An Introduction to FlexCache Volumes

Implementing File Caching on Filers

Rajesh Godbole | Network Appliance | May 2005 | TR 3399

TECHNICAL REPORT

Network Appliance, a pioneer and industry leader in data storage technology, helps organizations understand and meet complex technical challenges with advanced storage solutions and global data management strategies.

Abstract

This technical report (TR) provides an introduction to FlexCache, a feature that implements file caching in NetApp filers. The TR describes in detail what FlexCache volumes are, what business problem they solve and how they work. The benefits of using FlexCache are highlighted using deployment examples. Finally, the TR details FlexCache best practices for setup, sizing, scaling, and performance considerations.

Network Appliance Inc.

Table of Contents

1. Purpose and Scope

The purpose of this technical report is to provide an introduction to FlexCache, a feature of NetApp Data ONTAP™ that implements file caching. We will explain what FlexCache volumes are, describe how they work, and explain the benefits of caching by describing common deployment examples. Finally we will go over best practices for FlexCache configuration and sizing.

This technical report is targeted toward evaluators and implementers to help them understand the new capabilities in FlexCache and apply them to their organizations' storage environments.

2. Introduction

FlexCache is a new feature in Data ONTAP that allows organizations to reduce management and infrastructure costs by automatically replicating, storing, and serving data requested over NFS. The NFS caching model in FlexCache replaces the DNFS Appliance, which was previously available on the NetCache® platform.

In the caching model FlexCache volumes store the data that's most recently accessed from the origin. To ensure freshness,of cached data, FlexCache will reclaim space by ejecting less recently accessed data from the cache. Additionally, if the source volume's data is modified FlexCache fetches the modified data upon access to ensure cache consistency. The FlexCache volume can therefore perform optimally when the source volume does not change very often.

In compute farm environments, where the file server becomes a bottleneck, FlexCache can be used to increase overall performance while keeping administration costs to a minimum. Typically, computational capacity is increased by adding new clients to the compute farm to the point where file server throughput becomes the limiting factor.

For employees located at remote offices, WAN-associated latencies and bandwidth constraints often lead to the deployment of additional servers at the remote site. Additional servers offer LAN-like access to remote employees but inevitably lead to increased management and infrastructure costs in the form of complex data replication processes and hardware. FlexCache alleviates much of this burden by caching and serving only the requested portion of a data file, while simultaneously maintaining data consistency with the origin file server.

3. What Are FlexCache Volumes?

FlexCache volumes are sparsely populated volumes on a local (caching) NetApp filer that is backed by a volume on a different, possibly remote (origin), NetApp filer. When data is requested from the FlexCache volume, it is read through the network from the origin system and cached on the FlexCache volume. Subsequent requests for that data are then served directly from the FlexCache volume. In this way, clients in remote locations are provided with direct access to cached data. This improves performance when data is accessed repeatedly, because after the first request, the data no longer has to travel across the network.

The initial release of FlexCache supports read caching of NFS. FlexCache volumes are writable, and updates to a FlexCache volume are written through to the origin volume. The caching filer as well as the origin filer must run Data ONTAP 7.0.1, the release that supports this feature, or later. Data ONTAP 7G introduced FlexVol™, which allows creation of virtualized logical containers for more flexible storage allocation. The following terms are used throughout the paper and are important for understanding Data ONTAP 7G because they relate to FlexCache volumes:

Aggregate. A physical container of disks from which logical volumes and RAID groups can be created.

Traditional volume. A single physical volume that is associated with a collection of disks and managed by a single administrative unit.

Flexible volume. A logical volume available in Data ONTAP 7G that resides within an aggregate and can grow or decrease in size. It is constrained only by the soft limits set when created and the hard limits of the aggregate. It is

the enabling technology for all other flexible features.

FlexCache volumes are special FlexVol volumes that are purpose-built for caching files from another filer. A FlexCache volume maps to a single volume on the origin server. The origin volume can be either a FlexVol volume or traditional volume. The caching filer implements the data protection and business continuance features of a filer, including clustered failover (CFO), allowing for a more redundant design without a single point of failure.

