NAME

lvmcache — LVM caching

DESCRIPTION

The **cache** logical volume type uses a small and fast LV to improve the performance of a large and slow LV. It does this by storing the frequently used blocks on the faster LV. LVM refers to the small fast LV as a **cache pool LV**. The large slow LV is called the **origin LV**. Due to requirements from dm-cache (the kernel driver), LVM further splits the cache pool LV into two devices - the **cache data LV** and **cache metadata LV**. The cache data LV is where copies of data blocks are kept from the origin LV to increase speed. The cache metadata LV holds the accounting information that specifies where data blocks are stored (e.g. on the origin LV or on the cache data LV). Users should be familiar with these LVs if they wish to create the best and most robust cached logical volumes.

Cache Terms

origin LV	OriginLV	large slow LV
cache data LV	CacheDataLV	small fast LV for cache pool data
cache metadata LV	CacheMetaLV	small fast LV for cache pool metadata
cache pool LV	CachePoolLV	CacheDataLV + CacheMetaLV
cache LV	CacheLV	OriginLV + CachePoolLV

Cache Usage

The primary method for using a cache type logical volume:

0. create OriginLV

Create an LV or identify an existing LV to be the origin LV.

lvcreate −n OriginLV −L LargeSize VG SlowPVs

Example

```
# lvcreate −n lvol0 −L 100G vg
```

1. create CacheDataLV

Create the cache data LV. This LV will hold data blocks from the OriginLV. The size of this LV is the size of the cache and will be reported as the size of the cache pool LV.

lvcreate −n CacheDataLV −L CacheSize VG FastPVs

Example

```
# lvcreate −n cache0 −L 10G vg /dev/fast
```

2. create CacheMetaLV

Create the cache metadata LV. This LV will hold cache pool metadata. The size of this LV should be 1000 times smaller than the cache data LV, with a minimum size of 8MiB.

lvcreate −n CacheMetaLV −L MetaSize VG FastPVs

Example

```
# lvcreate −n cache0meta −L 12M vg /dev/fast
# lvs -a vg
  LV         VG   Attr       LSize   Pool Origin
  cache0     vg   -wi-a-----  10.00g
  cache0meta vg   -wi-a-----  12.00m
  lvol0      vg   -wi-a----- 100.00g
```

3. create CachePoolLV

Combine the data and metadata LVs into a cache pool LV. The behavior of the cache pool LV can be set in this step.
CachePoolLV takes the name of CacheDataLV.
CacheDataLV is renamed CachePoolLV_cdata and becomes hidden.
CacheMetaLV is renamed CachePoolLV_cmeta and becomes hidden.

lvconvert −−type cache-pool −−poolmetadata VG/CacheMetaLV
VG/CacheDataLV

Example

```
# lvconvert −−type cache−pool −−poolmetadata vg/cache0meta vg/cache0

# lvs -a vg
  LV              VG   Attr       LSize   Pool Origin
  cache0          vg   Cwi---C---  10.00g
  [cache0_cdata]  vg   Cwi-------  10.00g
  [cache0_cmeta]  vg   ewi-------  12.00m
  lvol0           vg   -wi-a----- 100.00g
```

4. create CacheLV

Create a cache LV by linking the cache pool LV to the origin LV. The user accessible cache LV takes the name of the origin LV, while the origin LV becomes a hidden LV with the name OriginLV_corig. This can be done while the origin LV is in use.
CacheLV takes the name of OriginLV.
OriginLV is renamed OriginLV_corig and becomes hidden.

lvconvert −−type cache −−cachepool VG/CachePoolLV VG/OriginLV

Example

```
# lvconvert −−type cache −−cachepool vg/cache0 vg/lvol0

# lvs -a vg
  LV              VG   Attr       LSize   Pool   Origin
  cache0          vg   Cwi---C---  10.00g
  [cache0_cdata]  vg   Cwi-ao----  10.00g
  [cache0_cmeta]  vg   ewi-ao----  12.00m
  lvol0           vg   Cwi-a-C--- 100.00g cache0 [lvol0_corig]
  [lvol0_corig]   vg   -wi-ao---- 100.00g
```

Cache Removal

Split a cache pool LV off of a cache LV

A cache pool LV can be disconnected from a cache LV, leaving an unused cache pool LV, and an uncached origin LV. This command writes back data from the cache pool to the origin LV when necessary.

lvconvert --splitcache VG/CacheLV

Removing a cache pool LV without removing its linked origin LV

This writes back data from the cache pool to the origin LV when necessary, then removes the cache pool LV, leaving the uncached origin LV.

lvremove VG/CachePoolLV

An alternative command that also disconnects the cache pool from the cache LV, and deletes the cache pool:

lvconvert --uncache VG/CacheLV

Example

```
# lvs vg
  LV     VG   Attr       LSize   Pool   Origin
  cache0 vg   Cwi---C---  10.00g
  lvol0  vg   Cwi-a-C--- 100.00g cache0 [lvol0_corig]

# lvremove vg/cache0

# lvs vg
  LV    VG   Attr       LSize   Pool Origin
  lvol0 vg   -wi-a----- 100.00g
```

