Technical Report

Flash Cache Best Practice Guide

Skip Shapiro, NetApp
November 2013 | TR-3832

Abstract

NetApp® Flash Cache™ and Flash Cache 2 cards, and caching software embedded in the Data ONTAP® operating system, enable better performance with NetApp storage systems. This guide describes how Flash Cache and Flash Cache 2 work, provides essential information for successful implementation, and explains how to measure the effectiveness of Flash Cache in a deployed system.

TABLE OF CONTENTS

LIST OF FIGURES

1 Overview

NetApp Flash Cache and Flash Cache 2 (referred to collectively in this report as Flash Cache, except where noted) integrate caching software with flash-based PCI-Express (PCIe) cards to improve storage system performance. The caching software is embedded in the Data ONTAP® operating system, and both generations of Flash Cache cards – Flash Cache and Flash Cache 2 – use the same software.

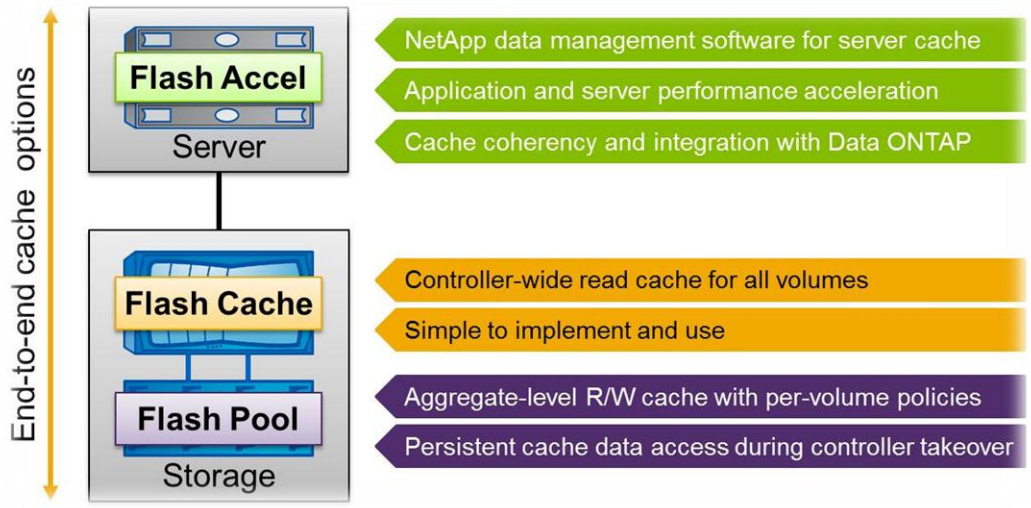
Flash Cache offers millisecond access to cached read data, which is 8 to 20 times faster than typical read latencies from hard disk drives (HDDs).

Flash Cache cards are available in capacities of 256GB, 512GB, and 1TB, and Flash Cache 2 cards are available in capacities of 512GB, 1TB, and 2TB. Multiple cards can be installed in some storage controller models, with the capacity of all cards operating as a single shared cache resource.

1.1 NetApp Virtual Storage Tier

The NetApp Virtual Storage Tier (VST) is a portfolio of flash-based products for Data ONTAP systems that collectively provide end-to-end data caching from server to storage. VST includes server caching software – Flash Accel™ – and two complementary storage caching options: Flash Cache and Flash Pool™. This technical report focuses on Flash Cache; other similar reports cover Flash Pool and Flash Accel.

Figure 1) The NetApp VST product family.



1.2 The Benefits of Using Flash Cache

Flash Cache and Flash Cache 2 enable data (application data and storage system metadata) that is initially read from HDDs to be retained in cache longer so that subsequent read requests for the same data are served from low-latency cache instead being retrieved again from HDDs, which respond slower than cache does. Implementing Flash Cache on a system can provide one or more of the following benefits:

- **Fast read response times for large application datasets.** NetApp systems configured with Flash Cache can cache up to 85 times more data than systems that have no supplemental cache, and data can be read from cache 2 to 10 times faster than from HDDs.

- **Providing more HDD operations for other workloads.** By serving repetitive read requests from cache, Flash Cache offloads those operations from HDDs, enabling the disk drives to handle other read and write requests.
- **Increased system throughput (IOPS).** For systems where throughput is constrained due to high HDD utilization, Flash Cache can increase total system IOPS by serving repetitive reads from cache and allowing the HDDs to handle other requests.
- **HDD reduction.** A NetApp storage system configured with Flash Cache to support a specified set of workloads typically requires fewer of the same type (rpm) of HDDs than a system that is configured without supplemental cache. Flash Cache also often enables systems to be configured with slower, lower-cost-per-TB HDDs. The result of configuring a storage system with Flash Cache and fewer – and sometimes slower – HDDs is a lower overall system cost than a system that is configured only with HDDs.
- **Lower data center operating expense.** A storage system configured with Flash Cache and fewer or slower HDDs often consumes less data center rack space, power, and cooling than a comparable system that is configured with HDDs only and no Flash Cache.

As just explained, configuring a NetApp storage system with Flash Cache can provide significant performance and/or storage efficiency benefits. However, there are a few things that Flash Cache does not do:

- **Reduce or alleviate high system memory or CPU utilization.** Flash Cache uses system memory and CPU cycles to manage the cache; therefore it increases rather than reduces utilization of these resources. Consequently, a system must have sufficient memory and CPU cycles available for Flash Cache to provide any benefit.
- **Increase the maximum throughput capability of a storage controller.** Achieving the maximum throughput (IOPS or MB/sec) of a system is a function of the memory and CPU resources of the storage controller. Caching technologies do not increase the system memory or CPU cycles available in a system. As a result, the maximum throughput capability of a NetApp storage system is not higher when configured with Flash Cache as compared to a configuration with HDDs only.
- **Accelerate write operations.** Flash Cache is a read cache; it accelerates repetitive read operations. The Data ONTAP operating system is write-optimized through the use of in-memory write cache and nonvolatile memory (NVRAM or NVMEM), which enable fast response to incoming write operations.

