IBM Systems MEDIA

AIX Flash Cache

Flash Cache for AIX v7.2: Flash Cache allows an LPAR to use SSDs or flash storage as a read-only cache to improve read performance for spinning disks.

By Jaqui Lynch

05/10/2017

On October 5, 2015, IBM made some significant new announcements, one of which was Flash Cache in AIX v7.2. It should be noted that AIX 7.2 only runs on POWER7 or higher servers. Since the announcement, IBM has back ported Flash Cache to AIX v7.1 tl4 SP2.

Flash Cache is also referred to as server side caching of data. It allows an LPAR to use SSDs or flash storage as a read-only cache to improve read performance for spinning disks. The cache can be significantly smaller than the data it is caching and can be direct attached or SAN based.

The supported flash memory technology for use with flash cache includes SSDs in an EXP24s drawer attached to the PCIe2/PCIe3 SAS RAID adapter with write cache, SSDs in the POWER servers system unit run by the integrated SAS controller and the IBM FlashSystem.

Once flash cache is set up, AIX decides what data is hot and stores a copy in the flash cache. The caching function can be enabled and disabled dynamically and is transparent to the workloads taking advantage of it. When caching is enabled, read requests for the target devices are sent to the caching software, which checks whether the block is in the cache. If it is, then the disk block is provided from the cache. All other reads and all writes will be sent through to the original disk.

Flash cache requires AIX v7.2, which in turn requires POWER7 or higher servers. The LPAR must have a minimum of 4GB of memory—but I would recommend much more memory than this, as the cache uses memory to keep track of read access and other things. I also would not recommend using it on small LPARs. Typically for a small LPAR you will get more benefit from adding memory or CPU. But for larger LPARs there can be significant benefits with the right workload.

In his AIXpert Blog on Flash Cache, Nigel Griffiths reminds us that there may not be a benefit if the SAN is already using internal caching for the data.

Terminology

There are several terms that you need to understand when setting up and using flash cache.

- Cache device—The SSD or flash device that will be used for caching
- Cache pool—The group of cache devices set up to be used for caching
- Cache partition—A logical cache device that exists in the cache pool
- Target device—The storage device that is being cached

In addition, there are two components you need to be aware of:

1. Cache Management—There’s a new command that is used to create, assign, destroy and report on the flash

cache. The command is /usr/sbin/cache_mgt.

2. Cache engine—This is the algorithm that determines what will be cached and that retrieves the data from the cache.

Setting up flash cache

There are several ways to implement flash cache. The flash can be direct attached to an AIX LPAR or it can be virtualized through a VIO server. Using a VIO server provides support for LPM, however, it is always possible to turn off caching prior to an LPM move, especially if the target hardware does not have a Flash cache. It should also be noted that the cache can only be provisioned to one LPAR or VIO server—as of right now it cannot be shared.

In order to test flash cache there are some prerequisites that have to be met for operating system and firmware—the key is to be current on both. Today you can only have one cache pool and one cache partition. However, there are no published limits on the pool size. For the system I was testing on, we had the following setup:

- Power System E850 LPAR
- LPAR: whole system using all 20 cores at 3.72 GHz
- Memory: 512 GB
- Flash cache: Four x 775GB SSDs—these were installed in the CEC with two on each side of the split bus. We did not have VIO servers so the flash cache was directly assigned to the LPAR.
- Disks: 88 x 250 GB NetApp provided SAN LUNs as the targets—these are all in one huge filesystem that is used by SAS. Once the data is loaded it is primarily read only.
- Our operating system was AIX 7.2.1.1

