Data Protection with Dell PowerScale SnapshotIQ

April 2024

H15048.18

White Paper

Abstract

This paper focuses on Dell PowerScale OneFS, a modern file system that meets the unique needs of big data and PowerScale SnapshotIQ. Snapshot IQ is a data protection capability that helps to reduce storage costs and increase data availability.

Copyright

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.

This proliferation of unstructured data has left traditional storage architectures unable to satisfy the demands of this growth and has necessitated the development of a new generation of storage technologies. Also, broader data retention requirements, regulatory compliance, tighter availability service level agreements (SLAs) with internal or external customers, and cloud and virtualization initiatives are only serving to compound this issue.

Audience

This paper presents information for deploying and managing snapshots on a Dell PowerScale 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 about the OneFS architecture.

The target audience for this white paper is anyone configuring and managing snapshots in a PowerScale 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
November 2013	H15048	Initial release for OneFS 7.1
June 2014	H15048.1	Updated for OneFS 7.1.1
November 2014	H15048.2	Updated for OneFS 7.2
June 2015	H15048.3	Updated for OneFS 7.2.1
November 2015	H15048.4	Updated for OneFS 8.0
September 2016	H15048.5	Updated for OneFS 8.0.1
April 2017	H15048.6	Updated for OneFS 8.1
November 2017	H15048.7	Updated for OneFS 8.1.1
February 2019	H15048.8	Updated for OneFS 8.1.3
April 2019	H15048.9	Updated for OneFS 8.2
August 2019	H15048.10	Updated for OneFS 8.2.1
December 2019	H15048.11	Updated for OneFS 8.2.2
June 2020	H15048.12	Updated for OneFS 9.0

Date	Part number/ revision	Description
September 2020	H15048.13	Updated for OneFS 9.1
April 2021	H15048.14	Updated for OneFS 9.2
September 2021	H15048.15	Updated for OneFS 9.3
March 2022	H15048.16	Updated for OneFS 9.4
January 2023	H15048.17	Updated for OneFS 9.5
January 2024	H15048.18	Updated for OneFS 9.7
April 2024	H15048.19	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.

Snapshots overview

The availability and protection of data can be effectively described in terms of a continuum:

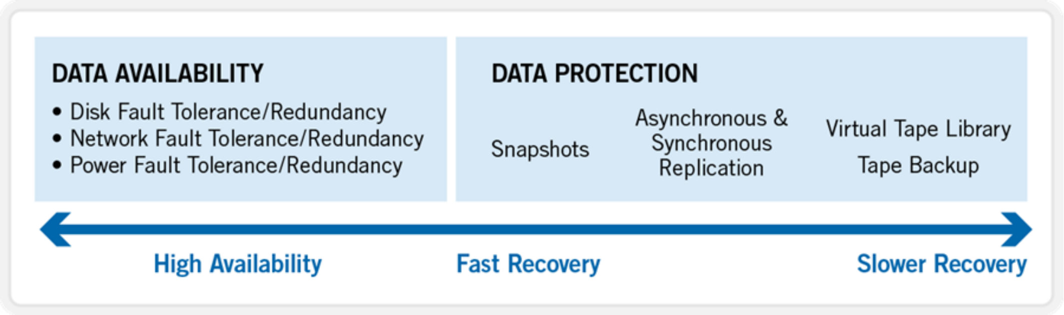

DATA AVAILABILITY
• Disk Fault Tolerance/Redundancy
• Network Fault Tolerance/Redundancy
• Power Fault Tolerance/Redundancy
DATA PROTECTION
Snapshots
Asynchronous & Synchronous Replication
Virtual Tape Library
Tape Backup
High Availability
Fast Recovery
Slower Recovery

Figure 1. Data protection continuum

At the beginning of the continuum sits high availability. This requirement is usually satisfied by redundancy and fault tolerant designs. The goal here is continuous availability and the avoidance of downtime by the use of redundant components and services.

Further along the continuum lie the data recovery approaches in order of decreasing timeliness. These solutions typically include point-in-time snapshots for fast recovery, followed by replication and, finally, backup to tape or a virtual tape library.

The following diagram illustrates how the core components of the PowerScale data protection portfolio align with the notion of an availability and protection continuum and associated recovery objectives.

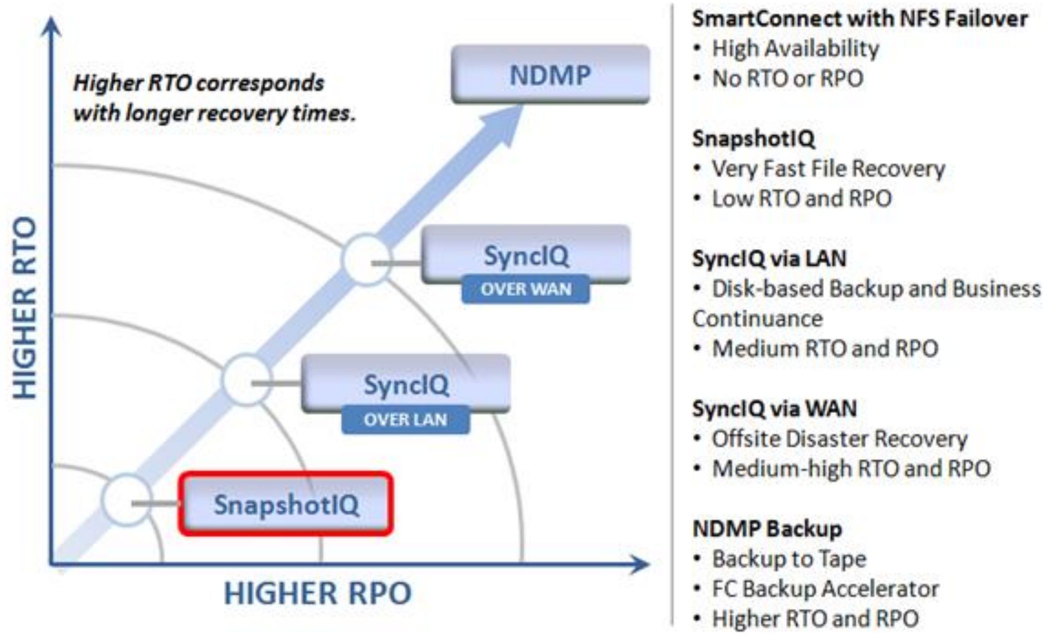


Figure 2. PowerScale data protection technology alignment with protection continuum

The recovery time objective (RTO) of a snapshot can be very small and the recovery point objective (RPO) is also highly flexible with the use of rich policies and granular schedules.

Data protection with SnapshotIQ

SnapshotIQ can take read-only, point-in-time copies (snapshots) of any directory or subdirectory within OneFS. When a snapshot is taken, it preserves the exact state of a file system at that instant, which can then be accessed later. This immutable, point-in-time copy has various applications. For example, snapshots can be used to make consistent backups, or to restore files which were inadvertently changed or deleted. Snapshots are also for quickly identifying file system changes.

Typically, only a small percentage of the file system changes in a given time, so most storage systems do not keep fully redundant copies. Instead, generally, only the differences from the current file system state are stored.

A OneFS snapshot is basically a logical pointer to data that is stored on a cluster at a particular point in time. Each snapshot references a specific directory under OneFS and includes all the files stored in that directory and its subdirectories. If the data referenced by a snapshot is modified, the snapshot stores a physical copy of the data that was modified. Snapshots are either created according to user configuration or are automatically generated by OneFS to facilitate system operations.

Snapshots can be identified and located either by a unique name or a system-generated snapshot ID. A snapshot name is specified by a user and assigned to the virtual directory which contains the snapshot. A snapshot ID is a numerical identifier that OneFS automatically assigns to a snapshot.

OneFS Snapshots are highly scalable and typically take less than one second to create. They create little performance overhead, regardless of the level of activity of the file system, the size of the file system or the size of the directory being copied. Also, only the changed blocks of a file are stored in a snapshot thereby ensuring highly-efficient storage capacity utilization. User access to the available snapshots is over a special hidden ‘portal’ directory under each file system directory.

SnapshotIQ can also create up to twenty thousand snapshots on a cluster. This large quantity provides a substantial benefit over most other snapshot implementations, because it allows the snapshot intervals to be far shorter, and hence offer more granular recovery point objectives (RPOs).

Snapshot architecture

SnapshotIQ has several fundamental differences as compared to most snapshot implementations. The most significant of these are:

- Directory based: OneFS snapshots are per-directory based. This is in contrast to the traditional approach where snapshots are taken at a file system or volume boundary.
- Logical snapshot process: Since OneFS manages and protects data at the file-level, there is no inherent block-level indirection layer for snapshots to use. Instead, OneFS takes copies of files or pieces of files (logical blocks and inodes) in what is termed a logical snapshot process.
- Snapshot space allocation: There is no requirement for reserved space for snapshots in OneFS. Snapshots can use as much or little of the available file system space as desirable. A snapshot reserve can be configured if preferred, although this will be an accounting reservation rather than a hard limit.

The process of capturing a snapshot in OneFS is near instantaneous. However, there is a small amount of snapshot preparation work that has to occur. The moment a snapshot is taken, it consumes zero space until any file writes occur to the data it contains.

Any changes to a dataset are then recorded in the pertinent snapshot inodes, which contain only referral (‘ditto’) records, until any of the logical blocks they reference are altered or another snapshot is taken. In order to reconstruct data from a particular snapshot, OneFS will look though all of the more recent versions snapshot tracking files (STFs) until it reaches HEAD (current version). In so doing, it will systematically find all the changes and recreate the point-in-time view of that dataset.

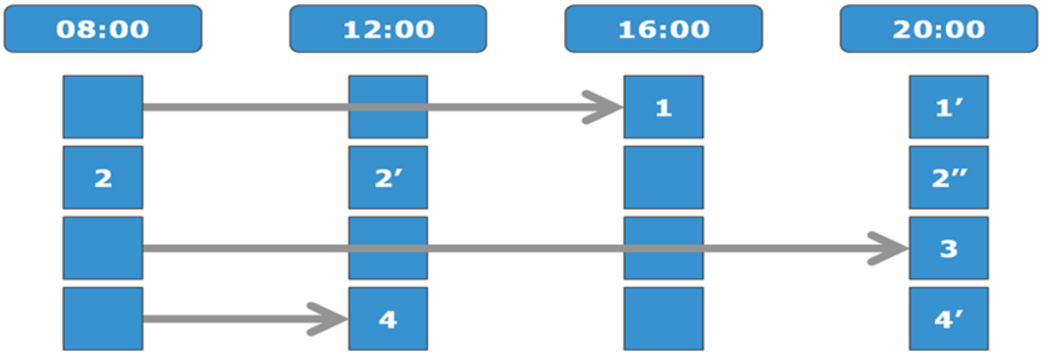


Figure 3. Snapshot read chain

In this example, to reconstruct file 1-2-3-4 as it appeared at 08:00 SnapshotIQ would need to read snapshots 12:00, 16:00 (4PM) and 20:00 (8PM).

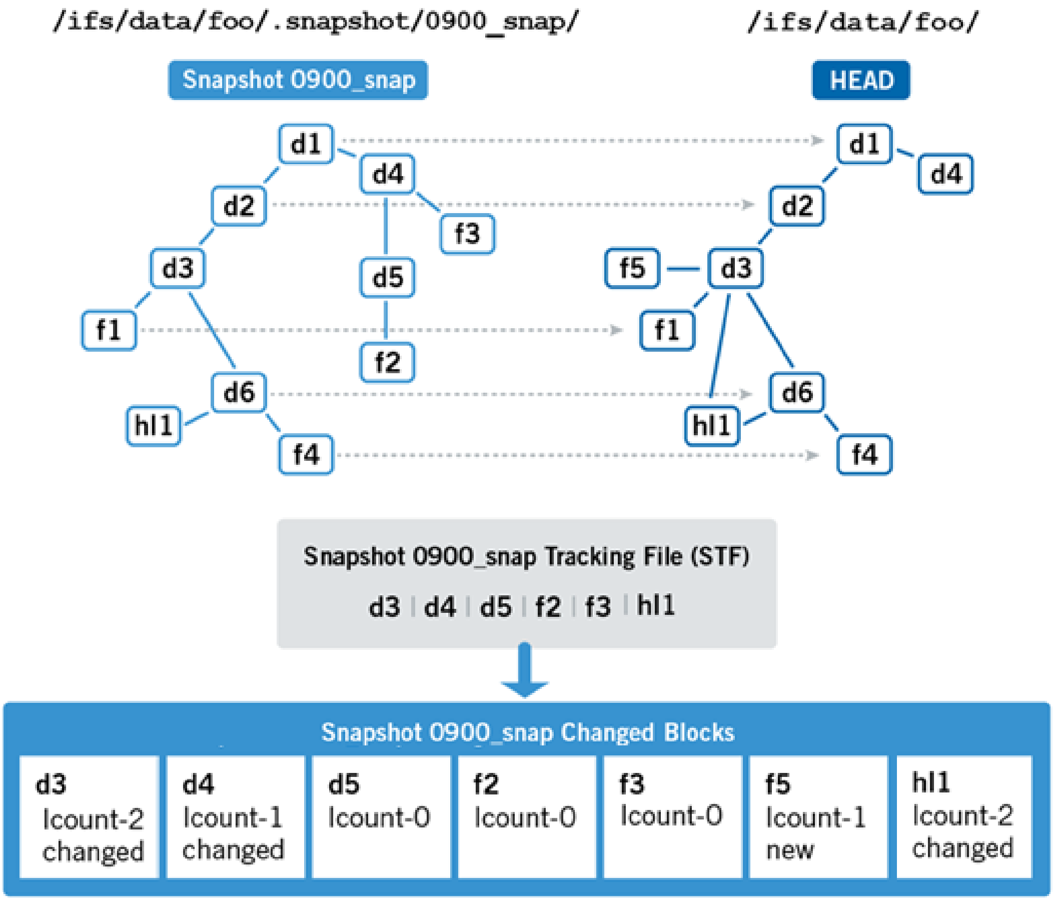
In extreme cases, OneFS might have to ‘paint’ through hundreds or thousands of snapshots to reconstruct the correct older version.