Removing a cache LV: both origin LV and the cache pool LV

Removing a cache LV removes both the origin LV and the linked cache pool LV.

lvremove VG/CacheLV

Cache Topics

Tolerate device failures in a cache pool LV

Users who are concerned about the possibility of failures in their fast devices that could lead to data loss might consider making their cache pool sub-LVs redundant.

```
0. Create an origin LV we wish to cache
# lvcreate −L 10G −n lv1 vg /dev/slow_devs

1. Create a 2-way RAID1 cache data LV
# lvcreate −−type raid1 −m 1 −L 1G -n cache1 vg \
        /dev/fast1 /dev/fast2

2. Create a 2-way RAID1 cache metadata LV
```

```
# lvcreate −−type raid1 −m 1 −L 8M -n cache1meta vg \
        /dev/fast1 /dev/fast2
```

3. Create a cache pool LV combining cache data LV and cache metadata LV

```
# lvconvert −−type cache−pool −−poolmetadata vg/cache1meta vg/cache1
```

4. Create a cached LV by combining the cache pool LV and origin LV

```
# lvconvert −−type cache −−cachepool vg/cache1 vg/lv1
```

Cache mode

The default cache mode is "writethrough". Writethrough ensures that any data written will be stored both in the cache pool LV and on the origin LV. The loss of a device associated with the cache pool LV 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 pool back to the origin LV. This mode will increase performance, but the loss of a device associated with the cache pool LV can result in lost data.

The cache mode can be specified with the --cachemode option when a cache pool LV is created.

lvm.conf(5) **cache_pool_cachemode**
defines the default cache mode.

0. Create an origin LV we wish to cache (yours may already exist)

```
# lvcreate −L 10G −n lv1 vg /dev/slow
```

1. Create a cache data LV

```
# lvcreate −L 1G −n cache1 vg /dev/fast
```

2. Create a cache metadata LV

```
# lvcreate −L 8M −n cache1meta vg /dev/fast
```

3. Create a cache pool LV specifying cache mode "writethrough"

```
# lvconvert −−type cache−pool −−poolmetadata vg/cache1meta \
        −−cachemode writethrough vg/cache1
```

4. Create a cache LV by combining the cache pool LV and origin LV

```
# lvconvert −−type cache −−cachepool vg/cache1 vg/lv1
```

Cache policy & policy settings

The cache subsystem has an additional per-LV parameter, namely the cache policy to use, and possibly the tunable parameters of the said cache policy. In the current implementation, two policies are available, "mq" which is the default policy and "cleaner" which is used to force the cache to write back (flush) all cached writes to the origin LV. Moreover, the "mq" policy has a number of tunable parameters: the defaults are chosen to be suitable for the vast majority of systems. However, under special circumstances, changing the tunable settings of the cache policy can improve performance.

On an existing cache LV, the policy can be set (to "mq") and the cache settings can be changed using commands like these:

Example

```
# lvchange −−cachepolicy mq vg/lv1
# lvchange −−cachesettings ’migration_threshold=2048 random_threshold=4’ \
    vg/lv1
```

Both commands can be combined, setting both cache policy and its settings together. Moreover, when creating a cache LV for the first time (using lvcreate), the −−cachepolicy and −−cachesettings parameters can be used as well. The current policy and the policy settings can be listed using the lvs command, using ’cache_policy’ and ’cache_settings’ fields:

```
# lvs -o +cache_policy,cache_settings
```

Spare metadata LV

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

Automatic pool metadata LV

A cache data LV can be converted to cache pool LV without specifying a cache pool metadata LV. LVM will automatically create a metadata LV from the same VG.

```
lvcreate -n CacheDataLV -L CacheSize VG
lvconvert --type cache−pool VG/CacheDataLV
```

Create a new cache LV without an existing origin LV

A cache LV can be created using an existing cache pool without an existing origin LV. A new origin LV is created and linked to the cache pool in a single step.

```
lvcreate −−type cache −L LargeSize −n CacheLV
         −−cachepool VG/CachePoolLV VG SlowPVs
```

Single step cache pool LV creation

A cache pool LV can be created with a single lvcreate command, rather than using lvconvert on existing LVs. This one command creates a cache data LV, a cache metadata LV, and combines the two into a cache pool LV.

```
lvcreate −−type cache−pool −L CacheSize −n CachePoolLV VG FastPVs
```

Convert existing LVs to cache types

When an existing origin LV is converted to a cache LV, the specified cache pool may be a normal LV, rather than a cache pool LV. In this case, lvm will first convert the normal LV to a cache pool LV. A pool metadata LV may optionally be specified.

```
lvcreate -n OriginLV -L LargeSize VG
lvcreate -n CacheDataLV -L CacheSize VG
lvconvert --type cache --cachepool VG/CataDataLV VG/OriginLV
```

This is equivalent to:

```
lvcreate -n OriginLV -L LargeSize VG
lvcreate -n CacheDataLV -L CacheSize VG
lvconvert --type cache-pool VG/CacheDataLV
lvconvert --type cache --cachepool VG/CachePoolLV VG/OriginLV
```

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)