The first step is to ensure the correct LPPs are installed—these consist of bos.pfcdd and cache.mgt and can be found on the AIX 7.2 base install DVD. After installation you should see something like:

```
lslpp -l | grep Cache
  bos.pfcdd.rte          7.2.1.0  COMMITTED  Power Flash Cache
  cache.mgt.rte         7.2.1.0  COMMITTED  AIX SSD Cache Device
  bos.pfcdd.rte          7.2.1.0  COMMITTED  Power Flash Cache
  cache.mgt.rte         7.2.1.0  COMMITTED  AIX SSD Cache Device

lslpp -l | grep lash
  bos.pfcdd.rte          7.2.1.0  COMMITTED  Power Flash Cache
        7.2.1.0  COMMITTED  Common CAPI Flash Adapter
        7.2.0.0  COMMITTED  CAPI Flash Adapter Diagnostics
        7.2.0.0  COMMITTED  CAPI Flash Adapter Device
  devices.common.IBM.cflash.rte
        7.2.1.0  COMMITTED  Common CAPI Flash Device
  bos.pfcdd.rte         7.2.1.0  COMMITTED  Power Flash Cache
        7.2.1.0  COMMITTED  Common CAPI Flash Adapter
  devices.common.IBM.cflash.rte
        7.2.0.0  COMMITTED  Common CAPI Flash Device
```

If you forget to install pfcdd you will have the cache_mgt command but you won't have a caching engine so make sure these are both installed. The next step is to set up the cache pool of SSDs. Our 4 disks were hdisk2, hdisk3, hdisk8 and hdisk9 and were not in any volume group.

```
cache_mgt pool create -d hdisk2,hdisk3,hdisk8,hdisk9 -p cmpool0
```

Behind the covers: This creates a volume group called cmpool0 as well as a pool called cmpool0 consisting of 2888 PPs or 2957312MB (all the PPS across the 4 SSDs). Note: don't create the volume group directly yourself—use cache_mgt.

The next step is to create the cache partition

```
cache_mgt partition create -p cmpool0 -s 2957308M -P cm1part1

lsvg -l cmpool0
cmpool0:
LV NAME             TYPE       LPs     PPs     PVs  LV STATE      MOUNT POINT
cm1part1                jfs        2888    2888    4    closed/syncd  N/A
```

The partition it creates is a JFS partition, not JFS2. This does not matter as (per Nigel) the type is just a property string and is not used to enforce a how the disks are actually accessed.

```
lsvg -p cmpool0
cmpool0:
PV_NAME                PV STATE          TOTAL PPs   FREE PPs    FREE DISTRIBUTION
hdisk2                 active                722         0           00..00..00..00..00
hdisk3                 active                722         0           00..00..00..00..00
hdisk8                 active                722         0           00..00..00..00..00
hdisk9                 active                722         0           00..00..00..00..00
```

Above we see how the PPs are spread amongst the SSDs that were assigned.

Below is the list of disks that could be candidates for the cache pool. Cmpool0 is the pool we are using.

```
cache_mgt device list -l
hdisk0,rootvg
hdisk1,nimvg
hdisk2,cmpool0
hdisk3,cmpool0
hdisk4,ssdvg
hdisk5,ssdvg
hdisk6,rootvg
hdisk7,ssdvg
hdisk8,cmpool0
hdisk9,cmpool0
hdisk10,ssdvg
hdisk11,ssdvg
```

You can also check the pool allocations as follows:

```
cache_mgt pool list -l
cmpool0,hdisk2,hdisk3,hdisk8,hdisk9
```

At this point we can assign hdisks to be targets for caching. For my initial test I just used four disks: Assign hdisk106-109 as the source disks to be cached from

```
cache_mgt partition assign -t hdisk106 -P cm1part1
cache_mgt partition assign -t hdisk107 -P cm1part1
cache_mgt partition assign -t hdisk108 -P cm1part1
cache_mgt partition assign -t hdisk109 -P cm1part1
```

Youi will see the following responses

```
Partition cm1part1 assigned to target hdisk106
Partition cm1part1 assigned to target hdisk107
Partition cm1part1 assigned to target hdisk108
Partition cm1part1 assigned to target hdisk109
```

I then checked the partition and assignments:

```
cache_mgt partition list -l
cm1part1,2957312M,cmpool0,hdisk106,hdisk107,hdisk108,hdisk109
```

```
cache_mgt cache list
hdisk106,cm1part1,inactive
hdisk107,cm1part1,inactive
hdisk108,cm1part1,inactive
hdisk109,cm1part1,inactive
```

INACTIVE means cache is not started!

