zodcache

I. So what is this anyway?

zodcache is a device mapper (dm-cache)-based block device caching mechanism for Linux. It provides a simple interface to create and auto-start cache dm-cache devices.

A. How is this different from LVM cache?

- LVM cache uses LVM physical volumes as component devices. Both the fast (SSD) and slow (HDD) physical volumes must be part of the volume group that will contain the cached logical volume. Many people find this counterintuitive, and it eliminates some of the convenience that LVM usually provides. (Now I have to specify the PV(s) that I want to use when creating/extending an LV in this VG.)
- LVM cache uses dm-cache to created cached **logical volumes**. Although zodcache also uses dm-cache, zodcache devices are not logical volumes. (zodcache devices can be used as LVM physical volumes.) In this regard, zodcache is similar to bcache.

B. How is this different from bcache?

- bcache is an entirely separate kernel subsystem; it does not use dm-cache at all.
- If your distribution's kernel is built without bcache support (such as Red Hat Enterprise Linux 7 and derivatives), you cannot use bcache without running a rebuilt/custom/3rd-party kernel. zodcache should work on any distribution that supports dm-cache and udev (any reasonably modern distribution).

II. How does it work?

zodcache is based on dm-cache, so a zodcache device is built on the same 3 component devices as any other dm-cache device – the origin device, the cache device, and the metadata device. For convenience, the cache and metadata devices can reside on a single underlying block device (a “combined cache” device).

Each of the 2 (if using a combined cache) or 3 underlying devices has a zodcache superblock at offset 0. The superblock identifies the device as a zodcache component and describes the device.

Here is an example of the superblock on an origin device:

```
# zcdump /dev/sda3
magic:          20DCAC8E8EACDC20
checksum:       5945361681611887526
version:        0
size:           128
type:           origin
dev_major:      8
uuid:           4e71cacb-3d28-4ef0-acbe-d4710f3207cb
block_size:     256 KiB
cache_mode:     writeback
o_offset:       4 KiB
o_size:         94,063,612 KiB
c_offset:       0 bytes
c_size:         0 bytes
md_offset:      0 bytes
md_size:        0 bytes
```

When **zcstart** processes this device (/dev/sda3), it will create a dm-linear device that maps the origin area (94,063,612 KiB starting at 4 KiB). This linear device will be used as the dm-cache origin.

```
# ls /dev/mapper/*-origin
/dev/mapper/zodcache-4e71cacb-3d28-4ef0-acbe-d4710f3207cb-origin
```

And here is a superblock on a combined cache device:

```
# zcdump /dev/sdb1
magic:          20DCAC8E8EACDC20
checksum:       1112998762071856021
version:        0
size:           128
type:           combined
dev_major:      8
uuid:           4e71cacb-3d28-4ef0-acbe-d4710f3207cb
block_size:     256 KiB
cache_mode:     writeback
o_offset:       0 bytes
o_size:         0 bytes
c_offset:       4,864 KiB
c_size:         10,479,872 KiB
md_offset:      4 KiB
md_size:        4,860 KiB
```

Because this is a combined cache device, **zcstart** will create 2 dm-linear devices, which will act as the dm-cache metadata and cache devices.

```
# ls /dev/mapper/*-{cache,metadata}
/dev/mapper/zodcache-4e71cacb-3d28-4ef0-acbe-d4710f3207cb-cache
/dev/mapper/zodcache-4e71cacb-3d28-4ef0-acbe-d4710f3207cb-metadata
```

Once all 3 dm-linear devices are present, **zcstart** will assemble the dm-cache device.

```
# ls /dev/mapper/zodcache-*
/dev/mapper/zodcache-4e71cacb-3d28-4ef0-acbe-d4710f3207cb
/dev/mapper/zodcache-4e71cacb-3d28-4ef0-acbe-d4710f3207cb-cache
/dev/mapper/zodcache-4e71cacb-3d28-4ef0-acbe-d4710f3207cb-metadata
/dev/mapper/zodcache-4e71cacb-3d28-4ef0-acbe-d4710f3207cb-origin
```