2 How Flash Cache Works

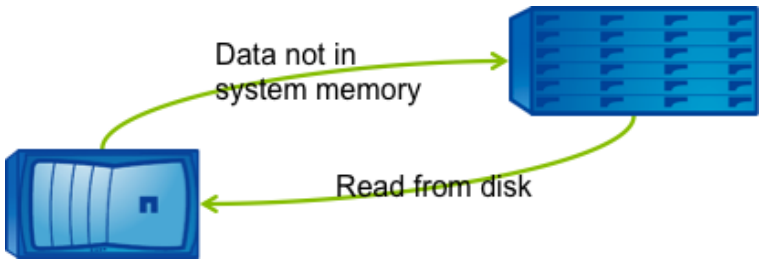
Flash Cache provides performance and storage efficiency benefits by augmenting the native caching capability of Data ONTAP systems. Data ONTAP systems cache data for read operations in controller memory buffers. However, the ratio of controller memory to the total amount of data stored on most systems is typically very low (5% or lower); only a small fraction of stored data can be cached in memory buffers. Adding one or more Flash Cache cards to a system controller can dramatically increase the amount of data that can be retained in cache, thereby enabling fast response times for more read requests.

The rest of this section explains how read and caching operations with Flash Cache work.

2.1 Data ONTAP Disk Read Operation

When a read request arrives and the requested data is not already held in controller memory or Flash Cache cache, the data is retrieved from disk and placed into memory buffers. The data is then forwarded to the requesting host or client. Figure 2 shows this operation for a system that does not have Flash Cache cards installed.

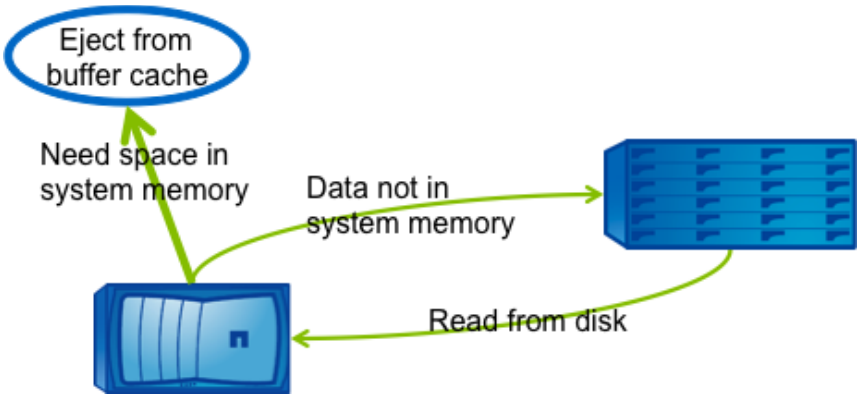
Figure 2) A read on a Data ONTAP storage system that is not configured with Flash Cache.



2.2 Data ONTAP Clearing Space in System Memory for More Data

As more data is inserted into memory from read requests for other data, eventually all memory buffers become full. Data ONTAP determines which data is the least valuable to retain in memory, and that data is evicted from memory buffers to make room for new data that is being read from disk.

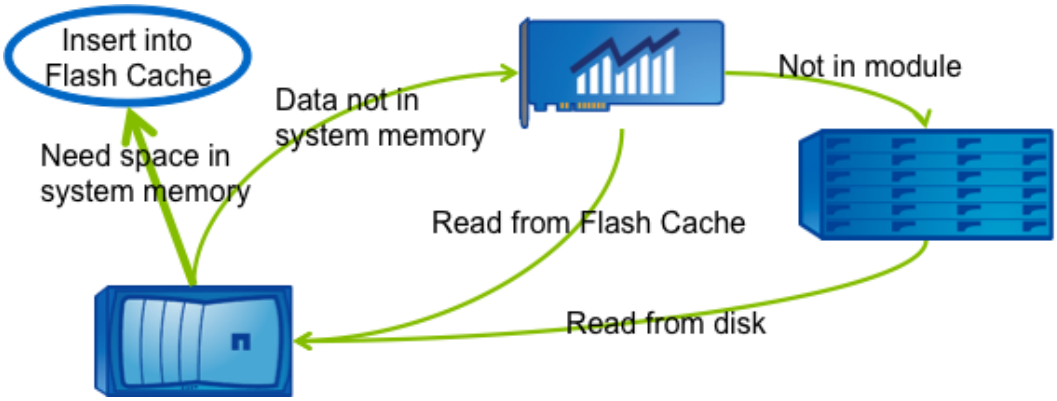
Figure 3) Clearing memory on a Data ONTAP storage system that is not configured with Flash Cache.



2.3 Retaining Active Data in Flash Cache

On a system that is configured with Flash Cache, data that is designated for eviction from controller memory buffers is inserted into Flash Cache cache before it is evicted from memory.

Figure 4) Inserting data into a Flash Cache cache.

