PowerScale OneFS Small File Storage Efficiency for Archive

April 2024

H15459.11

White Paper

Abstract

This white paper describes a storage solution based on Dell PowerScale scale-out NAS that helps organizations to drive down storage management cost and complexity. OneFS storage efficiency provides a simple, scalable solution for enhancing small file storage efficiency in archive applications and workloads.

DELL Technologies

Copyright

Published in the USA April 2024 H15459.11.

Contents

Executive summary

Overview

Unstructured data continues to grow at an astonishing rate, making the need for optimized file-based data storage, and its simplified and automated management, more crucial than ever.

To help customers maximize the long-term value of their critical business data and drive down storage management cost and complexity, Dell offers Small File Storage Efficiency: A simple, scalable solution for enhancing small file storage efficiency in archive applications based on Dell PowerScale NAS storage.

Audience

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

For more details about the OneFS architecture, see the OneFS Technical Overview.

The target audience for this white paper is anyone configuring and managing a OneFS powered clustered storage environment. It is assumed that the reader has an understanding and working knowledge of the 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
February 2019	H15459	Updated for OneFS 8.1.3
April 2019	H15459.1	Updated for OneFS 8.2
August 2019	H15459.2	Updated for OneFS 8.2.1
December 2019	H15459.3	Updated for OneFS 8.2.2
June 2020	H15459.4	Updated for OneFS 9.0
September 2020	H15459.5	Updated for OneFS 9.1
April 2021	H15459.6	Updated for OneFS 9.2
September 2021	H15459.7	Updated for OneFS 9.3
April 2022	H15459.8	Updated for OneFS 9.4
January 2023	H15459.9	Updated for OneFS 9.5
February 2024	H15459.10	Updated for OneFS 9.7
April 2024	H15459.11	Updated for OneFS 9.8

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

Customers, such as those in the healthcare industry, often have datasets that are increasingly dominated by small files.

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 consist of 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

Architecture overview

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 datasets 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 128 K 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.