Snapshot tracking files

Snapshot tracking files (STF) are the main data structure associated with a snapshot. A snapshot tracking file has three major purposes:

- Indicating which snapshots are active.
- Storing snapshot attributes, such as usage data, creation time, and root directory paths.
- Recording a list of LINs modified in the snapshot, which can be freed when the snapshot is deleted.

Snapshot Tracking Files are a special file type with several unique characteristics, and are involved in the full snapshot life cycle, including the creation, storing any changes, and deletion of snapshots.



Snapshot 0900_snap Tracking File (STF)

d3 | d4 | d5 | f2 | f3 | hl1

Snapshot 0900_snap Changed Blocks

d3	d4	d5	f2	f3	f5	hl1
lcount-2 changed	lcount-1 changed	lcount-0	lcount-0	lcount-0	lcount-1 new	lcount-2 changed

Figure 4. Snapshot change tracking

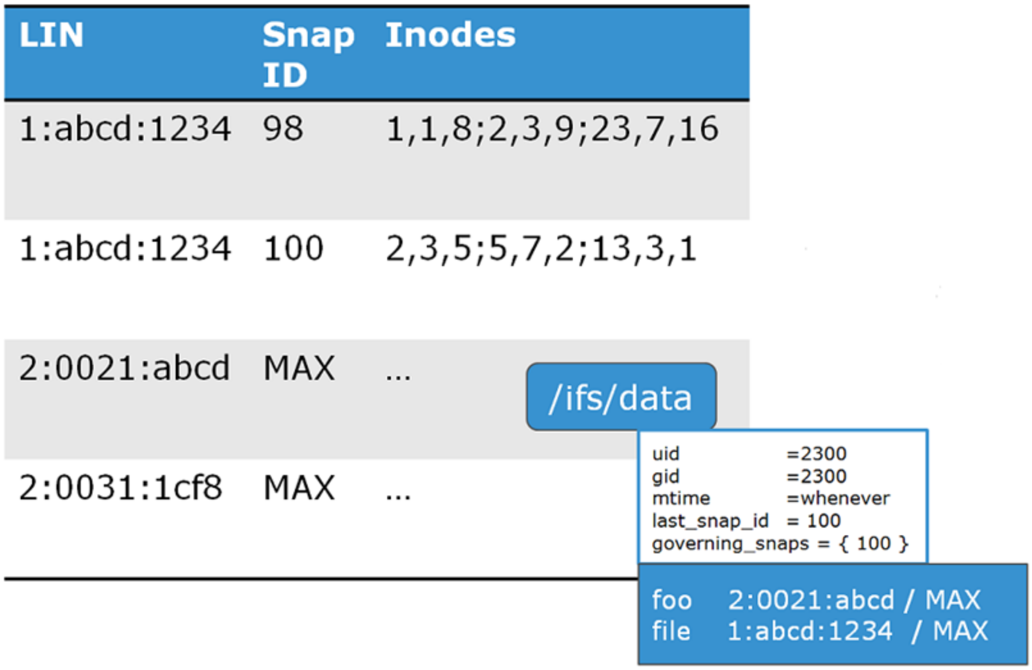
LIN table

Finding the next version of a particular file is a fast operation due to the structure of the LIN (logical inode) table. While the LIN table is frequently called a table, it is actually a B-Tree. It is sorted by (LIN,Version), so finding the next newer version of a LIN is an inexpensive operation.

A LIN is fundamentally the identity of an object and possess the following attributes:

- Directories point to LINs.
- The LIN Table maps a LIN to blocks on disk for the inode.
- The Inode contains the map of data ranges to blocks on disk.
- The LIN table maps the LIN,SnapID that identifies a file version to the location on disk that houses the metadata about that file.

Given a LIN:SnapID pairing, finding the next higher SnapID is a fast and efficient process, and the HEAD version is always represented by the biggest SnapID.



LIN	Snap ID	Inodes
1:abcd:1234	98	1,1,8;2,3,9;23,7,16
1:abcd:1234	100	2,3,5;5,7,2;13,3,1
2:0021:abcd	MAX	...
2:0031:1cf8	MAX	...

uid =2300
gid =2300
mtime =whenever
last_snap_id = 100
governing_snaps = { 100 }

foo	2:0021:abcd / MAX
file	1:abcd:1234 / MAX

Figure 5. LIN table example

In the example above, 1:abcd:1234/MAX is not in the LIN Tree. This suggests that the file was deleted sometime after snapshot 100 was taken.

Copy or redirect on write

SnapshotIQ uses both copy on write (CoW) and redirect on write (RoW) strategies for its differential snapshots and uses the most appropriate method for a given situation. Both have pros and cons, and OneFS dynamically picks which flavor to use in order to maximize performance and keep overhead to a minimum.

With copy on write, as the name suggests, a new write to HEAD results in the old blocks being copied out to the snapshot version first. Although this incurs a double write penalty, it results in less fragmentation of the HEAD file, which is better for cache prefetch. Typically, CoW is most prevalent in OneFS and is primarily used for small changes, inodes, and directories.

Redirect on write, on the other hand, avoids the double write penalty by writing changes to a snapshot protected file directly to another free area of the file system. However, the flip side to this is increased file fragmentation. Because file contiguity is not maintained by virtue of writing changes to other file system regions, RoW in OneFS is used for more substantial changes such as deletes and large sequential writes.

Note: There is no reserved space requirement for snapshots in OneFS. Snapshots can use as much or little of the available file system space as desirable. A snapshot reserve can be configured, if preferred.

Accessing snapshots through the '.snapshot' directory

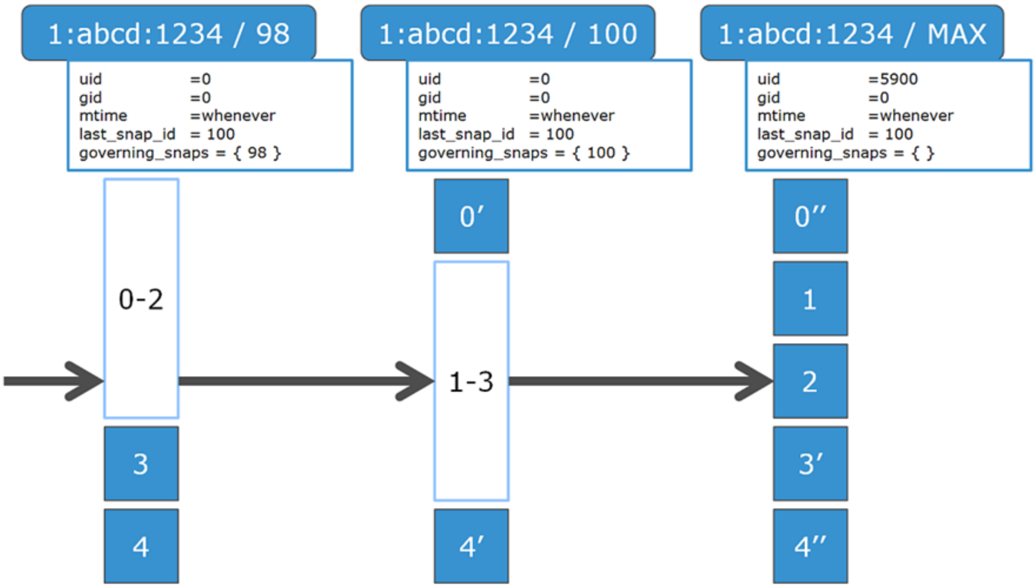
The '.snapshot' directory is the namespace entry point for accessing snapshots. It is also referred to as a 'portal' since it provides a path from the HEAD version of the file system into the associated snapshot(s). Much like the UNIX '.' and '..' directory notations, the '.snapshot' directory is a virtual entry in that it does not have an associated record in the directory B-tree, much like '.' and '..'.

Within the '.snapshot' directory, there is a virtual entry for every snapshot which either governs the parent directory or was taken on a descendant of the directory. The virtual entries consist of the parent directory's LIN and the snapshot's ID. This is implemented by listing the snapshot names directory and filtering the entries which do not apply. If a process opens an entry, it gets a snapshot version of the portal's parent. The '.snapshot' portal can also be disabled entirely, have access restricted to the root and subdirectories, and be renamed to 'snapshot', rather than '.snapshot'.

Snapshot management

Reading from a snapshot

The following figure shows three snapshots of a given file with logical inode (LIN) value "1:abcd:1234." In this case, blocks 3 and 4 were changed after the first snapshot (Snap_ID 98) was taken and before the second (Snap_ID 100), and blocks 0 and 4 were changed after the second snapshot was taken.



```
uid            =0
gid            =0
mtime          =whenever
last_snap_id  = 100
governing_snaps = { 98 }
```

```
uid            =0
gid            =0
mtime          =whenever
last_snap_id  = 100
governing_snaps = { 100 }
```

```
uid            =5900
gid            =0
mtime          =whenever
last_snap_id  = 100
governing_snaps = { }
```

Figure 6. Reading from a snapshot

When the data is not in the snapshot, the block tree of the inode on the snapshot does not point to a real data block. Instead it has a flag marking it as a "Ditto Block." A Ditto-block means that the data is the same as the next newer version of the file, so OneFS will automatically look ahead to find the newer version of the block.

The arrow represents the logic for reading block 2 from snapshot 98. Since it was not changed in snapshot 98, the read has to fall forward to snapshot 100. It was not changed there either, so the write falls forward to the head version of the file.

The performance implications here should be clear: If you have thousands of snapshots of the same unchanged file, reading from the oldest snapshot can potentially be somewhat slow.

Painting algorithm

When a file is written to, the system needs to do a small amount of work to determine if the file is part of a snapshot. If so, a copy of the old data needs to be kept. This is done using a process known as the “painting algorithm.”

The system keeps the most recent snapshot ID in a cluster-wide global variable. When a file is modified, OneFS looks first at the file’s last_snap_id. If the last_snap_id is not the most recent snap_id, there is a likelihood that the governing_snaps information in the file is out of date. In this case, OneFS recursively searches the parent directories until it finds up-to-date information, and then uses the correct directory’s governing-snaps information. For example:

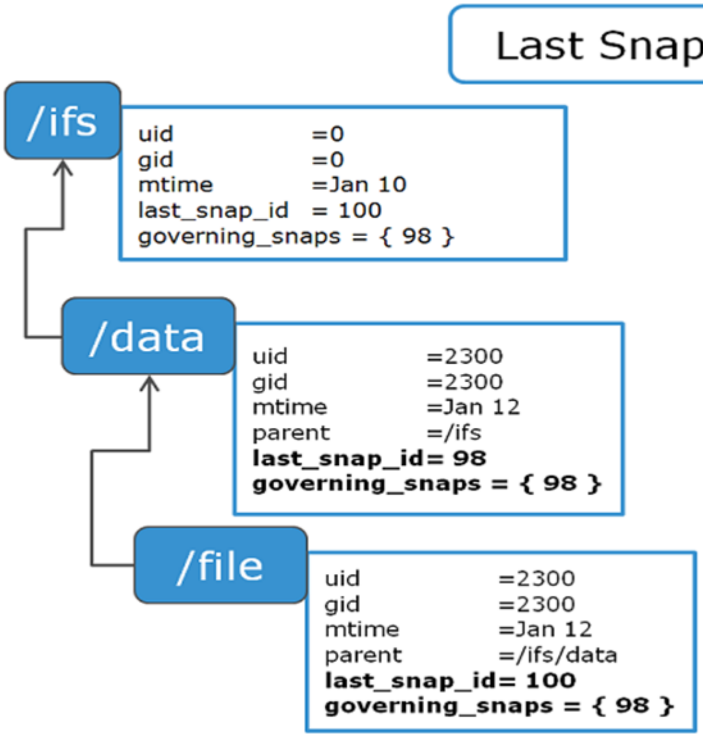


Figure 7. SnapshotIQ painting algorithm

In the instance above, /ifs has a governing_snaps ID of “98”. This implies that snapshots 98 and 99 were taken on another directory; possibly /ifs/home.