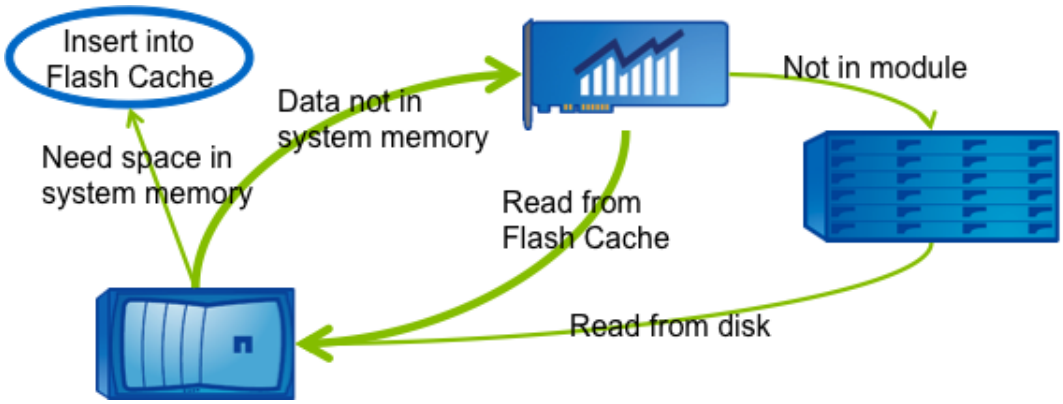


2.4 Reading Data from Flash Cache

When a read request arrives for data that resides in Flash Cache cache and is no longer in memory, it is served from the Flash Cache cache (instead of from disk) and inserted into memory buffers again. Once the data is in memory, it is forwarded to the requesting host or client.

Fetching data from Flash Cache is much faster than retrieving it from disk – typically at least 10 times faster. The typical result is improved application response times and server utilization, due to shorter wait times for data.

Figure 5) Reads from Flash Cache are typically 10 times faster than from disk.



2.5 Removing Data from Flash Cache

Even though Flash Cache can hold much more data than controller memory, Flash Cache is also a finite resource; it eventually becomes full and data will be evicted to make room for newer data. Flash Cache uses a first in, first out (FIFO) replacement algorithm for the data it caches. The data that is removed from the cache to free up space for new data inserts is the oldest data in the cache. All data blocks in Flash Cache are treated equally with this method, and hot data is retained for fast access in either memory buffer cache or Flash Cache cache.

2.6 Workloads That Are Accelerated by Flash Cache

Flash Cache can improve the response time of any repetitive read workload, regardless of the protocol that is used to access data on the storage system. Flash Cache works with both block (FC, FCoE, or iSCSI) and file (CIFS/SMB and NFS) access workloads. Section 3, “Flash Cache Modes of Operation,” explains how Flash Cache policy options are used to tune the caching behavior of Flash Cache.

2.7 Workloads That Are Not Accelerated by Flash Cache

There a two types of workload that generally do not benefit much from Flash Cache:

- Workloads that consist of mostly write requests
- Workloads that consist of mostly read requests that are not repeated or that are repeated infrequently

Write data is not cached by Flash Cache and therefore write workloads are not accelerated. In a mixed read-write workload where the write portion is dominant, there can be an indirect benefit from using Flash Cache: If the reads are served from cache, then the HDDs are able to handle more write I/Os.

Nonrepetitive read workloads don’t benefit from Flash Cache because the data inserted into cache is not read again. The situation is similar for workloads where the read requests are repeated infrequently; it is unlikely that the requested data will still be in cache long after the previous read.

3 Flash Cache Modes of Operation

Flash Cache provides three modes of operation, or caching policies: metadata mode, normal user data mode, and low-priority mode. The different modes enable Flash Cache caching behavior to be tuned to best match the workload the storage system is handling. The scope of data that is cached is narrowest with metadata mode and broadest with low-priority mode.

3.1 Metadata Mode

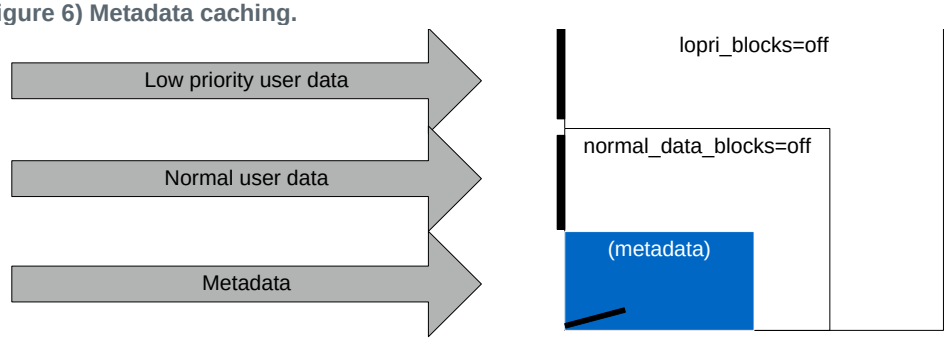
In the context of Data ONTAP, metadata consists primarily of indirect blocks (blocks that refer to the blocks that contain application or user data) and special files such as directories. When metadata mode is used, only the metadata that is about to be evicted from memory buffers is inserted into the Flash Cache cache; user data is evicted from memory without being cached.

All applications use metadata to some degree, and some workloads have a large amount of metadata in comparison to the application data. An example of a high metadata workload is one that consists of tens or hundreds of millions of small files. However, the NetApp FAS and V-Series storage controllers currently for sale support Flash Cache cards that are 512GB or larger in size. These cards are large enough to cache all the metadata for even high-file-count workloads with room to spare for some user data. As a result, using metadata mode is now rarely recommended.

Metadata mode is selected by setting the following FlexScale® options in Data ONTAP:

```
flexscale.enable              on
flexscale.lopri_blocks        off
flexscale.normal_data_blocks off
```

Figure 6) Metadata caching.



