Technical Report

FlexCache in ONTAP

ONTAP 9.8

Chris Hurley, NetApp
December 2020 | TR-4743

Abstract

NetApp® FlexCache® is a caching technology that creates sparse, writable replicas of volumes on the same or different NetApp ONTAP® clusters. It can bring data and files closer to the user for faster throughput with a smaller footprint. This document provides a deeper explanation of how FlexCache technology works and provides best practices, limits, recommendations, and considerations for design and implementation.

TABLE OF CONTENTS

LIST OF BEST PRACTICES

LIST OF FIGURES

Data Is Everywhere

As information becomes more global, data is dispersed worldwide. Both enterprises and smaller companies are creating, storing, and accessing data from all over the world. Data centers are popping up globally, resulting in a need for increased data management to corral all that data.

The recent revolutions in IT storage combined with the movement of many applications to the cloud is creating even more disparate datasets and namespaces. Managing data is becoming a massive challenge.

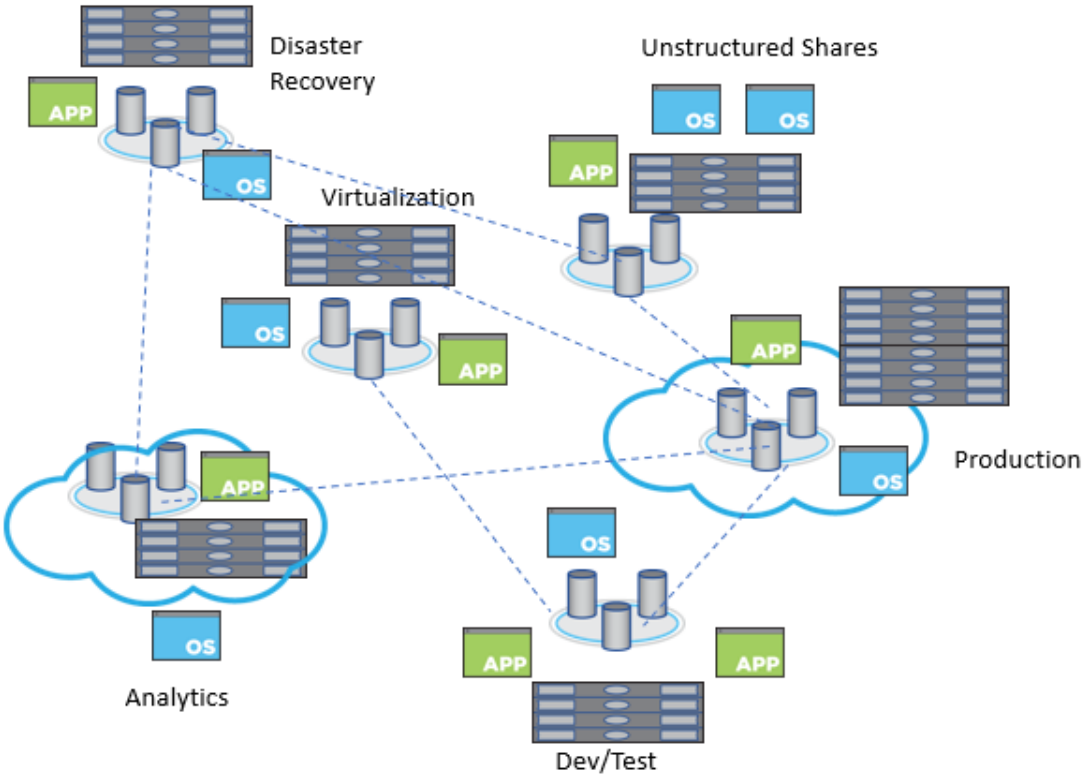
Large Datasets Prove Difficult to Manage

With the advent of artificial intelligence (AI), machine learning (ML), and deep learning (DL), data is typically ingested from many devices and places. The resulting datasets are growing and must be applied in many different places. Not only is the volume of data increasing, but locations where data resides and from where data is accessed are also growing exponentially. In use cases such as media rendering, financial applications, electronic design automation (EDA), and general unstructured file sharing, workflows create data in one location that is accessed or analyzed in another. These workflows can cause problems for data consumers, which can be applications, users, or other resources.

The problem is two-fold. First, when the data sits in multiple places, there is a namespace issue. You cannot access the data through a single namespace without some infrastructure consolidation to create that single namespace. This infrastructure must see the data as one file system to enable the writability, readability, and currency of the data in all places. Second, there is a latency issue. Accessing data from across the globe, from a cloud provider, or from some other physically separated location can be slow and prone to problems.

Figure 1 shows how large datasets can create islands of storage and data silos in which collaboration, efficiency, time, and other key data strategies are not available to your enterprise.

Figure 1) Storage silos.



Data Replication

Many institutions have turned to data replication to solve the problems associated with the data explosion. However, replicating data between two sites becomes costly in a number of ways:

- **Duplication of equipment.** If you have 100TB of data at site A and you want to access it at site B, then you need space to store 100TB of data. This means that the storage platforms must at least be similar so that the data consumers at site B have a similar experience to data consumers at site A.
- **Potentially extra equipment.** Not only do you need to duplicate the infrastructure to handle the replicated data, but you must also deploy new infrastructure to handle the replication configuration and monitor it.
- **Delays, delays and … more delays.** Replication schemas can only move the changed data, but you still incur the cost and delay of moving the data on a scheduled basis. On-demand replication does exist, but, depending on your data structures, it can inadvertently cause more delays due to unnecessary bandwidth usage. Either way, you are paying the price in delays and the possibility of serving stale data as a result. You must also confront questions such as “Are we working with data that is current?” or “When was the last time data was synchronized?” In addition, when replication schemes break, so does everything downstream.
- **Complication.** Because you must manage multiple relationships, the additional effort needed to managing duplicate equipment and extra infrastructure makes data management more complicated.
- **Writability.** Replication might not allow writability to the destination dataset.

Data Synchronization

Data synchronization (sync) can make sure that your destination data is writable. You can configure two-way synchronization, but doing so can create more costs in addition to the ones mentioned in replication.

Keeping data in sync means that replication conflicts can occur. Writes to the same file can happen in site A and site B. Reconciling these replication conflicts is time consuming, costly, and can compromise data.

FlexCache in ONTAP: The Evolution

FlexCache in ONTAP solves these problems by providing a writable, persistent cache of a volume in a remote place that is consistent, coherent and current.

A cache is a temporary storage location that resides between a host and a source of data. The objective of a cache is to store frequently accessed portions of source data in a way that allows the data to be served faster than it would be by fetching the data from the source. Caches are most beneficial in read-intensive environments where data is accessed more than once and is shared by multiple hosts. A cache can serve data faster in one of two ways:

- The cache system is faster than the system with the data source. This can be achieved through faster storage (for example, solid-state drives (SSD) versus HDD), increased processing power, or increased (or faster) memory in the platform that serves the cache.
- The storage space for the cache is physically closer to the host, so it does not take as long to reach the data.

Caches are implemented with different architectures, policies, and semantics so that the integrity of the data is protected as it is stored in the cache and served to the host.

FlexCache offers the following benefits:

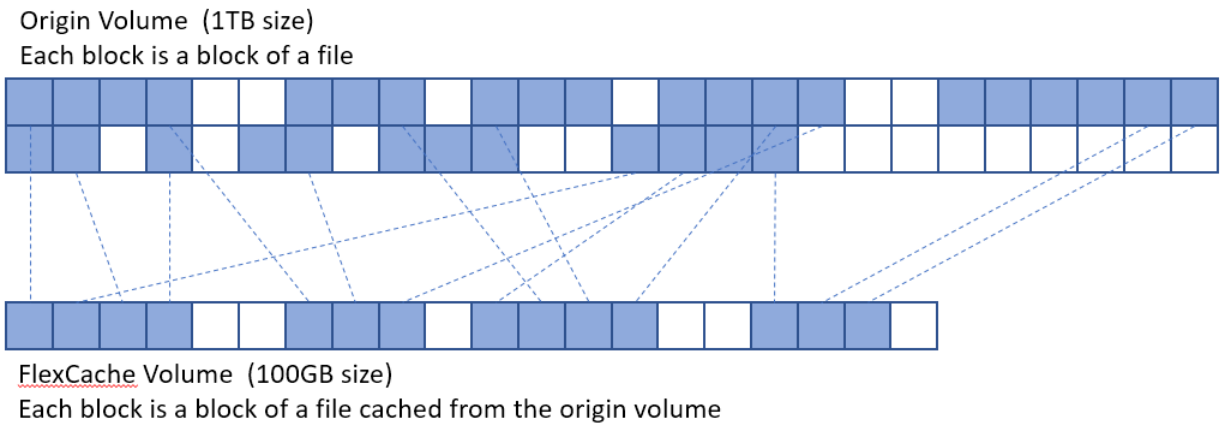
- Improved performance by providing load distribution
- Reduced latency by locating data closer to the point of client access
- Enhanced availability by serving cached data in a network disconnection situation

FlexCache provides all of the above advantages while maintaining cache coherency, data consistency, data currency, and efficient use of storage in a scalable and high-performing manner.

A FlexCache is a sparse container; not all files from the origin dataset are cached, and, even then, not all data blocks of a cached inode can be present in the cache. Storage is used efficiently by prioritizing retention of the working dataset (recently used data).

With FlexCache, the management of disaster recovery and other corporate data strategies only needs to be implemented at the origin. Because data management is only on the source, FlexCache enables better and more efficient use of resources and simpler data management and disaster recovery strategies.

Figure 2) Sparse volume details.



Terminology

Many of the usual ONTAP terms, such as storage virtual machine, logical interface (LIF), NetApp FlexVol® technology, and so on, are covered in TR-3982: NetApp Clustered Data ONTAP 8.3.x and 8.2.x. FlexCache-specific terminology is covered in this section.

- **Origin.** The source volume of the FlexCache relationship.
- **FlexCache volume (or cache volume, or just FlexCache).** The destination volume that is the sparse cache of the origin.
- **FlexGroup volume.** A FlexGroup volume is a single namespace that is made up of multiple constituent member volumes and that is managed and acts like FlexVol volumes to storage administrators. Files in a FlexGroup volume are allocated to individual member volumes and are not striped across volumes or nodes. This is the default volume style for the cache volume.
- **Read-heavy workloads.** Data access is read-heavy when most operations are reads versus writes.
- **Write-back (also called write-behind).** The write operation is applied only to the cache volume on which the operation landed. The write is applied at the origin later based on cache write-back policies.
- **Write-through.** The write operation is applied at both the cache volume on which the operation landed and at the origin before responding to the client.
- **Write-around.** The write operation is applied directly at the origin, bypassing the cache. To see the write at the cache, the cache must pull the information from the origin.
- **Working dataset.** The subset of the total data that is stored at the origin to be cached at the FlexCache. The content of this dataset depends on what the clients mounted to the FlexCache volume request. For most applications (EDA, AI, media rendering), this is a well-defined set of files and directories that are read at the FlexCache.
- **Remote Access Layer (RAL).** The RAL is a feature in the NetApp WAFL® system that enables FlexCache to have a revocable read/write or read-only cache granted on an inode by the origin to a cache. This is the feature that enables FlexCache functionality.
- **Remote Entry Metafile (REM).** A file at the origin that holds delegation information for all the files that are being actively cached in a FlexCache.
- **Remote Index Metafile (RIM).** A file at the cache that holds delegation information for all the files that are being cached at that FlexCache.
- **FCMSID.** FlexGroup MSID of the cache FlexGroup.
- **Fan-out.** The total number of caches that can be attached to a single origin.
- **Disconnected mode.** When the ONTAP cluster hosting the origin cannot communicate with the ONTAP cluster hosting the FlexCache.
- **NLM.** (**N**etwork **L**ock **M**anager) NFSv3's sideband protocol that provides the NFS client the ability to send lock and unlock commands to the NFS server.
- **UNC Path.** (**U**niversal **N**aming **C**onvention) The path provided to map a drive. The first part is the host, the second part is the share and path to map (for example, `\\svm1\share1`).

Comparison to Data ONTAP Operating in 7-Mode

Data ONTAP operating in 7-mode had a FlexCache feature with similar functionality. The new FlexCache in ONTAP is configured to be a replacement for this feature. The two solutions are comparable, but not the same because the technical specifications of the FlexCache feature in 7-mode are different than the technical specifications of the FlexCache in ONTAP feature.

The main difference between the 7-mode feature and the current feature is the protocol that FlexCache uses and how FlexCache communicates with the origin. 7-mode used the NetApp Remote Volume (NRV) protocol running over the data ports. Now, the RAL protocol links the FlexCache to the origin. This is explained in more detail in the technical overview section. In addition, because ONTAP has cluster and storage virtual machine (SVM) peering concepts, the protocol now runs over the intercluster LIFs.

For ease of migration, a FlexCache origin volume on ONTAP 9.5 or later can serve both a cache volume running on a 7-mode system and a cache volume running ONTAP 9.5 or later simultaneously.

Differences between FlexCache and similar ONTAP Features

There are several ONTAP features that provide functionality similar to FlexCache. These features are compared at a high level in this section.

SnapMirror

NetApp SnapMirror® technology provides a read-only copy of a volume at a remote location. This copy can be provided either asynchronously (on a schedule) or synchronously (SnapMirror Synchronous). It can only be a read-only copy unless the relationship is broken. In addition, asynchronous only operates on a schedule, so data currency is delayed. When creating a SnapMirror secondary, the entire dataset is copied over to the new volume, and, when the scheduled sync time passes, all changes are copied from the primary to the secondary. At any point in time during this operation, the SnapMirror secondary volume physically contains 100% of all data that is in the primary. This method requires 100% of the capacity of the data size.

Load Sharing Mirrors

Load sharing (LS) mirrors are a concept in ONTAP in which data volumes can be replicated within the same SVM. This functionality was deprecated in ONTAP 9.1, and FlexCache is considered to be a replacement for this feature. FlexCache is not intended to replace LS mirrors for SVM root volumes; however, this is still an active ONTAP feature. For more information about LS mirrors, see the Data Protection Power Guide or the SVM Root Volume Protection Express Guide.

Data volume LS mirrors provided a read-only copy of a volume on a different node than the original volume, and any changes are automatically replicated to the LS mirror. This operation provides faster access to the data, because the LS mirror volume is on the same node as the LIF. It also insulates against HA-pair failure. This feature does not allow the LS mirror to be external to the SVM.

This link provides a quick configuration of FlexCache as an LS mirror replacement.

Use Cases

The FlexCache in ONTAP design offers the most benefit in specific use cases, and those specific use cases are listed as “ideal.” Other use cases for a FlexCache volume are possible, but the benefits have not been fully vetted. In most instances, the use case is limited to the supported feature set. Non-ideal use cases are not discouraged, but you should compare the benefits of FlexCache to the costs associated with the non-ideal use case.

