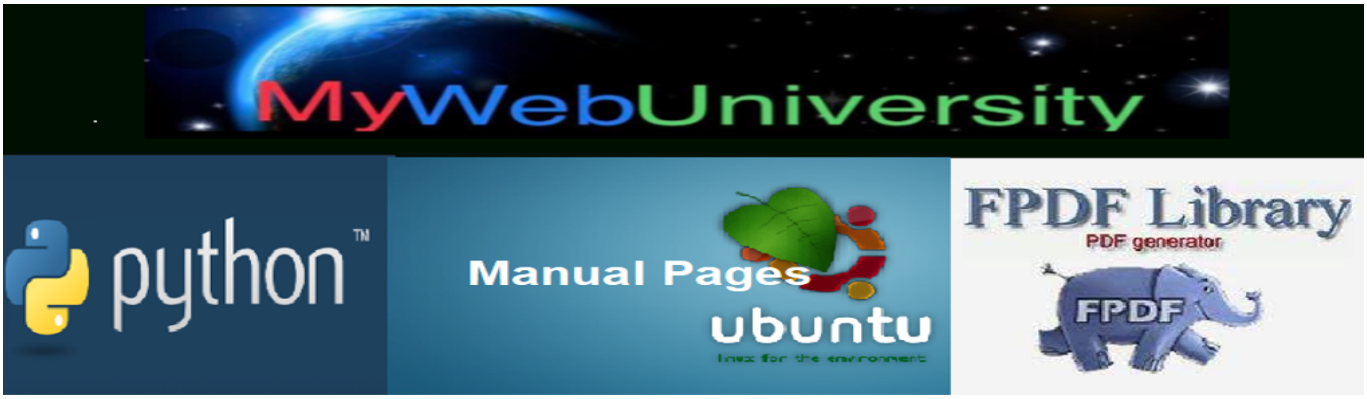

Rocky Enterprise Linux 9.2 Manual Pages on command 'lvmraid.7'

$ man lvmraid.7

LVMRAID(7) LVMRAID(7)

NAME

lvmraid ? LVM RAID

DESCRIPTION

lvm(8) RAID is a way to create a Logical Volume (LV) that uses multiple physical devices to improve performance or tolerate device failures. In LVM, the physical devices are Physical Volumes (PVs) in a single Volume Group (VG).

How LV data blocks are placed onto PVs is determined by the RAID level. RAID levels are commonly referred to as 'raid' followed by a number, e.g. raid1, raid5 or raid6. Select? ing a RAID level involves making tradeoffs among: physical device requirements, fault tol? erance, and performance. A description of the RAID levels can be found at www.snia.org/sites/default/files/SNIA_DDF_Technical_Position_v2.0.pdf

LVM RAID uses both Device Mapper (DM) and Multiple Device (MD) drivers from the Linux ker? nel. DM is used to create and manage visible LVM devices, and MD is used to place data on physical devices.

LVM creates hidden LVs (dm devices) layered between the visible LV and physical devices. LVs in the middle layers are called sub LVs. For LVM raid, a sub LV pair to store data and metadata (raid superblock and write intent bitmap) is created per raid image/leg (see lvs command examples below).

Create a RAID LV

To create a RAID LV, use lvcreate and specify an LV type. The LV type corresponds to a RAID level. The basic RAID levels that can be used are: raid0, raid1, raid4, raid5, raid6, raid10.

lvcreate --type RaidLevel [OPTIONS] --name Name --size Size VG [PVs]

To display the LV type of an existing LV, run:

lvs -o name,segtype LV

(The LV type is also referred to as "segment type" or "segtype".)

LVs can be created with the following types:

raid0

Also called striping, raid0 spreads LV data across multiple devices in units of stripe size. This is used to increase performance. LV data will be lost if any of the devices fail.

lvcreate --type raid0 [--stripes Number --stripesize Size] VG [PVs]

--stripes specifies the number of devices to spread the LV across.

--stripesize specifies the size of each stripe in kilobytes. This is the amount of data that is written to one device before moving to the next.

PVs specifies the devices to use. If not specified, lvm will choose Number devices, one for each stripe based on the number of PVs available or supplied.

raid1

Also called mirroring, raid1 uses multiple devices to duplicate LV data. The LV data re? mains available if all but one of the devices fail. The minimum number of devices (i.e. sub LV pairs) required is 2.

lvcreate --type raid1 [--mirrors Number] VG [PVs]

--mirrors specifies the number of mirror images in addition to the original LV image, e.g. --mirrors 1 means there are two images of the data, the original and one mirror im? age.

PVs specifies the devices to use. If not specified, lvm will choose Number devices, one for each image.

raid4

raid4 is a form of striping that uses an extra, first device dedicated to storing parity blocks. The LV data remains available if one device fails. The parity is used to recal? culate data that is lost from a single device. The minimum number of devices required is 3.

lvcreate --type raid4 [--stripes Number --stripesize Size] VG [PVs]

--stripes specifies the number of devices to use for LV data. This does not include the extra device lvm adds for storing parity blocks. A raid4 LV with Number stripes

requires Number+1 devices. Number must be 2 or more.

--stripesize specifies the size of each stripe in kilobytes. This is the amount of data that is written to one device before moving to the next.

PVs specifies the devices to use. If not specified, lvm will choose Number+1 separate de? vices.

raid4 is called non-rotating parity because the parity blocks are always stored on the same device.

raid5

raid5 is a form of striping that uses an extra device for storing parity blocks. LV data and parity blocks are stored on each device, typically in a rotating pattern for perfor? mance reasons. The LV data remains available if one device fails. The parity is used to recalculate data that is lost from a single device. The minimum number of devices re? quired is 3 (unless converting from 2 legged raid1 to reshape to more stripes; see reshap? ing).

lvcreate --type raid5 [--stripes Number --stripesize Size] VG [PVs]

--stripes specifies the number of devices to use for LV data. This does not include the extra device lvm adds for storing parity blocks. A raid5 LV with Number stripes requires Number+1 devices. Number must be 2 or more.

--stripesize specifies the size of each stripe in kilobytes. This is the amount of data that is written to one device before moving to the next.

PVs specifies the devices to use. If not specified, lvm will choose Number+1 separate de? vices.

raid5 is called rotating parity because the parity blocks are placed on different devices in a round-robin sequence. There are variations of raid5 with different algorithms for placing the parity blocks. The default variant is raid5_ls (raid5 left symmetric, which is a rotating parity 0 with data restart.) See RAID5 variants below.

raid6

raid6 is a form of striping like raid5, but uses two extra devices for parity blocks. LV data and parity blocks are stored on each device, typically in a rotating pattern for per? fomramce reasons. The LV data remains available if up to two devices fail. The parity is used to recalculate data that is lost from one or two devices. The minimum number of de? vices required is 5.

lvcreate --type raid6 [--stripes Number --stripesize Size] VG [PVs]

--stripes specifies the number of devices to use for LV data. This does not include the extra two devices lvm adds for storing parity blocks. A raid6 LV with Number stripes requires Number+2 devices. Number must be 3 or more.

--stripesize specifies the size of each stripe in kilobytes. This is the amount of data that is written to one device before moving to the next.

PVs specifies the devices to use. If not specified, lvm will choose Number+2 separate de? vices.