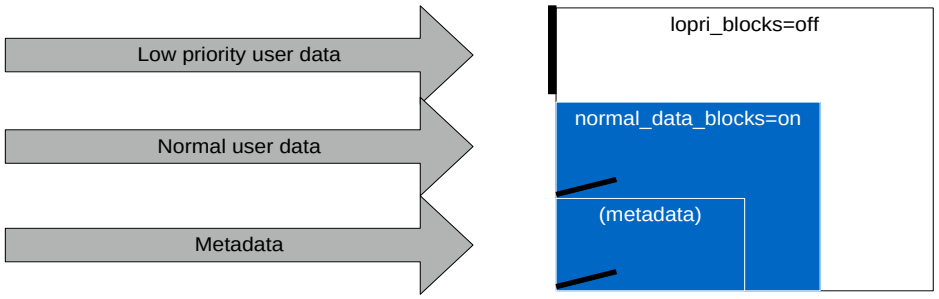
3.2 Normal Data Mode (Default Setting)

Normal data mode inserts both metadata and randomly read user data into Flash Cache before eviction from memory buffers. Normal data mode is the default caching mode for Flash Cache, and it is recommended for most deployments.

The FlexScale options settings to enable normal data mode are:

```
flexscale.enable              on
flexscale.lopri_blocks        off
flexscale.normal_data_blocks on
```

Figure 7) Normal data caching; both metadata and user data are cached.



3.3 Low-Priority Mode

Low-priority user data is application data that is normally the least beneficial to retain in memory buffers, and therefore this type of data is the first to be evicted to enable higher-priority data to be inserted into memory. When Flash Cache is used in normal mode, low-priority data is not inserted into Flash Cache. However, there are workloads where retaining low-priority data in cache does improve performance.

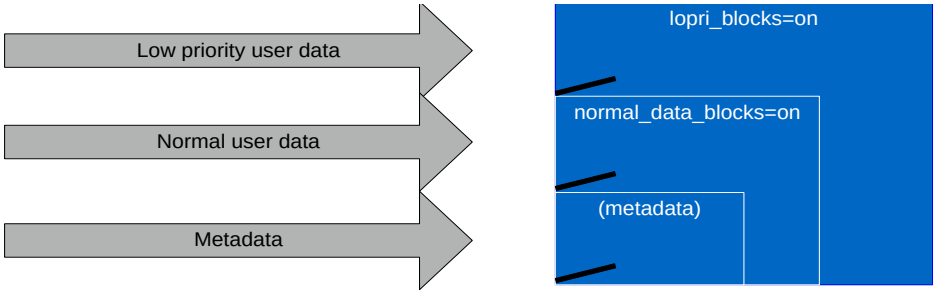
There are two categories of low-priority user data that are inserted into Flash Cache when this mode is used:

- **Write data.** With Data ONTAP, incoming write data is buffered in DRAM and logged to nonvolatile memory (NVRAM or NVMEM). After buffered write data has been committed to disk, it is flushed from NVRAM (or NVMEM) and retained at a low priority in DRAM. In other words, recently written data is evicted from memory as soon as room is needed for metadata or normal user data. There are some application workloads where recently written data is read back shortly after it has been written. For these workloads, Flash Cache improves read performance by retaining recently written blocks in cache before they are flushed from memory.
- **Long sequential read operations.** Data blocks from long sequential reads are retained in memory buffers only long enough to pass the data on to the requesting host or client. This data is evicted quickly from memory because it can rapidly overrun memory buffers with data that will be read only one time, thereby displacing metadata or random user data that is more likely to be read again. However, there are workloads where sequentially read data is requested by multiple clients over a short time period and retaining the data in Flash Cache provides benefit by offloading repetitive reads from HDDs.

The FlexScale options settings to enable low-priority mode are:

```
flexscale.enable              on
flexscale.lopri_blocks        on
flexscale.normal_data_blocks  on
```

Figure 8) Low-priority data caching; metadata, user data, and low-priority data are cached.



3.4 Sequential Read Caching with Normal Data Mode

Starting with Data ONTAP 8.2, there is an option to enable the caching of long sequential read data when Flash Cache is operating in normal data mode. Setting `flexscale.readahead_blocks` to `on` enables this option. This option is recommended instead of using low-priority mode when caching sequential read data is desired because the readahead option does not also cache write data.

3.5 Hybrid Caching with Data ONTAP Operating in 7-Mode

Flash Cache is a controller-wide resource, and the caching mode that is selected applies to data from all volumes provisioned on aggregates owned by a controller. Data ONTAP operating in 7-Mode includes a function called FlexShare® that when used with Flash Cache enables tuning cache retention policies at a volume level. For example, the best mode of operation might be caching metadata only for all but one

volume, for which normal mode data caching would be most beneficial. FlexShare priority setting can be used to modify Flash Cache caching behavior for the exception volume.

To use FlexShare with Flash Cache to tune caching behavior for specific volumes, follow these steps:

1. Enable FlexShare at the command prompt by entering:

   ```
   > priority on
   ```

2. Set FlexShare priorities to affect only caching behavior by entering:

   ```
   > priority set enabled_components=cache
   ```

3. At this point there are two options to choose from. The first option enables the retention of normal user data for a volume when Flash Cache is set to cache only metadata for all other volumes. To enable this behavior, use the command:

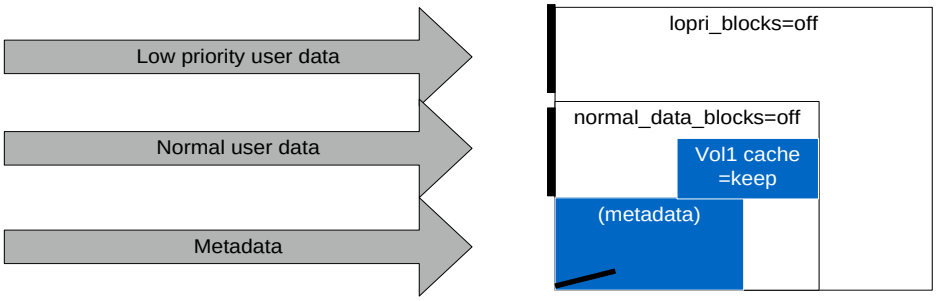
   ```
   > priority set volume <vol_name> cache=keep
   ```

   This command changes Flash Cache operation, for only the specified volume, so that both metadata and normal user data are cached.

