Technical Report

FlexCache Volumes in NetApp ONTAP

ONTAP 9.5

Chris Hurley, NetApp
November 2018 | TR-4743

Abstract

NetApp® FlexCache® volume in NetApp ONTAP® data management software is a caching technology that allows for sparse writable replicas of volumes on the same or different ONTAP clusters. It can bring the data and files closer to the consumer 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.

Information Classification

Public

NetApp®

Version History

Version	Date	Document Version History
Version 1.0	December 2018	Chris Hurley: Initial version for ONTAP 9.5.

TABLE OF CONTENTS

LIST OF BEST PRACTICES

LIST OF FIGURES

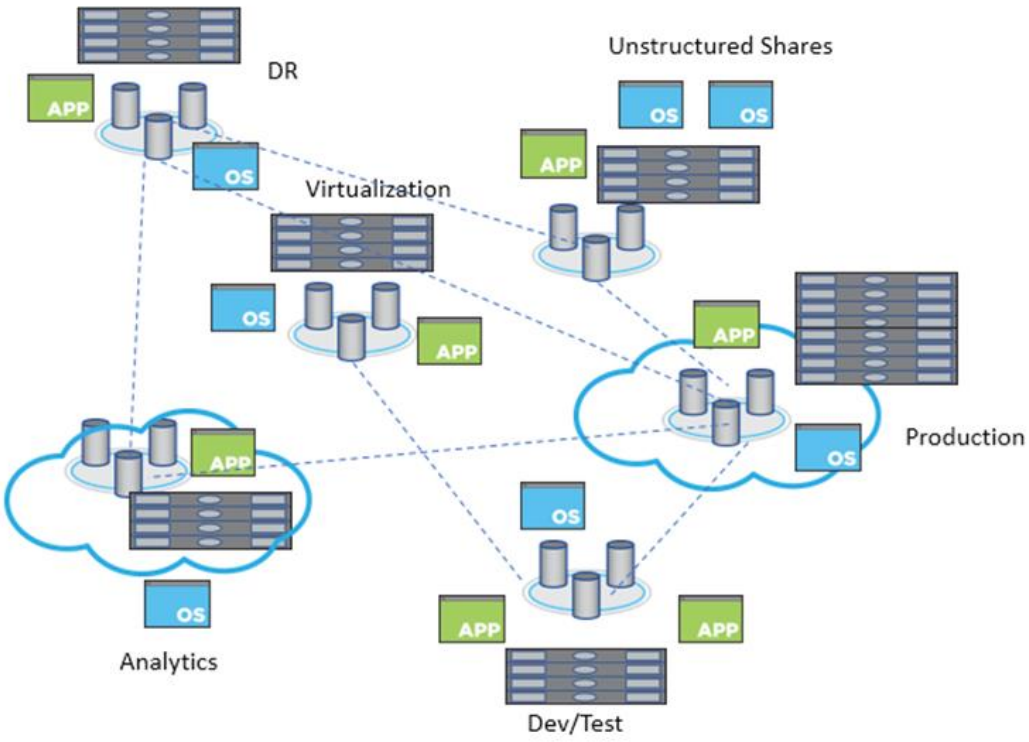
1 Data is Everywhere

As information becomes more global, data is being dispersed all over the globe. Enterprises and even small companies are creating, storing and accessing data from around the world. Datacenters are popping up everywhere, needing more and more data management to chorale all that data.

The recent revolutions in IT and storage, with many things moving to the cloud causes even more disparate data sets, name spaces and managing data can be a nightmare.

1.1 Large Datasets Prove Hard to Manage

With the advent of AI (Artificial Intelligence), ML (Machine Learning) and DL (Deep Learning), data sets are growing, ingested from a multitude of devices, places, and the resulting data sets are growing and need to be applied in many different places. This means that not only that the data is growing exponentially, but where the data resides and where the data is accessed is growing exponentially also. There are many other use cases where this is true. Media Rendering, Financial Applications, EDA (Electronic Design Automation), Home Directories, and plenty of other applications and data workflows create data in one place and it needs ingested or worked on in another location. This can cause headache for the data consumers. These consumers can be applications, users, and other resources that need to access it. The headache is two-fold. Firstly, because 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 is required to see the data as “one filesystem” and ensure the writability, readability and currency of the data in all places. The other is latency. Accessing the data across the globe, or from a cloud provider, or other physically separated location can be slow and prone to problems.



This ultimately creates islands of storage and data silos where colaboration, efficiency, time and other key data strategies are not available for your enterprise.

1.2 Data Replication

Many have turned to data replication in order to solve all the problems with the data explosion. The only way to accomplish this is to replicate the data between the two sites. This becomes costly in many ways.

1. 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 need to be at least similar so that the data consumers in site B have the same performance to the data as the data consumers in site A.
2. Possible extra equipment
   Not only will you need to duplicate the infrastructure to handle the replicated data, but you will also need to deploy new infrastructure to handle the replication configuration and monitor it.
3. Delays, delays and….more delays
   Replication schemas can only move only 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 will also be the question of “are we working with data that is current?” or “when was the last time data was synced?”. And when replication schemes break, so does everything downstream.
4. Complication
   Having to manage multiple relationships, the additional effort in managing duplicate equipment and extra infrastructure makes data management more complicated.
5. Writability
   Using replication may not allow writability to the destination dataset.

1.3 Data Sync

Data 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 Volumes in ONTAP: The Evolution

FlexCache volumes 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 main objective of a cache is to store frequently accessed portions of a source of data in a way that allows the data to be served faster and/or more efficiently than it would be by fetching the data from the source. Caches are beneficial in read-intensive environments in which data is accessed more than once and/or 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, 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.

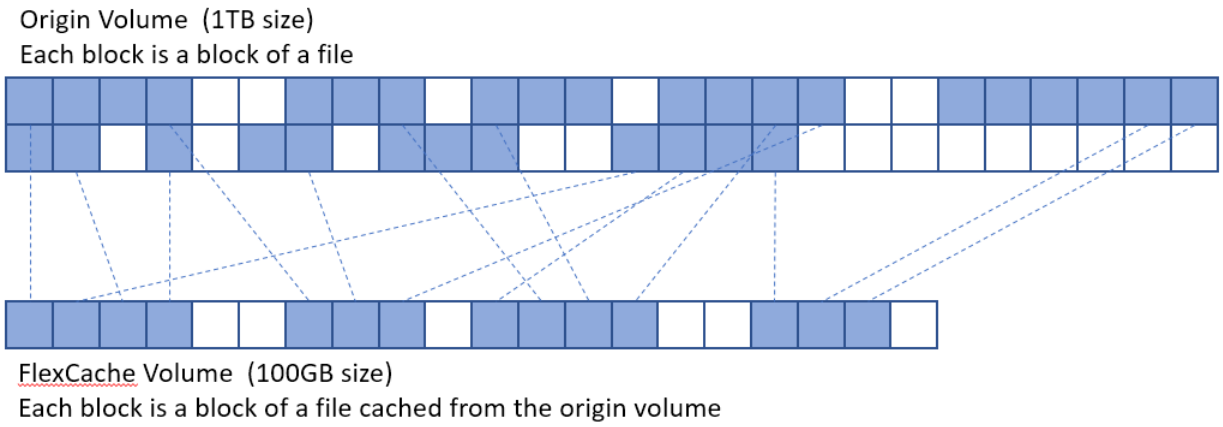
FlexCache volumes can be used for improved performance by providing load distribution, reduced latency through locating data closer to the point of client access, and enhanced availability by serving

cached data in a network disconnection situation. FlexCache volumes provide all the above while maintaining cache coherency, data consistency, efficient usage of storage in a scalable and performant manner.

