Dell PowerScale OneFS: Data Reduction and Storage Efficiency

May 2024

H18902.5

White Paper

Abstract

This paper focuses on OneFS, a modern file system that includes in-line data reduction, native real-time data compression and deduplication, SmartDedupe post-process deduplication, Cloning, Writable Snapshots, and Small File Storage Efficiency.

Contents

Executive summary

Overview

Information technology managers across most areas of commerce are grappling with the challenges presented by explosive file data growth, which significantly raises the cost and complexity of storage environments. Business data is often filled with significant amounts of compressible and duplicate information, plus a wide range of file sizes. Data compression and deduplication are specialized data reduction techniques that allow for the reduction in the physical size of data.

OneFS data reduction is yet another milestone in the Dell PowerScale industry-leading data efficiency solutions, and a key ingredient for organizations that want to maintain a competitive edge. Benefits include:

- **Simple:** Minimal configuration compression and deduplication eliminates management burden.
- **Efficient:** By using inline hardware offloading, PowerScale minimizes any performance impact while maximizing storage efficiency.
- **Transparent:** Compression and deduplication are natively integrated into the OneFS file system, making them seamless and transparent to applications and workflows.
- **Harmonious:** OneFS storage efficiency tools work in concert, significantly increasing efficiency and lowering the TCO.

Audience

This paper presents information for deploying and managing data reduction and storage efficiency on a Dell PowerScale all-flash cluster. This paper does not intend to provide a comprehensive background to the OneFS architecture.

See the OneFS Technical Overview white paper for further details on the OneFS architecture.

The target audience for this white paper is anyone configuring and managing data reduction in a Dell PowerScale clustered storage environment. It is assumed that the reader has an understanding and working knowledge of OneFS components, architecture, commands, and features.

More information about OneFS commands and feature configuration is available in the OneFS Administration Guide.

Revisions

Date	Part number/ revision	Description
October 2021	H18902	Initial release—OneFS 9.3
March 2022	H18902.1	Updated for OneFS 9.4
January 2023	H18902.2	Updated for OneFS 9.5
February 2024	H18902.3	Updated for OneFS 9.7
April 2024	H18902.4	Updated for OneFS 9.8

Date	Part number/ revision	Description
May 2024	H18902.5	Updated for PowerScale F910

Note: This document may contain language that is not consistent with Dell Technologies' current guidelines. Dell Technologies plans to update the document over subsequent future releases to revise the language accordingly.

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.

Overview

Introduction

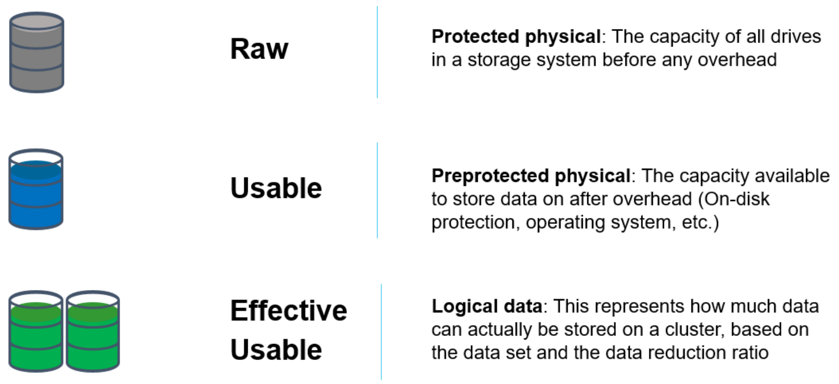
Data reduction comes in various forms and includes techniques such as compression and deduplication. These techniques can be applied to data at various points in their storage life cycle, such as in real time as files are written from a client to storage, or after data has already been stored on disk.

Compression typically uses a lossless algorithm both to reduce the physical size of data when it is written to disk and to decompress the data when it is read back, while retaining full fidelity. More specifically, lossless compression reduces the number of bits in each file by identifying and reducing or eliminating statistical redundancy. No information is lost in lossless compression, and a file can easily be decompressed to its original form.

Deduplication differs from data compression in that it eliminates duplicate copies of repeating data. Whereas compression algorithms identify redundant data inside individual files and encode the redundant data more efficiently, deduplication inspects data and identifies sections, or even entire files that are identical and replaces them with a shared copy.

Both compression and deduplication are transparent to all applications that sit on top of the file system, including protocol-based services such as NFS, SMB, HDFS, and S3. The primary purpose of OneFS inline data reduction is to reduce the storage requirements for data, resulting in a smaller storage footprint, reduced power and cooling requirements, and a reduction in the overall per-TB storage cost. Real time, inline data reduction also helps to shrink the total amount of physical data written to storage devices. This can be beneficial for solid state drives (SSDs) and other media with finite overwrite limits, by reducing flash drive wear rates.

There are three primary measures of storage capacity that are relevant here:



Raw	**Protected physical**: The capacity of all drives in a storage system before any overhead
Usable	**Preprotected physical**: The capacity available to store data on after overhead (On-disk protection, operating system, etc.)
Effective Usable	**Logical data**: This represents how much data can actually be stored on a cluster, based on the data set and the data reduction ratio

Figure 1. The primary measures of storage capacity

Note: Savings due to data reduction are highly dependent on the data itself and can vary considerably. This means that accurate rates of savings are not able to be predicted without comprehensive analysis of the actual data set. Any estimates provided in this document are for broad guidance only.

Terminology

The following table provides definitions for some of the terms that are used in this document.

Table 1. Data Reduction Terminology and Definitions.

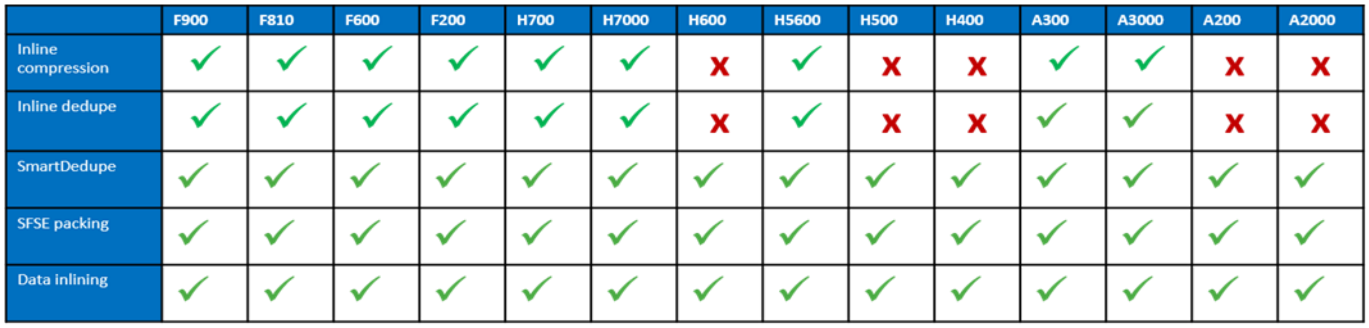
Term	Description
Compression	The process of modifying, encoding or converting the bits structure of data in such a way that it consumes less space on the storage media.
Deduplication	The elimination of duplicate or redundant data, thereby lowering the actual physical storage required.
Inflation	Uncompression of data.
Rehydration	Undeduplication of data.
Protection Group (PG)	OneFS embeds protection into each individual file. To do this, files are segmented into sections, or protection groups (PGs), which are independently protected. The PG tracks all the information about how that logical piece of the file is physically stored, and protected, on-disk.
Cluster	In protection group terminology, a cluster is a division of the logical-block space with IFS MAXCONTIG size and alignment.
Compression Chunk Size	The maximum amount of logical file data that is compressed as a single entity (128KB for OneFS). When an I/O occurs to a compressed region, this also represents the smallest possible I/O, since the I/O must be expanded to include the entire compression chunk.
Packing	The combining of multiple compression chunks together to reduce lost space.
Write Amplification	The cost of additional file system writes to the storage device due to the compression implementation.
Hardware offload	Compression is efficiently performed by a dedicated FPGA on a PCIe card, rather than using the node’s CPU cycles.
In-line Compression	Data is compressed and decompressed on the fly as it is written to and read from to disk
Post-process Compression	Data is compressed in a second pass after it has already been written to disk.
Lossless compression	Reduction of a file’s size with no discernable loss of quality. Lossless compression rewrites the data of the original file in a more efficient way.
In-line Deduplication	Data is deduplicated on the fly as it is written to disk.
Post-process Deduplication	Data is deduplicated in a second pass after it has already been written to disk.
Zero Block Removal	Blocks that contain only zeros are detected and prevented from being written to disk.
Raw	Equivalent to ‘Protected physical’. Total footprint of data including protection overhead FEC erasure coding) and excluding data efficiency savings (compression and deduplication).

Term	Description
Usable	Equivalent to ‘Preprotected Physical’. Data size excluding protection overhead and including data efficiency savings (compression and deduplication).
Effective	Equivalent to ‘Logical data’. Data size excluding protection overhead and excluding data efficiency savings (compression and deduplication).
Logical Data	Equivalent to ‘Effective’. Data size excluding protection overhead and excluding data efficiency savings (compression and deduplication).
Dedupe Saved	Capacity savings from deduplication.
Compression Saved	Capacity savings from inline compression.
Preprotected Physical	Equivalent to ‘Usable’. Data size excluding protection overhead and including data efficiency savings (compression and deduplication).
Protection Overhead	Size of erasure coding used to protect data.
Protected Physical	Equivalent to ‘Raw’. Total footprint of data including protection overhead FEC erasure coding) and excluding data efficiency savings (compression and deduplication).
Effective to Raw Ratio	Compression ratios stated as Effective to Raw are calculated including the data’s protection overhead. OneFS compression ratios are typically calculated and reported using this method.
Effective to Usable Ratio	Compression ratios stated as Effective to Usable are calculated omitting protection overhead. Competitors’ compression ratios are often calculated and reported using this method.
COW	Copy on write, to preserve data in snapshots.
EC	Endurant cache.
BAM	Block allocation manager.
BSW	BAM safe write.
Coalescer	OneFS’ non-volatile write cache.
Packing	Storage efficiency technique involving scanning on-disk data for small files and packing, or containerizing, them into shadow stores, which are FEC protected rather than mirrored.
Data inlining	Storing a small file’s data in the unused space within its inode block.

OneFS data reduction suite

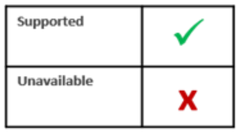
The OneFS data reduction portfolio consists of an array of several features, including compression, deduplication, and small file efficiency. While these components all coexist, they are not universally supported across the full hardware platform portfolio. The following table shows which features are available on each PowerScale hardware platform:

Table 2. Data reduction feature and hardware platform compatibility matrix



	F900	F810	F600	F200	H700	H7000	H600	H5600	H500	H400	A300	A3000	A200	A2000
Inline compression	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✓	✓	✗	✗
Inline dedupe	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✓	✓	✗	✗
SmartDedupe	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
SFSE packing	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Data inlining	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Key:



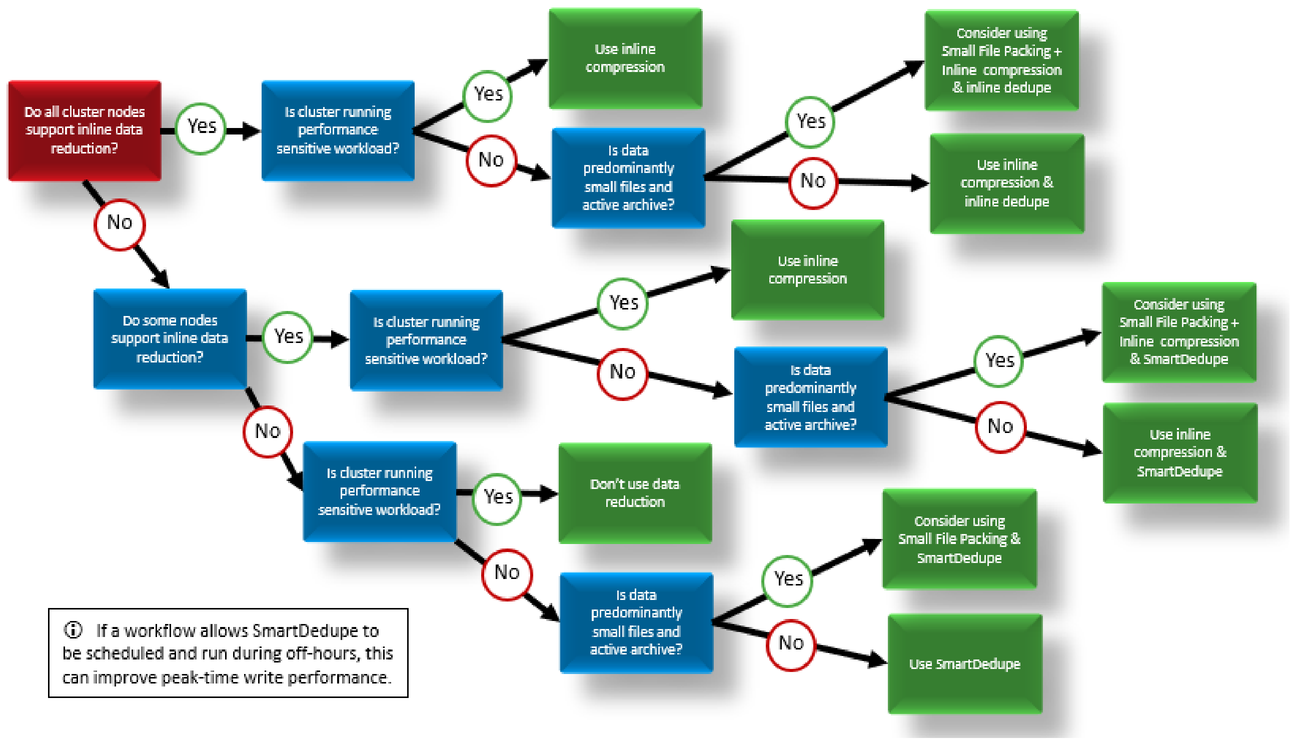
Supported	✓
Unavailable	✗

Furthermore, these OneFS data reduction features are individually licensed and have slightly differing use cases and performance profiles:

Table 3. Data reduction feature licensing requirements

Data Reduction Feature	Licensing Requirement
In-line compression	No license required.
In-line dedupe	No license required.
SmartDedupe	Cluster-wide SmartDedupe License.
SFSE packing	No license required.
Data in-lining	No license required.

The following decision tree can assist with identifying the available and appropriate data reduction features for particular cluster configurations, data sets, and workloads:


Do all cluster nodes support inline data reduction?
Yes
No
Is cluster running performance sensitive workload?
Yes
No
Use inline compression
Is data predominantly small files and active archive?
Yes
No
Consider using Small File Packing + Inline compression & inline dedupe
Use inline compression & inline dedupe
Do some nodes support inline data reduction?
Yes
No
Is cluster running performance sensitive workload?
Yes
No
Use inline compression
Is data predominantly small files and active archive?
Yes
No
Consider using Small File Packing + Inline compression & SmartDedupe
Use inline compression & SmartDedupe
Is cluster running performance sensitive workload?
Yes
No
Don't use data reduction
Is data predominantly small files and active archive?
Yes
No
Consider using Small File Packing & SmartDedupe
Use SmartDedupe

ⓘ If a workflow allows SmartDedupe to be scheduled and run during off-hours, this can improve peak-time write performance.

Figure 2. OneFS data reduction and storage efficiency feature decision tree

The following sections of this paper examine each of the OneFS data reduction and storage efficiency features in more detail:

- In-line Data Reduction, including single instancing, deduplication, and compression
- SmartDedupe post-process deduplication
- Small File Efficiency, including small file packing and inode data in-lining

Each section provides best practices and considerations and contains information about the available tools and strategies for estimating and assessing the efficiency and performance impact of each solution. The inline and post-process deduplication sections also discuss the interoperability between the two approaches.

In-line data reduction

Architecture

OneFS data reduction offers both inline data compression and inline deduplication, and the supporting OneFS architecture is consists of the following principal 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 inline data reduction control path consists of the OneFS command line interface (CLI) and RESTful platform API and is responsible for managing the configuration and reporting of the feature.

In-line data reduction platform

In-line data reduction is supported on the Dell PowerScale F910, F900, F710, F600, F210, and F200 SSD nodes, PowerScale H700/7000 and A300/3000 node, and the Isilon F810 and H5600 platforms.

The specific OneFS versions required to support a cluster or node pool with the following characteristics include:

- OneFS 8.2.1 or later for an F810 node pool.
- OneFS 8.2.2 or later for an H5600 node pool.
- OneFS 9.0 or later for an F600 or F200 node pool.
- OneFS 9.2 or later for an F900 node pool.
- OneFS 9.2.1 or later for an H700, H7000, A300, or A3000 node pool.
- OneFS 9.3 or later for PowerScale P100 and B100 accelerator nodes.
- OneFS 9.7 or later for an F710 or F210 node pool.
- OneFS 9.8 or later for an F910 node pool.

Unlike the other platforms above, each F810 node includes a data reduction hardware off-load adapter. This adapter off-loads certain tasks from the CPU. Specifically, data compression and inflation are transparently performed by the off-load adapter with minimal latency, avoiding the need for consuming a node's expensive CPU and memory resources.

Each F810 node's data reduction off-load adapter contains an FPGA chip, which is dedicated to the compression of data received by means of client connections to the node. These cards reside in the backend PCI-e slot in each of the four nodes. The two Ethernet ports in each adapter are used for the node's redundant backend network connectivity.

In-line data reduction workflow

Data from network clients is accepted as is and makes its way through the OneFS write path until it reaches the BSW engine, where it is broken up into individual chunks. The inline data reduction write path consists of three main phases:

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

If both inline 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.

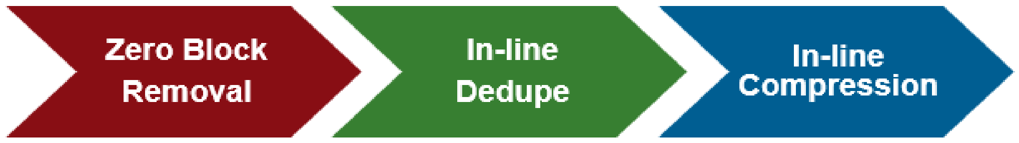


Figure 3. In-line data reduction workflow

Zero block removal

The inline data reduction zero block removal phase detects blocks that contain only zeros and prevents them from being written to disk. This both reduces disk space requirements and avoids unnecessary writes to SSD, resulting in increased drive longevity.

Zero block removal occurs first in the OneFS inline data reduction process. As such, it has the potential to reduce the amount of work that both inline deduplication and compression need to perform. The check for zero data does incur some overhead. However, for blocks that contain non-zero data the check is terminated on the first non-zero data found, which helps to minimize the impact.

The following characteristics are required for zero block removal to occur:

- A full 8KB block of zeroes
- A partial block of zeroes being written to a sparse or prealloc block

The write will convert the block to sparse if not already. A partial block of zeroes being written to a non-sparse, non-preallocated block will not be zero eliminated.

In-line deduplication

While OneFS has offered a native file system deduplication solution for several years, until OneFS 8.2.1 this was always accomplished by scanning the data after it has been written to disk, or post-process. With inline data reduction, deduplication is now performed in real time, as data is written to the cluster. Storage efficiency is achieved by scanning the data for identical blocks as it is received and then eliminating the duplicates.

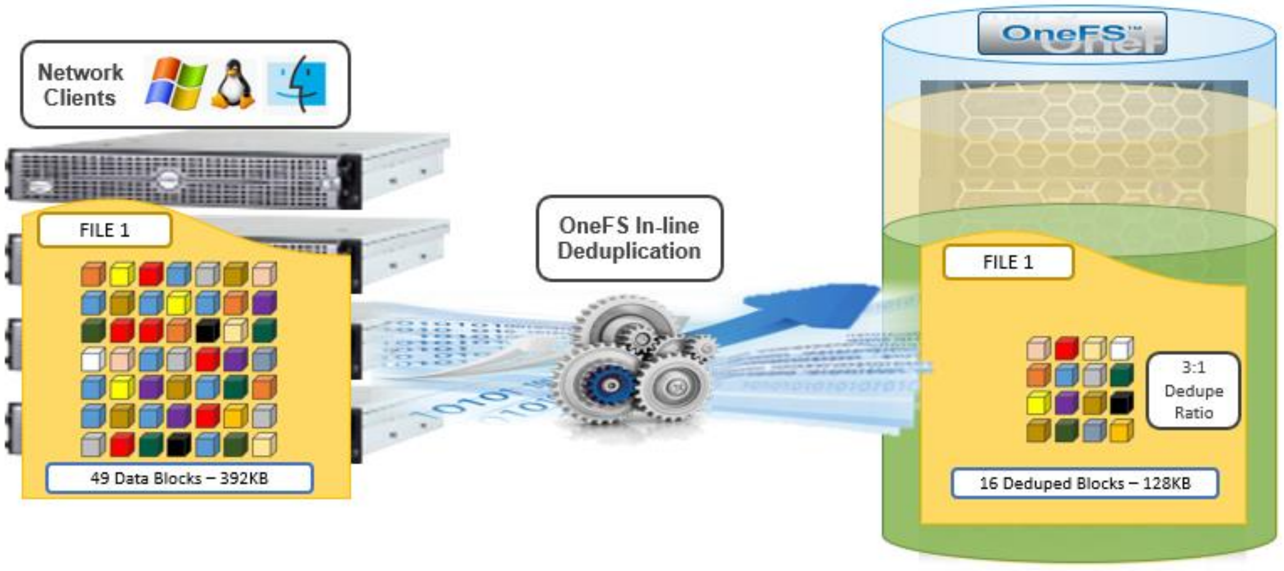


Figure 4. OneFS inline deduplication

When a duplicate block is discovered, inline deduplication moves a single copy of the block to a special set of files known as shadow stores. OneFS shadow stores are file system containers that allow data to be stored in a shareable manner. As such, files on OneFS can contain both physical data and pointers, or references, to shared blocks in shadow stores.

Shadow stores were first introduced in OneFS 7.0, initially supporting OneFS file clones, and there are many overlaps between cloning and deduplicating files. The other main consumer of shadow stores is OneFS Small File Storage Efficiency. This feature maximizes the space utilization of a cluster by decreasing the amount of physical storage required to house the small files that comprise a typical healthcare dataset.

Shadow stores are similar to regular files but are hidden from the file system namespace, so cannot be accessed using a pathname. A shadow store typically grows to a maximum size of 2GB, which is around 256K blocks, with each block able to be referenced by 32,000 files. If the reference count limit is reached, a new block is allocated, which may or may not be in the same shadow store. Also, shadow stores do not reference other shadow stores. And snapshots of shadow stores are not permitted because the data contained in shadow stores cannot be overwritten.

When a client writes a file to a node pool configured for inline deduplication on a cluster, the write operation is divided up into whole 8KB blocks. Each of these blocks is then hashed and its cryptographic ‘fingerprint’ compared against an in-memory index for a match. At this point, one of the following operations will occur:

1. If a match is discovered with an existing shadow store block, a byte-by-byte comparison is performed. If the comparison is successful, the data is removed from the current write operation and replaced with a shadow reference.
2. When a match is found with another LIN, the data is written to a shadow store instead and replaced with a shadow reference. Next, a work request is generated and queued that includes the location for the new shadow store block, the matching

LIN and block, and the data hash. A byte-by-byte data comparison is performed to verify the match and the request is then processed.

3. If no match is found, the data is written to the file natively and the hash for the block is added to the in-memory index.

In order for inline deduplication to be performed on a write operation, the following conditions need to be true:

- In-line dedupe must be globally enabled on the cluster.
- The current operation is writing data (that is, not a truncate or write zero operation).
- The ‘no_dedupe’ flag is not set on the file.
- The file is not a special file type, such as an alternate data stream (ADS) or an EC (endurant cache) file.
- Write data includes fully overwritten and aligned blocks.
- The write is not part of a ‘rehydrate’ operation.
- The file has not been packed (containerized) by SFSE (small file storage efficiency).

OneFS inline deduplication uses the 128-bit CityHash algorithm, which is both fast and cryptographically strong. This contrasts with OneFS’ post-process SmartDedupe, which uses SHA-1 hashing.

Each F910, F900, F810, F710, F600, F210, F200, F700/7000, H5600, or A300/3000 node in a cluster with inline dedupe enabled has its own in-memory hash index that it compares block ‘fingerprints’ against. The index lives in system RAM and is allocated using physically contiguous pages and accessed directly with physical addresses. This avoids the need to traverse virtual memory mappings and does not incur the cost of translation lookaside buffer (TLB) misses, minimizing deduplication performance impact.

The maximum size of the hash index is governed by a pair of sysctl settings, one of which caps the size at 16GB, and the other which limits the maximum size to 10% of total RAM. The strictest of these two constraints applies. While these settings are configurable, the recommended best practice is to use the default configuration. Any changes to these settings should only be performed under the supervision of Dell support.

Since inline dedupe and SmartDedupe use different hashing algorithms, the indexes for each are not shared directly. However, the work performed by each dedupe solution can be leveraged by each other. For instance, if SmartDedupe writes data to a shadow store, when those blocks are read, the read hashing component of inline dedupe will see those blocks and index them.

When a match is found, inline dedupe performs a byte-by-byte comparison of each block to be shared to avoid the potential for a hash collision. Data is prefetched prior to the byte-by-byte check and then compared against the L1 cache buffer directly, avoiding unnecessary data copies and adding minimal overhead. Once the matching blocks have been compared and verified as identical, they are then shared by writing the matching data to a common shadow store and creating references from the original files to this shadow store.

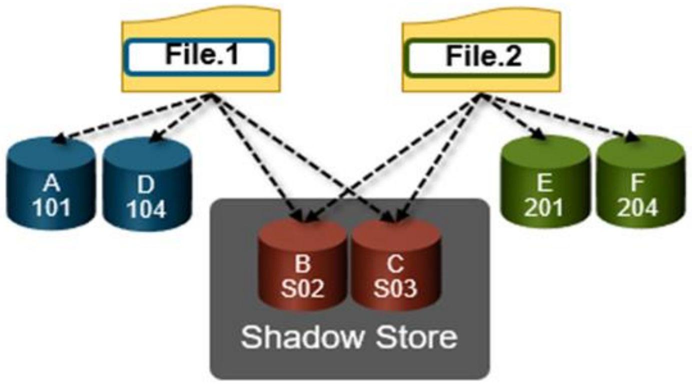


Figure 5. OneFS duplicate block sharing

In-line dedupe samples every whole block written and handles each block independently, so it can aggressively locate block duplicity. If a contiguous run of matching blocks is detected, inline dedupe will merge the results into regions and process them efficiently.

In-line dedupe also detects dedupe opportunities from the read path, and blocks are hashed as they are read into L1 cache and inserted into the index. If an existing entry exists for that hash, inline dedupe knows there is a block sharing opportunity between the block it just read and the one previously indexed. It combines that information and queues a request to an asynchronous dedupe worker thread. As such, it is possible to deduplicate a data set purely by reading it all. To help mitigate the performance impact, the hashing is performed out-of-band in the prefetch path, rather than in the latency-sensitive read path.

In-line compression

Under the hood, the F810 nodes use an FPGA-based hardware offload engine resident on the backend PCI-e network adapter to perform real-time data compression. This occurs as files are written to from a node in the cluster over a connected client session. Similarly, files are reinflated on demand as they are read by clients.

On top of the FPGA, the OneFS hardware off-load engine uses a proprietary implementation of DEFLATE with the highest level of compression, while incurring minimal to no performance penalty for highly compressible datasets.

The compression engine consists of three main components:

Table 4. OneFS data reduction engine components

Engine component	Description
Search Module	LZ77 search module analyzes inline file data chunks for repeated patterns.
Encoding Module	Performs data compression (Huffman encoding) on target chunks.
Decompression Module	Regenerates the original file from the compressed chunks.

Note: Because they reside on the same card, the data compression engine shares PCI-e bandwidth with the node’s backend Ethernet interfaces. In general, there is plenty of bandwidth available. A best practice is to run highly compressible datasets through the F810 nodes with compression enabled. However, it is not advisable to run non-compressible datasets with compression enabled.

OneFS provides software-based compression for the F910, F900, F710, F600, F210, F200, F700/7000, H5600, and A300/3000 platforms. Compression in software is also used as fallback in the event of an F810 hardware failure, and in a mixed cluster for use in nodes without a hardware offload capability. Both hardware and software compression implementations are DEFLATE compatible.

Compression file chunking

When a file is written to OneFS using inline data compression, the file's logical space is divided up into equal sized chunks called compression chunks. Compaction is used to create 128KB compression chunks, with each chunk consisting of sixteen 8KB data blocks. This is optimal since 128KB is the same chunk size that OneFS uses for its data protection stripe units, providing simplicity and efficiency, by avoiding the overhead of additional chunk packing.

For example, consider the following 128KB chunk:

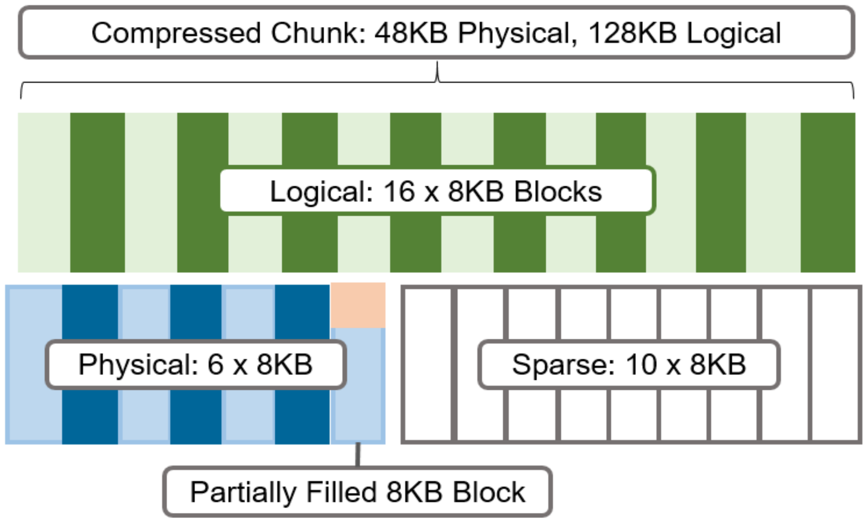

Compressed Chunk: 48KB Physical, 128KB Logical
Logical: 16 x 8KB Blocks
Physical: 6 x 8KB
Sparse: 10 x 8KB
Partially Filled 8KB Block

Figure 6. Compression chunks and OneFS transparent overlay

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. The orange sector in the figure above represents the trailing, partially filled 8KB block in the chunk. Depending on how each 128KB chunk compresses, the last block may be under-utilized by up to 7KB after compression.

Efficiency savings must be at least 8KB (one block) 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 protected with Forward Error Correction

(FEC) parity blocks, reducing the number of FEC blocks and therefore providing further overall storage savings.

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

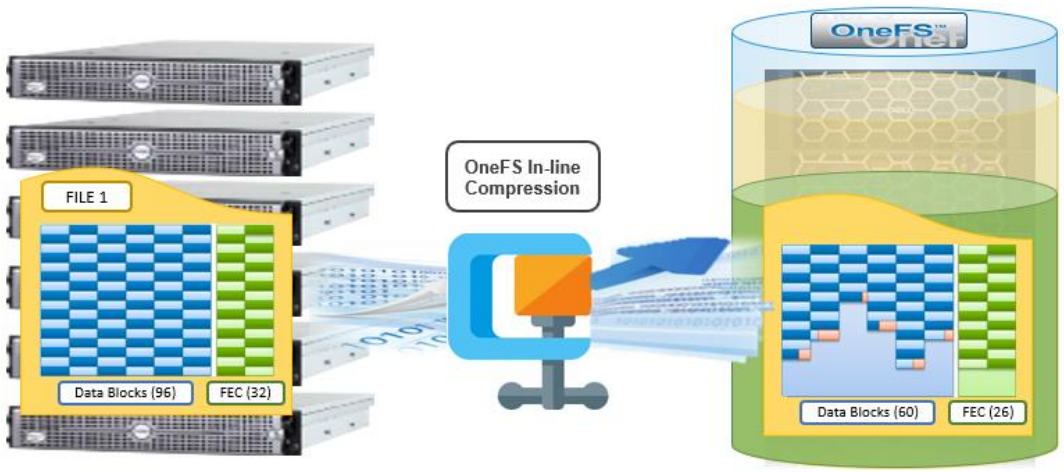


Figure 7. OneFS inline compression

In the figure above, a 768KB file (file1) is written from a Windows client to an F810 cluster. After passing through the OneFS inline compression engine, the logical data footprint of that file is reduced from ninety-six to sixty 8KB blocks, across six chunks. This is represented by the blue data blocks. The file is then FEC protected using twenty-six parity blocks, shown in green.