Like raid5, there are variations of raid6 with different algorithms for placing the parity blocks. The default variant is raid6_zr (raid6 zero restart, aka left symmetric, which is a rotating parity 0 with data restart.) See RAID6 variants below.

raid10

raid10 is a combination of raid1 and raid0, striping data across mirrored devices. LV data remains available if one or more devices remains in each mirror set. The minimum number of devices required is 4.

lvcreate --type raid10

[--mirrors NumberMirrors]

[--stripes NumberStripes --stripesize Size]

VG [PVs]

--mirrors specifies the number of mirror images within each stripe. e.g. --mirrors 1 means there are two images of the data, the original and one mirror image.

--stripes specifies the total number of devices to use in all raid1 images (not the number of raid1 devices to spread the LV across, even though that is the effective re? sult). The number of devices in each raid1 mirror will be NumberStripes/(Number? Mirrors+1), e.g. mirrors 1 and stripes 4 will stripe data across two raid1 mirrors, where each mirror is devices.

--stripesize specifies the size of each stripe in kilobytes. This is the amount of data that is written to one device before moving to the next.

PVs specifies the devices to use. If not specified, lvm will choose the necessary de? vices. Devices are used to create mirrors in the order listed, e.g. for mirrors 1, stripes 2, listing PV1 PV2 PV3 PV4 results in mirrors PV1/PV2 and PV3/PV4.

RAID10 is not mirroring on top of stripes, which would be RAID01, which is less tolerant of device failures.

Synchronization

Synchronization is the process that makes all the devices in a RAID LV consistent with each other.

In a RAID1 LV, all mirror images should have the same data. When a new mirror image is added, or a mirror image is missing data, then images need to be synchronized. Data blocks are copied from an existing image to a new or outdated image to make them match.

In a RAID 4/5/6 LV, parity blocks and data blocks should match based on the parity calculation. When the devices in a RAID LV change, the data and parity blocks can become inconsistent and need to be synchronized. Correct blocks are read, parity is calculated, and recalculated blocks are written.

The RAID implementation keeps track of which parts of a RAID LV are synchronized. When a RAID LV is first created and activated the first synchronization is called initialization.

A pointer stored in the raid metadata keeps track of the initialization process thus allowing it to be restarted after a deactivation of the RaidLV or a crash. Any writes to the RaidLV dirties the respective region of the write intent bitmap which allow for fast recovery of the regions after a crash. Without this, the entire LV would need to be synchronized every time it was activated.

Automatic synchronization happens when a RAID LV is activated, but it is usually partial because the bitmaps reduce the areas that are checked. A full sync becomes necessary when devices in the RAID LV are replaced.

The synchronization status of a RAID LV is reported by the following command, where "Cpy%Sync" = "100%" means sync is complete:

lvs -a -o name,sync_percent

Scrubbing

Scrubbing is a full scan of the RAID LV requested by a user. Scrubbing can find problems that are missed by partial synchronization.

Scrubbing assumes that RAID metadata and bitmaps may be inaccurate, so it verifies all RAID metadata, LV data, and parity blocks. Scrubbing can find inconsistencies caused by hardware errors or degradation. These kinds of problems may be undetected by automatic synchronization which excludes areas outside of the RAID write-intent bitmap.

The command to scrub a RAID LV can operate in two different modes:

lvchange --syncaction check|repair LV

check Check mode is read-only and only detects inconsistent areas in the RAID LV, it does not correct them.

repair Repair mode checks and writes corrected blocks to synchronize any inconsistent ar?
eas.

Scrubbing can consume a lot of bandwidth and slow down application I/O on the RAID LV. To control the I/O rate used for scrubbing, use:

--maxrecoveryrate Size[k|UNIT]

Sets the maximum recovery rate for a RAID LV. Size is specified as an amount per second for each device in the array. If no suffix is given, then KiB/sec/device is used. Setting the recovery rate to 0 means it will be unbounded.

--minrecoveryrate Size[k|UNIT]

Sets the minimum recovery rate for a RAID LV. Size is specified as an amount per second for each device in the array. If no suffix is given, then KiB/sec/device is used. Setting the recovery rate to 0 means it will be unbounded.

To display the current scrubbing in progress on an LV, including the syncaction mode and percent complete, run:

lvs -a -o name,raid_sync_action,sync_percent

After scrubbing is complete, to display the number of inconsistent blocks found, run:

lvs -o name,raid_mismatch_count

Also, if mismatches were found, the lvs attr field will display the letter "m" (mismatch) in the 9th position, e.g.

lvs -o name,vgname,segtype,attr vg/lv

```
  LV VG   Type  Attr
  lv vg   raid1 Rwi-a-r-m-
```

Scrubbing Limitations

The check mode can only report the number of inconsistent blocks, it cannot report which blocks are inconsistent. This makes it impossible to know which device has errors, or if the errors affect file system data, metadata or nothing at all.

The repair mode can make the RAID LV data consistent, but it does not know which data is correct. The result may be consistent but incorrect data. When two different blocks of data must be made consistent, it chooses the block from the device that would be used dur? ing RAID intialization. However, if the PV holding corrupt data is known, lvchange --re? build can be used in place of scrubbing to reconstruct the data on the bad device.

Future developments might include:

Allowing a user to choose the correct version of data during repair.

Using a majority of devices to determine the correct version of data to use in a 3-way RAID1 or RAID6 LV.

Using a checksumming device to pin-point when and where an error occurs, allowing it to be rewritten.

SubLVs

An LV is often a combination of other hidden LVs called SubLVs. The SubLVs either use physical devices, or are built from other SubLVs themselves. SubLVs hold LV data blocks, RAID parity blocks, and RAID metadata. SubLVs are generally hidden, so the lvs -a option is required to display them:

lvs -a -o name,segtype,devices

SubLV names begin with the visible LV name, and have an automatic suffix indicating its role:

? SubLVs holding LV data or parity blocks have the suffix _rimage_#. These SubLVs are sometimes referred to as DataLVs.

? SubLVs holding RAID metadata have the suffix _rmeta_#. RAID metadata includes su? perblock information, RAID type, bitmap, and device health information. These SubLVs are sometimes referred to as MetaLVs.

SubLVs are an internal implementation detail of LVM. The way they are used, constructed and named may change.

The following examples show the SubLV arrangement for each of the basic RAID LV types, us? ing the fewest number of devices allowed for each.

Examples

raid0

Each rimage SubLV holds a portion of LV data. No parity is used. No RAID metadata is used.

```
# lvcreate --type raid0 --stripes 2 --name lvr0 ...
# lvs -a -o name,segtype,devices
  lvr0            raid0  lvr0_rimage_0(0),lvr0_rimage_1(0)
  [lvr0_rimage_0] linear /dev/sda(...)
  [lvr0_rimage_1] linear /dev/sdb(...)
```

raid1

Each rimage SubLV holds a complete copy of LV data. No parity is used. Each rmeta SubLV holds RAID metadata.

```
# lvcreate --type raid1 --mirrors 1 --name lvr1 ...
# lvs -a -o name,segtype,devices
  lvr1           raid1  lvr1_rimage_0(0),lvr1_rimage_1(0)
  [lvr1_rimage_0] linear /dev/sda(...)
  [lvr1_rimage_1] linear /dev/sdb(...)
  [lvr1_rmeta_0]  linear /dev/sda(...)
  [lvr1_rmeta_1]  linear /dev/sdb(...)
```