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

A FlexCache volume also allows for the management of disaster recovery and other corporate data strategies to only need implementation at the origin. This can allow for better and more efficient utilization of resources, and for simpler data management and disaster recovery strategies.

Figure 1) Sparse volume details.



2.1 Terminology

Many of the usual NetApp ONTAP terms (such as storage virtual machine, logical interface [LIF], NetApp FlexVol® volume, 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 the following 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

NetApp ONTAP FlexGroup volume is a single namespace that is made up of multiple constituent member volumes and that is managed and acts like a FlexVol volume 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 the majority of operations are reads vs 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 cache volume where the operation landed and origin before responding to the client.

Write-around

The write operation is applied directly at the origin, bypassing the cache. In order to see the write at the cache, the cache will need to pull the information from the origin.

Working dataset

The subset of the total data that is stored at the origin that will be cached at the FlexCache volume. This is highly dependent 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 will be read at the FlexCache volume.

Remote Access Layer (RAL)

The remote access layer (RAL) is a feature in the NetApp WAFL® (Write Anywhere File Layout) system that allows the FlexCache volume 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.

REM

Remote Entry Metafile is a file at the origin that holds delegation information of all the files that are being actively cached in a FlexCache volume.

RIM

Remote Index Metafile is a file at the cache that holds delegation information of all the files that are being cached at that FlexCache volume.

FCMSID

FlexGroup MSID of the cache FlexGroup

Fan-out

The total number of caches that can be attached to a single origin

2.2 Comparison to NetApp Data ONTAP Operating in 7-Mode

Data ONTAP operating in 7-mode had a FlexCache feature that allowed for similar functionality. The new FlexCache volume in ONTAP is configured to be a replacement for this feature. The two solutions are comparable, but not the same since the technical specifications of the FlexCache feature in Data ONTAP operating 7-mode are different than the technical specification of the FlexCache volumes in ONTAP feature. The major difference between the Data ONTAP operating 7-mode feature and the current feature is the protocol FlexCache is using and how the FlexCache volume communicates with the origin. In 7-mode, the protocol was NRV (NetApp Remote Volume) and it ran over the data ports. Now, the protocol linking the FlexCache volume to the origin is RAL (Remote Access Layer). This will be explained more in detail in the Technical Overview section in this document. In addition, that ONTAP has cluster and SVM peering concepts, the protocol now runs over the intercluster LIFs.

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

2.3 Differences to Similar ONTAP Features

There are several ONTAP features that provide similar functionality. A high-level comparison of the features to FlexCache volume is as follows.

NetApp SnapMirror

NetApp SnapMirror® provides a read-only copy of a volume to a remote location. This can be done asynchronously (on a schedule) or synchronously (Sync SnapMirror). This 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 data set is copied over to the new volume and then when the scheduled sync time passes, all changes are copied from the primary to the secondary. This means that at any point in time, the SnapMirror secondary volume physically contains 100% of all data that is in the primary. This requires 100% of the capacity of the data size.

Load Sharing Mirrors

Load-sharing (LS) mirror was a concept in ONTAP where data volumes can be replicated within the same SVM. This has been deprecated in ONTAP 9.1 and FlexCache volume is considered as a replacement for this feature. This is not intended to replace LS Mirrors for SVM root volumes as this is still an ONTAP feature. Please see the Data Protection Power Guide or the SVM Root Volume Protection Express Guide for more information on LS Mirrors for SVM root volumes

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 helped both in providing faster access to the data, as the LS mirror volume would be on the same node as the LIF, and in insulating 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 a LS Mirror replacement.

3 Use Cases

The FlexCache volumes in ONTAP design is most beneficial 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, you just need to evaluate the benefits of a FlexCache volume to the costs associated to the non-ideal use case.

3.1 Ideal Use Cases

Since FlexCache volume is currently limited to a write-around model, it works best with workloads that are read heavy. Writes will incur latency and when there are fewer writes, the latency will 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

3.2 Non-Ideal Use Cases

Due to the supported or non-supported features for FlexCache volumes in ONTAP, there are some non-ideal use cases. The value of these will need to be evaluated with your specific need to see if the benefits outweigh the negatives. These include:

- Write heavy workloads at the cache

- Workloads that overwrite many of the files at the origin (this can cause unnecessary invalidations)
- Workloads that require specific features and functionality that are not currently available with FlexCache volumes
- Workloads that leverage Cloud Volumes ONTAP or a NetApp Cloud Volume Service (not currently supported)

3.3 Supported Features

This section covers which NetApp ONTAP features 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 and therefore, with the write-around nature of FlexCache volumes, 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.

Supported Feature	Supported on Origin?	Supported on Cache?
SnapMirror	Yes	No
FlexVol volume	Yes	No
FlexGroup volume	No	Yes
CIFS/SMB	Yes	No
NFSv3	Yes	Yes
NFSv4	Yes	No
FPolicy	No	No
VScan	No	No
VServer Migrate	No	No
MetroCluster	No	No
SnapLock	No	No
FabricPool	Yes	No
Volume Cloning	No*	No
NetApp Volume Encryption	No	No
NetApp Snapshot	Yes	No
VServer DR	Yes	No
Vol Move	Yes	No
NAS auditing	No	No
SLAG (Storage Level Access Guard)	No	No
Qtrees	No	No
Quotas	Yes**	No
Nested volume mounts	Yes	No
System Manager configuration	No***	No

* A FlexCache relationship can be established to a cloned volume and a volume can be cloned prior to establishing the FlexCache relationship and will not have any impact to the relationship. Once the FlexCache relationship has been established, the origin volume can no longer be cloned.

** As of this document, support for quotas at the origin will not be available until 9.5 GA (general availability). Availability of this feature is subject to change.

*** The only configuration required on the origin is having the FlexCache application in the Vserver peer relationship. Currently, there is no way to manage the Vserver peer applications through System Manager.

4 FlexCache Volumes in ONTAP Technical Overview

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

4.1 Sparse Data Details

FlexCache volume only caches the data that is read in the client requests. When a client mounts the FlexCache volume, initially, there is no data taking up space in that volume. The cache is populated as the data is read by an attached client. See Figure 1) 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 volume. This means that every cache is unique and no two caches will be exactly the same even though they all point back to the same origin. From the client’s perspective that has mounted the FlexCache volume though, the volume looks to be exactly the same as the origin volume, regardless of what is cached or not. In Figure 2, the origin is a 1TB volume and the cache is only 100GB in size. The volume has just been created, so there is no data actually cached. This figure illustrates that the origin and the FlexCache volume look exactly alike despite the actual difference in size and actual data in the respective volumes.