In-line data reduction write path

In a PowerScale cluster, data, metadata, and inodes are all distributed across multiple drives on multiple nodes. When reading or writing to the cluster, a client is directed by SmartConnect to the desired protocol head on a particular node, or initiator. This node then acts as the ‘captain’ for the operation, performing the chunking and data compression, orchestrating the layout of data and metadata, the creation of erasure codes, and the normal operations of lock management and permissions control.

Take for example a four node F810 cluster. A Windows client connects to the top half, or initiator, on node 1 to write a file. Once the SMB session is established and write request granted, the client begins to send the file data across the front-end network to the cluster where it is initially buffered in the coalescer, or OneFS write cache. The purpose of the coalescer is to build up a large contiguous range of data that will make the write operation more efficient.

When the coalescer is flushed, data chunks, typically sized on protection group boundaries, are passed through the data reduction pipeline. First, if inline dedupe is enabled, the incoming data is scanned for zero block removal and deduplication opportunities. When found, any zero blocks are stripped out and any matching blocks are deduplicated. Next, chunks that meet the ‘compressibility’ criteria described above are compressed by the FPGA. Finally, the initiator executes its ‘write plan’ for the file data, optimizing for layout efficiency and the selected protection policy, and the chunks/stripes are written to SSDs on the bottom half of the participant nodes.

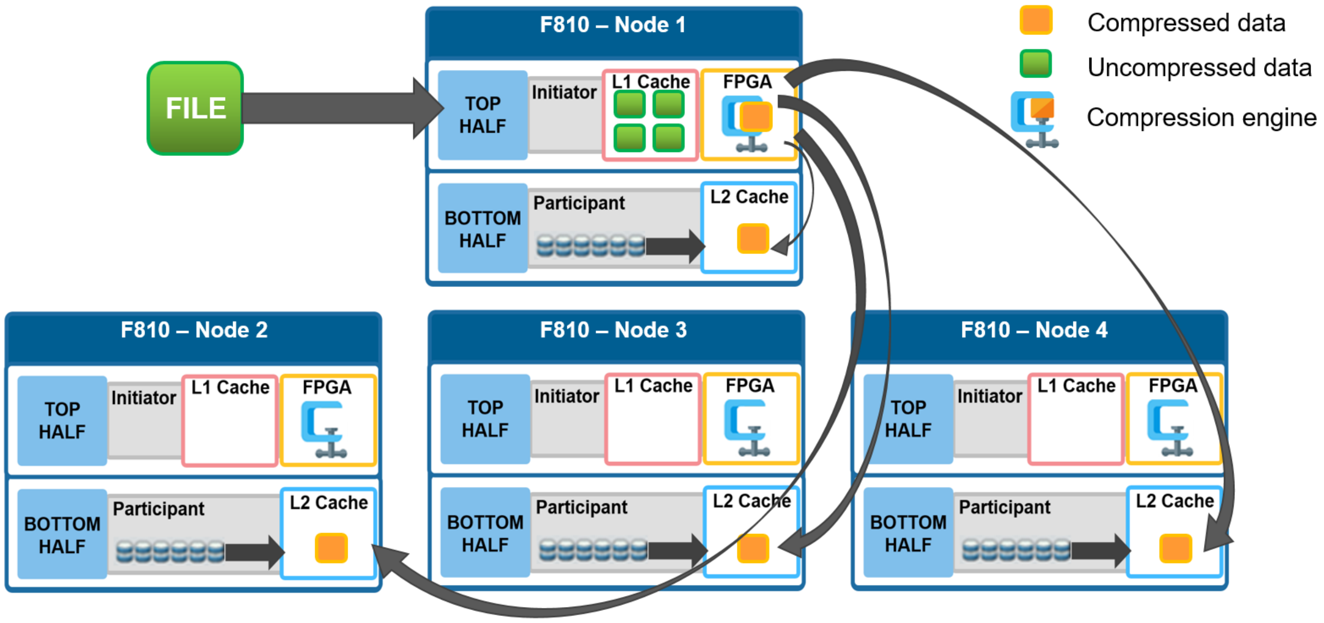


Figure 8. File writes with compression

OneFS stripes data across all nodes—and not simply across disks—and protects the files, directories, and associated metadata by software erasure-code or mirroring technology. 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. For any given file, the stripe width is the number of nodes (not disks) that a file is written across. For example, on the 4-node F810 cluster above with the recommended +2d:1n protection level, OneFS will use a stripe width of 8 and protection level of 6+2, where each node is used twice, that is: two data stripe units are written to each of three nodes, and two FEC units to the remaining node.

For details about OneFS data protection, see the OneFS Technical Overview white paper.

In-line data reduction read path and caching integration

In the diagram below, an NFS client attaches to node 1 and issues a read request for a file. Node 1, the captain, gathers all the chunks of data from the various nodes in the cluster and presents it in a cohesive way to the requesting client. Since the file's data has been stored in a compressed form on nodes' SSDs, node 1 needs to gather all the constituent chunks and decompress the data so the file can be sent across the wire to the client in its original form.

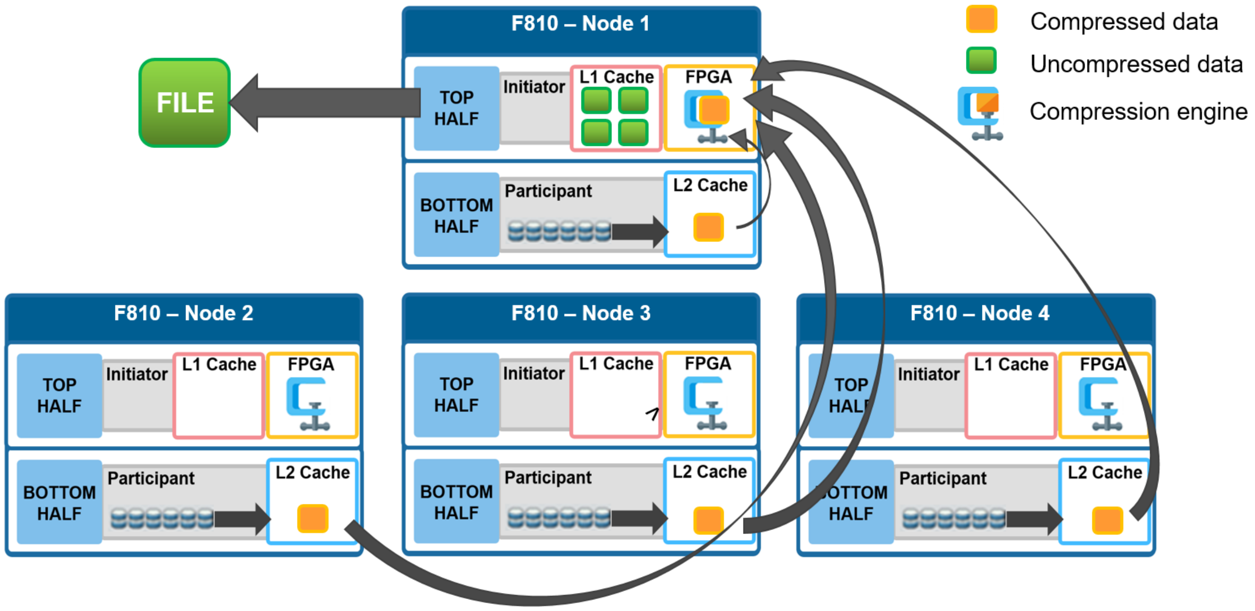


Figure 9. File reads with compression

During this read process, the L2 read cache on the participant nodes (nodes 2 through 4) is populated with the compressed data chunks that are sent to node 1. This means that any additional read requests for this file can be served straight from low latency cache, rather than reading again from the drives. This process both accelerates read performance and reduces wear on the SSDs.

To support OneFS inline compression, a node's L1, or client-side, read cache is divided into separate address spaces so that both the on-disk compressed data and the logical uncompressed data can be cached. The address space for the L1 cache is already split for data and FEC blocks, so a similar technique is used to divide it again. Data in the uncompressed L1 cache is fed from data in the compressed L1 cache which, in turn, is fed from disk.

OneFS prefetch caching has also been enhanced to accommodate compressed data. Since reading part of a compressed chunk results in the entire compression chunk being cached, it will effectively mean that prefetch requests are rounded to compression chunk boundaries. Since a prefetch request is not complete until the uncompressed data is available in cache, the callback used for prefetch requests performs the decompression.

Inline data reduction in a mixed cluster

A mixed, or heterogeneous, cluster is one that consists of two or more different types or node. For compression, there are two main concepts in play in a mixed cluster:

- Reading compressed data to return the logical data (for client traffic, NDMP, or otherwise).
- Writing a previously compressed file to disk in uncompressed format (because the target tier does not support compression).

The former happens in L1 memory and not on-disk. As such, only F910, F900, F810, F710, F600, F210, F200, F700/7000, H5600, and A300/3000 storage pools may contain compressed data.

In general, OneFS does not allow deduplication across different disk pool policies. For inline dedupe, a deduplicated file that is moved to another tier will retain the shadow references to the shadow store on the original disk pool. While this behavior violates the rule for deduping across different disk pool policies, it is preferred to do this than rehydrate files that are moved. Further deduplication activity on that file will no longer be allowed to reference any blocks in the original shadow store. The file will need to be deduped against other files in its new node pool.

Writes to a mixed F810 cluster

Figure 10 depicts a file write in a mixed cluster environment. A client connects to an H400 node and writes a file with a path-based file pool policy that directs the file to an F810 nodepool. In this scenario, the H400 node first scans the incoming data for zero block removal and/or deduplication opportunities.

Note: Zero block elimination and inline dedupe will only be enabled on nodes that have local drives in a disk pool with the data reduce flag set.

The data reduce flag will only be set on F910, F900, F810, F710, F600, F210, F200, H700/7000, H5600, and A300/3000 disk pools and therefore zero block elimination and inline dedupe will only be performed on those platforms. When found, any zero blocks are stripped out and any matching blocks are deduplicated.

Next, the data is divided into 128KB compression chunks as usual. However, since the H400 node does not have an FPGA offload card, it instead performs compression on the chunks in software.

Note: Unlike inline dedupe, compression does not require the initiator node to be a member of a disk pool with the data reduce flag set. If the target disk pool for the write has data reduce set and the cluster has inline compression enabled, compression will be performed.

A different compression algorithm is used to help minimize the performance impact. Each compressed chunk is then FEC protected, and the H400 uses its write plan to place blocks on the participant nodes. The chunks are written in compressed form over the back-end network to the appropriate F810 nodes.

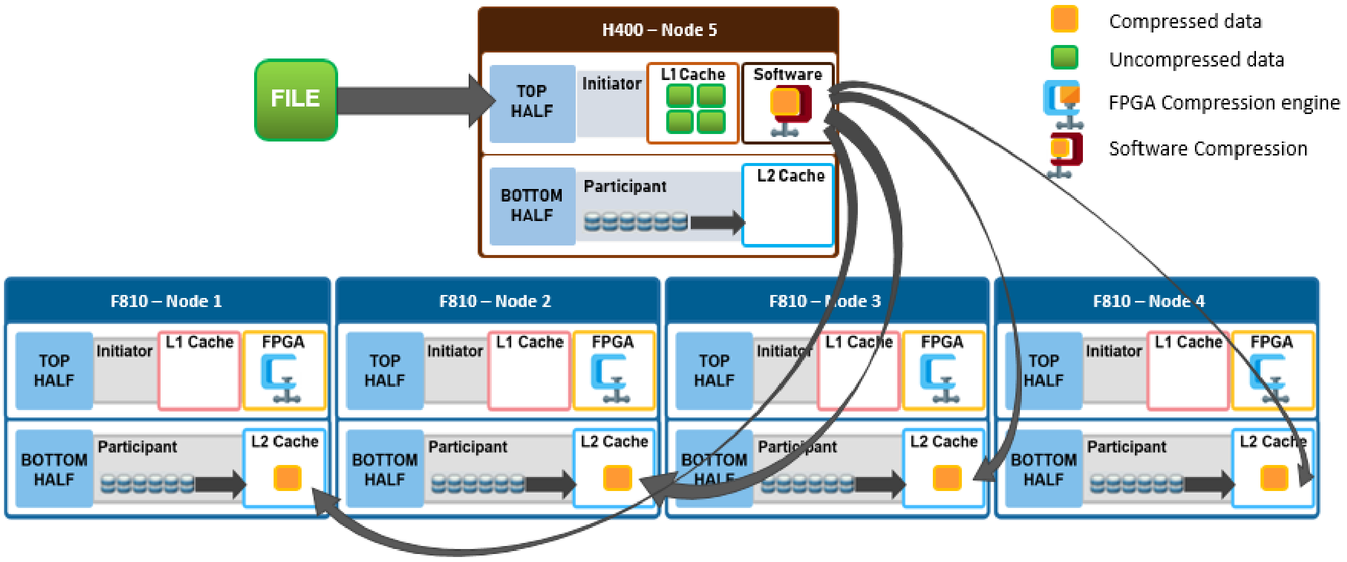


Figure 10. File write in a mixed cluster with software compression

Reading from a mixed F810 cluster

In the following mixed cluster scenario, a client connects to an H400 node and issues a read request for a compressed file housed on an F810 nodepool. The H400 retrieves all the compressed chunks from the pertinent F810 nodes over the backend network. Since the H400 has no FPGA offload card, the decompression of the chunks is performed in software. Software compression uses a different DEFLATE-compatible algorithm to help minimize the performance impact of non-offloaded decompression. Once the chunks have been decompressed, the file is then reassembled and sent over Ethernet in uncompressed form to the requesting client.

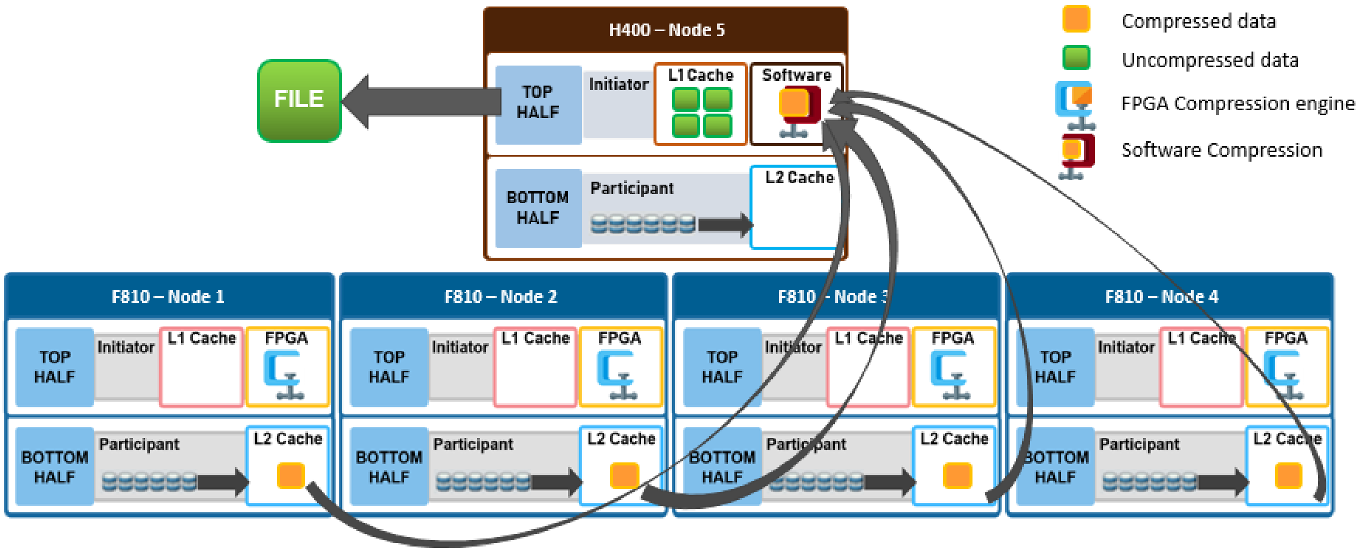


Figure 11. File read in a mixed cluster with software compression

Data reduction and tiering in a mixed cluster

Consider a mixed cluster that consists of an F810 flash performance tier and an A2000 archive tier. SmartPools is licensed, and a file pool policy is crafted to move data over three months old from the F810 flash tier to the archive tier. Files are stored in a compressed form on the F810 tier.

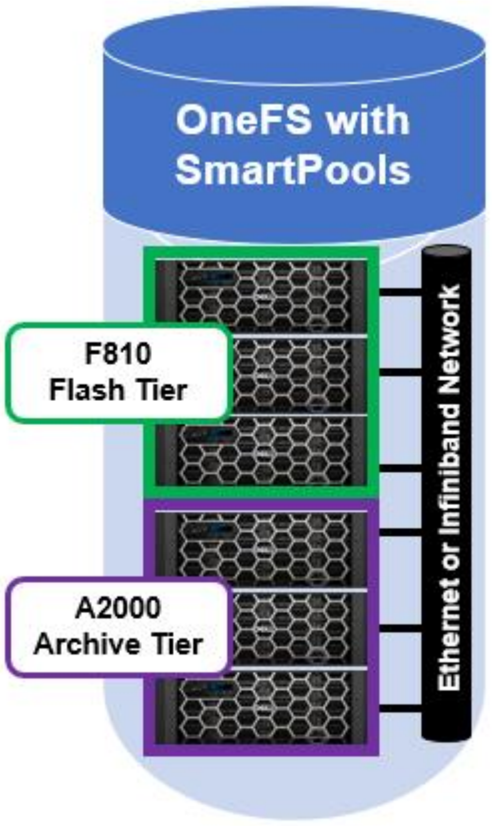


Figure 12. OneFS SmartPools running on a two tier cluster

When SmartPools runs, decompression occurs as the files targeted for down-tiering are restriped from the F810 tier to the A2000 tier inside the SmartPools job. The SmartPools job runs across all the nodes in the cluster - both F810 and A2000 - so whether hardware assisted decompression is available depends on which node(s) are running the job worker threads. Once on the A2000 tier, files will remain uncompressed.

If a file has been deduped, whether by inline or post process deduplication, the deduped state of the file will remain unchanged when tiered. This is true even if the file is moved from a disk pool that supports data reduce to a disk pool that does not.

If another file pool policy is created to up-tier files from the A2000 back up to the F810, the file chunks will be compressed when they are restriped onto the F810 nodes. Once again, all nodes in the cluster will participate in the SmartPools job.

As data moves between tiers in a mixed cluster, it must be rewritten from the old to new drives. If the target tier has a different ‘data-reduce’ setting than the source tier, data is compressed or decompressed as appropriate. The sizing of the F810 pool should be independent of whether it is participating in a mixed cluster or not. However, because up-tiering by the job-engine can run job tasks on the lower tier nodes, the achievable compression ratios may be slightly less efficient when software compression is used.

Data reduction and replication in a mixed cluster

SyncIQ is licensed on a mixed F810 and A2000 cluster and a SyncIQ policy is configured to replicate data to a target cluster. On the source cluster, replication traffic is isolated to just the A2000 nodes. When SyncIQ is run, worker threads on the A2000 nodes gather all the compressed chunks from the F810 nodes over the backend network (RBM). The A2000 nodes then perform decompression of the chunks in software. As discussed previously, software compression uses a different DEFLATE-compatible algorithm from hardware offload to help minimize the performance impact of non-offloaded decompression. Once the chunks have been decompressed, the data is then sent over Ethernet to the target cluster in uncompressed form.

Similarly, deduplicated data is always rehydrated when it exits a cluster. For a data service such as SyncIQ, data is replicated in its entirety and shadow stores and shadow links are not preserved on the target. This means that the target cluster must have sufficient space to house the full size of the replicated data set. If the target cluster happens to also be F810 hardware and inline data reduction is enabled, compression and/or deduplication will be performed as the replication data is ingested by each target node.

Compression and backup in a mixed cluster

A mixed F810 and A2000 cluster is configured for NDMP backup from the A2000 nodes. When a backup job runs, the A2000 nodes retrieve all the compressed chunks from the pertinent F810 nodes over the backend network (RBM). Since the A2000 has no FPGA offload card, the decompression of the chunks is performed in software. Once the chunks have been decompressed, each file is then reassembled and sent over Fibre Channel (2-way NDMP) or Ethernet (3-way NDMP) in uncompressed form to the backup device(s).

With NDMP, deduplicated data is rehydrated when it leaves the cluster, and shadow stores and shadow links are not preserved on the backup. The NDMP tape device or VTL will need to have sufficient space to house the full size of the data set.

Inline data reduction configuration and management

OneFS inline data reduction uses a simple administrative control plane. Configuration is by the command line interface (CLI), using the ‘isi compression’ and ‘isi dedupe inline’ commands. There are also utilities provided to decompress, or rehydrate, compressed and deduplicated files if necessary. Plus, there are tools for viewing on-disk capacity savings that inline data reduction has generated.

The ‘isi_hw_status’ CLI command can be used to confirm and verify the node(s) in a cluster. For example:

```
# isi_hw_status -i | grep Product
Product: F810-4U-Single-256GB-1x1GE-2x40GE SFP+-24TB SSD
```

Enabling compression

Since compression configuration is binary, either on or off across a cluster, it can be easily controlled by the OneFS command line interface (CLI). For example, the following syntax will enable compression and verify the configuration:

```
# isi compression settings view
    Enabled: No
# isi compression settings modify --enabled=True
# isi compression settings view
    Enabled: Yes
```

Note: In-line compression is enabled by default on new F810 clusters running OneFS 8.2.1 and later, on new H5600 clusters running OneFS 8.2.2 and later, on new F600 and F200 clusters running OneFS 9.0, on F900 clusters running OneFS 9.2, on H700/7000 and A300/3000 clusters running OneFS 9.2.1 and later, F710 and F210 clusters running OneFS 9.7 and later, and F910 clusters running OneFS 9,8 and later.

In a mixed cluster containing other node styles in addition to compression nodes, files will only be stored in a compressed form on F910, F900, F810, F710, F600, F210, F200, H5600, H700/7000, and A300/3000 node pool(s). Data that is written or tiered to storage pools of other hardware styles will be uncompressed on the fly when it moves between pools. A non-inline compression supporting node on the cluster can be an initiator for compressed writes in software to a compression node pool. However, this may generate significant CPU overhead for lower powered nodes, such as the A-series hardware and provide only software fallback based compression with lower compressibility.

Verifying compression

While there are no visible user space changes when files are compressed, the ‘isi get’ CLI command provides straightforward method to verify whether a file is compressed. If compression has occurred, both the ‘disk usage’ and the ‘physical blocks’ metric reported by the ‘isi get –DD’ CLI command will be reduced. Also, at the bottom of the command’s output, the logical block statistics will report the number of compressed blocks. For example:

```
Metatree logical blocks:
   zero=260814 shadow=0 ditto=0 prealloc=0 block=2 compressed=1328
```

For more detailed information, the –O flag, which displays the logical overlay, can be used with the ‘isi get’ command. This command is described in more detail later in this paper.

Disabling compression

OneFS inline data compression can be disabled from the CLI with the following syntax:

```
# isi compression settings modify --enabled=False
# isi compression settings view
    Enabled: No
```

Enabling inline deduplication

Since inline deduplication configuration is binary, either on or off across a cluster, it can be easily controlled by the OneFS command line interface (CLI). For example, the following syntax will enable inline deduplication and verify the configuration:

```
# isi dedupe inline settings view
    Mode: disabled
# isi dedupe inline settings modify --mode enabled
# isi dedupe inline settings view
    Mode: enabled
```

Note: In-line deduplication is enabled by default for new clusters running OneFS 9.4 or later. For earlier OneFS releases, inline dedupe is disabled by default

Verifying inline deduplication

While there are no visible user space changes when files are deduplicated, if deduplication has occurred, both the 'disk usage' and the 'physical blocks' metric reported by the 'isi get –DD' CLI command will be reduced. Also, at the bottom of the command's output, the logical block statistics will report the number of shadow blocks. For example:

```
Metatree logical blocks:
   zero=260814 shadow=362 ditto=0 prealloc=0 block=2 compressed=0
```

Disabling inline deduplication

OneFS inline data deduplication can be disabled from the CLI with the following syntax:

```
# isi dedupe inline settings modify --mode disabled
# isi dedupe inline settings view
    Mode: disabled
```

Pausing inline deduplication

OneFS inline data deduplication can be paused from the CLI with the following syntax:

```
# isi dedupe inline settings modify --mode paused
```

Assess mode

OneFS inline data deduplication can be run in assess mode from the CLI with the following syntax:

```
# isi dedupe inline settings modify --mode assess
```

Events and alerts

Issues with inline compression can generate the following OneFS events and alerts. These include:

Table 5. OneFS inline compression events and alerts

Event category	Alert condition	Event trigger	Event ID
Health	In-line compression has failed on the specific node	Falling back to software	40070001
Health	In-line compression hardware is unhealthy	Increased error rates Device is delisted	900160101
Availability	In-line compression hardware is unavailable	The device is missing	900160100

Similarly, problems with inline dedupe can generate the following OneFS events and alerts. These include:

Table 6. OneFS inline dedupe events and alerts

Event category	Alert condition	Event ID
Health	Inline dedupe index allocation failed	400180001

Event category	Alert condition	Event ID
Health	Inline dedupe index allocation in progress	400180002
Availability	Inline dedupe not supported	400180003
Health	Inline dedupe index is smaller than requested	400180004
Health	Inline dedupe index has non-standard layout	400180005

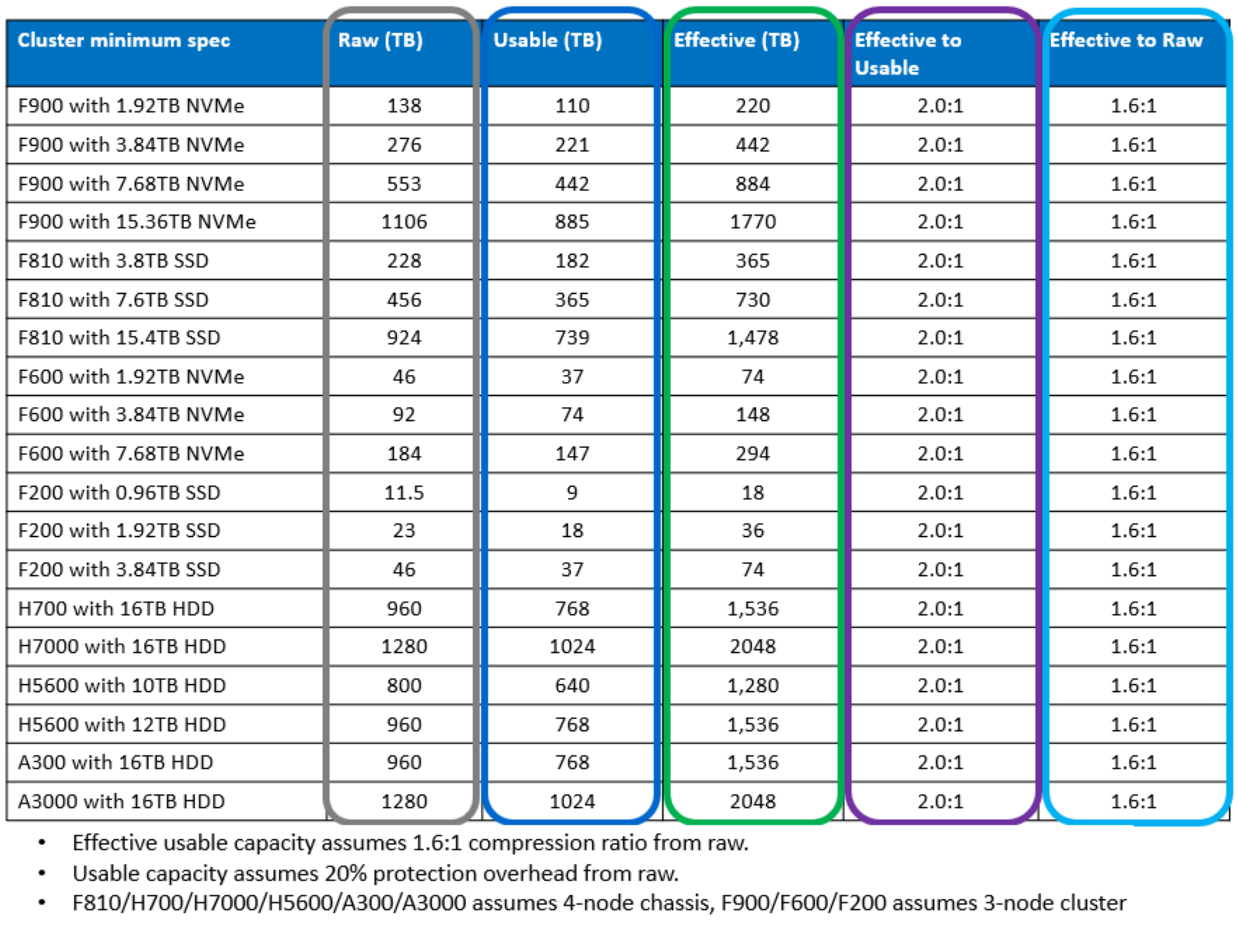
If inline deduplication encounters an unrecoverable error, it will restart the write operation with inline dedupe disabled. If any of the above alert conditions occur, please contact Dell Technical Support for further evaluation.

Inline data reduction efficiency

Compression and deduplication can significantly increase the storage efficiency of data. However, the actual space savings will vary depending on the specific attributes of the data itself.

The following table illustrates the relationship between the effective to usable and effective to raw ratios for the F910, F900, F810, F710, F600, F210, F200, H700, H7000, H5600, A300, and A3000 platforms:

Table 7. Effective to usable and raw relationships in various PowerScale configurations



Cluster minimum spec	Raw (TB)	Usable (TB)	Effective (TB)	Effective to Usable	Effective to Raw
F900 with 1.92TB NVMe	138	110	220	2.0:1	1.6:1
F900 with 3.84TB NVMe	276	221	442	2.0:1	1.6:1
F900 with 7.68TB NVMe	553	442	884	2.0:1	1.6:1
F900 with 15.36TB NVMe	1106	885	1770	2.0:1	1.6:1
F810 with 3.8TB SSD	228	182	365	2.0:1	1.6:1
F810 with 7.6TB SSD	456	365	730	2.0:1	1.6:1
F810 with 15.4TB SSD	924	739	1,478	2.0:1	1.6:1
F600 with 1.92TB NVMe	46	37	74	2.0:1	1.6:1
F600 with 3.84TB NVMe	92	74	148	2.0:1	1.6:1
F600 with 7.68TB NVMe	184	147	294	2.0:1	1.6:1
F200 with 0.96TB SSD	11.5	9	18	2.0:1	1.6:1
F200 with 1.92TB SSD	23	18	36	2.0:1	1.6:1
F200 with 3.84TB SSD	46	37	74	2.0:1	1.6:1
H700 with 16TB HDD	960	768	1,536	2.0:1	1.6:1
H7000 with 16TB HDD	1280	1024	2048	2.0:1	1.6:1
H5600 with 10TB HDD	800	640	1,280	2.0:1	1.6:1
H5600 with 12TB HDD	960	768	1,536	2.0:1	1.6:1
A300 with 16TB HDD	960	768	1,536	2.0:1	1.6:1
A3000 with 16TB HDD	1280	1024	2048	2.0:1	1.6:1

- Effective usable capacity assumes 1.6:1 compression ratio from raw.
- Usable capacity assumes 20% protection overhead from raw.
- F810/H700/H7000/H5600/A300/A3000 assumes 4-node chassis, F900/F600/F200 assumes 3-node cluster

The following table provides descriptions for the various OneFS reporting metrics, such as those returned by the 'isi statistics data-reduction' command described below. The table attempts, where appropriate, to equate the OneFS nomenclature with more general industry terminology:

Table 8. OneFS data reduction reporting metrics

Data Metric	Also Known As	Description
Protected logical	Application logical	Data size including sparse data, zero block eliminated data, and CloudPools data stubbed to a cloud tier.
Logical data	Effective Filesystem logical	Data size excluding protection overhead and spars data. and including data efficiency savings (compression and deduplication).
Zero-removal saved		Capacity savings from zero removal.
Dedupe saved		Capacity savings from deduplication.
Compression saved		Capacity savings from in-line compression.
Preprotected physical	Usable Application physical	Data size excluding protection overhead and including data efficiency savings (compression and deduplication).
Protection overhead		Size of erasure coding used to protect data.
Protected physical	Raw Filesystem physical	Total footprint of data including protection overhead FEC erasure coding) and excluding data efficiency savings (compression and deduplication).
Dedupe ratio		Estimated deduplication ratio. Will be displayed as 1.0:1 if there are no deduplicated blocks on the cluster.
Compression ratio	Effective to Usable	Usable efficiency ratio from compression, calculated by dividing 'logical data' (green) by 'unprotected physical' (blue) and expressed as x:1
Data reduction ratio		Usable efficiency ratio from compression and deduplication. Will display the same value as the compression ratio if there is no deduplication on the cluster.
Efficiency ratio	Effective to Raw	Overall raw efficiency ratio, calculated by dividing 'logical data' (green) by 'protected physical' (grey) and expressed as x:1

Note: The color scheme in this table is used throughout this paper to categorize and distinguish between the various data metrics.

