Technical Report

Flash Cache Best Practices Guide

Paul Updike, Chris Wilson, NetApp
April 2012 | TR-3832

Best Practice Considerations

Flash Cache (previously called PAM II) is a hardware and software solution that allows you to enhance a NetApp® storage controller's performance characteristics. This guide covers the essential information needed to make well-informed implementation decisions.

TABLE OF CONTENTS

LIST OF TABLES

1 Introduction

1.1 About Flash Cache

Flash Cache (previously called PAM II) is a solution that combines software and hardware within NetApp storage controllers to increase system performance without increasing the disk drive count. Flash Cache is implemented as software features in Data ONTAP® and PCIe-based modules with either 256GB, 512GB, or 1TB of Flash memory per module. Flash Cache cards are controlled by custom-coded field-programmable gate arrays (FPGAs). Multiple modules may be combined in a single system and are presented as a single unit. This technology allows submillisecond access to data that would previously have been served from disk at averages of 10 milliseconds or more.

1.2 Storage System Performance Versus Storage System Capacity

There are a variety of workloads with which a storage system may be presented, and most will have a random read component. Random reads are reads from noncontiguous locations on a storage system's disks. Because these reads are not physically located near one another, they can take longer to satisfy than a workload with more localized, sequential reads. This kind of work creates more seeks to locations on the disks, and latency increases as more time seeking is factored into performance. A workload with a high percentage of small-block random reads can be one of the most challenging for any storage system.

Historically, to address the performance requirements of a random read workload, you added more disks to the system. More disks mean more heads and less of a chance for two reads coming from the same disk back to back. The additional disks allowed a storage system to satisfy the read performance of such a workload. This never tended to be a problem, because more disks were also required to satisfy the capacity requirements of the workload. You could buy capacity in the form of more disks, and you received random read performance at the same time.

Over the last few years, disk drives have increased in capacity at a tremendous rate. For example, vendors introduced Fibre Channel drives that increased from 18GB to 36GB to 72GB to 144GB to 300GB in less than a decade. At the same time as this exponential increase in capacity took place, the time associated with a drive head seeking to a location on one of the drive platters improved by less than 20%. This resulted in a disparity between the performance growth of the drive and the capacity. You can store more data on fewer drives but might have to provide the same number of spindles to meet the performance requirement.

As a result, to satisfy the performance requirements of many workloads, you might have to buy extra capacity. As drive capacities continue to push their limits, this situation worsens. At the same time that this trend occurred, resource constraints pushed customers to drive toward less power consumption and to reduce space needs in their environments.

1.3 Addressing Resource Constraints

As information technology continues to grow in the business world, three fundamental resources are in short supply in many data centers: power, space, and cooling. As discussed in section 1.2, a number of workloads no longer require as many disk drives to satisfy capacity, but instead additional drives are required to make sure of performance. These additional disks create more space and power requirements, and, with more power, more cooling is required. This situation reinforces the need to be able to satisfy performance *or* capacity separately.

1.4 How Flash Cache Solves These Problems

Flash Cache replaces disk reads with access to an external cache contained in one or more hardware modules. Your workload is accelerated in direct proportion to the disk reads replaced. The remainder of

this document focuses on different workloads and how they are accelerated, how to choose and configure the best mode of operation, and how to observe Flash Cache at work.

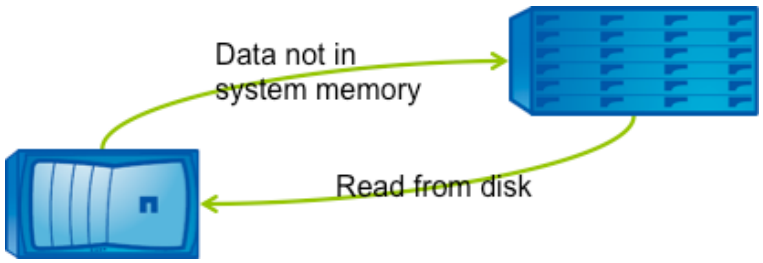
2 How Flash Cache Works

Flash Cache adds new capability to Data ONTAP. It accomplishes this by changing a basic system behavior, retrieving data to satisfy a client's or host's read operation. We look here at the way data is brought into the system and how it changes with Flash Cache.