And I then checked the engine status:

```
cache_mgt engine list -l
/usr/ccs/lib/libcehandler.a,max_pool=1,max_partition=1,tgt_per_cache=unlimited,cache_per_tgt=1
```

At this point it was time to start the caching—you can start it in two ways. The first is to start it for all assigned disks, the second is to start it disk by disk. All disks

```
cache_mgt cache start -t all
```

You will see something like: All caches have been started.

Disk by disk

```
cache_mgt cache start -t hdisk106
cache_mgt cache start -t hdisk107
cache_mgt cache start -t hdisk108
cache_mgt cache start -t hdisk109
```

Similarly, you stop the caching either disk by disk or using all:

```
cache_mgt cache stop -t all
```

or

```
cache_mgt cache stop -t hdisk106
```

If you want to remove target disks then you need to stop caching of that target prior to the unassign of the disk. The unassign is done as follows:

```
cache_mgt cache stop -t hdisk106
cache_mgt partition unassign -t hdisk106
```

After my initial tests we moved over to using the full 88 disks that we wanted to cache and discovered something interesting. To list the status of the cached disks you use the following command:

```
cache_mgt cache list
```

This provides a list of the targets and their status (inactive or active). What I discovered is that the command is fine up to 76 target disks but as soon as you add the 77th disk the command takes a core dump. This is being worked on by IBM and an apar will be out shortly. All other commands work fine and this does not affect the caching functions.

Once caching is up and running you can get statistics using the cache_mgt monitor command. By default it seems to start, but the 3 versions of the command are:

```
cache_mgt monitor start
        cache_mgt monitor stop
        cache_mgt monitor get -h -s
```

The issue with the “monitor get” command is that it reports by disk. This is fine if you have three or four target disks, but when you have 88 of them it is a lot of data to look at. It would be great if they also provided an overall average at the end. Here is an example of what it showed for one of the disks while I was running the iotest.sh script from SAS:

```
ETS Device I/O Statistics -- hdisk12
Start time of Statistics -- Wed Feb  1 12:02:30 2017
------------------------------------------------------
 Read Count:                                  63299
 Write Count:                                   69692
 Read Hit Count:                               6547
 Partial Read Hit Count:                         17
 Read Bytes Xfer:                      18253348864
 Write Bytes Xfer:                      18253611008
 Read Hit Bytes Xfer:                  1676115968
 Partial Read Hit Bytes Xfer:           10100736
 Promote Read Count:               7488929792
 Promote Read Bytes Xfer:                    7142
```

As you can see the read hits were low but this is most likely because the script was only reading the data one time so there was no real benefit from caching.

Testing

Testing flash cache turned out to be a challenge. First you have to remember that it takes time to select the blocks to be cached. This means that short simple tests such as a dd are of little to no value. I did some testing with ndisk64 and also with the SAS iotest.sh program. Unfortunately, iotest.sh uses dd under the covers so the results were very patchy. I got a little carried away with the ndisk64 testing and managed to run myself out of memory, surprising considering I had 512GB. These tests are being rerun in the next couple of weeks. Flash cache was very easy to set up and I was able to show that it was working and speeding up disk access very quickly. Note that you need to let it run for a while (at least 5 minutes) in order to select the disk blocks that would benefit the most from caching. This is why you should always have a warm up time whenever you do any benchmarks and is normal.

Summary

Flash cache has great potential for heavy read workloads for improving performance. It is still in its first version so there is still room to improve some of the functionality, but overall it is well worth exploring if you have a workload that can benefit from read caching.

NOTE: Thanks to Nigel Griffiths for his edits and the information he provided to me.

About the author
Jaqui Lynch has over 38 years of experience working with a projects and OSes across vendor platforms, including IBM Z, UNIX systems and more.

Related Content

Systems management (/systems-management) New Version 7.2 Features Reflect IBM's Commitment to AIX → (https://ibmsystemsmag.com/Power-Systems/11/2015/aix7-2-announcement)
IT infrastructure (/it-infrastructure) Improve Performance and Efficiency With Hybrid and All-Flash Storage → (https://ibmsystemsmag.com/IBM-Z/01/2017/improve-performance-efficiency)
Data management (/data-management) Why Flash is the Fastest Growing Form of Data Storage → (https://ibmsystemsmag.com/IT-Strategy/03/2017/flash-data-storage)