The interrelation of the data capacity metrics described above can be illustrated in a graphical representation.

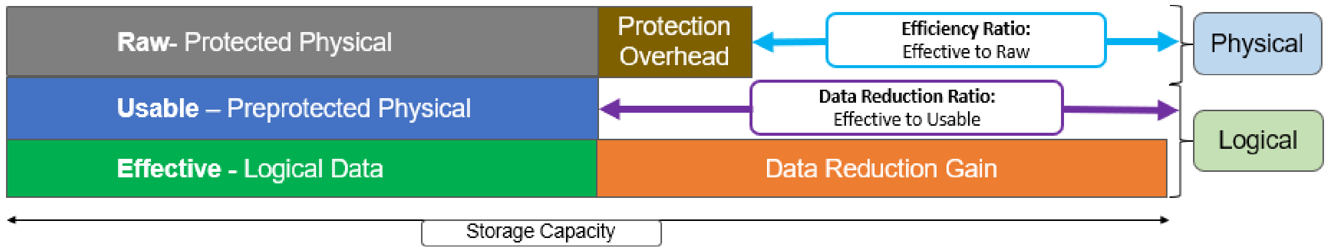


Figure 13. OneFS data capacity metrics interrelation

As we can see, the preprotected physical (usable) value is derived by subtracting the protection overhead from the protected physical (raw) metric. Similarly, the difference in size between preprotected physical (usable) and logical data (effective) is the efficiency savings. If OneFS SmartDedupe is also licensed and running on the cluster, this data reduction savings value will reflect a combination of compression, inline deduplication, and post-process deduplication savings.

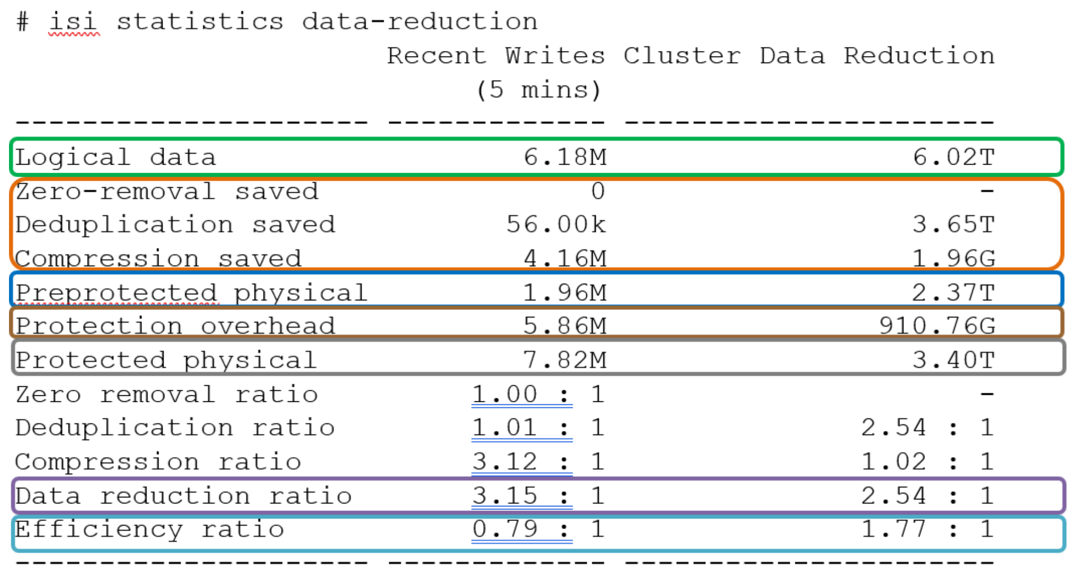
Inline data reduction efficiency reporting

OneFS provides these principal reporting methods for obtaining efficiency information with inline data reduction.

- Using the 'isi statistics data-reduction' CLI command
- Via the 'isi compression' CLI command
- Via the 'isi dedupe' CLI command and WebUI chart
- From the 'isi get -O' CLI command
- Configuring SmartQuotas reporting
- Via the 'isi status' CLI command
- OneFS WebUI Cluster Dashboard Storage Efficiency Summary

Isi statistics data-reduction command

The most comprehensive of the data reduction reporting CLI utilities is the 'isi statistics data-reduction' command. For example:



```
# isi statistics data-reduction
                      Recent Writes Cluster Data Reduction
                           (5 mins)
--------------------- ------------- ----------------------
Logical data                  6.18M                  6.02T
Zero-removal saved                0                      -
Deduplication saved          56.00k                  3.65T
Compression saved             4.16M                  1.96G
Preprotected physical         1.96M                  2.37T
Protection overhead           5.86M                910.76G
Protected physical            7.82M                  3.40T
Zero removal ratio         1.00 : 1                      -
Deduplication ratio        1.01 : 1               2.54 : 1
Compression ratio          3.12 : 1               1.02 : 1
Data reduction ratio       3.15 : 1               2.54 : 1
Efficiency ratio           0.79 : 1               1.77 : 1
--------------------- ------------- ----------------------
```

Figure 14. Example output from the 'isi statistics data-reduction' CLI command

The 'recent writes' data to the left of the output provides precise statistics for the five-minute period prior to running the command. By contrast, the 'cluster data reduction' metrics on the right of the output are slightly less real-time but reflect the overall data and efficiencies across the cluster.

Note: In OneFS 9.1 and earlier, the right-hand column metrics are designated by the 'Est' prefix, denoting an estimated value. However, in OneFS 9.2 and later, the 'logical data' and 'preprotected physical' metrics are now tracked and reported accurately, rather than estimated.

The ratio data in each column is calculated from the values above it. For instance, to calculate the data reduction ratio, the 'logical data' (effective) is divided by the 'preprotected physical' (usable) value. From the output above, this would be:

6.02 / 2.37 = 1.76 Or a **Data Reduction ratio** of **2.54:1**

Similarly, the 'efficiency ratio' is calculated by dividing the 'logical data' (effective) by the 'protected physical' (raw) value. From the output above, this yields:

6.02 / 3.40= 0.97 Or an **Efficiency ratio** of **1.77:1**

Isi compression stats command

From the OneFS CLI, the 'isi compression stats' command provides the option to either view or list compression statistics. When run in 'view' mode, the command returns the compression ratio for both compressed and all writes, plus the percentage of incompressible writes, for a prior five-minute (300 seconds) interval. For example:

```
# isi compression stats view
  stats for 300 seconds at: 2021-04-14 15:46:04 (1618429564)
    compression ratio for compressed writes:  3.12 : 1
    compression ratio for all writes:  3.12 : 1
    incompressible data percent:  6.25%
    total logical blocks: 784
    total physical blocks: 251
    writes for which compression was not attempted:  0.00%
```

Note: If the 'incompressible data' percentage is high in a mixed cluster, there is a strong likelihood that the majority of the writes are going to a non-compression pool.

The 'isi compression stats' CLI command also accepts the 'list' argument, which consolidates a series of recent reports into a list of the compression activity across the file system. For example:

```
# isi compression stats list
Statistic   compression overall    incompressible   logical     physical    compression
            ratio       ratio      %                blocks      blocks      skip %
1618425636  3.07:1      3.07:1     10.59%           68598       22849       1.05%
1618425636  3.20:1      3.20:1     7.73%            4142        1293        0.00%
1618425636  3.14:1      3.14:1     8.24%            352         112         0.00%
1618425636  2.90:1      2.90:1     9.60%            354         122         0.00%
1618425636  1.29:1      1.29:1     75.23%           10839207    8402380     0.00%
```

Figure 15. Example output from the 'isi compression stats list' CLI command.

The 'isi compression stats' data is used for calculating the right-hand side estimated 'Cluster Data Reduction' values in the 'isi statistics data-reduction' command described above. It also provides a count of logical and physical blocks and compression ratios, plus the percentage metrics for incompressible and skipped blocks.

The value in the 'statistic' column at the left of the table represents the epoch timestamp for each sample. This epoch value can be converted to a human readable form using the 'date' CLI command. For example:

```
# date -d 1618425636
Wed Apr 14 15:47:34 EDT 2021
```

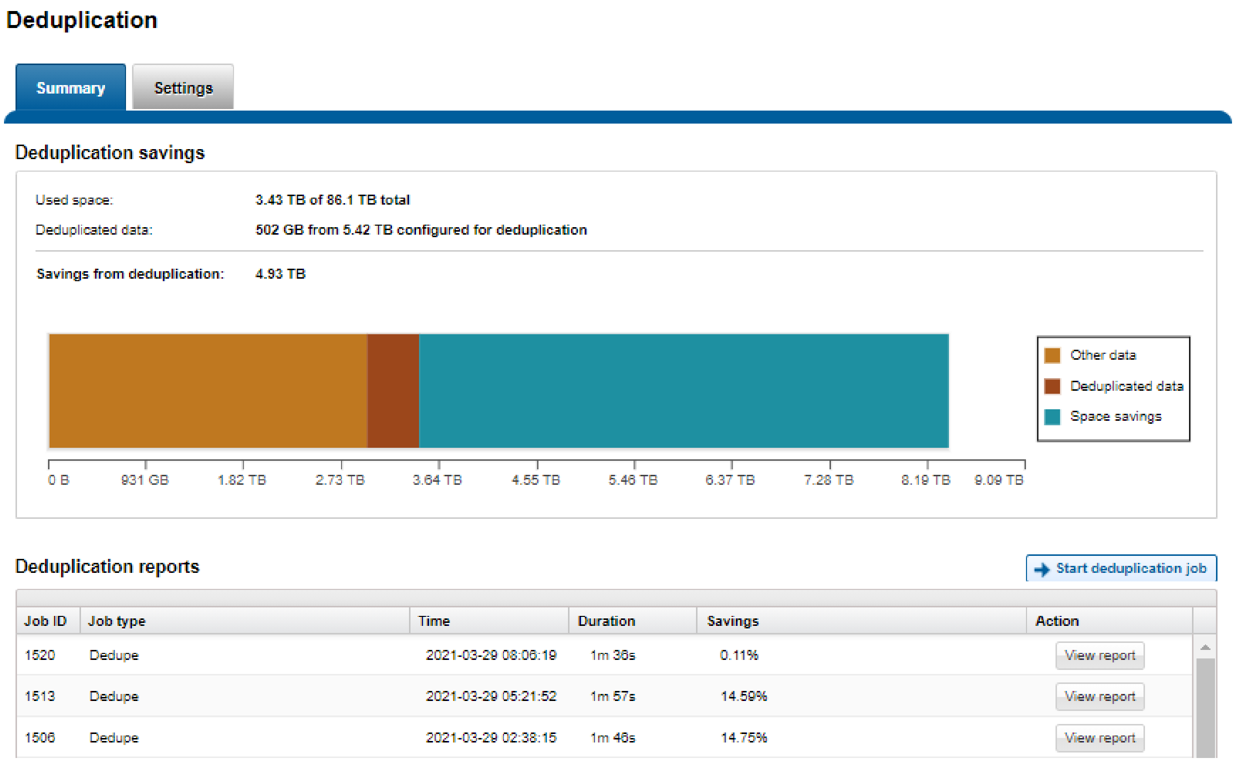
Isi dedupe stats command and WebUI chart

From the OneFS CLI, the 'isi dedupe stats' command provides cluster deduplication data usage and savings statistics, in both logical and physical terms. For example:

```
# isi dedupe stats
```

```
           Cluster Physical Size: 86.14T
               Cluster Used Size: 3.43T
       Logical Size Deduplicated: 4.01T
                  Logical Saving: 3.65T
     Estimated Size Deduplicated: 5.42T
       Estimated Physical Saving: 4.93T
```

In-line dedupe and post-process SmartDedupe both deliver similar results, just at different stages of data ingestion. Since both features use the same core components, the results are combined. As such, the isi dedupe stats output reflects the sum of both inline dedupe and SmartDedupe efficiency. Similarly, the OneFS WebUI's deduplication savings histogram combines the efficiency savings from both inline dedupe and SmartDedupe.



Used space:	3.43 TB of 86.1 TB total
Deduplicated data:	502 GB from 5.42 TB configured for deduplication
Savings from deduplication:	4.93 TB

Job ID	Job type	Time	Duration	Savings	Action
1520	Dedupe	2021-03-29 08:06:19	1m 36s	0.11%	View report
1513	Dedupe	2021-03-29 05:21:52	1m 57s	14.59%	View report
1506	Dedupe	2021-03-29 02:38:15	1m 46s	14.75%	View report

Figure 16. Deduplication cluster capacity savings WebUI chart

---

Note: The deduplication statistics do not include zero block removal savings. Because zero block removal is technically not due to data deduplication, it is tracked separately but included as part of the overall data reduction ratio.

Isi get statistics

OneFS 8.2.1 and later includes a '-O' logical overlay flag to 'isi get' CLI utility for viewing a file's compression details.

For example:

```
# isi get -DDO file1
* Size:           167772160
* PhysicalBlocks: 10314
* LogicalSize:    167772160
```

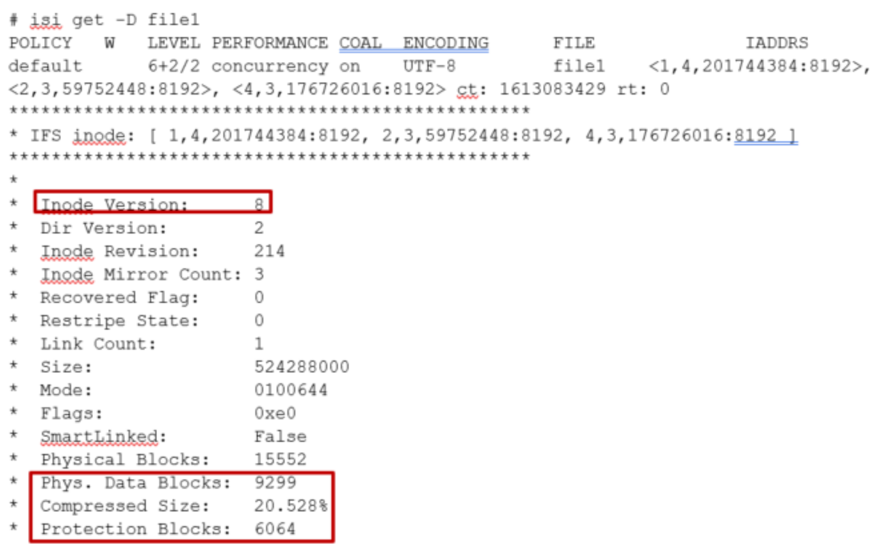
```
PROTECTION GROUPS
lbn0: 6+2/2
2,11,589365248:8192[COMPRESSED]#6
0,0,0:8192[COMPRESSED]#10
2,4,691601408:8192[COMPRESSED]#6
0,0,0:8192[COMPRESSED]#10

Metatree logical blocks:
zero=32 shadow=0 ditto=0 prealloc=0 block=0 compressed=64000
```

The logical overlay information is described under the ‘protection groups’ output. This example shows a compressed file where the sixteen-block chunk is compressed down to six physical blocks (#6) and ten sparse blocks (#10). Under the ‘Metatree logical blocks’ section, a breakdown of the block types and their respective quantities in the file is displayed - including a count of compressed blocks.

When compression has occurred, the ‘df’ CLI command will report a reduction in used disk space and an increase in available space. The ‘du’ CLI command will also report less disk space used. A file that for whatever reason cannot be compressed will be reported as such:

```
4,6,900382720:8192[INCOMPRESSIBLE]#1
```

OneFS 9.2 and later releases use inode version 8, which includes a couple of additional inode delta attributes for storing data reduction metrics. These new attributes are displayed by the ‘isi get -D’ CLI command, and report a file’s physical data blocks, compressed size, and protection blocks. For example:

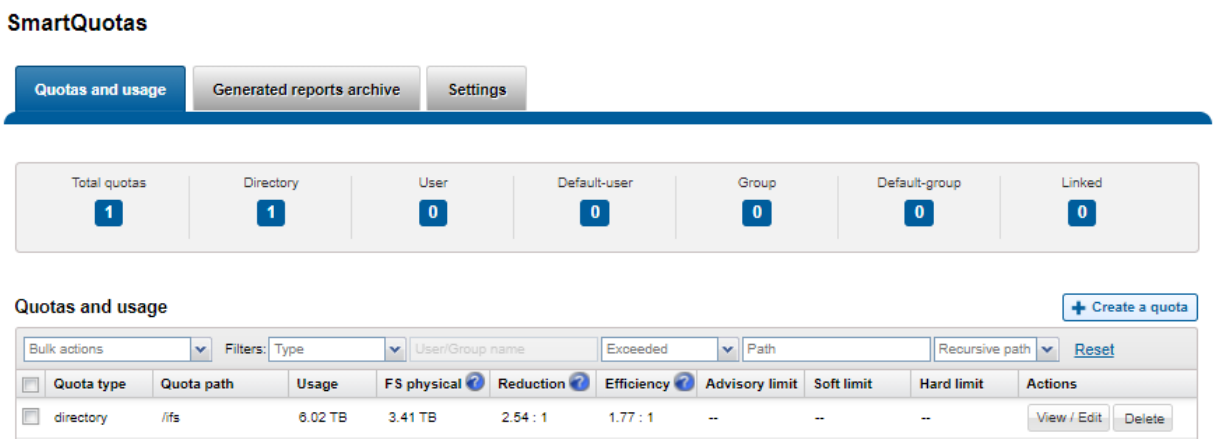


```
# isi get -D file1
POLICY    W   LEVEL PERFORMANCE COAL  ENCODING      FILE              IADDRS
default       6+2/2 concurrency on    UTF-8         file1     <1,4,201744384:8192>,
<2,3,59752448:8192>, <4,3,176726016:8192> ct: 1613083429 rt: 0
*************************************************
* IFS inode: [ 1,4,201744384:8192, 2,3,59752448:8192, 4,3,176726016:8192 ]
*************************************************
*
*  Inode Version:      8
*  Dir Version:        2
*  Inode Revision:     214
*  Inode Mirror Count: 3
*  Recovered Flag:     0
*  Restripe State:     0
*  Link Count:         1
*  Size:               524288000
*  Mode:               0100644
*  Flags:              0xe0
*  SmartLinked:        False
*  Physical Blocks:    15552
*  Phys. Data Blocks:  9299
*  Compressed Size:    20.528%
*  Protection Blocks:  6064
```

Figure 17. Example output from the ‘isi gett’ CLI command.

SmartQuotas data reduction efficiency reporting

In OneFS 8.2.1 and later, OneFS SmartQuotas has been enhanced to report the capacity saving from inline data reduction as a storage efficiency ratio. SmartQuotas reports efficiency as a ratio across the desired data set as specified in the quota path field. The efficiency ratio is for the full quota directory and its contents, including any overhead, and reflects the net efficiency of compression and deduplication. On a cluster with licensed and configured SmartQuotas, this efficiency ratio can be easily viewed from the WebUI by browsing to 'File System > SmartQuotas > Quotas and Usage'. In OneFS 9.2 and later, in addition to the storage efficiency ratio, the data reduction ratio is also displayed.



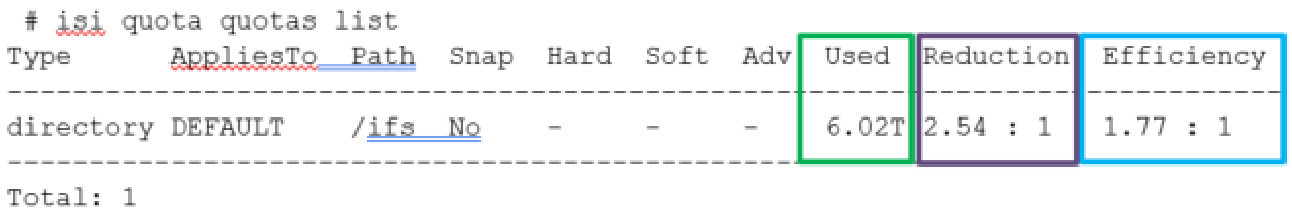
Total quotas	Directory	User	Default-user	Group	Default-group	Linked
1	1	0	0	0	0	0

Quotas and usage

Quota type	Quota path	Usage	FS physical	Reduction	Efficiency	Advisory limit	Soft limit	Hard limit	Actions
directory	/ifs	6.02 TB	3.41 TB	2.54 : 1	1.77 : 1	--	--	--	View / Edit Delete

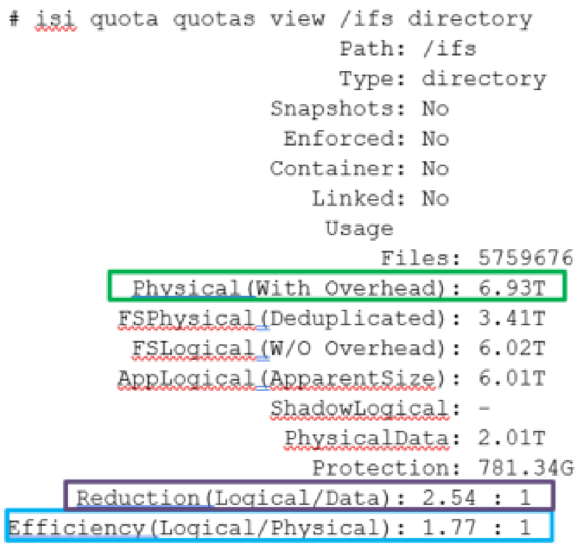
Figure 18. OneFS WebUI SmartQuotas quotas and usage status detailing data reduction and efficiency ratios

Similarly, the same data can be accessed from the OneFS command line by the 'isi quota quotas list' CLI command. For example:



```
 # isi quota quotas list
Type      AppliesTo  Path  Snap  Hard  Soft  Adv  Used   Reduction   Efficiency
-------------------------------------------------------------------------------
directory DEFAULT    /ifs  No    -     -     -    6.02T  2.54 : 1    1.77 : 1
-------------------------------------------------------------------------------
Total: 1
```

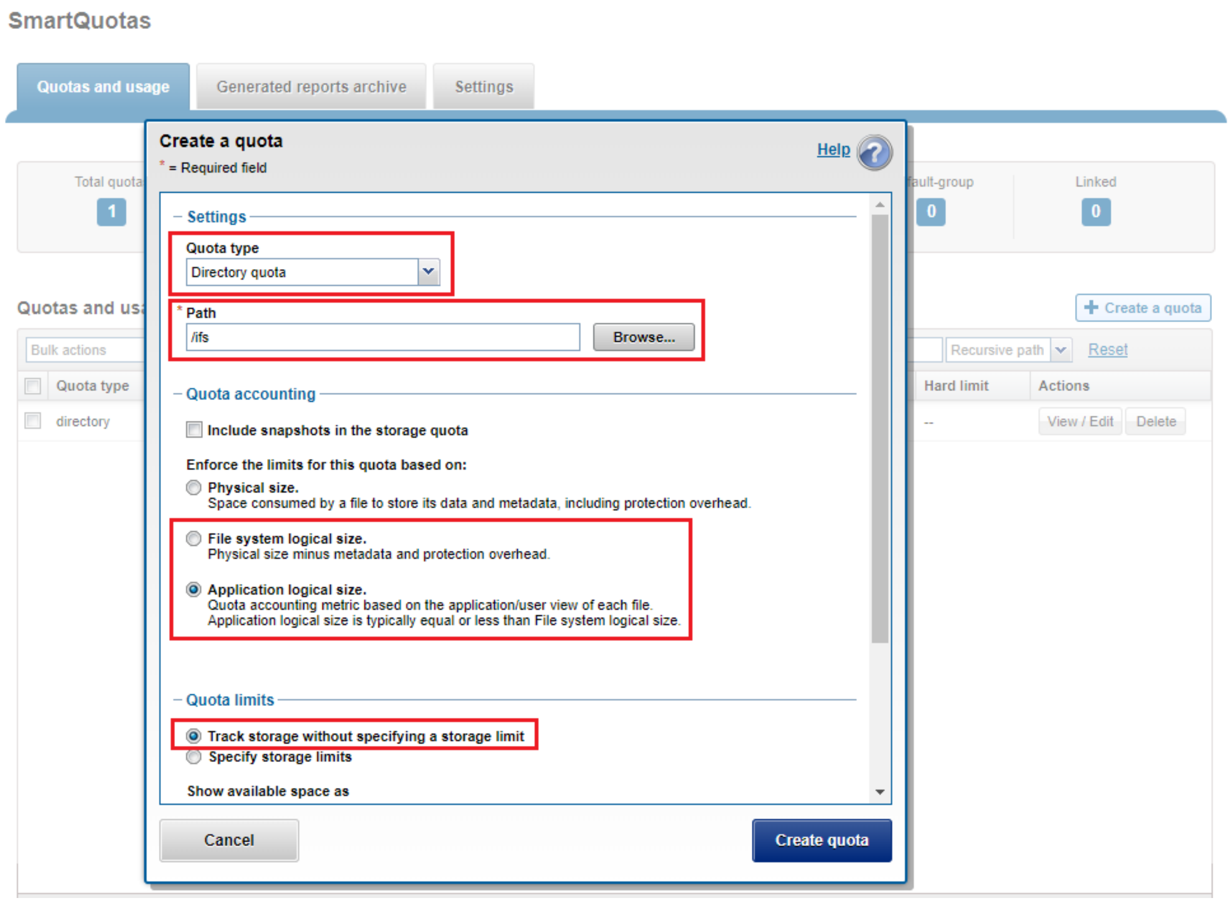
Example output from the 'isi quota quotas list' CLI command. More detail, including both the physical (raw) and logical (effective) data capacities, is also available using the '`isi quota quotas view <path> <type>`' CLI command. For example:



```
# isi quota quotas view /ifs directory
                        Path: /ifs
                        Type: directory
                   Snapshots: No
                    Enforced: No
                   Container: No
                      Linked: No
                       Usage
                           Files: 5759676
         Physical(With Overhead): 6.93T
        FSPhysical(Deduplicated): 3.41T
         FSLogical(W/O Overhead): 6.02T
        AppLogical(ApparentSize): 6.01T
                   ShadowLogical: -
                    PhysicalData: 2.01T
                      Protection: 781.34G
     Reduction(Logical/Data): 2.54 : 1
Efficiency(Logical/Physical): 1.77 : 1
```

Example output from the ‘isi quota quotas view’ CLI command. To configure SmartQuotas for inline data efficiency reporting, create a directory quota at the top-level file system directory of interest, for example /ifs. Creating and configuring a directory quota is a simple procedure and can be performed from the WebUI, as follows:

Browse to ‘File System > SmartQuotas > Quotas and Usage’ and select ‘Create a Quota’. In the create pane, field, set the Quota type to ‘Directory quota’, add the preferred top-level path to report on, select ‘File system logical size’ for Quota Accounting, and set the Quota Limits to ‘Track storage without specifying a storage limit’. Finally, select the ‘Create Quota’ button to confirm the configuration and activate the new directory quota.



Create a quota

* = Required field

Help

Settings

Quota type: Directory quota

* Path: /ifs [Browse...]

Quota accounting

☐ Include snapshots in the storage quota

Enforce the limits for this quota based on:

○ Physical size.
Space consumed by a file to store its data and metadata, including protection overhead.

○ File system logical size.
Physical size minus metadata and protection overhead.

● Application logical size.
Quota accounting metric based on the application/user view of each file.
Application logical size is typically equal or less than File system logical size.

Quota limits

● Track storage without specifying a storage limit

○ Specify storage limits

Show available space as

[Cancel] [Create quota]

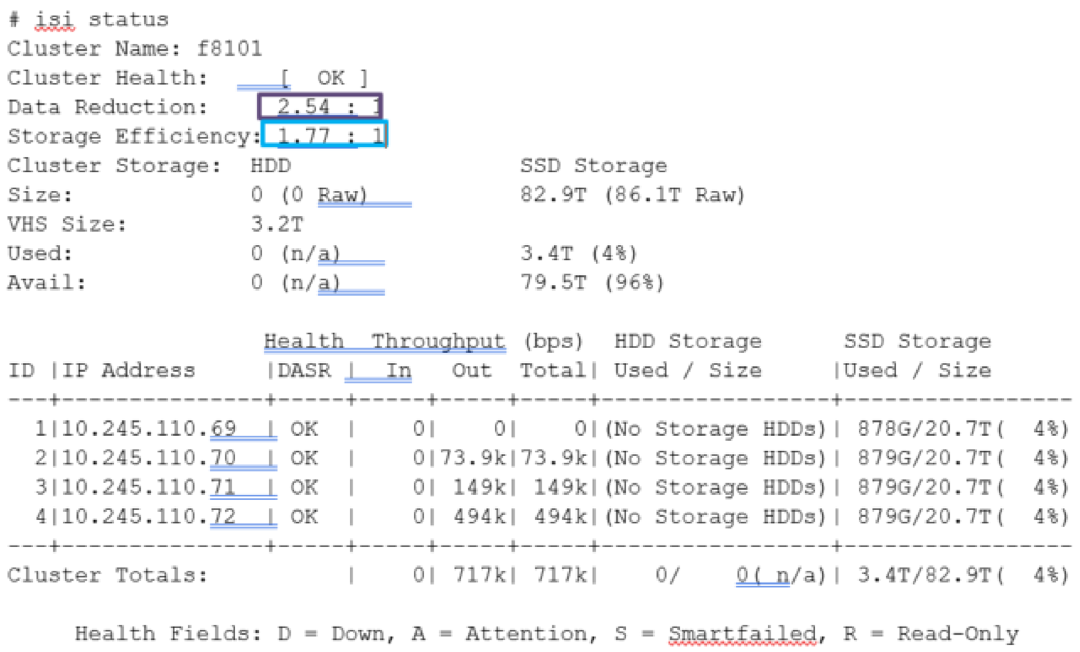
Figure 19. OneFS WebUI SmartQuotas directory quota configuration

The efficiency ratio is a single, current-in time efficiency metric that is calculated per quota directory and includes the sum of inline compression, zero block removal, inline dedupe and SmartDedupe. This is in contrast to a history of stats over time, as reported in the ‘isi statistics data-reduction’ CLI command output, described above. As such, the efficiency ratio for the entire quota directory will reflect what is actually there.

Note: The quota directory efficiency ratio and other statistics are not available through the platform API as of OneFS 9.0.

Isi status command

The ‘isi status’ CLI command output includes a ‘Data Reduction’ field:

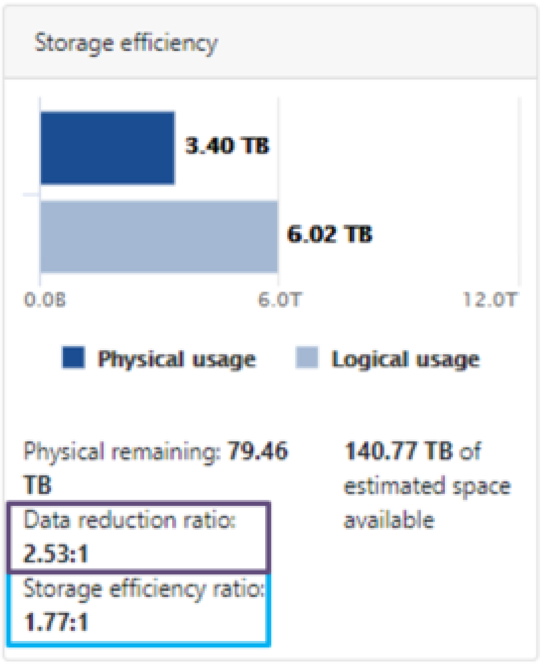


```
# isi status
Cluster Name: f8101
Cluster Health:     [  OK ]
Data Reduction:     2.54 : 1
Storage Efficiency: 1.77 : 1
Cluster Storage:  HDD                 SSD Storage
Size:             0 (0 Raw)           82.9T (86.1T Raw)
VHS Size:         3.2T
Used:             0 (n/a)             3.4T (4%)
Avail:            0 (n/a)             79.5T (96%)

                   Health  Throughput (bps)  HDD Storage      SSD Storage
ID |IP Address     |DASR |  In   Out  Total| Used / Size     |Used / Size
---+---------------+-----+-----+-----+-----+-----------------+-----------------
  1|10.245.110.69  | OK  |    0|    0|    0|(No Storage HDDs)| 878G/20.7T(  4%)
  2|10.245.110.70  | OK  |    0|73.9k|73.9k|(No Storage HDDs)| 879G/20.7T(  4%)
  3|10.245.110.71  | OK  |    0| 149k| 149k|(No Storage HDDs)| 879G/20.7T(  4%)
  4|10.245.110.72  | OK  |    0| 494k| 494k|(No Storage HDDs)| 879G/20.7T(  4%)
---+---------------+-----+-----+-----+-----+-----------------+-----------------
Cluster Totals:          |    0| 717k| 717k|    0/     0( n/a)| 3.4T/82.9T(  4%)

     Health Fields: D = Down, A = Attention, S = Smartfailed, R = Read-Only
```

Figure 20. Example output from the 'isi status' CLI command

OneFS WebUI cluster dashboard storage efficiency summary

In OneFS 8.2.1 and later, the OneFS WebUI cluster dashboard now displays a storage efficiency tile, which shows physical and logical space utilization histograms and reports the capacity saving from inline data reduction as a storage efficiency ratio. In OneFS 9.2 and later, a data reduction ratio is also included in the dashboard view. This cluster status view is displayed by default upon opening the OneFS WebUI in a browser and can be easily accessed by browsing to 'File System > Dashboard > Cluster Overview'.



