bcache and dm-cache

Linux Block Caching Choices in Stable Upstream Kernel

A Dell Technical Whitepaper

Ahmad Ali
Charles Rose

December, 2013

Table of contents

Introduction

Modern hard disk drives (HDDs) have come a long way. They rotate faster; have higher storage density with lower error rates as compared to HDDs of past and yet they are still considered as performance bottlenecks like their predecessors. Over time, we have seen many caching schemes for performance enhancement; there are block caching solutions that use some sort of cache memory near the disk drive (either on the disk or on the storage controller) or file system level caching solutions that use host system memory. With availability of enterprise level, large yet economical flash storage technology, the block caching solutions using Solid State Drives (SSDs) appear as an acceptable performance enhancing solution for an enterprise environment. With SSD based block-caching, we can look for SSD speeds and HDD capacities - fast and big yet affordable.

In a block-caching solution, conceptually, a logical device is presented to the file-system (or applications) instead of the actual destination HDD (or other destination block device like iSCSI LUN) where data was meant to be stored. The logical device thus presented is the same size as the original physical device while the SSD used for caching can be mapped across the logical device.

The Linux open-source community has multiple generic block-level caching solutions popular among them being bcache, dm-cache, and flashcache. The Linux kernel community merged dm-cache upstream in kernel 3.9 and bcache in kernel 3.10. While flashcache is not merged upstream, it has been used in some production environments for a while now. Recently released Fedora 20 includes both bcache and dm-cache. This paper shows the steps for configuring bcache and dm-cache using Fedora 20 distro.

1 dm-cache (Device Mapper Cache)

dm-cache is a device mapper target first committed to kernel-3.9. It uses I/O scheduling and cache management techniques optimized for flash-based SSDs. The device mapper target (dm-cache) reuses the metadata library used in the thin-provisioning library. Both write-back and write-through are supported by dm-cache with write-back being the default mode.

The virtual cache device created by the dm-cache is built using three physical devices called *origin device*, *cache device* and *metadata device* (figure-1).

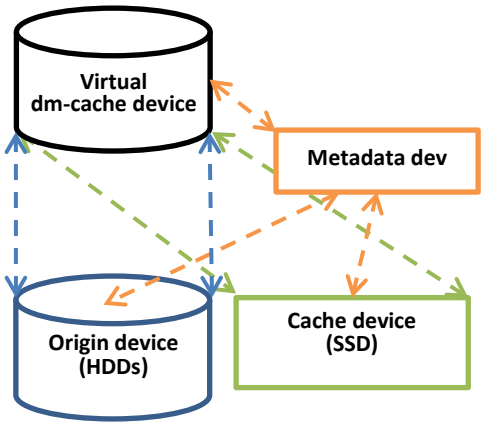

Figure-1: dm-cache

The *origin device* is the actual (slower) destination storage device while the *cache device* is the faster device being used to temporarily store the user data. The *metadata device* is recording blocks placement, their dirty flags, and other internal data required by a policy.

1.1 Hardware Setup

The example hardware setup has /dev/sdb1 as the origin device while /dev/sdc is the SSD to be used for both the cache and metadata devices.

1.2 Software Setup

The system is running Fedora20 installed with the default set of packages; including the device-mapper package. As of this writing, there are a couple of open source projects that assist with dm-cache setup. However, we will use the lvm2(2) and dmsetup(8) utilities that are already part of the fedora distribution.

Note: While dmsetup(8) is used here to setup and configure dm-cache, we expect lvm2(2) to eventually become the user-space tool for dm-cache configuration.

1.3 Configuration Steps

1. Create a 100GB partition (/dev/sdb1) on the hard disk (origin device),
   ```
   # parted -a optimal /dev/sdb mkpart primary 1 100G
   ```

2. We are using our SSD (/dev/sdc) for both block cache and metadata. For our setup, we created a 10GB partition on this SSD.
   ```
   # parted -a optimal /dev/sdc mkpart primary 1 10G
   ```

3. Prepare the *cache device* for LVM on the cache partition /dev/sdc1
   ```
   # vgcreate cache /dev/sdc1
   ```

4. Create a "meta" volume for metadata using 5% and "cache" volume using the rest of the space.
   ```
   # lvcreate cache –n meta –l 5%FREE
   # lvcreate cache -n block -l 100%FREE
   ```

Here we arbitrarily chose the metadata device to be just 5% of the allocated space on the SSD. The metadata size depends on the number of cache blocks and the size of cache blocks are a configurable parameter. The following summarizes the metadata size discussion for more information (see http://en.wikipedia.org/wiki/Dm-cache),

Having caching extents bigger than the HDD sectors is a compromise between the size of metadata, and the possibility for wasting cache space. Having too small of a caching extents increases the metadata size, both in the metadata device and in kernel memory. Having too large of a metadata extents increases the amount of wasted cache space, due to the whole extents being cached even in case of high hit rates only for some of their parts.