3.1. FlexCache, FlexVol, and Traditional Volume

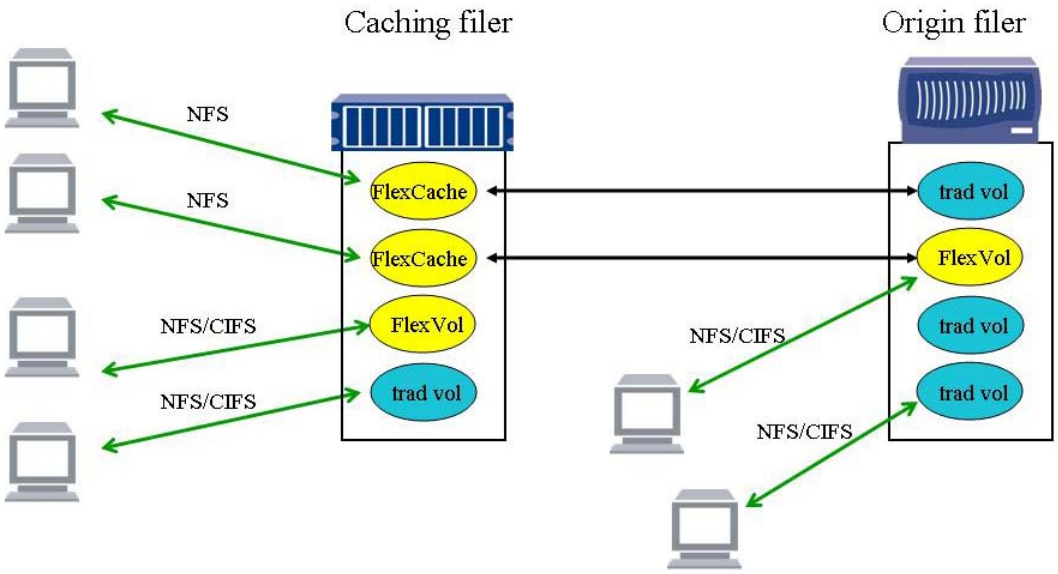

Figure 1) What are FlexCache volumes?

FlexCache volumes, FlexVol volumes, and traditional volumes can coexist on the caching filer. The type of volume and its properties will determine how space is managed within the aggregate. Figure 1 shows two FlexCache volumes, one backed by a traditional volume and the other by a FlexVol volume on an origin filer. In addition to the two FlexCache volumes, the caching filer also hosts a FlexVol volume as well as a traditional volume.

3.2. Benefits of FlexCache Volumes

Management and infrastructure costs can be reduced by deploying FlexCache volumes at remote offices and in compute farm environments. In the first release of FlexCache, NFS read requests are cached and write-through requests are proxied back to the origin filer. Deploying FlexCache provides the following benefits:

Reduced administration costs:
- No need to manage replication or synchronization processes
- Fewer servers and storage devices to manage

Faster time-to-market:
- New data is always accessible; no need to wait for next replication run
- Faster overall computation times in compute farms

Lower infrastructure costs:
- Lower bandwidth costs because only the data requested by users gets transferred
- Lower server and storage costs because only the data that is needed gets stored

4. Where Can FlexCache Volumes Be Used?

4.1. Acceleration of Compute Farms

One of the emerging trends in the technical computing marketplace is the proliferation of compute farms. The primary driver of this trend is the decrease in the price/performance of the PC-based computers through commoditization. Many of the high-end mainframe, supercomputer, or proprietary high-end SMP configurations are being complemented, or in some cases replaced, by small 1U/2U PCs, usually running a flavor of Linux®. For a perspective on this topic, see TR 3385- Distributed Storage for a Scalable Grid Infrastructure. This proliferation of compute farms can cause hot spots in storage that serves application datasets.

One of the primary benefits of using FlexCache volumes is their ability to create a copy of the data that is accessed most frequently. For applications workloads that are mostly read oriented, this ability of FlexCache can be used for offloading read operations from a filer serving application data to compute farms. In this configuration, the overall throughput of the compute farm accelerates because of the increased read bandwidth. Examples of applications that are mostly read intensive, and could therefore benefit from read offloading are Semiconductor Tapeout, Seismologic Analysis, and Rendering.

