Technical Report

FlexCache in NetApp ONTAP

ONTAP Version 9.6

Chris Hurley, NetApp
August 2019 | TR-4743

Abstract

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

NetApp®

TABLE OF CONTENTS

LIST OF BEST PRACTICES

LIST OF FIGURES

1 Overview

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

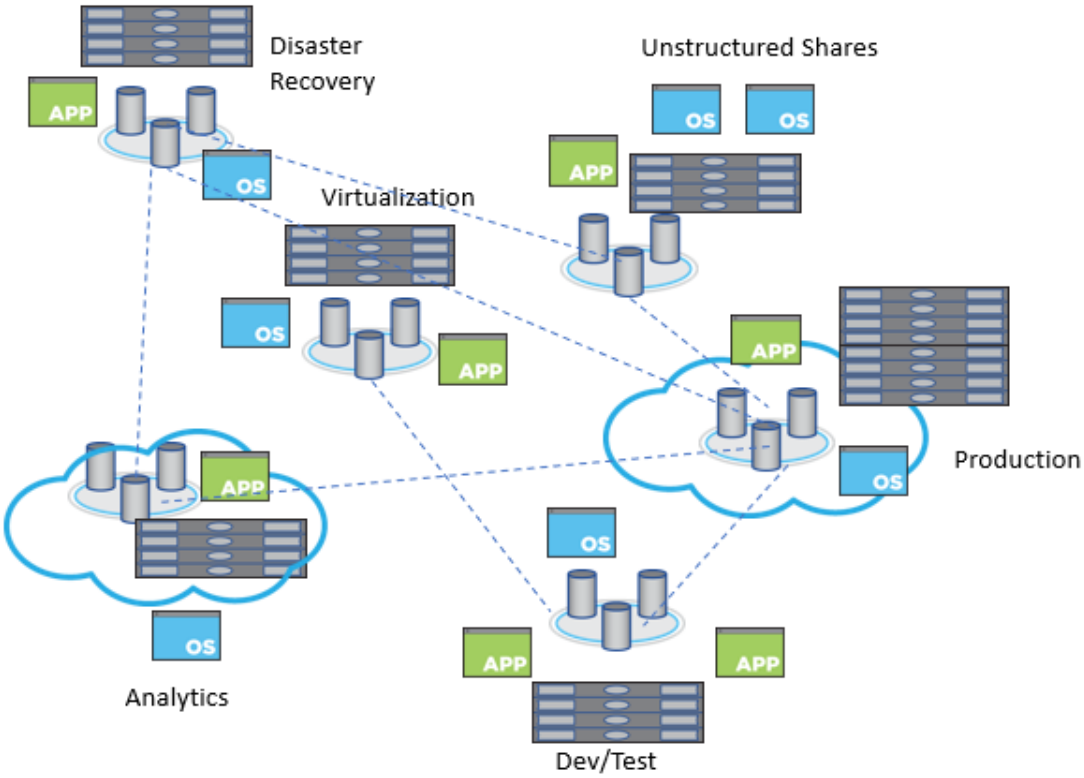
The recent revolutions in IT storage combined with the movement of many applications to the cloud is creating even more disparate datasets and namespaces. Managing data is becoming a massive challenge.

1.1 Large Datasets Prove Difficult to Manage

With the advent of AI, ML, and DL, data is ingested from a multitude of devices and places. The resulting datasets are growing and must be applied in many different places. Not only is the 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 home directories, among others, workflows create data in one place, but it is ingested or analyzed in another location. These workflows can cause problems for data consumers. These consumers can be applications, users, and other resources that need to access it. 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 amount of infrastructure consolidating this as a single namespace. This infrastructure is required to see the data as “one file system” and enable the writability, readability, and currency of the data in all places. Second, there is a latency issue. Accessing the data across the globe, or from a cloud provider, or 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 where collaboration, efficiency, time, and other key data strategies are not available for your enterprise.

Figure 1) Storage silos.



1.2 Data Replication

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

- **Duplication of equipment.** If you have 100TB of data in site A and you want it accessed in 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 in site B have the same experience as the data consumers in site A.
- **Possible extra equipment.** You not only need to duplicate the infrastructure to handle the replicated data, but you also must 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 are still incurring the cost and delay of moving the data on a scheduled basis. On-demand replication does exist, but depending on your data structures, 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 due to these delays. There is also questions such as “Are we working with data that is current?” or “When was the last time data was synchronized?”. And when replication schemes break, so does everything downstream.
- **Complication.** Having to manage multiple relationships, the additional effort in managing duplicate equipment and extra infrastructure makes data management more complicated.
- **Writability. Using** replication might not allow writability to the destination dataset.

1.3 Data Synchronization

Data synchronization (sync) can ensure that your destination data is writable. This can provide a two-way sync, but there are more costs in addition to the ones mentioned in replication.

- **Replication conflicts.** Keeping the 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 are time consuming, costly, and can compromise the data.

2 FlexCache in ONTAP: The Evolution

FlexCache in ONTAP solves the problems mentioned by providing a writable, persistent cache of a volume in a remote place.