   Note: If Flash Cache is used in normal or low-priority mode, the FlexShare setting in step 3 would have no effect.

   Figure 9 shows the caching behavior with the `cache=keep` priority set for a volume named Vol1.

Figure 9) Flash Cache operating in metadata mode with the `cache=keep` priority enabled for Vol1.

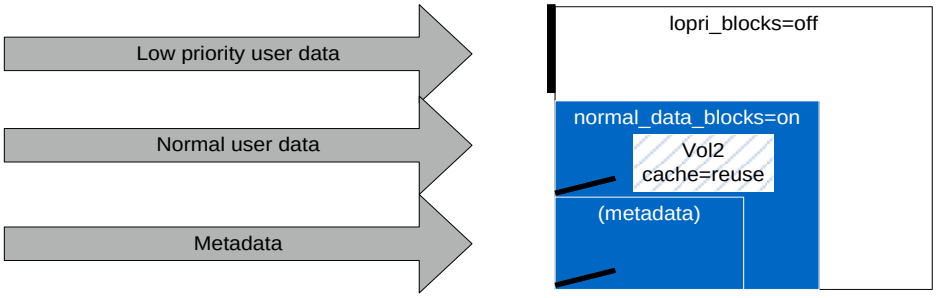


The second option is used when Flash Cache is operating in normal or low-priority mode, and the objective is to retain only metadata for a specific volume. In this case, the priority command to use is:

```
>  priority set volume Vol2 cache=reuse
```

Figure 10 shows this caching behavior for a volume named Vol2.

Figure 10) Flash Cache operating in normal mode with the `cache=reuse` priority enabled for Vol2.



FlexShare is available only with Data ONTAP operating in 7-Mode. With clustered Data ONTAP, the Flash Cache mode that is selected applies to all volumes that are provisioned on aggregates under a node (controller), and there is no capability to modify the caching behavior of individual volumes.

The “System Performance and Resources” section in the “System Administration Guide” contains more information about administering FlexShare. Additionally, NetApp TR-3459, “FlexShare Design and Implementation Guide,” discusses FlexShare in detail: http://www.netapp.com/us/library/technical-reports/tr-3459.html.

4 Flash Cache Operation with Clustered Data ONTAP

Flash Cache caching works the same in Data ONTAP operating in 7-Mode and in clustered Data ONTAP, except for the ability to use FlexShare with 7-Mode, as described in the previous section. However, the command line management and data access methods are different with clustered Data ONTAP; those are explained in this section

4.1 Management

Clustered Data ONTAP is managed from the cluster shell by default. Flash Cache is managed per node in the node shell. The node shell is reached by using the following command from the cluster shell:

```
Cluster::> node run –node <node_name>
```

At the node shell, the same commands covered in the previous section are used to modify caching settings. For example, normal data caching mode would be enabled on node “nodeA” with the following commands:

```
nodeA> options flexscale
flexscale.enable              on
flexscale.lopri_blocks        off
flexscale.normal_data_blocks  on
```

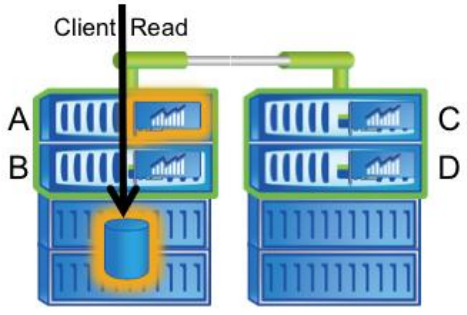
4.2 Operation

With Flash Cache, a volume’s data is cached on the node (controller) that is managing the aggregate on which the volume is provisioned. The following examples illustrate Flash Cache caching when data is accessed directly through the node where a volume is provisioned, and when it is accessed indirectly through a different node. Both examples depict a four-node (two HA pair) cluster.

Direct Data Access

In this example, a client is connected via LIF on Node A and data is being requested from a volume that is provisioned on an aggregate managed by Node A. Data is served back to the client by Node A directly, without traversing the cluster interconnect. If the requested data is cached, it is served from Flash Cache; if not, it is fetched from disk. Data from this volume is cached in Flash Cache on Node A only.

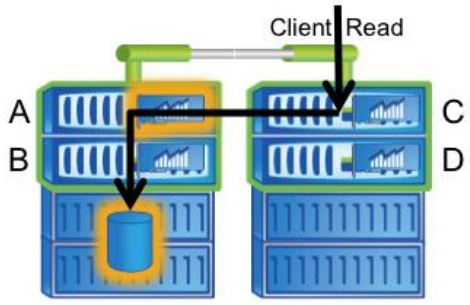
Figure 11) Direct data access with clustered Data ONTAP.



Indirect Data Access

In this example, a client is connected via a LIF on Node C and data is being requested from a volume that is provisioned on an aggregate managed by Node A. Data is served back to the client by Node C after it is retrieved from Node A and traverses the cluster interconnect to Node C. If the requested data is cached, it resides in Flash Cache on Node A and is served from cache. If the data is not cached, it is retrieved from disks on Node A. In this case, the requested data is cached on Node A only; it is never cached on Node C.