Snapshot “painting” is an expensive operation that has to be performed for every file in the system whenever a new snapshot is taken – even on files which are not part of the snapshot.

Snapshot domains

In OneFS 8.2 and later, SnapshotIQ adopts a domains model for governance of scheduled snapshots. By using the OneFS IFS domains infrastructure, recurring snapshot efficiency and performance is increased by limiting the scope of governance to a smaller, well defined domain boundary.

IFS Domains provide a Mark Job that proactively marks all the files in the domain. Creating a snapshot on a fully marked domain will not cause further "painting" operations, thereby avoiding a significant portion of the resource overhead caused by taking a new snapshot.

Once a domain has been fully marked, subsequent snapshot creation operations will not cause any further painting. The new snapshot ID is simply added to the domain data section, so the creation of a new snapshot will not trigger a system-wide painting event anymore. Domains are reused whenever possible.

Creating two domains of the same type on the same directory will cause the second domain to become an alias of the first domain. Aliases do not require marking since they share the existing marks. This benefits both snapshots and snapshot schedules taken on the same directory. For all these reasons, the number of I/O and locking operations needed to resolve snapshot governance is greatly reduced. Because the SnapIDs are stored in a single location (as opposed to being stored on individual inodes), this greatly simplifies Snapshot ID garbage collection whenever a Snapshot is deleted. By leveraging IFS Domains, creating a snapshot on a domain that is fully marked will not cause further "painting" operations, so a significant portion of the performance impact caused by taking a new snapshot is avoided.

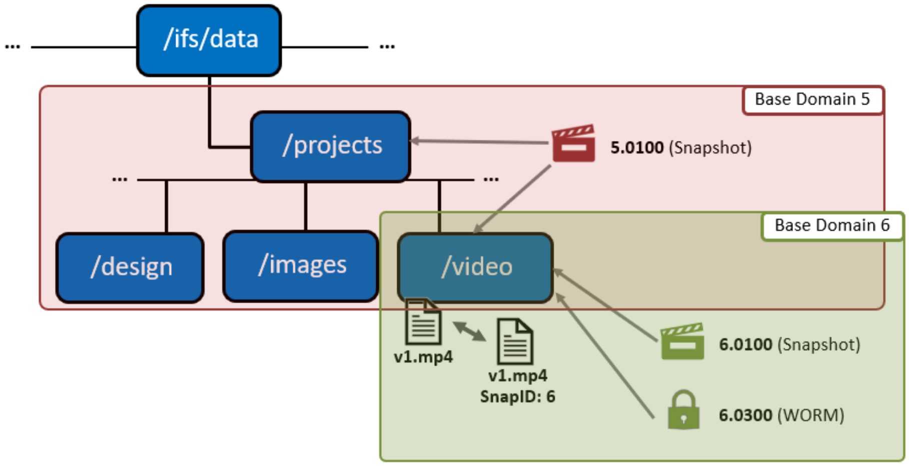


Figure 8. Scheduled snapshots and IFS Domains

The figure above shows an example of domain-based snapshots. In this case, a snapshot was taken on the 'projects' directory, and on the directory named 'video'. File v1.mp4 is tagged with the domain IDs, making it more efficient to determine snapshot governance.

A snapshot of file v1.mp4 creates a snap_ID in the domain's SBT (system b-tree) providing a single place to store snapshot metadata. In previous OneFS versions, snapIDs were stored in the inode, which resulted in duplication of the snap_IDs and metadata usage.

Only snapshots taken after an upgrade to OneFS 8.2 or later will use IFS domains backing. Any snapshots created prior to the upgrade will not be converted and will remain in their original form.

Also, IFS Domains brings other benefits including:

- Improved management of SnapIDs

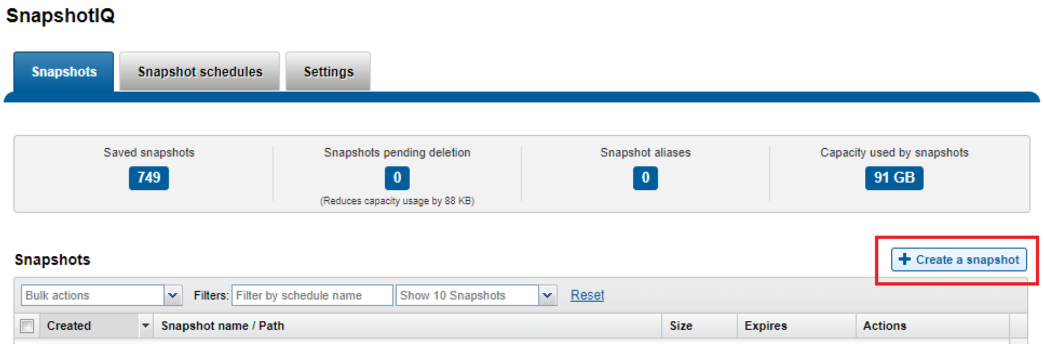
- Reduced number of operations needed to resolve snapshot governance.
- More efficient use of metadata
- The automatic exclusion of the cluster’s /ifs/.ifsvar subtree from all root (/ifs) snapshots – although this behavior is configurable.
- The write cache, or coalescer, is enhanced to better support parallel snapshot creates.
- The snapshot create path is improved to reduce contention on the STF during copy-on-write.

Taking a snapshot

There are two methods to create user snapshots:

- On-demand.
- From a preconfigured snapshot schedule.

The first, on-demand, method is initiated using the simple ‘Capture a new snapshot’ button in the ‘snapshot summary’ tab of the OneFS user interface.



SnapshotIQ

Saved snapshots 749

Snapshots pending deletion 0 (Reduces capacity usage by 88 KB)

Snapshot aliases 0

Capacity used by snapshots 91 GB

Snapshots

Filters: Filter by schedule name

Show 10 Snapshots

Figure 9. Taking an on-demand snapshot

On the backend, creating a snapshot involves synchronizing changes across several levels, from the write-back cache (coalescer) to the snapshot tracking file (STF). The main elements of this process are:

1. The snapshot acquires an exclusive snapshot lock.
2. Coalescers are coordinated and suspended across all nodes.
3. The write lock is upgraded to exclusive.
4. The maximum snapshot ID is incremented.
5. The snapshot (and associated minisnaps) is generated up to the snapshot root.
6. A snapshot tracking file is created and updated.
7. If requested, a snapshot alias is created.

Writing to a snapshot

This following figure illustrates the process of updating a file’s metadata, by way of a simple UID change. When this occurs, and OneFS detects that it needs to duplicate the old data, it copies the old blocks to the snapshot. In this case, there was no LIN 1:abcd:1234/98 prior to the write, so any attempt to read this LIN would have fallen forward to the head version.

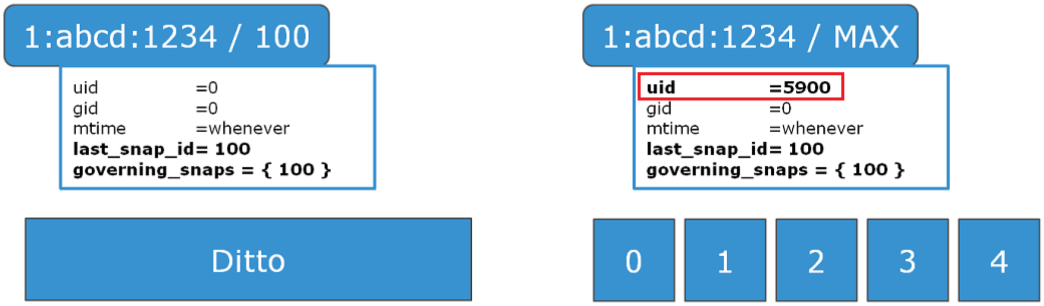


Figure 10. Changing the UID of a file in a snapshot – Part 1

When a file’s metadata is changed, OneFS needs to create a copy of the data, and increment the Snap_ID. Since there is no actual data written, there is no need to copy over any old data blocks to the new snapshot, so ditto-blocks are used instead. However, the first time a file is changed after a snapshot is taken, the LIN is recorded in the snapshot tracking file (STF). This STF lets OneFS efficiently know what data can be removed when a snapshot is deleted.

The file being written to is unchanged since the last snapshot. The data blocks from the head version of the file are copied to the snapshot version, replacing the ditto-blocks, and the head version is overwritten with the new data. This is an example of copy on write (COW).

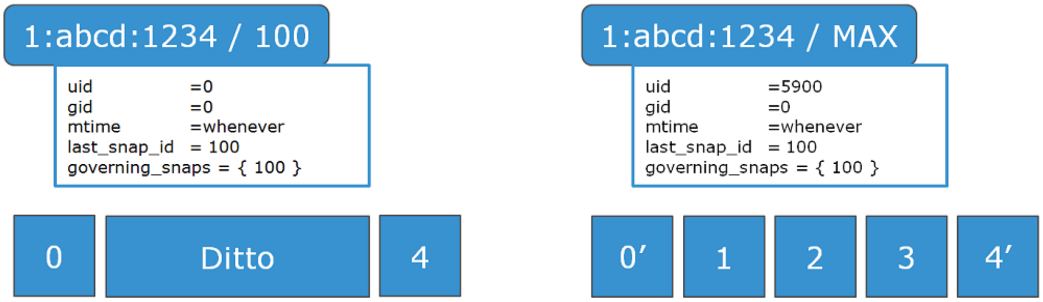


Figure 11. Changing the UID of a file in a snapshot – Part 2

Deleting a snapshot

When snapshots are manually deleted, OneFS marks the appropriate snapshot IDs and queue a job engine job to affect their removal. The SnapshotDelete job is queued immediately, but the job engine will typically wait a minute or so to actually start running it. During this interval, the snapshot will be listed as ‘delete pending’.

A similar procedure occurs with expired snapshots. Here, the snapshot daemon is responsible for checking expiration of snapshots and marking them for deletion. The daemon performs the check every 10 seconds. The job is queued to delete a snapshot completely. It is then up to the job engine to schedule it. The SnapshotDelete job might run immediately (after a minute or so of waiting) if the job engine determines that the job is executable and there are no other contending jobs with higher priority running at that moment. For SnapshotDelete, it is only run if the cluster is in a fully available state (no drives or nodes are down).

The most efficient method for deleting multiple snapshots simultaneously is to process older through newer, and SnapshotIQ will automatically attempt to orchestrate deletes in this manner. A SnapshotDelete job engine schedule can also be defined so that snapshot deletes only occur during wanted times.

On deletion of a snapshot, OneFS immediately simply modifies some of the tracking data and the snapshot disappears from view. However, the actual behind-the-scenes clean-up of the snapshot can involve a fair amount of work, which is performed in the SnapshotDelete job.

In the following example, snapshot ID 100 is being deleted. To accomplish this, any changes will likely need to be moved to the prior snapshot (ID 98), because that snapshot will no longer be able to read forward.

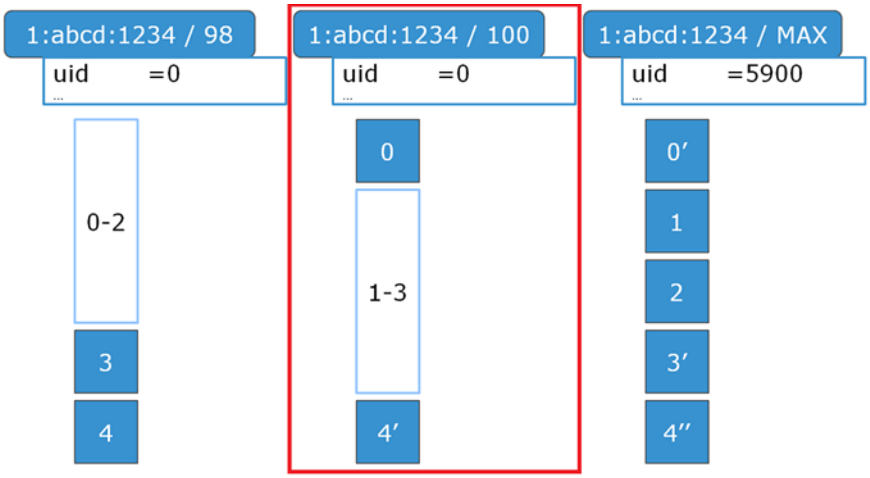


Figure 12. Deleting a snapshot – Part 1

Snapshot 100 has two changed blocks: block 0 and block 4. Snapshot 98 was changed after snapshot 98 was taken, so block 4 can be deleted, but block 0 needs to be moved over to snapshot 98.