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 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 in the cache (for example, solid-state drive (SSD) versus HDD), increased processing power in the cache, and increased (or faster) memory in 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:

- Improves performance by providing load distribution
- Reduces latency by locating data closer to the point of client access

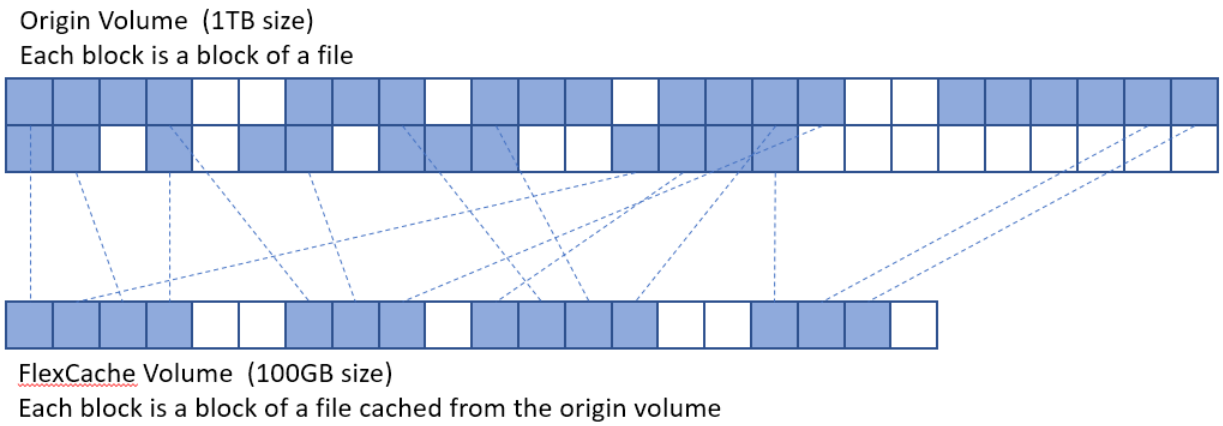
- Enhances availability by serving cached data in a network disconnection situation

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

A FlexCache is a sparse copy, not all files from the origin dataset can be 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).

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

Figure 2) Sparse volume details.



2.1 Terminology

Many of the usual ONTAP terms (such as storage virtual machine, logical interface (LIF), NetApp FlexVol®, 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 where 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 where the operation landed and 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 needs to 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. This 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).** 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. It is the feature that makes FlexCache happen.
- **Remote Entry Metafile (REM).** A a file at the origin that holds delegation information of all the files that are being actively cached in a FlexCache.
- **Remote Index Metafile (RIM).** A file at the cache that holds delegation information of 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.

2.2 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 is using and how FlexCache communicates with the origin. In 7-mode, the protocol was NetApp Remote Volume (NRV) and it ran over the data ports. Now, the protocol linking the FlexCache to the origin is RAL. 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.

2.3 Differences between FlexCache and ONTAP Features

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

SnapMirror

NetApp SnapMirror® provides a read-only copy of a volume to 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. During this operation, at any point in time, 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 was a concept in ONTAP where data volumes can be replicated within the same SVM. This was deprecated in ONTAP 9.1 and FlexCache is considered a replacement for this feature. FlexCache is not intended to replace LS Mirrors for SVM root volumes as this is still an 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 were automatically replicated to the LS mirror. This operation provided faster access to the data, as the LS mirror volume would be on the same node as the LIF. It also insulated against HA pair failure. This feature did not allow the LS mirror to be external to the SVM.

Provided in this document is a quick configuration of FlexCache as an LS Mirror replacement.

3 Use Cases

The FlexCache in ONTAP design offers the most benefit in specific use cases, but those 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 evaluate the benefits of FlexCache to the costs associated to the non-ideal use case.

3.1 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:

- Electronic design automation
- Media rendering
- AI/ML/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

3.2 Nonideal Use Cases

Because of the supported or nonsupported features for FlexCache in ONTAP, there are some non-ideal use cases. The value of these non-ideal use cases must be evaluated against your specific need to see if the benefits offset the drawbacks. The disadvantages include:

- Write heavy workloads at the cache
- Workloads that overwrite many of the files at the origin (this might cause unnecessary invalidations)
- Workloads that require specific features and functionality that are not currently available with FlexCache
- Virtualization workloads (datastore repositories for a hypervisor)

3.3 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 cache. Some features, although not supported at the cache, are supported at the origin. With the write-around nature of FlexCache, some features are not required to be supported at the cache. Since the origin coordinates all the 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	No
	NFSv3	Yes	Yes

	Supported Feature	Supported on origin?	Supported on cache?
	NFSv4	Yes	No
	NetApp FPolicy™	No	No
	Vscan	No	No
	NAS auditing	No	No
	SLAG (Storage Level Access Guard)	No	No
Volume features	FlexVol volume	Yes	Not applicable
	FlexGroup	No	Yes
	Qtrees	Yes	Not applicable*
	Quotas	Yes	Not applicable*
Data security	NetApp Volume Encryption	Yes	Yes
	NetApp Aggregate Encryption	Yes	Yes
Disaster recovery	NetApp Snapshot™	Yes	Not applicable
	SnapMirror	Yes	No
	SnapMirror Synchronous	No	No
	SVM Disaster Recovery	Yes**	Not applicable
	SnapLock	No	Not applicable
	MetroCluster™	No	No
Data mobility	SVM Migrate	No	No
	Volume Cloning (NetApp FlexClone®)	Yes	No
	Vol Move	Yes	Yes
	File Clone (SIS Clone)	No	No
	Vol Rehost	No	No
Data tiering	FabricPool	Yes	Not applicable
Configuration	Nested volume mounts	Yes	No
	System Manager configuration	Yes	Yes

* Qtrees and quotas are configured on the origin only. They will be enforced on both the origin and the caches without any configuration at the cache.