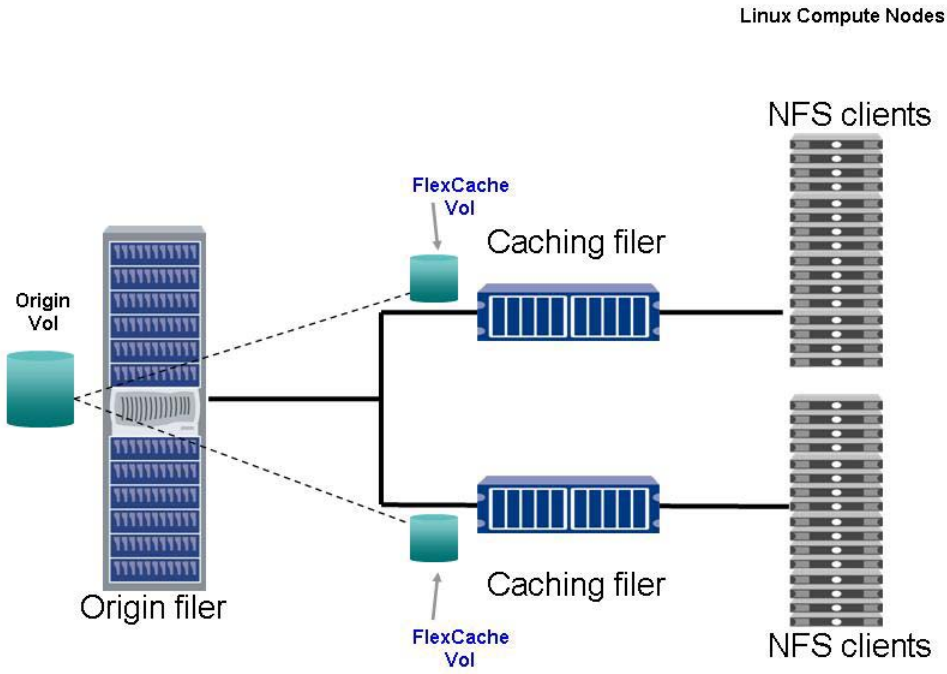

Figure 3) Accelerating compute farms: read offloading using FlexCache.

Figure 3 shows clients in a compute farm deployment accessing the application dataset over NFS from a filer. The application dataset can consist of software, reference libraries, and the application data. FlexCache can be configured in this environment in such a way that groups of NFS clients access cached copies of the entire application dataset.

Whenever an NFS client tries to access a reference library, the caching filer will fetch it from the origin and cache it locally. Subsequent accesses to this library will be from the local cache. Writes to any application data are written through the origin, making this deployment transparent to the application.

This approach reduces load on the server and at the same time scales up server throughput with no reduction in client access latency.

4.2. Software Distribution to Remote Locations

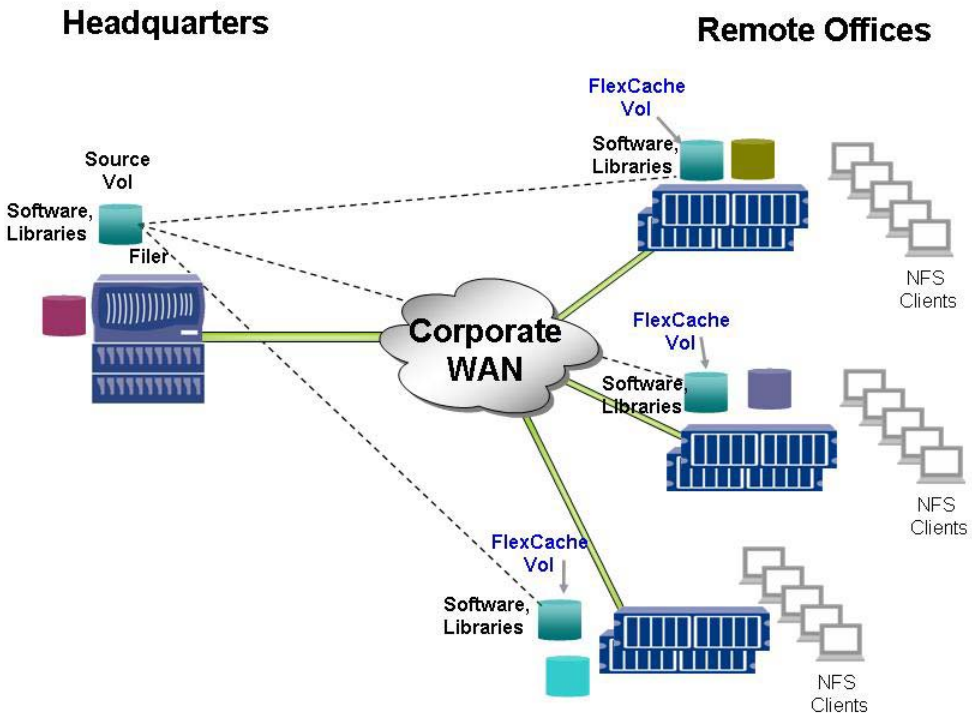

Figure 4) Software distribution to remote offices: reduce latency using FlexCache.

In a distributed global enterprise, a common problem is associated with the distribution of software uniformly and consistently throughout the enterprise. In the UNIX® world, software is often made available at a central location or locations by sharing the central location over NFS. Software distribution servers are often used in a tree topology, and the software distribution and libraries are replicated to the software distribution servers in the remote locations using tools such as `rsync` or `rdist`.