2.1 Data ONTAP Disk Read Operation

In Data ONTAP prior to Flash Cache, when a client or host needed data and it wasn't currently in the WAFL® buffer cache in system memory, a disk read resulted. Essentially, the system asked itself if it had the data in RAM, and, if the answer was no, it went to the disks to retrieve it.

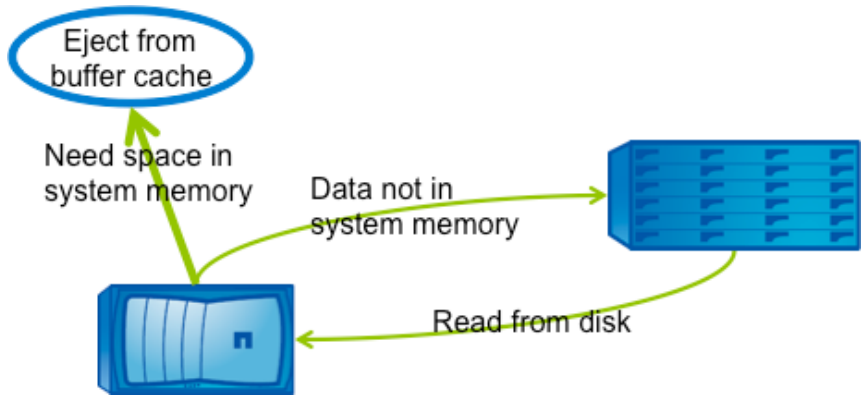
Figure 1) A read before introducing Flash Cache.



2.2 Data ONTAP Clearing Space in System Memory for More Data

When more space was needed in memory, Data ONTAP analyzed what it currently held and looked for the lowest priority data to clear out to make more space. Depending on the workload, this data could have resided in the buffer cache for seconds or hours; either way it had to be cleared.

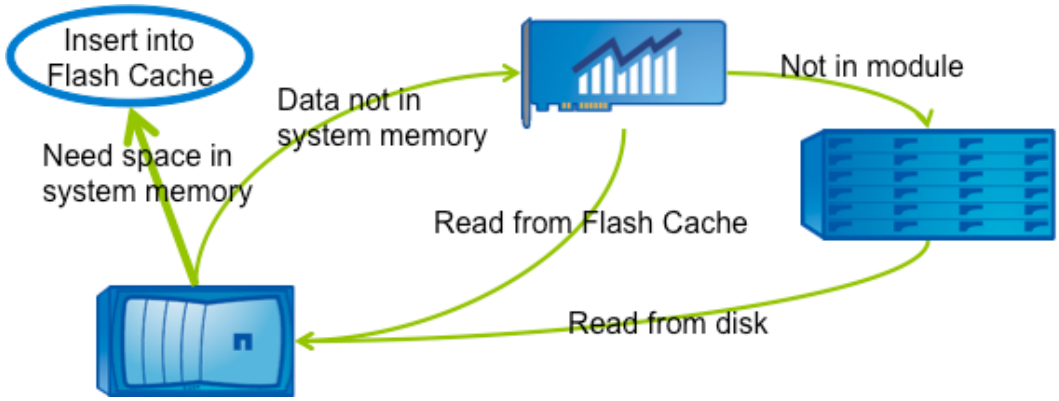
Figure 2) Clearing memory before introducing Flash Cache.



2.3 Saving Useful Data in Flash Cache

With the addition of Flash Cache, the data that would have previously been cleared is now placed in Flash Cache. Data is always read from disk into memory and then stored in Flash Cache when it needs to be cleared from the buffer cache.