Ideal Use Cases

Because FlexCache is limited to a write-around model, it works better with workloads that are read heavy. Writes incur latency, and, when there are fewer writes, the latency does not affect the overall performance of the application accessing the dataset. Some examples include, but are not limited to, the following:

- Electronic design automation
- Media rendering
- AI, ML, and DL workloads
- Unstructured NAS data such as home directories
- Software-build environments such as Git

- Common tool distribution
- Hot volume performance balancing
- Cloud bursting, acceleration, and caching
- Stretched NAS volumes across NetApp MetroCluster™ configurations

Supported Features

This section covers the NetApp ONTAP features that are supported for use with FlexCache volumes, whether that feature applies to the origin or the cache. Some features that are not supported at the cache are supported at the origin. With the write-around nature of FlexCache, some features do not need to be supported at the cache. Because the origin coordinates all writes, the feature is supported for the data and container represented by both the cache and the origin.

Table 1) Supported features.

	Supported Feature	Supported on Origin?	Supported on Cache?
NAS Features	CIFS/SMB	Yes	Yes (9.8)
	NFSv3	Yes	Yes
	NFSv4	Yes	No
	NetApp FPolicy™	Yes (9.7)	No[1]
	Vscan	Yes (9.7)	No
	NAS auditing	Yes (9.7)	Yes[2]
	SLAG (Storage Level Access Guard)	No	No
Volume Features	FlexVol volume	Yes	Not applicable
	FlexGroup	Yes (9.7)	Yes
	Qtrees	Yes (9.6)	Not applicable[3]
	Quotas	Yes (9.6)	Not applicable[3]
Data Security	NetApp Volume Encryption	Yes (9.6)	Yes (9.6)
	NetApp Aggregate Encryption	Yes (9.6)	Yes (9.6)
Disaster Recovery	NetApp Snapshot™	Yes	Not applicable
	SnapMirror	Yes[4]	No
	SnapMirror Synchronous	No	No
	SVM Disaster Recovery	Yes[5]	Not applicable
	SnapLock	No	Not applicable
	MetroCluster™	Yes (9.7)	Yes (9.7)
Data Mobility	SVM Migrate	No	No
	Volume Cloning (NetApp FlexClone®)	Yes (9.6)	No
	Vol Move	Yes	Yes (9.6)
	File Clone (SIS Clone)	No	No
	Vol Rehost	No	No
Data Tiering	FabricPool	Yes	Not applicable[6]
Configuration	Nested volume mounts	Yes	No
	System Manager configuration	Yes (9.6)	Yes (9.6)

[1] Some FPolicy solutions might work at the cache. For more information, see your FPolicy solution's vendor.

	Supported Feature	Supported on Origin?	Supported on Cache?

[2] Auditing must be enabled and configured at the cache SVM similar to the origin's SVM. The auditing and logging at the cache results in a file only registering events at that cache volume. There are hints in the log to link those events to the origin volume.

[3] Qtrees and quotas are only configured on the origin. However, they are enforced on both the origin and the cache without any configuration at the cache.

[4] ONTAP 9.8 allows for the origin volume to be a SnapMirror secondary volume. Prior to 9.8, only a SnapMirror primary volume could be a FlexCache origin—a SnapMirror secondary volume could not.

[5] When an origin is part of an SVM disaster recovery relationship, the cutover to the disaster recovery SVM requires recreation of all the caches. The FlexCache relationships are not replicated with the disaster recovery SVM.

[6] A FlexCache volume can reside on an aggregate that is FabricPool enabled, but the FlexCache volume is not tiered.

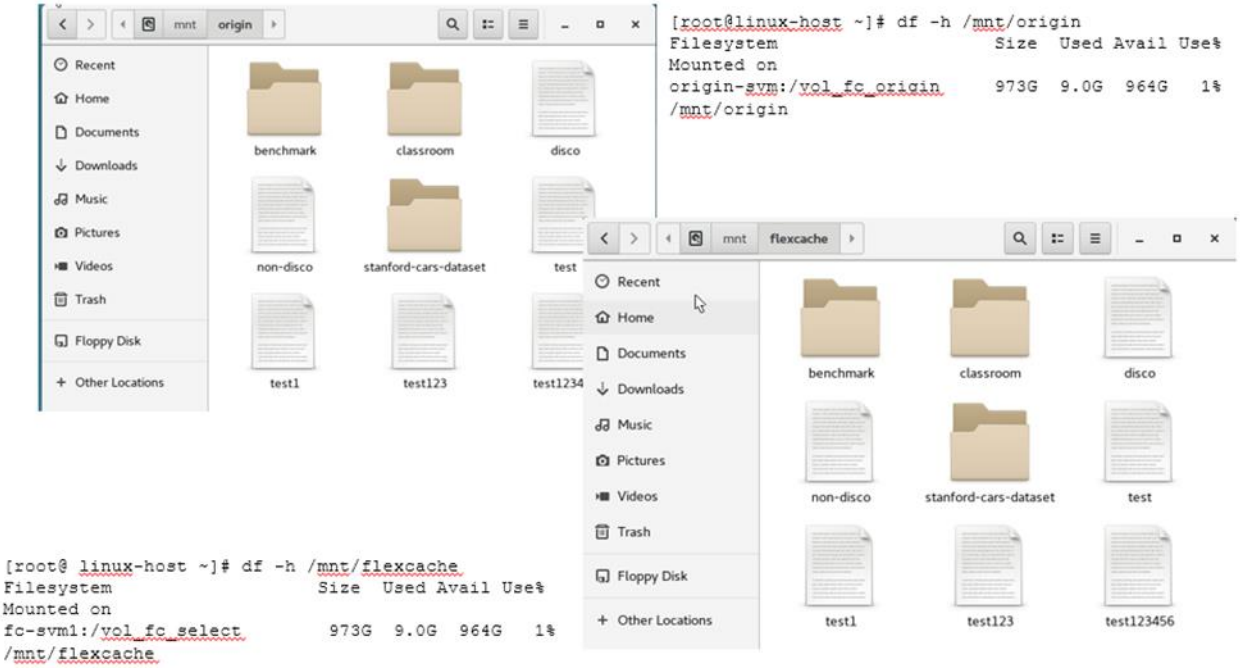
FlexCache in ONTAP Technical Overview

This section provides a technical overview of how FlexCache operates. FlexCache is mainly a "set and forget" feature, so there is not much configuration needed.

Sparse Data Details

FlexCache only caches the data that is read in the client requests. When a FlexCache is created and the client first mounts the FlexCache volume, there is no data taking up space in that volume. However, from the perspective of the client that has mounted the FlexCache volume, the volume looks the same as the origin volume, whether or not anything is cached. In Figure 3, the origin is a 1TB volume, and the cache is only 100GB in size. The volume has been created, so there is no data cached. This figure illustrates that the origin and the FlexCache look exactly alike despite the differences in size and data content in the respective volumes.

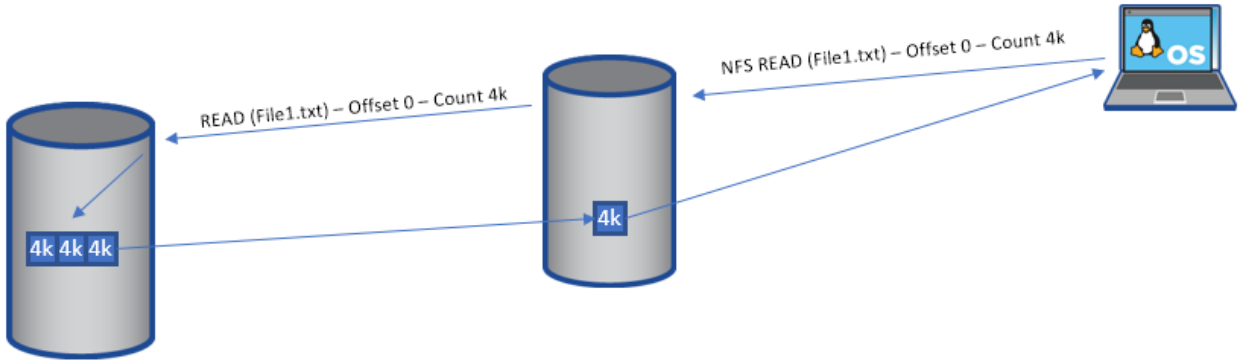
Figure 3) View of origin volume and FlexCache volume.



The cache is populated as the data is read by an attached client. See Figure 2 for details. When a client reads blocks of a file, those blocks are also written to the cache volume while being served to the client attached to the FlexCache. As a result, every cache is unique, and no two caches are the same even though they all point back to the same origin. Different blocks of data are cached at the various caches.

Figure 4 illustrates a client reading 4096 bytes of data from file 1 at cache 1. Since file 1 is a 12k file, only a third of the file is retrieved from the origin and cached in the FlexCache volume. If the client then requests the rest of the file, it is also retrieved from the origin and then cached. Any subsequent reads of any part of the file are then served directly from cache.

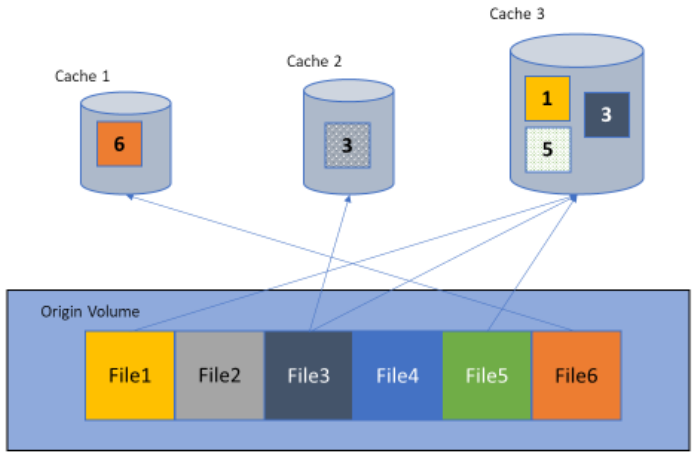
Figure 4) Sparse data details.



Since a FlexCache only has data that is requested by the clients that are mounted to it, each cache's data is different. Figure 5 demonstrates how multiple caches against a single origin can all look different. The origin contains six files. Cache 1 is the smallest cache volume, and it caches the entirety of File 6. Cache 2 is slightly larger than Cache 1, and it caches only a few blocks of File 3. Cache 3 is the largest of the caches and it caches all of File 1, a few blocks of File 5, and all of File 3.

These differences are a direct result of the client requests at the caches. The clients mounted to Cache 1 have only requested File 6 in its entirety, be it through a single request or multiple requests, even across multiple clients. The clients mounted to Cache 2 have only requested a portion of File 3, and the clients mounted to Cache 3 have requested all of File1, a few blocks of File 5, and all of File 3. Again, this could have been performed with any permutation of requests across the clients attached to the cache.

Figure 5) Cache variation.



Export Policies and FlexCache

To mount NFS exports, an export policy and rules associated with the volume are required for access. If the origin volume exists and there are NFS client attached to it, then there is already an export policy associated with the volume for NFS access.

Export policies are not replicated with the creation of a FlexCache. They are independent, even if you are creating a FlexCache in the same SVM. Independent export policies mean that each cache can have different export policies associated with it and can provide different access for different clients. This feature can be used to deny access to certain caches from certain subnets because they should be connected to a different cache or the origin. You could also have a cache with read-only rules so that no clients can write to the cache.

To grant access to the newly created FlexCache, the export policy associated with the FlexCache volume must have the correct rules associated with it. There is no way to replicate the policies in ONTAP between SVMs in different clusters. If the FlexCache is in the same cluster as the origin, but it is in a different SVM, then the `vserver export-policy copy` command can be used to copy the export-policy from one SVM to another. If the FlexCache and origin are in the same SVM, then the same export-policy can be applied to both volumes.

SMB Shares and FlexCache

Beginning in ONTAP 9.8, the SMB protocol is supported at the cache volume so that an SMB share can be created that points to a FlexCache volume. As with export policies, SMB shares are not replicated with the creation of a FlexCache volume. They are also independent, even if you are creating a FlexCache volume in the same SVM. This also means that different share permissions can be implemented at the cache. It allows for granular control of cache data access and can also limit caches to read-only if there is a need.

When using SMB shares to access FlexCache data, most likely the origin volume security style will be NTFS. NTFS ACLs and share permission application is highly dependent on the SMB server's configuration in ONTAP so there are a few requirements to ensure that permissions are applied the same at the origin and at the cache.

The SMB/CIFS server

When creating a FlexCache volume on an SVM separate from the origin, that SVM is a different logical entity. In order to use the SMB protocol at that cache, an SMB server must be created on that SVM. Assuming the origin volume security style is NTFS, there are already NTFS ACL's present within the data in that origin volume. At the lowest level, the permissions are stored in a SID format, and the cache SVM needs to understand how to interpret those permissions it needs to have the SMB server created in the same domain as the origin's SVM. Creating the SMB server is a trusted domain is supported, but it is highly recommended to create the SMB server in the same domain as the origin SVM.

Best Practice 1: Cache and origin SVMs should be in the same domain
To properly enforce ACLs and permissions, the cache and origin SVMs should be in the same domain.

Workgroup Mode

Since the ACLs are in SID format, when an SVM operates in workgroup mode, the domain portion of the SID is different for every SVM, and therefore one SVM cannot interpret the user SIDs of another SVM. For this reason, using workgroup mode at the cache with an NTFS style volume is not supported. Using a UNIX security-style volume is supported because the UIDs and GIDs can be matched between both SVMs.

RAL Overview

The RAL is the function FlexCache uses to enable loading and manipulation of content from any volume inside or outside of the cluster to the origin. RAL is a traffic cop to keep the requests, data, delegations, lock requests, and any other information between the FlexCache and the origin in sync. Sometimes, the FlexCache and the origin are in different clusters. Different clusters require cluster peering for RAL to forward on the I/O operation to the node that owns the origin volume.

When the RAL function is invoked, any operations that are ready to be forwarded to the origin are then redirected over the cluster interconnect LIFs between the nodes. The nodes that initiate and receive the requests are dependent on the originating volume location and origin volume location. For operational efficiency, ONTAP is engineered so that the node that initiates the FlexCache-to-origin communication is the same node that owns the aggregate that the FlexCache is created on. The destination node is the node that owns the aggregate on which the origin volume lies.

FlexCache uses metafiles at both the origin and at the cache to track what has been cached and whether data or lock delegations are valid or not. Data delegations are tracked in the REM and RIM files for the origin and cache respectively. Lock delegations are tracked in the RLEM metafile which is present at both the origin and the caches. These are not meant to be visible to the storage administrator and are not query-able.

Read Processing

When a client issues a read for a file, there are several ways that a FlexCache forks from the standard read process. First, if the file is not found locally, the read request is forwarded to the origin. This process means that the origin is responsible for returning any ENOENTRY (or “File not found”) errors. Second, if the file is found locally, then the local RIM file at the FlexCache must be consulted to make sure that the delegation has not been revoked. If the delegation entry has been removed, then the blocks requested must be re-read from the origin. Following is a visual breakdown of each read scenario.