This approach to software distribution can be expensive in terms of infrastructure, productivity, and administration costs. Since all of the data gets replicated from the central server to remote locations independent of needs, the cost of bandwidth to the remote offices as well as remote storage needs can be unnecessarily high. In terms of productivity, any data that is not replicated will not be available. Additionally, data may not be available between replication windows. Administrative costs of maintaining the scripts and monitoring replication cron jobs can be high. Moreover the storage for the remote software distribution servers may require occasional management .

FlexCache can be used to simplify the remote software distribution within enterprises. In a typical remote office deployment, caching filers can be used as software distribution servers in the remote office. In a remote office configuration, a caching filer sits near the edge of the network or as close as possible to the remote office. Client requests are configured to explicitly mount the caching filer instead of the origin server. The caching filer will pull down only files that are requested by clients at the remote office and serve them locally on subsequent requests. This approach ensures that bandwidth to and the storage at the remote location is efficiently used. Additionally, local access to remote data significantly reduces the latency of access. Data in the local cache is immediately available as soon as the requested file or directory is pulled into the local cache, and the clients do not have to wait for bulk transfers to finish as described in the push model above. Since the FlexCache volumes can manage space allocation very efficiently based on the caching filer's space management feature, administration costs for software distribution can be significantly reduced. The clustered failover (CFO) feature of filers ensures that the distribute software is highly available at the remote site, where administrative resources may be scarce.

5. How Do FlexCache Volumes Work?

A newly created FlexCache volume starts out empty and is populated as responses are stored from client requests. Only the data that has been requested by a user is cached. Cached objects remain on the device as long as they are deemed consistent with the origin server and are requested by the end user.

5.1. Caching Granularity

A FlexCache volume can cache files, directories, and symbolic links. Caching is performed at a 4K block level granularity, and invalidation is performed at the file level. This means that a client doesn't have to request an entire file for FlexCache to cache it—only the 4K blocks accessed by the client will be cached. The immediate consequences of this specificity are efficient usage of disk space and the maximization of hits on "hot" or frequently requested objects. Whenever a file attribute is altered on the source file server, the entire file will need to be revalidated. Responses to directory-related requests, such as READDIR and READDIRPLUS, are cached by FlexCache. Subsequent requests for LOOKUPS, READDIR, and READDIRPLUS, are then served as hits to the client.

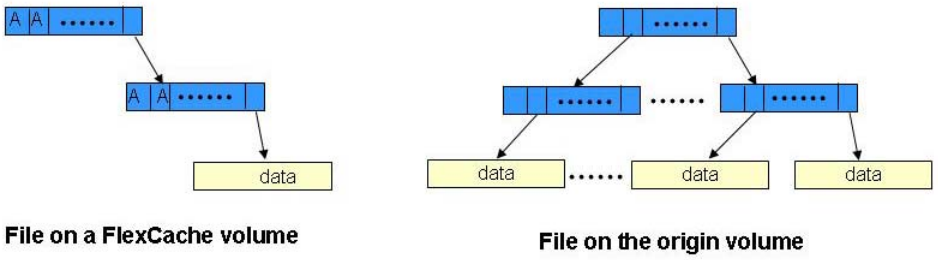

Figure 2) Sparse files.

5.2. Cache Consistency

Data is cached/verified through client read requests (e.g., READ, LOOKUP, READDIR) and invalidated through client modifying requests (e.g., WRITE, REMOVE). Cache consistency for FlexCache volumes is achieved using a combination of attribute cache timeouts, delegations and invalidation of cached objects while proxying writes.

5.2.1. Attribute Cache Timeouts

When data is retrieved from the origin volume, the file that contains that data is considered current in the FlexCache volume for a specified length of time, called the attribute cache timeout. During that time, if a client reads from that object and the requested data blocks are cached, the read request is fulfilled without any access to the origin volume. If a client requests data from a file for which the attribute cache timeout has been exceeded, the FlexCache volume verifies that the attributes of the file have not changed on the origin system. Then one of the following actions is taken:

- If the attributes of the file have not changed since the file was cached, the requested data is either directly returned to the client (if it was already in the FlexCache volume) or retrieved from the origin system and then returned to the client.
- If the attributes of the file have changed, the file is marked as invalid in the cache. Then the requested data blocks are read from the origin system as if it were the first time that file had been accessed from that FlexCache volume.

With attribute cache timeouts, clients can get stale data if they access a file on the FlexCache volume that has been changed on the origin volume, and the access is made before that file's attribute cache timeout is reached. To prevent clients from ever getting stale data, you can set the

attribute cache timeout to zero. However, this will negatively affect your caching performance, because then every data request causes an access to the origin system.