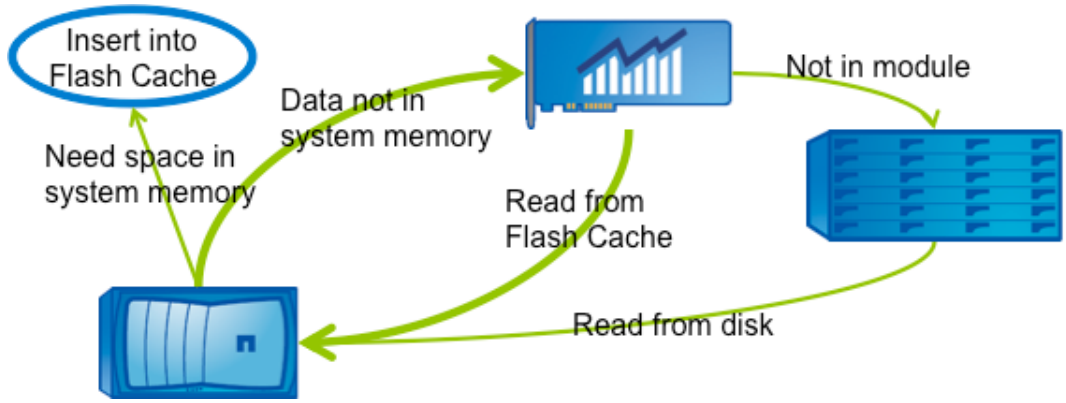
Figure 3) Data is stored in Flash Cache.



2.4 Reading Data from Flash Cache

Now that the data is stored in Flash Cache, Data ONTAP can check to see if the data is there the next time it is needed. When it is there, access to it is far faster than having to go to disk. This is how a workload is accelerated.

Figure 4) Reads from Flash Cache are 10 times faster than disk.



2.5 Removal of Data from Flash Cache

Flash Cache uses a first in, first out (FIFO) replacement algorithm for the data in cache. This means that the data removed from the cache to make room for new inserts is the oldest data in the cache. Although all blocks in Flash Cache are treated equally in this method, hot data will remain in the buffer cache/Flash Cache loop and continue to benefit from caching.

2.6 Read Acceleration and Maximum System Performance

An important thing to take away from the operation of Flash Cache is that it is a read cache. Flash Cache is designed to accelerate read performance only. Data ONTAP is already optimized for writes, and Flash Cache technology is not meant to improve them. Some benefit might be observed with write-intensive workloads because disk utilization might decrease due to read operations being handled by Flash Cache instead of going to disk. In general, write performance is not improved by Flash Cache.

Likewise, Flash Cache improves performance of the disk subsystem but does not accelerate the top line performance of a system. That is, if the most a system can perform is 100,000 IOPS, it will not be able to perform more IOPS after a Flash Cache module is included; that maximum number does not change.

2.7 Protocols That Are Accelerated

Flash Cache works at a layer below all network storage protocols and will therefore accelerate any of these protocols. As discussed in the next sections, the type of workload presented to Flash Cache determines the acceleration seen, not the protocol.

2.8 Workloads That Are Not Accelerated

Other workloads that might not see as large a benefit are:

- A workload that is mostly writes
- A workload that is mostly very large sequential reads

Writes are not directly accelerated by Flash Cache, although there might be a minimal improvement due to fewer possible disk reads needed to complete writes. Similarly, large sequential reads tend to be less likely to be reread in time for the data to still be retained in the module, and the system can already process sequential reads more efficiently than random reads. As a result, these types of workloads are less likely to benefit.

Either of these workloads might see an improvement, but it might not be as drastic as those mentioned earlier. If the size of the active data set fits well into Flash Cache, there is an exception. A mode of operation specific to this scenario is detailed in section 3.3 of this document.

3 Modes of Operation

There are three modes of operation for Flash Cache. These provide the ability to tune the caching behavior of the module to match the storage system's workload. As we move through each of the modes, Data ONTAP allows a broader amount of data to be stored in the module.