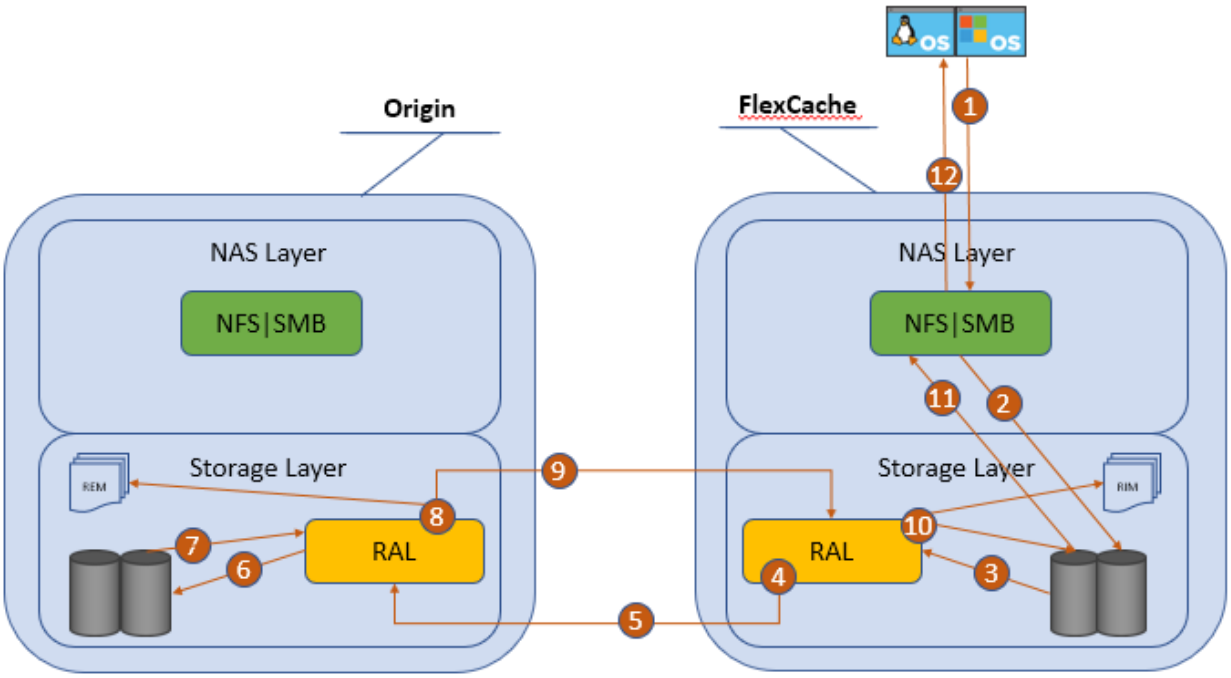
File Not Cached

This is the scenario that is encountered when a FlexCache is first created. Every read does not have a cached inode and must be forwarded on to the origin for data.

Figure 6 shows the steps as follows:

1. A protocol request from the client reaches the NAS layer.
2. The NAS layer then parses the operation and passes the optimized operation to the storage layer.
3. The storage layer then determines that the operation is on a FlexCache (remote) inode. When it tries to load the inode, it discovers that it’s not cached and triggers RAL. This discovery also pauses the storage operation.
4. RAL generates a remote storage operation to retrieve the inode from the origin.
5. The remote retrieval operation is sent over the cluster IC LIF to the origin node.
6. The RAL monitor on the origin receives the request and generates a storage operation for the disk.
7. The inode is retrieved from the disk. (Appropriate file locks are checked)
8. RAL then creates an entry into the REM file for delegation and generates the response to the FlexCache.
9. The response is sent over the cluster IC LIF back to the FlexCache node.
10. RAL receives the response, stores the data on the local disk for the inode, and creates an entry in the RIM file about this inode.
11. The original storage operation is restarted, the data is retrieved, and the response is sent back to the NAS layer.
12. The NAS layer then sends the protocol response back to the client.

Figure 6) Read steps for File Not Cached.

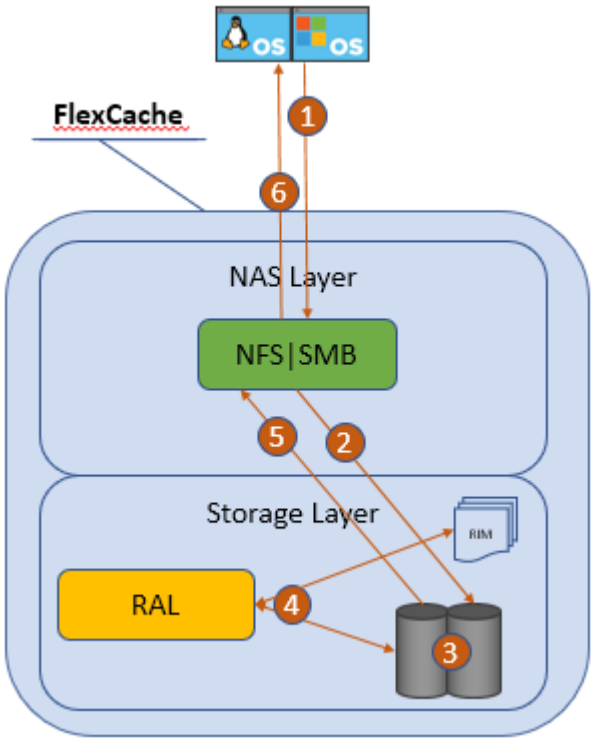


File Cached and Valid

File Cached and Valid is the ideal scenario in which the file is cached in the FlexCache and the delegation is still valid. Notice there is no contact with the origin and therefore no delays in serving the data from the cached file. This scenario works almost the same as a read from a non-FlexCache volume. The steps shown in Figure 7 are as follows:

1. A protocol request from the client reaches the NAS layer.
2. The NAS layer then parses the operation and passes the optimized operation to the storage layer.
3. The storage layer then determines that the operation is on a FlexCache (remote) inode. When it tries to load the inode, it finds it.
4. Because this is a FlexCache inode, RAL kicks in and determines whether the inode still has a delegation entry in the RIM file.
5. Because it found a delegation entry, the data is retrieved, and the response is sent back to the NAS layer.
6. Then NAS layer then sends the protocol response back to the client.

Figure 7) Steps for File Cached and Valid.



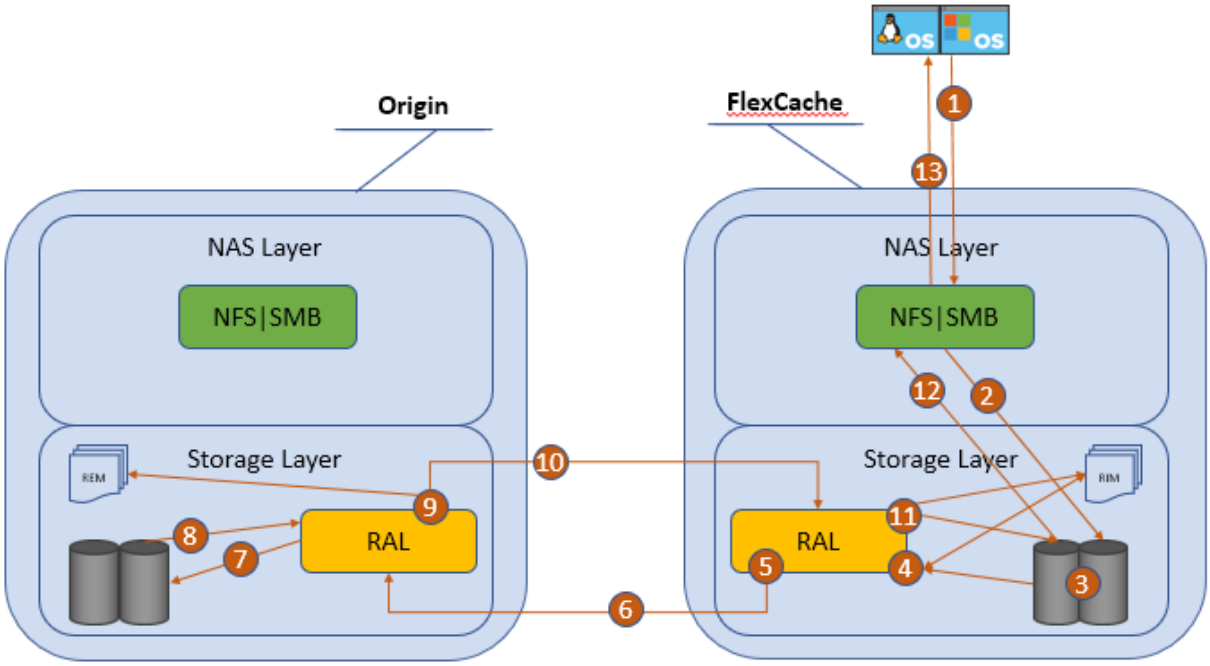
File Cached but Not Valid

File Cached but Not Valid is a scenario in which the file was originally cached, but something changed at the origin causing the cached delegation to become invalid. This scenario means that even though the file is cached, it is not current, and it must be re-fetched from the origin. Prior to 9.8, the invalidation happens only at the file level so any changes at the origin invalidate the whole file. 9.8 adds the option to enable Block Level Invalidation to invalidate cached file data at the block level. Figure 8 outlines the steps in this scenario as follows:

1. A protocol request from the client reaches the NAS layer.
2. The NAS layer then parses the operation and passes the optimized operation to the storage layer.
3. The storage layer then determines the operation is on a FlexCache (remote) inode. When it tries to load the inode, it finds it.
4. Because this is a FlexCache inode, RAL kicks in and determines whether the inode still has a delegation entry in the RIM file.
5. Since the delegation entry is not valid, the storage operation is paused and RAL generates a remote storage operation to retrieve the inode from the origin.
6. The remote retrieval operation is sent over the cluster IC LIF to the origin node.
7. The RAL monitor on the origin receives the request and then generates a storage operation for disk.
8. The inode is retrieved from disk.
9. RAL then updates the entry into the REM file for delegation and generates the response to the FlexCache.
10. The response is sent over the cluster IC LIF back to the FlexCache node.

11. RAL receives the response, stores the data on local disk for the inode, and updates the entry or creates an entry in the RIM file about this inode.
12. The original storage operation is restarted, the data is retrieved, and the response is sent back to the NAS layer.
13. The NAS layer then sends the protocol response back to the client.

Figure 8) Steps for file cached but not valid.



As you can see, there are a few different ways that the read requests can be directed, and the path taken depends on whether the file is cached and valid. There are several scenarios in which the origin can invalidate a cached file. The first is when there is a write to the file. This write can be performed from a cache or directly at the origin. The next scenario is when there is a read of a file at the origin. By default, the `atime-update` property of a volume is set to true, so, if there are many reads at the origin, then there can be many invalidations that cause the FlexCache volume to continually read from the origin. This setting is not optimal. Therefore, NetApp recommends disabling `atime-updates` at the origin volume. When a FlexCache volume is created, `atime-updates` are already disabled, so there is no need to perform this action on the FlexCache volumes.

Best Practice 2: Set `atime-updates` on Origin to False.

To avoid invalidations on files that are cached when there is only a read at the origin, turn off last accessed time updates on the origin volume.

```
origin_cluster::*> volume modify -vserver origin-svm -volume vol_fc_origin -atime-update
false
```

Write-Around Processing

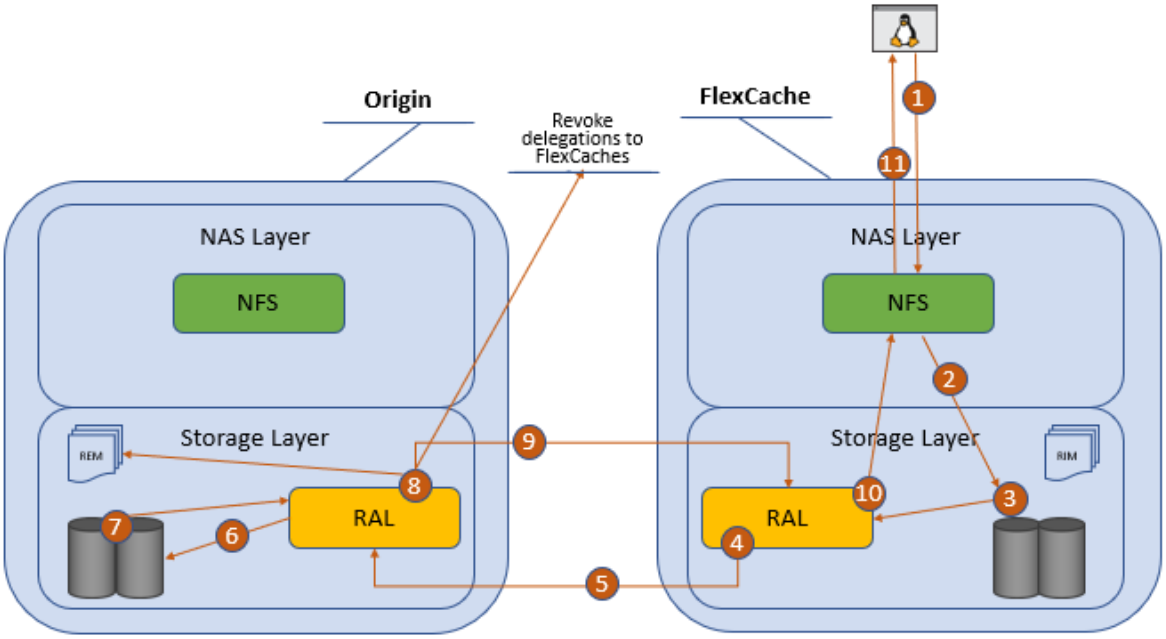
Write around is a simple fork in the writing process. Instead of writing locally to the disk, the optimized write request is forwarded directly to the origin because ONTAP knows this is a FlexCache volume. The

origin then processes the write exactly as if the write request was requested from a client directly attached to the origin. This process allows the origin to coordinate simultaneous writes, locking, and cache invalidations. The write never lands on the cache. Figure 9 illustrates this process. Even if the file being modified is cached at the FlexCache volume, the write is still executed by the origin. Then that cached file is invalidated.

The steps of the write are as follows:

1. A protocol request from the client reaches the NAS layer.
2. The NAS layer then parses the operation and passes the optimized operation to the storage layer.
3. The storage layer then determines that the operation is on a FlexCache (remote) inode. It diverts the write request to RAL.
4. A remote write request is generated by RAL to write the data to the origin.
5. The remote write request is sent over the IC LIF to the origin node.
6. The RAL monitor on the origin receives the request and generates a storage operation for disk.
7. The data is then written to the disk.
8. If it is an existing file, RAL then checks the REM file for any delegations. If the file entry exists and there are valid delegations in the REM file, it contacts each of the FlexCache volumes to revoke the delegation for that file. This invalidation could happen to the FlexCache volume writing the file if it has already cached it.
9. The response is sent over the cluster IC LIF back to the FlexCache node.
10. RAL receives the response and the response is sent back to the NAS layer.
11. The NAS layer then sends the protocol response back to the client.

Figure 9) Write around.