When the dm-cache device is assembled, udev treat it like any other block device, so LVM physical volumes, etc., will be processed appropriately.

```
# vgdisplay -v
    Using volume group(s) on command line.
  --- Volume group ---
  VG Name               vg_root
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  2
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               1
  Max PV                0
  Cur PV                1
  Act PV                1
  VG Size               89.70 GiB
  PE Size               4.00 MiB
  Total PE              22964
  Alloc PE / Size       2560 / 10.00 GiB
  Free  PE / Size       20404 / 79.70 GiB
  VG UUID               3MrmTQ-q8GQ-h0qQ-eOT7-lXHp-JcOX-R9DcXw

  --- Logical volume ---
  LV Path                /dev/vg_root/lv_root
  LV Name                lv_root
  VG Name                vg_root
  LV UUID                8wqMw3-TDeM-22Fk-3uXb-DNQN-U51l-irIle6
  LV Write Access        read/write
  LV Creation host, time localhost.localdomain, 2015-12-16 22:49:38 -0600
  LV Status              available
  # open                 1
  LV Size                10.00 GiB
  Current LE             2560
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     8192
  Block device           253:4

  --- Physical volumes ---
  PV Name               /dev/mapper/zodcache-4e71cacb-...-d4710f3207cb
  PV UUID               JRbWaB-w61B-zt7z-iLoi-HsFJ-oBaY-9e8R9c
  PV Status             allocatable
  Total PE / Free PE    22964 / 20404
```

III. Cool! How do I use it?

A. Red Hat Enterprise Linux (and derivatives) / Fedora

1. Ensure that the **rpm-build**, **gcc**, **device-mapper-devel**, and **libuuid-devel** packages are installed.

2. Download the most recent tarball (currently zodcache-0.0.1.tar.gz) from github.

3. Build a binary RPM.

   ```
   $ rpmbuild -tb  zodcache-0.0.1.tar.gz
   ```

4. Install it.

5. Initialize the component devices. For this example, /dev/sda3 is the (large, slow) origin device and /dev/sdb1 is the (small, fast) cache device.

   ```
   # mkzc -o /dev/sda3 -c /sdb1
   b63032ed-96b8-48b5-9981-009cc3da6607
   ```

   mkzc (which really needs a help message) also accepts these other options:

   - `-m` *`METADATA_DEVICE`* – If not specified a combined cache device is created.
   - `-b` *`BLOCK_SIZE`* – Must be a multiple of 32 KiB between 32 Kib and 1 GiB; K, M, and G suffixes are accepted. The default block size is 256 KiB.
   - `-M` *`CACHE_MODE`* – **writeback**, **writethrough**, or **passthrough**. The default cache mode is writeback.

6. (Optional) Examine the superblocks that **mkzc** created.

   ```
   # zcdump /dev/sda3
   # zcdump /dev/sdb1
   ```

7. Start the cache devices (cache and metadata) but don't start the origin device.

   ```
   # zcstart /dev/sdb1
   ```

8. Clear the metadata device. (**mkzc** should probably do this.)

   ```
   # dd if=/dev/zero \
         of=/dev/mapper/zodcache-b63032ed-...-009cc3da6607-metadata
   ```

9. Start the origin device. This will also assemble the zodcache device.

   ```
   # zcstart /dev/sda3
   ```

10. Verify that the zodcache device has been assembled.

    ```
    # ls /dev/mapper/zodcache-b63032ed-96b8-48b5-9981-009cc3da6607
    ```

11. Reboot and verify that the zodcache device was assembled by udev.

B. Other distributions

zodcache should work on any distribution that includes dm-cache and udev. Below is a list of the files in the binary RPM, along with information on how to build them and what they do.

