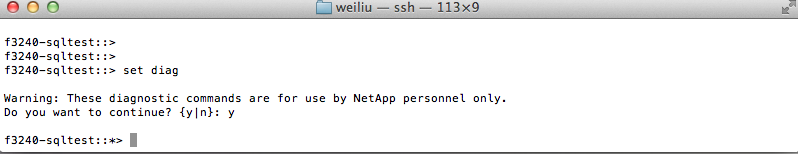
<https://community.netapp.com/t5/Technology/How-To-Set-Flash-Pool-Policies/ba-p/82160>

**CUSTOMIZING FLASH POOL’s**



In the blog posts “[Flash Pool Read Policies](https://community.netapp.com/community/netapp-blogs/pseudo_benchmark/blog/2012/07/18/flash-pool-read-policies)” and “[Flash Pool Write Policies](https://community.netapp.com/community/netapp-blogs/pseudo_benchmark/blog/2012/08/05/flash-pool-write-policies)”, I discussed what are the Flash Pool policies. But, how do you set these policies? So far, I’ve only discovered one way to do it: by using [node CLI](https://community.netapp.com/community/netapp-blogs/pseudo_benchmark/blog/2013/07/19/cluster-cli-and-node-cli). Below are some examples.

Suppose you have two database FlexVol volumes (db1\_fv, db2\_fv) and two log volumes (log1\_fv, log2\_fv). Let’s say you want to cache random reads and random writes for both database volumes; and you don’t want to cache anything for the log volumes.

First of all, in order to set Flash Pool policies, you need to use the ***diag*** mode. Figure 1 below shows how to switch to the diag mode.

Random reads and random writes for both database volumes; and you don’t want to cache anything for the log volumes.

First of all, in order to set Flash Pool policies, you need to use the ***diag*** mode. Figure 1 below shows how to switch to the diag mode.

Figure 1. Setting the ***diag*** mode.

Now, we can actually set some Flash Pool policies. Figure 2 shows how this can be done. Note that the policies are set for each volume separately.

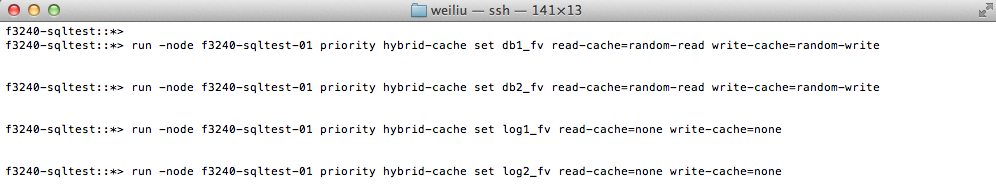


Figure 2. Setting the Flash Pool policies.

Figure 3 shows how you can verify the settings on the per-volume basis.

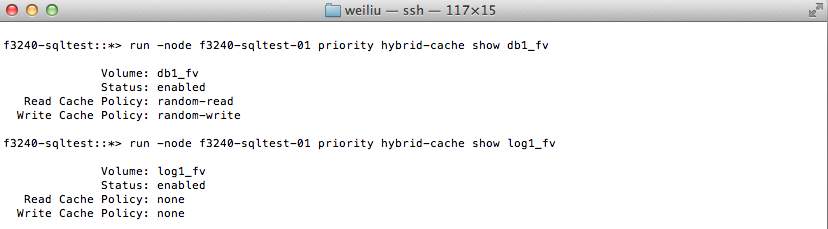


Figure 3. Checking the settings.

That’s it. In summary, for each volume, you can set a read-cache policy and a write-cache policy.  The syntax of the command is:

*priority hybrid-cache set <volume name> <read-cache>=<value> <write-cache>=<value>*

**FLASH POOL READ POLICIES**

<https://community.netapp.com/t5/Technology/How-To-Set-Flash-Pool-Policies/ba-p/82160>

In the blog “[How to Create and Use Flash Pool?](https://community.netapp.com/community/netapp-blogs/pseudo_benchmark/blog/2012/06/30/how-to-create-and-use-flash-pool)”, I outlined the steps to create and use Flash Pool. I also mentioned that you are not required to tune it manually to place the data properly, as Flash Pool will manage the data movement and placement for you automatically.

What if you enjoy performance tuning and you want to fine tune Flash Pool yourself? No problem. Flash Pool provides users with some knobs to tune. Let’s explore the four Flash Pool’s read policies below.

**1. read-cache = none.** If this policy is set on a particular volume, no data from that volume will be placed on SSDs for any host read operations from this volume thereafter. You may want to do that if you know data on this volume is infrequently used, or fast read latency from this volume is not a concern.

**2. read-cache = meta.** Metadata is data structures or entities that the controller uses to keep track of user data. This policy means only metadata (and no user data) will be placed on SSDs for host read operations. In general, the size of metadata is a fraction of the user data size. If the user datasets are too large to be effectively cached on SSDs, you may choose to use this policy.

**3. read-cache = random-read.** This policy will cause both metadata and randomly read data being placed on SSDs for accelerating subsequent host random read operations. This means if the host reads are sequential in nature, those sequential reads will not be cached on SSDs. This is the default policy, as it is likely the common use case.