** 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.

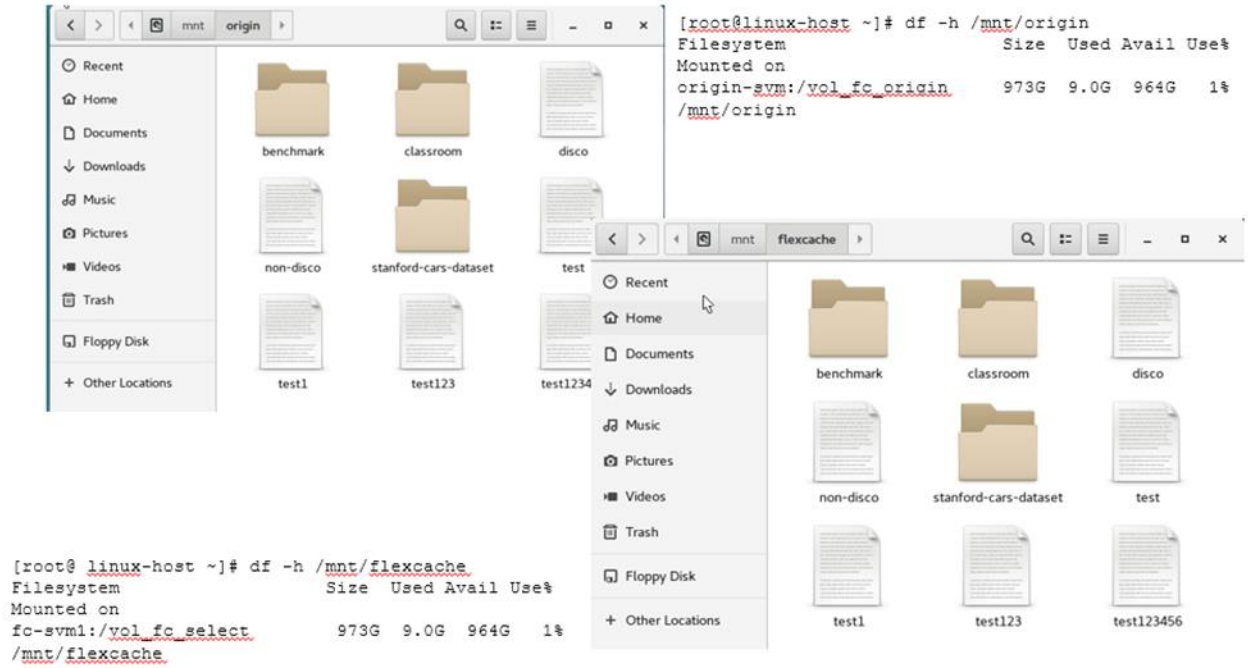
4 FlexCache in ONTAP Technical Overview

This section provides a technical overview of how FlexCache operates. This overview is provided so that proper business decisions can be made based on how the feature works. FlexCache is mainly a “set and forget” feature so there is not much configuration needed.

4.1 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. From the client’s perspective that has mounted the FlexCache volume though, the volume looks to be the same as the origin volume, regardless of if anything is cached or not. 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 actual difference in size and actual data in the respective volumes.

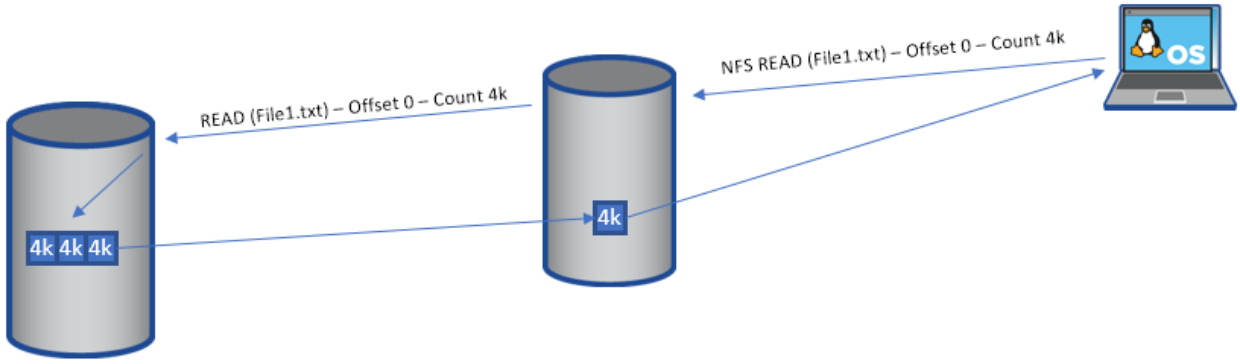
Figure 3) View of origin volume and FlexCache volume.



The cache is populated as the data is read by an attached client. See Figure 2) Sparse volume details for a graphic detail of this. 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.

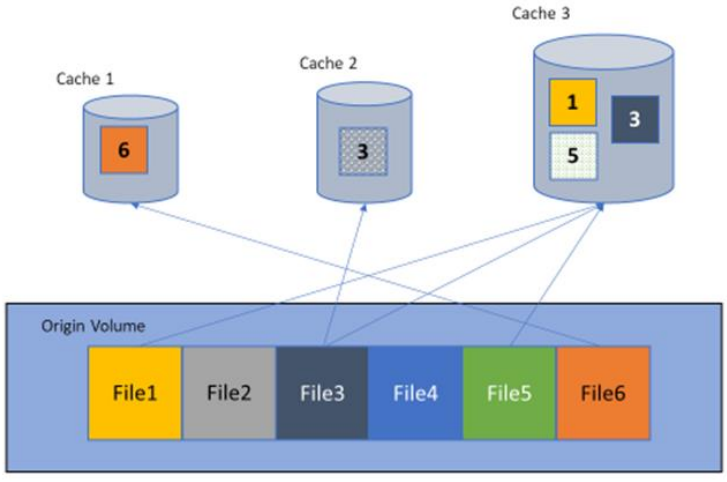
Figure 4 illustrates a client reading 4096 bytes of data from file 1 at cache 1. Since file 1 is a 12k file, only a third of the file is retrieved from the origin and cached in the FlexCache volume. If the client then requests the rest of the file, it is also retrieved from the origin and then cached. 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 will be different. Figure 5 demonstrates how multiple caches against a single origin can all look different. The origin has six files in it. Cache 1 is the smallest cache volume and it is caching the entirety of File 6. Cache 2 is slightly larger than Cache 1 and it is caching only a few blocks of File 3. Cache 3 is the largest of the caches and it is caching the entire File 1, a few blocks of File 5, and the entire 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 the entire File1, a few blocks of File 5, and the entire File 3. Again, this could have been done with any permutation of requests across the clients attached to the cache.