Because there is no write at the cache, any applications that perform read-after-write processing are going to experience performance problems. These are usually applications that have a setting to confirm what was written. This configuration is not suitable for FlexCache and should be tested extensively for adequate performance if FlexCache is part of the infrastructure of that application.

Best Practice 3: Do Not Use Read-After-Write

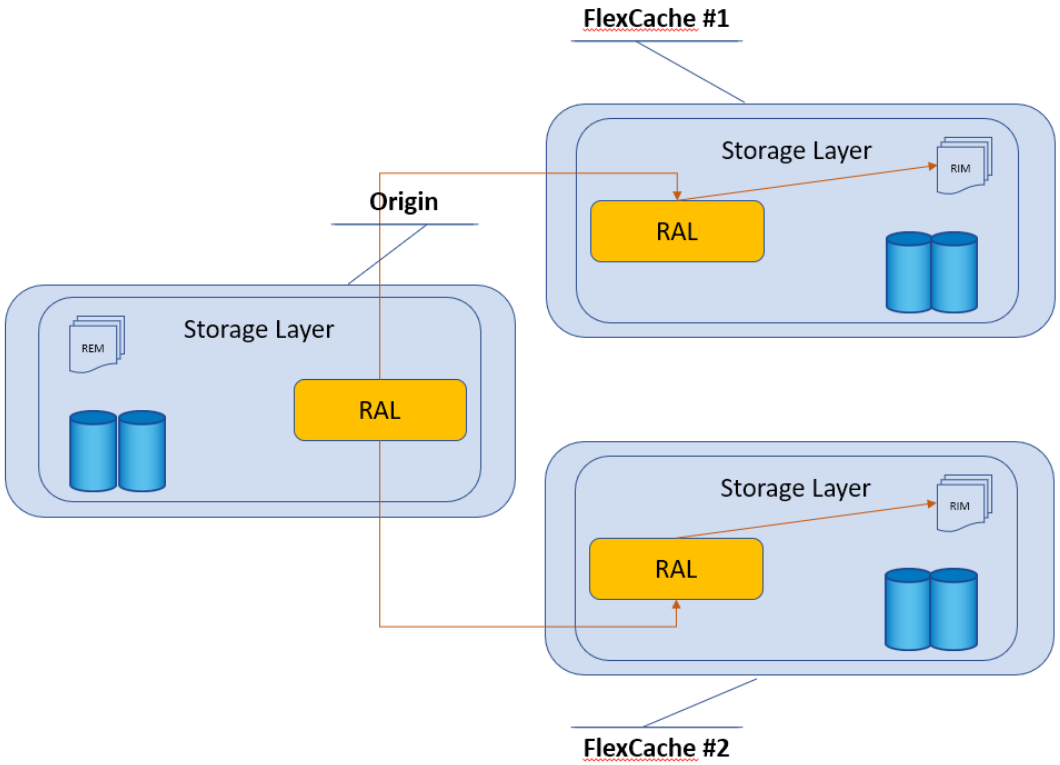
Try not to use applications that confirm writes with a read-after-write. The write-around nature of FlexCache can cause delays for such applications.

Cache Invalidations

The REM file at the origin and the RIM file at the cache keep everything in sync. These files record which files are cached, where the files are cached, and if the delegation entry is valid or has been removed. When a write occurs, the RAL layer at the origin checks the REM file. As mentioned before, the REM file holds the information about whether a file has been cached on a FlexCache and which cache it was cached on. If the file being written has an entry in the REM file, then RAL creates a message to invalidate the inode's delegation entry and sends it to all the caches that have the file cached through the IC LIF. The cache then takes that message and invalidates, or removes, the delegation of that inode in the RIM. The next read of the file at any cache then results in a read to the origin because it has been invalidated.

Figure 10 illustrates the cache invalidation.

Figure 10) Cache invalidation.



Block-Level Invalidation

ONTAP 9.8 introduces block-level invalidation where instead of invalidating the whole file at caches when there are writes, only the modified blocks are invalidated at the cache. This process is beneficial when there are large files and only a few blocks of the file are modified. When read at the cache after the modification, only the changed blocks need to be reread from the origin as opposed to the whole file. The caveat to this is that during writes, each write request ends up in an invalidation request sent to each cache. This can result in come write delays. Make sure that your particular workflow is conducive to using BLI and does not cause unneeded delays for writes. If the write delays become unacceptable with BLI enabled, disabling BLI is the recommended action.

Locking

While data retrieval and invalidation is centrally managed, locking is centrally coordinated. Locking is mostly distributed for performance, but also has some centralized properties. Prior to ONTAP 9.8, NLM was the only locking mechanism supported at the cache and all locks were forwarded to the origin to track centrally. NLM locks are advisory, therefore this coordination was sufficient for the functionality to provide global NLM locking. SMB support at the cache introduces mandatory locking, and the concept of opening a file and requesting locks on a file to cache reads, writes, and file handles. This process allows for the client to be more efficient and to cache operations to send them in bulk back to the server. ONTAP also needs to be efficient in how it handles these locks so that caches can be delegated the ability to grant certain types of locks without having to contact the origin. This concept is managed with the RLEM metafile talked about earlier and is managed very similarly to the way data delegations are handled. This distributed locking architecture in ONTAP for FlexCache still makes it easy to keep all the locks in order and to ensure all locks are accounted for and easily enforced.

Both read and write lock delegations are tracked in the RLEM metafiles. In this distributed paradigm, all read locks are delegated and distributed to the caches. Write locks have a hybrid architecture where NLM write locks are still forwarded and granted only at the origin but SMB write locks are delegated to the caches when they are requested. Details of when and how ONTAP requests locks are described in the following sections.

Read Lock Delegation Caching at First Read

When the first read of a file occurs but none of the file's data has been cached, the FlexCache node also requests appropriate lock delegations for that file when requesting the data, regardless of whether the client is requesting a lock or not. This process optimizes the traffic so that only one operation is sent to the origin and then the lock delegation, along with the data is retrieved. As a result, any subsequent client operation that requests certain types of locks on the file can be granted without the cache needing to contact the origin. The initial lock delegation that is requested is a read, deny none, or handle cache lock. This means that if any other client requests read, deny none or handle cache locks, the cache cluster does not have to contact the origin to grant those types of locks on cached files.

Read Lock Processing for Open Operations

SMB read operations at the cache also requests a file open request before the client can read the data in the file. This open request includes lock states that the client is requesting to streamline operations and cache certain operations. Most of the time, the client only asks for read, deny none, and handle cache locks. Since these are requested at the initial cache operations for the particular file, unless there was an operation that caused the lock delegation to be invalidated, the cache is able to grant those locks without contacting the origin. If the lock delegation has been invalidated, then the cache must contact the origin through the same connection as a data read to request the locks prior to responding to the client's open call. The data delegations and lock delegations are independent of each other so having an invalid data delegation and valid lock delegation, or a valid data delegation and invalid lock delegation are both possible. ONTAP requests the lock delegations from the origin in the exact same manner and transport as the data.

NLM Write Locking

As previously mentioned, NLM locking is coordinated at the origin. Therefore, it is easy to keep all the locks in order and to make sure that all locks are accounted for. When a client requests a file lock through NLM, that request is then transferred to the origin, resulting in a FlexCache-aware NLM lock at the origin. To see these locks, you must look at them from the origin because the FlexCache cluster has no information about NLM locks. Here is an example of a FlexCache NLM lock at the origin.

```
origin_cluster::> vserver locks show
```

```
Notice: Using this command can impact system performance. It is recommended
that you specify both the vserver and the volume when issuing this command to
minimize the scope of the command's operation. To abort the command, press Ctrl-C.

Vserver: origin-svm
Volume  Object Path        LIF     Protocol Lock Type  Client
------- ------------------------ ----------- --------- ----------- ----------
vol_fc_origin
     /vol_fc_origin/test    -      nlm    byte-range 198.51.100.193
        Bytelock Offset(Length): 0 (18446744073709551615)
        FlexCache Lock: true
```

Client `198.51.100.193` owns an NLM byte-range lock on file `/vol_fc_origin/test`. The only difference between the client locking the file directly on the origin and locking it from the FlexCache is the presence of the FlexCache Lock flag.

NLM Lock Recovery

When NLM locks are issued, there is a way to call back to the client to either release a lock or to reclaim locks after a server event through the protocol standard. With NFSv3, this is accomplished through the sideband NSM protocol. Because the lock requests are transported through to the origin, there are no locks held at the FlexCache. You can confirm this by looking at the locks indicated at the FlexCache. ONTAP does not track any locks at the FlexCache. Therefore, lock recovery is not required for any FlexCache event (for example, takeover, giveback, or LIF migrate).

The only time you need lock recovery is when the node that owns the origin volume experiences a takeover or giveback event. When a takeover or giveback event happens at the origin, all lock reclaims for locks that were originated by clients at the origin are processed normally. For locks that have the FlexCache lock flag set, the lock reclaim happens in the same way. However, the lock reclaim message is forwarded to the FlexCache node that originated that lock. Then the FlexCache node sends out the NSM message to the client.

SMB Write Locking

When the client requests a write lock for a file at a cache via SMB, that lock request is forwarded to the origin in the same manner and transport as other read and write data calls described earlier. The origin then checks for any other exclusive write locks on that file and then instead of granting a single client write lock, it grants a write delegation to the cache. This write delegation is tracked via the RLEM file both at the cache and at the origin. After this write lock has been successfully delegated, then that cache can grant byte-range write locks to any client requesting it at that cache. The lifetime of that write delegation is until another client requests a write lock at the origin or another cache or the origin revokes that write lock delegation for another reason. Since the write locks are actually granted and tracked at the cache, they can be viewed at the cache cluster with the `vserver locks show` command as seen below.

```
cache_cluster::> vserver locks show

Notice: Using this command can impact system performance. It is recommended
that you specify both the vserver and the volume when issuing this command to
minimize the scope of the command's operation. To abort the command, press Ctrl-C.

Vserver: cache_svm
Volume   Object Path               LIF         Protocol  Lock Type   Client
-------- ------------------------- ----------- --------- ----------- ----------
cache_of_vol_fc_origin/file.doc cache_svm_lif2 cifs share-level 198.51.100.194
                Sharelock Mode: all-deny_read_write
                                                                      op-lock 198.51.100.194
                Oplock Level: null
                                                                      byte-range
198.51.100.194
                Bytelock Offset(Length): 0 (2147483647)
```

Client `198.51.100.194` owns several locks at the cache. The shared lock denies read and writes. In addition, there is a byte-range lock starting at offset 0. Write lock delegations are exclusively tracked by ONTAP via the RLEM metafile and are not viewable via the `vserver locks show` command on either the origin or the cache.

Write Lock Conflict Checking and Coordination

The origin is still coordinating and orchestrating the write locks. Instead of giving the cache a single write lock for a single client though, it grants a delegation to that cache to grant write locks to any client it chooses to create some efficiencies. As long as that cache has a valid write lock delegation for the file, it can grant write locks to any client at the cache without having to contact the origin. Because there can only be one cache that has this write lock delegation, neither the origin nor any other caches can grant write locks.

When a write lock delegation has been granted to a cache, and there is a client that wants to write a file at the origin, then that origin reaches out to the cache to revoke that write delegation. The semantics of this delegation revocation are not unlike when a client wants to write to a file that already has a lock. If there are no outstanding write locks at the cache currently, then the cache revokes the delegation and responds to the origin as such. The origin can then grant the write lock to the new client. If there are outstanding write locks, then the cache reaches out to the clients holding the locks and asks to recall them. This is the same callback to the client owning the lock that would occur if the clients were attempting to write were accessing it from the same volume. The cache then accepts the answer from the client and take appropriate action. If the client owning the current write lock allows for the downgrade, then the cache clears the write lock, the write lock delegation and then responds to the origin. The origin can then grant the write lock to the new client. If the client denies the write lock downgrade, then the cache responds back to the origin with the denial. The origin then denies the request for a write lock to the new client.

If the new write lock request comes at a different cache than one holding the write lock delegation, the same process above occurs, however the caches communicate with each other through the origin, and not with each other directly. If the write lock delegation revocation is successful, the end result is that ONTAP grants a write lock delegation at the new cache requesting the write lock. After that cache has the write lock delegation, then it can grant write locks to client as expected.

NFSv4 Locking at the Origin

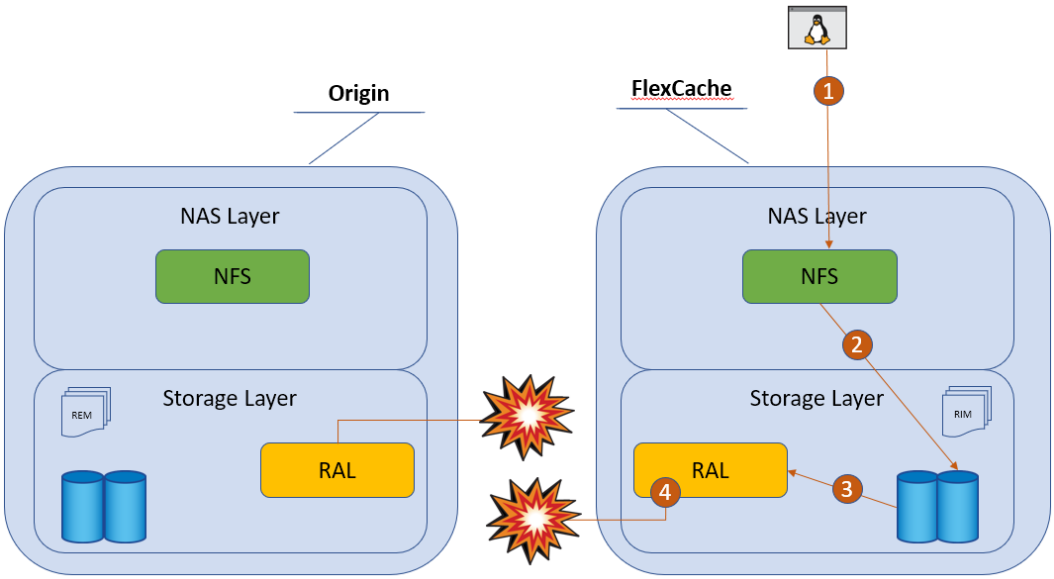
Even though it is not supported at the cache, the NFSv4 protocol family is still supported at the origin. NFSv4 locking is similar to SMB locking, therefore locking operations, when there are FlexCache volumes, occur in a similar fashion as SMB operations at the origin. NFSv4 operations might result in requests to caches to revoke write or read locks at the cache, depending on the operation (like with the SMB operations described above).

Disconnected Mode

Sometimes the cluster that houses the origin and the cluster that houses the FlexCache volume cannot talk to each other over the IC LIFs due to WAN issues or other problems. This situation is called disconnected mode. Both the origin and the FlexCache volume can detect when there is a disconnection because there is bidirectional traffic. Disconnected mode can be in either direction; when any node can’t communicate with the other node in the cluster peer, disconnected mode is the result.

Figure 11 illustrates the FlexCache disconnected mode.