5.2.2. Delegation

When a request to the FlexCache volume results in a miss, the caching filer contacts the origin to fetch a fresh copy. The Caching Filer may also request a read delegation. A read delegation on a file is a guarantee that no other clients are writing to the file. Once a file delegation is granted, the file cache is able to trust that it has the latest copy of the file and its attributes, and it can return portions of the file and attributes to clients without going back to the origin server for freshness checking. This mechanism reduces revalidation requirements between the cache and the origin and is more efficient than having to poll the file for freshness.

5.2.3. Write Operation Proxy

If the client modifies the file, that operation is proxied through to the origin filer, and the file is ejected from the cache. This also changes the attributes of the file on the origin volume, so any other FlexCache volume that has that data cached will re-request the data once the attribute cache timeout is reached and a client requests that data.

5.2.4. Cache Hits and Misses

When a client makes a read request, if the relevant block is cached in the FlexCache volume, the data is read directly from the FlexCache volume. This is called a cache hit. Cache hits are the result of a previous request.

A cache hit can be one of the following types:

- Hit—The requested data is cached and no verify is required; the request is fulfilled locally and no access to the origin filer is made.

- Hit-Verify—The requested data is cached but the verification timeout has been exceeded, so the file attributes are verified against the origin system. No data is requested from the origin system. In the case of a file being delegated, a file need not be verified, and hence there is no user visible latency.

If data is requested that is not currently on the FlexCache volume, or if that data has changed since it was cached, the caching system loads the data from the origin system and then returns it to the requesting client. This is called a cache miss.

A cache miss can be one of the following types:

- Miss—The requested data is not in the cache; it is read from the origin system and cached.

- Miss-Verify—The requested data is cached, but the file attributes have changed since the file was cached; the file is ejected from the cache and the requested data is read from the origin system and cached.

5.3. Space Management

Space Management in FlexCache is designed to ensure that that space within an aggregate is efficiently and optimally allocated across multiple FlexCache volumes and FlexVols. During the normal course of operation, objects in the cache that are not frequently used are ejected in favor of objects that are accessed most often. At the same time, space guarantees of FlexVols are honored. Because of this design, very little administrative intervention is required once FlexCache volumes are created within an aggregate.

5.4. File Locking

Locking is critical in maintaining the consistency of files that can be modified by multiple clients over NFS. FlexCache uses a distributed Network Lock Manager (NLM) to ensure that multiple processes don't simultaneously modify a file, as well as coordinate the modification of a shared file between cooperating processes. FlexCache acts as a proxy on behalf of the NLM server on the origin filer, while all locks are held by the origin server. Since

NLM is a stateful protocol, it is imperative that clients can reestablish locks if the servers go down and that these servers know to release client locks if the client reboots. Both of these issues are solved by the Network Status Monitor (NSM). The network status monitor notifies NLM about the current state of the network, including any system crashes that occur. NLM can then reestablish any of the stale sessions. FlexCache offers support for all NLM related requests, as well as NSM feedback.

5.5. Requirements of FlexCache

The following requirements and limitations apply to the first release of FlexCache.

A FlexCache volume must always be a FlexVol volume. FlexCache volumes can be created in the same aggregate as regular FlexVol volumes.

The origin volume can be a FlexVol volume or a traditional volume; it can also be a SnapLock™ volume. The origin volume cannot be a FlexCache volume itself, nor can it be a qtree.

Both the caching and origin filers must run Data ONTAP 7.0.1 or later in order to support the full feature set of FlexCache as well as the back-end protocol between the caching and origin filers. They can be upgraded independently of each other as long as both of them are running Data ONTAP 7.0.1 at a minimum.

FlexCache requires NFS v2 and/or v3 clients .

6. FlexCache Best Practices

6.1. Setting Up Volumes

The following steps are required to set up a FlexCache volume. In this example `heckle` is the caching filer and `split` is the origin filer.

Enable the FlexCache license on the caching filer

```
heckle> license add XXXXXXX
A flex_cache license has been installed.
 FlexCache enabled.
heckle> Mon Apr 11 15:38:37 PDT [rc:notice]: flex_cache licensed.
```

Enable the FlexCache on the origin filer

```
split> options flexcache.enable on
```

Allow the caching filer (heckle) to access the origin filer (split)

```
split> options flexcache.access host=heckle
```