Figure 5) Cache variation.



4.2 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 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 a different SVM, 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, the same export-policy can be applied to both volumes.

4.3 RAL Protocol Overview

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 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 where the FlexCache is created on. The destination node is the node that owns the aggregate where the origin volume lies.

There are also two new files introduced for FlexCache; REM and RIM. REM is kept at the origin to keep all the cache delegations. When a file is read from a cache, the SVM UUID and the file inode are placed in the REM file at the origin. This becomes a delegation on the origin side. On the cache side, the RIM is populated with the inode that has been cached which serves as a delegation entry.

4.4 Read Processing

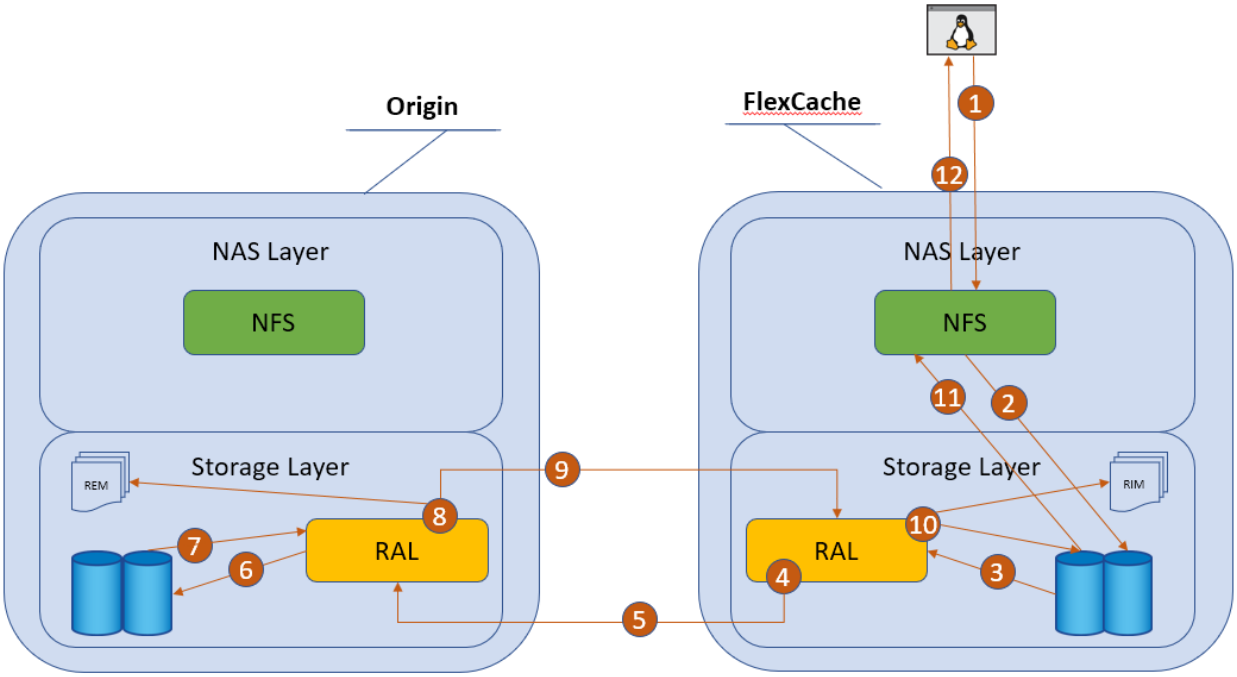
When a client issues a read for a file, there are several ways that a FlexCache forks from the standard read process. First, if the file is not found locally, the read request is forwarded to the origin. This process means that the origin is responsible for returning any ENOENTRY (or File not found)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

File Not Cached

This is the scenario that you encounter when you first create the FlexCache. Every read does not have a cached inode and must be forwarded on to the origin for data. The steps 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 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. RAL monitor on the origin receives the request, then generates a storage operation for the disk.
7. The inode is retrieved from the disk.

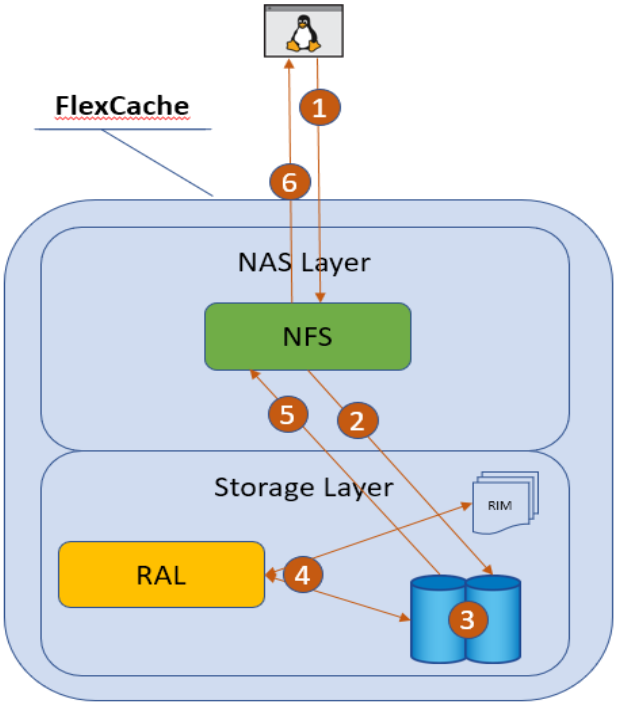
8. RAL then creates an entry into the REM file for delegation and generates the response to the FlexCache.
9. The response is sent over the cluster IC LIF back to the FlexCache node.
10. RAL receives the response, stores the data on local disk for the inode, and creates an entry in the RIM file about this inode.
11. The original storage operation is restarted, the data is retrieved, and the response is sent back to the NAS layer.
12. Then NAS layer then sends the protocol response back to the client.