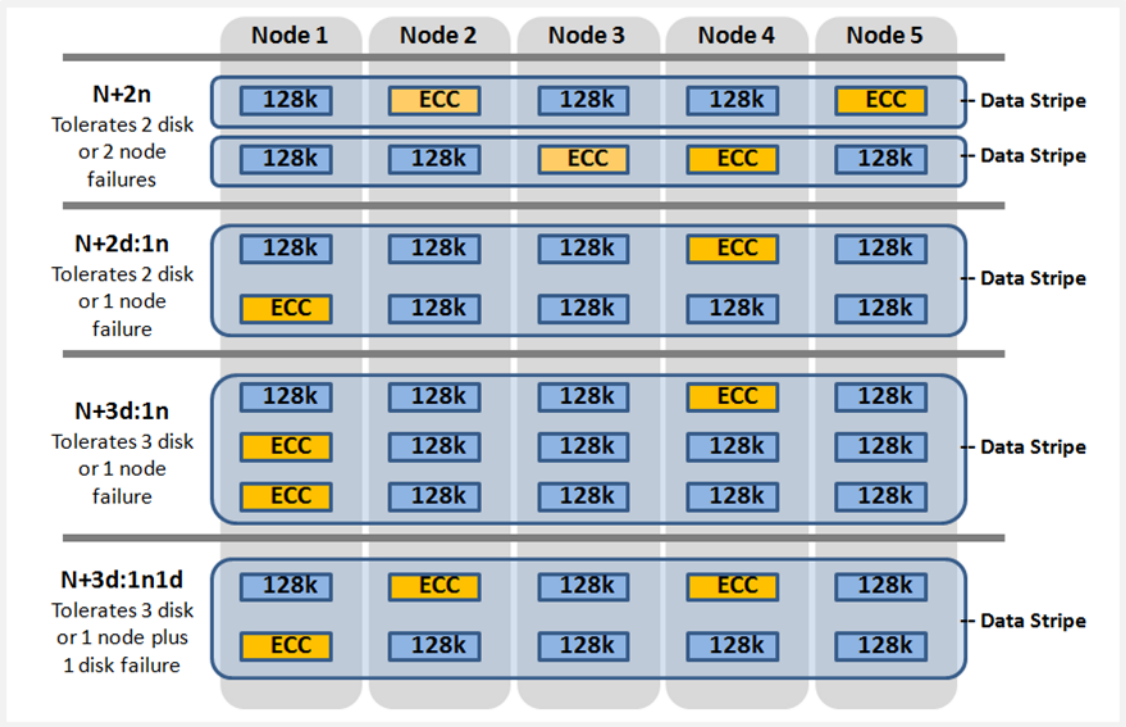


Figure 1. OneFS on-disk FEC protection

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

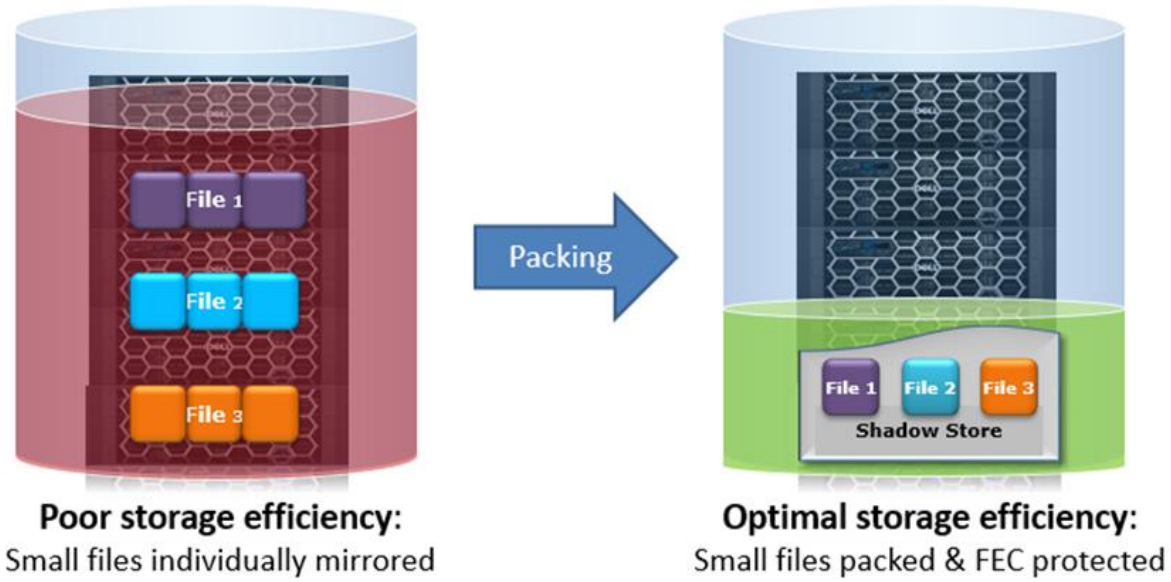


Figure 2. OneFS small file containerization

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 150 KB to 300 KB 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) 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 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 or SetProtectPlus 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 prerequisites 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 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 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 using 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
```

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 through 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 InsightIQ license, if the FSA (file system analytics) job has run, a limited efficiency report will be available. This can be viewed using the following command:

```
# isi_packing --fsa
```

For clusters using CloudPools, 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
```

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 recently modified files from being containerized. The default value for this is 24 hours, but this can be reconfigured using 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 three main CLI commands that report on the status and effect of small file efficiency:

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

In 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,
},
}
```

The second command, isi_packing –fsa, provides a storage efficiency percentage in the last line of its output. This command requires InsightIQ to be licensed on the cluster and 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.

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

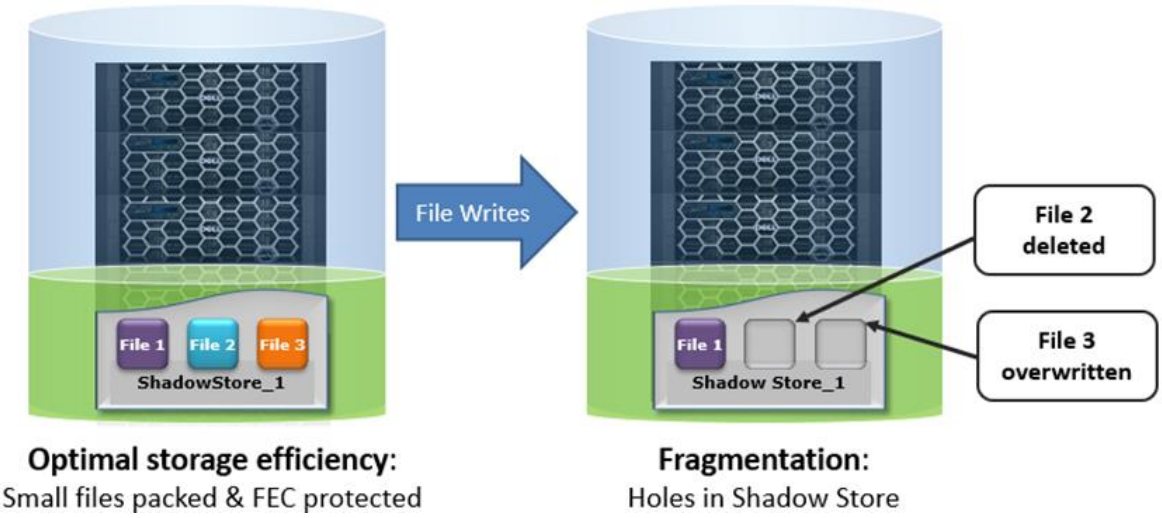
There is also a Storage Efficiency assessment tool available in OneFS 8.2 and later. This can be run as from the CLI with the following syntax: `# isi_sfse_assess <options>`

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%)
```