raid4

At least three rimage SubLVs each hold a portion of LV data and one rimage SubLV holds parity. Each rmeta SubLV holds RAID metadata.

```
# lvcreate --type raid4 --stripes 2 --name lvr4 ...
# lvs -a -o name,segtype,devices
  lvr4           raid4  lvr4_rimage_0(0),\
                        lvr4_rimage_1(0),\
                        lvr4_rimage_2(0)
  [lvr4_rimage_0] linear /dev/sda(...)
  [lvr4_rimage_1] linear /dev/sdb(...)
  [lvr4_rimage_2] linear /dev/sdc(...)
  [lvr4_rmeta_0]  linear /dev/sda(...)
  [lvr4_rmeta_1]  linear /dev/sdb(...)
  [lvr4_rmeta_2]  linear /dev/sdc(...)
```

raid5

At least three rimage SubLVs each typcially hold a portion of LV data and parity (see sec? tion on raid5) Each rmeta SubLV holds RAID metadata.

```
# lvcreate --type raid5 --stripes 2 --name lvr5 ...
# lvs -a -o name,segtype,devices
  lvr5           raid5  lvr5_rimage_0(0),\
                        lvr5_rimage_1(0),\
                        lvr5_rimage_2(0)
  [lvr5_rimage_0] linear /dev/sda(...)
  [lvr5_rimage_1] linear /dev/sdb(...)
  [lvr5_rimage_2] linear /dev/sdc(...)
  [lvr5_rmeta_0]  linear /dev/sda(...)
```

[lvr5_rmeta_1] linear /dev/sdb(...)

[lvr5_rmeta_2] linear /dev/sdc(...)

raid6

At least five rimage SubLVs each typically hold a portion of LV data and parity. (see section on raid6) Each rmeta SubLV holds RAID metadata.

```
# lvcreate --type raid6 --stripes 3 --name lvr6
# lvs -a -o name,segtype,devices
  lvr6            raid6  lvr6_rimage_0(0),\
                         lvr6_rimage_1(0),\
                         lvr6_rimage_2(0),\
                         lvr6_rimage_3(0),\
                         lvr6_rimage_4(0),\
                         lvr6_rimage_5(0)
  [lvr6_rimage_0] linear /dev/sda(...)
  [lvr6_rimage_1] linear /dev/sdb(...)
  [lvr6_rimage_2] linear /dev/sdc(...)
  [lvr6_rimage_3] linear /dev/sdd(...)
  [lvr6_rimage_4] linear /dev/sde(...)
  [lvr6_rimage_5] linear /dev/sdf(...)
  [lvr6_rmeta_0]  linear /dev/sda(...)
  [lvr6_rmeta_1]  linear /dev/sdb(...)
  [lvr6_rmeta_2]  linear /dev/sdc(...)
  [lvr6_rmeta_3]  linear /dev/sdd(...)
  [lvr6_rmeta_4]  linear /dev/sde(...)
  [lvr6_rmeta_5]  linear /dev/sdf(...)
```

raid10

At least four rimage SubLVs each hold a portion of LV data. No parity is used. Each rmeta SubLV holds RAID metadata.

```
# lvcreate --type raid10 --stripes 2 --mirrors 1 --name lvr10
# lvs -a -o name,segtype,devices
  lvr10           raid10 lvr10_rimage_0(0),\
                         lvr10_rimage_1(0),\
                         lvr10_rimage_2(0),\
```

lvr10_rimage_3(0)

[lvr10_rimage_0] linear /dev/sda(...)

[lvr10_rimage_1] linear /dev/sdb(...)

[lvr10_rimage_2] linear /dev/sdc(...)

[lvr10_rimage_3] linear /dev/sdd(...)

[lvr10_rmeta_0] linear /dev/sda(...)

[lvr10_rmeta_1] linear /dev/sdb(...)

[lvr10_rmeta_2] linear /dev/sdc(...)

[lvr10_rmeta_3] linear /dev/sdd(...)

Device Failure

Physical devices in a RAID LV can fail or be lost for multiple reasons. A device could be disconnected, permanently failed, or temporarily disconnected. The purpose of RAID LVs (levels 1 and higher) is to continue operating in a degraded mode, without losing LV data, even after a device fails. The number of devices that can fail without the loss of LV data depends on the RAID level:

? RAID0 (striped) LVs cannot tolerate losing any devices. LV data will be lost if any devices fail.

? RAID1 LVs can tolerate losing all but one device without LV data loss.

? RAID4 and RAID5 LVs can tolerate losing one device without LV data loss.

? RAID6 LVs can tolerate losing two devices without LV data loss.

? RAID10 is variable, and depends on which devices are lost. It stripes across multiple mirror groups with raid1 layout thus it can tolerate losing all but one device in each of these groups without LV data loss.

If a RAID LV is missing devices, or has other device-related problems, lvs reports this in the health_status (and attr) fields:

lvs -o name,lv_health_status

partial

Devices are missing from the LV. This is also indicated by the letter "p" (partial) in the 9th position of the lvs attr field.

refresh needed

A device was temporarily missing but has returned. The LV needs to be refreshed to use the device again (which will usually require partial synchronization). This is also indi? cated by the letter "r" (refresh needed) in the 9th position of the lvs attr field. See

Refreshing an LV. This could also indicate a problem with the device, in which case it should be be replaced, see Replacing Devices.

mismatches exist

See Scrubbing.

Most commands will also print a warning if a device is missing, e.g.

WARNING: Device for PV uItL3Z-wBME-DQy0-... not found or rejected ...

This warning will go away if the device returns or is removed from the VG (see vgreduce --removemissing).

Activating an LV with missing devices

A RAID LV that is missing devices may be activated or not, depending on the "activation mode" used in lvchange:

lvchange -ay --activationmode complete|degraded|partial LV

complete

The LV is only activated if all devices are present.

degraded

The LV is activated with missing devices if the RAID level can tolerate the number of missing devices without LV data loss.

partial

The LV is always activated, even if portions of the LV data are missing because of the missing device(s). This should only be used to perform extreme recovery or repair opera? tions.

lvm.conf(5) activation/activation_mode

controls the activation mode when not specified by the command.

The default value is printed by:

lvmconfig --type default activation/activation_mode

Replacing Devices

Devices in a RAID LV can be replaced by other devices in the VG. When replacing devices that are no longer visible on the system, use lvconvert --repair. When replacing devices that are still visible, use lvconvert --replace. The repair command will attempt to re? store the same number of data LVs that were previously in the LV. The replace option can be repeated to replace multiple PVs. Replacement devices can be optionally listed with either option.

lvconvert --repair LV [NewPVs]

lvconvert --replace OldPV LV [NewPV]

lvconvert --replace OldPV1 --replace OldPV2 LV [NewPVs]

New devices require synchronization with existing devices, see Synchronization.

Refreshing an LV

Refreshing a RAID LV clears any transient device failures (device was temporarily discon?
nected) and returns the LV to its fully redundant mode. Restoring a device will usually
require at least partial synchronization (see Synchronization). Failure to clear a tran?
sient failure results in the RAID LV operating in degraded mode until it is reactivated.

Use the lvchange command to refresh an LV:

lvchange --refresh LV

```
# lvs -o name,vgname,segtype,attr,size vg
  LV VG   Type  Attr       LSize
  lv vg   raid1 Rwi-a-r-r- 100.00g
# lvchange --refresh vg/lv
# lvs -o name,vgname,segtype,attr,size vg
  LV VG   Type  Attr       LSize
  lv vg   raid1 Rwi-a-r--- 100.00g
```

Automatic repair

If a device in a RAID LV fails, device-mapper in the kernel notifies the dmeventd(8) moni?
toring process (see Monitoring). dmeventd can be configured to automatically respond us?
ing:

lvm.conf(5) activation/raid_fault_policy

Possible settings are:

warn

A warning is added to the system log indicating that a device has failed in the RAID LV.
It is left to the user to repair the LV, e.g. replace failed devices.

allocate

dmeventd automatically attempts to repair the LV using spare devices in the VG. Note that
even a transient failure is treated as a permanent failure under this setting. A new de?
vice is allocated and full synchronization is started.