File Cached and Valid

File cached and valid is the best scenario where the file is cached in the FlexCache and the delegation is still valid. 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 are as follows:

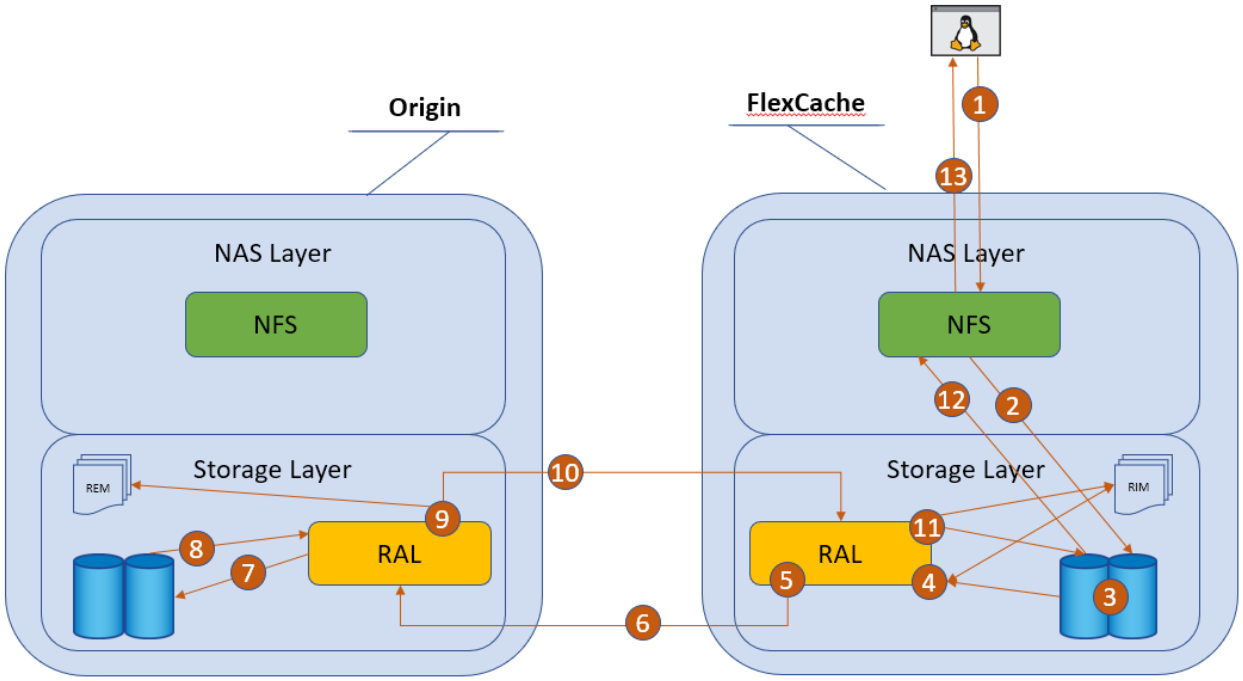
1. A protocol request from the client reaches the NAS layer.
2. The NAS layer then parses the operation and passes the optimized operation to the storage layer.
3. The storage layer then determines the operation is on a FlexCache (remote) inode. When it tries to load the inode, it finds it.
4. Because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



File Cached but Not Valid

File cached but not valid is a scenario where the file was originally cached, but something changed at the origin causing the cached file delegation to become invalid. This scenario means that even though the file is cached, it is not current, and it needs to be refetched from the origin. The invalidation happens at the file level so any changes at the origin invalidate the file. The figure below outlines the steps in this scenario as:

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. 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. RAL monitor on the origin receives the request, then generated a storage op 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. Then NAS layer then sends the protocol response back to the client.



As you can see, there are a few different ways that the read requests can be directed, and it depends on whether the file is cached and is valid. There are several scenarios where the origin can invalidate a cached file. First is when there is a write to the file. This write can be done from a cache or directly at the origin. Next 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, and 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 FlexCaches.

Best Practice 1: Set atime update 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.

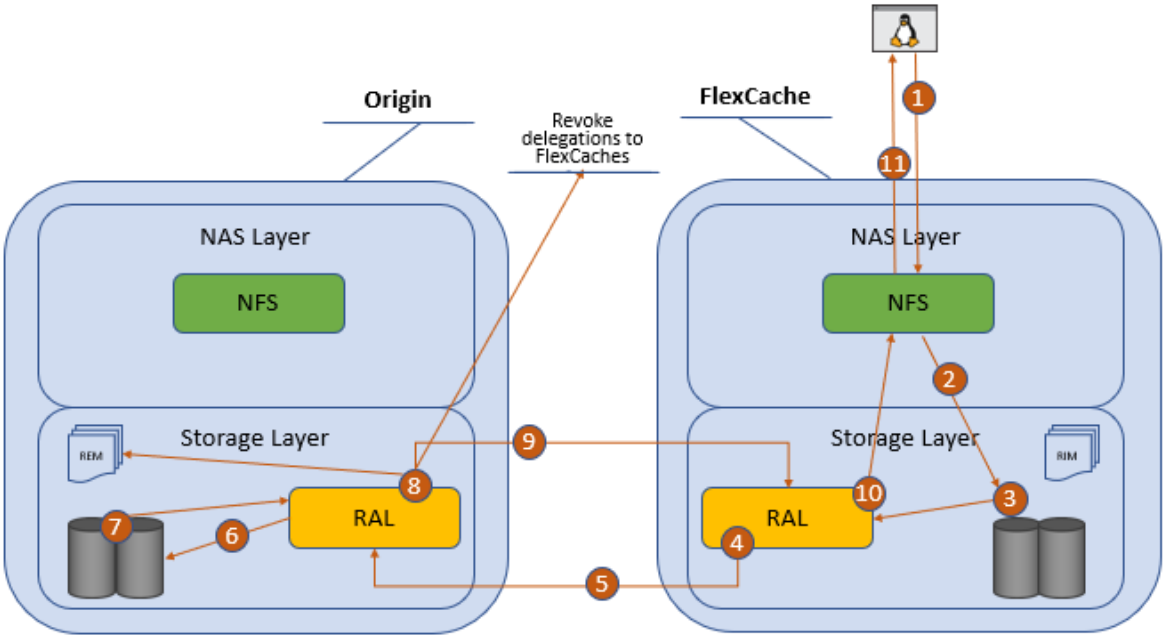
```
origin_cluster::*> volume modify -vserver origin-svm -volume vol_fc_origin
-atime-update false
```