Physical remaining: 79.46 TB

140.77 TB of estimated space available

Data reduction ratio: 2.53:1

Storage efficiency ratio: 1.77:1

Figure 21. OneFS WebUI cluster status dashboard – storage efficiency summary tile

---

Note: All of the above storage efficiency tools are available on any cluster running OneFS 8.2.1 and later. However, the inline compression metrics are only relevant for clusters containing compression node pools.

Performance with inline data reduction

As with most things in life, data efficiency is a compromise. To gain increased levels of storage efficiency, additional cluster resources (CPU, memory, and disk IO) are used to execute the compressing and deduping and re-inflating of files. As such, the following factors can affect the performance of inline data reduction and the I/O performance of compressed and deduplicated pools:

- Application and the type of dataset being used
- Data access pattern (for example, sequential versus random access, the size of the I/O)
- Compressibility and duplicity of the data
- Amount of total data
- Average file size
- Nature of the data layout
- Hardware platform: the amount of CPU, RAM, and type of storage in the system
- Amount of load on the system
- Level of protection

Note: Clearly, hardware offload compression will perform better, both in terms of speed and efficiency, than the software fallback option – both on F810 nodes where the hardware compression engine has been disabled, and on all other nodes types where software data reduction is the only available option.

Another important performance impact consideration with inline data efficiency is the potential for data fragmentation. After compression or deduplication, files that previously enjoyed contiguous on-disk layout will often have chunks spread across less optimal file system regions. This can lead to slightly increased latencies when accessing these files directly from disk, rather than from cache.

Because inline data reduction is a data efficiency feature rather than performance enhancing tool, in most cases the consideration will be around cluster impact management. This is from both the client data access performance front and from the data reduction execution perspective, as additional cluster resources are consumed when shrinking and inflating files.

With inline data reduction enabled, highly incompressible data sets may experience a small performance penalty. Compression adds latency to the write path. If you consider that initially, writes go to memory (journal), simply performing compression is not free, and that added latency can impact performance. Workloads performing small, random operations will likely see a small performance degradation. Conversely, for highly compressible and duplicate data there may be a performance boost.

Since they reside on the same card, the compression FPGA engine shares PCI-e bandwidth with the node’s backend Ethernet interfaces. In general, there is plenty of bandwidth available. However, a best practice is to run incompressible performance streaming workflows on F810 nodes with inline data reduction disabled to avoid any potential bandwidth limits.

In general, rehydration (that is, un-deduplication) requires considerably less overhead than compression.

Note: When considering effective usable space on a cluster with inline data reduction enabled, bear in mind that every capacity saving from file compression and deduplication also serves to reduce the per-TB compute ratio (CPU, memory, and so on). For performance workloads, the recommendation is to size for performance (IOPS, throughput, and so on) rather than effective capacity.

Similarly, it is challenging to broadly characterize the inline dedupe performance overhead with any accuracy because it depends on various factors, including the duplicity of the data set, and whether matches are found against other LINs or SINs. Workloads requiring a large amount of deduplication might see an impact of 5-10%, but they enjoy an attractive efficiency ratio. In contrast, certain other workloads may see a slight performance gain because of inline dedupe. If there is block scanning but no deduplication to perform, the overhead is typically in the 1-2% range.

Typically, SmartDedupe space savings in addition to inline deduplication fail a cost benefit analysis against performance trade-offs. Minimally intrusive Cost-Benefit analysis can be performed in the following manner:

- With only inline deduplication in operation, obtain a performance baseline of user sensitive Key Performance Indicators (KPIs).

- While monitoring the chosen KPIs, run the DedupeAssessment job to obtain an estimate of additional space savings. Note that that assessment job puts less load on the cluster than the regular SmartDedupe job.
- Review estimated space saving vs performance changes but be aware that SmartDedupe will add more load than observed during the DedupeAssessment job run.
- Enable SmartDedupe if estimated space savings offer business benefit above performance trade-off.

Inline data reduction licensing

In-line data reduction is included as a core component of OneFS on the F910, F900, F810, F710, F600, F210, F200, H700/7000, H5600, and A300/3000 hardware platforms and does not require a product license key to activate. In-line compression is enabled by default, and inline deduplication can be activated by the following CLI command:

```
# isi dedupe inline settings modify --enabled=True
```

Note: An active SmartQuotas license is required to use quota reporting. A SmartQuotas license key can be purchased through your Dell account team. An unlicensed cluster will show a SmartQuotas warning until a valid product license has been purchased and applied to the cluster.

License keys can be easily added in the 'Activate License' section of the OneFS WebUI, accessed by browsing to Cluster Management > Licensing.

Inline data reduction and workflows

Below are some examples of typical space reclamation levels that have been achieved with OneFS inline data efficiency.

Note: These data efficiency space savings values are provided solely as rough guidance. Because no two data sets are alike (unless they are replicated), actual results can and will vary considerably from these examples.

Table 9. Typical workload space savings with inline data reduction

Workflow/data type	Typical efficiency ratio	Typical space savings
Home Directories / File Shares	1.3:1	25%
Engineering Source Code	1.4:1	30%
EDA Data	2:1	50%
Genomics data	2.2:1	55%
Oil and gas	1.4:1	30%
Pre-compressed data	N/A	No savings

Note: Tests of various datasets have demonstrated that data efficiency ratios can easily range from 1:1 (no reduction) to over 3:1.

Inline compression estimation with Live Optics Dossier

The Dell Live Optics Dossier utility can be used to estimate the potential benefits of OneFS' inline data reduction on a data set. Dossier is available for Windows and has no dependency on a Dell PowerScale cluster. This makes it useful for analyzing and estimating efficiency across real data in place, without the need for copying data onto a cluster.

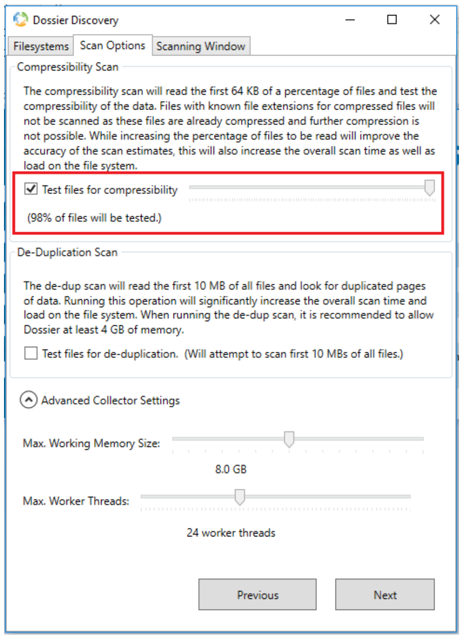
Dossier operates in three phases:

Table 10. Live Optics Dossier phases

Dossier phase	Description
Discovery	Users manually browse and select root folders on the local host to analyze.
Collection	Once the paths to folders have been selected, Dossier will begin walking the file system trees for the target folders. This process will likely take up to several hours for large file systems. Walking the filesystem has a similar impact to a malware/anti-virus scan in terms of the CPU, memory, and disk resources that will be used during the collection. A series of customizable options allows the user to deselect more invasive operations and governs the CPU and memory resources allocated to the Dossier collector.
Reporting	Users upload the resulting '.dossier' file to create a PowerPoint report.

To obtain a Live Optics Dossier report, first download, extract, and run the Dossier collector. Local and remote UNC paths can be added for scanning. Ensure that you are authenticated to the desired UNC path before adding it to Dossier's 'custom paths' configuration.

Note: The Dossier compression option only processes the first 64KB of each file to determine its compressibility. Also, the default configuration samples only 5% of the dataset, but this is configurable with a slider. Increasing this value improves the accuracy of the estimation report, albeit at the expense of extended job execution time.



Compressibility Scan

The compressibility scan will read the first 64 KB of a percentage of files and test the compressibility of the data. Files with known file extensions for compressed files will not be scanned as these files are already compressed and further compression is not possible. While increasing the percentage of files to be read will improve the accuracy of the scan estimates, this will also increase the overall scan time as well as load on the file system.

☑ Test files for compressibility

(98% of files will be tested.)

De-Duplication Scan

The de-dup scan will read the first 10 MB of all files and look for duplicated pages of data. Running this operation will significantly increase the overall scan time and load on the file system. When running the de-dup scan, it is recommended to allow Dossier at least 4 GB of memory.

☐ Test files for de-duplication. (Will attempt to scan first 10 MBs of all files.)

Advanced Collector Settings

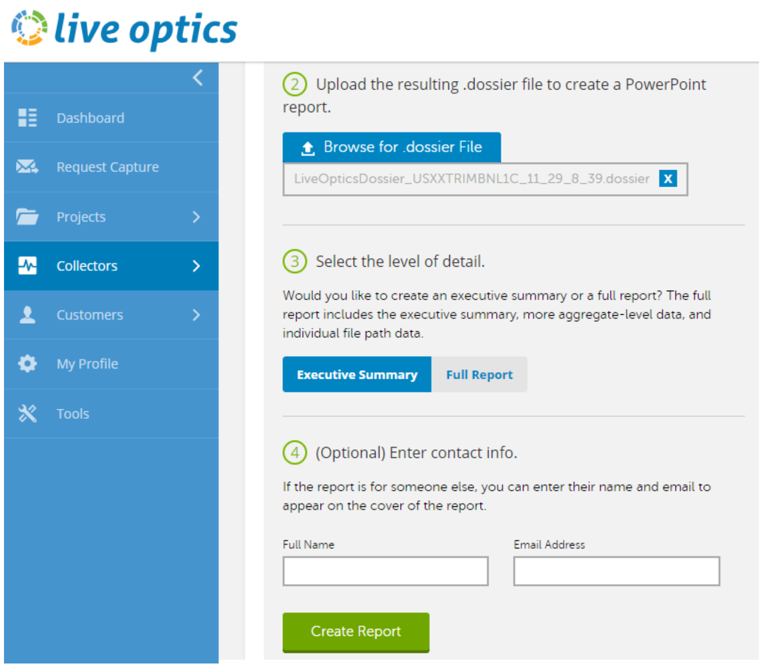
Max. Working Memory Size: 8.0 GB

Max. Worker Threads: 24 worker threads

Figure 22. Dossier discovery configuration

The compressibility scan executes rapidly, with minimal CPU and memory resource consumption. It also provides thread and memory usage controls, progress reporting, and a scheduling option to allow throttling of scanning during heavy usage windows.

When the scan is complete, a '*.dossier' file is generated. This file is then uploaded to the Live Optics website:



2 Upload the resulting .dossier file to create a PowerPoint report.

LiveOpticsDossier_USXXTRIMBNL1C_11_29_8_39.dossier

3 Select the level of detail.

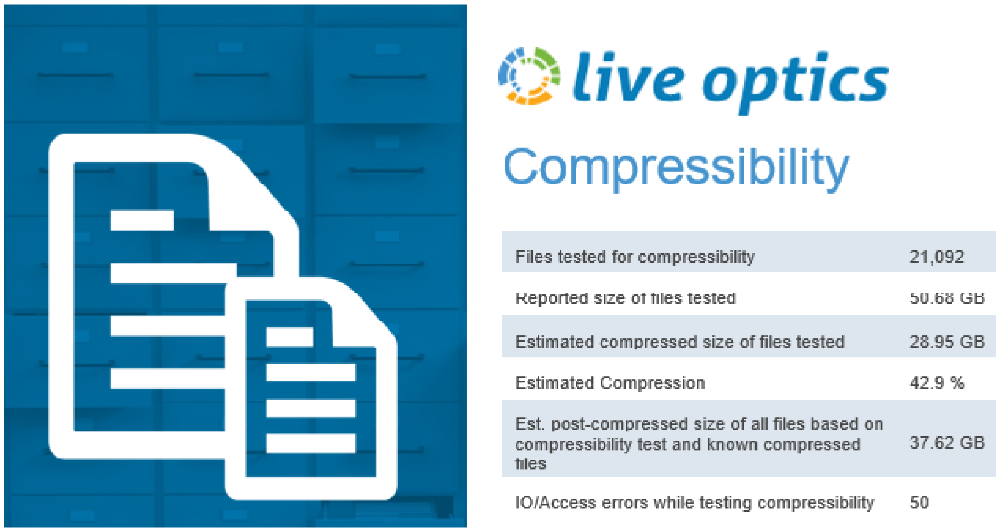
Would you like to create an executive summary or a full report? The full report includes the executive summary, more aggregate-level data, and individual file path data.

4 (Optional) Enter contact info.

If the report is for someone else, you can enter their name and email to appear on the cover of the report.

Figure 23. Live Optics Dossier data upload

Once uploaded and processed, a PowerPoint report is generated in real time and delivered by email.



Files tested for compressibility	21,092
Reported size of files tested	50.68 GB
Estimated compressed size of files tested	28.95 GB
Estimated Compression	42.9 %
Est. post-compressed size of all files based on compressibility test and known compressed files	37.62 GB
IO/Access errors while testing compressibility	50

Figure 24. Dossier compressibility report

Compression reports are easy to comprehend. If multiple SMB shares or paths are scanned, a summary is generated at the beginning of the report, followed by the details of each individually selected path.

Live Optics Dossier can be found at URL https://app.liveoptics.com/tools/dossier.

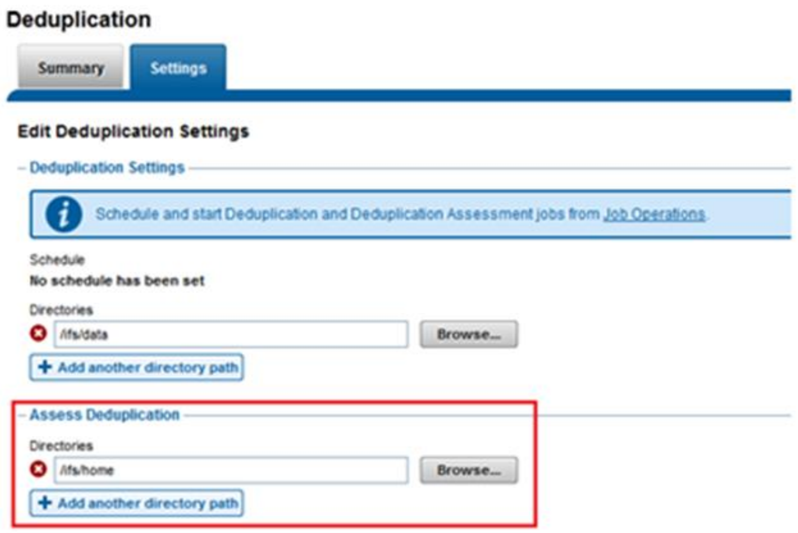
Documentation is at: https://support.liveoptics.com/hc/en-us/articles/229590207-Dossier-User-Guide

When running the Live Optics Dossier tool, please keep the following considerations in mind:

- Does not provide the same algorithm as the OneFS hardware inline compression.
- Dossier looks at the software compression, not the hardware compression. So actual results will generally be better than Dossier report.
- There will be some data for which dossier overestimates compression, for example with files whose first blocks are significantly more compressible than later blocks.
- Intended to be run against any SMB shares on any storage array or DAS. No NFS export support.
- Dossier tool can take a significant amount of time to run against a large data set.
- By default, it only samples a portion (first 64KB) of the data, so results can be inaccurate.
- Dossier does not attempt to compress files with certain known extensions that are uncompressible.
- Dossier assessment tool only provides the size of the uncompressed and compressed data. It does not provide performance estimates of different compression algorithms.

Inline deduplication efficiency estimation

A dry run Dedupe Assessment job is provided to help estimate the amount of space savings that will be seen on a dataset. Run against a specific directory or set of directories on a cluster, the dedupe assessment job reports a total potential space savings. The assessment job uses a separate configuration. It also does not require a product license and can be run prior to purchasing F910, F900, F810, F710, F600, F210, F200, F700/7000, H5600, and A300/3000 hardware to determine whether deduplication is appropriate for a particular data set or environment.



Edit Deduplication Settings

Deduplication Settings

Schedule and start Deduplication and Deduplication Assessment jobs from Job Operations.

Schedule
No schedule has been set

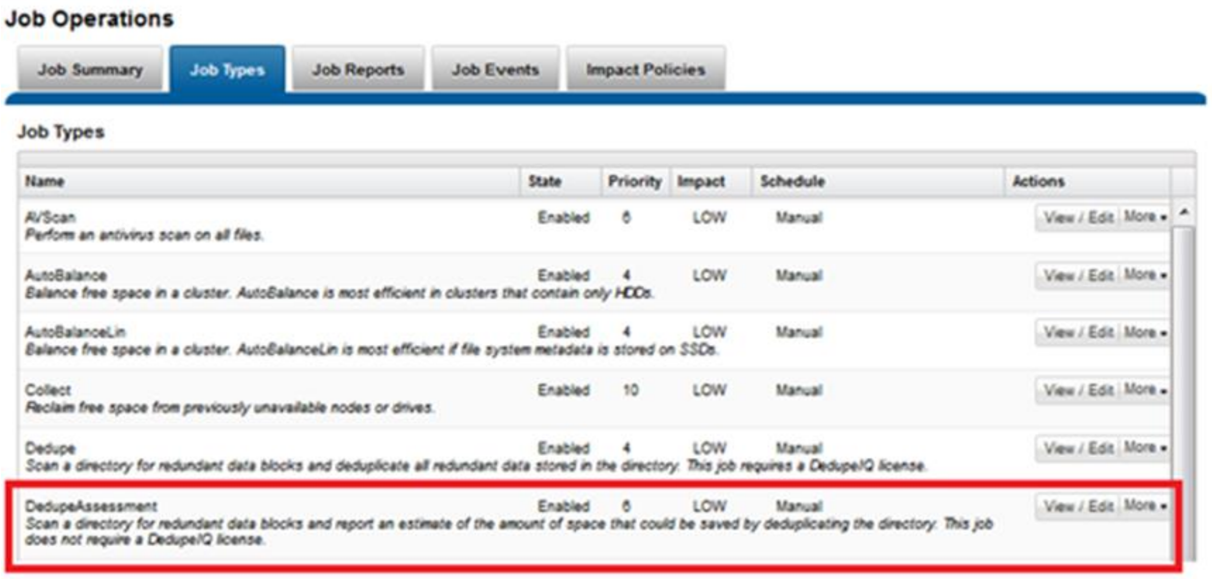
Directories
/ifs/data

Assess Deduplication

Directories
/ifs/home

Figure 25. Deduplication assessment job configuration

The dedupe assessment job uses a separate index table to both inline dedupe and SmartDedupe. For efficiency, the assessment job also samples fewer candidate blocks and does not actually perform deduplication. Using the sampling and consolidation statistics, the job provides a report which estimates the total dedupe space savings in bytes.



Job Types

Name	State	Priority	Impact	Schedule	Actions
AVScan *Perform an antivirus scan on all files.*	Enabled	6	LOW	Manual	View / Edit More
AutoBalance *Balance free space in a cluster. AutoBalance is most efficient in clusters that contain only HDDs.*	Enabled	4	LOW	Manual	View / Edit More
AutoBalanceLin *Balance free space in a cluster. AutoBalanceLin is most efficient if file system metadata is stored on SSDs.*	Enabled	4	LOW	Manual	View / Edit More
Collect *Reclaim free space from previously unavailable nodes or drives.*	Enabled	10	LOW	Manual	View / Edit More
Dedupe *Scan a directory for redundant data blocks and deduplicate all redundant data stored in the directory. This job requires a DedupeIQ license.*	Enabled	4	LOW	Manual	View / Edit More
DedupeAssessment *Scan a directory for redundant data blocks and report an estimate of the amount of space that could be saved by deduplicating the directory. This job does not require a DedupeIQ license.*	Enabled	6	LOW	Manual	View / Edit More

Figure 26. Dedupe assessment job control using the OneFS WebUI

The dedupe assessment job can also be run from the OneFS command line (CLI):

```
# isi job jobs start DedupeAssessment
```

Alternatively, inline deduplication can be enabled in assessment mode:

```
# isi dedupe inline settings modify –mode assess
```

When the job has finished, review the following three metrics from each node:

```
# sysctl efs.sfm.inline_dedupe.stats.zero_block
# sysctl efs.sfm.inline_dedupe.stats.dedupe_block
# sysctl efs.sfm.inline_dedupe.stats.write_block
```

The formula to calculate the estimated dedupe rate from these statistics is:

dedupe_block / write_block * 100 = dedupe%

The dedupe assessment does not differentiate the case of a fresh run from the case where a previous SmartDedupe job has already performed some sharing on the files in that directory. Dell Technologies recommends that the user should run the assessment job once on a specific directory, since it does not provide incremental differences between instances of the job.

Inline data reduction and OneFS feature integration

The following table describes the integration, influence, and compatibility between inline data reduction and the various OneFS data services.

Note: Except for the job engine and non-disruptive upgrade (NDU), the following services each require a product license and are not enabled, configured, and active by default on a cluster.

Table 11. OneFS inline data reduction and data services integration

OneFS feature	Detail
SyncIQ	If compressed and/or deduplicated data is replicated to a target cluster with SyncIQ, those files are automatically decompressed and rehydrated on read and transferred and written to the target in their uncompressed form. However, if the target happened to also be a compression node pool, inline data reduction would occur.
NDMP Backup	Because files are backed up as if the files were not compressed or deduplicated, backup and replication operations are not faster for compressed or deduplicated data. OneFS NDMP backup data will not be compressed unless compression is provided by the backup vendor's DMA software. However, compression is often provided natively by the backup tape or VTL device.
SnapshotIQ	Compression will not affect the data stored in a snapshot. However, snapshots can be created of compressed data. If a data reduction tier is added to a cluster that already has a significant amount of data stored in snapshots, it will take time before the snapshot data is affected by compression. Newly created snapshots will contain compressed data, but older snapshots will not. Deduplicated data can also end up in a snapshot if the HEAD file is deduped and then the shadow references are COWed to the snapshot. While OneFS inline compression works with writable snapshots data, deduplication is not supported, and existing files under writable snapshots will be ignored by inline deduplication. However, inline dedupe can occur on any new files created fresh on the writable snapshot.
SmartLock	In-line data reduction is compatible with SmartLock, OneFS' data retention and compliance product. Compression and deduplication deliver storage efficiency for immutable archives and write once, read many (or WORM) protected data sets. The F910, F900, F810, F710, F600, F210, F200, F700/7000, H5600, and A300/3000 hardware all support compressing and deduping data in files from a SmartLock/WORM domain, including compressing existing files that are currently stored in an uncompressed state. Since the logical content of the file data is unchanged, WORM compliance is unaffected.

OneFS feature	Detail
SED Encryption	Encryption with SED drives is supported on clusters with F810 nodes running OneFS 8.2.1 and later(15.4TB SSD drives only), H5600 nodes running OneFS 8.2.2 and later, F600 & F200 nodes running OneFS 9.0 and later, F900 nodes running OneFS 9.2 and later, F710 & F210 nodes running OneFS 9.7 and later, and F910 nodes running OneFS 9.8 and later.
SmartQuotas	OneFS SmartQuotas is one of the principal methods for inline data reduction efficiency reporting. Quotas account for compressed files as if they consumed both shared and unshared data. From the quota side, compressed files appear no differently than regular files to standard quota policies.
SmartPools	Compressed files will only reside on compression nodes and will not span SmartPools node pools or tiers. This is to avoid potential performance or protection asymmetry which could occur if portions of a file live on different classes of storage. SmartPooled data will be uncompressed before it is moved, so full uncompressed capacity will be required on the compressed pool.
CloudPools	Although CloudPools can use compression to transfer data to the service provider, in OneFS 8.2.1 and later, compressed or deduplicated data cannot be exported directly from disk without incurring a decompression/compression cycle. CloudPools sees uncompressed data and then re-compresses data. CloudPools uses a different chunk size that inline compression.
Non-disruptive Upgrade	In-line data reduction is only available on F810 nodes in OneFS 8.2.1, H5600 nodes in OneFS 8.2.2, F710, F600, F210, & F200 nodes in OneFS 9.0, F900 nodes in OneFS 9.2, and H700/7000 and A300/3000 nodes in OneFS 9.2.1. Gen6 clusters with an Ethernet backend, running earlier versions of OneFS 8.x code can be non-disruptively upgraded to OneFS 8.2.1 and later.
Non-disruptive Upgrade	In-line data reduction is only available on F810 nodes in OneFS 8.2.1, H5600 nodes in OneFS 8.2.2 and later, F600 & F200 nodes in OneFS 9.0 and later, F900 nodes in OneFS 9.2 and later, and H700/7000 and A300/3000 nodes in OneFS 9.2.1 and later, F710 and F210 nodes in OneFS 9.7 and later, and F910 nodes in OneFS 9.8 and later. Gen6 clusters with an Ethernet backend, running earlier versions of OneFS 8.x code can be non-disruptively upgraded to OneFS 8.2.1 and later.
File Clones	File cloning places data in shadow stores and notifies SmartDedupe by flagging the inode of the cloned LIN so that SmartDedupe samples the shadow references (normally it skips them).

<table>
<tr><th>OneFS feature</th><th>Detail</th></tr>
<tr><td>SmartDedupe</td><td>SmartDedupe post process dedupe is compatible with inline data reduction and vice versa. In-line compression can compress OneFS shadow stores. However, for SmartDedupe to process compressed data, the SmartDedupe job will have to decompress it first in order to perform deduplication, which is an addition resource overhead.

Neither SmartDedupe nor inline dedupe are immediately aware of the duplicate matches that each other finds. Both inline dedupe and SmartDedupe could dedupe blocks containing the same data to different shadow store locations, but OneFS is unable to consolidate the shadow blocks together. When blocks are read from a shadow store into L1 cache, they are hashed and added into the in-memory index where they can be used by inline dedupe.

Unlike SmartDedupe, inline dedupe can deduplicate a run of consecutive blocks to a single block in a shadow store. Avoid running both inline dedupe and SmartDedupe on the same node pools.

<table><tr><th>Inline dedupe</th><th>SmartDedupe</th></tr><tr><td>Globally enabled</td><td>Directory tree based</td></tr><tr><td>Will process small files</td><td>Skips files < 32KB (by default)</td></tr><tr><td>Will dedupe sequential runs of blocks of same data to single blocks</td><td>Can only dedupe between files</td></tr><tr><td>Per-node, non-persistent in-memory index</td><td>Large persistent on-disk index</td></tr><tr><td>Can convert copy operations to clone</td><td>Post process only</td></tr><tr><td>Opportunistic</td><td>Exhaustive</td></tr></table></td></tr>
<tr><td>Small File Storage Efficiency (SFSE)</td><td>SFSE is mutually exclusive to all the other shadow store consumers (file clones, inline-dedupe, SmartDedupe). Files can either be packed with SFSE or cloned/deduped, but not both.

<i>Inlined files (small files with their data stored in the inode) will not be deduplicated and non-inlined data files that are once deduped will not inline afterwards.</i></td></tr>
<tr><td>Job Engine</td><td>Only the jobs which access logical data will incur compression and/or decompression overhead costs. These include:

SmartPools, when moving data to or from a compressed node pool.

IntegrityScan, when working on compressed data.

FlexProtect, if there is spillover to another nodepool.

SmartDedupe must decompress data first to perform deduplication, which is an addition resource expense.

Other jobs working on metadata and physical data will be unaffected by inline data reduction.</td></tr>
</table>

OneFS feature	Detail
Job Engine	Only the jobs which access logical data will incur compression and/or decompression overhead costs. These include: SmartPools, when moving data to or from a compressed node pool. IntegrityScan, when working on compressed data. FlexProtect, if there is spillover to another nodepool. SmartDedupe must decompress data first to perform deduplication, which is an addition resource expense. Other jobs working on metadata and physical data will be unaffected by inline data reduction.
DataIQ & InsightIQ	While OneFS, DataIQ, and InsightIQ (PowerScale's multi-cluster reporting and trending analytics tools), are compatible, InsightIQ is not yet fully integrated with inline data reduction and will not report efficiency savings.

Inline data reduction best practices

For optimal cluster performance, Dell Technologies recommends observing the following inline data reduction best practices. Note that some of this information may be covered elsewhere in this paper.

- In-line data reduction is supported on F910, F900, F810, F710, F600, F210, F200, H700/7000, H5600, A300/3000 nodepools only. Legacy F800 nodes cannot be upgraded or converted to F810 nodes.
- Run the assessment tool on a subset of the data to be compressed/deduplicated.
- When replicating compressed and/or deduplicated data, to avoid running out of space on target, it is important to verify that the logical data size (that is, the amount of storage space saved plus the actual storage space consumed) does not exceed the total available space on the target cluster.

Note: In general, additional capacity savings may not warrant the overhead of running SmartDedupe on node pools with inline deduplication enabled. Refer to Performance with inline data reduction for additional details.

- Data reduction can be disabled on a cluster if the overhead of compression and deduplication is considered too high and/or performance is impacted.
- The software data reduction fall back option on F810 nodes is less performant, more resource intensive, and less efficient (lower compression ratio) that hardware data reduction. Consider removing F810 nodes with failing offload hardware from the node pool.
- Run the dedupe assessment job on a single root directory at a time. If multiple directory paths are assessed in the same job, you will not be able to determine which directory should be deduplicated.
- Recommend enabling inline deduplication just prior to rebooting the F910, F900, F810, F710, F600, F210, F200, and H5600 nodes in a cluster.

Inline data reduction considerations

In-line data reduction is supported with the following caveats:

- OneFS 8.2.1 and later will support from 4 to 252 F810 nodes, or 36 chassis, per cluster.

- OneFS 9.0 and later will support from 4 to 252 F810 or H5600 nodes, or from 3 to 252 F600 or F200 nodes per cluster.
- OneFS 9.2 and later will support from 4 to 252 F810 or H5600 nodes, or from 3 to 252 F900, F600, or F200 nodes per cluster.
- OneFS 9.2.1 and later will support from 4 to 252 F810, H5600, F700/7000, or A300/3000 nodes, or from 3 to 252 F900, F600, or F200 nodes per cluster.
- OneFS 9.7 and later will support from 4 to 252 F810, H5600, F700/7000, or A300/3000 nodes, or from 3 to 252 F900, F710, F600, F210, or F200 nodes per cluster.
- OneFS 9.8 and later will support from 4 to 252 F810, H5600, F700/7000, or A300/3000 nodes, or from 3 to 252 F910, F900, F710, F600, F210, or F200 nodes per cluster.
- Data reduction savings depend heavily on factors like the data, cluster composition, and protection level.
- Compressed and deduplicated data does not exit the file system compressed or deduplicated in any shape or form.
- Decompression is substantially less expensive than compression.
- Inline data reduction is exclusive to the F910, F900, F810, F710, F600, F210, F200, H700/7000, H5600, and A300/3000 platforms and does not require a software license.
- There is no compatibility or equivalency between F800 and F810 nodes: They cannot share the same node pool and the F800 nodes will not be able to store compressed data.
- There is no OneFS WebUI support for data reduction. Configuration and management are by the CLI only.
- Partial writes to compression chunks may require reading the entire compression chunk first and decompressing it. This is true even if most of the compression chunk is being written.
- Modifications to compression chunks may require rewriting the entire compression chunk even if only a single logical block is changed.
- Some workloads will have data access patterns that exacerbate the above issues and have the potential to cause more writes than if compression was not used.
- Data integrity failures with compressed data will likely mean that corruption does not affect only a single block but instead the entire compression chunk.
- If SmartPools is used on a mixed cluster containing F910, F900, F810, F710, F600, F210, F200, H700/7000, H5600, or A300/3000 nodes, data will only be compressed and/or inline deduplicated when it physically resides on these node pool(s). If data is tiered to non-compression node pools it will be uncompressed before it is moved, so full uncompressed capacity will be required on the compressed pool. Conversely, if SmartPools moves data between compression pools, for example F810 to F200, the data will remain in a compressed state throughout the transfer.
- Post-process SmartDedupe can run in concert with compression and inline deduplication. It is supported but not widely used. The SmartDedupe job will have

to decompress data first to perform deduplication, which is an addition resource expense.