Figure 2) How the origin volume and FlexCache volume look to the client.

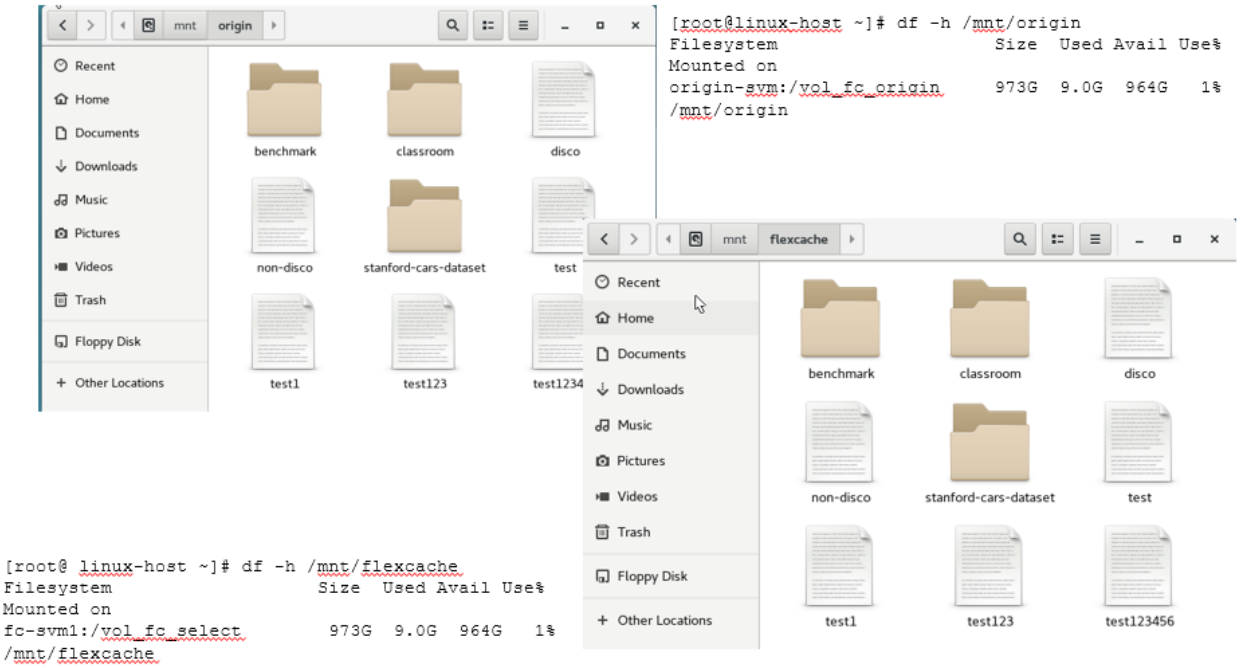
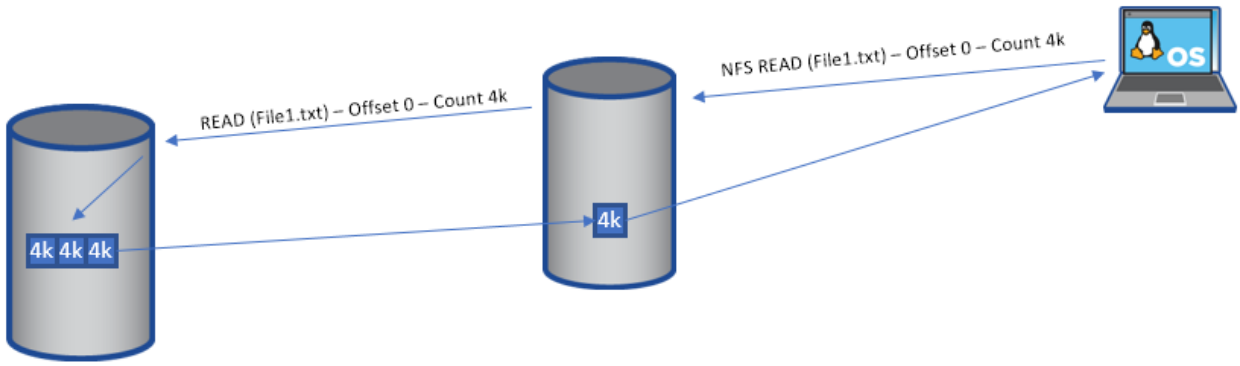


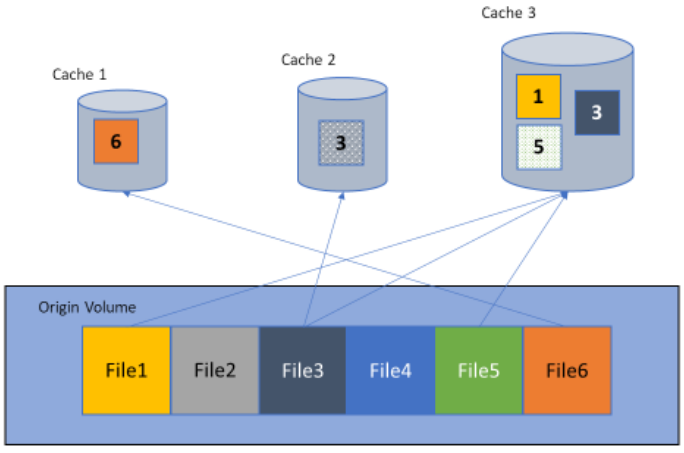
Figure 3 illustrates a client reading 4096 bytes of data from file 1 at cache 1. Since file 1 is a 12k file, only a third of the file is actually retrieved from the origin and cached in the FlexCache volume. If the client then requests the rest of the file, it is also retrieved from the origin and then cached. Any subsequent reads of any part of the file are then served directly from cache.

Figure 3) Sparse data details.



Since a FlexCache volume only has data that is requested by the clients that are mounted to it, each cache's data will be different. Figure 4 demonstrates how multiple caches against a single origin can all look different. The origin has 6 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 cache 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 client(s) 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

Figure 4) All caches are different.



4.2 RAL Protocol Overview

Remote Access Layer is the function FlexCache volume uses to allow loading and manipulation of content from any volume inside or outside of the cluster to the origin. This is essentially a traffic cop to keep the requests, data, delegations, lock requests and any other information between the FlexCache volume and the origin in sync. In some cases, the FlexCache volume and the origin are in different

clusters. This requires cluster peering for RAL to be able to forward on the IO operation to the node that owns the origin volume. When the RAL function is invoked, any operations that need 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 the node that originates the FlexCache volume to origin communication is the node that owns the aggregate where the FlexCache volume 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. The REM (Remote Entry Metafile) 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 is placed in the REM file at the origin. This becomes a delegation on the origin side. On the cache side, the RIM (Remote Index Metafile) is populated with the inode that has been cached and a delegation flag.