- **`/usr/sbin/mkzc`** – Utility to write zodcache superblocks on component devices. Build with:

  ```
  gcc -O3 -Wall -Wextra -o mkzc mkzc.c lib.c -luuid
  ```

- **`/usr/sbin/zcdump`** – Utility to examine the zodcache superblock on a component device. Build with:

  ```
  gcc -O3 -Wall -Wextra -o zcdump zcdump.c lib.c
  ```

- **`/usr/sbin/zcstart`** – Utility to probe/start component devices and assemble the zodcache device. Build with:

  ```
  gcc -O3 -Wall -Wextra -o zcstart zcstart.c lib.c -ldevmapper
  ```

- **`/usr/lib/udev/rules.d/69-zodcache.rules`** – udev rules which run **zcstart** in "udev mode" on (almost) all block devices. In this mode, **zcstart** silently exits when it encounters a non-zodcache device, and error messages are logged instead of printed.
- **`/usr/lib/dracut/modules.d/90zodcache/module-setup.sh`** – dracut module to support root filesystem on zodcache devices.
- **`/etc/dracut.conf.d/50-zodcache.conf`** – Edit this file to enable the dracut module.

C. Did I hear you say root filesystem?

Yes. Enabling the dracut module (by editing `/etc/dracut.conf.d/50-zodcache.conf)` will cause dracut to include the files required to start zodcache devices in any initramfs that it creates. The following steps provide an overview of moving a Red Hat Enterprise Linux (or derivative) or Fedora system onto a zodcache-based device.

1. Create the zodcache device for your root filesystem (or a logical volume on a zodcache-backed volume group, etc.).
2. Reboot with live media that matches your installed operating system (CentOS live media should work for Red Hat Enterprise Linux-derived systems).
3. Mount your existing root filesystem and cd to the directory that contains the **zcstart** utility. For example:

   ```
   # mkdir /mnt/sysimage
   # mount -o ro /dev/sda2 /mnt/sysimage
   # cd /mnt/sysimage/usr/sbin
   ```

4. Manually start the zodcache device.

   ```
   # ./zcstart /dev/sda3
   # ./zcstart /dev/sdb1
   ```

5. Unmount the existing root filesystem.

   # umount /mnt/sysimage

6. Copy the root filesystem onto the zodcache device.

   ```
   # dd if=/dev/sda2 \
         of=/dev/mapper/zodcache-b63032ed-96b8-48b5-9981-009cc3da6607
   ```

7. Mount the new root filesystem and other filesystems required to create a chroot environment.

   ```
   # mount /dev/mapper/zodcache-b63032ed-96b8-48b5-9981-009cc3da6607 \
         /mnt/sysimage
   # mount /dev/sda1 /mnt/sysimage/boot
   # for FS in dev proc run sys tmp ; \
         do mount -o bind /$FS /mnt/sysimage/$FS ; done
   ```

8. chroot into the new root filesystem.

   ```
   # chroot /mnt/sysimage
   ```

9. Enable the dracut module by editing `/etc/dracut.conf.d/50-zodcache.conf`.

10. Rebuild the initramfs, for the most recent installed kernel.

    ```
    # ls /boot/initramfs*
    /boot/initramfs-3.10.0-327.3.1.el7.x86_64.img
    /boot/initramfs-3.10.0-327.el7.x86_64.img
    # dracut -f /boot/initramfs-3.10.0-327.3.1.el7.x86_64.img \
          3.10.0-327.3.1.el7.x86_64
    ```

11. Verify that the zodcache module was included in the initramfs.

    ```
    # lsinitrd /boot/initramfs-3.10.0-327.3.1.el7.x86_64.img | \
          grep zodcache
    zodcache
    -rw-r--r--   1 root     root ... usr/lib/udev/rules.d/69-zodcache.rules
    ```

12. Edit `/etc/fstab` and update the root filesystem line.

13. Modify the kernel parameters in the bootloader configuration (`/boot/grub/grub.conf`, `/etc/default/grub`, etc.) to reflect the new location of the root filesystem. Note that dracut does not require any specific parameters to start the zodcache device; the udev rule (and **zcstart** utility) in the initramfs should accomplish that. If necessary, run any command required to make the new bootloader configuration effective (such as grub2-mkconfig).

14. Exit from the chroot environment and unmount the filesystems.

    ```
    # exit
    # for FS in boot dev proc run sys tmp ; do umount /mnt/sysimage/$FS ; done
    # umount /mnt/sysimage
    ```

15. Reboot. If the initramfs is unable to locate the root filesystem, use the dracut emergency shell to investigate the cause. You should be able to start the zodcache device manually with the **zcstart** utility (if it was correctly included in the initramfs).