- Even though compressed files are unintelligible when stored on disk, this does not satisfy the encryption requirements for secure data at rest compliance. However, PowerScale nodes are available with SED drives.
- InsightIQ is not yet fully integrated with inline data reduction and will not report compression savings. This will be addressed in a future release.
- As discussed earlier, inline compression is not free. There is always trade-off between cluster resource consumption (CPU, memory, disk), the potential for data fragmentation and the benefit of increased space efficiency.
- Because compression extends the capacity of a cluster, it also reduces the per-TB compute resource ratio (CPU, memory, I/O, and so on).
- Depending on an application's I/O profile and the effect of In-line data reduction on the data layout, read and write performance and overall space savings can vary considerably.
- OneFS metadata structures (inodes, b-trees, and so on) are not compressed.
- Since compression trades cluster performance for storage capacity savings, compression may not be ideally suited for heavily accessed data, or high-performance workloads.
- SmartFlash (L3) caching is not applicable to F810 nodes since they contain exclusively SSD flash media anyway.
- If a heterogeneous cluster contains F810 nodes plus F800 or other non-compression nodes, data will be uncompressed on the fly when it moves between pools. A non-compression node on the cluster can be an initiator for compressed writes to an F810 or H5600 pool and will perform compression in software. However, this may generate significant overhead for lower powered Archive class nodes.
- In-line dedupe will not permit block sharing across different hardware types or node pools to reduce the risk of performance asymmetry.
- In-line dedupe will not share blocks across files with different protection policies applied.
- OneFS metadata is not deduplicated.
- In-line dedupe will not deduplicate the data stored in a snapshot.
- There is no inline deduplication of CloudPools files.
- In-line dedupe can deduplicate common blocks within the same file and a sequence of consecutive blocks to a single block in a shadow store, resulting in even better data efficiency.
- Compression and/or deduplication of data contained within a writable snapshot is not supported in OneFS 9.3 and later.

OneFS SmartDedupe

Overview

Dell PowerScale 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 post-process, or asynchronous, deduplication.

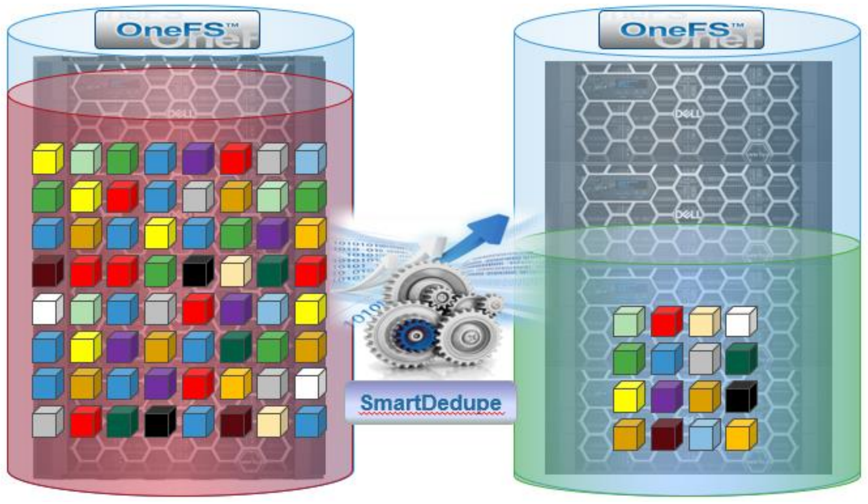


Figure 27. Storage efficiency with SmartDedupe

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 the initial file-write or modify performance is not impacted, since no additional computation is required in the write path.

- SmartDedupe Architecture
- Architecturally, SmartDedupe consists of five principal modules:
- Deduplication Control Path
- Deduplication Job
- Deduplication Engine
- Shadow Store
- Deduplication Infrastructure

The SmartDedupe control path consists of 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.

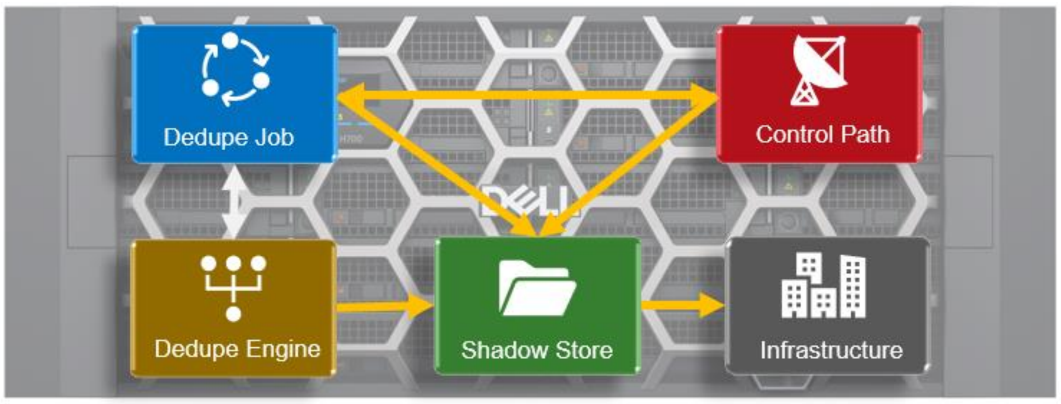


Figure 28. SmartDedupe modular architecture

Deduplication engine – sampling, fingerprinting, and matching

One of the most fundamental components of SmartDedupe, and deduplication in general, is 'fingerprinting'. In this part of the deduplication process, unique digital signatures, or fingerprints, are calculated using the SHA-1 hashing algorithm, one for each 8KB data block in the sampled set.

When SmartDedupe runs for the first time, it scans the data set and selectively samples data blocks from it, creating the fingerprint index. This index contains a sorted list of the digital fingerprints, or hashes, and their associated blocks. After the index is created, the fingerprints are checked for duplicates. When a match is found, during the sharing phase, a byte-by-byte comparison of the blocks is performed to verify that they are identical and to ensure that there are no hash collisions. Then, if they are determined to be identical, the block's pointer is updated to the existing data block and the new, duplicate data block is released.

Hash computation and comparison is only used during the sampling phase. The deduplication job phases are covered in detail below. For the block sharing phase, full data comparison is employed. SmartDedupe also operates on the premise of variable length deduplication, where the block matching window is increased to encompass larger runs of contiguous matching blocks.

Shadow stores

OneFS shadow stores are file system containers that allow data to be stored in a shareable manner. As such, files on OneFS can contain both physical data and pointers, or references, to shared blocks in shadow stores. Shadow stores were introduced in OneFS 7.0, initially supporting OneFS file clones, and there are many overlaps between cloning and deduplicating files. The other main consumer of shadow stores is OneFS Small File Storage Efficiency (SFSE) for archive. This feature maximizes the space utilization of a cluster by decreasing the amount of physical storage required to house a small file archive repository, such as a typical healthcare PACS dataset.

Shadow stores are similar to regular files but are hidden from the file system namespace, so cannot be accessed via a pathname. A shadow store typically grows to a maximum size of 2GB (or about 256K blocks), with each block able to be referenced by 32,000 files. If the reference count limit is reached, a new block is allocated, which may or may not be in the same shadow store. Additionally, shadow stores do not reference other shadow stores. And snapshots of shadow stores are not permitted because the data stored in shadow stores cannot be overwritten.

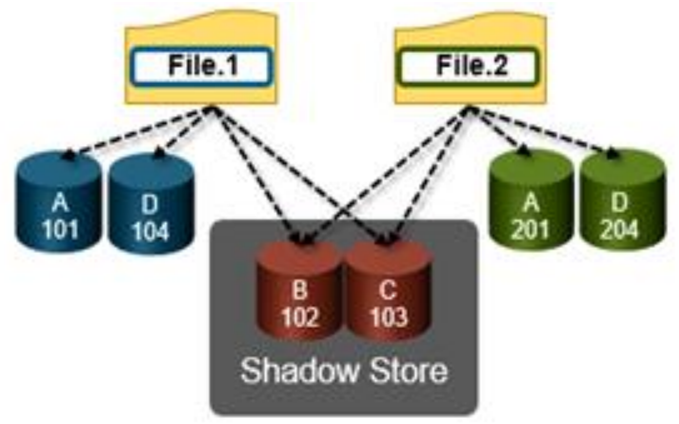
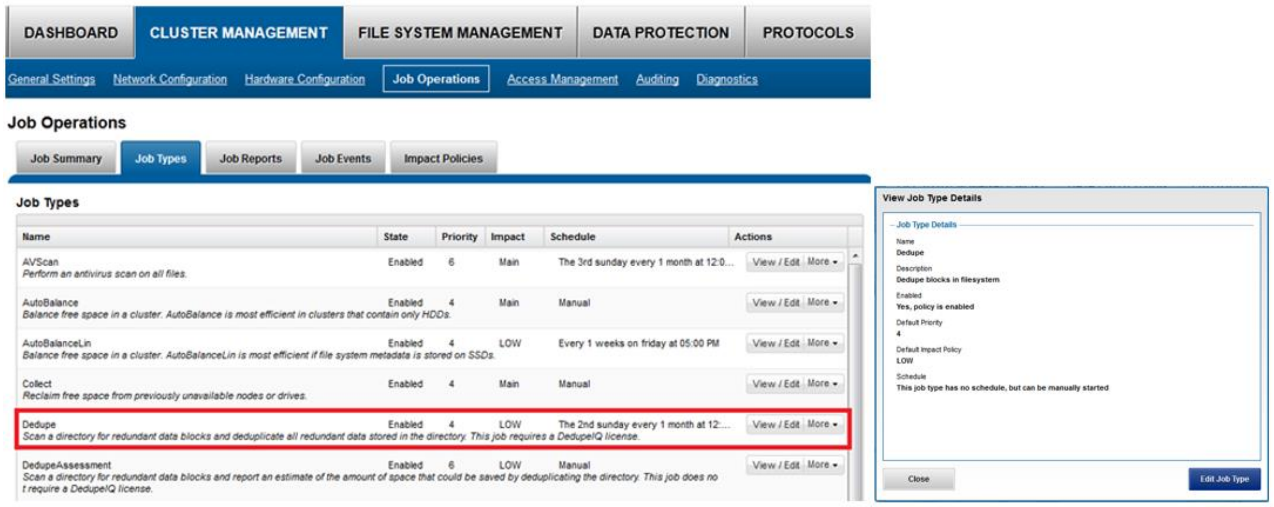


Figure 29. OneFS duplicate block sharing

Deduplication job and infrastructure

Deduplication is performed in parallel across the cluster by the OneFS Job Engine using a dedicated deduplication job, which distributes worker threads across all nodes. This distributed work allocation model allows SmartDedupe to scale linearly as a cluster grows and additional nodes are added.

The control, impact management, monitoring and reporting of the deduplication job is performed by the Job Engine in a similar manner to other storage management and maintenance jobs on the cluster.



Job Operations

Job Types

Name	State	Priority	Impact	Schedule	Actions
AVScan *Perform an antivirus scan on all files.*	Enabled	6	Main	The 3rd sunday every 1 month at 12:0...	View / Edit More
AutoBalance *Balance free space in a cluster. AutoBalance is most efficient in clusters that contain only HDDs.*	Enabled	4	Main	Manual	View / Edit More
AutoBalanceLin *Balance free space in a cluster. AutoBalanceLin is most efficient if file system metadata is stored on SSDs.*	Enabled	4	LOW	Every 1 weeks on friday at 05:00 PM	View / Edit More
Collect *Reclaim free space from previously unavailable nodes or drives.*	Enabled	4	Main	Manual	View / Edit More
Dedupe *Scan a directory for redundant data blocks and deduplicate all redundant data stored in the directory. This job requires a DedupeIQ license.*	Enabled	4	LOW	The 2nd sunday every 1 month at 12:...	View / Edit More
DedupeAssessment *Scan a directory for redundant data blocks and report an estimate of the amount of space that could be saved by deduplicating the directory. This job does no t require a DedupeIQ license.*	Enabled	6	LOW	Manual	View / Edit More

View Job Type Details

Job Type Details

Name
Dedupe

Description
Dedupe blocks in filesystem

Enabled
Yes, policy is enabled

Default Priority
4

Default Impact Policy
LOW

Schedule
This job type has no schedule, but can be manually started

Figure 30. SmartDedupe job control using the OneFS WebUI

While deduplication can run concurrently with other cluster jobs, only a single instance of the deduplication job, albeit with multiple workers, can run at any one time. Although the overall performance impact on a cluster is relatively small, the deduplication job does consume CPU and memory resources.

The primary user facing component of SmartDedupe is the deduplication job. This job performs a file system tree-walk of the configured directory, or multiple directories, hierarchy.

Note: The deduplication job will automatically ignore (not deduplicate) the reserved cluster configuration information located under the /ifs/.ifsvar/ directory, and also any file system snapshots.

Architecturally, the duplication job, and supporting dedupe infrastructure, consist of the following four phases:

- Sampling
- Duplicate Detection
- Block Sharing
- Index Update

These four phases are described in more detail below.

Because the SmartDedupe job is typically long running, each of the phases is executed for a set time period, performing as much work as possible before yielding to the next phase. When all four phases have been run, the job returns to the first phase and continues from where it left off. Incremental dedupe job progress tracking is available from the OneFS Job Engine reporting infrastructure.

Sampling phase

In the sampling phase, SmartDedupe performs a tree-walk of the configured data set in order to collect deduplication candidates for each file.

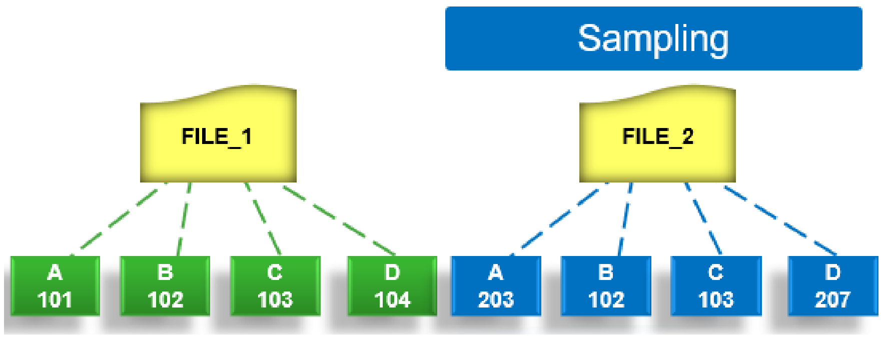


Figure 31. SmartDedupe job sampling phase

The rationale is that a large percentage of shared blocks can be detected with only a smaller sample of data blocks represented in the index table. By default, the sampling phase selects one block from every sixteen blocks of a file as a deduplication candidate. For each candidate, a key/value pair consisting of the block's fingerprint (SHA-1 hash) and file system location (logical inode number and byte offset) is inserted into the index. Once a file has been sampled, the file is flagged and will not be re-scanned until it has been modified. This drastically improves the performance of subsequent deduplication jobs.

Duplicate detection phase

During the duplicate, or commonality, detection phase, the dedupe job scans the index table for fingerprints (or hashes) that match those of the candidate blocks.

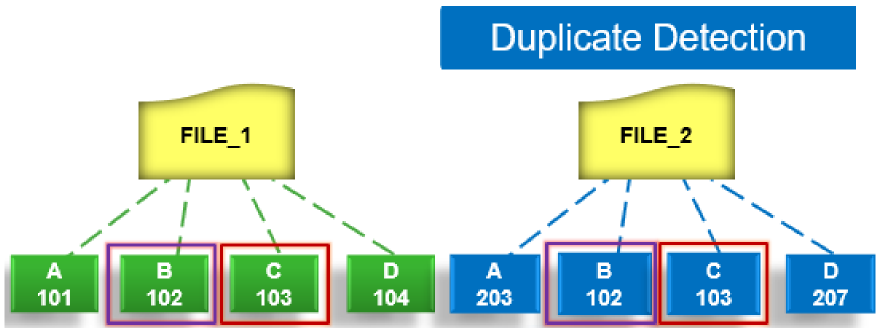


Figure 32. SmartDedupe job duplicate detection phase

If the index entries of two files match, a request entry is generated. In order to improve deduplication efficiency, a request entry also contains pre and post limit information. This information contains the number of blocks in front of and behind the matching block which the block sharing phase should search for a larger matching data chunk, and typically aligns to a OneFS protection group's boundaries.

Block sharing phase

During the block sharing phase, the deduplication job calls into the shadow store library and dedupe infrastructure to perform the sharing of the blocks.

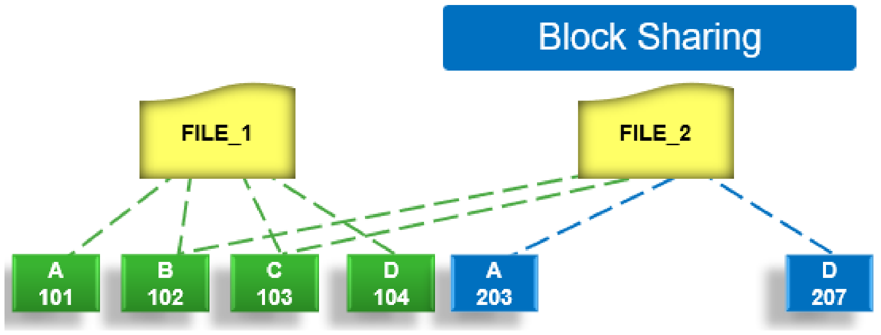


Figure 33. SmartDedupe job block sharing phase

Multiple request entries are consolidated into a single sharing request which is processed by the block sharing phase and ultimately results in the deduplication of the common blocks. The file system searches for contiguous matching regions before and after the matching blocks in the sharing request; if any such regions are found, they too will be shared. Blocks are shared by writing the matching data to a common shadow store and creating references from the original files to this shadow store.

Index update phase

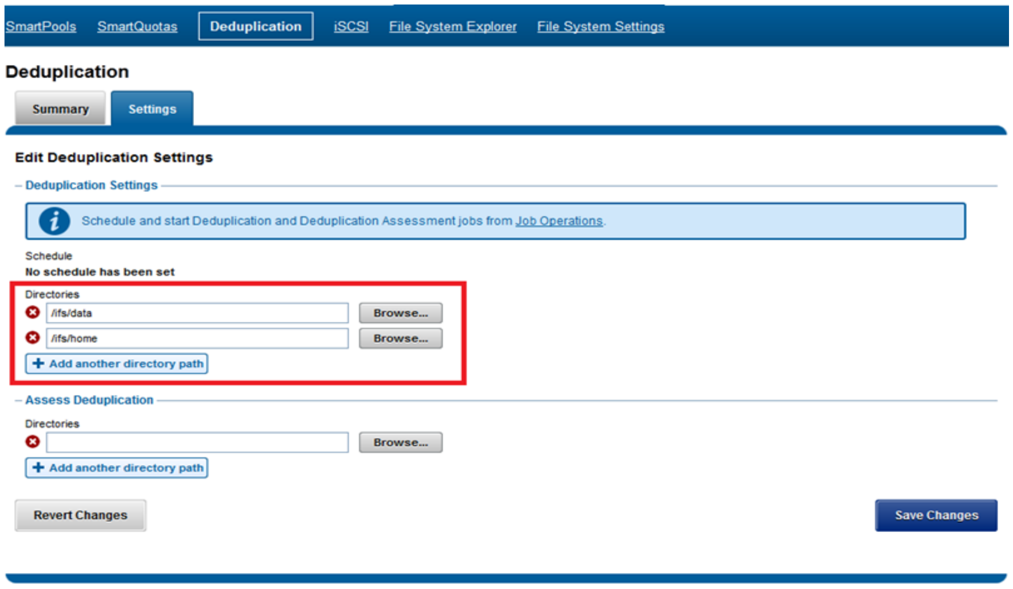
This phase populates the index table with the sampled and matching block information gathered during the previous three phases. After a file has been scanned by the dedupe job, OneFS may not find any matching blocks in other files on the cluster. Once a number of other files have been scanned, if a file continues to not share any blocks with other files on the cluster, OneFS will remove the index entries for that file. This helps prevent OneFS from wasting cluster resources searching for unlikely matches. SmartDedupe scans each file in the specified data set once, after which the file is marked, preventing subsequent dedupe jobs from rescanning the file until it has been modified.

SmartDedupe management

There are two principal elements to managing deduplication in OneFS. The first is the configuration of the SmartDedupe process itself. The second involves the scheduling and execution of the Dedupe job. These are both described below.

Configuring SmartDedupe

SmartDedupe works on data sets which are configured at the directory level, targeting all files and directories under each specified root directory. Multiple directory paths can be specified as part of the overall deduplication job configuration and scheduling.



Deduplication

Edit Deduplication Settings

Deduplication Settings

Schedule and start Deduplication and Deduplication Assessment jobs from Job Operations.

Schedule
No schedule has been set

Directories
/ifs/data
/ifs/home

Assess Deduplication

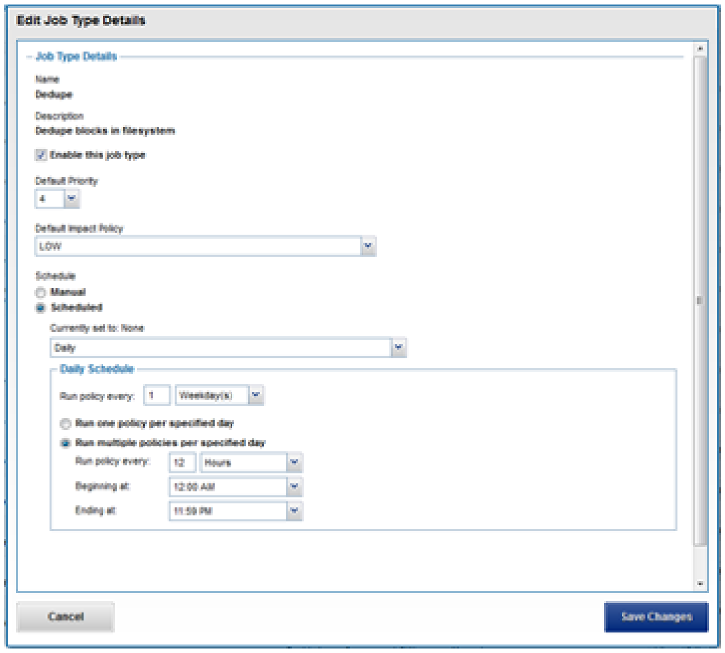
Directories

Figure 34. SmartDedupe configuration using the OneFS WebUI

Note: The permissions required to configure and modify deduplication settings are separate from those needed to run a deduplication job. For example, a user's role must have job engine privileges to run a deduplication job. However, to configure and modify dedupe configuration settings, they must have the deduplication role privileges.

Running SmartDedupe

SmartDedupe can be run either on-demand (started manually) or according to a predefined schedule. This is configured in the cluster management 'Job Operations' section of the WebUI.



Edit Job Type Details

Job Type Details

Name
Dedupe

Description
Dedupe blocks in filesystem

[✓] Enable this job type

Default Priority
4

Default Impact Policy
LOW

Schedule
Manual [✓ Scheduled]

Currently set to: None
Daily

Daily Schedule

Run policy every: 1 Weekday(s)

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

Run policy every: 12 Hours
Beginning at: 12:00 AM
Ending at: 11:59 PM

Figure 35. SmartDedupe job configuration and scheduling using the OneFS WebUI

Dell Technologies recommends scheduling and running deduplication during off-hours, when the rate of data change on the cluster is low. If clients are continually writing to files, the amount of space saved by deduplication will be minimal because the deduplicated blocks are constantly being removed from the shadow store.

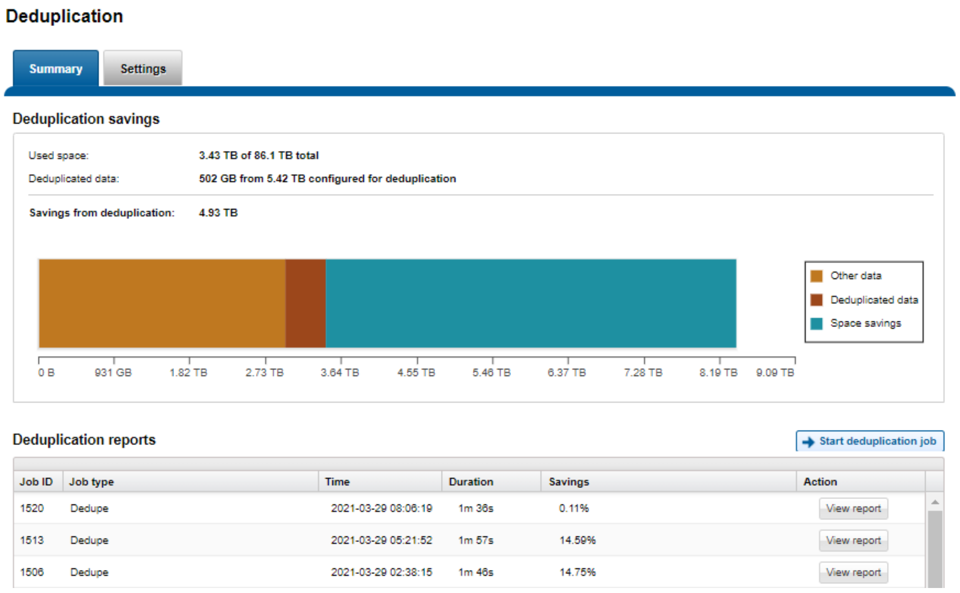
For most clusters, after the initial deduplication job has completed, the recommendation is to run an incremental deduplication job once every two weeks.

SmartDedupe monitoring and reporting

Deduplication efficiency reporting

OneFS provides several command-line and WebUI tools to help assess the benefits of deduplication across a cluster:

1. The amount of disk space saved by SmartDedupe can be determined by viewing the cluster capacity usage chart and deduplication reports summary table in the WebUI. The cluster capacity chart and deduplication reports can be found by browsing to **File System Management** > **Deduplication** > **Summary**.



Deduplication savings

Used space:	3.43 TB of 86.1 TB total
Deduplicated data:	502 GB from 5.42 TB configured for deduplication
Savings from deduplication:	4.93 TB

Deduplication reports

Job ID	Job type	Time	Duration	Savings	Action
1520	Dedupe	2021-03-29 08:06:19	1m 36s	0.11%	View report
1513	Dedupe	2021-03-29 05:21:52	1m 57s	14.59%	View report
1506	Dedupe	2021-03-29 02:38:15	1m 46s	14.75%	View report

Figure 36. SmartDedupe cluster capacity savings WebUI chart

2. In addition to the bar chart and accompanying statistics (above), which graphically represents the data set and space efficiency in actual capacity terms, the dedupe job report overview field also displays the SmartDedupe savings as a percentage.

   SmartDedupe space efficiency metrics are also provided by the '`isi dedupe stats`' CLI command:

```
# isi dedupe stats
      Cluster Physical Size: 86.14T
          Cluster Used Size: 3.43T
  Logical Size Deduplicated: 4.01T
             Logical Saving: 3.65T
Estimated Size Deduplicated: 5.42T
  Estimated Physical Saving: 4.93T
```

Figure 37. SmartDedupe Efficiency Statistics from the CLI

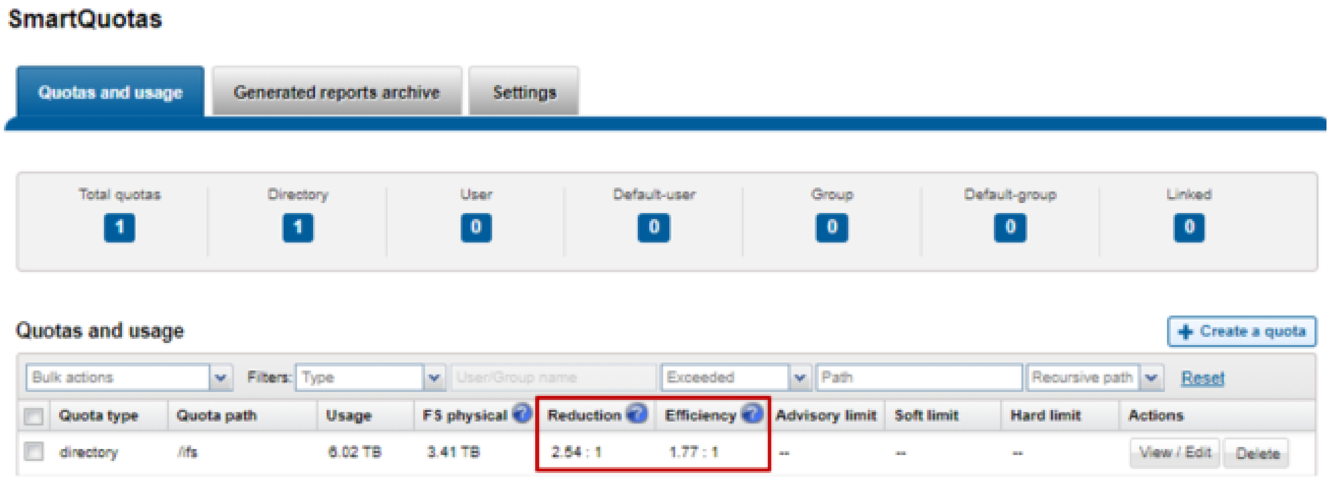
3. The most comprehensive of the data reduction reporting CLI utilities is the 'isi statistics data-reduction' command, which reports deduplication capacity savings and the deduplication ratio. For example:

```
# isi statistics data-reduction
                      Recent Writes Cluster Data Reduction
                           (5 mins)
--------------------- ------------- ----------------------
Logical data                  6.18M                  6.02T
Zero-removal saved                0                      -
Deduplication saved          56.00k                  3.65T
Compression saved             4.16M                  1.96G
Preprotected physical         1.96M                  2.37T
Protection overhead           5.86M                910.76G
Protected physical            7.82M                  3.40T
Zero removal ratio         1.00 : 1                      -
```

```
Deduplication ratio       1.01 : 1             2.54 : 1
Compression ratio         3.12 : 1             1.02 : 1
Data reduction ratio      3.15 : 1             2.54 : 1
Efficiency ratio          0.79 : 1             1.77 : 1
--------------------- ------------- ----------------------
```

The ‘recent writes’ data to the left of the output provides precise statistics for the five-minute period prior to running the command. By contrast, the ‘cluster data reduction’ metrics on the right of the output are slightly less real-time but reflect the overall data and efficiencies across the cluster.

4. In OneFS 8.2.1 and later, SmartQuotas has been enhanced to report the capacity saving from deduplication, and data reduction in general, as a storage efficiency ratio. SmartQuotas reports efficiency as a ratio across the desired data set as specified in the quota path field. The efficiency ratio is for the full quota directory and its contents, including any overhead, and reflects the net efficiency of compression and deduplication. On a cluster with licensed and configured SmartQuotas, this efficiency ratio can be easily viewed from the WebUI by browsing to ‘File System > SmartQuotas > Quotas and Usage’.