The specific command run by dmeventd to warn or repair is:

lvconvert --repair --use-policies LV

Corrupted Data

Data on a device can be corrupted due to hardware errors without the device ever being disconnected or there being any fault in the software. This should be rare, and can be detected (see Scrubbing).

Rebuild specific PVs

If specific PVs in a RAID LV are known to have corrupt data, the data on those PVs can be reconstructed with:

lvchange --rebuild PV LV

The rebuild option can be repeated with different PVs to replace the data on multiple PVs.

Monitoring

When a RAID LV is activated the dmeventd(8) process is started to monitor the health of the LV. Various events detected in the kernel can cause a notification to be sent from device-mapper to the monitoring process, including device failures and synchronization completion (e.g. for initialization or scrubbing).

The LVM configuration file contains options that affect how the monitoring process will respond to failure events (e.g. raid_fault_policy). It is possible to turn on and off monitoring with lvchange, but it is not recommended to turn this off unless you have a thorough knowledge of the consequences.

Configuration Options

There are a number of options in the LVM configuration file that affect the behavior of RAID LVs. The tunable options are listed below. A detailed description of each can be found in the LVM configuration file itself.

mirror_segtype_default

raid10_segtype_default

raid_region_size

raid_fault_policy

activation_mode

Data Integrity

The device mapper integrity target can be used in combination with RAID levels 1,4,5,6,10 to detect and correct data corruption in RAID images. A dm-integrity layer is placed above each RAID image, and an extra sub LV is created to hold integrity metadata (data check? sums) for each RAID image. When data is read from an image, integrity checksums are used to detect corruption. If detected, dm-raid reads the data from another (good) image to re? turn to the caller. dm-raid will also automatically write the good data back to the image

with bad data to correct the corruption.

When creating a RAID LV with integrity, or adding integrity, space is required for integrity metadata. Every 500MB of LV data requires an additional 4MB to be allocated for integrity metadata, for each RAID image.

Create a RAID LV with integrity:

lvcreate --type raidN --raidintegrity y

Add integrity to an existing RAID LV:

lvconvert --raidintegrity y LV

Remove integrity from a RAID LV:

lvconvert --raidintegrity n LV

Integrity options

--raidintegritymode journal|bitmap

Use a journal (default) or bitmap for keeping integrity checksums consistent in case of a crash. The bitmap areas are recalculated after a crash, so corruption in those areas would not be detected. A journal does not have this problem. The journal mode doubles writes to storage, but can improve performance for scattered writes packed into a single journal write. bitmap mode can in theory achieve full write throughput of the device, but would not benefit from the potential scattered write optimization.

--raidintegrityblocksize 512|1024|2048|4096

The block size to use for dm-integrity on raid images. The integrity block size should usually match the device logical block size, or the file system sector/block sizes. It may be less than the file system sector/block size, but not less than the device logical block size. Possible values: 512, 1024, 2048, 4096.

Integrity initialization

When integrity is added to an LV, the kernel needs to initialize the integrity metadata (checksums) for all blocks in the LV. The data corruption checking performed by dm-integrity will only operate on areas of the LV that are already initialized. The progress of integrity initialization is reported by the "syncpercent" LV reporting field (and under the Cpy%Sync lvs column.)

Integrity limitations

To work around some limitations, it is possible to remove integrity from the LV, make the change, then add integrity again. (Integrity metadata would need to initialized when added again.)

LVM must be able to allocate the integrity metadata sub LV on a single PV that is already in use by the associated RAID image. This can potentially cause a problem during lvextend if the original PV holding the image and integrity metadata is full. To work around this limitation, remove integrity, extend the LV, and add integrity again.

Additional RAID images can be added to raid1 LVs, but not to other raid levels.

A raid1 LV with integrity cannot be converted to linear (remove integrity to do this.)

RAID LVs with integrity cannot yet be used as sub LVs with other LV types.

The following are not yet permitted on RAID LVs with integrity: lvreduce, pvmove, snap? shots, splitmirror, raid syncaction commands, raid rebuild.

RAID1 Tuning

A RAID1 LV can be tuned so that certain devices are avoided for reading while all devices are still written to.

lvchange --[raid]writemostly PV[:y|n|t] LV

The specified device will be marked as "write mostly", which means that reading from this device will be avoided, and other devices will be preferred for reading (unless no other devices are available.) This minimizes the I/O to the specified device.

If the PV name has no suffix, the write mostly attribute is set. If the PV name has the suffix :n, the write mostly attribute is cleared, and the suffix :t toggles the current setting.

The write mostly option can be repeated on the command line to change multiple devices at once.

To report the current write mostly setting, the lvs attr field will show the letter "w" in the 9th position when write mostly is set:

lvs -a -o name,attr

When a device is marked write mostly, the maximum number of outstanding writes to that de? vice can be configured. Once the maximum is reached, further writes become synchronous. When synchronous, a write to the LV will not complete until writes to all the mirror im? ages are complete.

lvchange --[raid]writebehind Number LV

To report the current write behind setting, run:

lvs -o name,raid_write_behind

When write behind is not configured, or set to 0, all LV writes are synchronous.

RAID Takeover

RAID takeover is converting a RAID LV from one RAID level to another, e.g. raid5 to raid6. Changing the RAID level is usually done to increase or decrease resilience to de? vice failures or to restripe LVs. This is done using lvconvert and specifying the new RAID level as the LV type:

lvconvert --type RaidLevel LV [PVs]

The most common and recommended RAID takeover conversions are:

linear to raid1

Linear is a single image of LV data, and converting it to raid1 adds a mirror image which is a direct copy of the original linear image.

striped/raid0 to raid4/5/6

Adding parity devices to a striped volume results in raid4/5/6.

Unnatural conversions that are not recommended include converting between striped and non-striped types. This is because file systems often optimize I/O patterns based on device striping values. If those values change, it can decrease performance.

Converting to a higher RAID level requires allocating new SubLVs to hold RAID metadata, and new SubLVs to hold parity blocks for LV data. Converting to a lower RAID level re? moves the SubLVs that are no longer needed.

Conversion often requires full synchronization of the RAID LV (see Synchronization). Con? verting to RAID1 requires copying all LV data blocks to N new images on new devices. Con? verting to a parity RAID level requires reading all LV data blocks, calculating parity, and writing the new parity blocks. Synchronization can take a long time depending on the throughpout of the devices used and the size of the RaidLV. It can degrade performance (rate controls also apply to conversion; see --minrecoveryrate and --maxrecoveryrate.)

Warning: though it is possible to create striped LVs with up to 128 stripes, a maximum of 64 stripes can be converted to raid0, 63 to raid4/5 and 62 to raid6 because of the added parity SubLVs. A striped LV with a maximum of 32 stripes can be converted to raid10.

The following takeover conversions are currently possible:

? between striped and raid0.

? between linear and raid1.

? between mirror and raid1.

? between raid1 with two images and raid4/5.

? between striped/raid0 and raid4.

? between striped/raid0 and raid5.

? between striped/raid0 and raid6.

? between raid4 and raid5.

? between raid4/raid5 and raid6.