3.1 Metadata Caching

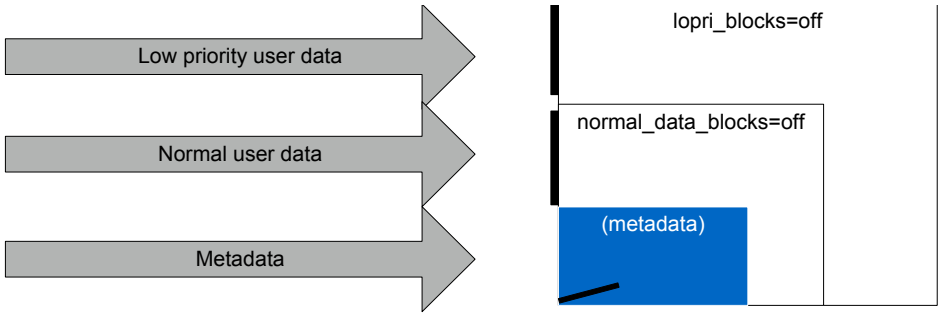
Metadata mode: The first way to improve performance with Flash Cache is in workloads with a heavy metadata overhead. Metadata consists of the WAFL special files and indirect blocks. All applications use metadata; however, workloads have such a large working set that caching metadata only might provide the most improvement. For these workloads, placing the metadata in the Flash Cache module allows low-latency access to the metadata; it also provides higher speed access to the application data. Because of the much larger size of Flash Cache, this mode is more applicable to PAM I, the original 16GB DRAM–based Performance Acceleration Module, than Flash Cache.

```
flexscale.enable              on

flexscale.lopri_blocks        off

flexscale.normal_data_blocks  off
```

Figure 5) Metadata-only caching.

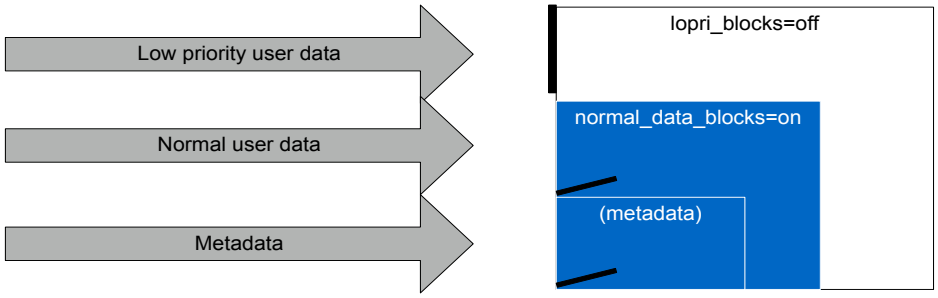


3.2 Normal User Data Caching (Default)

Normal mode: The second caching mode for Flash Cache is normal user data mode and the default for Flash Cache. In this mode, both metadata and randomly read data blocks are eligible for caching. On successive accesses, instead of going to disk, which is higher latency, the data is served from the memory onboard the Flash Cache module. The options for this mode look like:

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

Figure 6) User data caching; both metadata and user data are cached.



3.3 Low-Priority Data Caching

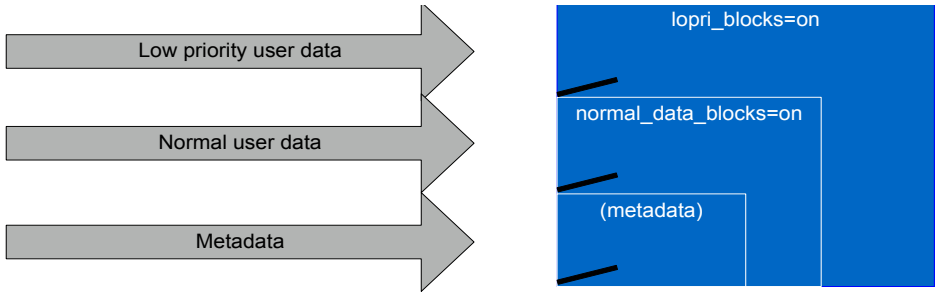
Low-priority blocks mode: A third way to improve performance is through capturing application data that normally would have been forced to disk or not cached at all. These are called low-priority buffers and exist in a couple of forms:

- Write data: Normally, writes are buffered in RAM and logged to NVRAM, and, once committed to disk, they are flushed from NVRAM and retain a lower priority in RAM to avoid overrunning the system memory. In other words, recently written data is the first to be ejected. In some workloads, recently written data may be immediately accessed after being written. For these workloads, Flash Cache improves performance by caching recently written blocks in the module rather than flushing them to disk and forcing a disk access for the next read.
- Long sequential read blocks: Long sequential reads can overrun the system memory by overflowing it with a great amount of data that will only be accessed once. By default, Data ONTAP does not keep this data in preference to holding data that is more likely to see reuse. The large amount of memory space provided by Flash Cache allows sequential reads to potentially be stored without negatively

affecting other cached data. If these blocks are reaccessed, they will see the performance benefit of the module rather than going to disk.