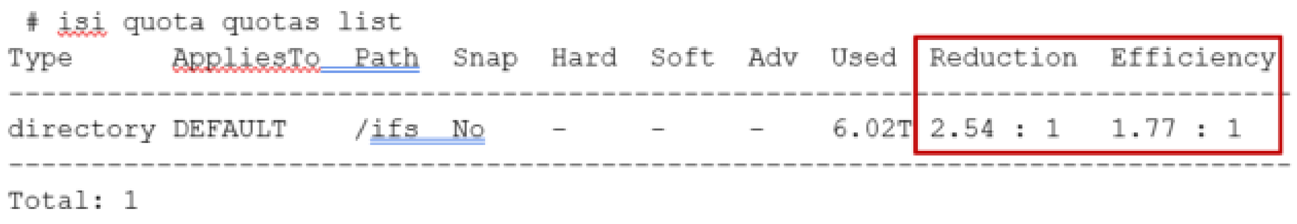
Total quotas	Directory	User	Default-user	Group	Default-group	Linked
1	1	0	0	0	0	0

Quotas and usage

	Quota type	Quota path	Usage	FS physical	Reduction	Efficiency	Advisory limit	Soft limit	Hard limit	Actions
	directory	/ifs	6.02 TB	3.41 TB	2.54 : 1	1.77 : 1	--	--	--	View / Edit Delete

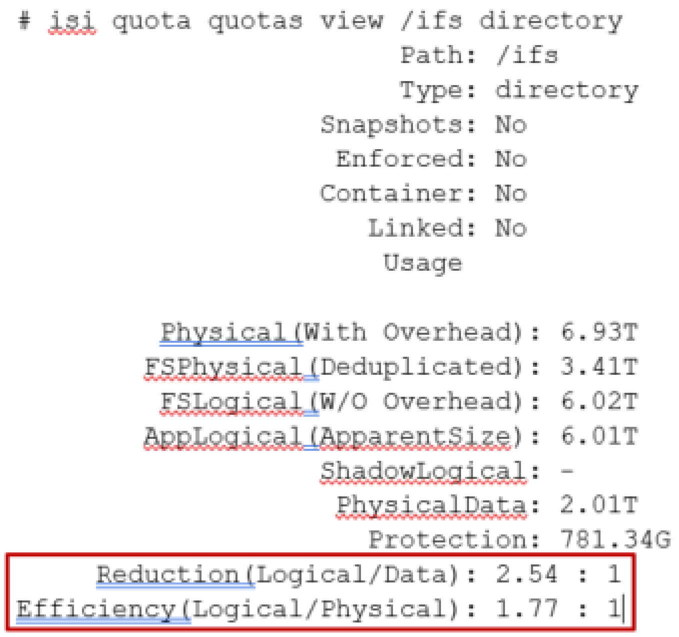
Figure 38. OneFS WebUI SmartQuotas quotas and usage status detailing efficiency and data reduction ratio

5. Similarly, the same data can be accessed from the OneFS command line by the ‘isi quota quotas list’ CLI command. For example:



```
 # isi quota quotas list
Type      AppliesTo  Path  Snap  Hard  Soft  Adv  Used  Reduction  Efficiency
-----------------------------------------------------------------------------
directory DEFAULT    /ifs  No    -     -     -    6.02T 2.54 : 1   1.77 : 1
-----------------------------------------------------------------------------
Total: 1
```

6. Example output from the ‘isi quota quotas list’ CLI command.More detail, including both the physical (raw) and logical (effective) data capacities, is also available using the ‘`isi quota quotas view <path> <type>`’ CLI command. For example:

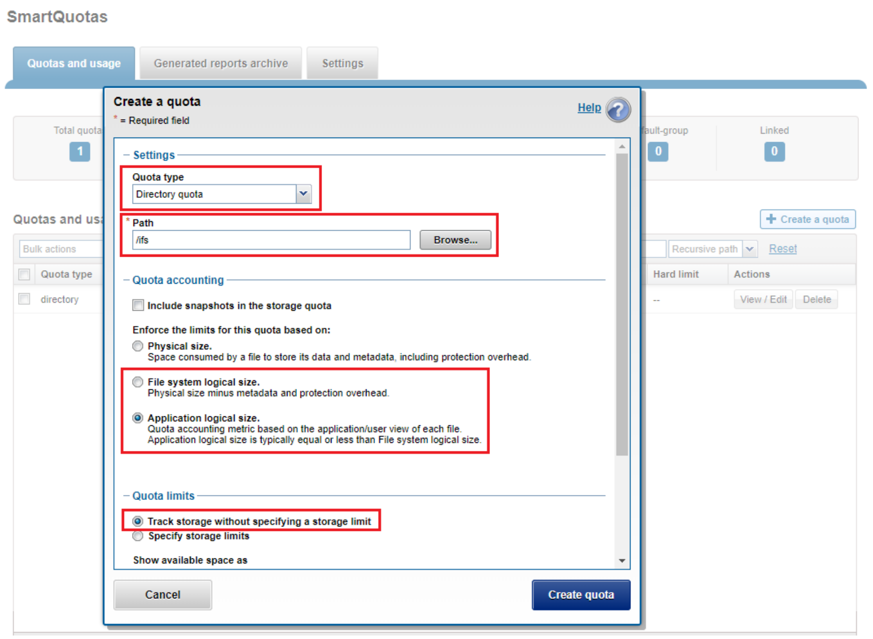


```
# isi quota quotas view /ifs directory
                        Path: /ifs
                        Type: directory
                   Snapshots: No
                    Enforced: No
                   Container: No
                      Linked: No
                       Usage

       Physical(With Overhead): 6.93T
      FSPhysical(Deduplicated): 3.41T
       FSLogical(W/O Overhead): 6.02T
     AppLogical(ApparentSize): 6.01T
                ShadowLogical: -
                 PhysicalData: 2.01T
                   Protection: 781.34G
     Reduction(Logical/Data): 2.54 : 1
Efficiency(Logical/Physical): 1.77 : 1
```

Figure 39. Example output from the 'isi quota quotas view' CLI command

To configure SmartQuotas for data efficiency reporting, create a directory quota at the top-level file system directory of interest, for example /ifs. Creating and configuring a directory quota is a simple procedure and can be performed from the WebUI, as follows:

Browse 'File System > SmartQuotas > Quotas and Usage' and select 'Create a Quota'. In the create pane, field, set the Quota type to 'Directory quota', add the preferred top-level path to report on, select 'File system logical size' for Quota Accounting, and set the Quota Limits to 'Track storage without specifying a storage limit'. Finally, select the 'Create Quota' button to confirm the configuration and activate the new directory quota.



Create a quota

* = Required field

Settings

Quota type: Directory quota

* Path: /ifs

Quota accounting

[ ] Include snapshots in the storage quota

Enforce the limits for this quota based on:

( ) Physical size.
Space consumed by a file to store its data and metadata, including protection overhead.

( ) File system logical size.
Physical size minus metadata and protection overhead.

[✓ Application logical size.]
Quota accounting metric based on the application/user view of each file.
Application logical size is typically equal or less than File system logical size.

Quota limits

[✓ Track storage without specifying a storage limit]

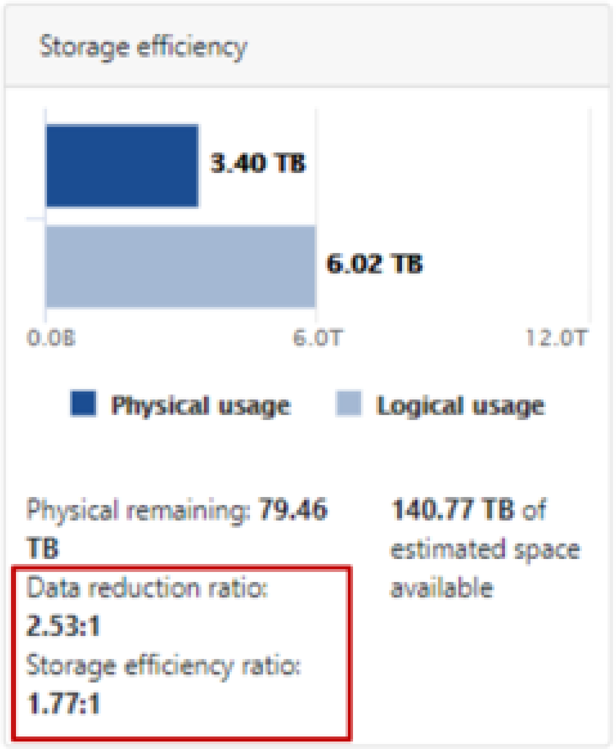
( ) Specify storage limits

Show available space as

Figure 40. OneFS WebUI SmartQuotas directory quota configuration

The efficiency ratio is a single, current-in time efficiency metric that is calculated per quota directory and includes the sum of SmartDedupe plus inline data reduction. This is in contrast to a history of stats over time, as reported in the ‘isi statistics data-reduction’ CLI command output, described above. As such, the efficiency ratio for the entire quota directory will reflect what is there.

7. The OneFS WebUI cluster dashboard also now displays a storage efficiency tile, which shows physical and logical space utilization histograms and reports the capacity saving from inline data reduction as a storage efficiency ratio. This dashboard view is displayed by default when opening the OneFS WebUI in a browser and can be easily accessed by browsing to ‘File System > Dashboard > Cluster Overview’.



Physical remaining: 79.46 TB

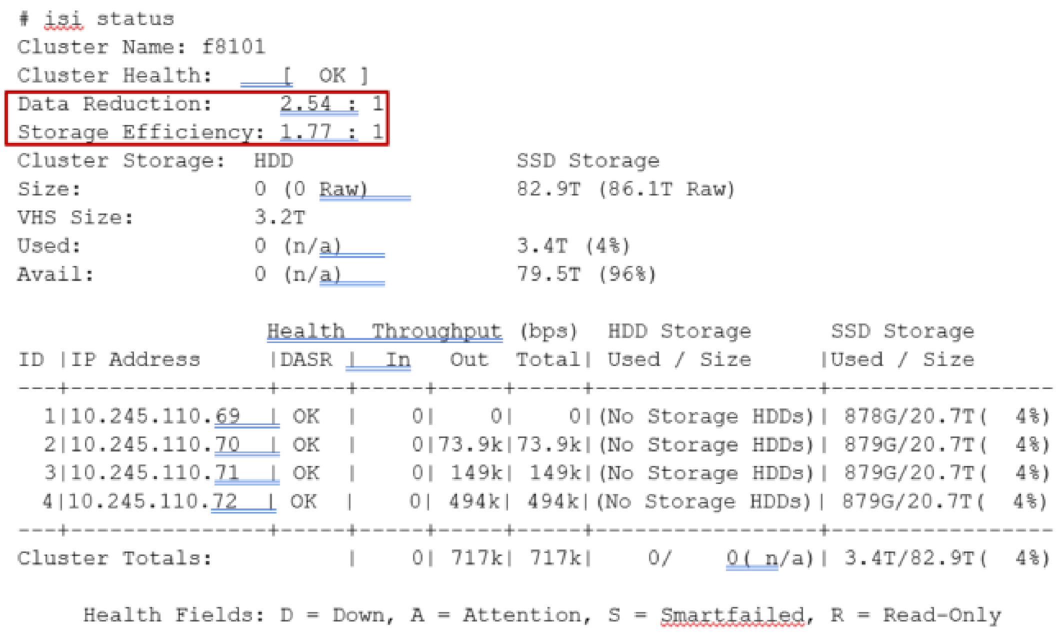
140.77 TB of estimated space available

Data reduction ratio: 2.53:1

Storage efficiency ratio: 1.77:1

Figure 41. OneFS WebUI cluster status dashboard – storage efficiency summary tile

8. Similarly, the ‘isi status’ CLI command output includes a ‘Data Reduction’ field:



```
# isi status
Cluster Name: f8101
Cluster Health:     [  OK ]
Data Reduction:      2.54 : 1
Storage Efficiency: 1.77 : 1
Cluster Storage:  HDD                 SSD Storage
Size:             0 (0 Raw)           82.9T (86.1T Raw)
VHS Size:         3.2T
Used:             0 (n/a)             3.4T (4%)
Avail:            0 (n/a)             79.5T (96%)

                   Health  Throughput (bps)  HDD Storage      SSD Storage
ID |IP Address     |DASR |  In   Out  Total| Used / Size     |Used / Size
---+---------------+-----+-----+-----+-----+-----------------+-----------------
  1|10.245.110.69  | OK  |    0|    0|    0|(No Storage HDDs)| 878G/20.7T(  4%)
  2|10.245.110.70  | OK  |    0|73.9k|73.9k|(No Storage HDDs)| 879G/20.7T(  4%)
  3|10.245.110.71  | OK  |    0| 149k| 149k|(No Storage HDDs)| 879G/20.7T(  4%)
  4|10.245.110.72  | OK  |    0| 494k| 494k|(No Storage HDDs)| 879G/20.7T(  4%)
---+---------------+-----+-----+-----+-----+-----------------+-----------------
Cluster Totals:          |    0| 717k| 717k|    0/     0( n/a)| 3.4T/82.9T(  4%)

     Health Fields: D = Down, A = Attention, S = Smartfailed, R = Read-Only
```

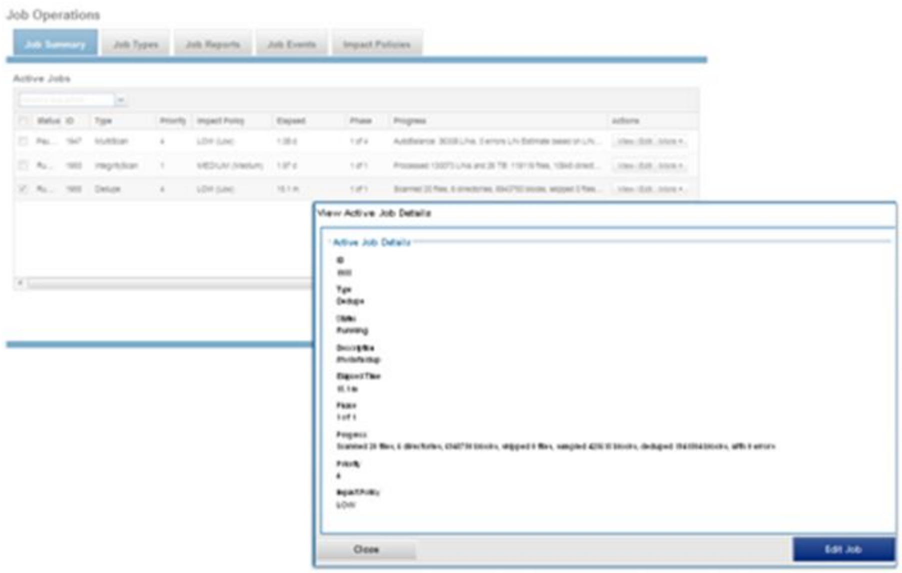
Figure 42. Example output from the ‘isi status’ CLI command showing data reduction and storage efficiency ratios.

SmartDedupe job progress

The Job Engine parallel execution framework provides comprehensive run time and completion reporting for the deduplication job.

While SmartDedupe is underway, job status is available at a glance in the progress column in the active jobs table. This information includes the number of files, directories, and blocks that have been scanned, skipped, and sampled, and any errors that may have been encountered.

Additional progress information is provided in an Active Job Details status update, which includes an estimated completion percentage based on the number of logical inodes (LINs) that have been counted and processed.



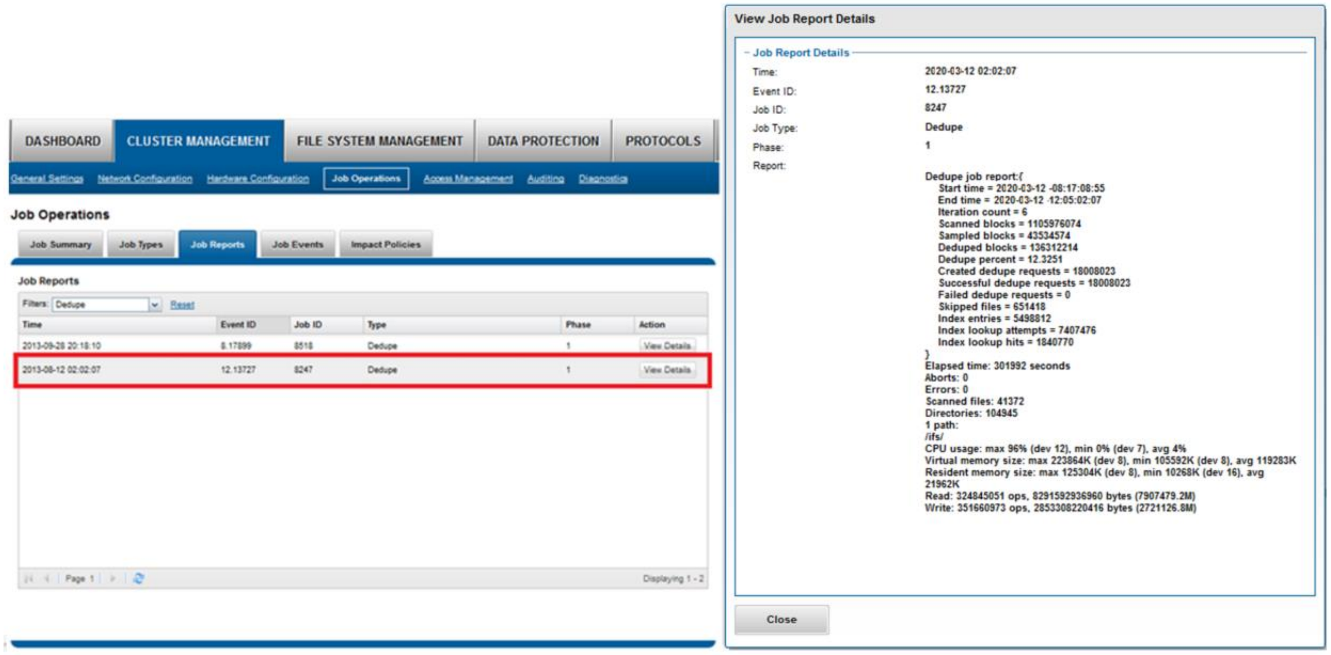
	Status	ID	Type	Priority	Impact Policy	Elapsed	Phase	Progress	Actions
	Pa...	[illegible]	MultiScan	4	LOW (Low)	[illegible]	1 of 4	AutoBalance: [illegible] LINs, [illegible] errors LIN Estimate based on LIN...	View / Edit / More ▾
	Ru...	[illegible]	IntegrityScan	1	MEDIUM (Medium)	[illegible]	1 of 1	Processed [illegible] LINs and [illegible] TB: [illegible] files, [illegible] direct...	View / Edit / More ▾
✓	Ru...	[illegible]	Dedupe	4	LOW (Low)	[illegible]	1 of 1	Scanned [illegible] files, [illegible] directories, [illegible] blocks, skipped [illegible] files...	View / Edit / More ▾

ID
[illegible]
Type
Dedupe
State
Running
Description
[illegible]
Elapsed Time
[illegible]
Phase
1 of 1
Progress
Scanned [illegible] files, [illegible] directories, [illegible] blocks, skipped [illegible] files, sampled [illegible] blocks, deduped [illegible] blocks, with [illegible] errors
Priority
4
Impact Policy
LOW

Figure 43. Example of active job status update

SmartDedupe job reports

After the SmartDedupe job has run to completion, or has been terminated, a full dedupe job report is available. This can be accessed from the WebUI by browsing to **Cluster Management > Job Operations > Job Reports** and selecting 'View Details' action button on the desired Dedupe job line item.



Time	Event ID	Job ID	Type	Phase	Action
2013-09-28 20:18:10	8.17899	8518	Dedupe	1	View Details
2013-08-12 02:02:07	12.13727	8247	Dedupe	1	View Details

Time:	2020-03-12 02:02:07
Event ID:	12.13727
Job ID:	8247
Job Type:	Dedupe
Phase:	1

Report:

```
Dedupe job report:{
    Start time = 2020-03-12 -08:17:08:55
    End time = 2020-03-12 -12:05:02:07
    Iteration count = 6
    Scanned blocks = 1105976074
    Sampled blocks = 43534574
    Deduped blocks = 136312214
    Dedupe percent = 12.3251
    Created dedupe requests = 18008023
    Successful dedupe requests = 18008023
    Failed dedupe requests = 0
    Skipped files = 651418
    Index entries = 5498812
    Index lookup attempts = 7407476
    Index lookup hits = 1840770
}
Elapsed time: 301992 seconds
Aborts: 0
Errors: 0
Scanned files: 41372
Directories: 104945
1 path:
/ifs/
CPU usage: max 96% (dev 12), min 0% (dev 7), avg 4%
Virtual memory size: max 223864K (dev 8), min 105592K (dev 8), avg 119283K
Resident memory size: max 125304K (dev 8), min 10268K (dev 16), avg
21962K
Read: 324845051 ops, 8291592936960 bytes (7907479.2M)
Write: 351660973 ops, 2853308220416 bytes (2721126.8M)
```

Figure 44. Example of WebUI dedupe job report

The job report contains the following relevant dedupe metrics.

Table 12. Dedupe job report statistics

Report field	Description of metric
Start time	When the dedupe job started.
End time	When the dedupe job finished.
Scanned blocks	Total number of blocks scanned under configured path(s).
Sampled blocks	Number of blocks that OneFS created index entries for.
Created dedupe requests	Total number of dedupe requests created. A dedupe request gets created for each matching pair of data blocks. For example, three data blocks all match, two requests are created: One request to pair file1 and file2 together, the other request to pair file2 and file3 together.
Successful dedupe requests	Number of dedupe requests that completed successfully.
Failed dedupe requests	Number of dedupe requests that failed. If a dedupe request fails, it does not mean that the also job failed. A deduplication request can fail for any number of reasons. For example, the file might have been modified since it was sampled.
Skipped files	Number of files that were not scanned by the deduplication job. The primary reason is that the file has already been scanned and has not been modified since. Another reason for a file to be skipped is if it is less than 32KB in size. Such files are considered too small and do not provide enough space saving benefit to offset the fragmentation they will cause.
Index entries	Number of entries that exist in the index.

Report field	Description of metric
Index lookup attempts	Cumulative total number of lookups that have been done by prior and current deduplication jobs. A lookup is when the deduplication job attempts to match a block that has been indexed with a block that hasn't been indexed.
Index lookup hits	Total number of lookup hits that have been done by earlier deduplication jobs plus the number of lookup hits done by this deduplication job. A hit is a match of a sampled block with a block in index.

Dedupe job reports are also available from the CLI by the `isi job reports view <job_id>` command.

Note: From an execution and reporting stance, the Job Engine considers the 'dedupe' job to contain a single process or phase. The Job Engine events list will report that Dedupe Phase1 has ended and succeeded. This indicates that an entire SmartDedupe job, including all four internal dedupe phases (sampling, duplicate detection, block sharing, and index update), has successfully completed.

For example:

```
# isi job events list --job-type dedupe
Time                Message
------------------------------------------------------
2020-02-01T13:39:32 Dedupe[1955] Running
2020-02-01T13:39:32 Dedupe[1955] Phase 1: begin dedupe
2020-02-01T14:20:32 Dedupe[1955] Phase 1: end dedupe
2020-02-01T14:20:32 Dedupe[1955] Phase 1: end dedupe
2020-02-01T14:20:32 Dedupe[1955] Succeeded
```

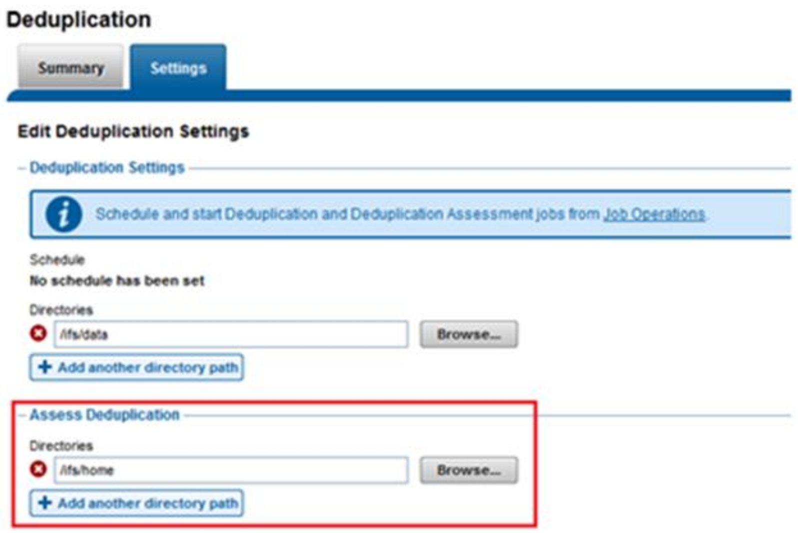
Figure 45. Example of command line (CLI) dedupe job events list

For deduplication reporting across multiple OneFS clusters, SmartDedupe is also integrated with InsightIQ cluster reporting and analysis product. A report detailing the space savings delivered by deduplication is available from InsightIQ's File Systems Analytics module.

Space savings estimation with the SmartDedupe assessment job

To complement the actual Dedupe job, a dry run Dedupe Assessment job is also provided to help estimate the amount of space savings that will be seen by running deduplication on a particular directory or set of directories. The dedupe assessment job reports a total potential space savings. The dedupe assessment does not differentiate the case of a fresh run from the case where a previous dedupe job has already done some sharing on the files in that directory. The assessment job does not provide the incremental differences between instances of this job. Dell Technologies recommends that the user should run the assessment job once on a specific directory before starting an actual dedupe job on that directory.

The assessment job runs similarly to the actual dedupe job but uses a separate configuration. It also does not require a product license and can be run prior to purchasing SmartDedupe in order to determine whether deduplication is appropriate for a particular data set or environment.


Deduplication

Edit Deduplication Settings

Deduplication Settings

Schedule and start Deduplication and Deduplication Assessment jobs from Job Operations.

Schedule
No schedule has been set

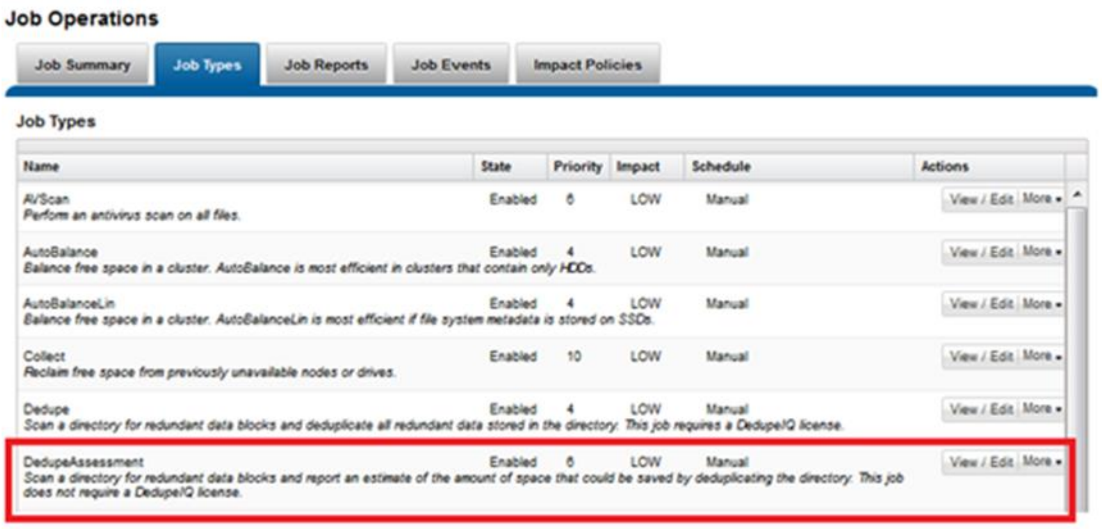
Directories
/ifs/data

Assess Deduplication

Directories
/ifs/home

Figure 46. **Deduplication assessment job configuration**

The dedupe assessment job uses a separate index table. For efficiency, the assessment job also samples fewer candidate blocks than the main dedupe job and does not actually perform deduplication. Using the sampling and consolidation statistics, the job provides a report which estimates the total dedupe space savings in bytes.


Job Operations

Job Types

Name	State	Priority	Impact	Schedule	Actions
AVScan *Perform an antivirus scan on all files.*	Enabled	6	LOW	Manual	View / Edit More ▾
AutoBalance *Balance free space in a cluster. AutoBalance is most efficient in clusters that contain only HDDs.*	Enabled	4	LOW	Manual	View / Edit More ▾
AutoBalanceLin *Balance free space in a cluster. AutoBalanceLin is most efficient if file system metadata is stored on SSDs.*	Enabled	4	LOW	Manual	View / Edit More ▾
Collect *Reclaim free space from previously unavailable nodes or drives.*	Enabled	10	LOW	Manual	View / Edit More ▾
Dedupe *Scan a directory for redundant data blocks and deduplicate all redundant data stored in the directory. This job requires a DedupeIQ license.*	Enabled	4	LOW	Manual	View / Edit More ▾
DedupeAssessment *Scan a directory for redundant data blocks and report an estimate of the amount of space that could be saved by deduplicating the directory. This job does not require a DedupeIQ license.*	Enabled	6	LOW	Manual	View / Edit More ▾

Figure 47. **Dedupe assessment job control using the OneFS WebUI**

Performance with SmartDedupe

As with most things in life, deduplication is a compromise. In order to gain increased levels of storage efficiency, additional cluster resources (CPU, memory, and disk IO) are used to find and execute the sharing of common data blocks.

Another important performance impact consideration with dedupe is the potential for data fragmentation. After deduplication, files that previously enjoyed contiguous on-disk layout will often have chunks spread across less optimal file system regions. This can lead to slightly increased latencies when accessing these files directly from disk, rather than from cache. To help reduce this risk, SmartDedupe will not share blocks across node pools or data tiers and will not attempt to deduplicate files smaller than 32KB in size. On the other end of the spectrum, the largest contiguous region that will be matched is 4MB.

Because deduplication is a data efficiency product rather than performance enhancing tool, in most cases the consideration will be around cluster impact management. This is from both the client data access performance front, since, by design, multiple files will be sharing common data blocks, and also from the dedupe job execution perspective, as additional cluster resources are consumed to detect and share commonality.

The first deduplication job run will often take a substantial amount of time to run, since it must scan all files under the specified directories to generate the initial index and then create the appropriate shadow stores. However, deduplication job performance will typically improve significantly on the second and subsequent job runs (incrementals), once the initial index and the bulk of the shadow stores have already been created.

If incremental deduplication jobs do take a long time to complete, this is most likely indicative of a data set with a high rate of change. If a deduplication job is paused or interrupted, it will automatically resume the scanning process from where it left off.

As mentioned previously, deduplication is a long running process that involves multiple job phases that are run iteratively. SmartDedupe typically processes around 1TB of data per day, per node.

SmartDedupe licensing

SmartDedupe is included as a core component of OneFS but requires a valid product license key in order to activate. This license key can be purchased through your Dell account team. An unlicensed cluster will show a SmartDedupe warning until a valid product license has been purchased and applied to the cluster.

License keys can be easily added in the ‘Activate License’ section of the OneFS WebUI, accessed by browsing to Cluster Management > Licensing.

The SmartDedupe dry-run estimation job can be run without any licensing requirements, allowing an assessment of the potential space savings that a dataset might yield before making the decision to purchase the full product.

Note: The PowerScale H700/7000 and A300/3000 nodes ship with a zero-cost SmartDedupe license and are supported on OneFS 9.2.1 and later. On addition of an H700/7000 or A300/3000 nodepool to an existing cluster without a SmartDedupe license, SmartDedupe will automatically be available for configuration and use across the entire cluster. An alert will be generated warning of unlicensed nodes, but this can be safely ignored and quiesced.

Deduplication efficiency

Deduplication can significantly increase the storage efficiency of data. However, the actual space savings will vary depending on the specific attributes of the data itself. As mentioned above, the deduplication assessment job can be run to help predict the likely space savings that deduplication would provide on a given data set.

Virtual machines files often contain duplicate data, much of which is rarely modified. Deduplicating similar OS type virtual machine images (such as VMware VMDK files that have been block-aligned) can significantly decrease the amount of storage space consumed. However, as noted previously, the potential for performance degradation as a result of block sharing and fragmentation should be carefully considered first.

SmartDedupe does not deduplicate across files that have different protection settings. For example, if two files share blocks, but file1 is parity protected at +2:1, and file2 has its protection set at +3, SmartDedupe will not attempt to deduplicate them. This ensures that all files and their constituent blocks are protected as configured. Also, SmartDedupe will not deduplicate files that are stored on different SmartPools storage tiers or node-pools. For example, if file1 and file2 are stored on tier 1 and tier 2 respectively, and tier1 and tier2 are both protected at 2:1, OneFS will not deduplicate them. This helps guard against performance asynchronicity, where some of a file's blocks could live on a different tier, or class of storage, from the others.

Below are some examples of typical space reclamation levels that have been achieved with SmartDedupe.

Note: These dedupe space savings values are provided solely as rough guidance. Since no two data sets are alike (unless they are replicated), actual results can vary considerably from these examples.

Table 13. Typical workload space savings with SmartDedupe

Workflow/data type	Typical space savings
Virtual Machine Data	35%
Home Directories / File Shares	25%
Email Archive	20%
Engineering Source Code	15%
Media Files	10%

SmartDedupe and OneFS feature integration

SyncIQ replication and SmartDedupe

When deduplicated files are replicated to another cluster using SyncIQ, or backed up to a tape device, the deduplicated files are inflated (or rehydrated) back to their original size, since they no longer share blocks on the target cluster. However, once replicated data has landed, SmartDedupe can be run on the target cluster to provide the same space efficiency benefits as on the source.

Shadow stores are not transferred to target clusters or backup devices. Because of this, deduplicated files do not consume less space than non-deduplicated files when they are replicated or backed up. To avoid running out of space on target clusters or tape devices, it is important to verify that the total amount of storage space saved, and storage space consumed, does not exceed the available space on the target cluster or tape device. To reduce the amount of storage space consumed on a target cluster, you can configure deduplication for the target directories of your replication policies. Although this will deduplicate data on the target directory, it will not allow SyncIQ to transfer shadow stores. Deduplication is still performed post-replication, by a deduplication job running on the target cluster.

Backup and SmartDedupe

Because files are backed up as if the files were not deduplicated, backup and replication operations are not faster for deduplicated data. You can deduplicate data while the data is being replicated or backed up.