4.3 Read Processing

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

File Not Cached

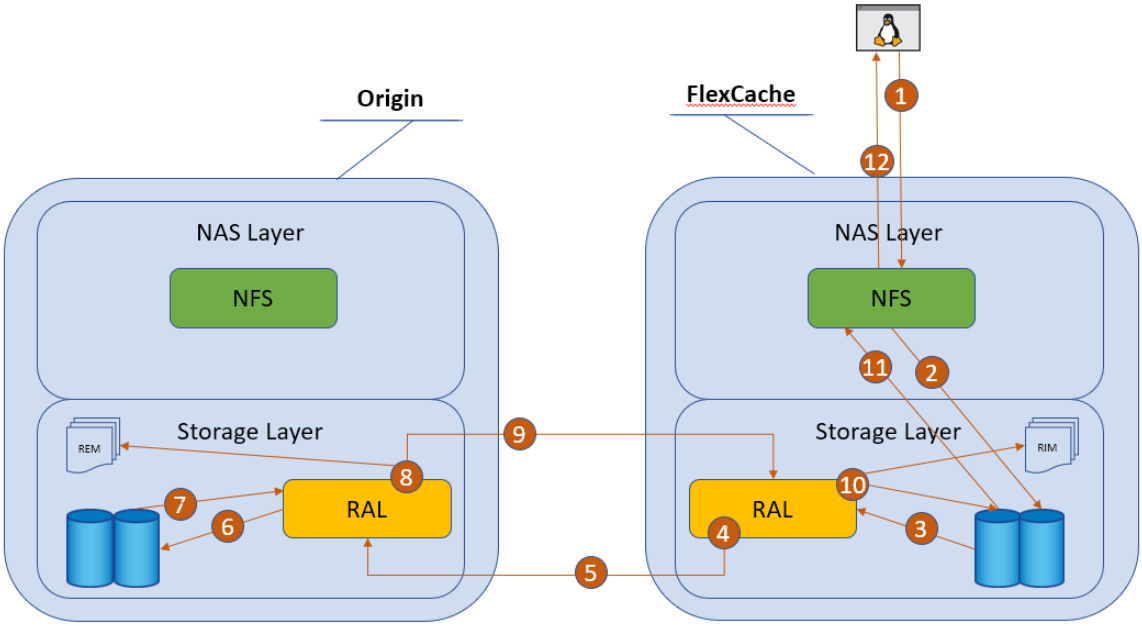
1. This is the scenario that you will encounter when you first create the FlexCache volume. Every read will not have a cached inode and will need to be forwarded on to the origin for data. A protocol request from the client reaches the NAS layer.
2. The NAS layer then parses the operation, and passes 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 its not cached so RAL kicks in. This 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 generated a storage op for disk.
7. The inode is retrieved from disk.
8. RAL then creates an entry into the REM file for delegation and generates the response to the FlexCache volume.
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.

Figure 5 shows the steps as follows:

13. A protocol request from the client reaches the NAS layer.
14. The NAS layer then parses the operation, and passes the optimized operation to the storage layer.
15. Storage layer then determines the operation is on a FlexCache (remote) inode. When it tries to load the inode, it finds its not cached so RAL kicks in. This also pauses the storage operation.
16. RAL generates a remote storage operation to retrieve the inode from the origin.

17. The remote retrieval operation is sent over the cluster IC LIF to the origin node.
18. RAL monitor on the origin receives the request, then generated a storage op for disk.
19. The inode is retrieved from disk.
20. RAL then creates an entry into the REM file for delegation and generates the response to the FlexCache volume.
21. The response is sent over the cluster IC LIF back to the FlexCache node.
22. RAL receives the response, stores the data on local disk for the inode, and creates an entry in the RIM file about this inode.
23. The original storage operation is restarted, the data is retrieved, and the response is sent back to the NAS layer.
24. Then NAS layer then sends the protocol response back to the client.

Figure 5) File not cached read steps.

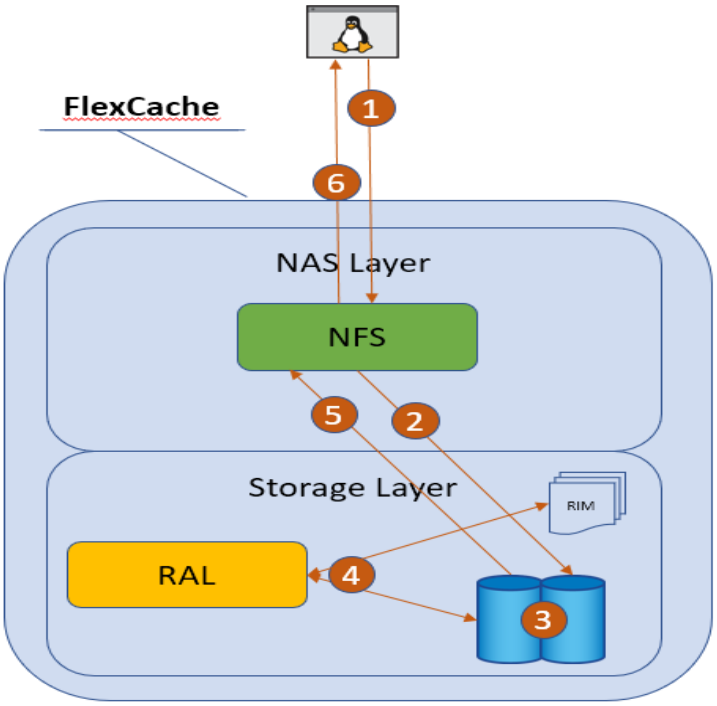


File Cached and Valid

This is the best scenario where the file is cached in the FlexCache volume 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 will work almost the same as a read from a non-FlexCache volume. The steps shown in Figure 6 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. When it tries to load the inode, it finds it.
4. Because this is a FlexCache inode, RAL kicks in and determines whether the delegation in the RIM file is still valid
5. Since it found a valid delegation, 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 6) File cached and valid steps.