Figure 11) FlexCache disconnected mode.



An event management system (EMS) message is generated at the FlexCache volume when disconnected mode occurs. It is a FlexCache-specific EMS message. There are also EMS messages about the cluster and SVM peer relationships, but the disconnected mode message is specific to the FlexCache volume attempting to contact the origin. See the following example of that EMS message:

```
10/7/2018 19:15:28 fc_cluster-01   EMERGENCY   Nblade.fcVolDisconnected: Attempt to access
FlexCache volume with MSID 2150829978 on Vserver ID 2 failed because FlexCache origin volume with
MSID 2163227163 is not reachable.
```

The origin does not have any specific EMS messages to FlexCache when there are disconnects of a FlexCache, but there are normal EMS messages about cluster and SVM peering relationship problems.

What Can I do While in Disconnected Mode?

There are few restrictions when you are operating in disconnected mode. This section discusses what you can and cannot do while in disconnected mode.

Note: The following information assumes that all best practices outlined in this document have been followed and implemented.

At the Origin

- All reads proceed as normal.
- All writes to new files proceed as normal.
- Writes to existing files that have not yet been cached at a FlexCache proceed as normal.
- Writes to files that have active data delegations at a cache tracked by REM to the disconnected FlexCache are not allowed and hang. ONTAP 9.6 and later allows this write after a set timeout. For more information, see Disconnected Mode TTL and resync.

At the Disconnected FlexCache Volume

- Reads for data that is already cached proceed as normal.

  Note: Some applications also request write locks or other exclusive locks when attempting to open a file. The cache will not be able to grant read locks if there is no read lock delegation, nor will it be able to grant write locks at all. That read at the disconnected cache might fail.

- Reads for data that has not been cached will hang.
- Writes to the FlexCache will hang.
- An `ls` or `dir` command only works if there was an `ls` or equivalent command performed on that directory at that cache before the disconnection.

  Note: A directory is just another inode. If it has been cached, it is served. If it has not been cached, it is not served.

At a Nondisconnected FlexCache Volume

Although there is one disconnected FlexCache volume, other nondisconnected FlexCache volumes operate as normal, with the same restrictions as outlined in the section titled, “At the Origin.”

READDIRPLUS, FlexGroup Volumes, and Disconnected Mode

To speed up `ls` commands to a NetApp ONTAP FlexGroup, a change was implemented in the FlexGroup code that bulk loads directory entries instead of loading them one at a time. This is called `fast-readdir`. Although this accelerated directory listings in FlexGroup volumes, it does not promote caching in FlexCache. When in disconnected mode, `fast-readdir` prevents any `ls` from being run at the disconnected FlexCache. You can revert this behavior to the previous process with a bootarg. NetApp recommends reverting if you need to run `ls` or any `dir-type` operation on the FlexCache while it is disconnected.

Best Practice 4: Change FlexGroup behavior to prevent `ls` hanging in disconnected mode

Set the following bootarg to revert the RAL or FlexGroup behavior to previous so that the “ls” command does not hang in disconnected mode.

```
fc_cluster::> node run <node> “priv set diag; flexgroup set fast-readdir=false persist”
```

Note: A reboot is required for this to take effect.

Disconnected Mode TTL and Resync

Beginning in ONTAP 9.6, FlexCache in ONTAP has a heartbeat mechanism to determine whether the connection between the origin and cache nodes is working. When a cache is disconnected, all writes from the origin or other nondisconnected caches to the files that are still sent to that cache are stalled until the cache reconnects to the origin. The heartbeat mechanism determines the state of the connection and serves the details at the time of revocations by the origin. The connection state is available by the following advanced-mode CLI command.

```
cluster1::*> volume flexcache connection-status show

Node: cluster1-01

              Remote  Remote   Remote      Remote   Connection
+Vserver    Volume   Vserver  Volume   Cluster     Endpoint  Status
+------------ ---------- ---------- ----------  ----------    --------  -----------
 cache-svm1  vol_cache1 origin-svm1 vol_origin1 cluster2    origin   connected

Node: cluster1-02

              Remote  Remote        Remote      Remote   Connection
+Vserver    Volume   Vserver  Volume        Cluster     Endpoint  Status
+------------ ---------- ---------- ----------      ----------    --------  -----------

origin-svm2  vol_origin2 cache-svm2 vol_cache2__0001  cluster2     cache   connected
origin-svm2  vol_origin2 cache-svm2 vol_cache2__0002  cluster2     cache   connected
origin-svm2  vol_origin2 cache-svm2 vol_cache2__0003  cluster2     cache   connected
origin-svm2  vol_origin2 cache-svm2 vol_cache2__0004  cluster2     cache   connected
5 entries were displayed.
```

The code shows that cluster1 has two FlexCache relationships:

1. The first is a cache in which SVM cache-svm1 has a FlexCache linked to origin vol_origin1 in SVM origin-svm1 on cluster2.
2. The second is an origin in which SVM origin-svm2's volume vol_origin2 serves the FlexCache volume vol_cache2 in SVM cache-svm2 on cluster2.

When the cluster is acting as an origin, the FlexCache volumes that are listed might have more than one entry. This duplication occurs because the FlexCache volume is a FlexGroup that has multiple volumes. For more information about FlexGroup volumes, see the NetApp documentation for FlexGroup volumes.

When the connection between the origin and cache is marked as disconnected, changes to files cached at the disconnected cache proceed after the time-to-live (TTL) period is reached. Starting in ONTAP 9.6, the TTL is around 120 seconds. The status in the output of the `volume flexcache connection-status show` command also changes from connected to disconnected.

After the origin and cache nodes re-establish communication, the cache the marks all the files with entries in the RIM file as soft-evicted. Soft-evict verifies that the cache content is marked as invalid and not served until it is certified by the origin that the cache remains safe to serve. Certification is achieved by retrieving metadata associated with the cached content and comparing it to determine any changes while the cache was disconnected. The cache does not initiate any revalidation on its own.

Note: When the connection link between origin and cache is broken, it takes about two minutes for the origin to declare the cache as disconnected. The cache, however, takes about one minute to mark the origin as disconnected. The origin disconnect time is greater than the cache disconnect time to weed out false positives for which the cache hasn't determined the same result.

File Granular Revalidation

ONTAP 9.8 introduces a more efficient way of revalidating the cached files after a disconnection event. This method includes a payload of data from the origin that identifies which files have changed at the origin in the volume. Because of this enhancement, the cache no longer needs to consult with the origin on every file that has been soft evicted after a disconnection event. The payload sent by the origin can be consulted, and many trips back to the origin can be averted. This results in less traffic to the origin and faster file-access times at the cache after a disconnect event.

Last Access Time

By default, last access time (or `atime`) is a file property that is updated whenever there is a read. In essence, a read acts as a write. FlexCache volumes have `atime` tracking turned off during creation to prevent excessive writes back to the origin for this value every time a read at the FlexCache is performed. Because it is not turned off at the origin, `atime` might prevent some access at the origin during disconnection. Following are more reasons to disable `atime` updates at the origin volume:

- Files that have been cached at a FlexCache that is disconnected might not be readable at the origin.
- The `ls` command cannot be run at the origin in certain cases.

In certain cases, disconnected mode might affect access to the origin volume, as outlined in Best Practice 2: Set atime-updates on Origin to False.

MetroCluster Cluster Support

Beginning in ONTAP 9.7, both FlexCache origin volumes and cache volumes can be hosted on an ONTAP MetroCluster (MCC) system. This can be on either mirrored or unmirrored aggregates in the MCC. See TR-4375 NetApp MetroCluster FC for information on mirrored and unmirrored aggregates and their requirements.

If the FlexCache volume is on a mirrored aggregate in the MCC system, then there are additional considerations when using FlexCache. The REM and RIM metafiles needed for FlexCache operations are mirrored from one side to the other with SyncMirror. This can cause some additional delays in FlexCache operations that must write to those files and wait for a SyncMirror operation to complete. This is most common in write operations to a file for which the origin is on a mirrored aggregate and the file has been cached at one or more caches. Since the REM file is modified, it must be mirrored to the other plex before the write can return successfully. Another common operation where this can have an affect includes the first read of a file at a cache hosted on a mirrored aggregate.

Cases in Which Manual Intervention Is Needed After Switchover or Switchback

In certain cases, some manual intervention is needed for FlexCache to function properly after an MCC switchover or switchback. This is because, in certain cases, SVM repeering cannot occur or fails during the switchover operation. The same restrictions apply with SnapMirror. The two scenarios are as follows:

1. The origin and the FlexCache are on different SVMs on the same MCC cluster and site.
2. The origin or FlexCache is on a different cluster, and switchover or switchback occurred while cluster peer traffic was unavailable.

Origin SVM and Cache SVM Are in the Same MCC Site

After switchover, the SVMs at the other site must be peered. This must occur after every switchover. The same restriction applies to similar SnapMirror relationships when the primary volume and secondary volumes are on different SVMs on the same MCC site. Until the peer is established, FlexCache operates in disconnected mode while in switchover. There might also be implications at the origin volume because it detects a disconnected cache.

Origin or FlexCache on Other Cluster and No Cluster Peer Communication during Switchover or Switchback

When there are cluster and SVM peers to a cluster outside of the MCC performing a switchover or switchback operation, ONTAP automatically repeers the SVMs after the event completes. If the MCC cluster cannot communicate with the peer cluster it is attempting to repeer with, the operation fails and FlexCache operates in disconnected mode until the repeer command is issued successfully. The log of the MCC site switch clearly indicates that manual repeering is needed post event. There might also be implications at the origin volume because it detects a disconnected cache.

Creating a Multiprotocol Global File System Namespace with FlexCache

FlexCache can create a true multiprotocol global namespace for your data that spans the globe and is not limited to a single ONTAP cluster. Creating a cache is just the first step of creating a global namespace. In order to create a complete autonomous global namespace, the clients also need to know about the cache and mount or map the data at the cache. This can be achieved manually, or autonomously in several ways.

Duplicating Name Services

The first step to creating a global namespace is to make sure that the name services configurations that are configured at the origin are supplicated at the cache. To properly enforce permissions and auditing, and to guarantee the same access at the cache as at the origin, these settings need to be configured the same. For more information about some of the name services settings, see TR-4668: Name Services Best Practice Guide.

DNS Servers

First, configure the DNS servers at the cache SVM to point to DNS servers that are similar to the servers configured at the origin. The main goal of this step is to ensure that when ONTAP needs to perform name resolution through a DNS server at the cache, the result of the query is the same as if it were performed at the origin. This process ensures that whenever host names are encountered, and ONTAP needs to translate them to an IP address at the cache, the query result will be exactly the same as the origin.

User Name Mapping

The next name service item that must be configured the same is user name mapping. These are the UNIX-to-Windows or Windows-to-UNIX mapping configurations that were configured at the origin. Ensuring that the output of `vserver name-mapping show` matches is sufficient.

```
Vserver:   cache-svm1
Direction: win-unix
Position Hostname         IP Address/Mask
-------- ---------------- ----------------
1       -                 -                   Pattern: (.+)\\testerdwu(.+)
                                          Replacement: win_ldap_user\2

Vserver:   cache-svm1
Direction: unix-win
Position Hostname         IP Address/Mask
-------- ---------------- ----------------
1       -                 -                   Pattern: root
                                          Replacement: CACHE-SVM1\\Administrator
2       -                 -                   Pattern: testerdlu0003
                                          Replacement: DOMAIN\\win_ldap_user0003
```

NS-Switch

The next name service configuration item that should match the origin is the `ns-switch` settings. The `ns-switch` configures which type of name servers ONTAP communicate with when making queries for host names, UNIX users, UNIX groups, and others. This process is as simple as matching the exact output as the origin SVM at the cache.

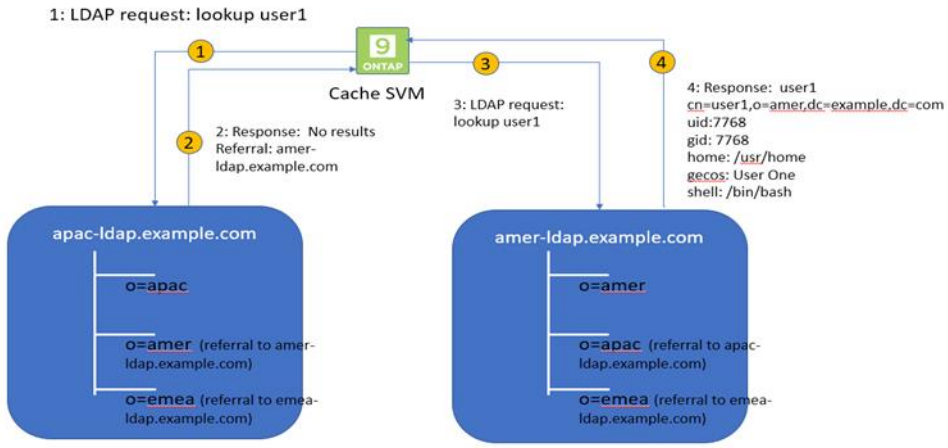
```
cluster1::> vserver services name-service ns-switch show
                                Source
Vserver         Database        Order
--------------- ------------    ---------
cache-svm1      hosts           files,
                                dns
cache-svm1      group           files,
                                ldap
cache-svm1      passwd          files,
                                ldap
cache-svm1      netgroup        files,
                                ldap
cache-svm1      namemap         files
```

LDAP

Lightweight Directory Access Protocol (LDAP) configurations at the cache and at the origin must be similar insofar that an LDAP query at the cache results in the same output as the same LDAP query at the origin. When there are geo-dispersed, multi-realm LDAP configurations, an LDAP server in the same LDAP realm as the origin might not be available in the same location as the cache. This is solved by ensuring LDAP referrals work, or there is a Global Catalog available that covers the realm configured at the origin at the LDAP server that will be configured at the cache. To configure referrals or Global Catalogs, refer to your LDAP vendor documentation. Single realm configurations so not need referrals or global catalogs. An LDAP server in the same location as the cache is sufficient.

Figure 12 shows a common multi-geo LDAP configuration where a referral is needed to perform LDAP lookups. Creating replicas, Global Catalogs and other LDAP configurations can help with the LDAP lookup performance when spanning WAN links. For strategies on improving LDAP lookup performance in a multi-realm configuration, see your LDAP vendor.

Figure 12) Multi-geo LDAP request.



After you have determined which type of LDAP configuration is needed, the ONTAP LDAP client must be configured. Settings of the LDAP client vary depending on the implementation mentioned above. Make sure that the LDAP servers and all the DN settings make sense for your implementation and the results of LDAP queries are the same at the origin and the cache.