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

Figure 7) Low-priority caching; metadata, user data, and low-priority data cached.



3.4 Hybrid Modes with FlexShare

The FlexShare® tool provides quality-of-service functionality to NetApp storage controllers. When combined with the intelligent caching of the Performance Acceleration Module it allows you to set caching policies on specified volumes, adding or subtracting from the systemwide mode set with the FlexScale™ software options.

For example, you might have determined the best mode of operation to be that of metadata only for the system, but there is one volume in which you want normal data cached as well. Set the system's FlexScale options as normal. Then, at the command prompt, either enable FlexShare as normal or only the caching components.

To enable FlexShare at the command prompt, type:

```
> priority on
```

If you only want the caching of FlexShare enabled (if you intend to use all FlexShare functionality, do not perform this step), also type:

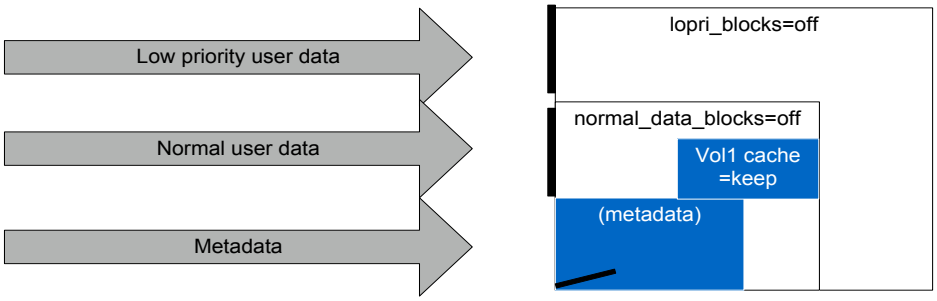
```
> priority set enabled_components=cache
```

Once the components are enabled, you have two options to combine with Flash Cache caching mode:

```
> priority set volume Vol1 cache=keep
```

This changes the caching behavior of FlexShare to more aggressively retain data for the flexible volume, Vol1. When enabled, and the module is in normal or low-priority mode, it has no effect. But, as illustrated in the following diagram, when the system is set to metadata mode and the volume has the "cache=keep" setting, the behavior of the cache allows retaining metadata for all volumes and additionally allows normal data for Vol1.

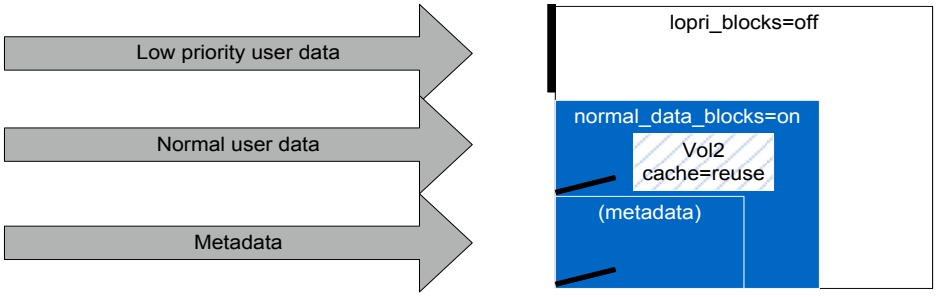
Figure 8) System setting is metadata only, but "keep" for Vol1 means normal user data for Vol1 will be cached.



Similarly, if the module is set to normal or low-priority mode and it makes sense for a volume, in this case Vol2, to not have its normal data cached, you may subtract that data from the Flash Cache module with the following command:

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

Figure 9) System setting is normal user data mode, but "reuse" for Vol2 means that only metadata will be cached for Vol2.



The section in the “System Administration Guide” called “System Performance and Resources” contains more information on administering FlexShare. Additionally, NetApp TR-3459: “FlexShare Design and Implementation Guide” discusses FlexShare in detail and is available under “Technical Reports” at www.netapp.com (http://www.netapp.com/us/library/technical-reports/tr-3459.html).

4 Flash Cache in Data ONTAP Operating in Cluster-Mode

The following sections describe how Flash Cache operates and can be managed in Data ONTAP systems operating in Cluster-Mode.