4.5 Write Around Processing

Write around is a simple fork in the writing process. Instead of writing local to the disk, because ONTAP knows this is a FlexCache volume, the optimized write request is forwarded directly to the origin. 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 for the origin to coordinate simultaneous writes, locking, and cache invalidations. The write never lands on the cache. The figure below illustrates this process. Even if the file being modified is cached at the FlexCache, the write is still done by the origin and 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. Storage layer then determines 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. RAL monitor on the origin receives the request, then 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 FlexCaches to revoke the delegation for that file. This invalidation could happen to the FlexCache 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. Then NAS layer then sends the protocol response back to the client.



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

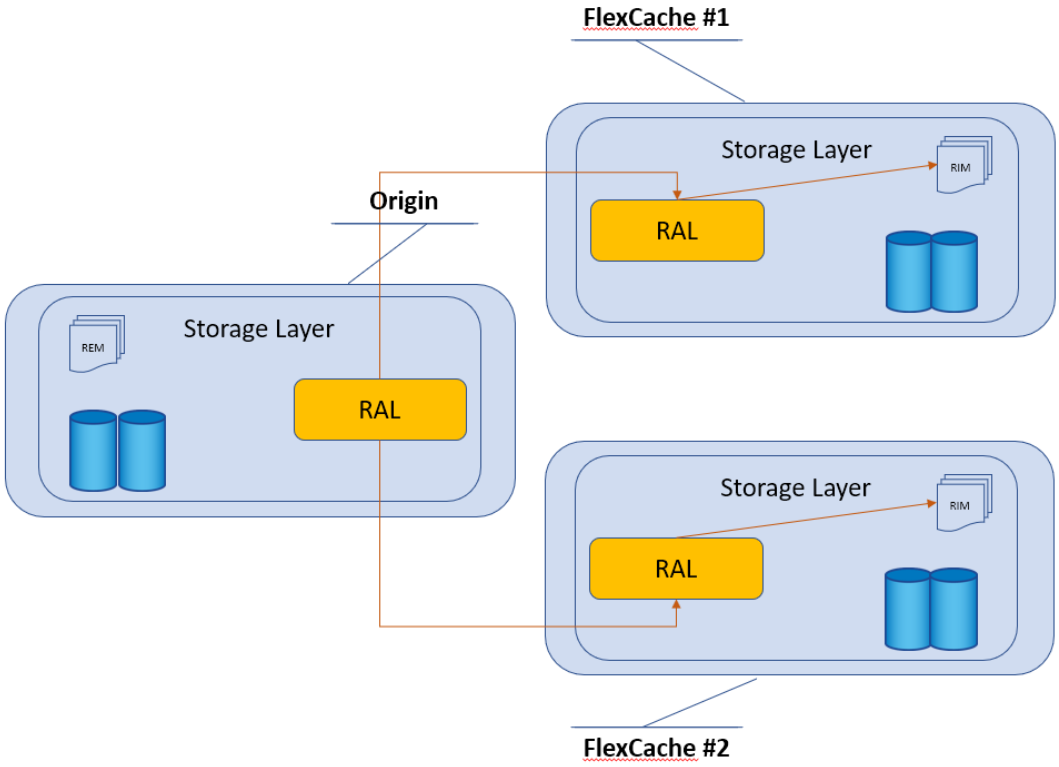
Best Practice 2: No read-after-write.
Try not to use applications that confirm writes with a read-after-write. The write around nature of FlexCache can cause delays in such applications.

Cache Invalidations

The REM file at the origin and the RIM file at the cache keep everything in sync. They know what files are cached, where the files are cached, and if the delegation entry is valid or has been removed. When the write is happening, the RAL layer at the origin then 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 6) Cache invalidation.



4.6 Locking

As mentioned before, locking is all coordinated at the origin. Being coordinated at the origin makes it easy to keep all the locks in order and to ensure that all locks are accounted for. Because the FlexCache volume currently only supports NFSv3 traffic, only NLM locking is supported at the FlexCache. When a client requests a file lock through NLM, that request is then transferred to the origin and the result is a FlexCache-aware NLM lock at the origin. To see the locks, you must look at them from the origin. 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.

Lock Recovery

When 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 is not tracking any locks at the FlexCache which means that lock recovery is not required for any FlexCache event (takeover, giveback, and 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.