? between striped/raid0 and raid10.

? between striped and raid4.

Indirect conversions

Converting from one raid level to another may require multiple steps, converting first to intermediate raid levels.

linear to raid6

To convert an LV from linear to raid6:

1. convert to raid1 with two images
2. convert to raid5 (internally raid5_ls) with two images
3. convert to raid5 with three or more stripes (reshape)
4. convert to raid6 (internally raid6_ls_6)
5. convert to raid6 (internally raid6_zr, reshape)

The commands to perform the steps above are:

1. lvconvert --type raid1 --mirrors 1 LV
2. lvconvert --type raid5 LV
3. lvconvert --stripes 3 LV
4. lvconvert --type raid6 LV
5. lvconvert --type raid6 LV

The final conversion from raid6_ls_6 to raid6_zr is done to avoid the potential write/re? covery performance reduction in raid6_ls_6 because of the dedicated parity device. raid6_zr rotates data and parity blocks to avoid this.

linear to striped

To convert an LV from linear to striped:

1. convert to raid1 with two images
2. convert to raid5_n
3. convert to raid5_n with five 128k stripes (reshape)
4. convert raid5_n to striped

The commands to perform the steps above are:

1. lvconvert --type raid1 --mirrors 1 LV
2. lvconvert --type raid5_n LV

3. lvconvert --stripes 5 --stripesize 128k LV

4. lvconvert --type striped LV

The raid5_n type in step 2 is used because it has dedicated parity SubLVs at the end, and can be converted to striped directly. The stripe size is increased in step 3 to add extra space for the conversion process. This step grows the LV size by a factor of five. After conversion, this extra space can be reduced (or used to grow the file system using the LV).

Reversing these steps will convert a striped LV to linear.

raid6 to striped

To convert an LV from raid6_nr to striped:

1. convert to raid6_n_6

2. convert to striped

The commands to perform the steps above are:

1. lvconvert --type raid6_n_6 LV

2. lvconvert --type striped LV

Examples

Converting an LV from linear to raid1.

```
# lvs -a -o name,segtype,size vg
  LV   Type   LSize
  lv   linear 300.00g
# lvconvert --type raid1 --mirrors 1 vg/lv
# lvs -a -o name,segtype,size vg
  LV           Type   LSize
  lv           raid1  300.00g
  [lv_rimage_0] linear 300.00g
  [lv_rimage_1] linear 300.00g
  [lv_rmeta_0]  linear   3.00m
  [lv_rmeta_1]  linear   3.00m
```

Converting an LV from mirror to raid1.

```
# lvs -a -o name,segtype,size vg
  LV           Type   LSize
  lv           mirror 100.00g
  [lv_mimage_0] linear 100.00g
```

[lv_mimage_1] linear 100.00g

[lv_mlog] linear 3.00m

lvconvert --type raid1 vg/lv

lvs -a -o name,segtype,size vg

LV Type LSize

lv raid1 100.00g

[lv_rimage_0] linear 100.00g

[lv_rimage_1] linear 100.00g

[lv_rmeta_0] linear 3.00m

[lv_rmeta_1] linear 3.00m

Converting an LV from linear to raid1 (with 3 images).

lvconvert --type raid1 --mirrors 2 vg/lv

Converting an LV from striped (with 4 stripes) to raid6_n_6.

lvcreate --stripes 4 -L64M -n lv vg

lvconvert --type raid6 vg/lv

lvs -a -o lv_name,segtype,sync_percent,data_copies

LV Type Cpy%Sync #Cpy

lv raid6_n_6 100.00 3

[lv_rimage_0] linear

[lv_rimage_1] linear

[lv_rimage_2] linear

[lv_rimage_3] linear

[lv_rimage_4] linear

[lv_rimage_5] linear

[lv_rmeta_0] linear

[lv_rmeta_1] linear

[lv_rmeta_2] linear

[lv_rmeta_3] linear

[lv_rmeta_4] linear

[lv_rmeta_5] linear

This convert begins by allocating MetaLVs (rmeta_#) for each of the existing stripe de?

vices. It then creates 2 additional MetaLV/DataLV pairs (rmeta_#/rimage_#) for dedicated

raid6 parity.

If rotating data/parity is required, such as with raid6_nr, it must be done by reshaping (see below).

RAID Reshaping

RAID reshaping is changing attributes of a RAID LV while keeping the same RAID level. This includes changing RAID layout, stripe size, or number of stripes.

When changing the RAID layout or stripe size, no new SubLVs (MetaLVs or DataLVs) need to be allocated, but DataLVs are extended by a small amount (typically 1 extent). The extra space allows blocks in a stripe to be updated safely, and not be corrupted in case of a crash. If a crash occurs, reshaping can just be restarted.

(If blocks in a stripe were updated in place, a crash could leave them partially updated and corrupted. Instead, an existing stripe is quiesced, read, changed in layout, and the new stripe written to free space. Once that is done, the new stripe is unquiesced and used.)

Examples

(Command output shown in examples may change.)

Converting raid6_n_6 to raid6_nr with rotating data/parity.

This conversion naturally follows a previous conversion from striped/raid0 to raid6_n_6 (shown above). It completes the transition to a more traditional RAID6.

```
# lvs -o lv_name,segtype,sync_percent,data_copies
  LV              Type       Cpy%Sync #Cpy
  lv              raid6_n_6  100.00      3
  [lv_rimage_0] linear
  [lv_rimage_1] linear
  [lv_rimage_2] linear
  [lv_rimage_3] linear
  [lv_rimage_4] linear
  [lv_rimage_5] linear
  [lv_rmeta_0]  linear
  [lv_rmeta_1]  linear
  [lv_rmeta_2]  linear
  [lv_rmeta_3]  linear
  [lv_rmeta_4]  linear
  [lv_rmeta_5]  linear
```

```
# lvconvert --type raid6_nr vg/lv
# lvs -a -o lv_name,segtype,sync_percent,data_copies
  LV            Type     Cpy%Sync #Cpy
  lv            raid6_nr 100.00      3
  [lv_rimage_0] linear
  [lv_rimage_0] linear
  [lv_rimage_1] linear
  [lv_rimage_1] linear
  [lv_rimage_2] linear
  [lv_rimage_2] linear
  [lv_rimage_3] linear
  [lv_rimage_3] linear
  [lv_rimage_4] linear
  [lv_rimage_5] linear
  [lv_rmeta_0]  linear
  [lv_rmeta_1]  linear
  [lv_rmeta_2]  linear
  [lv_rmeta_3]  linear
  [lv_rmeta_4]  linear
  [lv_rmeta_5]  linear
```

The DataLVs are larger (additional segment in each) which provides space for out-of-place reshaping. The result is:

```
# lvs -a -o lv_name,segtype,seg_pe_ranges,dataoffset
  LV            Type     PE Ranges          DOff
  lv            raid6_nr lv_rimage_0:0-32 \
                         lv_rimage_1:0-32 \
                         lv_rimage_2:0-32 \
                         lv_rimage_3:0-32
  [lv_rimage_0] linear   /dev/sda:0-31      2048
  [lv_rimage_0] linear   /dev/sda:33-33
  [lv_rimage_1] linear   /dev/sdaa:0-31     2048
  [lv_rimage_1] linear   /dev/sdaa:33-33
  [lv_rimage_2] linear   /dev/sdab:1-33     2048
```

```
  [lv_rimage_3] linear   /dev/sdac:1-33     2048
  [lv_rmeta_0]  linear   /dev/sda:32-32
  [lv_rmeta_1]  linear   /dev/sdaa:32-32
  [lv_rmeta_2]  linear   /dev/sdab:0-0
  [lv_rmeta_3]  linear   /dev/sdac:0-0
```

All segments with PE ranges '33-33' provide the out-of-place reshape space. The dataoff? set column shows that the data was moved from initial offset 0 to 2048 sectors on each component DataLV.