Figure 12) Indirect data access with clustered Data ONTAP.



5 Flash Cache Interoperability with Other Data ONTAP Features

Flash Cache operates at a level below most of the other software features of Data ONTAP. This enables Flash Cache to accelerate workloads in systems that are also using data protection features such as SnapMirror® and SnapVault®, and storage efficiency features such as deduplication, FlexClone®, and compression.

Flash Cache incorporates the storage efficiency capabilities of deduplication and FlexClone, however it works differently with data compression. Interoperability with these features, as well as using Flash Cache and Flash Pool together, is covered in this section.

5.1 Deduplication

Deduplicated blocks are cachable in the same way as nondeduplicated blocks. In addition, blocks cached in Flash Cache are deduplicated just as they are on disk; that is, only one physical block is needed to cache multiple logical references to the same data. In this way, deduplication enables Flash Cache to store more unique data, thereby increasing the effectiveness of the cache.

5.2 FlexClone

Flash Cache caches blocks from file, LUN, and volume clones, and only one physical block is needed to cache multiple references to a cloned block. As with deduplication, Flash Cache is able to cache more unique data when FlexClone is used, which increases the effectiveness of the cache.

5.3 Compression

Flash Cache can cache data from volumes on which compression has been enabled. However, only the uncompressed blocks from these volumes are cachable; compressed blocks are not inserted into Flash Cache.

5.4 Flash Pool

Flash Cache and Flash Pool can be used together on one node (controller), within an HA pair, and within a single clustered Data ONTAP cluster. Data from volumes that are provisioned in a Flash Pool aggregate or an all-SSD aggregate is automatically excluded from being cached in Flash Cache. Data from volumes provisioned in a Flash Pool aggregate uses Flash Pool cache, and data from volumes provisioned in an SSD aggregate would not benefit from using Flash Cache.

Although Flash Cache and Flash Pool can be configured on the same node or within the same HA pair, there are limits on the total cache that can be configured. The maximum cache sizes when using Flash Cache and Flash Pool vary by controller model and Data ONTAP release; detailed information can be found in the NetApp Hardware Universe, for customers who have NetApp Support site access.

6 Performance Expectations and Performance Monitoring

6.1 Cache Warm-Up Period

The amount of Flash Cache supported per controller varies by model and scales up to 8 TiB on the largest model. Consequently, it can take many hours to fully warm up the cache on a system. To understand the difference between a cache filling up and warming up, and the implications of a cache that is empty, it's necessary to understand a little more about how a storage system uses a cache. At boot time, or after the giveback in a takeover event after a failure or if rewarming was disabled, the Flash Cache module does not have data in it that can be used to accelerate the workload. This results in all the initial reads coming from the disk subsystem. As the data flows through the system, the cache is populated. Eventually the system reaches a point at which it either doesn't need to take in more data; the only data it is taking in replaces data that has been invalidated due to an overwrite; or the amount of data flowing through the system is more than the cache can hold, so it replaces valid entries to make room for new data.

In the second scenario, the cache is at 100% effectiveness only at the point at which it is full. The usage does not indicate how "busy" the cache is, but rather how full it is. In many cases, the cache never fills to 100% but hovers at slightly less than 100% usage because of the algorithm that Data ONTAP uses to manage the cache.

To determine how long it might take to completely fill up the cache use the following simple calculation:

time to fill the cache = (size of cache) / (read throughput per second)

For example:

512GB/100MB per second = 5,120 seconds = 85 minutes

At first glance this scenario might be daunting, but the missing part here is the behavior of a cache and cached data over time. As a cache fills up and is used by the system, the data most likely to be reused is the data most recently placed in the cache. As the cache fills, the relative effect of more data in the system lessens. Because of this, a caching technology has the most effect in the first 10% of usage and the least effect in the last 10% as the cache is filled. Caches need to be large to have the largest possible effect and to maximize the benefit for large datasets, but the benefit of the cache begins with the first reusable entries placed into it.

In looking at cache warm-up times, it's necessary to consider:

- What is the read throughput of the workload?
- What is the working set size?
- How much reuse is in the working set?

These factors combine to affect the warm-up time. The throughput, as described earlier, directly affects the rate of data that the cache can see. The working set size can be larger or smaller than the actual amount of data being cached; in fact, many workloads have hot spots within their working set in which smaller portions of data are active, and these hot spots migrate to other locations in the working set. Finally, if there is a large amount of reuse in the working set, the warm-up sees more immediate effect than when there is less reuse.

6.2 Flash Cache Rewarming

Flash Cache rewarming is a feature introduced with Data ONTAP 8.1 that allows the storage system to retain the contents of the cache after user-initiated controller takeovers or shutdowns. Rewarming can significantly reduce the warm-up time after a controller reboot or giveback, which allows the system to regain optimal performance more quickly. Rewarming is enabled by default and requires no additional configuration in Data ONTAP. Rewarming does not take place if the system panics or loses power, or if certain commands are issued that prevent rewarming. Details about these commands are available in the "System Administration Guide."

In highly available environments, during a user-initiated takeover, the takeover partner maintains a log of data that must be invalidated in the failed partner's cache. If a large amount of data stored in the cache is invalidated during the takeover, less data is able to be rewarmed when the node comes back online, and the cache might require some additional warming time.