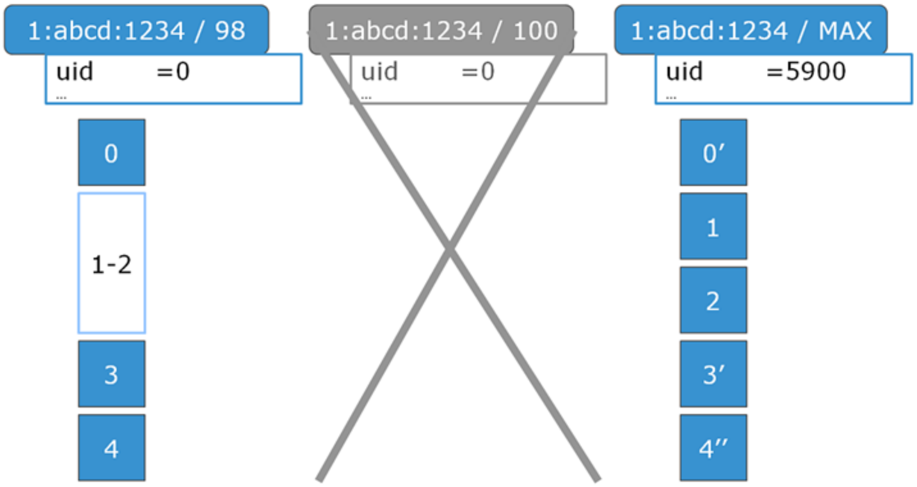


Figure 13. Deleting a snapshot – Part 2

The oldest snapshot can be deleted very quickly. An ordered deletion is the deletion of the oldest snapshot of a directory and is a recommended best practice for snapshot management. An unordered deletion is the removal of a snapshot that is not the oldest in a directory. This can often take approximately twice as long to complete and consume more cluster resources than ordered deletions.

Restoring a snapshot

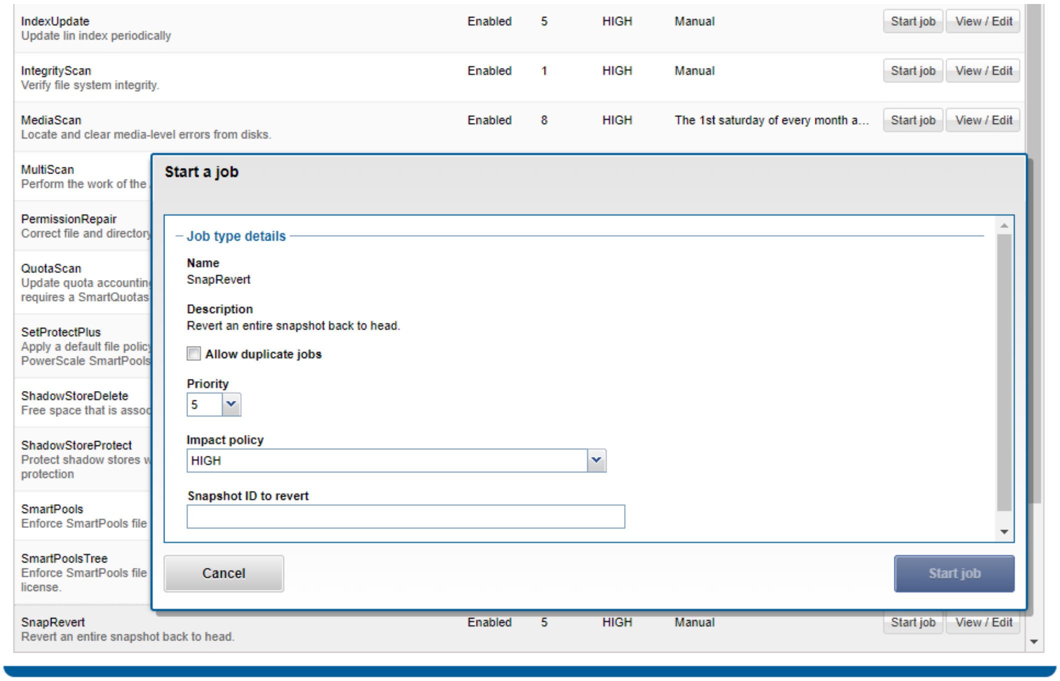
There are three main methods for restoring data from a snapshot:

- Copying specific files and directories directly from the snapshot
- Cloning a file from the snapshot
- Reverting the entire snapshot using the SnapRevert job

Copying a file from a snapshot duplicates that file, which roughly doubles the amount of storage space it consumes. Even if the original file is deleted from HEAD, the copy of that file will remain in the snapshot.

Cloning a file from a snapshot also duplicates that file. However, unlike a copy, a clone does not consume any additional space on the cluster, unless either the clone or original file is modified. File cloning is covered in more detail towards the end of this paper.

However, the most efficient of these approaches is the SnapRevert job, which automates the restoration of an entire snapshot to its top-level directory. This is invaluable for quickly reverting to a previous, known-good recovery point, for example, if a virus or malware outbreak occurs. The SnapRevert job can be run from the Job Engine WebUI, and requires adding the wanted snapshot ID.



IndexUpdate Update lin index periodically	Enabled	5	HIGH	Manual	Start job · View / Edit
IntegrityScan Verify file system integrity.	Enabled	1	HIGH	Manual	Start job · View / Edit
MediaScan Locate and clear media-level errors from disks.	Enabled	8	HIGH	The 1st saturday of every month a…	Start job · View / Edit
MultiScan Perform the work of the					
PermissionRepair Correct file and directory					
QuotaScan Update quota accountin requires a SmartQuotas					
SetProtectPlus Apply a default file policy PowerScale SmartPools					
ShadowStoreDelete Free space that is assoc					
ShadowStoreProtect Protect shadow stores w protection					
SmartPools Enforce SmartPools file					
SmartPoolsTree Enforce SmartPools file license.					
SnapRevert Revert an entire snapshot back to head.	Enabled	5	HIGH	Manual	Start job · View / Edit

Start a job

Job type details

Name
SnapRevert

Description
Revert an entire snapshot back to head.

☐ Allow duplicate jobs

Priority
5

Impact policy
HIGH

Snapshot ID to revert

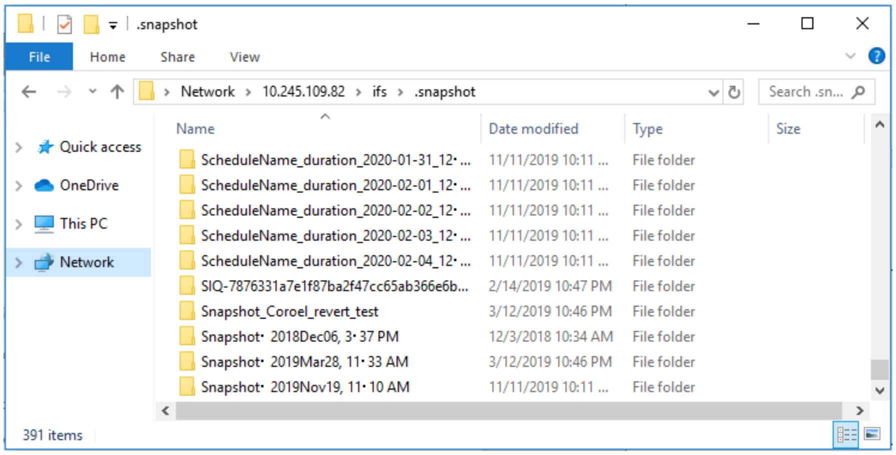
Figure 14. Running a snapshot revert job

Before a snapshot is reverted, SnapshotIQ creates a snapshot of the data that is being replaced. This enables the snapshot revert action to be undone later, if necessary.

Also, individual files, rather than entire snapshots, can also be restored in place using the isi_file_revert command-line utility. This can help drastically simplify virtual machine management and recovery.

User driven file recovery

With the appropriate access credentials and permissions, NFS and SMB users can view and recover data from OneFS snapshots. The snapshots are accessed through the '.snapshot' directory, as described previously in this paper. The following screenshot from a Windows client shows the list of snapshots available on a OneFS SMB share:



Name	Date modified	Type	Size
ScheduleName_duration_2020-01-31_12·...	11/11/2019 10:11 ...	File folder	
ScheduleName_duration_2020-02-01_12·...	11/11/2019 10:11 ...	File folder	
ScheduleName_duration_2020-02-02_12·...	11/11/2019 10:11 ...	File folder	
ScheduleName_duration_2020-02-03_12·...	11/11/2019 10:11 ...	File folder	
ScheduleName_duration_2020-02-04_12·...	11/11/2019 10:11 ...	File folder	
SIQ-7876331a7e1f87ba2f47cc65ab366e6b...	2/14/2019 10:47 PM	File folder	
Snapshot_Coroel_revert_test	3/12/2019 10:46 PM	File folder	
Snapshot· 2018Dec06, 3·37 PM	12/3/2018 10:34 AM	File folder	
Snapshot· 2019Mar28, 11·33 AM	3/12/2019 10:46 PM	File folder	
Snapshot· 2019Nov19, 11·10 AM	11/11/2019 10:11 ...	File folder	

391 items

Figure 15. Accessing snapshots from the '.snapshot' portal

In the example below, a user accidentally deletes a file `'/ifs/data/foo/bar.txt'` at 9.10am and notices it is gone a couple of minutes later. By accessing the 9am snapshot, the user can recover the deleted file themselves at 9.14am, by copying it directly from the snapshot directory `'/ifs/data/foo/.snapshot./0900_snap/bar.txt'` back to its original location at `'/ifs/data/foo/bar.txt'`.

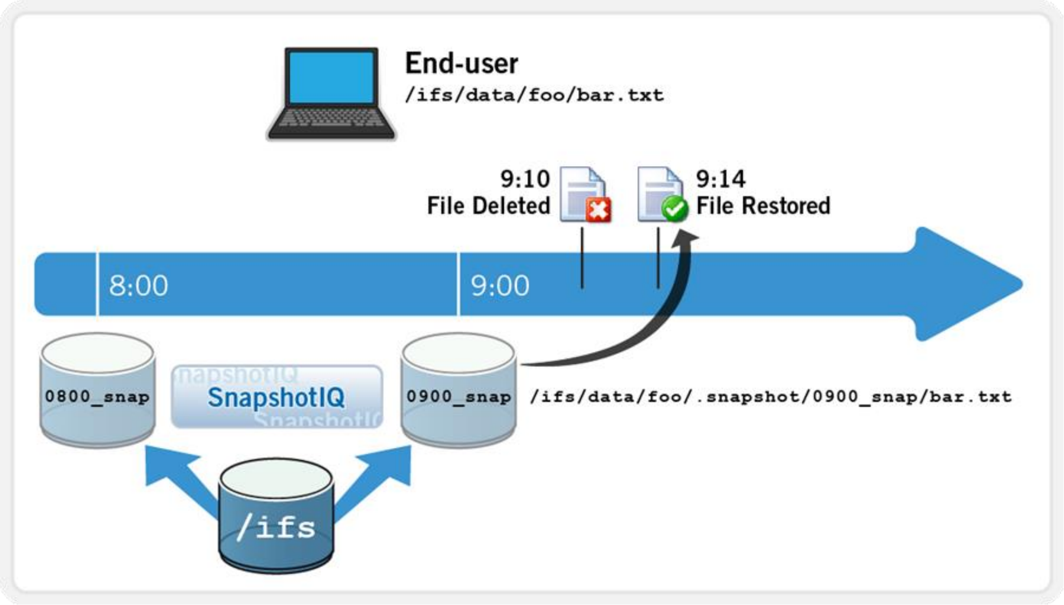


Figure 16. User driven file recovery with SnapshotIQ

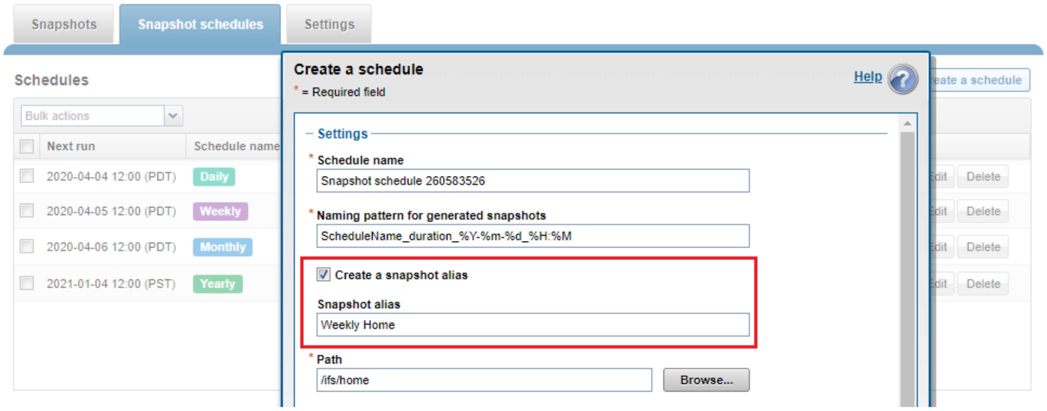
SnapshotIQ integration with Windows Volume Snapshot Manager allows Windows users a simple way to restore data on a cluster's SMB share from the "Previous Versions" tab on their desktop.

