

NAME

lvmcache — LVM caching

DESCRIPTION

lvm(8) includes two kinds of caching that can be used to improve the performance of a Logical Volume (LV). Typically, a smaller, faster device is used to improve i/o performance of a larger, slower LV. To do this, a separate LV is created from the faster device, and then the original LV is converted to start using the fast LV.

The two kinds of caching are:

- A read and write hot-spot cache, using the dm-cache kernel module. This cache is slow moving, and adjusts the cache content over time so that the most used parts of the LV are kept on the faster device. Both reads and writes use the cache. LVM refers to this using the LV type **cache**.
- A streaming write cache, using the dm-writocache kernel module. This cache is intended to be used with SSD or PMEM devices to speed up all writes to an LV. Reads do not use this cache. LVM refers to this using the LV type **writocache**.

USAGE

Both kinds of caching use similar lvm commands:

1. Identify main LV that needs caching

A main LV exists on slower devices.

```
$ lvcreate -n main -L Size vg /dev/slow
```

2. Identify fast LV to use as the cache

A fast LV exists on faster devices. This LV will be used to hold the cache.

```
$ lvcreate -n fast -L Size vg /dev/fast
```

```
$ lvs -a
```

```
LV Attr Type Devices
```

```
fast -wi----- linear /dev/fast
```

```
main -wi----- linear /dev/slow
```

3. Start caching the main LV

To start caching the main LV using the fast LV, convert the main LV to the desired caching type, and specify the fast LV to use:

using dm-cache:

```
$ lvconvert --type cache --cachevol fast vg/main
```

using dm-writocache:

```
$ lvconvert --type writocache --cachevol fast vg/main
```

using dm-cache (with cachepool):

```
$ lvconvert --type cache --cachepool fast vg/main
```

4. Display LVs

Once the fast LV has been attached to the main LV, lvm reports the main LV type as either **cache** or **write-cache** depending on the type used. While attached, the fast LV is hidden, and renamed with a `_cvol` or `_cpool` suffix. It is displayed by `lvs -a`. The `_corig` or `_wcorig` LV represents the original LV without the cache.

using dm-cache:

```
$ lvs -a
LV      Pool      Type  Devices
main    [fast_cvol] cache  main_corig(0)
[fast_cvol]      linear /dev/fast
[main_corig]     linear /dev/slow
```

using dm-writecache:

```
$ lvs -a
LV      Pool      Type  Devices
main    [fast_cvol] writecache main_wcorig(0)
[fast_cvol]      linear /dev/fast
[main_wcorig]     linear /dev/slow
```

using dm-cache (with cachepool):

```
$ lvs -a
LV      Pool      Type  Devices
main    [fast_cpool] cache  main_corig(0)
[fast_cpool]      cache-pool fast_pool_cdata(0)
[fast_cpool_cdata]      linear /dev/fast
[fast_cpool_cmeta]      linear /dev/fast
[main_corig]     linear /dev/slow
```

5. Use the main LV

Use the LV until the cache is no longer wanted, or needs to be changed.

6. Stop caching

To stop caching the main LV, separate the fast LV from the main LV. This changes the type of the main LV back to what it was before the cache was attached.

```
$ lvconvert --splitcache vg/main
```

```
$ lvs -a
LV  VG Attr      Type  Devices
fast vg -wi----- linear /dev/fast
main vg -wi----- linear /dev/slow
```

OPTIONS

option args

--cachevol *LV*

Pass this option a standard LV. With a cachevol, cache data and metadata are contained within the single LV. This is used with dm-writecache or dm-cache.

--cachepool *CachePoolLV|LV*

Pass this option a cache pool object. With a cache pool, lvm places cache data and cache metadata on different LVs. The two LVs together are called a cache pool. This permits specific placement of data and metadata. A cache pool is represented as a special type of LV that cannot be used directly. (If a standard LV is passed to this option, lvm will first convert it to a cache pool by combining it with another LV to use for metadata.) This can be used with dm-cache.

dm-cache block size

A cache pool will have a logical block size of 4096 bytes if it is created on a device with a logical block size of 4096 bytes.