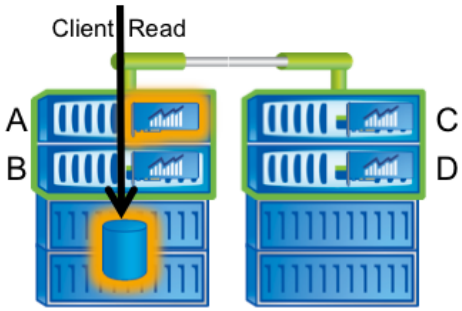
4.1 Operation

The caching operation of Flash Cache does not change when operating in Cluster-Mode compared to 7-Mode. Data is only cached on the node that is responsible for the aggregate containing the data. Failover operation also remains the same between high-availability pairs. The following examples illustrate how data is cached in direct and indirect data access situations in a four-node (two HA-pair system) Cluster-Mode system.

Direct Data Access

In this example, a client is attached to a LIF on node A and is requesting data that is stored on an aggregate physically connected to node A. Data will be served back to the client by node A without traversing the cluster interconnect. If the data was already cached, it will be served out of Flash Cache; if not, disk access is necessary. The data will be cached on node A only.

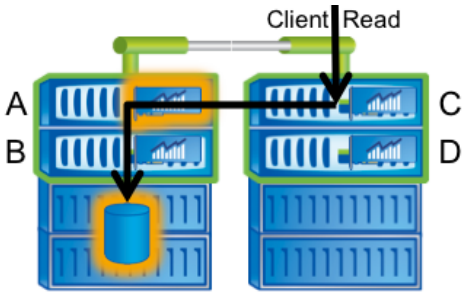
Figure 10) Direct data access in Cluster-Mode.



Indirect Data Access

In this example, a client is attached to a LIF on node C and is requesting data that is stored on an aggregate physically connected to node A. Data will be served back to the client by node C after traversing the cluster interconnect to retrieve the data from node A. If it was cached on node A, it will be served from the cache; otherwise, disk access is necessary. The data will be cached on node A and not on node C.

Figure 11) Indirect data access in Cluster-Mode.



4.2 Management

The operating modes of Flash Cache are the same between Data ONTAP operating in 7-Mode and Cluster-Mode. Flash Cache is managed per node on the node shell. Enter the node shell by using the following command on the cluster shell:

```
Cluster::> node run -node nodeA
```

Once on the node shell, use the same commands illustrated in previous sections to modify the caching settings. The statistics can also be viewed the same way on the node shell.

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

5 Predictable Performance Expectations

5.1 Cache Warm-Up Period

With multiples of 256GB, 512GB, and 1TB, the Flash Cache module can take some time to fill up. To understand the difference between a cache filling up and warming up and the implications of a cache that is empty, we have 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. At some point in time 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.

If we assume the second scenario, the cache will only be at 100% effectiveness at the point at which it is full. The usage is not indicative of how "busy" the cache is but rather how full it is. In many cases, the cache will never fill to 100% but will hover in the slightly less than 100% usage range because of the algorithm Data ONTAP uses to manage the cache.

To determine how long it might take to completely fill up, we can do a simple calculation:

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

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 a 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. We want caches to be large to have the most possible effect and to apply to the largest datasets, but you start to reap the rewards of the cache with the first reusable entries placed into it.

When we look at cache warm-up times we want to consider a few things:

- 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 inside their working set in which even smaller portions of data are active, and these hot spots migrate to other locations within the working set. Finally, if there is a large amount of reuse in the working set, the warm-up will see more immediate effect than when there is less reuse.

5.2 Flash Cache Rewarming

Flash Cache rewarming is a feature available in Data ONTAP 8.1 and later that allows the storage system to maintain the contents of Flash Cache during a user-initiated downtime. Rewarming can significantly reduce the warm-up time after rebooting, allowing the system to regain the optimized performance more quickly. It is enabled by default and requires no additional configuration within Data ONTAP. Rewarming will not take place if the system panics, loses power, or 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 will be able to be rewarmed when the node comes back online, and the cache might require some additional warming time.

5.3 Availability Considerations

Takeover Events

When designing a solution that includes Flash Cache, keep in mind a couple of points.

- NetApp recommends a symmetric approach to the size of the cache on either system in an HA pair. If one of the systems has 2TB of cache, for example, we recommend having 2TB of cache in the other system as well. This is to enable 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. Warming time might be reduced if cache rewarming is possible.