For performance reasons the raid6_nr RaidLV can be restriped. Convert it from 3-way striped to 5-way-striped.

```
# lvconvert --stripes 5 vg/lv
  Using default stripesize 64.00 KiB.
  WARNING: Adding stripes to active logical volume vg/lv will \
  grow it from 99 to 165 extents!
  Run "lvresize -l99 vg/lv" to shrink it or use the additional \
  capacity.
  Logical volume vg/lv successfully converted.
# lvs vg/lv
  LV   VG     Attr       LSize   Cpy%Sync
  lv   vg     rwi-a-r-s- 652.00m 52.94
# lvs -a -o lv_name,attr,segtype,seg_pe_ranges,dataoffset vg
  LV            Attr       Type     PE Ranges          DOff
  lv            rwi-a-r--- raid6_nr lv_rimage_0:0-33 \
                                    lv_rimage_1:0-33 \
                                    lv_rimage_2:0-33 ... \
                                    lv_rimage_5:0-33 \
                                    lv_rimage_6:0-33   0
  [lv_rimage_0] iwi-aor--- linear   /dev/sda:0-32      0
  [lv_rimage_0] iwi-aor--- linear   /dev/sda:34-34
  [lv_rimage_1] iwi-aor--- linear   /dev/sdaa:0-32     0
  [lv_rimage_1] iwi-aor--- linear   /dev/sdaa:34-34
  [lv_rimage_2] iwi-aor--- linear   /dev/sdab:0-32     0
  [lv_rimage_2] iwi-aor--- linear   /dev/sdab:34-34
```

```
  [lv_rimage_3] iwi-aor--- linear   /dev/sdac:1-34     0
  [lv_rimage_4] iwi-aor--- linear   /dev/sdad:1-34     0
  [lv_rimage_5] iwi-aor--- linear   /dev/sdae:1-34     0
  [lv_rimage_6] iwi-aor--- linear   /dev/sdaf:1-34     0
  [lv_rmeta_0]  ewi-aor--- linear   /dev/sda:33-33
  [lv_rmeta_1]  ewi-aor--- linear   /dev/sdaa:33-33
  [lv_rmeta_2]  ewi-aor--- linear   /dev/sdab:33-33
  [lv_rmeta_3]  ewi-aor--- linear   /dev/sdac:0-0
  [lv_rmeta_4]  ewi-aor--- linear   /dev/sdad:0-0
  [lv_rmeta_5]  ewi-aor--- linear   /dev/sdae:0-0
  [lv_rmeta_6]  ewi-aor--- linear   /dev/sdaf:0-0
```

Stripes also can be removed from raid5 and 6. Convert the 5-way striped raid6_nr LV to 4-way-striped. The force option needs to be used, because removing stripes (i.e. image SubLVs) from a RaidLV will shrink its size.

```
# lvconvert --stripes 4 vg/lv
  Using default stripesize 64.00 KiB.
  WARNING: Removing stripes from active logical volume vg/lv will \
  shrink it from 660.00 MiB to 528.00 MiB!
  THIS MAY DESTROY (PARTS OF) YOUR DATA!
  If that leaves the logical volume larger than 206 extents due \
  to stripe rounding,
  you may want to grow the content afterwards (filesystem etc.)
  WARNING: to remove freed stripes after the conversion has finished,\
  you have to run "lvconvert --stripes 4 vg/lv"
  Logical volume vg/lv successfully converted.
# lvs -a -o lv_name,attr,segtype,seg_pe_ranges,dataoffset vg
  LV           Attr       Type     PE Ranges          DOff
  lv           rwi-a-r-s- raid6_nr lv_rimage_0:0-33 \
                                   lv_rimage_1:0-33 \
                                   lv_rimage_2:0-33 ... \
                                   lv_rimage_5:0-33 \
                                   lv_rimage_6:0-33   0
  [lv_rimage_0] Iwi-aor--- linear   /dev/sda:0-32      0
```

```
  [lv_rimage_0] lwi-aor--- linear   /dev/sda:34-34
  [lv_rimage_1] lwi-aor--- linear   /dev/sdaa:0-32     0
  [lv_rimage_1] lwi-aor--- linear   /dev/sdaa:34-34
  [lv_rimage_2] lwi-aor--- linear   /dev/sdab:0-32     0
  [lv_rimage_2] lwi-aor--- linear   /dev/sdab:34-34
  [lv_rimage_3] lwi-aor--- linear   /dev/sdac:1-34     0
  [lv_rimage_4] lwi-aor--- linear   /dev/sdad:1-34     0
  [lv_rimage_5] lwi-aor--- linear   /dev/sdae:1-34     0
  [lv_rimage_6] lwi-aor-R- linear   /dev/sdaf:1-34     0
  [lv_rmeta_0]  ewi-aor--- linear   /dev/sda:33-33
  [lv_rmeta_1]  ewi-aor--- linear   /dev/sdaa:33-33
  [lv_rmeta_2]  ewi-aor--- linear   /dev/sdab:33-33
  [lv_rmeta_3]  ewi-aor--- linear   /dev/sdac:0-0
  [lv_rmeta_4]  ewi-aor--- linear   /dev/sdad:0-0
  [lv_rmeta_5]  ewi-aor--- linear   /dev/sdae:0-0
  [lv_rmeta_6]  ewi-aor-R- linear   /dev/sdaf:0-0
```

The 's' in column 9 of the attribute field shows the RaidLV is still reshaping. The 'R' in the same column of the attribute field shows the freed image Sub LVs which will need removing once the reshaping finished.

```
# lvs -o lv_name,attr,segtype,seg_pe_ranges,dataoffset vg
  LV   Attr       Type     PE Ranges          DOff
  lv   rwi-a-r-R- raid6_nr lv_rimage_0:0-33 \
                           lv_rimage_1:0-33 \
                           lv_rimage_2:0-33 ... \
                           lv_rimage_5:0-33 \
                           lv_rimage_6:0-33   8192
```

Now that the reshape is finished the 'R' atribute on the RaidLV shows images can be re? moved.

```
# lvs -o lv_name,attr,segtype,seg_pe_ranges,dataoffset vg
  LV   Attr       Type     PE Ranges          DOff
  lv   rwi-a-r-R- raid6_nr lv_rimage_0:0-33 \
                           lv_rimage_1:0-33 \
                           lv_rimage_2:0-33 ... \
```

```
                    lv_rimage_5:0-33 \
                    lv_rimage_6:0-33   8192
```

This is achieved by repeating the command ("lvconvert --stripes 4 vg/lv" would be suffi?