Snapshot aliases

As mentioned earlier, a snapshot can be referenced by a common name, in addition to a system defined snapshot ID. Snapshot names must be unique, and often incorporate a date, timestamp, and other context in their nomenclature.

Most users and applications are interested in referencing a snapshot relative to the HEAD version of data. Common naming schemes include, for example, “latest-weekly” or “latest-monthly,” for which the underlying snapshot ID changes as time goes by. Snapshot aliases provide for snapshot name indirection and enable the administrator to create user-friendly names, which can point at a different underlying snapshot over time.

Aliases are added and modified using the --alias option to the snapshot create and modify CLI subcommands, or through the snapshot schedule configuration pages of the graphical WebUI.



□	Next run	Schedule name
□	2020-04-04 12:00 (PDT)	Daily
□	2020-04-05 12:00 (PDT)	Weekly
□	2020-04-06 12:00 (PDT)	Monthly
□	2021-01-04 12:00 (PST)	Yearly

Create a schedule

* = Required field

Settings

* Schedule name: Snapshot schedule 260583526

* Naming pattern for generated snapshots: ScheduleName_duration_%Y-%m-%d_%H:%M

☑ Create a snapshot alias

Snapshot alias: Weekly Home

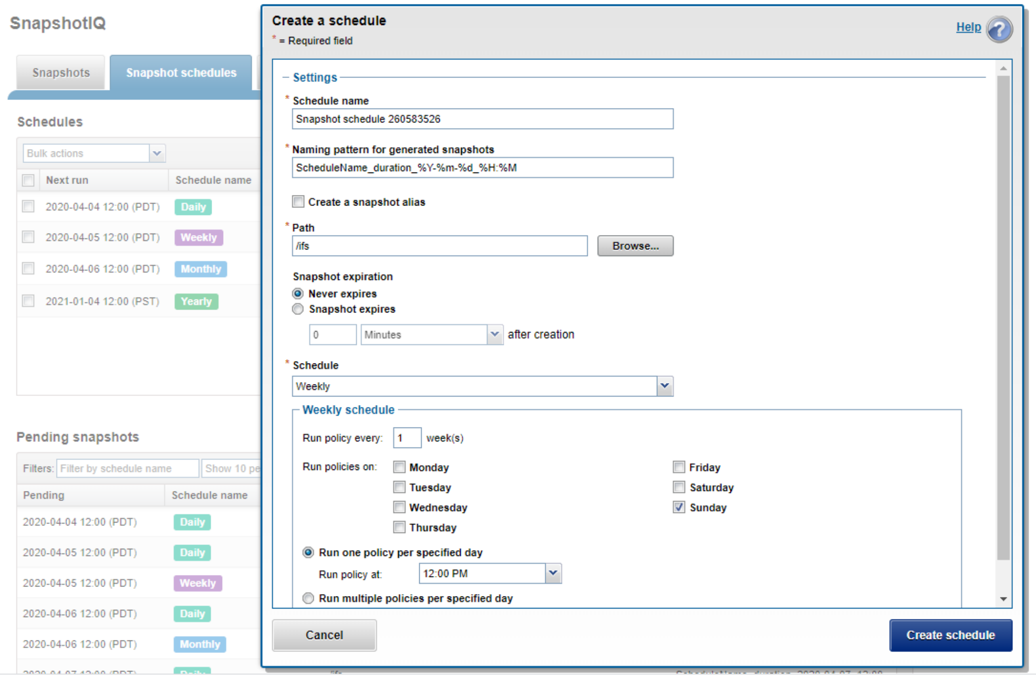
* Path: /ifs/home Browse...

Figure 17. Creating a snapshot alias

Only one alias per snapshot is allowed, which is enforced in the file system. Aliases also have an alias snapshot ID. This is provided as a convenient reference and to make interfaces consistent when referencing snapshots using IDs.

Snapshot scheduling

Snapshot scheduling allows cluster administrators to automatically generate snapshots according to a pre-defined itinerary. OneFS snapshot schedules can be configured at daily, weekly, monthly, or yearly intervals, with single or multiple job frequency per schedule, and down to a per-minute granularity. OneFS snapshot schedules can be configured from either the CLI or the WebUI.



SnapshotIQ

Schedules

Next run	Schedule name
2020-04-04 12:00 (PDT)	Daily
2020-04-05 12:00 (PDT)	Weekly
2020-04-06 12:00 (PDT)	Monthly
2021-01-04 12:00 (PST)	Yearly

Pending snapshots

Pending	Schedule name
2020-04-04 12:00 (PDT)	Daily
2020-04-05 12:00 (PDT)	Daily
2020-04-05 12:00 (PDT)	Weekly
2020-04-06 12:00 (PDT)	Daily
2020-04-06 12:00 (PDT)	Monthly

Create a schedule

* = Required field

Settings

* Schedule name: Snapshot schedule 260583526

* Naming pattern for generated snapshots: ScheduleName_duration_%Y-%m-%d_%H:%M

□ Create a snapshot alias

* Path: /ifs

Snapshot expiration
● Never expires
○ Snapshot expires
0 Minutes after creation

* Schedule: Weekly

Weekly schedule

Run policy every: 1 week(s)

Run policies on: □ Monday □ Tuesday □ Wednesday □ Thursday □ Friday □ Saturday ☑ Sunday

● Run one policy per specified day
Run policy at: 12:00 PM
○ Run multiple policies per specified day

Figure 18. Configuring a weekly snapshot schedule

Similarly, automatic snapshot deletion can be configured per defined schedule at an hourly-through-yearly range.

The following table provides a recommended snapshot schedule for both ordered and unordered deletion configurations:

Deletion Type	Snapshot Frequency	Snapshot Time	Snapshot Expiration	Max Retained Snapshots
Ordered deletion (for mostly static data)	Every four hours	Start at 12:00AM End at 11:59AM	1 month	180
Unordered deletion (for frequently modified data)	Every other hour	Start at 12:00AM End at 11:59AM	1 day	27
	Every day	At 12:00AM	1 week	
	Every week	Saturday at 12:00AM	1 month	
	Every month	First Saturday of the month at 12:00AM	3 months	

Figure 19. Snapshot schedule recommendations

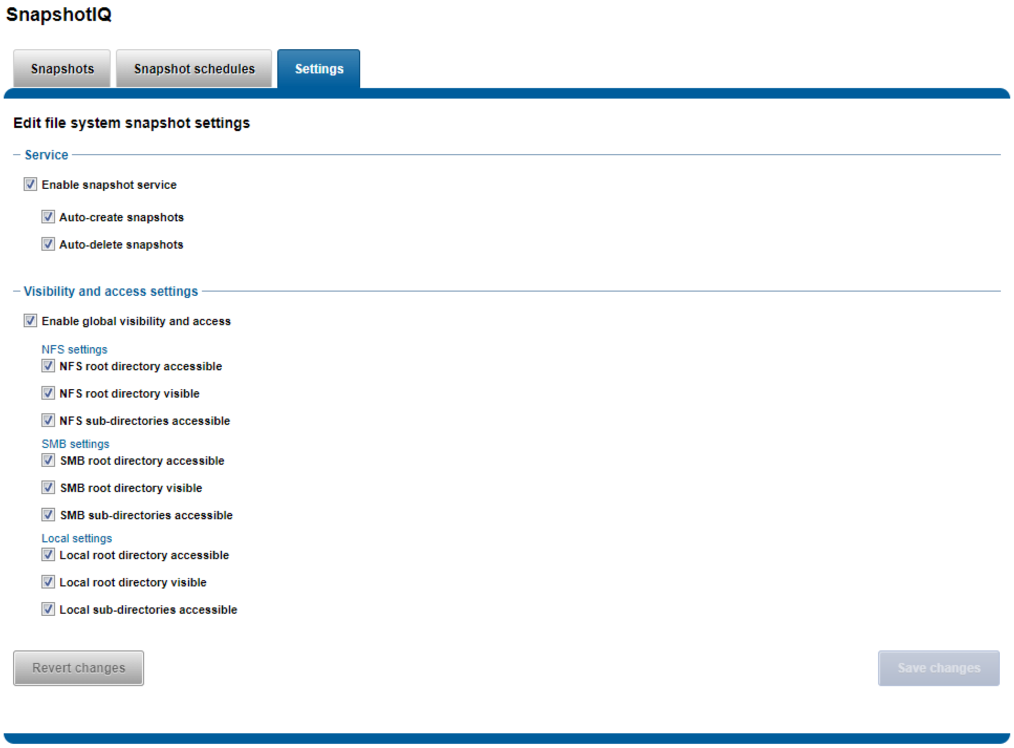
A snapshot schedule cannot span multiple days. For example, you cannot specify to begin generating snapshots at 5:00 PM Monday and end at 5:00 AM Tuesday. To continuously generate snapshots for a period greater than a day, two individual snapshot schedules are required.

In order to generate snapshots from 5:00 PM Monday to 5:00 AM Tuesday, for example, create one schedule that generates snapshots from 5:00 PM to 11:59 PM on Monday, and another schedule that generates snapshots from 12:00 AM to 5:00 AM on Tuesday.

Configuring SnapshotIQ

The SnapshotIQ pages of the OneFS WebUI contains a ‘settings’ tab, as shown below. This provides several global snapshot settings, including:

- The ability to enable and disable the snapshot service
- Control of auto-creation and deletion of expired snapshots
- Per-protocol and complete control of snapshot visibility and accessibility



Edit file system snapshot settings

Service

☑ Enable snapshot service

☑ Auto-create snapshots

☑ Auto-delete snapshots

Visibility and access settings

☑ Enable global visibility and access

NFS settings

☑ NFS root directory accessible

☑ NFS root directory visible

☑ NFS sub-directories accessible

SMB settings

☑ SMB root directory accessible

☑ SMB root directory visible

☑ SMB sub-directories accessible

Local settings

☑ Local root directory accessible

☑ Local root directory visible

☑ Local sub-directories accessible

Figure 20. Global snapshot configuration settings

Snapshot performance

Creating a snapshot in OneFS is a relatively instantaneous process. It typically takes substantially less than a second to perform, depending on the amount of snapshot prep work that has to happen first.

First, OneFS’ write-back cache (coalescer) is paused, and any uncommitted writes are flushed to allow the file system to be quiesced for the short period of time required to create the snapshot. Next, a marker is placed at the top-level directory inode for a particular snapshot and a unique snapshot ID is assigned.

Once this is done, the coalescer resumes and writes continue as normal, and any changes to HEAD (current version) are recorded in the snapshot inodes when any of the logical blocks they reference are altered - until another snapshot is taken.

The moment a snapshot is created, it essentially consumes zero space until file creates, deletes, modifies, and truncates start occurring in the structure underneath the marked top-level directory.

Snapshot locks

As the name suggests, the exclusive snapshot lock which synchronizes the process of creating and deleting snapshots. If a snapshot has one or more locks applied to it, the snapshot cannot be deleted and is referred to as a locked snapshot. If the duration period of a locked snapshot expires, OneFS will not delete the snapshot until all locks on the snapshot have been deleted.