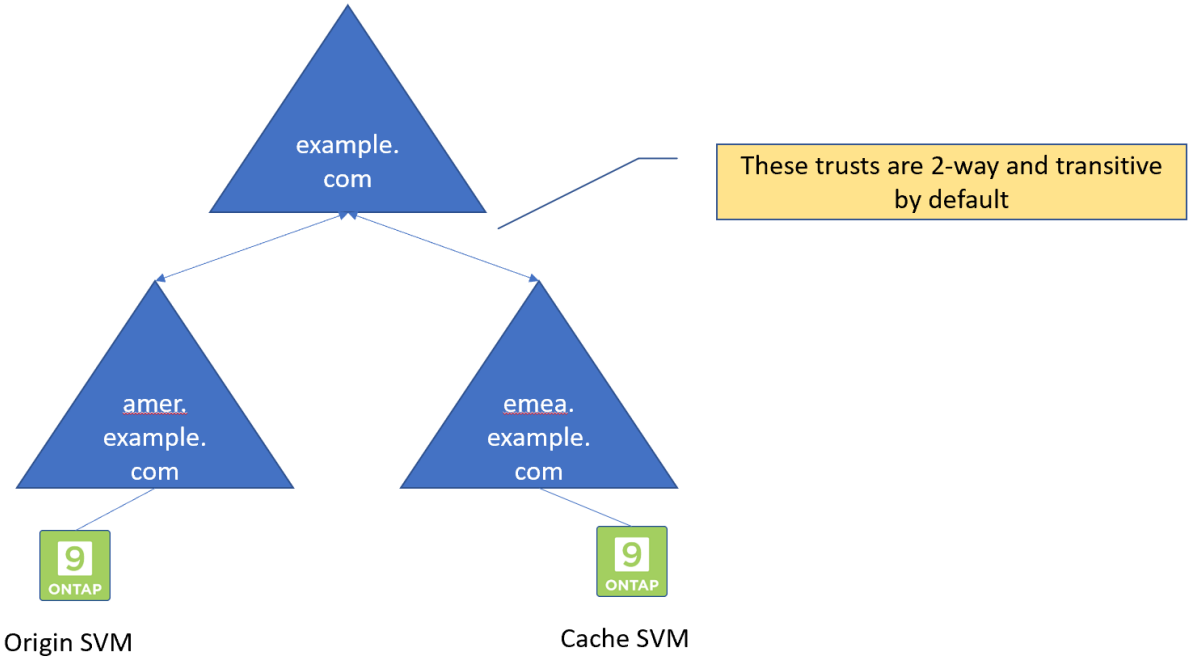
LDAP Schema

Another item to account for at the cache is the LDAP schema that is being used. If there is a custom schema being used at the origin, it is most likely that a custom schema also must be configured at the cache. Make sure that the LDAP schema configured in the FlexCache SVM's LDAP configuration is appropriate.

Active Directory

When using NTFS security style or creating SMB shares, an Active Domain directory must be configured to properly apply the permissions. For this to happen at the cache as at the origin, the cache SMB server must be created either in the same domain as the origin, or a domain that has a two-way trust with the domain the origin is configured to. There are two reasons this is needed. First, NTFS ACLs are actually stored in SID format, not in user name format. This means that ONTAP must look up the SIDs to properly apply permissions. Then ONTAP also needs to properly authenticate clients. When SMB clients and servers are not in the same domain, or in domains that have two-way trusts between them, authentication is a challenge and can result in no access to the data. With globally dispersed clients, multiple domains can be created in order to better serve the client population. The most common type of this configuration is a single forest multiple domain configuration. Figure 14 shows an example of an Active Directory forest with multiple domains where the origin and the cache are in separate domains, but in the same forest with transitive trust relationships.

Figure 13) Origin and cache in separate domains with trust.



Workgroup Mode

ONTAP does allow for the SMB Server to be in a workgroup as opposed to an Active Directory domain to present shares to clients. There are specific use cases where workgroups are preferred for business reasons. FlexCache does work with workgroup mode at the cache when sharing out FlexCache volumes, but only when the security style of the origin volume is NFS. As mentioned earlier, ACLs are stored in a SID format. In workgroup mode, SIDs are specific to the SMB server so that stored NTFS ACLs are not able to be translated to user names because the cache SVM is not the same SMB server as the origin SVM. In addition, in workgroup mode, there is no way to connect SMB servers together to provide those lookups.

Best Practice 5 Workgroup Mode Unix Security Style

If the origin SVM is using workgroup mode, the only volumes that can have FlexCache volumes created on them are volumes where the security style is set to UNIX.

Creating a Global Namespace for SMB Clients

After the necessary configurations described above have been created, then can you set up a means for clients to map drives according to the location of the client and the closest target, whether it's the origin or the cache. SMB and NFS clients operate differently, therefore each requires its own method of configuring the targets. This section explains how to provide SMB clients information in order to map the closest share target to a client when using FlexCache.

Windows DFS Namespace

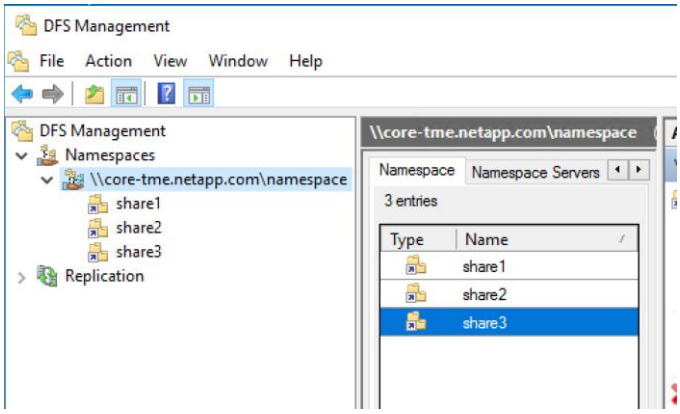
In order to have Windows clients map the share target that is logically closest to the client, you must use the Windows DFS Namespace server. Windows DFS provides the ability to logically group shares on multiple servers and to transparently link shares into a single hierarchical namespace. DFS organizes shared resources on a network in a treelike structure. There are two components to the Windows DFS

feature: the DFS Namespaces server and the DFS Replication. ONTAP only requires the DFS Namespaces feature to create a Global Namespace with FlexCache. The DFS Replication service is only needed for Windows-only DFS structures. For DFS Namespace server best practices or if further information about Windows DFS Namespaces and Replication is needed, refer to the Microsoft documentation.

Creating a Namespace

After the Windows DFS Namespace server has been installed, a namespace must be configured. This is the initial path that all clients will need to start their mapped drive. The first part of the path is mainly to resolve the DFS Namespace server. The Namespace server can either host a domain-based namespace, using the domain name as the first part of the UNC path, or a separate host name can be used to start the Namespace. It is very common for the namespace to be a domain-based namespace. This puts the first part of the UNC path as \\domain.local. You cannot use the SMB server name of any SVM to be the root of the namespace. Figure 14 shows a domain-based namespace. The second part of the path is the root of the namespace. This is an arbitrary name to enter a particular DFS Namespace tree. This does not have to match any share names or any other SMB server settings.

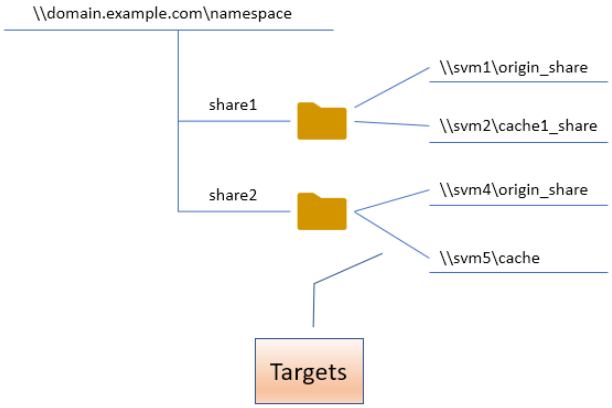
Figure 14) Domain-based DFS namespace.



Configure the Targets

After the Namespace has been created, then begin creating your globally available shares. You determine how many targets and which targets, origin, and or caches you put in the list. The list of targets is the actual shares that the client will access when attempting to map a drive. Figure 15 shows the relationship of the Namespace to the folder tree and targets. It is not required to have all caches or the origin in the list of targets. In addition, you can disable targets at will in the DFS Namespace server without impact to the clients. They will stay connected to the target they first connected to until their session with that SMB server is terminated. Even if the DFS lookup cache expires, the targets are not reevaluated until the current session is terminated and the client attempts to connect to the share again.

Figure 15) Namespace tree relationships



Accessing the Namespace and Shares

When a client accesses a share in a DFS namespace, there are a few things that happen. First, it looks up the DNS entry for the first part of the path. After it acquires an IP address to contact, the client contacts the server. In the case of a domain-based namespace, the DC is contacted. It then refers the client to the Namespace server for the domain-based namespace through a DFS referral. After the Namespace server is contacted, it then checks its targets and refers the client to the closest target using the algorithm configured. There are three different algorithms available: Random Order, Lowest Cost, and Exclude Targets Outside of the Client's Site.

AD Sites and Services

Active Directory sites and services play a crucial role in determining which targets are presented to the client attempting to access the share. Using either the Lowest Cost or the Exclude Targets Outside of the Client's Site requires AD sites and services to be set up with some granularity in order to determine the location of the client in relation to the target SMB server. These relationships are determined by IP address ranges stored in the Active Directory Sites and Services MMC Snap-In. For more information about how to best configure AD sites and services for your infrastructure, refer to the Microsoft documentation.

Creating a Global Namespace for NFS Clients

As previously mentioned, NFS clients require a different method to configure the targets. After the necessary Nameserver configurations have been created, NFS clients can use the automounter service in Linux/UNIX to provide a global namespace. The automounter service provides maps of targets for a mount point to the NFS client. With automounter, the client is responsible for coordinating MOUNT calls and determining which NFS server listed in the map is the closest to the client. The following section provides details about how to use the automounter.

Install Autofs

Most of the standard repositories have `autofs` available as a package. This can be done by using one of the following commands (Linux distribution dependent):

```
yum install autofs
apt-get install autofs
dnf install autofs
```

Create Map Files

With NFS, the only method of informing a client that there are multiple targets for a single export is through a map file. The map file indicates which targets are available for a given path that is to be mounted. In other words, when mounting local path `/mnt/vol1`, there are targets at `svm1:/vol1`, `cache-svm1:/cache_of_vol1`, and `cache-svm2:/cache_of_vol1`.

There are many advanced methods to use the automounter map file and there are many online resources to help if you want to know them in-depth. This section provides a very simple example of a map file that directs a client to multiple NFS server export targets for a single mount. The O'Reilly *Managing NFS and NIS* book is a great resource. There are other O'Reilly books that also talk about automount maps and also how to add them to LDAP for centralized management and distribution.

In the following map file example, we have a mount path `/mnt/data`. Depending on where the client is, there are four possible NFS server exports that are targets to mount:

```
svm1:/data, cache-svm1:/cache_of_data, cache-svm2:/data, svm3:/cache_of_vol1.
/mnt/data -rw,hard  svm1:/data \
            cache-svm1:/cache_of_data \
            cache_svm2:/data \
            svm3:/cache_of_vol1
```

As you can see from the above example, the SVM (NFS server) name or volume junction path naming is not important and does not require any consistency between the origin and caches. The only requirement is that all the entries are either the origin or a cache of that origin.

How the NFS Client Determines the Closest Mount

Now that the `autofs` package has been installed, the automount maps have been created, and the `autofs` service has been started, the only required next step is to actually access the data. Because the mount directory is defined in the map file, the directory is automatically created in the UNIX file system. if it doesn't already exist. In addition, `autofs` makes the mount on demand, where the NFS export is not actually mounted until you change into the mount path directory or access a file within that path.

When the client accesses the mount path, a few things happen. First, the host names in the map file for that mount point are resolved. Next, the client checks to see if any of the targets are in the same subnet of the client. This is done for all IP addresses configured at the client if it is multihomed. If there is a single target in the client's subnet, then that target is used exclusively and the `MOUNT` command is sent to that NFS server. If there are multiple targets in the same subnet as the client, the `MOUNT` command is sent to all targets in the subnet and the first to respond is the NFS server used. If there are no targets in the client's local subnet, or none of the targets respond, then a `MOUNT` command is set to the remaining targets in the list. The first response to the `MOUNT` call received by the client becomes the NFS server used.

Because the client uses the first response of the `MOUNT` command as the NFS server, it is assumed that this server is the fastest and most optimal for this mount path. This is highly dependent on network performance and throughput at the time of the `MOUNT` command.

Note: NFSv3 File Handles will not match between the origin and any of the caches. This means that any movement of the mount point from origin to cache, or from one cache to another, requires a remount to avoid "stale file handle" errors.

Counters, Statistics, and Licensing

There are various counters and statistics that can help in the analysis of FlexCache operational efficiency and performance. These counters help with determining the following:

- How many FlexCache operations are going to the origin

- How long it takes to retrieve the information from the origin
- How long it takes to revoke delegations
- If there were any problems with the communication

The catalog entries are as follows:

```
Object: waflremote
  Counter           Description
  ------------------------- ------------------------------------------------
  cancel_flexcache_ops    Number of times flexcache remote ops were
                canceled due to ARL or HA
  cancel_flexcache_ops_aborted
                Number of times cancelling of flexcache ops
                during HA or ARL was aborted because run time
                was over limit
  fc_buff_hole_size_hist   Buffer size histogram for retreived hole
                buffers for flexcache
  fc_buff_raw_size_hist    Buffer size histogram for retreived raw
                buffers for flexcache
  fc_buff_zero_size_hist   Buffer size histogram for retreived zero
                buffers for flexcache
  fc_bulk_attr_latency    Bulk Attribute Latency
  fc_bulk_attr_ops      Number of times bulk_attr has been performed
                for flexcache
  fc_bulk_attribute_hist   Latency histogram for bulk attribute for
                flexcache
  fc_evict_local_hist     Latency histogram for evicts started by cache
                to itself for flexcache
  fc_evict_local_latency   Evict local latency, when a cache starts an
                evict message to itself
  fc_evict_local_ops     Number of times evict has been sent localy
                (from cache to iteself) for flexcache
  fc_evict_remote_hist    Latency histogram for evict origin-cache for
                flexcache
  fc_evict_remote_latency   Evict remote latency, when an origin sends an
                evict message to a cache
  fc_evict_remote_ops     Number of times evict has been sent from
                origin to cache for flexcache
  fc_retrieve_hist      Latency histogram for remote retrieve for
                flexcache
  fc_retrieve_latency     Remote Retrieve Latency
  fc_retrieve_message_hist  Latency histogram for retrieve at origin for
                flexcache
  fc_retrieve_message_latency retrieve message latency at origin for
                flexcache
  fc_retrieve_message_ops   Number of times retrieve messages that has
                been performed at the origin of a flexcache
  fc_retrieve_ops       Number of times remote_retrieve has been
                performed for flexcache
  fc_store_message_hist    Latency histogram for store_message for
                flexcache
  fc_store_message_latency  store message latency Latency
  fc_store_message_ops    Number of times store message has been
                performed for flexcache
  retrieve_flexcache_full   Number of inbound retrieve messages that
                started revoke because flexcache cache list
                was full
Object: spinhi
  Counter           Description
  ------------------------- ------------------------------------------------
  spinhi_flexcache_forward_base
                Array of select FlexCache spinhi
                op-forwarding file operations count for
                latency calculation
  spinhi_flexcache_forward_csm_errors
                Array of select FlexCache spinhi
                op-forwarding file operations csm error count
  spinhi_flexcache_forward_fileops
                Array of select FlexCache spinhi
```

```
              op-forwarding file operations count
  spinhi_flexcache_forward_latency
              Array of latencies of select FlexCache spinhi
              op-forwarding file operations
  spinhi_flexcache_forward_latency_histogram
              Histogram of FlexCache spinhi op-forwarding
              latency
  spinhi_flexcache_forward_max_time
              Array of select FlexCache spinhi
              op-forwardring file operations representing
              maximum time taken in receiving response from
              the origin
  spinhi_flexcache_forward_sent
              Array of select FlexCache spinhi
              op-forwarding file operations that are
              forwaded
  spinhi_flexcache_forward_total_errors
              Array of select FlexCache spinhi
              op-forwarding file operations error count
  spinhi_flexcache_forward_write_size_histogram
              Histogram of FlexCache spinhi op-forwarding
              write size
```