If a main LV has logical block size 512 (with an existing xfs file system using that size), then it cannot use a cache pool with a 4096 logical block size. If the cache pool is attached, the main LV will likely fail to mount.

To avoid this problem, use a mkfs option to specify a 4096 block size for the file system, or attach the cache pool before running mkfs.

dm-writecache block size

The dm-writecache block size can be 4096 bytes (the default), or 512 bytes. The default 4096 has better performance and should be used except when 512 is necessary for compatibility. The dm-writecache block size is specified with `--cachesettings block_size=4096|512` when caching is started.

When a file system like xfs already exists on the main LV prior to caching, and the file system is using a block size of 512, then the writecache block size should be set to 512. (The file system will likely fail to mount if writecache block size of 4096 is used in this case.)

Check the xfs sector size while the fs is mounted:

```
$ xfs_info /dev/vg/main
Look for sectsz=512 or sectsz=4096
```

The writecache block size should be chosen to match the xfs sectsz value.

It is also possible to specify a sector size of 4096 to mkfs.xfs when creating the file system. In this case the

writecache block size of 4096 can be used.

dm-writecache settings

Tunable parameters can be passed to the dm-writecache kernel module using the `--cachesettings` option when caching is started, e.g.

```
$ lvconvert --type writecache --cachevol fast \
  --cachesettings 'high_watermark=N writeback_jobs=N' vg/main
```

Tunable options are:

- `high_watermark = <count>`

Start writeback when the number of used blocks reach this watermark

- `low_watermark = <count>`

Stop writeback when the number of used blocks drops below this watermark

- `writeback_jobs = <count>`

Limit the number of blocks that are in flight during writeback. Setting this value reduces writeback throughput, but it may improve latency of read requests.

- `autocommit_blocks = <count>`

When the application writes this amount of blocks without issuing the FLUSH request, the blocks are automatically committed.

- `autocommit_time = <milliseconds>`

The data is automatically committed if this time passes and no FLUSH request is received.

- `fua = 0|1`

Use the FUA flag when writing data from persistent memory back to the underlying device. Applicable only to persistent memory.

- `nofua = 0|1`

Don't use the FUA flag when writing back data and send the FLUSH request afterwards. Some underlying devices perform better with fua, some with nofua. Testing is necessary to determine which. Applicable only to persistent memory.

dm-cache with separate data and metadata LVs

When using dm-cache, the cache metadata and cache data can be stored on separate LVs. To do this, a

"cache pool" is created, which is a special LV that references two sub LVs, one for data and one for metadata.

To create a cache pool from two separate LVs:

```
$ lvcreate -n fast -L DataSize vg /dev/fast1
$ lvcreate -n fastmeta -L MetadataSize vg /dev/fast2
$ lvconvert --type cache-pool --poolmetadata fastmeta vg/fast
```

Then use the cache pool LV to start caching the main LV:

```
$ lvconvert --type cache --cachepool fast vg/main
```

A variation of the same procedure automatically creates a cache pool when caching is started. To do this, use a standard LV as the `--cachepool` (this will hold cache data), and use another standard LV as the `--poolmetadata` (this will hold cache metadata). LVM will create a cache pool LV from the two specified LVs, and use the cache pool to start caching the main LV.

```
$ lvcreate -n fast -L DataSize vg /dev/fast1
$ lvcreate -n fastmeta -L MetadataSize vg /dev/fast2
$ lvconvert --type cache --cachepool fast --poolmetadata fastmeta vg/main
```

dm-cache cache modes

The default dm-cache cache mode is "writethrough". Writethrough ensures that any data written will be stored both in the cache and on the origin LV. The loss of a device associated with the cache in this case would not mean the loss of any data.

A second cache mode is "writeback". Writeback delays writing data blocks from the cache back to the origin LV. This mode will increase performance, but the loss of a cache device can result in lost data.

