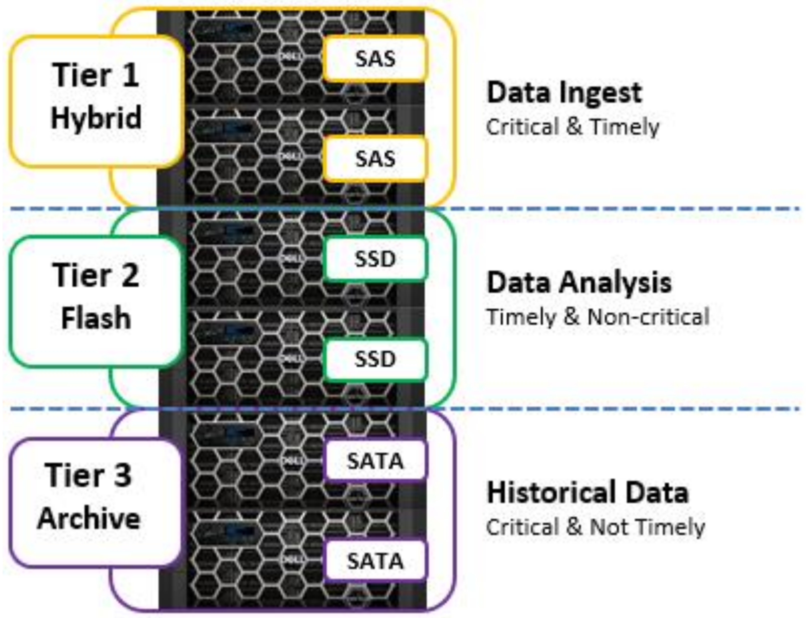


Figure 26. Financial services workload example

The production environment is demanding, and data life is long. There is a strong preference for scalable, durable systems that combine performance and archive capabilities with a compelling cost of ownership.

In this case, the preferred architecture would typically be a three-tiered SmartPools cluster using L3 cache. New data is ingested onto mid-performance spinning media, analysis is run on high-performance spinning media, possibly with SSDs. Older data that is used only intermittently is moved to slower, cost optimized disk.

The original equipment stays online for many years. moving down the hierarchy as completely capitalized capacity. New drives and new nodes with better price and/or performance characteristics are added above them. This OneFS investment protection approach helps avoid the pain and disruption of frequent hadware replacement.

Example D: Metadata performance for seismic interpretation

The seismic data required for energy exploration is typically vast, and growing, with many organizations keeping petabytes of data online for faster analysis and more accurate drilling predictions. This data might be arranged in tens or hundreds of thousands of files across thousands of directories. Often less than 30 percent of those files might be used in any given month. However, administrators constantly must locate files for pre- and post-stack analysis, interpretation, and project planning. The latencies involved with traversing enormous directory structures and file quantities, listing directory contents, and other metadata operations, are often measured in minutes. This latency is clearly unacceptable for operations that are repeated multiple times per day. Seismic data is the core of exploration and is, therefore, critical, and while for rarely accessed files the data might not be timely, the metadata is. In summary, seismic data is critical. Its metadata is timely. Its data might be timely or not timely. In this case, metadata read acceleration, as opposed to L3 cache, is the preferred approach.

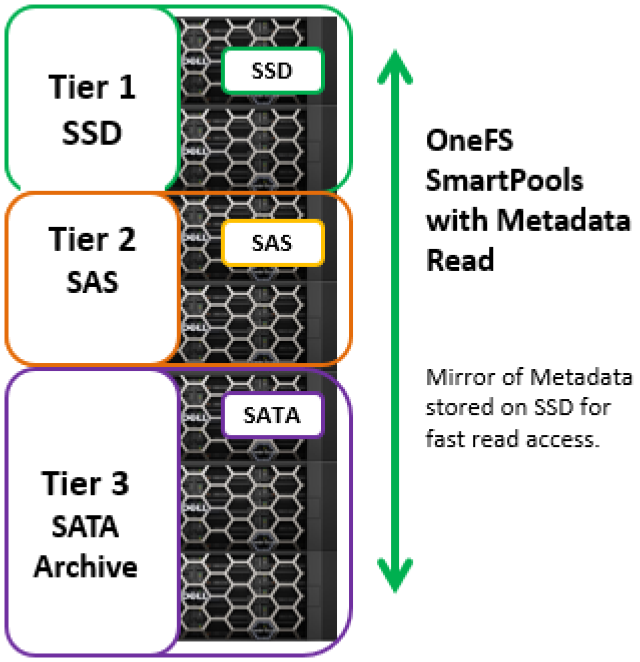

Tier 1
SSD
SSD
Tier 2
SAS
SAS
Tier 3
SATA
Archive
SATA
OneFS SmartPools with Metadata Read
Mirror of Metadata stored on SSD for fast read access.

Figure 27. Tiering with metadata-read acceleration

A mirror of all metadata is stored on SSDs, but data might be relegated to faster or slower disk based on its relative timeliness. The result is the ability to instantly locate any data, while being able to still cost-reduce most of the data on the system. Because metadata is accelerated, many other activities, including backups, migration, and replication, realize performance benefits as well.

Conclusion

To date, traditional storage tiering implementations have typically been expensive, technically risky, and administratively complex. More importantly, they have often failed to achieve their primary goal of aligning data value with accessibility, protection, and performance requirements.

The unique integration of SmartPools with OneFS, the PowerScale scale-out NAS architecture, delivers simple storage tiering with significant storage cost savings, without sacrificing performance or data protection.

With its simple, powerful interface, flexible options, and intelligent default settings, SmartPools is easy to configure and manage, and scales from terabytes to petabytes. Scalability to petabytes and the ability to add new capacity and new technologies while retaining older capacity in the same system means strong investment protection. Integration with OneFS core functions eliminates data migration risks and gives the user control over what system resources are allocated to data movement.

Take the next step

Contact your Dell sales representative or authorized reseller to learn more about how Dell PowerScale scale-out NAS storage solutions can benefit your organization. Visit Dell PowerScale to compare features and get more information.