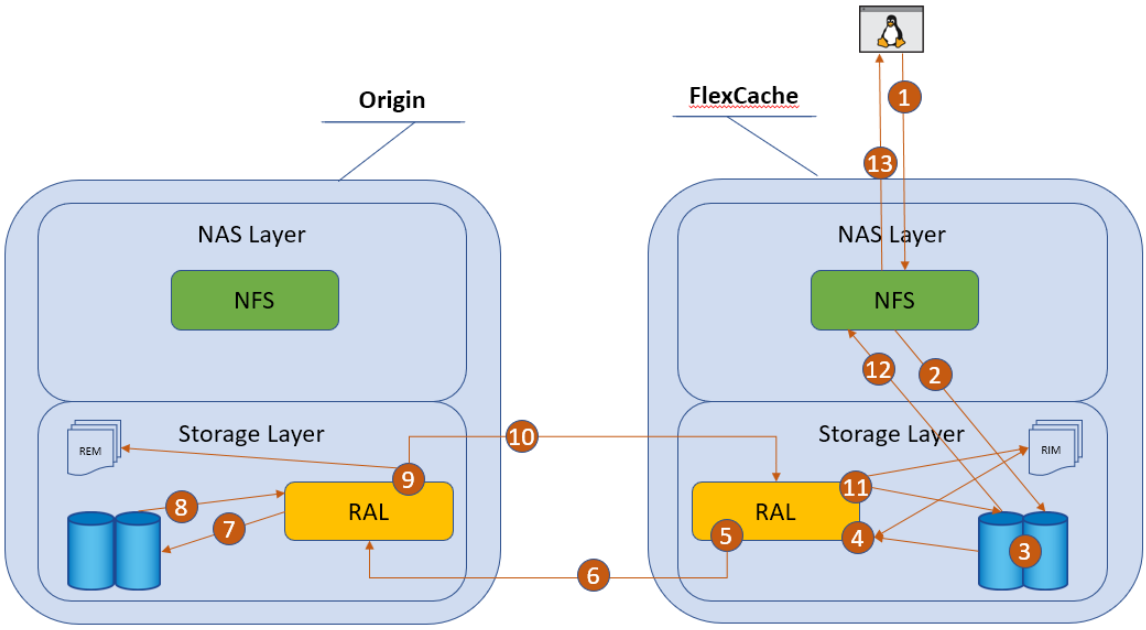


File Cached, but Not Valid

This is a scenario where the file was originally cached, but something changed at the origin causing the cached file delegation to become invalid. This means that even though the file is cached, it is not current and it needs to be re-fetched from the origin. In ONTAP 9.5, the invalidation happens at the file level so any changes at the origin invalidate the file as a whole. Figure 7 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 delegation in the RIM file is still valid.
5. Since the delegation 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 in the RIM file about this inode with a valid delegation.
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.

Figure 7) File cached but not valid steps.



As you can see from the above, there are only a few different ways that the read requests can get directed and it all 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 can be done from a cache or directly at the origin. Next is when there is a read of a file at the origin. This is due to the fact that by default, the atime-update property of a volume is set to true. This means that if there are many reads at the origin, then there can be many invalidations and cause the FlexCache volume to continually read from the origin. This is not an optimal setting, and therefore the recommendation is to disable 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 1: Set atime update on origin to false

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

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

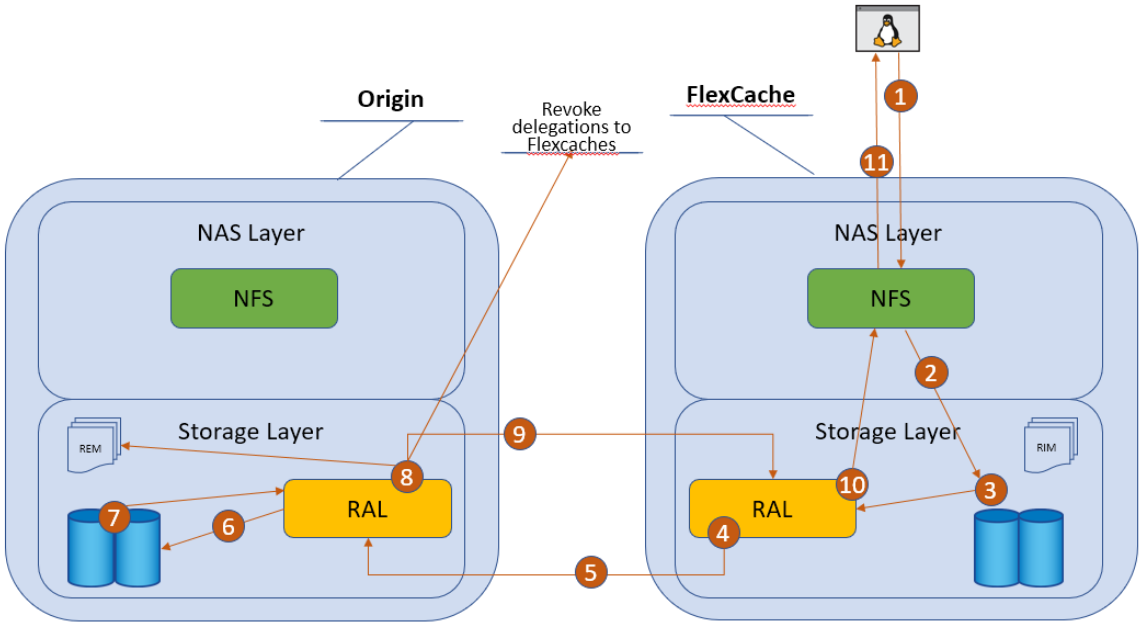
4.4 Write Around Processing

Write around is a fairly simple fork in the writing process. Instead of writing local to the disk, since ONTAP knows this is a FlexCache volume, the optimized write request gets forwarded directly to the origin. The origin then processes the write the exact same way as if the write request was requested from a client directly attached to the origin. This allows for the origin to coordinate simultaneous writes, locking and cache invalidations. The write never lands on the cache. Figure 8 illustrates this process. Even if the file being modified is cached at the FlexCache volume, the write is still done by the origin and then that cached file will be 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 cluster IC LIF to the origin node.
6. RAL monitor on the origin receives the request, then generated a storage op for disk
7. The data is then written to disk.
8. RAL then checks the REM file for any delegations if it is an existing file. If the file entry exists and there are valid delegations in the REM file, it contacts each of the FlexCaches volume to revoke the delegation for that file. (This invalidation could happen to the FlexCache volume writing the file if it has it cached).
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.

Figure 8) Write around.



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 that can confirm what was written. These are not suitable for FlexCache volume and should be tested extensively for adequate performance if FlexCache volume is to be part of the infrastructure of that application.