Nondisruptive Failures

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

6 Interaction with Other Data ONTAP Features

Flash Cache works in Data ONTAP at a level below the software functionality of other features. In this way, the module is able to accelerate these features in the same way it would standard NAS or SAN protocol access.

6.1 Deduplication

Deduplicated blocks can be cached in Flash Cache just like any other block. Only one copy is kept in Flash Cache, just as it is kept on disk. This also means that it’s possible to store a larger amount of logical data in the cache, potentially increasing the effectiveness of the cache.

6.2 FlexClone

Flash Cache is fully interoperable with flexible clones. If data is being accessed from a flexible clone and the access pattern fits those mentioned in section 2.5, those accesses will be accelerated as any other data would. As with FAS deduplication, only one copy of a flexible clone data block is kept in the module, regardless of how many flexible clones share that block.

6.3 FlexCache

Flash Cache works with NetApp FlexCache® technology. By working at a level below the network protocols, it can hold data for FlexCache volumes just as it would for any other volumes.

6.4 SSD Aggregates and Flash Pool

Data that's stored on SSD will not see much, if any, benefit from caching in Flash Cache. Therefore, any data that's already stored on an SSD aggregate or a Flash Pool will not be cached in Flash Cache. However, Flash Cache can be installed alongside these technologies to provide caching benefits for aggregates that do not already receive the benefits of Flash.

Although the technologies can be combined on a single system, there are limitations on the maximum amount of Flash that the system can support. Consult the controller configuration guides on the NetApp Support site (http://support.netapp.com) to determine the allowable combination for your system.

7 How Do You Monitor the Performance of Flash Cache?

Flash Cache products utilize 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 within 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
```

While complete, this gives you only a point-in-time view of the data in the counters. For this reason, NetApp recommends using the iterative form of the command with a preset for the Flash Cache counters. This will give you an output every five seconds of per-second data rates for the most critical counters. This command looks like:

```
>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(s)
- **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 main items to note are the hit rate and the number of disk reads replaced. It also might be useful to capture sysstat data to understand the amount of workload on the system.

8 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 served out of Flash Cache instead of going to disk. For most workloads, the number of hits will continue to increase asymptotically toward the maximum rate possible given the workload characteristics as the cache size is increased. The following graphs are examples of a workload executed against two different working set sizes 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 percent. The second graph shows the incremental benefit of more cache, or the additional hit percent achieved by adding the extra cache.

Figure 12) Example hit percent with constant workload and increasing cache size with various working set sizes.