Note: OneFS NDMP backup data will not be deduplicated unless deduplication is provided by the backup vendor's DMA software. However, compression is often provided natively by the backup tape or VTL device.

Snapshots and SmartDedupe

SmartDedupe will not deduplicate the data stored in a snapshot. However, snapshots can be created of deduplicated data. If a snapshot is taken of a deduplicated directory, and then the contents of that directory are modified, the shadow stores will be transferred to the snapshot over time. Because of this, more space will be saved on a cluster if deduplication is run prior to enabling snapshots.

If deduplication is enabled on a cluster that already has a significant amount of data stored in snapshots, it will take time before the snapshot data is affected by deduplication. Newly created snapshots will contain deduplicated data, but older snapshots will not.

It is also good practice to revert a snapshot before running a deduplication job. Restoring a snapshot will cause many of the files on the cluster to be overwritten. Any deduplicated files are reverted to normal files if they are overwritten by a snapshot revert. However, once the snapshot revert is complete, deduplication can be run on the directory again and the resulting space savings will persist on the cluster.

Deduplication of writable snapshot data is not supported. SmartDedupe will ignore the files under writable snapshots.

SmartLock and SmartDedupe

SmartDedupe is also fully compatible with SmartLock, OneFS' data retention and compliance solution. SmartDedupe delivers storage efficiency for immutable archives and write once, read many (or WORM) protected data sets.

SmartQuotas and SmartDedupe

OneFS SmartQuotas accounts for deduplicated files as if they consumed both shared and unshared data. From the quota side, deduplicated files appear no differently than regular files to standard quota policies. However, if the quota is configured to include data-protection overhead, the additional space used by the shadow store will not be accounted for by the quota.

SmartPools and SmartDedupe

SmartDedupe does not deduplicate files that span SmartPools node pools or tiers, or that have different protection levels, access patterns, or caching configurations set. This is to avoid potential performance or protection asymmetry which could occur if portions of a file live on different classes of storage.

However, a deduped file that is moved by SmartPools to a different pool or tier will retain the shadow references to the shadow store on the original pool. This breaks the rule for deduping across different disk pool policies, but it is less impactful to do this than rehydrate files that are moved. Further dedupe activity on that file will no longer be allowed to reference any blocks in the original shadow store. The file will need to be deduped against other files in the same disk pool policy. If the file had not yet been deduped, the dedupe index may have knowledge about the file and will still think it is on the original pool. This will be discovered and corrected when a match is made against blocks in the file.

Because the moved file has already been deduped, the dedupe index will have knowledge of the shadow store only. Since the shadow store has not moved, it will not cause problems for further matching. However, if the shadow store is moved as well (but not both files), then a similar situation occurs and the SmartDedupe job will discover this and purge knowledge of the shadow store from the dedupe index.

Inline compression and SmartDedupe

SmartDedupe post process dedupe is compatible with inline compression, currently available on the PowerScale F910, F900, F810, F710, F600, F210, F200, H700/7000, H5600, and A300/3000 platforms, and vice versa. In-line compression can compress OneFS shadow stores. However, in order for SmartDedupe to process compressed data, the SmartDedupe job will have to decompress it first in order to perform deduplication. In general, additional capacity savings may not warrant the overhead of running SmartDedupe on node pools with inline deduplication enabled.

Inline deduplication and SmartDedupe

While OneFS has offered a native file system deduplication solution for several years, until OneFS 8.2.1 this was always accomplished by scanning the data after it has been written to disk, or post-process. With inline data reduction, deduplication is now performed in real time as data is written to the cluster. Storage efficiency is achieved by scanning the data for identical blocks as it is received and then eliminating the duplicates using shadow stores.

Since inline dedupe and SmartDedupe use different hashing algorithms, the indexes for each are not shared directly. However, the work performed by each dedupe solution can be leveraged by each other. For instance, if SmartDedupe writes data to a shadow store, when those blocks are read, the read hashing component of inline dedupe will see those blocks and index them.

When a match is found, inline dedupe performs a byte-by-byte comparison of each block to be shared to avoid the potential for a hash collision. Data is prefetched prior the byte-by-byte check and then compared against the L1 cache buffer directly, avoiding unnecessary data copies and adding minimal overhead. Once the matching blocks have been compared and verified as identical, they are then shared by writing the matching data to a common shadow store and creating references from the original files to this shadow store.

In-line dedupe samples every whole block written and handles each block independently, so it can aggressively locate block duplicity. If a contiguous run of matching blocks is detected, inline dedupe will merge the results into regions and process them efficiently.

In-line dedupe is also detects dedupe opportunities from the read path, and blocks are hashed as they are read into L1 cache and inserted into the index. If an existing entry exists for that hash, inline dedupe knows there is a block sharing opportunity between the block it just read and the one previously indexed. It combines that information and queues a request to an asynchronous dedupe worker thread. As such, it is possible to deduplicate a data set purely by reading it all. To help mitigate the performance impact, all the hashing is performed out-of-band in the prefetch path, rather than in the latency-sensitive read path.

Small File Storage Efficiency (SFSE) and SmartDedupe

SFSE is mutually exclusive to all the other shadow store consumers (file clones, inline dedupe, SmartDedupe). Files can either be packed with SFSE or cloned/deduped, but not both. Inlined files (small files with their data stored in the inode) will not be deduplicated and non-inlined data files that are once deduped will not inline afterwards.

InsightIQ and SmartDedupe

InsightIQ, the Dell PowerScale multi-cluster reporting and trending analytics suite, is integrated with SmartDedupe. Included in the data provided by the File Systems Analytics module is a report detailing the space savings efficiency delivered by deduplication.

SmartDedupe use cases

As mentioned above, an enterprise's data typically contains substantial quantities of redundant information. And home directories, file shares and data archives are great example of workloads that consistently yield solid deduplication results. Each time a spreadsheet, document or email attachment is saved by multiple employees, the same file is stored in full multiple times, taking up valuable disk capacity. SmartDedupe is typically used in the following ways:

Example A: File shares and home directory deduplication

By architecting and configuring home directory and file share repositories under unifying top-level directories (for example, /ifs/home and /ifs/data, respectively), an organization can easily and efficiently configure and run deduplication against these data sets.

Performance-wise, home directories and file shares are typically mid-tier workloads, usually involving concurrent access with a reasonable balance of read and write and data and metadata operations. As such, they make great candidates for SmartDedupe.

SmartDedupe should ideally be run during periods of low cluster load and client activity (nights and weekends, for example). Once the initial job has completed, the deduplication job can be scheduled to run every two weeks or so, depending on the data's rate of change.

Example B: Storage efficient archiving

SmartDedupe is an ideal solution for large, infrequently accessed content repositories. Examples of these include digital asset management workloads, seismic data archives for energy exploration, document management repositories for legal discovery, compliance archives for financial or medical records, and so on.

These are all excellent use cases for deduplication, since the performance requirements are typically low and biased towards metadata operations, and there are typically numerous duplications of data. As such, trading system resources for data efficiency produces significant, tangible benefits to the bottom line. SmartDedupe is also ideal for SmartLock-protected immutable archives and other WORM data sets, typically delivering attractive levels of storage efficiency.

For optimal results, where possible ensure that archive data is configured with the same level of protection. For data archives which are frequently scanned or indexed, metadata read acceleration on SSDs.

Example C: Disaster recovery target cluster deduplication

For performance-oriented environments that would prefer not to run deduplication against their primary dataset, the typical approach is to deduplicate the read-only data replica on their target, or disaster recovery (DR), cluster.

Once the initial dedupe job has successfully completed, subsequent incremental dedupe jobs can be scheduled to run soon after completion of each SyncIQ replication job, or as best fits the rate of data change and frequency of cluster replication.

SmartDedupe and OneFS storage utilization

SmartDedupe is one of several components of OneFS that enables a cluster to deliver a very high level of raw disk utilization. Another major storage efficiency attribute is the way that OneFS natively manages data protection in the file system. Unlike most file systems that rely on hardware RAID, OneFS protects data at the file level and, using software-based erasure coding, allows most customers to enjoy raw disk space utilization levels in the 80% range or higher. This is in contrast to the industry mean of around 50-60% raw disk capacity utilization. SmartDedupe serves to further extend this storage efficiency headroom, bringing an even more compelling and demonstrable TCO advantage to primary file-based storage.

SmartDedupe best practices

For optimal cluster performance, Dell Technologies recommends observing the following SmartDedupe best practices. Note that some of this information may be covered elsewhere in this paper.

- Deduplication is most effective when applied to data sets with a low rate of change – for example, archived data.
- Enable SmartDedupe to run at subdirectory level(s) below /ifs.
- Avoid adding more than ten subdirectory paths to the SmartDedupe configuration policy,
- SmartDedupe is ideal for home directories, departmental file shares and warm and cold archive data sets.
- Run SmartDedupe against a smaller sample data set first to evaluate performance impact versus space efficiency.
- Schedule deduplication to run during the cluster's low usage hours – such as overnight and on weekends.
- After the initial dedupe job has completed, schedule incremental dedupe jobs to run every two weeks or so, depending on the size and rate of change of the dataset.
- Always run SmartDedupe with the default 'low' impact Job Engine policy.
- Run the dedupe assessment job on a single root directory at a time. If multiple directory paths are assessed in the same job, you will not be able to determine which directory should be deduplicated.
- When replicating deduplicated data, to avoid running out of space on target, it is important to verify that the logical data size (that is, the amount of storage space

saved plus the actual storage space consumed) does not exceed the total available space on the target cluster.

- Run a deduplication job on an appropriate data set prior to enabling a snapshots schedule.
- Where possible, perform any snapshot restores (reverts) before running a deduplication job. And run a dedupe job directly after restoring a prior snapshot version.

SmartDedupe considerations

As discussed earlier, deduplication is not free. There is always trade-off between cluster resource consumption (CPU, memory, disk), the potential for data fragmentation and the benefit of increased space efficiency.

- Since deduplication trades cluster performance for storage capacity savings, SmartDedupe is not ideally suited for heavily trafficked data, or high-performance workloads.
- Depending on an application’s I/O profile and the effect of deduplication on the data layout, read and write performance and overall space savings can vary considerably.
- SmartDedupe will not permit block sharing across different hardware types or node pools to reduce the risk of performance asymmetry.
- SmartDedupe will not share blocks across files with different protection policies applied.
- OneFS metadata, including the deduplication index, is not deduplicated.
- Deduplication is a long running process that involves multiple job phases that are run iteratively.
- SmartDedupe will not attempt to deduplicate files smaller than 32KB in size.
- Dedupe job performance will typically improve significantly on the second and subsequent job runs, once the initial index and the bulk of the shadow stores have already been created.
- SmartDedupe will not deduplicate the data stored in a snapshot. However, snapshots can certainly be created of deduplicated data.
- If dedupe is enabled on a cluster that already has a significant amount of data stored in snapshots, it will take time before the snapshot data is affected by deduplication. Newly created snapshots will contain deduplicated data, but older snapshots will not.
- SmartDedupe deduplicates common blocks within the same file, resulting in even better data efficiency.
- In general, additional capacity savings may not warrant the overhead of running SmartDedupe on node pools with inline deduplication enabled.
- Deduplication of data contained within a writable snapshot is not supported in OneFS 9.3 and later.

Small file storage efficiency

Introduction

Many workloads across the spectrum of commerce contain significant quantities of small files. Workloads that could potentially benefit from OneFS small file storage efficiency include:

- SW development
- EDA
- Life sciences workloads (sequencers)
- Healthcare (PACS)
- Database operations
- Analytics
- Machine Learning/ADAS
- Financial services ticker data

Small file packing

Archive applications such as next generation healthcare Picture Archiving and Communication Systems (PACS) are moving away from housing large archive file formats (such as tar and zip files) to storing the smaller files individually. To directly address this trend, the OneFS operating system includes Storage Efficiency feature. 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, such as a typical healthcare DICOM dataset.

Efficiency is achieved by scanning the on-disk data for small files and packing them into larger OneFS data structures, known as shadow stores. These shadow stores are then parity protected using erasure coding, and typically provide storage efficiency of 80% or greater.

OneFS Small File Storage Efficiency is specifically designed for infrequently modified, archive datasets. As such, it trades a small read latency performance penalty for improved storage utilization. Files obviously remain writable, since archive applications are assumed to periodically need to update at least some of the small file data.

Architecture

OneFS Small File Storage Efficiency is predicated on the notion of containerization of files, and consists of six main components:

- File pool configuration policy
- SmartPools Job
- Shadow Store
- Configuration control path
- File packing and data layout infrastructure
- Defragmenter

The way data is laid out across the nodes and their respective disks is fundamental to a cluster's functionality. OneFS is a single file system providing one vast, scalable namespace—free from multiple volume concatenations or single points of failure. As such, a OneFS powered cluster can support data sets with hundreds of billions of small files all within the same file system.

OneFS lays data out across multiple nodes allowing files to benefit from the resources (spindles and cache) of up to twenty nodes. Reed-Solomon erasure coding is used to protecting at the file-level, enabling the cluster to recover data quickly and efficiently, and providing exceptional levels storage utilization. OneFS provides protection against up to four simultaneous component failures respectively. A single failure can be as little as an individual disk or an entire node.

Various mirroring options are also available, and OneFS typically uses these to protect metadata and small files. Striped, distributed metadata coupled with continuous auto-balancing affords OneFS near linear performance characteristics, regardless of the capacity utilization of the system. Both metadata and file data are spread across the entire cluster keeping the cluster balanced at all times.

The OneFS file system employs a native block size of 8KB, and sixteen of these blocks are combined to create a 128KB stripe unit. Files larger than 128K are protected with error-correcting code parity blocks (FEC) and striped across nodes. This allows files to use the combined resources of up to twenty nodes, based on per-file policies.

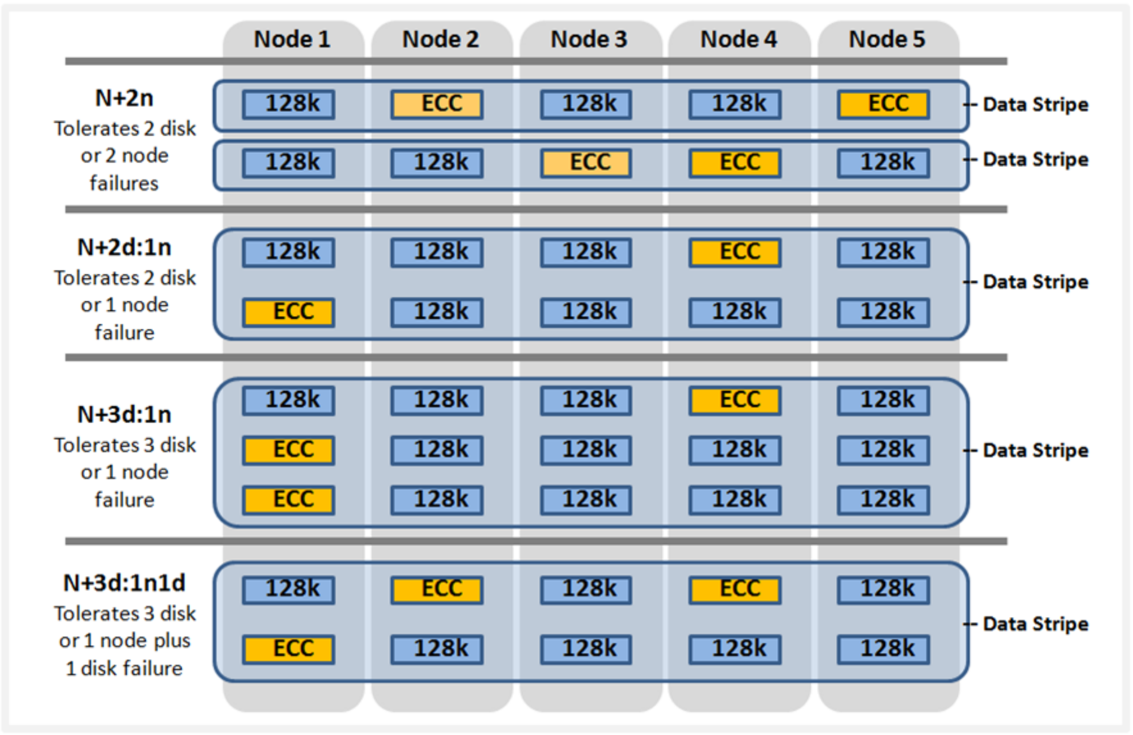

Node 1
Node 2
Node 3
Node 4
Node 5
N+2n
Tolerates 2 disk or 2 node failures
N+2d:1n
Tolerates 2 disk or 1 node failure
N+3d:1n
Tolerates 3 disk or 1 node failure
N+3d:1n1d
Tolerates 3 disk or 1 node plus 1 disk failure
128k
ECC
Data Stripe

Figure 48. OneFS on-disk FEC protection

Files smaller than 128KB are unable to fill a stripe unit, so are mirrored rather than FEC protected, resulting in a less efficient on-disk footprint. For most data sets, this is rarely an issue, since the presence of a smaller number of larger FEC protected files offsets the mirroring of the small files.

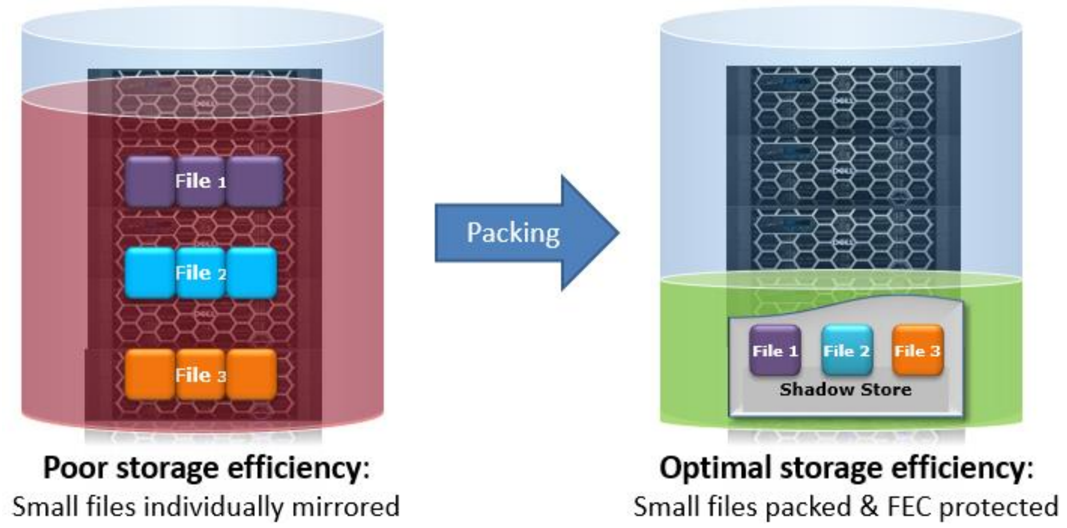

File 1
File 2
File 3
Packing
File 1
File 2
File 3
Shadow Store
Poor storage efficiency:
Small files individually mirrored
Optimal storage efficiency:
Small files packed & FEC protected

Figure 49. OneFS small file containerization

For example, if a file is 24KB in size, it will occupy three 8KB blocks. If it has two mirrors for protection, there will be a total of nine 8KB blocks, or 72KB, that will be needed to protect and store it on disk. Clearly, being able to pack several of these small files into a larger, striped, and parity protected container will provide a great space benefit.

Also, files in the 150KB to 300KB range typically see utilization of around 50%, as compared to 80% or better when containerized with the OneFS Small File Storage Efficiency feature.

Under the hood, the small file packing has similarities to the OneFS file cloning process, and both operations use the same underlying infrastructure – the shadow store.

Shadow stores are similar to regular files, but do not contain all the metadata typically associated with regular file inodes. In particular, time-based attributes (creation time, modification time, and so on) are explicitly not maintained. The shadow stores for storage efficiency differ from existing shadow stores in a few ways in order to isolate fragmentation, to support tiering, and to support future optimizations which will be specific to single-reference stores.

Containerization is managed by the SmartPools job. This job typically runs by default on a cluster with a 10pm nightly schedule and a low impact management setting but can also be run manually on-demand. Also, the SetProtectPlus, SmartPoolsTree job, isi filepool apply, and the isi set command are also able to perform file packing.

File attributes indicate each file's pack state:

- **packing_policy**: container or native. This indicates whether the file meets the criteria set by your file pool policies and is eligible for packing. Container indicates that the file is eligible to be packed; native indicates that the file is not eligible to be packed. Your file pool policies determine this value. The value is updated by the SmartPools job.
- **packing_target**: container or native. This is how the system evaluates a file's eligibility for packing based on additional criteria such as file size, type, and age.

Container indicates that the file should reside in a container shadow store. Native indicates that the file should not be containerized.

- **packing_complete**: complete or incomplete. This field establishes whether or not the target is satisfied. Complete indicates that the target is satisfied, and the file is packed. Incomplete indicates that the target is not satisfied, and the packing operation is not finished. It is worth noting that several healthcare archive applications can natively perform file containerization. In these cases, the benefits of OneFS small file efficiency will be negated.

Configuration and use

Before configuring small file storage efficiency on a cluster, please ensure the following pre-requisites are met:

- **Only enable on an archive workflow**: This is strictly an archive solution. An active dataset, particularly one involving overwrites and deletes of containerized files, can generate fragmentation which impacts performance and storage efficiency.
- **Most of the archived data consists of small files**. By default, the threshold target file size is from 0-1 MB.

Also, it is highly recommended to have DataIQ or InsightIQ software licensed on the cluster. This enables the file systems analysis (FAS) job to be run, which provides enhanced storage efficiency reporting statistics. This is covered later in this paper.

The first step in configuring small file storage efficiency on a cluster is to enable the packing process. To do so, run the following command from the OneFS CLI:

```
# isi_packing --enabled=true
```

Once the isi_packing variable is set, and the licensing agreement is confirmed, configuration is done using a filepool policy. The following CLI example will containerize data under the cluster directory /ifs/data/dicom.

```
# isi filepool policies create dicom --enable-packing=true --
begin-filter --path=/ifs/data/pacs --end-filter
```

The SmartPools configuration for the resulting ‘dicom’ filepool can be verified with the following command:

```
# isi filepool policies view dicom
                          Name: dicom
                   Description: -
                         State: OK
                 State Details:
                   Apply Order: 1
         File Matching Pattern: Birth Time > 1D AND Path == dicom (begins with)
      Set Requested Protection: -
           Data Access Pattern: -
              Enable Coalescer: -
                Enable Packing: Yes
...
```

Note: There is no dedicated WebUI for OneFS small file storage efficiency, so configuration is performed by the CLI.

The isi_packing command will also confirm that packing has been enabled.

```
# isi_packing --ls
Enabled:                            Yes
Enable ADS:                         No
Enable snapshots:                   No
Enable mirror containers:           No
Enable mirror translation:          No
Unpack recently modified:           No
Unpack snapshots:                   No
Avoid deduped files:                Yes
Maximum file size:                  1016.0k
SIN cache cutoff size:              8.00M
Minimum age before packing:        0s
Directory hint maximum entries:     16
Container minimum size:             1016.0k
Container maximum size:             1.000G
```

Note: While the defaults will work for most use cases, the two values you may want to adjust are maximum file size (--max-size <bytes>) and minimum age for packing (--min-age <seconds>).

Files are then containerized in the background using the SmartPools job, which can be run on-demand, or using the nightly schedule.

```
# isi job jobs start SmartPools
Started job [1016]
```

After enabling a new filepool policy, the SmartPools job may take a relatively long time due to packing work. However, subsequent job runs should be significantly faster.

Small file storage efficiency reporting can be viewed using the SmartPools job reports, which detail the number of files packed. For example:

```
#  isi job reports view -v 1016
```

For clusters with a valid DataIQ or InsightIQ license, if the FSA (file system analytics) job has run, a limited efficiency report will be available. This can be viewed by the following command:

```
# isi_packing --fsa
```

Note: For clusters using CloudPools, be aware that you cannot containerize stubbed files. SyncIQ data will be unpacked, so packing will need to be configured on the target cluster.

File unpacking

To unpack previously packed, or containerized, files, in this case from the 'dicom' filepool policy, run the following command from the OneFS CLI:

```
# isi filepool policies modify dicom --enable-packing=false
```

Note: Ensure there is sufficient free space on the cluster before unpacking. Also, if the data is in a snapshot, it will not be packed – only HEAD file data will be containerized

A threshold is provided, which prevents very recently modified files from being containerized. The default value for this is 24 hours, but this can be reconfigured by the isi_packing –min-age <seconds> command, if wanted. This threshold guards against accidental misconfiguration within a filepool policy, which could potentially lead to containerization of files which are actively being modified, which could result in container fragmentation.

Monitoring and reporting

There are four main CLI commands that report on the status and effect of small file efficiency:

- `isi job reports view <job_id>`
- `isi_packing –fsa`
- `isi_cstat`
- `isi_sfse_assess`

isi job report view

When running the `isi job report view` command, enter the job ID as an argument. In the command output, the 'file packed' field will indicate how many files have been successfully containerized. For example, for job ID 1018:

```
# isi job reports view -v 1018
SmartPools[1018] phase 1 (2021-02-31T10:29:47
---------------------------------------------
Elapsed time                       12 seconds
Working time                       12 seconds
Group at phase end                 <1,6>: { 1:0-5, smb: 1, nfs:
1, hdfs: 1, swift: 1, all_enabled_protocols: 1}
Errors
'dicom':
      {'Policy Number': 0,
      'Files matched': {'head':512, 'snapshot': 256}
      'Directories matched': {'head': 20, 'snapshot': 10},
      'ADS containers matched': {'head':0, 'snapshot': 0},
      'ADS streams matched': {'head':0, 'snapshot': 0},
      'Access changes skipped': 0,
'Protection changes skipped': 0,
'Packing changes skipped': 0,
'File creation templates matched': 0,
'Skipped packing non-regular files': 2,
'Files packed': 48672,
'Files repacked': 0,
'Files unpacked': 0,
},
}
```

isi_packing –fsa

The second command, ‘isi_packing –fsa’, provides a storage efficiency percentage in the last line of its output. This command requires a successful run of the file system analysis (FSA) job. If FSA has not been run previously, it can be kicked off with the following isi job jobs start FSAnalyze command. For example:

```
# isi job jobs start FSAnalyze
Started job [1018]
```

When this job has completed, run:

```
# isi_packing --fsa --fsa-jobid 1018
FSAnalyze job: 1018 (Mon Mar 1 22:01:21 2021)
Logical size:  47.371T
Physical size: 58.127T
Efficiency:    81.50%
```

In this case, the storage efficiency achieved after containerizing the data is 81.50%, as reported by isi_packing.

If you do not specify an FSAnalyze job ID, the –fsa defaults to the last successful FSAnalyze job run results.

Note: The `isi_packing --fsa` command reports on the whole /ifs filesystem. This means that the overall utilization percentage can be misleading if other, non-containerized data is also present on the cluster.

isi_cstats

The ‘isi_cstats’ CLI command also reports on packed files and includes both the total capacity of containerized data and a container efficiency ratio:

```
# isi_cstats
Total files                  : 397234451
Total inlined files          : 379948336
Total directories            : 32380092
Total logical data           : 18471 GB
Total shadowed data          : 624 GB
Total physical data          : 26890 GB
Total reduced data           : 14645 GB
Total protection data        : 2181 GB
Total inode data             : 9748 GB

Current logical data         : 18471 GB
Current shadowed data        : 624 GB
Current physical data        : 26878 GB
Snapshot logical data        : 0 B
Snapshot shadowed data       : 0 B
Snapshot physical data       : 32768 B

Total inlined data savings   : 2899 GB
Total inlined data ratio     : 1.1979 : 1
Total compression savings    : 303 GB
Total compression ratio     : 1.0173 : 1
Total deduplication savings : 624 GB
Total deduplication ratio   : 1.0350 : 1
Total containerized data    : 0 B
Total container efficiency  : 1.0000 : 1
Total data reduction ratio  : 1.0529 : 1
Total storage efficiency    : 0.6869 : 1

Raw counts
{ type=bsin files=3889 lsize=314023936 pblk=1596633 refs=81840315
data=18449 prot=25474 ibyte=23381504 fsize=8351563907072 iblocks=0
}
{ type=csin files=0 lsize=0 pblk=0 refs=0 data=0 prot=0 ibyte=0
fsize=0 iblocks=0 }
{ type=hdir files=32380091 lsize=0 pblk=35537884 refs=0 data=0
prot=0 ibyte=1020737587200 fsize=0 iblocks=0 }
{ type=hfile files=397230562 lsize=19832702476288 pblk=2209730024
refs=81801976 data=1919481750 prot=285828971 ibyte=9446188553728
fsize=17202141701528 iblocks=379948336 }
{ type=sdir files=1 lsize=0 pblk=0 refs=0 data=0 prot=0
ibyte=32768 fsize=0 iblocks=0 }
{ type=sfile files=0 lsize=0 pblk=0 refs=0 data=0 prot=0 ibyte=0
fsize=0 iblocks=0 }
```

Note: Capacity savings from SFSE are included in the overall storage efficiency ratio, where SFSE primarily saves space by reducing protection overhead (FEC blocks). By combining multiple small files into compression chunks in the containers, SFSE helps to improve the compression ratio which in turn improves the data reduction ratio. However, OneFS does not currently separate out the SFSE improvements from the global storage efficiency statistics.

isi_sfse_assess

There is also a Storage Efficiency assessment tool available in OneFS 8.2 and later. This can be run as from the CLI with the following syntax:

```
# isi_sfse_assess <options>
```

Estimated storage efficiency is presented in the tool's output in terms of raw space savings as a total and percentage and a percentage reduction in protection group overhead.

```
SFSE estimation summary:
* Raw space saving: 1.7 GB (25.86%)
* PG reduction: 25978 (78.73%)
```

Defragmentation

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 reduce the storage efficiency.

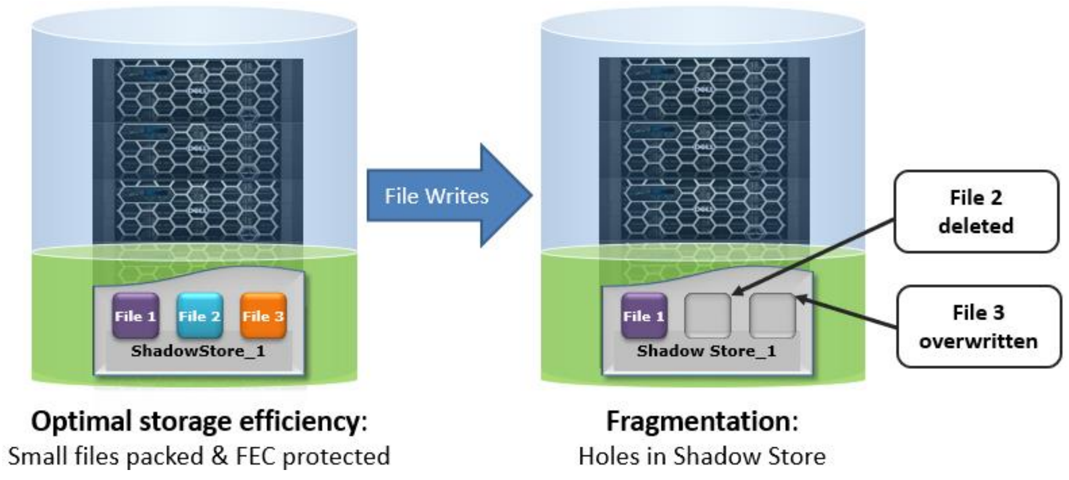


Figure 50. OneFS containerization with fragmentation

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.

In OneFS 8.2 and later, a shadow store defragmenter is added to reduce fragmentation resulting of overwrites and deletes of files. This defragmenter is integrated into the ShadowStoreDelete job. The defragmentation process works by dividing each containerized file into logical chunks (~32MB each) and assessing each chunk for fragmentation.