OneFS applies snapshot locks to ensure that snapshots generated by OneFS applications are not deleted prematurely. For this reason, it is recommended that you do not delete snapshot locks or modify the duration period of snapshot locks. A limited number of locks can be applied to a snapshot at a time. If you create snapshot locks, the limit for a snapshot might be reached, and OneFS could be unable to apply a snapshot lock when necessary. For this reason, it is recommended that you do not create snapshot locks.

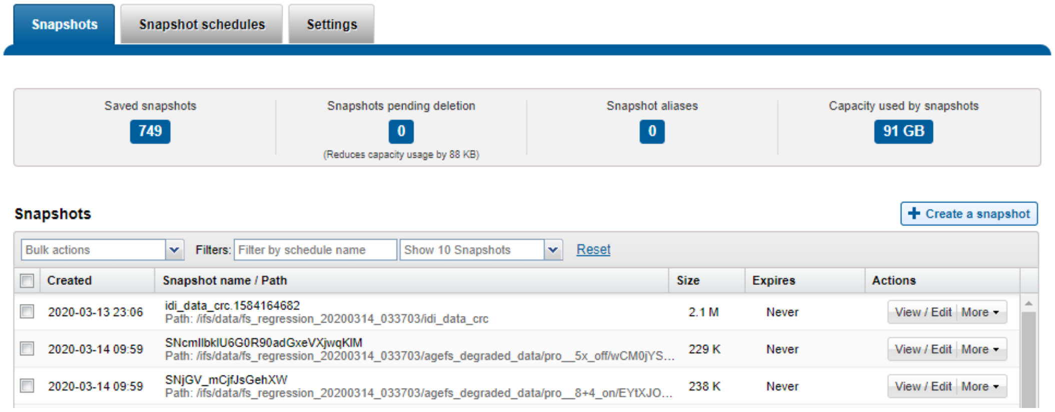
The OneFS CLI command `‘isi snapshot locks delete <snap_ID>’` allows you to remove existing snapshot locks, if necessary.

SnapshotIQ monitoring and reporting

Once a snapshot create request has completed, or has been terminated, a report is available. This can be accessed from the WebUI by going to **Data Protection > Snapshots > Saved Snapshots** and selecting the **View Details** action button on the wanted line item.

OneFS provides a variety of information about snapshots, including the total amount of space consumed by all snapshots. The following information is displayed in the Saved Snapshots area of the WebUI:

- **SnapshotIQ Status** - Indicates whether a SnapshotIQ license has been activated on the cluster.
- **Total Number of Saved Snapshots** - Indicates the total number of snapshots that exist on the cluster.
- **Total Number of Snapshots Pending Deletion** - Indicates the total number of snapshots that were deleted on the cluster since the last snapshot delete job was run. The space consumed by the deleted snapshots is not freed until the snapshot delete job is run again.
- **Total Number of Snapshot Aliases** - Indicates the total number of snapshot aliases that exist on the cluster.
- **Capacity Used by Saved Snapshots** - Indicates the total amount of space consumed by all snapshots.



Saved snapshots 749

Snapshots pending deletion 0 (Reduces capacity usage by 88 KB)

Snapshot aliases 0

Capacity used by snapshots 91 GB

Snapshots

Bulk actions | Filters: Filter by schedule name | Show 10 Snapshots

□	Created	Snapshot name / Path	Size	Expires	Actions
□	2020-03-13 23:06	idi_data_crc.1584164682 Path: /ifs/data/fs_regression_20200314_033703/idi_data_crc	2.1 M	Never	View / Edit More
□	2020-03-14 09:59	SNcmIlbkIU6G0R90adGxeVXjwqKIM Path: /ifs/data/fs_regression_20200314_033703/agefs_degraded_data/pro__5x_off/wCM0jYS…	229 K	Never	View / Edit More
□	2020-03-14 09:59	SNjGV_mCjfJsGehXW Path: /ifs/data/fs_regression_20200314_033703/agefs_degraded_data/pro__8+4_on/EYtXJO…	238 K	Never	View / Edit More

Figure 21. Viewing snapshots information

For SnapRevert and SnapshotDelete jobs, the Job Engine framework provides comprehensive run time and completion reporting for each individual job instance.

While a snapshot related job is underway, its status is available at a glance in the progress column in the active jobs table. Other 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.

SnapshotIQ licensing

SnapshotIQ 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. To create and manage snapshots, you must activate a SnapshotIQ license on the cluster. An unlicensed cluster will show a SnapshotIQ 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 selecting **Cluster Management > Licensing**.

Some applications, such as SyncIQ and InsightIQ, must generate snapshots to function, but do not require an active SnapshotIQ license. By default, these snapshots are automatically deleted when OneFS no longer needs them. However, if you activate a SnapshotIQ license, you can retain these snapshots. Snapshots generated by other modules can still be viewed without a SnapshotIQ license.

Snapshot reserve

There is also no requirement for reserved space for snapshots in OneFS. Snapshots can use as much or little of the available file system space as desirable and necessary.

A snapshot reserve can be configured if preferred, although this will be an accounting reservation rather than a hard limit and is not a recommend best practice. If wanted, snapshot reserve can be set using the OneFS command-line interface (CLI) by running the ‘isi snapshot settings modify –reserve’ command.

For example, the following command will set the snapshot reserve to 20%:

```
# isi snapshot settings modify --reserve 20
```

Snapshot reserve does not limit the amount of space that snapshots can consume on the cluster. Snapshots can consume a greater percentage of storage capacity specified by the snapshot reserve.

Also, when using SmartPools, snapshots can be stored on a different disk tier than the one the original data resides on.

Snapshot overhead

The following guidelines can be used to help calculate snapshot usage overhead:

1. For a given directory, decide how frequently snapshots are taken, and calculate how much the primary data changes within the frequency interval
2. Decide how many snapshots are retained at any given time.

Overhead is the product of multiplying the rate of change (1) and the number of snapshots to retain (2). For example, consider the following scenario:

- Snapshots under /ifs/data/example are taken every day, and the data change rate is 5% daily.
- Three copies of the snapshots are retained at any given moment.

In this case, overhead is 5% x 3 (copies) = 15% for that directory only.

Within OneFS, file system snapshot overhead is very low and, in most cases, marginal. It can be complex to calculate, and changes based on the number of files and file sizes in the given dataset. However, even in the most extreme cases (such as many small files) the file system snapshot overhead is typically only 1-3%. For example, the file system snapshot overhead for a dataset with 1 million files of 1 K size, 5% daily change rate, and three snapshots retained, is 1%.

Sometimes, however, a cluster has many old snapshots that take up a lot of space. Reasons for this may include:

- The cluster is configured to take a large number of snapshots but is not configured to have the snapshots expire as quickly as they are created.
- There is a device that is down or smartfailed on the cluster (in other words, the cluster is in a “degraded protection” state). You cannot run a SnapshotDelete job (or any job other than FlexProtect or FlexProtectLin) while the cluster is in this state. This could cause the number of snapshots to increase without being noticed.
- There is no SnapshotIQ license on the cluster, so you cannot delete old snapshots that still exist.

Writable snapshots

Introduced in OneFS 9.3, writable snapshots enable the creation and management of a space and time efficient, modifiable copy of a regular OneFS snapshot. As such, they present a writable copy of a source snapshot, accessible at a directory path within the /ifs namespace, which can be accessed and edited using any of the cluster’s file and object protocols, including NFS, SMB, and S3.

The writable snapshot architecture provides an overlay to a read-only source snapshot, allowing a cluster administrator to create a lightweight copy of a production dataset using a simple CLI command, and present and use it as a separate writable namespace.

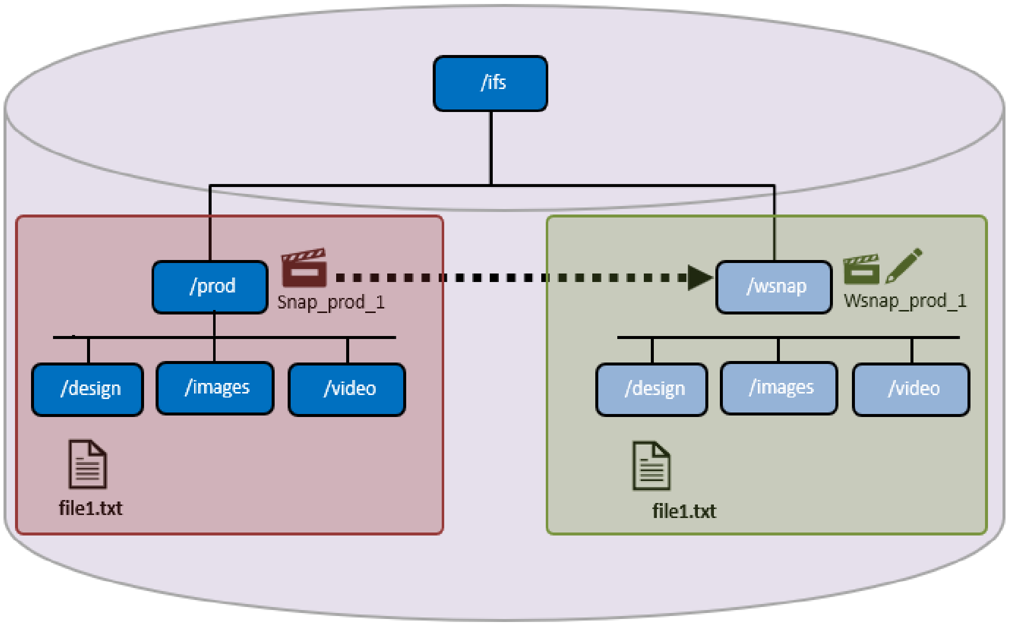


Figure 22. OneFS writable snapshot architecture

In the scenario above, a SnapshotIQ snapshot (snap_prod_1) is taken of the /ifs/prod directory. The read-only 'snap_prod_1' snapshot is then used as the backing for a writable snapshot created at /ifs/wsnap. This writable snapshot contains the same subdirectory and file structure as the original 'prod' directory, just without the added data capacity footprint.

Internally, OneFS 9.3 and later contain a new protection group data structure, 'PG_WSNAP', which provides an overlay that allows unmodified file data to be read directly from the source snapshot, while storing only the changes in the writable snapshot tree.

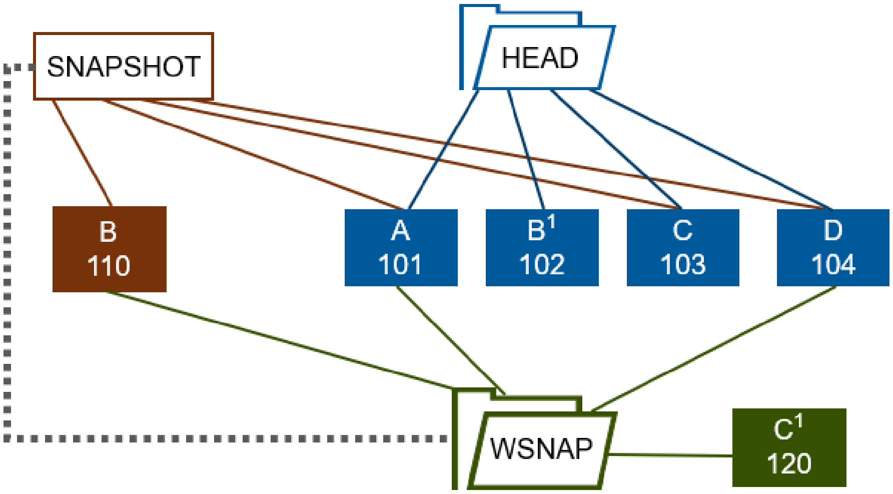


Figure 23. OneFS writable snapshot block overlay and redirection

In this example, a file (Head) consists of four data blocks, A through D. A read-only snapshot is taken of the directory containing the Head file. This file is then modified using a copy-on-write operation. As a result, the new Head data, B^1, is written to block 102, and the original data block 'B' is copied to a new physical block (110). The snapshot pointer now references block 110 and the new location for the original data 'B', so the snapshot has its own copy of that block.

Next, a writable snapshot is created using the read-only snapshot as its source. This writable snapshot is then modified, so its updated version of block C is stored in its own protection group (PG_WSNAP). A client then issues a read request for the writable snapshot version of the file. This read request is directed, using the read-only snapshot, to the Head versions of blocks A and D, the read-only snapshot version for block B and the writable snapshot file's own version of block C (C^2 in block 120).

OneFS directory quotas provide the writable snapshots accounting and reporting infrastructure, allowing users to easily view the space utilization of a writable snapshot. Also, IFS domains are also used to bound and manage writable snapshot membership. In OneFS, a domain defines a set of behaviors for a collection of files under a specified directory tree. More specifically, a protection domain is a marker which prevents a configured subset of files and directories from being deleted or modified. If a directory has a protection domain applied to it, that domain will also affect all of the files and subdirectories under that top-level directory.