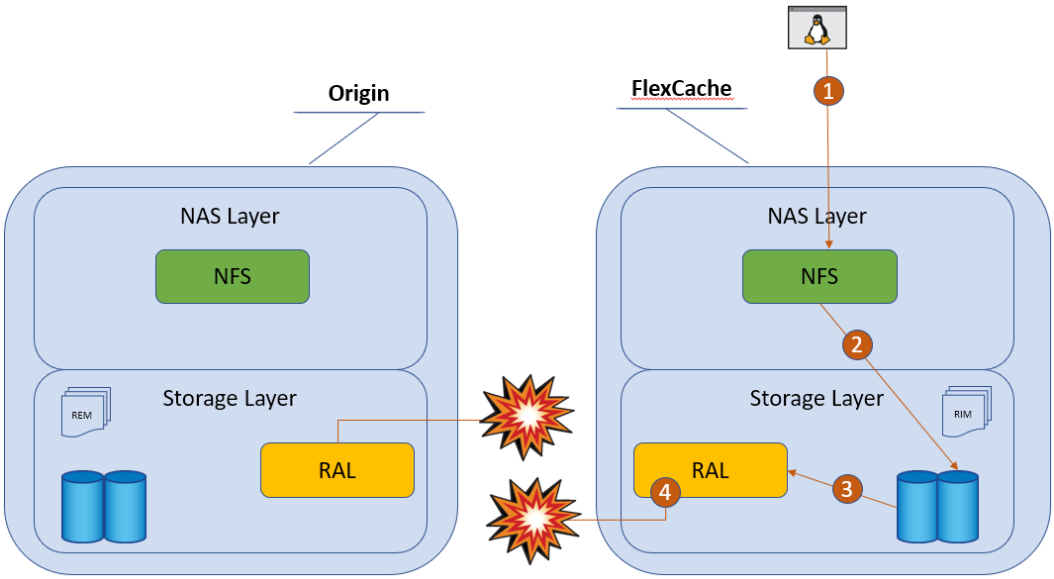
SMB and NFSv4 Locking at the Origin

SMB and NFSv4 are stateful protocols and have operations that result in exclusive read and/or write locks. This means that no other client should be able to read and/or write to a file. When these operations happen, ONTAP treats the FlexCache the same as any other client. Because the FlexCache has a read delegation when a file is cached at the FlexCache, that read delegation needs to be revoked. Upon locking, the origin also reaches out to any FlexCaches that have the file that is being locked and revokes the delegation. Therefore, the client sees the exact same result as if the access was straight to the origin.

4.7 Disconnected Mode

Sometimes the cluster that houses the origin and the cluster that houses the FlexCache cannot talk to each other over the IC LIFs due to WAN issues or other problems. When this happens, it is referred to as disconnected mode. Both the origin and the FlexCache 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 7) FlexCache disconnected mode.



An event management system (EMS) message is generated at the FlexCache when disconnected mode happens. 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 attempting to contact the origin. Following is an 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. There are normal EMS messages about the 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 been delegated through REM to the disconnected FlexCache are not allowed and will time out (NFS client receives an EJUKE error message). 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.
  - Reads for data that has not been cached time-out (NFS client receives an EJUKE error message)
  - Writes to the FlexCache time out (NFS client receives an EJUKE error message)
  - An “ls” only works if there was an “ls” or equivalent done on that directory before the disconnection.
  - A directory is just another inode. If it has been cached, it will be served. If it has not been cached, it won’t be served.
- At a nondisconnected FlexCache volume:
  - While there is one disconnected FlexCache volume, other nondisconnected FlexCache volumes operate as normal, with the same restrictions as outlined in the “At the origin” section .

READDIRPLUS, FlexGroups, and Disconnected Mode

To speed up `ls` commands to a NetApp ONTAP FlexGroup, there was a change 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 FlexGroups, 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 3: 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 heart beating mechanism to determine whether the connection between origin and cache nodes is good or bad. 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 where SVM cache-svm1 has a FlexCache linked to origin vol_origin1 in SVM origin-svm1 on cluster2.
2. The second is an origin where 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 which has multiple volumes. For more information about FlexGroups, see the NetApp documentation for FlexGroups volumes.

When the connection between origin and cache is marked disconnected, changes to files cached at the disconnected cache will proceed after the time-to-live (TTL) is reached. The TTL in ONTAP 9.6 is about 120 seconds. The status in 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 origin to declare the cache as disconnected. Cache, however, takes about one minute to mark origin as disconnected. Origin disconnect time is greater than cache disconnect time to weed out false positives where the cache hasn't determined the same result.

Last Access Time

By default, last access time, or atime, is a property of a file that gets 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, it 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 origin in certain cases.

In certain cases, disconnected mode might affect access to the origin volume, as outlined in Best Practice 1: Set atime update on origin to false.

5 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:

- 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. An example is as follows:

```
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 FlexCache traffic latencies which show average latencies per I/O operation for FlexCache. This view is in the 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     0ms
User-Default     2  543.00us  228.00us     0ms  315.00us     0ms     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

FlexCache licenses are now included in the price of ONTAP, but the license file is 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.

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 being used nor does it take origin volume size or 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 consider and autogrow limits. If each of those volumes could grow to 2TB, the additional 2.5TB of capacity potential is not counted against the license until autogrow increases the volume size.

When Do My Licenses Expire?

You can see the license expiration 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.
```

6 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 and are served from a non-FlexCache FlexGroup. Any performance information for FlexCache is identical to what it is for a FlexGroup.

When the FlexCache is across a latent WAN, then performance is more sensitive during the first read of a file. This sensitivity can be seen in the statistics mentioned before; `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 might be reduced in a few 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 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:

```
[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 an `ls -R <dir>` and replace the <dir> with the same as above.

7 Best Practices

This section discusses the best practices associated with FlexCache volumes.

7.1 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 is to keep the number of constituents at one. When the number of constituents is limited to one, then there is optimal space to store all the files that are being cached. You can accomplish this by only using a single aggregate in the -aggr-list option or combining the -aggr-list with the -aggr-list-multiplier to result in only one constituent volume.

Best Practice 4: FlexCaches should have one constituent.

Create the FlexCache with only the -aggr-list option so it creates 1 constituent.