cient).

```
# lvconvert --stripes 4 vg/lv
  Using default stripesize 64.00 KiB.
  Logical volume vg/lv successfully converted.
# lvs -a -o lv_name,attr,segtype,seg_pe_ranges,dataoffset vg
  LV           Attr       Type     PE Ranges          DOff
  lv           rwi-a-r--- raid6_nr lv_rimage_0:0-33 \
                                   lv_rimage_1:0-33 \
                                   lv_rimage_2:0-33 ... \
                                   lv_rimage_5:0-33   8192
  [lv_rimage_0] iwi-aor--- linear   /dev/sda:0-32      8192
  [lv_rimage_0] iwi-aor--- linear   /dev/sda:34-34
  [lv_rimage_1] iwi-aor--- linear   /dev/sdaa:0-32     8192
  [lv_rimage_1] iwi-aor--- linear   /dev/sdaa:34-34
  [lv_rimage_2] iwi-aor--- linear   /dev/sdab:0-32     8192
  [lv_rimage_2] iwi-aor--- linear   /dev/sdab:34-34
  [lv_rimage_3] iwi-aor--- linear   /dev/sdac:1-34     8192
  [lv_rimage_4] iwi-aor--- linear   /dev/sdad:1-34     8192
  [lv_rimage_5] iwi-aor--- linear   /dev/sdae:1-34     8192
  [lv_rmeta_0]  ewi-aor--- linear   /dev/sda:33-33
  [lv_rmeta_1]  ewi-aor--- linear   /dev/sdaa:33-33
  [lv_rmeta_2]  ewi-aor--- linear   /dev/sdab:33-33
  [lv_rmeta_3]  ewi-aor--- linear   /dev/sdac:0-0
  [lv_rmeta_4]  ewi-aor--- linear   /dev/sdad:0-0
  [lv_rmeta_5]  ewi-aor--- linear   /dev/sdae:0-0
# lvs -a -o lv_name,attr,segtype,reshapelen vg
  LV           Attr       Type     RSize
  lv           rwi-a-r--- raid6_nr 24.00m
  [lv_rimage_0] iwi-aor--- linear    4.00m
  [lv_rimage_0] iwi-aor--- linear
```

[lv_rimage_1] iwi-aor--- linear 4.00m

[lv_rimage_1] iwi-aor--- linear

[lv_rimage_2] iwi-aor--- linear 4.00m

[lv_rimage_2] iwi-aor--- linear

[lv_rimage_3] iwi-aor--- linear 4.00m

[lv_rimage_4] iwi-aor--- linear 4.00m

[lv_rimage_5] iwi-aor--- linear 4.00m

[lv_rmeta_0] ewi-aor--- linear

[lv_rmeta_1] ewi-aor--- linear

[lv_rmeta_2] ewi-aor--- linear

[lv_rmeta_3] ewi-aor--- linear

[lv_rmeta_4] ewi-aor--- linear

[lv_rmeta_5] ewi-aor--- linear

Future developments might include automatic removal of the freed images.

If the reshape space shall be removed any lvconvert command not changing the layout can be used:

lvconvert --stripes 4 vg/lv

Using default stripesize 64.00 KiB.

No change in RAID LV vg/lv layout, freeing reshape space.

Logical volume vg/lv successfully converted.

lvs -a -o lv_name,attr,segtype,reshapelen vg

LV Attr Type RSize

lv rwi-a-r--- raid6_nr 0

[lv_rimage_0] iwi-aor--- linear 0

[lv_rimage_0] iwi-aor--- linear

[lv_rimage_1] iwi-aor--- linear 0

[lv_rimage_1] iwi-aor--- linear

[lv_rimage_2] iwi-aor--- linear 0

[lv_rimage_2] iwi-aor--- linear

[lv_rimage_3] iwi-aor--- linear 0

[lv_rimage_4] iwi-aor--- linear 0

[lv_rimage_5] iwi-aor--- linear 0

[lv_rmeta_0] ewi-aor--- linear

```
  [lv_rmeta_1]  ewi-aor--- linear
  [lv_rmeta_2]  ewi-aor--- linear
  [lv_rmeta_3]  ewi-aor--- linear
  [lv_rmeta_4]  ewi-aor--- linear
  [lv_rmeta_5]  ewi-aor--- linear
```

In case the RaidLV should be converted to striped:

```
# lvconvert --type striped vg/lv
  Unable to convert LV vg/lv from raid6_nr to striped.
  Converting vg/lv from raid6_nr is directly possible to the \
  following layouts:
    raid6_nc
    raid6_zr
    raid6_la_6
    raid6_ls_6
    raid6_ra_6
    raid6_rs_6
    raid6_n_6
```

A direct conversion isn't possible thus the command informed about the possible ones.

raid6_n_6 is suitable to convert to striped so convert to it first (this is a reshape changing the raid6 layout from raid6_nr to raid6_n_6).

```
# lvconvert --type raid6_n_6
  Using default stripesize 64.00 KiB.
  Converting raid6_nr LV vg/lv to raid6_n_6.
Are you sure you want to convert raid6_nr LV vg/lv? [y/n]: y
  Logical volume vg/lv successfully converted.
```

Wait for the reshape to finish.

```
# lvconvert --type striped vg/lv
  Logical volume vg/lv successfully converted.
# lvs -o lv_name,attr,segtype,seg_pe_ranges,dataoffset vg
  LV   Attr       Type    PE Ranges  DOff
  lv   -wi-a----- striped /dev/sda:2-32 \
                          /dev/sdaa:2-32 \
                          /dev/sdab:2-32 \
```

```
                    /dev/sdac:3-33
  lv   -wi-a----- striped /dev/sda:34-35 \
                    /dev/sdaa:34-35 \
                    /dev/sdab:34-35 \
                    /dev/sdac:34-35
```

From striped we can convert to raid10

```
# lvconvert --type raid10 vg/lv
  Using default stripesize 64.00 KiB.
  Logical volume vg/lv successfully converted.
# lvs -o lv_name,attr,segtype,seg_pe_ranges,dataoffset vg
  LV   Attr       Type   PE Ranges          DOff
  lv   rwi-a-r--- raid10 lv_rimage_0:0-32 \
                    lv_rimage_4:0-32 \
                    lv_rimage_1:0-32 ... \
                    lv_rimage_3:0-32 \
                    lv_rimage_7:0-32   0
# lvs -a -o lv_name,attr,segtype,seg_pe_ranges,dataoffset vg
  WARNING: Cannot find matching striped segment for vg/lv_rimage_3.
  LV            Attr       Type   PE Ranges          DOff
  lv            rwi-a-r--- raid10 lv_rimage_0:0-32 \
                                  lv_rimage_4:0-32 \
                                  lv_rimage_1:0-32 ... \
                                  lv_rimage_3:0-32 \
                                  lv_rimage_7:0-32   0
  [lv_rimage_0] iwi-aor--- linear /dev/sda:2-32      0
  [lv_rimage_0] iwi-aor--- linear /dev/sda:34-35
  [lv_rimage_1] iwi-aor--- linear /dev/sdaa:2-32     0
  [lv_rimage_1] iwi-aor--- linear /dev/sdaa:34-35
  [lv_rimage_2] iwi-aor--- linear /dev/sdab:2-32     0
  [lv_rimage_2] iwi-aor--- linear /dev/sdab:34-35
  [lv_rimage_3] iwi-XXr--- linear /dev/sdac:3-35     0
  [lv_rimage_4] iwi-aor--- linear /dev/sdad:1-33     0
  [lv_rimage_5] iwi-aor--- linear /dev/sdae:1-33     0
```

```
  [lv_rimage_6] iwi-aor--- linear /dev/sdaf:1-33     0
  [lv_rimage_7] iwi-aor--- linear /dev/sdag:1-33     0
  [lv_rmeta_0]  ewi-aor--- linear /dev/sda:0-0
  [lv_rmeta_1]  ewi-aor--- linear /dev/sdaa:0-0
  [lv_rmeta_2]  ewi-aor--- linear /dev/sdab:0-0
  [lv_rmeta_3]  ewi-aor--- linear /dev/sdac:0-0
  [lv_rmeta_4]  ewi-aor--- linear /dev/sdad:0-0
  [lv_rmeta_5]  ewi-aor--- linear /dev/sdae:0-0
  [lv_rmeta_6]  ewi-aor--- linear /dev/sdaf:0-0
  [lv_rmeta_7]  ewi-aor--- linear /dev/sdag:0-0
```

raid10 allows to add stripes but can't remove them.