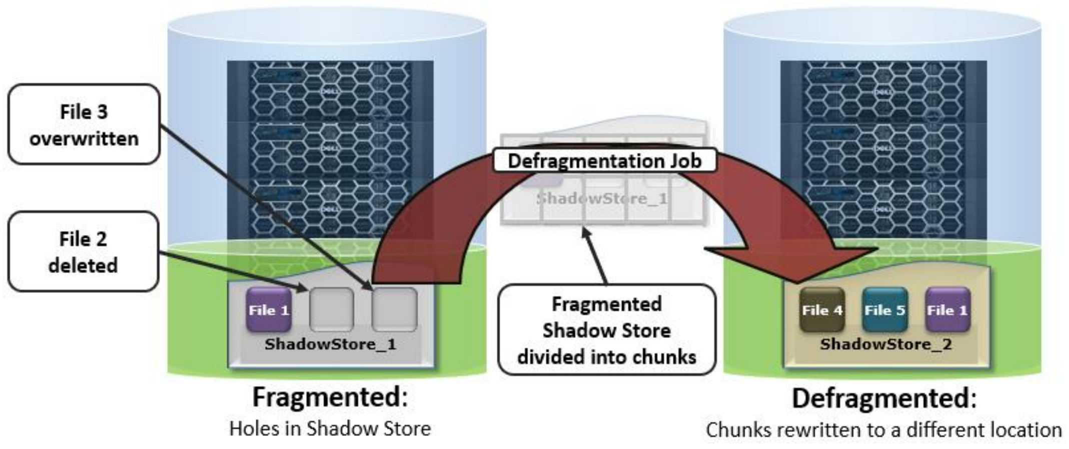
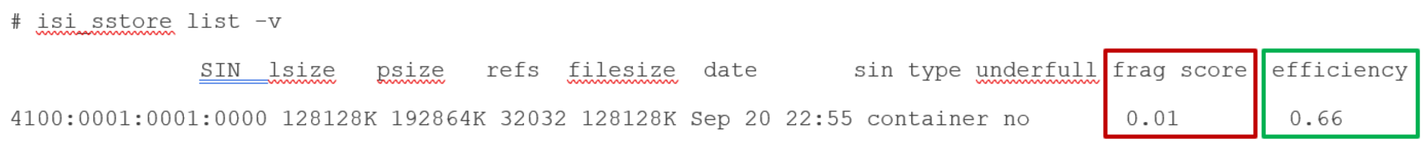


Figure 51. OneFS defragmentation

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.

The ‘isi_sstore list’ CLI command can be run to report fragmentation and efficiency scores. For example:



```
# isi sstore list -v
                SIN   lsize    psize    refs  filesize  date          sin type underfull frag score  efficiency
4100:0001:0001:0000 128128K 192864K 32032 128128K Sep 20 22:55 container no           0.01          0.66
```

Figure 52. Example output from the ‘isi_sstore list’ CLI command, showing an efficiency value and a fragmentation score

The fragmentation score is the ratio of holes in the data where FEC is still required, whereas the efficiency value is a ratio of logical data blocks to total physical blocks used by the shadow store. Fully sparse stripes do not need FEC so are not included. The guideline is that lower fragmentation scores and higher efficiency scores are better.

The defragmenter does not require a license to run and is disabled by default in OneFS 8.2 and later. It can be easily activated using the following CLI commands:

```
# isi_gconfig -t defrag-config defrag_enabled=true
```

Once enabled, the defragmenter can be started by the job engine’s ShadowStoreDelete job, either from the OneFS WebUI or by the following CLI command:

```
# isi job jobs start ShadowStoreDelete
```

The defragmenter can also be run in an assessment mode. This reports on and helps to determine the amount of disk space that will be reclaimed, without moving any actual data. The ShadowStoreDelete job can run the defragmenter in assessment mode but the statistics generated are not reported by the job. The isi_sstore CLI command has a ‘defrag’ option and can be run with the following syntax to generate a defragmentation assessment:

```
# isi_sstore defrag -d -a -c -p -v
…
Processed 1 of 1 (100.00%) shadow stores, space reclaimed 31M
Summary:
    Shadows stores total: 1
    Shadows stores processed: 1

    Shadows stores skipped: 0
    Shadows stores with error: 0
    Chunks needing defrag: 4
    Estimated space savings: 31M
```

Best practices

Recommended best practices for Small File Storage Efficiency include:

- Only enable storage efficiency on an archive workflow with a high percentage of small files.
- Most logical space used on cluster is for small files. In this case, small files are considered as less than 512 KB in size.
- The default minimum age for packing is anything over one day, and this will override anything configured in the filepool policy.

- Where possible, limit changes (overwrites and deletes) to containerized files, which cause fragmentation and impact both file read performance and storage efficiency
- Ensure there is sufficient free space available on the cluster before unpacking any containerized data.
- Ensure the archive solution being used does not natively perform file containerization, or the benefits of OneFS small file storage efficiency will likely be negated.
- Use a path-based filepool policy for configuration, where possible, rather than more complex filepool filtering logic.
- Avoid configuring the maximum file size value inside the file pool filter itself. Instead set this parameter by the isi_packing command.
- Use SFSE to archive static small file workloads, or those with only moderate overwrites and deletes.
- If necessary, run the defragmentation job on a defined schedule (that is, weekly) to eliminate fragmentation.

Considerations

Small File Storage Efficiency for Archive is not free. There is always trade-off between cluster resource consumption (CPU, memory, disk), the potential for data fragmentation and the benefit of improved space utilization. Bear the following in mind:

- This is a storage efficiency product, not a performance product.
- The time to retrieve a packed archive image should not be much greater than an unpacked image data – unless fragmentation has occurred.
- Configuration is only by the OneFS CLI, rather than the WebUI, at this point.
- After enabling a filepool policy, the first SmartPools job may take a relatively long time due to packing work, but subsequent runs should be much faster.
- For clusters using CloudPools you cannot containerize stubbed files.
- SyncIQ data will be unpacked during replication, so packing will need to be configured on the target cluster.
- If the data is in a snapshot, it will not be packed – only HEAD file data will be containerized.
- The isi_packing --fsa command reports on the whole file system, so the overall utilization percentage can be misleading if other, non-containerized data is also present on the cluster.
- Alternate data streams (ADS, that is the streams themselves, not the parent files) will not be containerized by default.
- Packing and unpacking will be logically preserving actions, they will not cause logical changes to a file and therefore will not trigger snapshot COW.
- If you have already run SmartDedupe data deduplication software on your data, you will not see much additional benefit because your data is already in shadow stores.
- If you run SmartDedupe against packed data, the deduped files will be skipped.

- SFSE is compatible with SmartLock WORM and files protected with SmartLock file retention policies will be packed as expected.
- You can clone files with packed data.
- Containerization is managed by the SmartPools job. However, the SmartPoolsTree jobs, isi filepool apply, and isi set will also be able to perform file packing.
- Small file packing (SFSE) will not be applied to inlined data files (where a small file's data is stored in its inode).
- Small file packing is not supported for data contained within writable snapshots.

Inode data inlining

Introduction

OneFS 9.3 and later releases contain a file system storage efficiency feature which stores a small file's data within the inode, rather than allocating additional storage space. The principal benefits of data inlining include:

- Reduced storage capacity utilization for small file datasets, generating an improved cost per TB ratio.
- Dramatically improved SSD wear life.
- Potential read and write performance for small files.
- Zero configuration, adaptive operation, and full transparency at the OneFS file system level.
- Broad compatibility with other OneFS data services, including compression and deduplication.

Architecture

Data inlining explicitly avoids allocation during write operations since small files do not require any data or protection blocks for their storage. Instead, the file content is stored directly in unused space within the file's inode. This approach is also highly flash media friendly since it significantly reduces the quantity of writes to SSD drives.

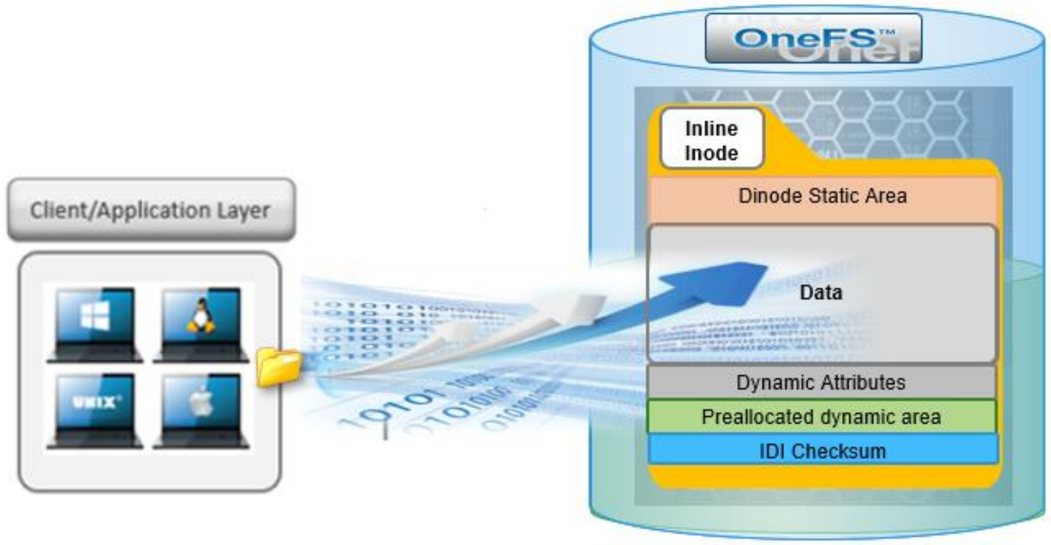


Figure 53. OneFS data inlining within an 8KB inode

OneFS inodes, or index nodes, are a special class of data structure that store file attributes and pointers to file data locations on disk. They serve a similar purpose to traditional UNIX file system inodes, but also have some additional, unique properties. Each file system object, whether it be a file, directory, symbolic link, alternate data stream container, shadow store, and so on, is represented by an inode.

Within OneFS, SSD node pools in F series all-flash nodes always use 8KB inodes. For hybrid and archive platforms, the HDD node pools are either 512 bytes or 8KB in size, and this is determined by the physical and logical block size of the hard drives or SSDs in a node pool. There are three different styles of drive formatting used in OneFS nodes, depending on the manufacturer's specifications:

Drive formatting	Characteristics
Native 4Kn (native)	A native 4Kn drive has both a physical and logical block size of 4096B.
512n (native)	A drive that has both physical and logical size of 512 is a native 512B drive.
512e (emulated)	A 512e (512 byte-emulated) drive has a physical block size of 4096, but a logical block size of 512B.

If the drives in a cluster's nodepool are native 4Kn formatted, by default the inodes on this nodepool will be 8KB in size. Alternatively, if the drives are 512e formatted, then inodes by default will be 512B in size. However, they can also be reconfigured to 8KB in size if the 'force-8k-inodes' setting is set to true.

A OneFS inode is composed of several sections. These include a static area, which is typically 134 bytes in size and contains fixed-width, commonly used attributes like POSIX mode bits, owner, and file size. Next, the regular inode contains a metatree cache, which is used to translate a file operation directly into the appropriate protection group. However, for inline inodes, the metatree is no longer required, so data is stored directly in this area instead. Following this is a preallocated dynamic inode area where the primary attributes, such as OneFS ACLs, protection policies, embedded B+ Tree roots, timestamps, and so on, are cached. And lastly a sector where the IDI checksum code is stored.

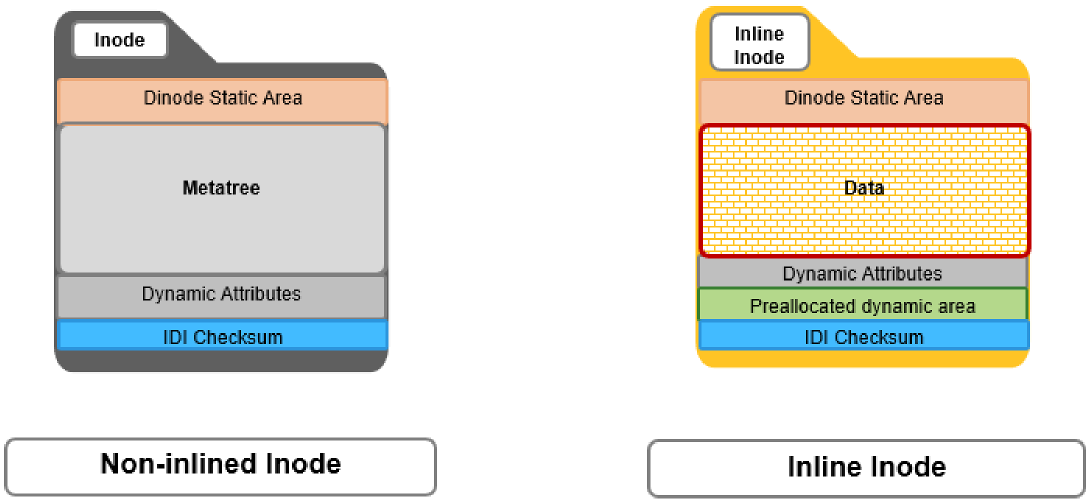


Figure 54. OneFS inode on-disk structure

Inline reads and writes

When a file write coming from the writeback cache, or coalescer, is determined to be a candidate for data inlining, it goes through a fast write path in BSW. Compression will be applied, if appropriate, before the inline data is written to storage.

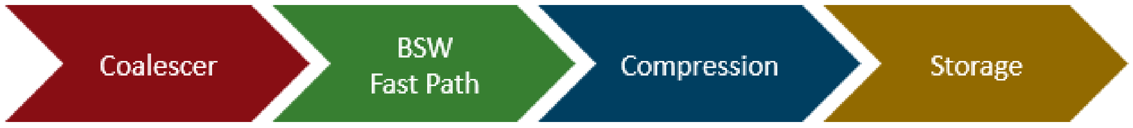


Figure 55. Inline inode write path

The read path for inlined files is similar to that for regular files. However, if the file data is not already available in the caching layers, it is read directly from the inode, rather than from separate disk blocks as with regular files.

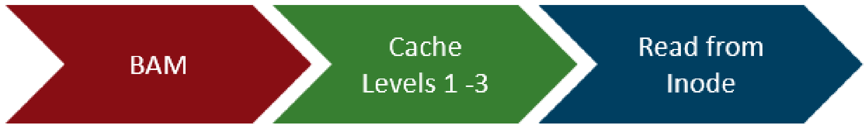


Figure 56. Inline inode read path

Inode protection

Protection for inlined data operates the same way as for other inodes and involves mirroring. OneFS uses mirroring as protection for all metadata because it is simple and does not require the additional processing overhead of erasure coding. The number of inode mirrors is determined by the nodepool's achieved protection policy, as in the following table:

OneFS protection level	Number of inode mirrors
+1n	2 inodes per file
+2d:1n	3 inodes per file
+2n	3 inodes per file
+3d:1n	4 inodes per file
+3d:1n1d	4 inodes per file
+3n	4 inodes per file
+4d:1n	5 inodes per file
+4d:2n	5 inodes per file
+4n	5 inodes per file

Note: Unlike file inodes above, directory inodes, which comprise the OneFS single namespace, are mirrored at one level higher than the achieved protection policy. The root of the LIN Tree is the most critical metadata type and is always mirrored at 8x.

Operation and management

Data inlining is automatically enabled by default on all 8KB formatted nodepools for clusters running OneFS 9.3 and later, and does not require any additional software, hardware, or product licenses in order to operate. Its operation is fully transparent and, as such, there are no OneFS CLI or WebUI controls to configure or manage inlining. However, the following OneFS sysctl can be used to disable data inlining in an emergency situation. Use of this sysctl is only recommended under the direction and supervision of Dell technical support:

```
# isi_for_array sysctl efs.bam.data_inline.enable=0
```

Similarly, restoring this sysctl value to “1” (the default) will re-enable data inlining.

In order to upgrade to OneFS 9.3 and later and benefit from data inlining, the cluster must be running a minimum OneFS 8.2.1 or later. A full upgrade commit to OneFS 9.3 and later is required before inlining becomes operational.

Note: Existing files will not be inlined during the upgrade process as of OneFS 9.3 and later. This limitation will be addressed in a future release.

Monitoring and reporting

The OneFS ‘isi_drivenum’ CLI command can be used to verify the drive block sizes in a node. For example, below is the output for a PowerScale Gen6 H-series node, showing drive bay 1 containing an SSD with 4KB physical formatting and 512byte logical sizes, and bays A to E consisting of hard disks (HDDs) with both 4KB logical and physical formatting.

```
# isi_drivenum -bz
Bay 1        Physical Block Size: 4096     Logical Block Size:     512
Bay 2        Physical Block Size: N/A      Logical Block Size:     N/A
Bay A0       Physical Block Size: 4096     Logical Block Size:     4096
Bay A1       Physical Block Size: 4096     Logical Block Size:     4096
Bay A2       Physical Block Size: 4096     Logical Block Size:     4096
Bay B0       Physical Block Size: 4096     Logical Block Size:     4096
Bay B1       Physical Block Size: 4096     Logical Block Size:     4096
Bay B2       Physical Block Size: 4096     Logical Block Size:     4096
Bay C0       Physical Block Size: 4096     Logical Block Size:     4096
Bay C1       Physical Block Size: 4096     Logical Block Size:     4096
Bay C2       Physical Block Size: 4096     Logical Block Size:     4096
Bay D0       Physical Block Size: 4096     Logical Block Size:     4096
Bay D1       Physical Block Size: 4096     Logical Block Size:     4096
Bay D2       Physical Block Size: 4096     Logical Block Size:     4096
Bay E0       Physical Block Size: 4096     Logical Block Size:     4096
Bay E1       Physical Block Size: 4096     Logical Block Size:     4096
Bay E2       Physical Block Size: 4096     Logical Block Size:     4096
```

Note: The SSD disk pools used in PowerScale hybrid nodes that are configured for meta-read or meta-write SSD strategies use 512 byte inodes by default. This can significantly save space on these pools, as they often have limited capacity. However, it will prevent data inlining from occurring. By contrast, PowerScale all-flash nodepools are configured by default for 8KB inodes.

The OneFS ‘isi get’ CLI command provides a convenient method to verify which size inodes are in use in a given node pool. The command’s output includes both the inode mirrors size and the inline status of a file. More information about the ‘isi get’ utility is provided in Efficiency reporting.

Feature integration

Data inlining does have some notable caveats. Specifically, data inlining will not be performed in the following instances:

- When upgrading to OneFS 9.3 or later from an earlier release which does not support inlining.
- During restriping operations, such as SmartPools tiering, when data is moved from a 512 byte diskpool to an 8KB diskpool.
- Writing CloudPools SmartLink stub files.
- On file truncation down to non-zero size.
- Sparse files (for example, NDMP sparse punch files) where allocated blocks are replaced with sparse blocks at various file offsets.
- For files within a writable snapshot.

Similarly, the following operations may cause inlined data inlining to be undone, or spilled:

- Restriping from an 8KB diskpool to a 512 byte diskpool.
- Forcefully allocating blocks on a file (for example, using the POSIX ‘madvise’ system call).
- Sparce punching a file.
- Enabling CloudPools BCM on a file.

These above limitations will be addressed in a future release.

Efficiency reporting

OneFS 9.3 and later releases provide three CLI tools for validating and reporting the presence and benefits of data inlining, namely:

1. The ‘isi statistics data-reduction’ CLI command has been enhanced to report inlined data metrics, including both a capacity saved and an inlined data efficiency ratio:

```
# isi statistics data-reduction
                      Recent Writes Cluster Data Reduction
                           (5 mins)
--------------------- ------------- ----------------------
Logical data                 90.16G                 18.05T
Zero-removal saved                0                      -
Deduplication saved           5.25G                624.51G
Compression saved             2.08G                303.46G
Inlined data saved            1.35G                  2.83T
Preprotected physical        82.83G                 14.32T
Protection overhead          13.92G                  2.13T
Protected physical           96.74G                 26.28T

Zero removal ratio         1.00 : 1                      -
```

```
Deduplication ratio        1.06 : 1               1.03 : 1
Compression ratio          1.03 : 1               1.02 : 1
Data reduction ratio       1.09 : 1               1.05 : 1
Inlined data ratio         1.02 : 1               1.20 : 1
Efficiency ratio           0.93 : 1               0.69 : 1
--------------------- ------------- ----------------------
```

Note: The effect of data inlining is not included in the data reduction ratio because it is not actually reducing the data in any way - just relocating it and protecting it more efficiently. However, data inlining is included in the overall storage efficiency ratio.

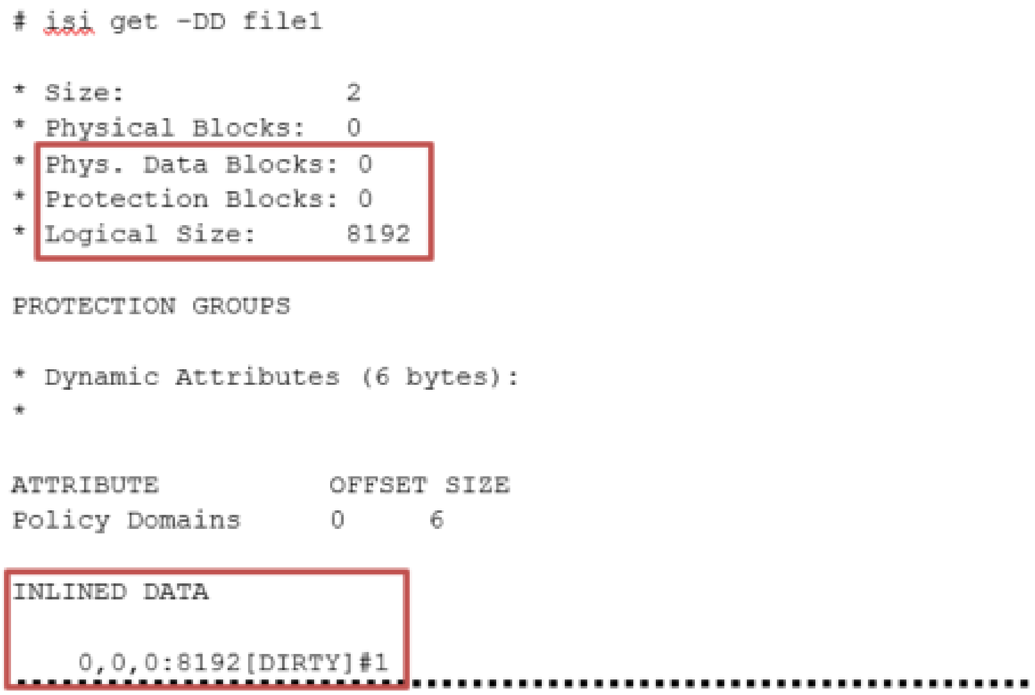
The ‘inline data saved’ value represents the count of files which have been inlined multiplied by 8KB (inode size). This value is required to make the compression ratio and data reduction ratio correct.

2. The ‘isi_cstats’ CLI command now includes the accounted number of inlined files within /ifs, which is displayed by default in its console output.

```
# isi_cstats
Total files                 : 397234451
Total inlined files         : 379948336
Total directories           : 32380092
Total logical data          : 18471 GB
Total shadowed data         : 624 GB
Total physical data         : 26890 GB
Total reduced data          : 14645 GB
Total protection data       : 2181 GB
Total inode data            : 9748 GB

Current logical data        : 18471 GB
Current shadowed data       : 624 GB
Current physical data       : 26878 GB
Snapshot logical data       : 0 B
Snapshot shadowed data      : 0 B
Snapshot physical data      : 32768 B

Total inlined data savings  : 2899 GB
Total inlined data ratio    : 1.1979 : 1
Total compression savings   : 303 GB
Total compression ratio     : 1.0173 : 1
Total deduplication savings : 624 GB
Total deduplication ratio   : 1.0350 : 1
Total containerized data    : 0 B
Total container efficiency  : 1.0000 : 1
Total data reduction ratio  : 1.0529 : 1
Total storage efficiency    : 0.6869 : 1

Raw counts
{ type=bsin files=3889 lsize=314023936 pblk=1596633
refs=81840315 data=18449 prot=25474 ibyte=23381504
fsize=8351563907072 iblocks=0 }
```

```
{ type=csin files=0 lsize=0 pblk=0 refs=0 data=0 prot=0
ibyte=0 fsize=0 iblocks=0 }
{ type=hdir files=32380091 lsize=0 pblk=35537884 refs=0
data=0 prot=0 ibyte=1020737587200 fsize=0 iblocks=0 }
{ type=hfile files=397230562 lsize=19832702476288
pblk=2209730024 refs=81801976 data=1919481750 prot=285828971
ibyte=9446188553728 fsize=17202141701528 iblocks=379948336 }
{ type=sdir files=1 lsize=0 pblk=0 refs=0 data=0 prot=0
ibyte=32768 fsize=0 iblocks=0 }
{ type=sfile files=0 lsize=0 pblk=0 refs=0 data=0 prot=0
ibyte=0 fsize=0 iblocks=0 }
```

3. The ‘isi get’ CLI command can be used to determine whether a file has been inlined. The output reports a file’s logical ‘size’, but indicates that it consumes zero physical, data, and protection blocks. There is also an ‘inlined data’ attribute further down in the output that also confirms that the file is inlined.



```
# isi get -DD file1

* Size:               2
* Physical Blocks:    0
* Phys. Data Blocks:  0
* Protection Blocks:  0
* Logical Size:       8192

PROTECTION GROUPS

* Dynamic Attributes (6 bytes):
*

ATTRIBUTE            OFFSET SIZE
Policy Domains       0      6

INLINED DATA

    0,0,0:8192[DIRTY]#1
```

Figure 57. Example output from the ‘isi get’ CLI command, showing logical size and inlined data layout

Performance

The storage efficiency potential of inode inlining can be significant for data sets consisting of large numbers of small files, which would have required a separate inode and data blocks for housing these files prior to OneFS 9.3.

Latency-wise, the write performance for inlined file writes is typically comparable or slightly better as compared to regular files, because OneFS does not have to allocate extra blocks and protect them. This is also true for reads, too, since OneFS does not have to search for and retrieve any blocks beyond the inode itself. This also frees up space in the OneFS read caching layers, as well as on disk, in addition to requiring fewer CPU cycles.

The following diagram illustrates the levels of indirection a file access request takes to get to its data. Unlike a standard file, an inline file will skip the later stages of the path which involve the inode metatree redirection to the remote data blocks.

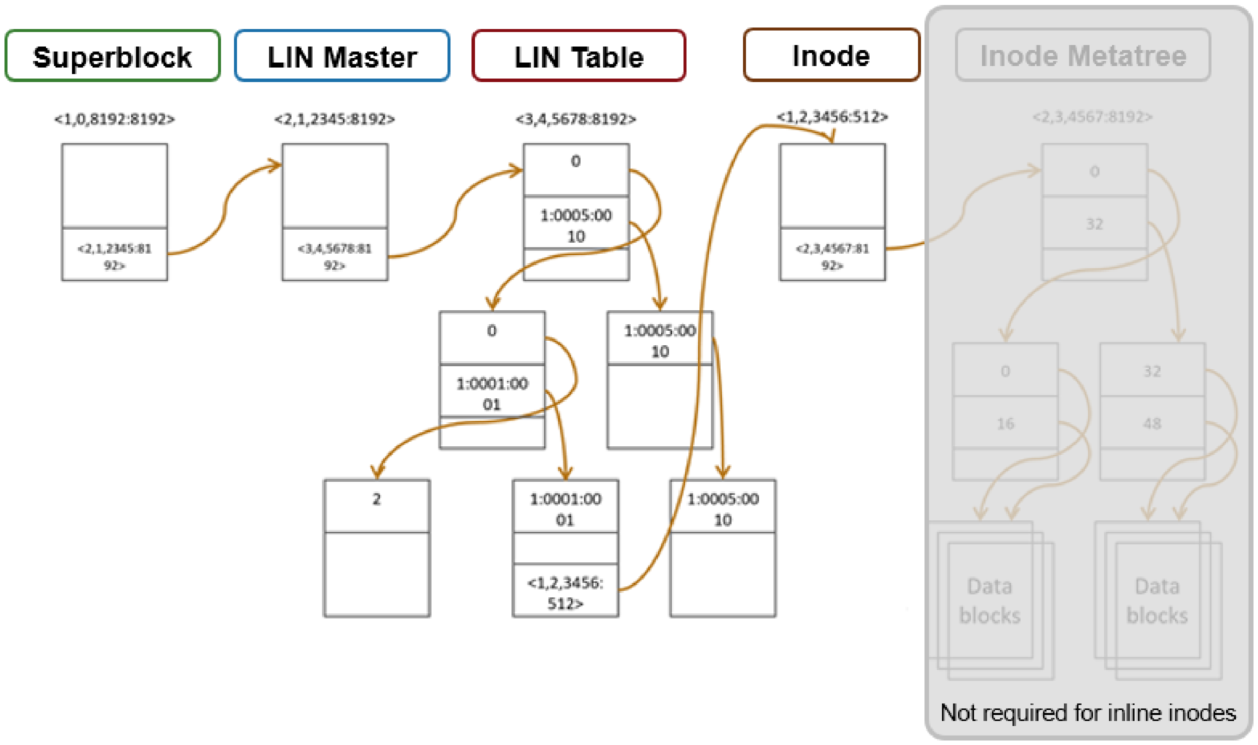


Figure 58. OneFS file access levels of indirection

Access starts with the Superblock, which is located at multiple fixed block addresses on every drive in the cluster. The Superblock contains the address locations of the LIN Master block, which contains the root of the LIN B+ Tree (LIN table). The LIN B+Tree maps logical inode numbers to the actual inode addresses on disk, which, in the case of an inlined file, also contains the data. This saves the overhead of finding the address locations of the file's data blocks and retrieving data from them.

For hybrid nodes with sufficient SSD capacity, using the metadata-write SSD strategy will automatically place inlined small files on flash media. However, since the SSDs on hybrid nodes default to 512byte formatting, when using metadata read/write strategies, these SSD metadata pools will need to have the '--force-8k-inodes' flag set in order for files to be inlined. This can be a useful performance configuration for small file HPC workloads, such as EDA, for data that is not residing on an all-flash tier.

Note: Keep in mind that forcing 8KB inodes on a hybrid pool's SSDs will result in a considerable reduction in available inode capacity than would be available with the default 512 byte inode configuration.

Best practices and considerations

Recommended best practices and considerations for data inlining in OneFS 9.3 and later releases include the following:

- Data inlining is opportunistic and is only supported on node pools with 8KB inodes.

- No additional software, hardware, or licenses are required for data inlining.
- There are no CLI or WebUI management controls for data inlining.
- Data inlining is automatically enabled on applicable nodepools after an upgrade to OneFS 9.3 or later is committed.
- However, data inlining will only occur for new writes and OneFS will not perform any inlining during the upgrade process to OneFS 9.3 or later. Any applicable small files will instead be inlined upon their first write.
- Since inode inlining is automatically enabled globally on clusters running OneFS 9.3 or later, OneFS will recognize any diskpools with 512 byte inodes and transparently avoid inlining data on them.
- In OneFS 9.3 and later, data inlining will not be performed on regular files during tiering, truncation, upgrade, and so on.
- CloudPools Smartlink stubs, sparse files, and writable snapshot files are also not candidates for data inlining in OneFS 9.3 and later releases.
- OneFS shadow stores will not apply data inlining. As such:
    - Small file packing will be disabled for inlined data files.
    - Cloning will work as expected with inlined data files..
    - Inlined data files will not apply deduping and non-inlined data files that are once deduped will not inline afterwards.
- Certain operations may cause data inlining to be reversed, such as moving files from an 8KB diskpool to a 512 byte diskpool, forcefully allocating blocks on a file, sparse punching, and so on.
- In OneFS 9.3 and later, there is no file system job available to perform an assessment scan for inline inode opportunities.

Efficient storage utilization

As we have seen, OneFS data reduction techniques enable Dell PowerScale to deliver a high level of storage efficiency. While compression, deduplication, file packing, and inode inlining are the most prevalent and broadly applicable - and the feature of this paper - additional tools such as SmartQuotas thin provisioning, cloning, writable snapshots, and so on, also contribute to the overall efficiency equation.

However, one of the most significant storage efficiency attributes is the way that OneFS natively manages data protection in the file system. Unlike most file systems that rely on hardware RAID, OneFS protects data at the file level and, using software-based erasure coding, allows most customers to enjoy raw to usable utilization levels of 85% or higher. This is in contrast to the scale up NAS industry mean of around 60% raw disk capacity utilization. In-line data reduction serves to further extend this storage efficiency headroom, bringing an even more compelling and demonstrable TCO advantage to primary file-based storage.

Conclusion

Conclusion

Up until now, traditional compression and deduplication implementations have often been resource intensive, limited to software, and detrimental to performance.

The OneFS data reduction and storage efficiency tools, integrated with the industry's leading Scale-Out NAS architecture, deliver on the promise of simple data efficiency at scale by providing significant storage cost savings, without sacrificing performance, ease of use or data protection.

With its extensible configuration and transparent operation, OneFS data reduction and storage efficiency is easy to manage on your Dell PowerScale cluster, delivering enterprise data efficiency within a single, highly extensible storage pool. Scalability to petabytes and the ability to seamlessly increase capacity and add new technologies, across multiple performance tiers in the same system, means strong investment protection. Integration with OneFS core functions eliminates data risks and gives the user control over what system resources are allocated to data movement.

To learn more about data reduction and other Dell PowerScale products, see https://www.delltechnologies.com/en-us/storage/powerscale.htm#tab0=0.

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