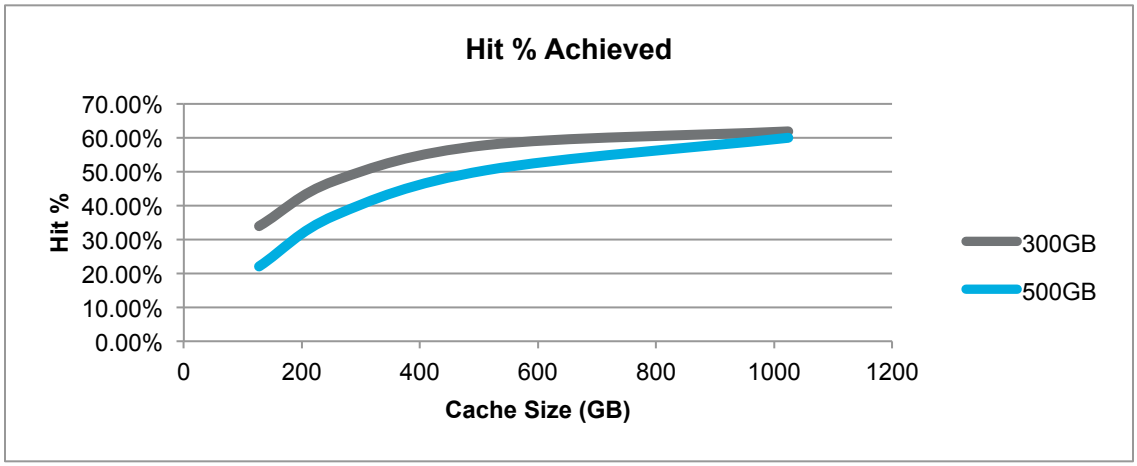
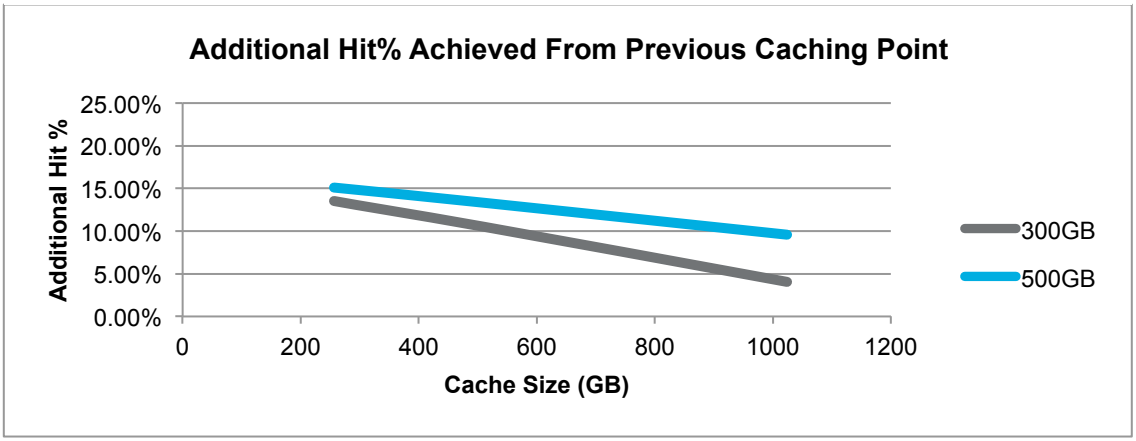


Figure 13) Additional hit percent achieved by adding more cache.



Deciding how much cache will provide the optimal effects is possible to determine using predictive cache statistics and other statistics can help indicate whether or not more cache could be useful for your system.

8.1 Using Predictive Cache Statistics

Predictive cache statistics (PCS) is a feature built into Data ONTAP that allows you to emulate a Flash Cache in software to capture cache statistics and observe the improvement might be with a real Flash Cache. You can try various sizes of total cache as well as the different operating modes described earlier to determine what configuration is optimal. Predictive cache statistics details are provided in TR-3801: Introduction to Predictive Cache Statistics.

8.2 Considering More Cache

Systems that already have some amount of Flash Cache installed do not have the ability to run PCS without first disabling Flash Cache. This often brings up the question “how do I know if more Flash Cache would help?” since disabling Flash Cache isn’t an optimal solution. Getting an idea of whether the system could benefit from additional Flash Cache is possible using the cache statistics provided by the system but ultimately comes down to the workload.

Generally more cache won’t help for situations where:

- The workload is mostly writes. Workloads dominated by writes might end up overwriting and invalidating data in the cache, meaning 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 would mean that the working set is already captured within the cache.

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

- Low hit percent: If the workload is dominated by reads, a cache of the proper size can achieve very high hit percentages.
- Data flowing through cache: If the working set is larger than cache, there will be churn of data that is cached. If more of the working set can be cached, the flow through the cache might slow, and the probability of hits might increase because data will remain in the cache longer. Data flowing through the cache can be observed by looking at the inserts and invalidates statistics.

If the indicators mentioned above are observed, it's possible that more cache will increase the amount of reads that are offloaded from disk. It's important to remember that given a single data point, it's not possible to determine the absolute additional benefit more cache will provide since the "benefit" curve is nonlinear and is dependent on variables that cannot be determined with the statistics available.

9 NetApp Performance Acceleration Module (PAM I)

Flash Cache is the second generation of NetApp controller-based caching solutions. The first generation, PAM I, has not been available since late 2010. PAM I was a PCI express card with 16GB of DRAM. PAM I operated in a similar manner to Flash Cache and had similar configuration options.

10 Conclusion

Flash Cache products provide performance acceleration to a wide variety of workloads. The configuration process is simple for most scenarios, and it can be flexible to fit specific workloads when combined with FlexShare. Both products accelerate all protocols and can benefit most Data ONTAP software features.

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

Go further, faster®

www.netapp.com