6.3 Availability Considerations

Takeover Events

When designing a solution that includes Flash Cache, keep these points in mind:

- NetApp recommends a symmetric configuration of the cache on each node of an HA pair. If one of the systems has 2TB of cache, for example, the other system should have 2TB as well, to enable more consistent performance in the event of a takeover.
- In the event of a takeover, the partner node becomes responsible for caching data. Upon giveback, the cache might need to be reinitialized, depending on the reason for the takeover. If cache rewarming is possible, warming time might be reduced.

Nondisruptive Failures

Flash Cache is designed to fail in a nonfatal way. In the event of a hardware problem, the module is taken offline and the system continues to operate without it. This allows data availability until downtime can be taken to replace the failed module.

6.4 Monitoring Flash Cache Performance

Flash Cache uses the Data ONTAP counter manager architecture to maintain performance data points. The name of the counter manager object for the modules is:

```
ext_cache_obj
```

To view information in the counter object, use the command:

```
>stats show ext_cache_obj

ext_cache_obj:ec0:type:IOMEM-FLASH

ext_cache_obj:ec0:blocks:402653184
```

```
ext_cache_obj:ec0:size:1560
ext_cache_obj:ec0:usage:1%
ext_cache_obj:ec0:accesses:0
ext_cache_obj:ec0:disk_reads_replaced:0/s
ext_cache_obj:ec0:hit:0/s
ext_cache_obj:ec0:hit_normal_lev0:0/s
ext_cache_obj:ec0:hit_metadata_file:0/s
ext_cache_obj:ec0:hit_directory:0/s
ext_cache_obj:ec0:hit_indirect:0/s
ext_cache_obj:ec0:total_metadata_hits:0/s
ext_cache_obj:ec0:miss:0/s
ext_cache_obj:ec0:miss_metadata_file:0/s
ext_cache_obj:ec0:miss_directory:0/s
ext_cache_obj:ec0:miss_indirect:0/s
ext_cache_obj:ec0:hit_percent:0%
ext_cache_obj:ec0:inserts:0/s
ext_cache_obj:ec0:inserts_normal_lev0:0/s
ext_cache_obj:ec0:inserts_metadata_file:0/s
ext_cache_obj:ec0:inserts_directory:0/s
ext_cache_obj:ec0:inserts_indirect:0/s
ext_cache_obj:ec0:evicts:0/s
ext_cache_obj:ec0:evicts_ref:0/s
ext_cache_obj:ec0:readio_solitary:0/s
ext_cache_obj:ec0:readio_chains:0/s
ext_cache_obj:ec0:readio_blocks:0/s
ext_cache_obj:ec0:readio_max_in_flight:236
ext_cache_obj:ec0:readio_avg_chainlength:0
ext_cache_obj:ec0:readio_avg_latency:0ms
ext_cache_obj:ec0:writeio_solitary:0/s
ext_cache_obj:ec0:writeio_chains:0/s
ext_cache_obj:ec0:writeio_blocks:0/s
ext_cache_obj:ec0:writeio_max_in_flight:67
ext_cache_obj:ec0:writeio_avg_chainlength:0
ext_cache_obj:ec0:writeio_avg_latency:0ms
ext_cache_obj:ec0:invalidates:0/s
```

Although complete, this output gives only a single point-in-time view of the counter data. Therefore, NetApp recommends using the iterative form of the command with a preset for the Flash Cache counters. This provides output every 5 seconds of per-second data rates for the most critical counters:

```
>stats show -p flexscale-access
```

The output has the following format:

```
Cache                                                 Reads       Writes      Disk Reads
Usage    Hit   Meta   Miss Hit Evict Inval Insert Chain Blocks Chain Blocks  Replaced
     %     /s     /s     /s   %    /s    /s     /s    /s     /s    /s     /s        /s
     0      0      0      0   0     0     0      0     0      0     0      0         0
```

Here are the definitions for those counters:

- **Cache Usage.** How much data is currently stored in the module or modules
- **Hit.** The 4kB disk block cache hit per second
- **Meta.** The 4kB metadata disk block cache hit per second
- **Miss.** The 4kB disk block cache missed per second
- **Hit.** The percentage of total hit/miss
- **Evict.** The 4kB disk blocks evicted (leaving cache because of set collision) from the cache per second
- **Inval.** The 4kB disk blocks invalidated (leaving cache because of overwrite or making room) from the cache per second
- **Insert.** The 4kB disk blocks inserted into the cache per second
- **Reads Chain.** The number of read I/O chains per second
- **Reads Blocks.** The number of 4kB disk blocks read per second
- **Writes Chain.** The number of write I/O chains per second
- **Writes Blocks** The number of 4kB disk blocks written per second
- **Disk Reads Replaced.** The number of reads that would have gone to disk that were replaced by the cache per second

In this output, the most important data are the hit rate and the number of disk reads replaced. It might also be useful to capture sysstat data to understand the amount of workload on the system.

7 Cache Sizing

Flash Cache can benefit a variety of workloads, but the overall improvement depends on the workload characteristics and working set size. There are many variables that affect the ability to cache data in Flash Cache. However, the more data of the working set that can be cached, the higher the probability that data will be served out of Flash Cache instead of going to disk. For most workloads, the number of hits continues to increase asymptotically as the cache size is increased toward the maximum rate possible given the workload characteristics. Figures 13 and 14 are examples of a workload executed against two different working set sizes (300GB and 500GB) at different caching points. The trend shows diminishing returns as more cache is added; however, overall benefit continues to increase. The first graph shows the overall hit percentage. The second graph shows the incremental benefit of more cache, or the additional hit percentage achieved by adding the extra cache.

Figure 13) Hit rates with constant workload and increasing cache size for two working set sizes.