Create the FlexCache volume using the syntax `vol create volname aggrname size -S remotehost:remotevolume`. The size of the FlexCache volume should be equal to the size of the hosting aggregate aggr0.

```
heckle> df -A aggr0
Aggregate               kbytes       used      avail capacity
aggr0                 654691532   10536236  644155296       2%
aggr0/.snapshot        34457448     796384   33661064       2%
heckle> vol create vol1cache aggr0 654691532k -S split:vol1
Creation of volume 'vol1cache' with size 654691532k on containing aggregate
'aggr0' has completed.
heckle> df -L vol1cache
Filesystem              kbytes       used      avail capacity  Mounted on
/vol/vol1cache/      654691532       1424  644153848       2%  /vol/vol1cache/
/vol/vol1cache/.snapshot          0          0          0     ---%  /vol/vol1cac
he/.snapshot
heckle> vol create vol2cache aggr0 654691532k -S brain:vol2
Creation of volume 'vol2cache' with size 654691532k on containing aggregate
'aggr0' has completed.
heckle> df -L vol2cache
Filesystem              kbytes       used      avail capacity  Mounted on
/vol/vol2cache/      654691532       1424  644051424       2%  /vol/vol2cache/
/vol/vol2cache/.snapshot          0          0          0     ---%  /vol/vol2cac
he/.snapshot
```

6.2. Migrating from DNFS

The following example shows how one would create a FlexCache mapping from a DNFS export:

The DNFS export:

```
config.nfs.exports = \\
 /vol/toaster/data/project toaster:/vol/data/project hard,max-age=1m,ro
 \\
```

Would become

```
vol create toaster_data <aggr> <size> -S toaster:data

vol options toaster_data actimeo 1m

exportfs -p actual=/vol/toaster_data/project,ro /vol/toaster/data/project
```

6.3. Sizing

When picking a filer appropriate for caching, two parameters are most important: the size of the most actively used set of data that the caching filer will serve and the performance of the filer head serving the active data set.

FlexCache volumes can be smaller than their origin volumes. However, making the FlexCache volume too small can negatively affect the caching performance. When the FlexCache volume begins to fill up, it flushes old data to make room for newly requested data. When that old data is requested again, it must be retrieved from the origin volume

The recommended best practice is to keep all FlexCache volumes set to the size of the containing aggregate. For example, if you have two FlexCache volumes sharing a single 2TB aggregate, set the size of both FlexCache volumes to 2TB. All FlexCache volumes are over-provisioned and will manage shared storage space to accelerate the client workload on both volumes. The aggregate should be provisioned with sufficient storage space to hold the clients' working set on both volumes.

The performance characteristics of a caching filer are similar to the performance of the same filer serving an NFS read workload, minus a small amount of resources required for the FlexCache implementation.

6.3.1. Overprovisioning and Space Management

The best practice is to overprovision FlexCache volumes and allow the caching filer to manage their space. Overprovisioning allows the creation of a large enough container for cached data without dedicating all of the physical space only to the caching volume. Overprovisioning ensures that the available physical space within an aggregate is more efficiently shared between the FlexCache and FlexVol volumes that the aggregate shares. For multiple FlexCache volumes within an aggregate, the caching filer’s space management feature will evenly expire objects from within all the volumes.

6.3.2. Administration of FlexCache Sizes

Once a FlexCache volume is created using recommended best practices, the aggregate’s space reclamation feature will manage the space allocated to it. It should not be necessary to manually change the size of the FlexCache volume as long as the aggregate’s size stays unchanged.

6.3.3. FlexVol and FlexCache Volumes in the Same Aggregate

If you have regular FlexVol volumes in the same aggregate as your FlexCache volumes, and you start to fill up the aggregate, the FlexCache volumes can lose some of their unreserved space if they aren’t currently using it. In this case, when the FlexCache volume needs to fetch a new data block and it does not have enough free space to accommodate it, a data block must be ejected from one of the FlexCache volumes to make room for the new data block.

If this situation causes too many cache misses, you can add more space to your aggregate or move some of your data to another aggregate.

6.4. Scaling

6.4.1. Pod-based Approach to Accelerate Compute Farms

Most often, Linux servers are configured in racks or cabinets in a data center. Small groups of Linux servers in racks are combined into discrete network segments that have an uplink to a backbone. The filers are also connected to this backbone, typically with multiple Gigabit Ethernet network cards to provide adequate bandwidth.

This architecture has an obvious limitation: the entire farm is limited to the throughput of a single filer. Although filers normally do their jobs well, there is a point of saturation when hundreds or even thousands of clients are requesting the same data. The filer will encounter a CPU and/or disk limitation, so that when more clients are added to the farm, system throughput does not increase and may even decrease. This hot spot–bound scenario can be addressed by deploying a NetApp solution. A possible architecture is shown in Figure 6.