FlexCache Misses

FlexCache misses are found within the `workload_volume object` and in the `read_io_type` counter. This counter tells you how many times a file was requested that was not cached in a FlexCache volume. See the following example:

```
fc_cluster::> statistics show -sample-id workload_cache -counter read_io_type

Object: workload_volume
Instance: vol_fc_cache1-wid48920
Start-time: 10/2/2018 00:29:02
End-time: 10/2/2018 00:38:31
Elapsed-time: 569s
Scope: fc_cluster
Number of Constituents: 2 (complete_aggregation)
  Counter                           Value
  -------------------------------- --------------------------------
  read_io_type                              -
               cache                 17
               pmem                  0
            ext_cache                   0
               disk                  0
          bamboo_ssd                    0
            hya_hdd                    0
           hya_cache                    0
        hya_non_cache                      0
               cloud                 0
               fc_miss                82
            cloud_s2c                  0

Object: workload_volume
Instance: vol_fc_cache1__0001-wid45107
Start-time: 10/2/2018 00:29:02
End-time: 10/2/2018 00:38:31
Elapsed-time: 569s
Scope: fc_cluster
Number of Constituents: 2 (complete_aggregation)
  Counter                           Value
  -------------------------------- --------------------------------
  read_io_type                              -
               cache                 0
               pmem                  0
            ext_cache                   0
               disk                  0
          bamboo_ssd                    0
            hya_hdd                    0
```

```
              hya_cache                 0
          hya_non_cache                   0
              cloud                 0
              fc_miss                 100
            cloud_s2c                   0
```

FlexCache Delays in Volume Workload Statistics

Statistics for FlexCache delays are also available in the volume workload details. For example:

```
fc_cluster::> statistics show -sample-id workload_detail -instance *FLEXCACHE*

Object: workload_detail_volume
Instance: vol_fc_cache1-wid48920.DELAY_CENTER_FLEXCACHE_RAL
Start-time: 10/2/2018 00:29:22
End-time: 10/2/2018 00:42:43
Elapsed-time: 801s
Scope: fc_cluster-01

  Counter                           Value
  -------------------------------- --------------------------------
  in_latency_path                         1
  process_name                            -
  resource_name             DELAY_CENTER_FLEXCACHE_RAL
  wait_time                         4786us

Object: workload_detail_volume
Instance: vol_fc_cache1-wid48920.DELAY_CENTER_FLEXCACHE_RAL
Start-time: 10/2/2018 00:29:22
End-time: 10/2/2018 00:42:43
Elapsed-time: 801s
Scope: fc_cluster-02

  Counter                           Value
  -------------------------------- --------------------------------
  in_latency_path                         1
  process_name                            -
  resource_name             DELAY_CENTER_FLEXCACHE_RAL

Object: workload_detail_volume
Instance: vol_fc_cache1-wid48920.DELAY_CENTER_FLEXCACHE_SPINHI
Start-time: 10/2/2018 00:29:22
End-time: 10/2/2018 00:42:43
Elapsed-time: 801s
Scope: fc_cluster-01

  Counter                           Value
  -------------------------------- --------------------------------
  in_latency_path                         1
  process_name                            -
  resource_name           DELAY_CENTER_FLEXCACHE_SPINHI
  wait_time                         13138us
```

There is also an aggregated view of average latencies per I/O operation for FlexCache. This view is in the following quality of service (QoS) statistics:

```
fc_cluster::*> qos statistics workload latency show -iterations 100
Workload      ID  Latency  Network  Cluster    Data    Disk    QoS   VRAM   Cloud FlexCache  SM
Sync
--------------- ------ ---------- ---------- ---------- ---------- ---------- ---------- --------
- ---------- ---------- ----------
-total-       -    0ms     0ms     0ms     0ms     0ms     0ms     0ms     0ms     0ms     0ms
-total-       -  543.00us  228.00us    0ms  315.00us    0ms     0ms     0ms     0ms     0ms     0ms
User-Default     2  543.00us  228.00us    0ms  315.00us    0ms     0ms     0ms     0ms     0ms     0ms
-total-       -    0ms     0ms     0ms     0ms     0ms     0ms     0ms     0ms     0ms     0ms
-total-       -  389.00us  213.00us    0ms  176.00us    0ms     0ms     0ms     0ms     0ms     0ms
User-Default     2  389.00us  213.00us    0ms  176.00us    0ms     0ms     0ms     0ms     0ms     0ms
```

```
-total-       -  394.00us  211.00us    0ms  183.00us    0ms    0ms    0ms    0ms    0ms    0ms
User-Default     2  394.00us  211.00us    0ms  183.00us    0ms    0ms    0ms    0ms    0ms    0ms
-total-       - 1160.00us  236.00us    0ms  711.00us    0ms    0ms    0ms    0ms  213.00us    0ms
User-Default     2 1160.00us  236.00us    0ms  711.00us    0ms    0ms    0ms    0ms  213.00us
0ms
-total-       -    0ms    0ms    0ms    0ms    0ms    0ms    0ms    0ms    0ms    0ms
-total-       -    0ms    0ms    0ms    0ms    0ms    0ms    0ms    0ms    0ms    0ms
-total-       -   9.02ms  630.00us  234.00us   4.35ms    0ms    0ms  30.00us    0ms   3.77ms
0ms
User-Default     2   9.02ms  630.00us  234.00us   4.35ms    0ms    0ms  30.00us    0ms   3.77ms
0ms
-total-       -   5.09ms  318.00us  488.00us    0ms    0ms    0ms    0ms    0ms   4.29ms    0ms
User-Default     2   5.09ms  318.00us  488.00us    0ms    0ms    0ms    0ms    0ms   4.29ms
0ms
```

Licensing

Licensing FlexCache is no longer required in ONTAP 9.7. The functionality is included in ONTAP and does not need activated. Prior to 9.7, FlexCache licenses were included in the price of ONTAP, but the license file was still required for FlexCache to function properly. The licenses are both capacity and subscription based. ONTAP alerts you through EMS when the total FlexCache size in the cluster reaches 80% of the licensed capacity. The following sections have been retained in this document to explain licensing. The master license that is distributed for ONTAP 9.5 and 9.6 to function properly has a 400TB capacity and is valid through 2099.

What is the Capacity Used Based On?

ONTAP determines the used capacity based on the current FlexCache volume sizes of the cluster. This measurement does not consider how much data is actually being used nor does it take the origin volume size or the data amount into account. If there are two FlexCache volumes in the cluster currently set to 1TB and 500GB, then you are using 1.5TB against the license. This number also does not take into account autogrow limits. If each of those volumes were allowed to grow to 2TB, the additional 2.5TB of capacity potential is not counted against the license until autogrow actually increases the volume size.

When do My Licenses Expire?

You can see the license expiration date by running the following CLI command:

```
fc_cluster::> license show
 (system license show)

Serial Number: 000000009
Owner: cluster
Package      Type   Description      Expiration
---------------- -------- -------------------- -------------------
FlexCache     capacity FlexCache License   11/6/2019 17:38:15

Serial Number: 1-80-000011
Owner: cluster
Package      Type   Description      Expiration
---------------- -------- -------------------- -------------------
Base       site   Cluster Base License -
NFS        site   NFS License      -
CIFS       site   CIFS License     -
iSCSI       site   iSCSI License     -
FCP        site   FCP License      -
SnapRestore    site   SnapRestore License  -
SnapMirror     site   SnapMirror License  -
FlexClone      site   FlexClone License   -
VE        site   Volume Encryption License
                      -
10 entries were displayed.
```

How do I Find Out How Much is Being Used?

There is a CLI command to determine current license usage. The license allows a 10% overage to avoid disrupting operations, but it is imperative to buy another increment of licensed capacity. The command is as follows:

```
fc_cluster::> system license show-status
Status  License       Scope   Detailed Status
--------- ------------------- --------- ----------------------
valid
     NFS         site   -
     CIFS         site   -
     iSCSI        site   -
     FCP         site   -
     SnapRestore     site   -
     SnapMirror      site   -
     FlexClone      site   -
     VE          site   -
     FlexCache       cluster  The system is using 1.56TB. The system can use up to 10TB. License
expires as of: 11/6/2019 17:38:15
not-installed
     SnapVault      -     -
     SnapLock       -     -
     SnapManagerSuite   -     -
     SnapProtectApps   -     -
     V_StorageAttach   -     -
     Insight_Balance   -     -
     OCShift       -     -
     TPM         -     -
     DP_Optimized     -     -
     FabricPool      -     -
     SnapMirror_Sync   -     -
     NVMe_oF       -     -
not-applicable
     Cloud        -     -
     Select        -     -
     CapacityPool     -     -
24 entries were displayed.
```

Where can I get a license?

If you need a license for ONTAP 9.5 or 9.6 and have a valid netapp.com account with support, then you can download the master license here.

Performance

For information about performance while serving already cached files at a FlexCache, see TR-4571 NetApp FlexGroup Volume Best Practices and Implementation Guide. Because the FlexCache volume is a FlexGroup, there is a negligible difference in the way the already cached files are served at the FlexCache versus how files are served from a non-FlexCache FlexGroup. Any performance information for FlexCache is identical to information for a FlexGroup.

When the FlexCache is across a latent WAN, performance is more sensitive during the first read of a file. This sensitivity can be seen in the statistics mentioned previously:

```
statistics show -object waflremote -counter fc_retrieve_hist -raw.
```

This counter shows a histogram of the extra latency that FlexCache encountered by reading from the origin. The latency can be reduced in a few different ways.

First you can reduce the actual network latency between the FlexCache cluster and the origin cluster. WAN optimizers might help, but, because it is not a public protocol, WAN optimizers might not always be able to accelerate the traffic and results vary.

Second, you can preload the data. There are several ways to preload the data depending on the use case.

The following section describes some of the possibilities.

Prewarming the cache

ONTAP 9.8 introduces an on-box method of pre-warming the cache with the data. The command is `flexcache prepopulate start`. This command takes a path variable so that the pre-population can be done at a volume, qtree, directory or single file level. This does not ensure that the files cached with this command are not evicted (pinning). Files prepopulated are still subject to eviction based on the standard eviction rules

Pre-warming can also be done at the client. For Linux/Unix clients, there is a bash command that preloads all the files in a certain directory. You can run this command with either a dot (.) in the `<dir>` to run it from the current directory, or you can give it a specific directory. The command is as follows:

```
[linux-host ~]# find <dir> -type f -print -exec sh -c "cat {} > /dev/null" \;
```

Usually, the command also warms the directory listings. If it does not, you can also run `ls -R <dir>` and replace the `<dir>` with the same information as above.

Windows clients can also use a command to pre-warm all the files in a certain directory. You can run this command with either a dot (.) in the `<dir>` to run it from the current directory, or you can give it a specific directory. The Windows command that will warm the cache is as follows:

```
C:\>for /f %i in ('dir /b /s <dir>') do @echo %i >NUL
```

As in Linux, usually, the command also warms the directory listings. If it does not, you can also run `dir /s <dir>` and replace the `<dir>` with the same information as above.

Best Practices

This section discusses the best practices associated with FlexCache volumes.

FlexGroup Constituents

A FlexCache volume is a FlexGroup volume. This means that the total space of the FlexCache volume is made up of smaller constituent volumes. For more information about FlexGroup volumes, see TR-4557: NetApp FlexGroup Volumes: A Technical Overview. FlexCache has two options for creating a FlexGroup:

- List aggregates for constituents (`-aggr-list`)
- Let FlexGroup autoselect (`-auto-provision-as`)

NetApp recommends using the `-aggr-list` option to specify the aggregates to create the constituent volumes in. The option `-aggr-list-multiplier` determines how many constituent volumes are being used per aggregate listed in the `-aggr-list` option. The recommendation on the total number of constituents is proportional to the size of the FlexCache volume:

- FlexCache volume < 100GB = 1-member volume
- FlexCache volume > 100GB < 1TB = 2 member volumes
- FlexCache volume > 1TB < 10TB = 4 member volumes
- FlexCache volume > 10TB < 20TB = 8 member volumes
- FlexCache volume > 20TB = the default number of member volumes (use `-auto-provision-as flexgroup`)

Best Practice 6: FlexCache Volumes Should Have Constituent Numbers Based on Size

Create the FlexCache with only the `-aggr-list` option so it creates the prescribed number of constituents.

```
fc_cluster::> flexcache create -vserver fc-svm1 -volume vol_fc_cache -origin-vserver origin-svm -origin-volume vol_fc_origin -aggr-list aggr1_node1 -size 5TB -aggr-list-multiplier 4 -junction_path /vol_fc_cache1
```

Reads Versus Writes

The rule of thumb for FlexCache is a read/write mix of at least 80% reads and 20% writes at the cache. This ratio works because of the write-around nature of FlexCache. Writes incur a latency penalty when forwarding the write operation to the origin. FlexCache does allow a higher write percentage, but it is not optimal for the way FlexCache in ONTAP processes the write.

Access Control Lists, Permissions, and Enforcement

FlexCache supports both the NTFS and the UNIX security style at the FlexCache volume. With Qtree support starting in ONTAP 9.6, it also supports a different qtree security style than the parent volume. This means that the FlexCache enforces the permissions that were configured at the origin volume. However, to enforce the permissions, the same directory configurations must be applied to the SVM owning the FlexCache as they are on the origin. Multiprotocol access is now spread across both the origin and the caches. In other words, if there was only SMB access at the origin, but now NFSv3 access is happening at the FlexCache, that volume is now multiprotocol globally. Multiprotocol configurations must now be applied to both the origin and the FlexCache volumes.

NTFS Style Volume at the Origin

If the origin is an NTFS security style, then the FlexCache will also be that NTFS security style. The FlexCache SVM must have a CIFS server configured. The CIFS server must be in the same domain or a in a trusted domain. NTFS permissions are saved with the SIDs instead of with user or group names, so the SVM that serves the FlexCache must be able to look up those SIDs. When the SVM is in the same domain, forest, or two-way trusted domain, then the permissions can be properly enforced. If there is no CIFS server setup, or the domain it's in has no trust to the domain configured at the origin, then permissions could have unintended results.