```
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 -junction_path /vol_fc_cache1
```

7.2 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 the penalty of the latency of 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.

7.3 ACLs, 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 FlexCaches.

NTFS Style Volume at the Origin

If the origin is an NTFS security style, then the FlexCache will also be NTFS security style. The FlexCache SVM must have a CIFS server configured. The CIFS server needs to be in the same domain or a trusted domain. NTFS permissions are saved with the SIDs instead of 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 LDAP client applied to the origin SVM, then there is no reason to create a configuration at the FlexCache. NFSv3 UIDs and 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 could find the correct information, but the risk of UID and GID conflicts is higher when relying on referrals to provide the same information as when they are configured to contact the LDAP domain directly.

If NFSv4 ACLs are present in the origin volume, then the LDAP client configurations are required to match. The NFSv4 protocol does not use UID and GID like the V3 protocol, but it uses group and usernames. For ACLs to be applied properly at the FlexCache, the system must locate these group and usernames.

Multiprotocol Access

If multiprotocol access is allowed and configured to the origin volume, then both above sections should apply to the FlexCache SVM. A CIFS server and LDAP client configuration to the same domain should be created and applied at the FlexCache.

User Mapping

If there is any user 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 5: Configure CIFS server, LDAP client, and user 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. If that host-name serves the same domain as the origin and the bind DNs have the exact same access, it's allowed.

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

7.4 Cache Volume Size

What is the optimal cache volume size? It depends on the data. The working set is the amount of data to be 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 worth of data in it, but a job only needs 75GB of data, then the optimal size for the FlexCache volume is the working set size (75GB) plus overhead; which should be 25%. In this case, 75GB +25% would be 93.75GB, or 94GB. This is the most aggressive type of sizing to ensure that all data is being cached at the FlexCache that is being read. There are some exceptions to this 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, any cache would be 5–7.5TB 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.

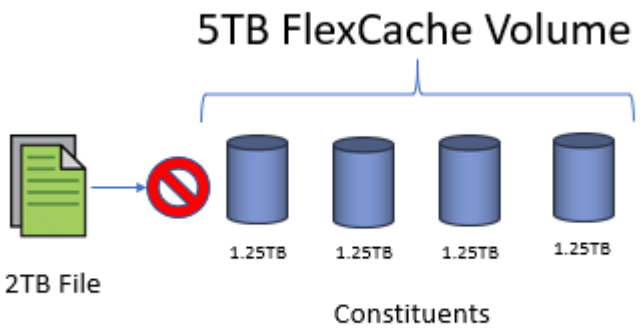
Best Practice 6: Specifically define the cache size.

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 considerations when deciding how large the FlexCache should be for optimal caching. FlexGroups creates constituent volumes to create the FlexGroup container. For more information about FlexGroups, see TR-4557: NetApp FlexGroup Volumes: A Technical Overview. If there is a file in the dataset that needs to be cached that is larger than any of the constituents, then the file is always be 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 5–7.5TB FlexCache, if there are four constituents and any file is over 1.25–1.8TB in size, then the file won't be cached.

Figure 8) File too large to be cached.



This is the main reason the default FlexGroup configuration for a FlexCache should create one constituent in a single aggregate. It optimizes the space needed to cache each file on the origin. The `flexcache create` command should be run with the `-aggr-list` modifier and the default number of constituents created is one constituent volume. This default can be modified by using the `-aggr-list-multiplier` option. This command changes the number of equal-sized volumes created for the FlexCache FlexGroup volume.

Note: If the option is used with a number greater than one, then the constituents will be smaller, and the file sizes must still be accounted for.

Best Practice 7: Cache size should be larger than the largest file

Since the FlexCache is a FlexGroup, a single constituent should not be any smaller than the largest file that needs to 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 in ONTAP, 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-used number. A high percentage-used is normal for steady-state and it should not be considered a problem if a FlexCache volume is consistently at 80–90% used.

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 would be set to 500GB–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–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 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 happens, EMS messages are generated. These messages show what a constituent is and by how much the constituent grows. For 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.
```

7.5 LS Mirror Replacement

LS mirrors for data volumes have been deprecated since ONTAP 9.1. FlexCache can be a replacement for the functionality that data LS mirrors provided to spread reads across multiple disks and nodes in a cluster. This functionality might also alleviate a hot volume use case, the only way to make this 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 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 need to 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 FlexCaches 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, you can define it in the map file as below. 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 with the same mount path, there is a shortcut to define it in the map file. Below is an example of a map file that has an origin on SVM origin-svm mounted to `/data_volume`, caches on cache1-svm, cache2-svm, and cache3-svm, all also mounted on `/data_volume` in each SVM.

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

After it is complete, restart the autofs service and then the data will be 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 keep an 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 will be used exclusively. If there are multiple entries in the map file that are in the same subnet, then the NFS client pings each by issuing a NULL call to each NFS service. The service that has the lowest response time is mounted.

8 Troubleshooting

When there is trouble with FlexCache, there are several places that can be examined. This section is targeted for problems that have a cause related to FlexCache.

Files not being served: There are several reasons a FlexCache will not serve 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 ensure that there is connection. The section of disconnected mode has details on what to expect when a FlexCache is disconnected from the origin. Understand the expectations about the operations that can and cannot be performed when a FlexCache is disconnected.

Some additional commands to ensure that there is proper communication over the IC LIFs are:

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

Files being served slowly: If there are files being served slowly, it indicates a file being loaded from the origin, or 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 of 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

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.

Copyright Information

Trademark Information

TR-4743-0819