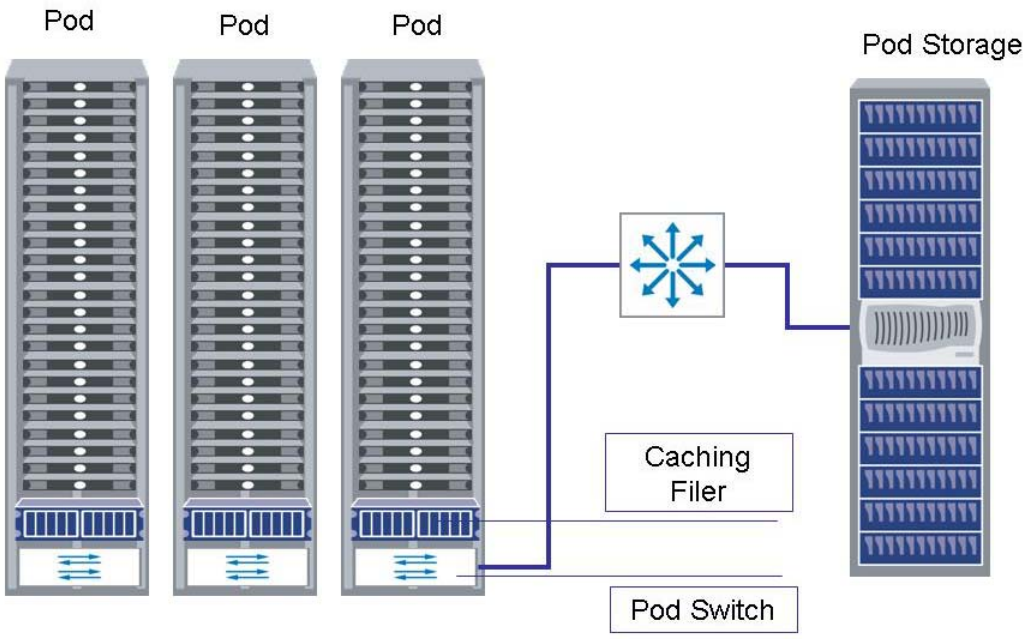

Figure 6) A pod-based approach for scaling Linux farms.

In the revised architecture, each pod still contains a certain number of Linux hosts and a dedicated switch. Now each pod also contains one caching filer with FlexCache volumes. The caching filer provides a high-speed local copy of the most requested input files for each pod so that the origin filer can be offloaded from a large proportion of the read requests of the farm. These requests are now served by a pod's local filer, without exchanging traffic with the origin filer.

By adding to each pod a caching filer with its own FlexCache volumes in heavy read environments such as this one, the farm has much more read bandwidth available to the application in aggregate. When the need for compute power grows, a new pod with compute nodes, a caching filer, and its dedicated switch can be brought online. This scaling methodology ensures that the available NFS read bandwidth scales far beyond the capabilities of a single filer.

6.5. Performance Considerations

6.5.1. Viewing the FlexCache Configuration

For FlexCache related configuration use `vol status –v`. The highlighted text shows FlexCache related configuration details. In this case the volume `vol1cache` is a FlexCache volume and its origin filer/volume are `split:vol1`.

```
heckle> vol status vol1cache -v
 Volume State Status Options Source
 vol1cache online raid_dp, flex nosnap=off, nosnapdir=off, split:vol1
 flexcache minra=off,
 no_atime_update=off,
 nvfail=off,
 snapmirrored=off,
 create_ucode=off,
 convert_ucode=off,
 maxdirsize=31457,
 fs_size_fixed=off,
 svo_enable=off,
 svo_checksum=off,
 svo_allow_rman=off,
 svo_reject_errors=off,
 fractional_reserve=100,
 flexcache_min_reserve=100m,
 actimeo=30, acregmax=30,
 acdirmax=30, acsymmax=30
 Containing aggregate: 'aggr0'

 Plex /aggr0/plex0: online, normal, active
 RAID group /aggr0/plex0/rg0: normal
```

6.5.2. Performance Benefits of Fitting Entire Dataset in Memory

Objects within a FlexCache volume can be ejected during normal operation if the size of the active data set exceeds the limits of the cache because of the caching filer's space management feature. Client requests that result in a cache hit are served from the caching filer disk, and, if possible, from the memory of the filer. Therefore a FlexCache volume will perform most optimally when the most actively accessed subset of data can fit in the memory of the filer hosting the volume. In this case when the data is being served out of caching filer memory, the caching filer will most likely be network bound.

The next base case is when the active dataset does not fit in filer memory and is served from cache disk. In this case the caching filer is disk bound.

The estimation of the size of the most actively accessed dataset is therefore an important performance consideration. The determination of the model for caching filer and associated amount of attached disk can be made based on this estimation.