Best Practice 2: No read-after-write

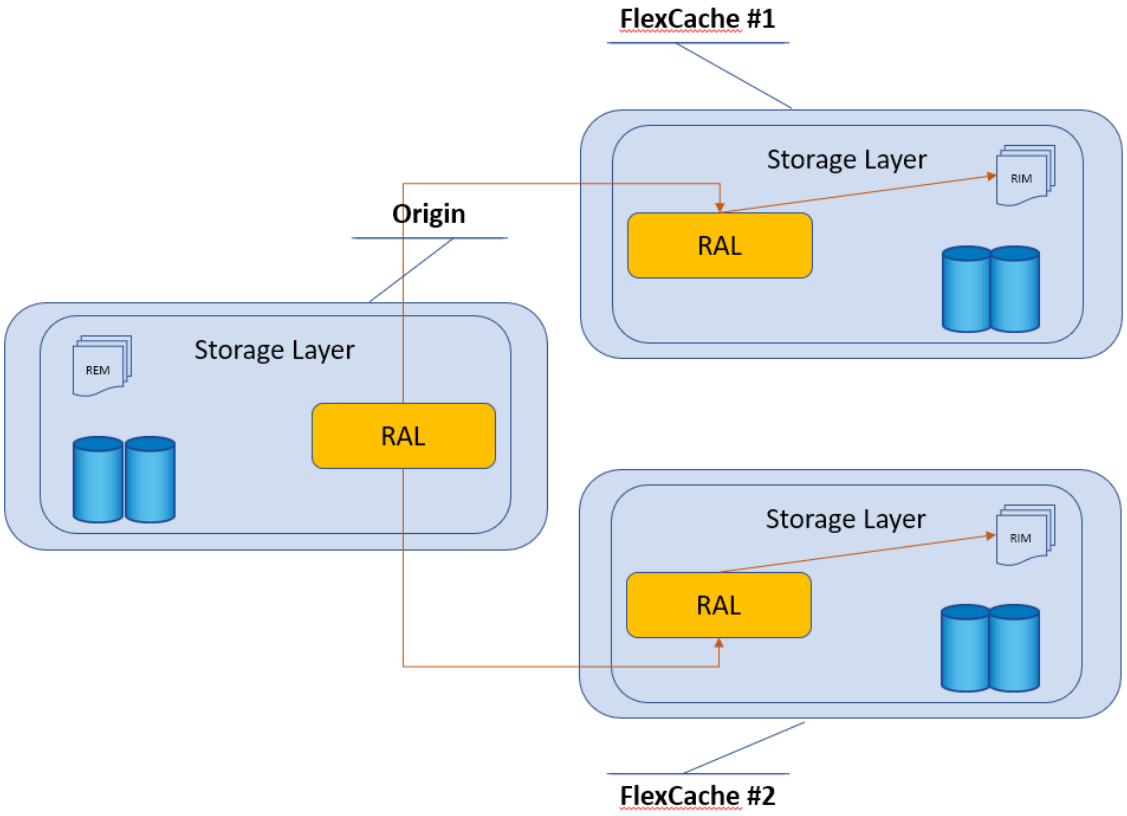
Try not to use applications that confirm writes with a read-after-write. This will cause delays in the application due to the nature of write around nature of FlexCache.

Cache Invalidations

The REM file at the origin and the RIM file at the cache keep everything in sync when it comes to knowing what files are cached, where the files are cached, and if the delegation is valid or has been revoked.

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 if a file has been cached on a FlexCache volume and which cache it was cached on. If the file being written has an entry in the REM file, then RAL will create a message to invalidate the file and send it to all the caches that have the file cached via the IC LIF. The cache then takes that message and invalidates, or revokes, the delegation of that file in the RIM. This way the next read of the file at any cache will result in a read to the origin because it has been invalidated.

Figure 9) Cache invalidation.



4.5 Locking

As mentioned before, locking is all coordinated at the origin. This makes it easy to keep all the locks in order and to ensure that all locks are accounted for. Since the FlexCache volume currently only supports NFSv3 traffic, only NLM locking is supported at the FlexCache volume. When a client requests a file lock via NLM, that request is then transferred to the origin and the result is a FlexCache-aware NLM lock at the origin. In order to see the locks, you must look at them from the origin. The FlexCache cluster will have no information on 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               LIFE         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
```

Above shows that 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 volume 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 via the protocol standard. With NFSv3, this is accomplished through the sideband NSM protocol. Since the lock requests are transported through to the origin, there are no locks held at the FlexCache volume. This is confirmed by looking at the locks indicated at the FlexCache volume. There will not be any locks that ONTAP is tracking at the FlexCache volume. This means that any event at the FlexCache; takeover, giveback, LIF migrate; do not require any lock recovery.

The only time lock recovery is needed in ONTAP 9.5 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 just the same, but 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.

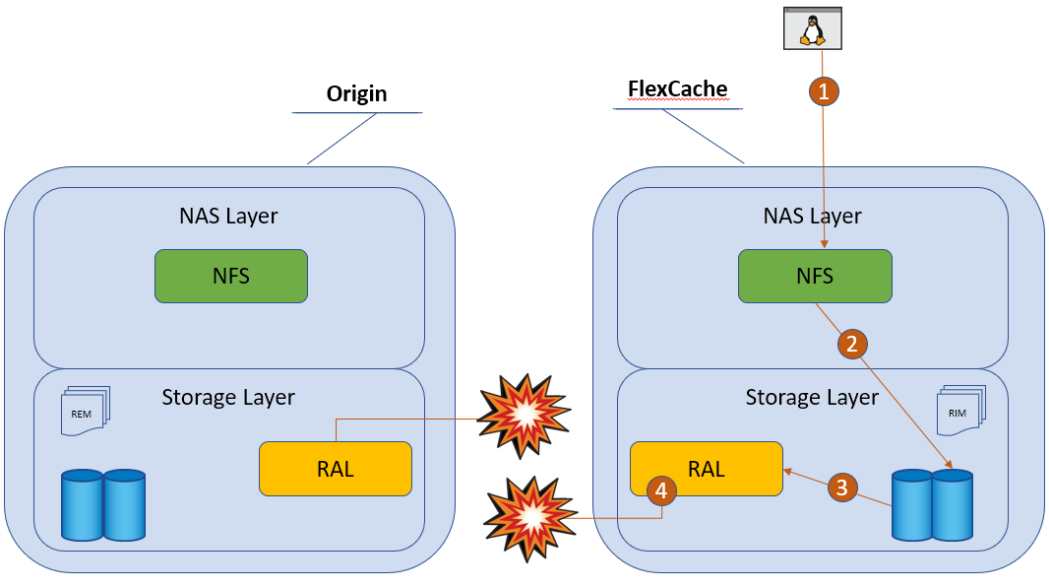
SMB and NFSv4 Locking at the Origin

SMB and NFSv4 are stateful protocols and therefore, 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 volume just the same as any other client. Since the FlexCache volume has, in essence, a read delegation when a file is cached at the FlexCache volume, that read delegation needs to be revoked. Upon locking, the origin will also reach out to any FlexCache volumes that have the particular file that is being locked, and will revoke the delegation. Because of this, the client will see the exact same result as if the access was straight to the origin.

4.6 Disconnected Mode

There are times when the cluster that houses the origin and the cluster that houses the FlexCache volume cannot talk with each other over the InterCluster LIFs due to WAN Issues or other problems. When this happens, this is called “disconnected mode”. Both the origin and the FlexCache volume can detect when there is a disconnection since there is bi-directional traffic. Disconnected mode can be in either direction; when any node cannot communicate with the other node in the Cluster peer, disconnected mode is the result.

Figure 10) FlexCache volume disconnected mode.



There is an EMS message generated at the FlexCache volume when this happens. This is a FlexCache - specific EMS message. There are EMS messages about the cluster and SVM peer relationships, obviously, but this one is specific to the FlexCache volume attempting to contact the origin. Below 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 the FlexCache volume when there are disconnects of a FlexCache volume. There are normal EMS messages about the cluster and SVM peering relationship problems.

What Can I Do While in Disconnected Mode?

There is very little that is restricted while operating in disconnected mode. Here is a quick rundown of what you can and cannot do while in disconnected mode. The below information assumes that all best practices outlined in this document have been followed and implemented.

At the Origin

- All reads will proceed as normal.
- All writes to new files will proceed as normal.
- Writes to existing files that have not yet been cached at a FlexCache volume will proceed as normal.
- Writes to files that have been delegated via REM to the disconnected FlexCache volume will not be allowed and will hang (NFS client will receive an EJUKE).

At the Disconnected FlexCache Volume

- Reads for data that is already cached will proceed as normal.
- Reads for data that has not been cached will hang (NFS client will receive an EJUKE).
- Writes to the FlexCache volume will hang (NFS client will receive an EJUKE).
- An “ls” will only work if there has been an “ls” or equivalent done before the disconnection. (A directory is just another inode. If it has been cached, it will be served. If it has not, it will not.)

At a Nonisconnected FlexCache Volume

While there is one disconnected FlexCache volume, other non-disconnected FlexCache volumes will operate as normal, with the same restrictions as outlined in the “At the origin” section.

READDIRPLUS, FlexGroup Volumes, and Disconnected Mode

In order to speed up “ls” commands to a FlexGroup volume, there was a change implemented into the FlexGroup code that bulk loads directory entries instead of loading them one at a time. This is called “fast-readdir”. Although this sped up directory listings in FlexGroup volumes, for FlexCache volume, this does not promote caching. When in disconnected mode, this change will prevent any “ls” from being run at the disconnected FlexCache volume. There is a way to revert this behavior to the previous process with a bootarg. This is the recommendation if the need to run “ls” or any dir-type operation on the FlexCache volume 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/FlexGroup volume 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.

Last Accessed Time

By default, last accessed time, or atime, is a property of a file that gets updated whenever there is a read. In essence, that means a read acts as a write. FlexCache volumes have atime tracking turned off on them at creation to prevent excessive writes back to the origin for this value every time a read at the FlexCache volume is performed. Since it is not turned off at the origin, this can prevent some access at the origin during disconnection.

- Files that have been cached at a FlexCache volume that is disconnected may not be able to be read at the origin
- “ls” command cannot be run at origin in certain cases

This is another reason to disable atime updates at the origin volume. In certain cases, disconnected mode can have an effect on access to the origin volume. Previously this was outlined in Best Practice 1: Set atime update on origin to false.

5 Counters and Statistics and Licensing

There are various counters and statistics that can help in the analysis of FlexCache volume operational efficiency and performance. These counters will assist in 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, and 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_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_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_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_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_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 Volume Misses

FlexCache volume misses can be found within the `workload_volume` object and in the `read_io_type` counter. This will tell you how many times a file was requested that was not cached in a particular 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 Volume Delays in Volume Workload Statistics

Statistics for a FlexCache volume delays are also available in the volume workload details. Example below:

```
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 volumes. This is in the 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

FlexCache licenses are both capacity and subscription based. AFF/FAS licenses come in 5TB increments while ONTAP Select licenses can be purchased in 1TB increments. There is also a 1 year or 3-year term on the licenses. ONTAP will alert via EMS when the total FlexCache size in the cluster reaches 80% of the licensed capacity. ONTAP will also alert.

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 will allow a 10% overage so as not to disrupt operation, 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 performance of serving already cached files at a FlexCache volume, please refer to TR-4571, NetApp FlexGroup Volume Best Practices and Implementation Guide. Since the FlexCache volume is a FlexGroup volume, there is only a negligible difference in the way the already cached files are served at the FlexCache volume and files served from a non-FlexCache FlexGroup volume. Any performance information for FlexCache is identical to what it is for a FlexGroup volume.

When the FlexCache volume is across a latent WAN, then performance is more sensitive during the first read of a file. This can be seen in the statistics mentioned before; `statistics show -object waflremote -counter fc_retrieve_hist -raw`. What this counter shows is the histogram of additional latency that FlexCache volume encountered by reading from the origin. This may be able to be reduced in a few ways.

First is the reduction of the actual network latency between the FlexCache cluster and the origin cluster. WAN optimizers may help, but it will be highly variable on whether or not it will actually reduce latency. Since this is not a public protocol, WAN optimizers may or may not be able to accelerate the traffic and results will vary.

Next is to pre-load the data. There is a bash command that will pre-load 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" \;
```

In most cases, the above command will also warm 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

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 on this, please refer to TR-4557: NetApp FlexGroup Volumes: A Technical Overview. A FlexCache volume has two options on how to create a FlexGroup, by listing aggregates for constituents (`-aggr-list`), or by letting FlexGroup volume autos select (`-auto-provision-as`). FlexGroup volumes recommend 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 4. This can be accomplished via only using a single aggregate in the -aggr-list option or combining the -aggr-list with the -aggr-list-multiplier to result in only 4 constituent volumes.

Best Practice 4: FlexCaches should have 4 constituents

Create the FlexCache with only the -aggr-list option so it creates 4 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 -junction_path /vol_fc_cache1
```

```
fc_cluster::> flexcache create -vserver fc-svm1 -volume vol_fc_cache1 -
origin-vserver vs_origin -origin-volume vol_fc_origin -aggr-list
aggr1_node1,aggr1_node2 -aggr-list-multiplier 2 -size 5TB -junction_path
/vol_fc_cache1
```

```
fc_cluster::> flexcache create -vserver fc-svm1 -volume vol_fc_cache1 -
origin-vserver origin-svm -origin-volume vol_fc_origin -aggr-list
aggr1_node1,aggr1_node2,aggr1_node3,aggr1_node4 -aggr-list-multiplier 1 -
size 5TB -junction_path /vol_fc_cache1t
```

7.2 Reads Versus Writes

The rule of thumb for A FlexCache volume is a read/write mix of at least 80% reads and 20% writes at the cache. This is due to the arite-around nature of FlexCache volumes. Writes will incur the penalty of the latency of forwarding the write operation to the origin. The FlexCache volume will certainly allow a higher write percentage, but it is not optimal for the way FlexCache volumes in ONTAP processes the write.

7.3 Cache Volume Size

What is the optimal cache volume size? That is highly dependent on the data. The “working set” is the amount of data that is to be used at the FlexCache volume for a particular “run” or “job”. This is how the a FlexCache volume should be sized. For example, if the origin volume has 1TB worth of data in it, but know that a particular job only needs 75GB worth of data, then the optimal size for the FlexCache volume would be 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 volume that is being read. There are some caveats to this and are outlined below.

The other method to determine optimal cache volume size is to just 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. This works in many cases when the working set is not clearly understood and then can use statistics, used sizes and other indicators to determine the overall optimality of the cache size.