With the `--cachemode` option, the cache mode can be set when caching is started, or changed on an LV that is already cached. The current cache mode can be displayed with the `cache_mode` reporting option:

```
lvs -o+cache_mode VG/LV
```

lvm.conf(5) allocation/cache_mode

defines the default cache mode.

```
$ lvconvert --type cache --cachevol fast \
  --cachemode writethrough vg/main
```

dm-cache chunk size

The size of data blocks managed by dm-cache can be specified with the `--chunksize` option when caching is started. The default unit is KiB. The value must be a multiple of 32KiB between 32KiB and 1GiB.

Using a chunk size that is too large can result in wasteful use of the cache, in which small reads and writes cause large sections of an LV to be stored in the cache. However, choosing a chunk size that is too small can result in more overhead trying to manage the numerous chunks that become mapped into the cache. Overhead can include both excessive CPU time searching for chunks, and excessive memory tracking chunks.

Command to display the chunk size:

lvs -o+chunksize VG/LV

lvm.conf(5) cache_pool_chunk_size

controls the default chunk size.

The default value is shown by:

lvmconfig --type default allocation/cache_pool_chunk_size

dm-cache cache policy

The dm-cache subsystem has additional per-LV parameters: the cache policy to use, and possibly tunable parameters for the cache policy. Three policies are currently available: "smq" is the default policy, "mq" is an older implementation, and "cleaner" is used to force the cache to write back (flush) all cached writes to the origin LV.

The older "mq" policy has a number of tunable parameters. The defaults are chosen to be suitable for the majority of systems, but in special circumstances, changing the settings can improve performance.

With the `--cachepolicy` and `--cachesettings` options, the cache policy and settings can be set when caching is started, or changed on an existing cached LV (both options can be used together). The current cache policy and settings can be displayed with the `cache_policy` and `cache_settings` reporting options:

lvs -o+cache_policy,cache_settings VG/LV

Change the cache policy and settings of an existing LV.

```
$ lvchange --cachepolicy mq --cachesettings \
    'migration_threshold=2048 random_threshold=4' vg/main
```

lvm.conf(5) allocation/cache_policy

defines the default cache policy.

lvm.conf(5) allocation/cache_settings

defines the default cache settings.

dm-cache spare metadata LV

See [lvmthin\(7\)](#) for a description of the "pool metadata spare" LV. The same concept is used for cache pools.

dm-cache metadata formats

There are two disk formats for dm-cache metadata. The metadata format can be specified with `--cachemetadadataformat` when caching is started, and cannot be changed. Format **2** has better performance; it is more compact, and stores dirty bits in a separate btree, which improves the speed of shutting down the cache. With **auto**, lvm selects the best option provided by the current dm-cache kernel module.

mirrored cache device

The fast LV holding the cache can be created as a raid1 mirror so that it can tolerate a device failure. (When using dm-cache with separate data and metadata LVs, each of the sub-LVs can use raid1.)

```
$ lvcreate -n main -L Size vg /dev/slow
$ lvcreate --type raid1 -m 1 -n fast -L Size vg /dev/fast1 /dev/fast2
$ lvconvert --type cache --cachevol fast vg/main
```

dm-cache command shortcut

A single command can be used to create a cache pool and attach that new cache pool to a main LV:

```
$ lvcreate --type cache --name Name --size Size VG/LV [PV]
```

In this command, the specified LV already exists, and is the main LV to be cached. The command creates a new cache pool with the given name and size, using the optionally specified PV (typically an `ssd`). Then it attaches the new cache pool to the existing main LV to begin caching.

(Note: ensure that the specified main LV is a standard LV. If a cache pool LV is mistakenly specified, then the command does something different.)

(Note: the `type` option is interpreted differently by this command than by normal `lvcreate` commands in which `--type` specifies the type of the newly created LV. In this case, an LV with type `cache-pool` is being created, and the existing main LV is being converted to type `cache`.)

SEE ALSO

lvm.conf(5), **lvchange(8)**, **lvcreate(8)**, **lvdisplay(8)**, **lvextend(8)**, **lvremove(8)**, **lvrename(8)**, **lvresize(8)**, **lvs(8)**, **vgchange(8)**, **vgmerge(8)**, **vgreduce(8)**, **vgsplit(8)**