6.5.3. How Do I Set the Time-to-Live (TTL) for Cached Objects?

The following volume-level options can be used to set the TTL

Vol option	Explanation	Default value
`acregmax`	Attribute Cache regular file timeout; the amount in time (s) files are valid before consulting the origin	30s
`acdirmax`	Attribute Cache directory timeout; the amount in time (s) directories are valid before consulting the origin	30s
`acsymmax`	Attribute Cache symbolic link timeout; the amount in time (s) symbolic links are valid before consulting the origin	30s
`actimeo`	Attribute Cache regular file timeout; the amount in time (s) files are valid before consulting the origin	30s
`flexcache_min_reserve`	Minimum space guaranteed in the aggregate for the given FlexCache volume	100MB

Table 1) FlexCache per volume TTL settings.

For many workloads, the working dataset does not change at all. An example of such a workload is rendering applications, where data is read from one file and the contents of that file do not change. After processing, writes are made to another file.

Caching can be implemented more effectively for such applications by setting the value of the appropriate ac* options to a large value, such as 60m, to increase the TTL of read-only cache objects.

6.5.4. How Do I See FlexCache-related Stats?

The following table shows caching related statistics for both the caching and the origin filer.

Statistic	Command on a FlexVol	Explanation
Volume size	`df`	Reports volume size of the origin volume
	`df -L`	Reports volume size of the local cache
Cache hit rate, bandwidth savings	`flexcache stats -C`	On the caching filer, displays client side bandwidth savings and hit/miss stats for the volume
NFS cache stats	`nfsstat -C`	On the caching filer, reports NFS cache hit rate
Cached volume(s)	`flexcache stats -S -c`	On the origin filer, shows which volumes are being cached by what caching filer `flexcache.per_client_stats` option must also be set to `on`

Table 2) FlexCache statistics.

6.5.5. TCP Window Size Optimization for WAN Traffic

One of the most relevant settings for TCP/IP communications over WAN is the TCP window. A TCP window is the amount of outstanding (unacknowledged by the recipient) data a sender can send on a particular connection before it gets an acknowledgment back from the receiver.

The TCP window size for FlexCache volumes can be set by using the command

```
options sparse.tcp_windowsize
```

6.5.6. How to Tell Reads from Memory vs. Reads from Disk on the Caching Filer

The `sysstat` command on the caching filer will report resource utilization.

```
heckle> sysstat -c 10 -s 1
 CPU NFS CIFS HTTP Net kB/s Disk kB/s Tape kB/s Cache
 in out read write read write age
 12% 2784 0 0 585 474 0 0 0 0 >60
 12% 2799 0 0 588 476 0 0 0 0 >60
 12% 2822 0 0 593 480 0 0 0 0 >60
 12% 2825 0 0 593 481 0 0 0 0 >60
 12% 2773 0 0 582 472 0 0 0 0 >60
 12% 2853 0 0 599 485 0 0 0 0 >60
 12% 2814 0 0 591 479 0 0 0 0 >60
 12% 2797 0 0 588 476 0 0 0 0 >60
 12% 2847 0 0 598 484 0 0 0 0 >60
 13% 2814 0 0 592 479 516 452 0 0 >60
--
Summary Statistics ( 10 samples 1 secs/sample)
 CPU NFS CIFS HTTP Net kB/s Disk kB/s Tape kB/s Cache
 in out read write read write age
Min
 12% 2773 0 0 582 472 0 0 0 0 >60
Avg
```

```
 12% 2812 0 0 590 478 51 45 0 0 >60
Max
 13% 2853 0 0 599 485 516 452 0 0 >60
```

We can see NFS traffic going out the wire with hardly any disk I/O, with the cache age being >60. In this case the caching filer is serving files out of memory.

If the disk I/O traffic is more significant and the cache age is smaller, files are being served from disk.

7. Conclusion

To enhance the productivity of workers in remote locations, organizations with a geographically distributed workforce must ensure that remote workers can access and share mission-critical data quickly and efficiently. However, limited wide area network (WAN) bandwidth between the central office and remote locations can make that difficult. Remote users often experience poor performance in receiving large files for applications, such as CAD/CAM or software development, and in accessing data repositories across the WAN (which must be shared to support remote office operations). Additionally, IT support at remote sites is often limited, complicating the management of operations such as backup and data replication. The Network Appliance™ FlexCache solution provides centralized IT administration of data while delivering improved data access to remote workers. In high-performance computing environments, such as movie postproduction, bio-informatics, and simulation, FlexCache can be used to scale overall performance without adding management overhead.

NetApp®

Network Appliance, Inc.
495 East Java Drive
Sunnyvale, CA 94089
www.netapp.com