UNIX-Style Volume at the Origin

If the origin is a UNIX security style volume, then there are several options. If there is no Lightweight Directory Access Protocol (LDAP) client applied to the origin SVM, then there is no reason to create a configuration at the FlexCache. NFSv3 user IDs (UIDs) and group IDs (GIDs) can work perfectly with UNIX mode bits on the files and folders in the volume without the need to lookup information in LDAP.

If there is an LDAP client configured and applied to the origin SVM, then the best practice is to configure the same LDAP client configuration at the FlexCache. The same LDAP domain is needed for UNIX-style volumes because of the possibility of UID and GID conflicts in the two LDAP domains. LDAP referrals can find the correct information, but the risk of UID and GID conflicts is higher if you rely on referrals to provide the same information rather than configuring the ONTAP LDAP client at the cache to contact the same LDAP domain as the origin directly. The same holds true for local UNIX users and groups. If there are users or group configured for access at the origin SVM, this access permission must be replicated at the FlexCache SVM.

If NFSv4 access control lists (ACLs) are present in the origin volume, then the LDAP client configurations must match. The NFSv4 protocol does not use UIDs and GIDs like the V3 protocol, but it does use group names and user names. For ACLs to be applied properly at the FlexCache, ONTAP must be able to locate these group names and user names.

Multiprotocol Access

If multiprotocol access is allowed and configured to the origin volume, then the two previous sections should also apply to the FlexCache SVM. A CIFS server and LDAP client configuration to the same domain should be created and applied at the FlexCache. In addition, user name mapping configurations must be replicated.

User name mapping

If there is any user name mapping configuration at the origin, then the configuration should be duplicated and applied at the FlexCache. This is to ensure that permissions are enforced, and credentials are created in the same way at the FlexCache as they are at the origin.

Best Practice 7: Configure CIFS server, LDAP client, and user name mapping in the same way as the origin

The FlexCache SVM should have a CIFS server in the same domain, forest, or a trusted domain of the origin's CIFS server. The FlexCache SVM should also have the same LDAP configuration as the origin. The LDAP server name can be different, as can the bind user. As long as that host name serves the same domain as the origin and the bind DNs (Distinguished Name) have the exact same access, it's allowed.

User name mapping configuration should also be duplicated and applied to the FlexCache SVM.

Auditing, FPolicy and Antivirus Scanning

ONTAP 9.7 introduces the ability to perform auditing, antivirus scanning, and FPolicy operations at the origin. Because the origin is where writes land, antivirus protection is only required there. After the origin is protected with antivirus software, all caches are compliant, and there is no need for antivirus configuration at the cache.

Antivirus protection in ONTAP can only be configured with SMB as the protocol to the antivirus server. Therefore, if antivirus protection is desired for the FlexCache origin, a CIFS server must be configured there. In addition, ONTAP only supports on-demand scanning for SMB operations. FlexCache operations are not SMB operations, and therefore you must use the on-demand scanning schedule for the volume and SVM.

Best Practice 8: Configure a CIFS server and use on-demand scanning for antivirus protection

If you want to use antivirus protection for a volume with FlexCache relationships, configure a CIFS server and on-demand antivirus scanning at the origin.

Auditing is only performed on the origin volume. Because all writes and modifications end up at the origin, the need to audit writes and modifications is covered for all volumes. Reads are only audited when the read is at the origin, not at the cache.

FPolicy is also only performed at the origin volume. This means that only reads and writes at the origin are available for FPolicy. Native FPolicy file blocking is available for all writes because writes land at the cache. Other third-party FPolicy servers are available, but consider that only the origin forwards events to the FPolicy server. The caches do not forward any events.

Cache Volume Size

What is the optimal cache volume size? It depends on your data. The working set is the amount of data used at the FlexCache for a particular run or job. The working set determines how the FlexCache should be sized. For example, if the origin volume has 1TB of data in it, but a particular job only needs 75GB of data, then the optimal size for the FlexCache volume is the working set size (75GB) plus overhead (approximately 25%). In this case, 75GB + 25% * 75 = 93.75GB or 94GB. This is the most aggressive sizing to make sure that all data that is being read is cached at the FlexCache. There are some exceptions to this rule that are outlined in this section.

The other method to determine optimal cache volume size is to take 10-15% of the origin volume size and apply it to the cache. For a 50TB origin volume size, a cache should be 5TB to 7.5TB in size. You can use this method in cases where the working set is not clearly understood and then use statistics, used sizes, and other indicators to determine the overall optimal cache size.

Best Practice 9: Specifically define the cache size

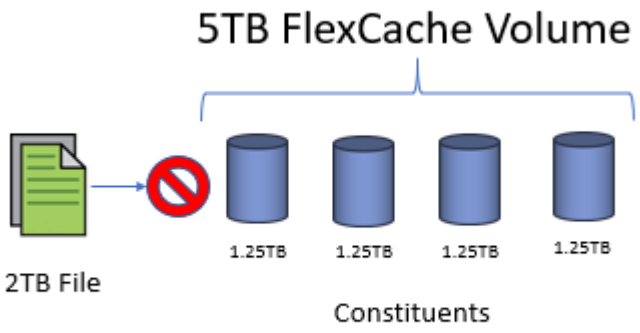
Always use the `-size` option for the FlexCache create to specify the FlexCache volume size.

File Sizes

Since the FlexCache is a FlexGroup, there are other considerations when deciding how large the FlexCache should be for optimal caching. FlexGroup volumes create constituent volumes to create the FlexGroup container. For more information about FlexGroup volumes, see TR-4557: NetApp FlexGroup Volumes: A Technical Overview. If there is a file in the dataset that must be cached that is larger than any of the constituents, then the file is always served from the origin. It is never cached because there is not enough room in the constituent volume for that file.

In the previous example of a 50TB origin and a 5TB to 7.5TB FlexCache, four is the recommended number of constituents from the previous section on constituent sizing. Any file over 1.25TB to 1.8TB in size won't be cached because the constituents are not large enough to cache it. Beginning in ONTAP 9.6, a FlexGroup feature called elastic sizing takes free space from one constituent and add it to another so that a larger file can be cached. However, NetApp recommends the proper sizing of constituent volumes so that you do not need to rely on this feature. See TR-4551 for more information about elastic sizing.

Figure 16) File too large to be cached.



This is the main reason NetApp recommends creating the correct numbers of constituents in the FlexCache volume is based on the volume size. It optimizes the space needed to cache each file on the origin.

Best Practice 10: Cache size should be larger than the largest file.

Because a FlexCache is a FlexGroup, a single constituent should not be any smaller than the largest file that must be cached. There is one constituent by default, so the FlexCache size should be at least as large as the largest file to be cached.

Note: If the constituent size is smaller than the file size being cached, ONTAP still attempts to cache the file. This results in evictions from the cache because of size.

Evictions

In FlexCache, files can only be evicted from the cache because of space constraints. The scrubber begins when any of the constituents are more than 90% full. ONTAP has a counter that indicates how many times the scrubber has run to evict files due to space.

```
fc_cluster::*> statistics show waflremote -counter scrub_need_freespace -raw

Object: waflremote
Instance: 0
Start-time: 11/8/2018 21:46:39
End-time: 11/8/2018 21:46:39
Scope: fc_cluster-01

  Counter                           Value
  -------------------------------- --------------------------------
  scrub_need_freespace                    172984
```

During steady state, it is not uncommon to see the FlexCache volume at a high percentage-use value. A high percentage-use value is normal for steady-state and it should not be considered a problem if FlexCache volume use is consistently at 80-90%.

Autogrow

Sometimes, autogrow might be a good option to use on the FlexCache to conserve space. You might consider using autogrow when you don't know what the working set size is or if you must be conservative with space on the FlexCache cluster.

To set up autogrow on a FlexCache volume, the FlexCache volume must be created with an initial size. NetApp recommends setting the initial FlexCache volume size to between 1% and 5% of the origin. You must also consider the file sizes of the data because larger file sizes affect space more than constituent sizes do when autogrow on the volume is enabled.

Let's reuse the previous example of a 50TB origin volume. The initial FlexCache volume size is set to 500GB to 2.5TB; any one file should not exceed that size to keep the optimal constituent sizes. These file sizes are more likely to be present in the origin than the 1.25TB to 1.8TB sizes mentioned before. Therefore, they have more of an effect than when using autogrow with the smaller cache and smaller constituent sizes. NetApp also recommends setting the maximum autogrow size to between 10% and 15% of the origin. This keeps the FlexCache from exceeding the maximum as it caches more data.

```
fc_cluster::*> volume autosize -volume vol_fc_cache1 -vserver fc-svm1 -maximum-size 5TB -mode
grow
```

Autogrow is only triggered at a certain threshold. By default, this threshold is 85%. When a particular constituent reaches 85% full, then it is grown to a specific number calculated by ONTAP. Also, the eviction threshold is 90%. So, if there is an ingest rate (first read from origin, which writes it to cache) of greater than 5%, then grow and evict loops could result in undesirable behavior.

When autogrow occurs, EMS messages are generated. These messages show what a constituent is and by how much the constituent grows. See the following example:

```
fc_cluster::*> event log show
Time        Node       Severity   Event
------------------- ---------------- ------------- ---------------------------
11/12/2018 19:45:15 fc_cluster-01   NOTICE    wafl.vol.autoSize.done: Volume autosize: Automatic
grow of volume 'vol_fc_cache1__0001@vserver:6cbc44db-cd6f-11e8-a1ce-00a0b8a0e70d' by 110MB is
complete.
```

LS Mirror Replacement

LS mirrors for data volumes have been deprecated since ONTAP 9.1. FlexCache can be a replacement for the ability of data LS mirrors to spread reads across multiple disks and nodes in a cluster. Spreading the load across multiple disks and nodes in a cluster can alleviate a hot-volume, hot file, hot directory or increase the load balancing of the data. The only way to make this feature available to clients is through automounter.

NFS clients can use advanced automount map tricks to mount different volumes, either one of the caches or the origin, in a way that spreads the connections over multiple LIFs and possibly SVMs. This method does not evenly spread the load across multiple SVMs, nodes, or volumes, but it can balance connections to a certain data volume over multiple LIFs, nodes, and SVMs.

Install Autofs

Most of the standard repositories have autofs available as a package. This can be done by using the following commands:

```
yum install autofs
apt-get install autofs
dnf install autofs
```

Create the FlexCache

The next step is to create the FlexCache. If the FlexCache is created in the same SVM as the origin, then no peering is necessary. If the FlexCache is being created in a different SVM, then the SVMs must be peered and the FlexCache application must be added to the peer. There is no advantage to one method or the other, but there is a management advantage to having the FlexCache in a different SVM. The mount point can be the same across all FlexCache volumes and the origin.

```
fc_cluster::> flexcache create -vserver fc-svm1 -volume data_volume -aggr-list aggr1_node1 -size
100G -origin-vserver origin-svm -origin-volume data_volume -junction-path /data_volume
```

Create an Automount Map File and Add it to the Auto.Master File

There are several different ways that the FlexCache can be defined in the automount map file. If the cache volumes are in the same SVM or do not have the same mount point as the origin, then you can define it in the map file as follows. The example shows the following:

- An origin volume on SVM `origin-svm` mounted to `data_volume`
- A cache on the same SVM mounted to `/data_volume_cache`
- A cache on SVM `cache1-svm` mounted to `/data_volume_cache1`
- A cache on SVM `cache1-svm` mounted to `/data_volume_cache2`

```
/mnt/data -rw,hard  origin-svm:/data_volume \
            origin-svm:/data_volume_cache \
            fc-svm1:/data_volume_cache1 \
            fc-svm1:/data_volume_cache2
```

If each FlexCache is in a separate SVM and uses the same mount path, there is a shortcut to define each FlexCache in the map file. An example follows of a map file that has an origin on the SVM `origin-svm` mounted to `/data_volume` and caches on `cache1-svm`, `cache2-svm`, and `cache3-svm`, which are also mounted on `/data_volume` in each SVM.

```
/mnt/data -rw,hard  origin-svm,fc-svm1, fc-svm2, fc-svm3:/data_volume
```

After the process is complete, restart the autofs service, and the data is then available at `/mnt/data`. You don't know which SVM the mount from the client will land on, but there are some rules that

automounter uses to maintain order. First, it looks for closeness in subnet masks. If one entry in the map file is in the same subnet as the client, then it is used exclusively. If there are multiple entries in the map file that are in the same subnet, then the NFS client pings each one by issuing a NULL call to each NFS service. The service that has the lowest response time is mounted.

Troubleshooting

This section discusses problems originating in FlexCache.

Files not being served. There are several possible reasons why a FlexCache is not serving a file. The primary one is that the file is not cached and there is no connection to the origin to retrieve the file. There are several commands that can be run to make sure that there is connection. The section covering disconnected mode has details on what to expect when a FlexCache is disconnected from the origin. It is important to understand which operations can and cannot be performed when a FlexCache is disconnected.

The following commands make sure that there is proper communication over the IC LIFs:

- Cluster peer ping
- Cluster peer connections show
- Network ping

Files are being served slowly. If are files being served slowly, then files are likely being loaded from the origin or from a slow link to the origin node. This error can be found with a combination of commands:

- Statistics show that `waflremote fc_retrieve_ops` shows an increase in operations that needed to go to the origin.
- Statistics show `waflremote fc_remote_latecy_histo` shows an increase in the range of the time it takes to retrieve the data from the origin.

Where to Find Additional Information

To learn more about the information described in this document, refer to the following documents and/or websites:

- FlexCache Volumes for Faster Data Access Power Guide
  http://docs.netapp.com/ontap-9/topic/com.netapp.doc.pow-fc-mgmt/home.html
- TR-4571: NetApp FlexGroup Volume Best Practices and Implementation Guide
  http://www.netapp.com/us/media/tr-4571.pdf
- TR-4557: NetApp FlexGroup Volumes A Technical Overview
  http://www.netapp.com/us/media/tr-4557.pdf
- TechOnTAP Podcast: Episode 165
  https://soundcloud.com/techontap_podcast/episode-165-flexcache

Contact Us

Let us know how we can improve this technical report. Contact us at docfeedback@netapp.com. Include TECHNICAL REPORT 4743 in the subject line.

Version History

Version	Date	Document Version History
Version 1.0	August 2018	Chris Hurley: Initial version for ONTAP 9.5.
Version 1.1	June 2019	Updated for ONTAP 9.6
Version 1.2	August 2019	Licensing updates for ONTAP 9.5 & 9.6
Version 1.3	December 2019	Updated for ONTAP 9.7
Version 1.4	December 2020	Updated for ONTAP 9.8

Refer to the Interoperability Matrix Tool (IMT) on the NetApp Support site to validate that the exact product and feature versions described in this document are supported for your specific environment. The NetApp IMT defines the product components and versions that can be used to construct configurations that are supported by NetApp. Specific results depend on each customer's installation in accordance with published specifications.

TR-4743-1220

NetApp