**4. read-cache = random-read-write.** This policy really makes it interesting. What it does is that it’ll let you cache metadata, random reads and random writes on SSDs for accelerating subsequent host random read operations. It is categorized as a read policy. But you wonder why. Why should a read caching policy deal with random writes? I think we can probably come up with a case where a host needs to read the data soon after it has been written to the hard drives.

Note that all these policies can be set on a per-volume basis. For example, you can set **read-cache=meta** for volume1 and at the same time set **read-cache=random-read** for volume2.

**FLASH POOL WRITE POLICIES**

Flash Pool allows you to cache both random reads and random writes. To fine tune read caching, you use the Flash Pool read caching policies, as discussed in the [Flash Pool Read Policies blog](https://community.netapp.com/community/netapp-blogs/pseudo_benchmark/blog/2012/07/18/flash-pool-read-policies). By the same token, if you want to fine tune the write caching, you use the Flash Pool’s write policies. There are two such policies, as described below.

* **write-cache = none.** If this policy is set on a particular volume, no random writes to this volume will be placed on SSDs thereafter. This means effectively, write caching for this volume is disabled. All random writes will go to hard disk drive (HDD) as they would on a normal aggregate. If your workload does not do random overwrites frequently, then this policy should be a good fit.

* **write-cache = random-write.** This policy will cause frequent random overwrites being written on SSDs rather than HDDs. This provides a new and fast destination for those overwrites, thus reducing the I/O load on HDDs. The data in the write cache could also satisfy host reads if they happen to be cache hits, thus further reducing the load on HDDs. This is the default policy, as it is a target use case of Flash Pool.

These policies look good on paper. But how do I know if my workload is doing random overwrites to the same block locations frequently? Often it is not obvious or trivial to pin point the exact I/O patterns. Therefore, it is not easy to choose the best write caching policy. What one could do then is to run a couple quick tests, with **write-cache=none** followed by **write-cache=random-write**; and then compare the results.

Note that both of the policies can be set on a per-volume basis. For example, you can set **write-cache=none** for volume1 and at the same time set **write-cache=random-write** for volume2.

<https://library.netapp.com/ecmdocs/ECMP1610202/html/statistics/cache/flash-pool/show.html>

**statistics cache flash-pool show**

Flash pool throughput metrics

**Availability:**This command is available to *cluster* and *Vserver* administrators at the *admin* privilege level.

**Description**

This command continuously displays performance data for flash pool caches at a regular interval. The command output displays data in the following columns:

* Aggregate - aggregate name.
* Vserver - vserver name.
* Volume - volume name.
* Read Hit - percent of IOs serviced from a cache level.
* Write Hit - percent of IOs serviced from a cache level.
* Cache Used - percent of cache used.
* Read Blocks - read blocks.
* Write Blocks - write blocks.
* Rejects - cache rejects.

**Parameters**

**[-aggregate <text>]** - Aggregate

Selects the aggregate for which you want to display performance data.

**[-vserver <vserver name>]** - Vserver

Selects the vserver for which you want to display performance data.

**[-volume <text>]** - Volume

Selects the volume for which you want to display performance data.

**[-sort-key <text>]** - Column to Sort By

If this parameter is specified, the command displays statistics sorted by the specified column.

**-interval <integer>** - Interval

Specifies, in seconds, the interval between statistics updates. The default setting is 5 seconds.

**-iterations <integer>** - Iterations

Specifies the number of iterations the command runs before terminating. The default setting is 1. If the number is 0 (zero), the command continues to run until you interrupt it by pressing Ctrl-C.

**-max <integer>** - Maximum Number of Instances

Specifies the maximum number of flash pools to display. The default setting is 25.

**Examples**

The following example displays flash pool statistics:

cluster1::> statistics cache flash-pool show

cluster1 : 12/31/2013 16:00:04

Read Write Cache

Hit Hit Used Read Write

Aggregate Vserver Volume (%) (%) (%) Blocks Blocks Rejects

----------- -------- ------- ---- ----- ----------- ------- ------- -------

aggr1 - -total- 0 0 0 0 0 0

aggr2 vs1 vol1 0 0 0 0 0 0

[...]

**SETTING THE CACHE-RETENTION POLICIES FOR FLASH POOL AGGREGATES**

<http://docs.netapp.com/ontap-9/index.jsp?topic=%2Fcom.netapp.doc.dot-cm-psmg%2FGUID-79D614D4-B8E8-44A9-9C46-22DF11002C63.html>

Beginning with ONTAP 9.0, you can assign cache-retention policies to volumes in Flash Pool aggregates. Data in volumes with a high cache-retention policy remains in cache longer and data in volumes with a low cache-retention policy is removed sooner. This increases performance of your critical workloads by making high priority information accessible at a faster rate for a longer period of time.

**Before you begin**

You should know whether your system has any conditions that might prevent the cache-retention policy from having an impact on how long your data remains in cache.

**About this task**

The task must be done in advanced privilege mode.

**Steps**

* Verify the volume's cache-retention policy: By default the cache retention policy is normal.
* Set the cache-retention policy: **priority hybrid-cache set *volume\_name* read-cache=*read\_cache\_value* write-cache=*write\_cache\_value* cache-retention-priority=*cache\_retention\_policy*** Set *cache\_retention\_policy* to high for data that you want to remain in cache longer. Set *cache\_retention\_policy* to low for data that you want to remove from cache sooner.

Verify that the volume's cache-retention policy is changed to the option you selected.