When files within a newly created writable snapshot are first accessed, data is read from the source snapshot, populating the files' metadata, in a process known as 'copy-on-read', or CoR. Unmodified data is read from the source snapshot and any changes are stored in the writable snapshot's namespace data structure (PG_WSNAP).

Since a new writable snapshot is not copy-on-read up front, its creation is extremely rapid. As files are subsequently accessed, they are enumerated and begin to consume metadata space.

On accessing a writable snapshot file for the first time, a read is triggered from the source snapshot and the file's data is accessed directly from the read-only snapshot. At this point, the MD5 checksums for both the source file and writable snapshot file are identical. If, for example, the first block of file is overwritten, just that single block is written to the writable snapshot, and the remaining unmodified blocks are still read from the source snapshot. At this point, the source and writable snapshot files are now different, so their MD5 checksums will also differ.

The writable snapshots feature is available after committing an upgrade to OneFS 9.3 or later and can be managed using the OneFS CLI and platform API. To create a writable snapshot you need to specify an existing source, read-only snapshot and a directory path where it will reside within the /IFS namespace. And once a writable snapshot has been created, it can be accessed and modified by the range of OneFS protocols, such as NFS, SMB, or S3.

Writable snapshots have the following compatibility caveats with SnapshotIQ:

- Taking a SnapshotIQ read-only snapshot of a writable snapshot is not permitted.
- Similarly, creating a writable snapshot of an existing writable snapshot is also not supported.

- Writable snapshots cannot be nested in the namespace under other writable snapshots. Such operations will return ENOTSUP.
- Only IFS domains-based snapshots are permitted as the source of a writable snapshot. This means that any snapshots taken on a cluster prior to OneFS 8.2 cannot be used as the source for a writable snapshot.
- Snapshot aliases cannot be used as the source of a writable snapshot ,even if using the alias target ID instead of the alias target name. The full name of the snapshot must be specified.
- The creation of SnapRevert domain is not permitted at or above a writable snapshot. Similarly, the creation of a writable snapshot inside a directory with a SnapRevert domain is not supported. Such operations will return ENOTSUP.
- The SnapshotDelete job has no interaction with writable snapshots. Instead, the TreeDelete job handles writable snapshot deletion.

More information on OneFS writable snapshots is available in the OneFS writable snapshots white paper.

Snapshot best practices

For optimal cluster performance, we recommend observing the following SnapshotIQ best practices. Some information may be covered elsewhere in this paper.

- Configure the cluster to take fewer snapshots, and for the snapshots to expire more quickly, so that less space will be consumed by old snapshots. Take only as many snapshots as you require and keep them active for only as long as you need them.
- Integration with Windows Volume Snapshot Manager allows Windows clients a method to restore from “Previous Versions”
- Snapshots are easily managed using flexible policies and schedules.
- Using SmartPools, snapshots can physically reside on a different disk tier than the original data.
- Recommend limiting snapshot creation to 1,024 per directory.
- The default snapshot limit is 20,000 per cluster.
- Avoid creating snapshots of directories that are already referenced by other snapshots. If you create a snapshot of a root directory, that snapshot counts towards the total number of snapshots for any subdirectories of the root directory. For example, if you create 500 snapshots of /ifs/data and 500 snapshots of /ifs/data/media, you have created 1000 snapshots of /ifs/data/media.
- It is recommended that you do not create more than 1000 hard links per file in a snapshot to avoid performance degradation.
- Always attempt to keep directory paths as shallow as possible. The deeper the depth of directories referenced by snapshots, the greater the performance degradation.
- If a cluster contains many old, unneeded snapshots taking up space, and no valid SnapshotIQ license, contact Dell Technical Support for assistance with deleting the old snapshots.

- Creating snapshots of directories higher on a directory tree will increase the amount of time it takes to modify the data referenced by the snapshot and require more cluster resources to manage the snapshot and the directory. However, creating snapshots of directories lower on directories trees will require more snapshot schedule, which can be difficult to manage.
- It is recommended that you do not create snapshots of /ifs. The main reason not to take a snapshot of /ifs is actually space: because OneFS stores /.ifsvar under /ifs, a snapshot of the /ifs directory will include /ifs/.ifsvar as well. The files within ifsvar are generally highly protected (8x mirrored), and many of them are written quite often. This can waste a lot of space, especially on a small cluster.
- Avoid taking snapshots of /ifs, /ifs/data, and /ifs/home in favor of more specific targets when possible. In particular, avoid taking nested snapshots, redundant snapshots, or overly scoped snapshots. For example, if you schedule snapshots of /ifs/data and /ifs/data/foo and /ifs/data/foo/bar, consider taking snapshots of only the intermediate or most granularly scoped part (/ifs/data/foo or /ifs/data/foo/bar).
- The recommendation is not to disable the snapshot delete job, since this prevents unused disk space from being freed and can also cause performance degradation.
- If you need to delete snapshots and there are down or smartfailed components, or the cluster is in an otherwise degraded state, contact Dell Technical Support for assistance.
- If the system clock is set to a time zone other than Coordinated Universal Time (UTC), SnapshotIQ modifies snapshot duration periods to match Daylight Savings Time (DST). Upon entering DST, snapshot durations are increased by an hour to adhere to DST; when exiting DST, snapshot durations are decreased by an hour to adhere to standard time.
- If you intend on reverting snapshots for a directory, it is recommended that you create SnapRevert domains for those directories while the directories are empty. Creating a domain for a directory that contains less data takes less time.
- Delete snapshots in order, beginning with the oldest. Where possible, avoid deleting snapshots from the middle of a time range. Newer snapshots are mostly pointers to older snapshots and appear larger than they really are. Removing the newer snapshots will not free up much space. Deleting the oldest snapshot ensures you will actually free up the space. You can determine snapshot order (if not by name or date) by using the ‘isi snapshot list -l’ command. The snapshot IDs (first column) are non-conserved, serial values.
- Create several snapshot schedules for a single directory, and then assign different snapshot duration periods for each schedule. Ensure that all snapshots are created simultaneously when possible.
- Do not delete SyncIQ snapshots (snapshots with names that start with SIQ), unless the only remaining snapshots on the cluster are SyncIQ snapshots, and the only way to free up space is to delete those SyncIQ snapshots. Deleting SyncIQ snapshots resets the SyncIQ policy state, which requires a reset of the policy and potentially a full sync or initial diff sync. A full sync or initial diff sync could take many times longer than a regular snapshot-based incremental sync.
- Avoid storing all snapshot data on lowest tier node pools. Where possible, snapshot data should live on the same tier as the production data that it protects. If all

snapshot data gets tiered to the archive pool, it will slow the more performant pool(s) down to the speed of the archive nodes for any write operation that needs to save modified data in a snapshot.

Note: If the oldest snapshot is a SyncIQ snapshot and you do not want to delete it, you can delete the next-oldest snapshot. Even if the next-oldest snapshot is in the same path, you will still free up some amount of data.

Snapshot considerations

As discussed earlier, snapshots are not free. There is always a trade-off between cluster resource consumption (CPU, memory, disk), the potential for data fragmentation, and the benefit of increased data availability, protection, and recovery.

- Snapshots are created at the directory-level instead of the volume-level, thereby providing improved granularity.
- There is no requirement for reserved space for snapshots in OneFS. Snapshots can use as much or little of the available file system space as desirable.
- Quotas can be used to calculate a file and directory count that includes snapshot revisions, provided the quota is configured to include snaps in its accounting using the “--snaps=true” configuration option.
- The Lincount job will also include snapshot revisions of LINs in its count.
- Files with alternate data streams or resource forks are fully supported by SnapshotIQ.
- The SmartDedupe job will automatically ignore (and not deduplicate) file system snapshots.
- SnapshotDelete will only run if the cluster is in a fully available state (no drives or nodes are down).
- Snapshots of file clones, and shadow stores in general, are not allowed, since shadow stores have no hard links.
- Snapshot data will not be containerized by the OneFS Storage Efficiency for Healthcare PACS feature.
- A snapshot schedule cannot span multiple days: To generate snapshots from 5:00 PM Monday to 5:00 AM Tuesday, create one schedule that generates snapshots from 5:00 PM to 11:59 PM on Monday, and another schedule that generates snapshots from 12:00 AM to 5:00 AM on Tuesday.
- If a directory is moved, you cannot revert any snapshots of that directory which were taken prior to its move.
- In OneFS 8.2 and later, the “Archive files with snapshots” configuration setting has been removed from the CloudPools user interface.
- Attempting to take a SnapshotIQ read-only snapshot of a writable snapshot is not permitted and will fail.

Snapshot and OneFS feature integration

SnapshotIQ allows an administrator to create a frozen, point-in-time view of OneFS, while allowing normal file system modifications to continue without interruption. Efforts are made to ensure that snapshots functionality consumes minimal system resources (disk space, CPU) while providing maximum flexibility to the system administrator and users.

As we have seen, snapshots can be used on their own to provide functionality like user-initiated file restoration and staging of exported content, or in conjunction with other OneFS features such as Backup and SyncIQ to enhance the power and flexibility of those applications.

CloudPools and snapshots

CloudPools, the OneFS cloud tiering product, can archive datasets that have associated snapshots. However, archiving snapshotted files to the cloud will not result in space savings on the cluster until all the snapshots taken prior to archiving have either expired or been deleted.

Also, as part of a DR process, the on-cluster stub files after a cloud archiving has completed can be snapshotted and replicated to another cluster.

In OneFS 8.2 and later, CloudPools 2.0 delivers increased snapshot efficiency for files with older, unexpired snapshots.

- Eliminates snapshots data CoW on archive
- Data consumed by snapshots pre-archive remains on-premises.
- Cache invalidation and write-back in snapshots
- Faster recall performance
- Caching is enabled on snapshots and RO/DR file systems
- Fast I/O to stubs in snapshots

Changelist job and snapshots

One of the classes of Job Engine jobs uses a ‘changelist’, rather than a full LIN-based scan, in order to discover its scope of work. The changelist approach analyzes two snapshots to find the LINs which changed (delta) between the snapshots, and from there determines the exact changes.

SyncIQ replication and the File System Analyze (FSAnalyze) cluster analytics are good examples of a job that leverages snapshot deltas and the ChangelistCreate mechanism.

The FilePolicy and FSAnalyze jobs in OneFS 8.2 and later automatically share the same snapshots and index, created and managed by the IndexUpdate job. When a cluster running FSAnalyze is upgraded to OneFS 8.2, the legacy FSAnalyze index and snapshots are removed and replaced by new snapshots the first time that IndexUpdate is run. The new index stores considerably more file and snapshot attributes than the old FSA index. Until the IndexUpdate job effects this change, FSA keeps running on the old index and snapshots.

SmartPools tiering and snapshots

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

To address this, the FilePolicy job, included with OneFS 8.2, and later provides a faster, lower impact method for applying file pool policies than the full-blown SmartPools job. In conjunction with the IndexUpdate job, FilePolicy improves job scan performance by using a ‘file system index’, or changelist, to find files needing policy changes, rather than a full tree scan. This dramatically decreases the amount of locking and metadata scanning work the job is required to perform, reducing impact on CPU and disk - albeit at the expense of not doing everything that SmartPools does. The FilePolicy job enforces just the SmartPools file pool policies, as opposed to the storage pool settings. For example, FilePolicy does not deal with changes to storage pools or storage pool settings, such as:

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

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

NDMP and snapshots

The NDMP Snapshot Management Extension Interface leverages the extensibility of NDMP v4 to define a mechanism and protocol for controlling primary storage file system images commonly referred to as snapshots.

Specifically, this interface supports the management of automated and manual snapshot creation, snapshot deletion, and general directory browsing as well as full snapshot recovery and selective file recovery. This interface provides functionality allowing snapshots to be used to implement near-line data protection solutions that offer faster backup and recovery times compared to traditional tape based secondary storage.