Best Practice 5: Specifically define the cache size

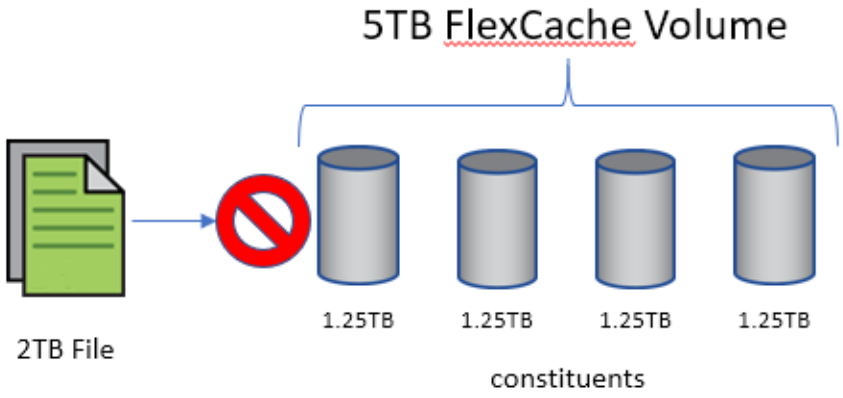
Always use the `-size` option for the FlexCache volume create to specify the FlexCache volume size.

File Sizes

Since the FlexCache is a FlexGroup volume, there are caveats to how large the FlexCache volume should be for optimal caching. This is due to how FlexGroup volumes create constituent volumes to create the FlexGroup container. For more information on FlexGroup volumes, please 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 will always be served from the origin. It will never be cached since 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 volume, if any file is over 1.25-1.8TB in size, then the file will not be cached.

Figure 11) File too large to be cached



The default FlexGroup volume configuration for a FlexCache volume is to create 4 constituents in a single aggregate. The `flexcache create` command is best run with the `-aggr-list` modifier and the default number of constituents created is 4 equal-sized volumes. This can be modified by using the `-aggr-list-multiplier` option. This changes the number of equal-sized volumes created for the FlexCache FlexGroup volume. Note that if the option is used with a number greater than 4, then the constituents will be smaller and the file sizes will still need to be accounted for.

Best Practice 6: Cache size should by 4X the largest file

Since the FlexCache is a FlexGroup volume, a single constituent should not be any smaller than the largest file that needs cached. There are 4 constituents by default, so the FlexCache size should be at least 4X the largest file that should be cached.

Note: If the constituent size is smaller than the file size being cached, ONTAP will still attempt to cache the file. This will result in evictions from the cache due to size.

Evictions

In ONTAP 9.5, there is only one way for files to get evicted on the cache. That is due to space constraints. This scrubber kicks in when any of the constituents are more than 90% full. ONTAP does have 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
```

This also means that during steady state, it will not be uncommon to always see the FlexCache volume at a high percentage used number. This is normal for steady-state and should not be considered a problem to see a FlexCache volume consistently at 80-90% used.

Autogrow

In some cases, autogrow may be a good option to use on the FlexCache volume to conserve space when you don't know what the working set size is going to be or there is a need to be overly conservative with the space on the FlexCache cluster.

In order to set autogrow on a FlexCache volume, the FlexCache volume needs to be created with an initial size. The recommendation is to set the initial FlexCache volume size to 1-5 % of the origin. In addition, the file sizes of the data need to be considered as they have even more impact when they are larger than the size of the constituent 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 and this means that any one file should not exceed 125 – 180GB in size in order to keep the optimal constituent sizes. These file sizes are far more likely to be present in the origin than the 1.25-1.8TB sizes mentioned before. This is why there would be more of an impact when using autogrow with the smaller cache and hence smaller constituent sizes. It is also recommended to set the maximum autogrow size to the 10-15% of the origin. This will keep the FlexCache volume from growing beyond that as it caches more and more data.

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

Autogrow only kicks in at a certain threshold. By default, this threshold is 85%. This means that when a particular constituent reaches 85% full, then it is grown to a specific number, calculated by ONTAP. Also, the eviction threshold is 90%. This means that if there is an ingest rate (first read from origin which writes it to cache) of > 5%, then there could be a case where there are grow/evict loops which result in undesirable behavior.

When autogrow does happen, there will be EMS messages. These messages will show what constituent and by how much the constituent grew. An example is below:

```
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.
```

7.4 LS Mirror Replacement

Load Sharing Mirrors for data volumes have been deprecated since ONTAP 9.1. FlexCache volume can be a replacement for the functionality that data LS mirrors provided to spread reads across multiple disks and nodes in a cluster. This is also the way to alleviate a hot volume use case. The only way this will be currently available to the clients is via automounter. NFS clients will use advanced automount map tricks to mount different volumes, be it one of the caches or the origin, in a fashion that will spread the connections over multiple LIFs and possibly SVMs. This does not evenly spread the load across multiple SVMs, nodes or volumes, but can certainly assist in the balancing of 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 via the following:

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

Create the FlexCache Volume

Next is to create the FlexCache volume. If the FlexCache volume is created in the same SVM as the origin, then no peering is necessary. If the FlexCache volume is being created in a different SVM, then the SVMs will need to be peered and the FlexCache volume application added to the peer. There is no real advantage to one method or the other, but there is a slight management advantage to having the FlexCache volume in a different SVM so that the mount point can be exactly the same across all FlexCaches volume 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's several different ways that the FlexCache volume 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 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" and 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 volume 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
```

Once complete, restart the autofs service and then the data will be available at `/mnt/data`. There is no guarantee on which SVM the mount from the client will land, 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 will end up being mounted.

7.5 Disclaimer

NetApp provides no representations or warranties regarding the accuracy, reliability, or serviceability of any information or recommendations provided in this publication, or with respect to any results that may be obtained by the use of the information or observance of any recommendations provided herein. The information in this document is distributed AS IS, and the use of this information or the implementation of any recommendations or techniques herein is a customer's responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. This document and the information contained herein may be used solely in connection with the NetApp products discussed in this document.

8 Troubleshooting

When there is trouble with the FlexCache volume, there are several places that need to be looked at. This section is specifically targeted for problems that have a cause related to the FlexCache volume.

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

Some additional commands to ensure there is proper communication over the IC (intercluster) LIFs are:

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

Files being served slowly: If there are files being served slowly, this is an indicator of the file being loaded from the origin, or a slow link to the origin node. This can be found with a combination of commands.

- Statistics show `waflremote fc_retrieve_ops` show an increase of operations that needed to go to the origin.
- Statistics show `waflremote fc_remote_latecy_histo` show 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 that is described in this document, review the following documents and/or websites:

- FlexCache Volumes for Faster Data Access Power Guide
  https://library.netapp.com/ecm/ecm_download_file/ECMLP2849754
- TR-4571: NetApp FlexGroup Volume Best Practices and Implementation Guide
  www.netapp.com/us/media/tr-4571.pdf
- TR-4557: NetApp FlexGroup Volumes A Technical Overview
  www.netapp.com/us/media/tr-4557.pdf
- NetApp Product Documentation
  https://docs.netapp.com

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.

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-4743-1218