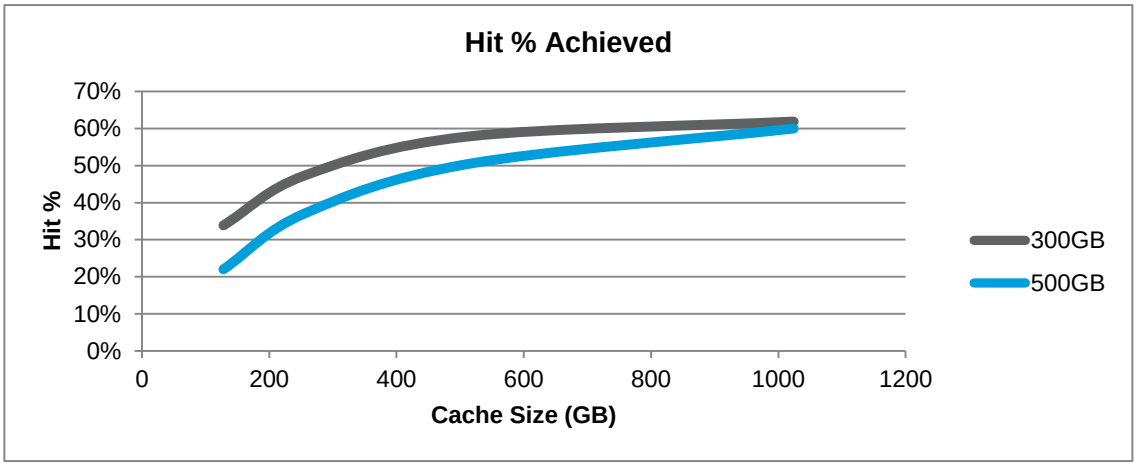
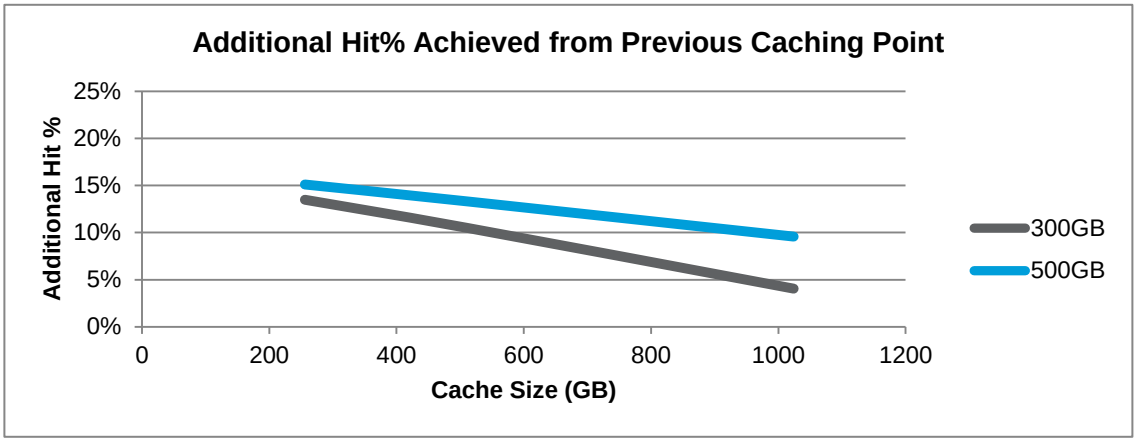


Figure 14) Additional hit rates achieved by adding more cache.



7.1 Using Predictive Cache Statistics

Predictive Cache Statistics (PCS) is a feature built into Data ONTAP that emulates a Flash Cache cache. PCS captures cache statistics and observes what the improvement might be with a real Flash Cache cache. PCS allows varying the cache size and using different operating modes (as described earlier in this report) to determine the optimal configuration. See TR-3801, “Introduction to Predictive Cache Statistics” for more information about how to use PCS.

7.2 Considering More Cache

Systems that already have Flash Cache modules installed cannot run PCS without first disabling Flash Cache. This brings up the question of how to determine whether more Flash Cache would help, since disabling Flash Cache isn’t typically acceptable. It’s possible to get an idea of whether the system could benefit from additional Flash Cache cache by using the cache statistics provided by the system, but ultimately the answer comes down to the workload.

Generally more cache won’t help in the following situations:

- **The workload is mostly writes.** Workloads dominated by writes might end up overwriting and invalidating data in the cache, meaning that hit percentages will be low.
- **The workload is known to have low reuse of reads.**

- **The cache is not full and is not continuing to fill,** as indicated by the usage value. This means that the working set is already fully captured in the cache.

If the workload is believed to be cacheable—that is, if it is has a large random read component, and the cache is full—here are some indicators that more cache could be beneficial:

- **Low hit percentage.** If the workload is dominated by reads, a cache of the proper size can achieve very high hit percentages.
- **Data flowing through the cache.** If the working set is larger than the cache, there is churn of data that is cached. If more of the working set can be cached, the flow through the cache might slower, and the probability of hits might increase because data remains in the cache longer. Data flowing through the cache can be observed by looking at the insert and invalidate statistics.

If these indicators are observed, it's possible that more cache will increase the amount of reads that are offloaded from disk.

Note: Given a single data point, it's not possible to determine the absolute additional benefit more cache can provide.

8 Conclusion

Flash Cache improves performance for a wide variety of application workloads by caching repetitive random reads. Flash Cache is supported with clustered Data ONTAP and Data ONTAP operating in 7-Mode it is simple to configure and use. Flash Cache works with block (SAN) and file (NAS) data and interoperates with most other the other data management features of Data ONTAP.

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.

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 andthe information contained herein may be used solely in connection with the NetApp products discussed in this document.

Go further, faster®

NetApp
www.netapp.com