5. Get the size of the origin device
   ```
   # blockdev --getsz /dev/sdb1
   ```

6. Create a mapping between the block cache device and the origin device (/dev/sdb1).
   ```
   # dmsetup create dmcache0 --table '0 195309568 cache /dev/cache/meta\
         /dev/cache/block /dev/sdb1 512 1 writeback default 0'
   ```

Note: Replace `writeback` with `writethrough` if that is desired.

7. We now have a new virtual block device at /dev/mapper/dmcache0 that can be used in place of origin device (/dev/sdb1) on which we can create a filesystem and mount it for use,
   ```
   # mkfs.ext4 /dev/mapper/dmcache0
   # mount /dev/mapper/dmcache0 /mnt/data
   ```

8. The cache status can be viewed with:
   ```
   # dmsetup –v status dmcache0
   ```

2 bcache

bcache was first committed to kernel-3.10. It is designed around the unique characteristics of flash-based SSDs and uses a hybrid btree/log to track cached extents. It is designed to avoid random writes at all costs. bcache fills up an erase block sequentially and then issues a discard before reusing it. Both write-through and write-back caching are supported. Write-back defaults to off, but can be switched on and off arbitrarily at runtime.

The virtual bcache device is created by attaching the *backing device* and *caching device* (figure-2).

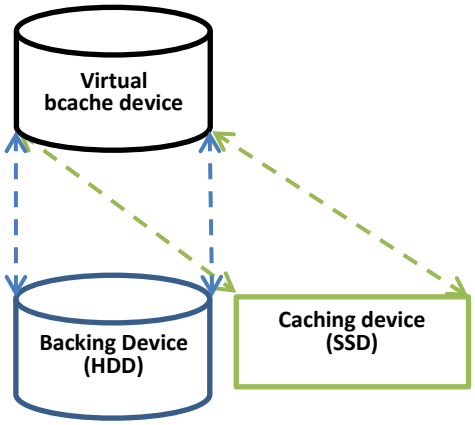

Figure-2: bcache

The *backing device* is the actual (slower) destination storage device while the *caching device* is the faster device. The *backing device* has to be formatted as a bcache block device; an existing formatted partition can't be used with bcache. (One can try blocks to-bcache to do an in-place conversion).

2.1 Hardware Setup

The example hardware setup has /dev/sdb1 as the *backing device* while /dev/sdc is the SSD we used for the *caching device*.

2.2 Software Setup

The system is running Fedora20 installed with the default set of packages. The bcache-tools package makes the configuration and use of bcache a lot easier but the bcache tools package is not native to Fedora20. Bcache-tools can be installed from any Fedora20 repository.

2.3 Configuration Steps

1. Install the bcache-tools package
   ```
   # yum install bcache-tools
   ```

2. We are using our SSD (/dev/sdc) for the *caching device*. For our setup, we created a 10GB partition on this SSD.
   ```
   # parted -a optimal /dev/sdc mkpart primary 1 10G
   ```

Note: Ensure /dev/sdc1 and /dev/sdb1 are "clean" (in case you had an older setup):
```
# wipefs –a /dev/sdb1; wipefs –a /dev/sdc1
```

3. Setup the bcache device by creating and attaching the caching device (/dev/sdc1) with a backing device (/dev/sdb1).
   ```
   # make-bcache –C /dev/sdc1 –B /dev/sdb1
   ```

4. We now have a new virtual block device at /dev/bcache0. We can use this bcache device to create a filesystem, mount,
   ```
   # mkfs.ext4 /dev/bcache0
   # mount /dev/bcache0 /mnt/data
   ```

5. We can monitor cache statistics and health with:
   ```
   # bcache-status –a
   ```

2.3.1 Enabling writeback

bcache supports multiple modes for caching writes. Write-back defaults to off, but can be switched on and off arbitrarily at runtime.

- The following shows that `writethrough` is the active write cache mode,
  ```
  # cat /sys/block/bcache0/bcache/cache_mode
  [writethrough] writeback writearound none
  ```

- To enable the `writeback` as the active write cache mode,
  ```
  # echo writeback > /sys/block/bcache0/bcache/cache_mode
  ```

- The bcache-status tool also shows the active write cache mode,
  ```
  # bcache-status | grep 'Cache Mode'
  Cache Mode                 writethrough [writeback] writearound none
  ```

References

- https://www.kernel.org/doc/Documentation/device-mapper/cache.txt
- https://www.kernel.org/doc/Documentation/device-mapper/cache-policies.txt
- http://bcache.evilpiepirate.org/