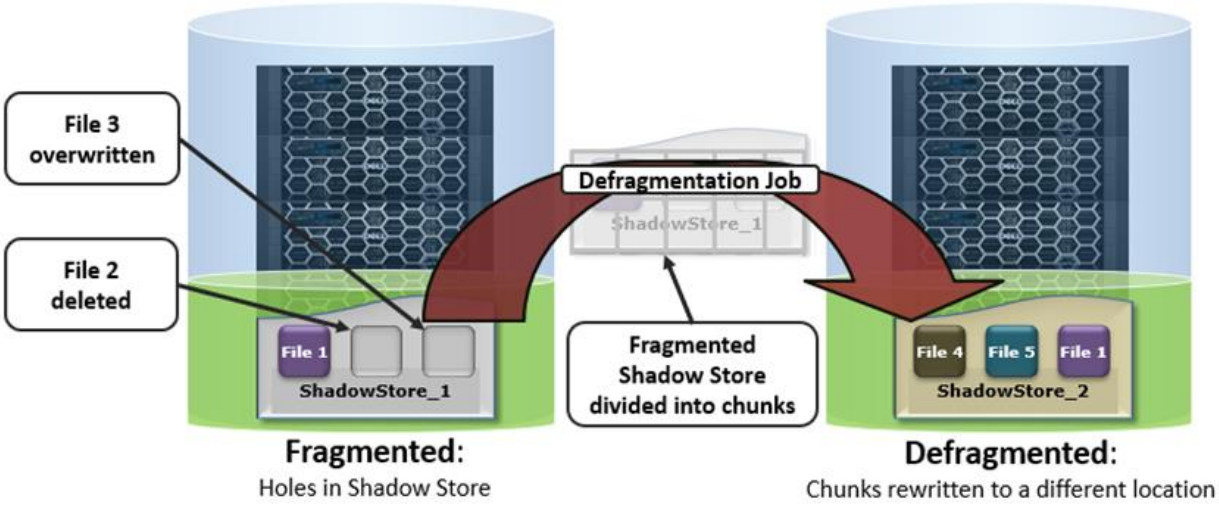
Defragmentation

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 reduces the storage efficiency.



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

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 (~32 MB each) and assessing each chunk for fragmentation.



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

In OneFS 8.2 and later, the ‘isi_sstore list’ command is enhanced to display 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
```

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 general 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 using the job engine’s ShadowStoreDelete job, either from the OneFS WebUI or using 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 and considerations

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.
- Do not configure the maximum file size value inside the file pool filter itself. Instead set this parameter using 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 (such as weekly) to eliminate fragmentation.

Considerations

Small File Storage Efficiency for Archive is not free. There is always a trade-off between cluster resource consumption (CPU, memory, disk), the potential for data fragmentation and the benefit of improved space utilization. Keep 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 through 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, or 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 deduplicated 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 SetProtectPlus, 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 datafiles (where a small file's data is stored in its inode).
- Small file packing is not supported for data contained within writable snapshots.

Summary

Until now, traditional archive implementations have typically been expensive, limited in scale, confined to secondary storage, and administratively complex. Small File Storage Efficiency integration with the industry's leading Scale-Out NAS architecture delivers on the promise of simple data efficiency at scale by providing significant storage cost savings, without sacrificing ease of use or data protection.

With its intelligent default settings, Small File Storage Efficiency is automated, extensible, and easy to manage, providing enterprise data efficiency within a single storage pool. Dell PowerScale clusters provide a scalable, multi-protocol archive solution. This solution is further evidence of the Dell commitment to developing next-generation of data management products and solutions for the enterprise.