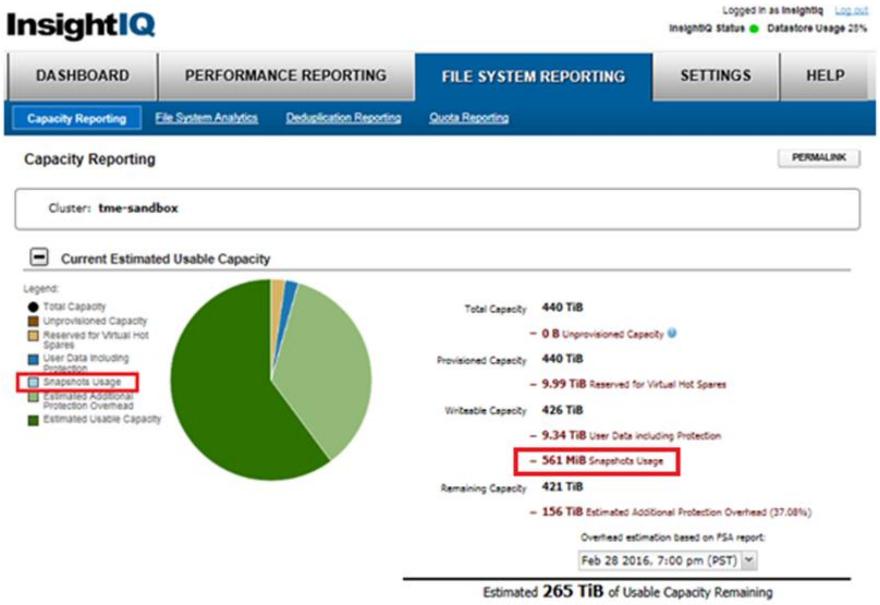
In-line data reduction and snapshots

In-line data reduction, introduced in OneFS 8.1.3 for the F810 platform, will not affect the data stored in a snapshot. However, snapshots can be created of compressed data.

If a compression 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.

InsightIQ and snapshots

The InsightIQ analytics engine provides statistics on the amount of space that snapshots consume on a cluster.



Capacity Reporting

Cluster: **tme-sandbox**

Current Estimated Usable Capacity

Legend:

- ● Total Capacity
- ■ Unprovisioned Capacity
- ■ Reserved for Virtual Hot Spares
- ■ User Data including Protection
- □ Snapshots Usage
- ■ Estimated Additional Protection Overhead
- ■ Estimated Usable Capacity

Total Capacity	**440 TiB**
	− **0 B** Unprovisioned Capacity
Provisioned Capacity	**440 TiB**
	− **9.99 TiB** Reserved for Virtual Hot Spares
Writeable Capacity	**426 TiB**
	− **9.34 TiB** User Data including Protection
	− **561 MiB** Snapshots Usage
Remaining Capacity	**421 TiB**
	− **156 TiB** Estimated Additional Protection Overhead (37.08%)

Overhead estimation based on FSA report:

Feb 28 2016, 7:00 pm (PST)

Estimated **265 TiB** of Usable Capacity Remaining

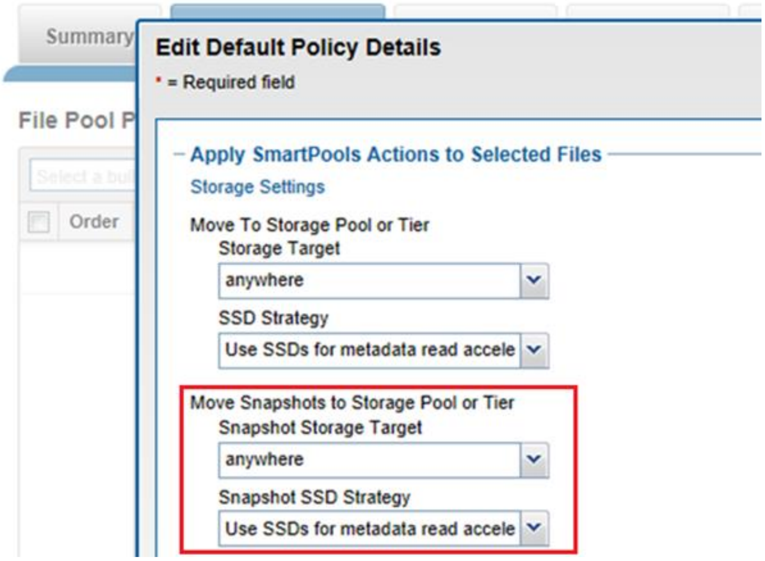
Figure 24. InsightIQ snapshot usage data

SmartPools and snapshots

When using SmartPools, snapshots can be stored on a different disk tier than the one the original data resides on. For example, the snapshots taken on a performance aligned tier can be physically housed on a more cost-effective archive tier.

For example, the snapshots taken on a performance aligned tier can be physically housed on a more cost-effective archive tier.

The snapshot storage target setting is applied to each file version by SmartPools. When a snapshot is taken, the storage pool setting is simply preserved. This means that the snapshot will initially be written to the default data pool and then moved. The SmartPools job subsequently finds the snapshot version and moves it to the intended pool during the next scheduled SmartPools job run.



Edit Default Policy Details

* = Required field

Apply SmartPools Actions to Selected Files

Storage Settings

Move To Storage Pool or Tier

Storage Target: anywhere

SSD Strategy: Use SSDs for metadata read accele

Move Snapshots to Storage Pool or Tier

Snapshot Storage Target: anywhere

Snapshot SSD Strategy: Use SSDs for metadata read accele

Figure 25. Saving snapshots to a different storage tier

SyncIQ replication and snapshots

SyncIQ also leverages snapshots for the consistency points required to facilitate replication, failover, and failback between PowerScale clusters. This means that only the changes between the source and target datasets need to be replicated between the two clusters, allowing for efficient replication and granular recovery objectives. The snapshots generated by SyncIQ can also be used for archival purposes on the target cluster.

Source cluster snapshots

SyncIQ creates snapshots on the source cluster to ensure that a consistent point-in-time image is replicated, and that unaltered data is not sent to the target cluster. Before running a replication job, SyncIQ takes a snapshot of the source directory. It then replicates data according to the snapshot rather than the current state of the cluster, allowing users to modify source-directory files while ensuring that an exact point-in-time image of the source directory is replicated.

For example, if a replication job of /ifs/home/user1/ starts at 10:00 PM and finishes at 10:20 PM, and /ifs/home/user1/file is modified at 10:10 PM, the modifications are not reflected on the target cluster, even if /ifs/home/user1/file is not replicated until 10:15 PM.

SyncIQ can also replicate data according to either an on-demand or scheduled snapshot generated directly by SnapshotIQ. If data is replicated using a SnapshotIQ snapshot, SyncIQ does not generate another snapshot of the source directory. This method can be useful if you want to replicate identical copies of data to multiple PowerScale clusters.

SyncIQ generates source snapshots to ensure that replication jobs do not transfer unmodified data. When a job is created for a replication policy, SyncIQ checks whether it is the first job created for the policy. If not, SyncIQ compares the snapshot generated for the earlier job with the snapshot generated for the new job.

SyncIQ replicates only data that has changed since the last time a snapshot was generated for the replication policy. When a replication job is completed, SyncIQ deletes the previous source-cluster snapshot and retains the most recent snapshot until the next job is run.

Target cluster snapshots

When a replication job is run, SyncIQ generates a snapshot on the target cluster to facilitate failover operations. When the next replication job is created for the replication policy, the job creates a snapshot and deletes the old one.

If a SnapshotIQ license has been activated on the target cluster, you can configure a replication policy to generate additional snapshots that remain on the target cluster even as subsequent replication jobs run.

SyncIQ generates target snapshots to enable failover on the target cluster regardless of whether a SnapshotIQ license has been configured on the target cluster. Failover snapshots are generated when a replication job completes. SyncIQ retains only one failover snapshot per replication policy and deletes the old snapshot after the new snapshot is created.

If a SnapshotIQ license has been activated on the target cluster, you can configure SyncIQ to generate archival snapshots on the target cluster that are not automatically deleted when subsequent replication jobs run. Archival snapshots contain the same data as the snapshots that are generated for failover purposes. However, you can configure how long archival snapshots are retained on the target cluster. You can access archival snapshots the same way that you access other snapshots generated on a cluster.

File clones

In complement to SnapshotIQ’s read-only snapshots, OneFS also provides the ability to create writable clones of files. OneFS File Clones provides a rapid, efficient method for provisioning multiple writable copies of files. Common blocks are shared between the original file and clone, providing space efficiency and offering similar performance and protection levels across both. This mechanism is ideal for the rapid provisioning and protection of virtual machine files and is integrated with VMware's linked cloning and block and file storage APIs. This uses the OneFS shadow store metadata structure, which can reference physical blocks, references to physical blocks, and nested references to physical blocks. This is in contrast to snapshots which, as we have seen, have ditto records which reference other versions.

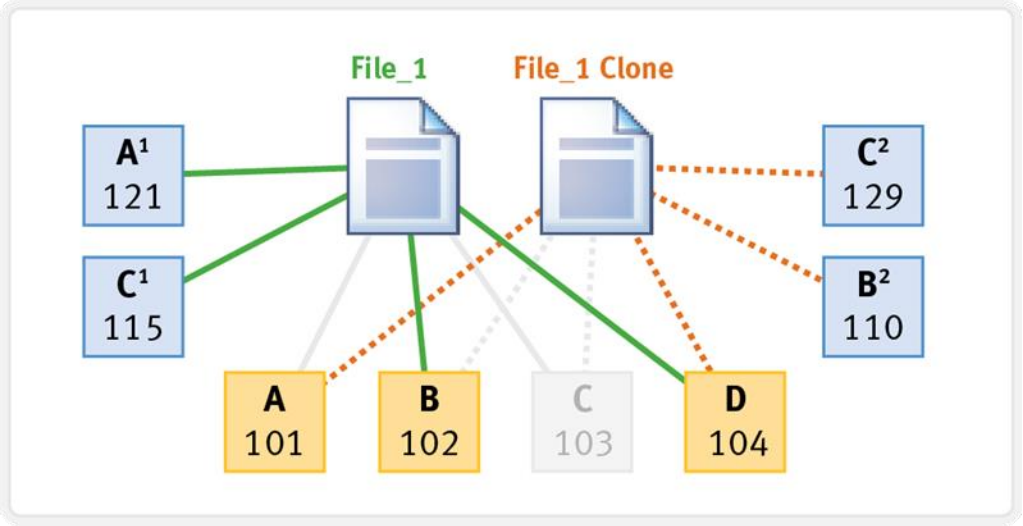


Figure 26. File clones

Shadow stores are hidden files, which behave differently than other files. For example:

- Reading shadow-store references might be slower than reading data directly. Specifically, reading non-cached shadow-store references is slower than reading non-cached data. Reading cached shadow-store references takes no more time than reading cached data.
- When files that reference shadow stores are replicated to another PowerScale cluster or backed up to a Network Data Management Protocol (NDMP) backup device, the shadow stores are not transferred to the target PowerScale cluster or backup device. The files are transferred as if they contained the data that they reference from shadow stores. On the target cluster or backup device, the files consume the same amount of space as if they had not referenced shadow stores.
- When OneFS creates a shadow store, OneFS assigns the shadow store to a storage pool of a file that references the shadow store. If you delete the storage pool that a shadow store resides on, the shadow store is moved to a pool occupied by another file that references the shadow store.
- OneFS does not delete a shadow-store block immediately after the last reference to the block is deleted. Instead, OneFS waits until the ShadowStoreDelete job is run to delete the unreferenced block. If a large number of unreferenced blocks exist on the cluster, OneFS might report a negative deduplication savings until the ShadowStoreDelete job is run.
- Shadow stores are protected at least as much as the most protected file that references it. For example, if one file that references a shadow store resides in a storage pool with n+2 protection and another file that references the shadow store resides in a storage pool with n+3 protection, the shadow store is protected at n+3.
- Quotas account for files that reference shadow stores as if the files contained the data referenced from shadow stores; from the perspective of a quota, shadow-store references do not exist. However, if a quota includes data protection overhead, the quota does not account for the data protection overhead of shadow stores.

Note: Clones cannot contain alternate data streams (ADS). If you clone a file that contains alternate data streams the clone will not contain the alternate data streams.

Conclusion

SnapshotIQ is designed to help you provide highly efficient, cost-effective, low recovery objective data protection. Once a baseline snapshot has been established, only changes to blocks that make up a file are reflected in updates to the current version of snapshots. This allows highly efficient snapshot storage utilization. Also, since snapshots are an integral part of the OneFS file system, there is no need to pre-allocate dedicated snapshot reserve space.

One of the largest IT costs associated with backup and restore is the sheer number of help desk calls from end users who accidentally delete a file or directory. To reduce these costs, SnapshotIQ can be used to empower end users by enabling them to easily find and restore their own accidentally deleted data – without any IT intervention. In this way, business users remain productive and the impact on IT staff can be significantly reduced.

Conclusion

SnapshotIQ transcends the limits of traditional approaches by reliably distributing a highly scalable number or snapshots across multiple, highly available PowerScale storage nodes. The result is a remarkably simple, scalable and efficient data backup and recovery capability that allows you to meet the demanding data availability requirements of your business.