A more elaborate example to convert from linear to striped with interim conversions to raid1 then raid5 followed by restripe (4 steps).

We start with the linear LV.

```
# lvs -a -o name,size,segtype,syncpercent,datastripes,\
           stripesize,reshapelenle,devices vg
  LV   LSize   Type   Cpy%Sync #DStr Stripe RSize Devices
  lv   128.00m linear             1     0       /dev/sda(0)
```

Then convert it to a 2-way raid1.

```
# lvconvert --mirrors 1 vg/lv
  Logical volume vg/lv successfully converted.
# lvs -a -o name,size,segtype,datastripes,\
           stripesize,reshapelenle,devices vg
  LV            LSize   Type   #DStr Stripe RSize Devices
  lv            128.00m raid1      2     0       lv_rimage_0(0),\
                                                 lv_rimage_1(0)
  [lv_rimage_0] 128.00m linear     1     0       /dev/sda(0)
  [lv_rimage_1] 128.00m linear     1     0       /dev/sdhx(1)
  [lv_rmeta_0]    4.00m linear     1     0       /dev/sda(32)
  [lv_rmeta_1]    4.00m linear     1     0       /dev/sdhx(0)
```

Once the raid1 LV is fully synchronized we convert it to raid5_n (only 2-way raid1 LVs can be converted to raid5). We select raid5_n here because it has dedicated parity SubLVs at the end and can be converted to striped directly without any additional conversion.

```
# lvconvert --type raid5_n vg/lv
  Using default stripesize 64.00 KiB.
  Logical volume vg/lv successfully converted.
# lvs -a -o name,size,segtype,syncpercent,datastripes,\
           stripesize,reshapelenle,devices vg
  LV            LSize   Type    #DStr Stripe RSize Devices
  lv            128.00m raid5_n     1 64.00k     0 lv_rimage_0(0),\
                                                   lv_rimage_1(0)
  [lv_rimage_0] 128.00m linear      1     0      0 /dev/sda(0)
  [lv_rimage_1] 128.00m linear      1     0      0 /dev/sdhx(1)
  [lv_rmeta_0]    4.00m linear      1     0        /dev/sda(32)
  [lv_rmeta_1]    4.00m linear      1     0        /dev/sdhx(0)
```

Now we'll change the number of data stripes from 1 to 5 and request 128K stripe size in one command. This will grow the size of the LV by a factor of 5 (we add 4 data stripes to the one given). That additonal space can be used by e.g. growing any contained filesystem or the LV can be reduced in size after the reshaping conversion has finished.

```
# lvconvert --stripesize 128k --stripes 5 vg/lv
  Converting stripesize 64.00 KiB of raid5_n LV vg/lv to 128.00 KiB.
  WARNING: Adding stripes to active logical volume vg/lv will grow \
  it from 32 to 160 extents!
  Run "lvresize -l32 vg/lv" to shrink it or use the additional capacity.
  Logical volume vg/lv successfully converted.
# lvs -a -o name,size,segtype,datastripes,\
           stripesize,reshapelenle,devices
  LV            LSize   Type    #DStr Stripe  RSize Devices
  lv            640.00m raid5_n     5 128.00k     6 lv_rimage_0(0),\
                                                    lv_rimage_1(0),\
                                                    lv_rimage_2(0),\
                                                    lv_rimage_3(0),\
                                                    lv_rimage_4(0),\
                                                    lv_rimage_5(0)
  [lv_rimage_0] 132.00m linear      1     0      1 /dev/sda(33)
  [lv_rimage_0] 132.00m linear      1     0        /dev/sda(0)
```

```
  [lv_rimage_1] 132.00m linear      1      0      1 /dev/sdhx(33)
  [lv_rimage_1] 132.00m linear      1      0        /dev/sdhx(1)
  [lv_rimage_2] 132.00m linear      1      0      1 /dev/sdhw(33)
  [lv_rimage_2] 132.00m linear      1      0        /dev/sdhw(1)
  [lv_rimage_3] 132.00m linear      1      0      1 /dev/sdhv(33)
  [lv_rimage_3] 132.00m linear      1      0        /dev/sdhv(1)
  [lv_rimage_4] 132.00m linear      1      0      1 /dev/sdhu(33)
  [lv_rimage_4] 132.00m linear      1      0        /dev/sdhu(1)
  [lv_rimage_5] 132.00m linear      1      0      1 /dev/sdht(33)
  [lv_rimage_5] 132.00m linear      1      0        /dev/sdht(1)
  [lv_rmeta_0]    4.00m linear      1      0        /dev/sda(32)
  [lv_rmeta_1]    4.00m linear      1      0        /dev/sdhx(0)
  [lv_rmeta_2]    4.00m linear      1      0        /dev/sdhw(0)
  [lv_rmeta_3]    4.00m linear      1      0        /dev/sdhv(0)
  [lv_rmeta_4]    4.00m linear      1      0        /dev/sdhu(0)
  [lv_rmeta_5]    4.00m linear      1      0        /dev/sdht(0)
```

Once the conversion has finished we can can convert to striped.

```
# lvconvert --type striped vg/lv
  Logical volume vg/lv successfully converted.
# lvs -a -o name,size,segtype,datastripes,\
          stripesize,reshapelenle,devices vg
  LV   LSize   Type    #DStr Stripe  RSize Devices
  lv   640.00m striped     5 128.00k       /dev/sda(33),\
                                           /dev/sdhx(33),\
                                           /dev/sdhw(33),\
                                           /dev/sdhv(33),\
                                           /dev/sdhu(33)
  lv   640.00m striped     5 128.00k       /dev/sda(0),\
                                           /dev/sdhx(1),\
                                           /dev/sdhw(1),\
                                           /dev/sdhv(1),\
                                           /dev/sdhu(1)
```

Reversing these steps will convert a given striped LV to linear.

Mind the facts that stripes are removed thus the capacity of the RaidLV will shrink and that changing the RaidLV layout will influence its performance.

"lvconvert --stripes 1 vg/lv" for converting to 1 stripe will inform upfront about the re? duced size to allow for resizing the content or growing the RaidLV before actually con? verting to 1 stripe. The --force option is needed to allow stripe removing conversions to prevent data loss.

Of course any interim step can be the intended last one (e.g. striped-> raid1).

RAID5 Variants

raid5_ls

? RAID5 left symmetric

? Rotating parity N with data restart

raid5_la

? RAID5 left symmetric

? Rotating parity N with data continuation

raid5_rs

? RAID5 right symmetric

? Rotating parity 0 with data restart

raid5_ra

? RAID5 right asymmetric

? Rotating parity 0 with data continuation

raid5_n

? RAID5 parity n

? Dedicated parity device n used for striped/raid0 conversions

? Used for RAID Takeover

RAID6 Variants

raid6

? RAID6 zero restart (aka left symmetric)

? Rotating parity 0 with data restart

? Same as raid6_zr

raid6_zr

? RAID6 zero restart (aka left symmetric)

? Rotating parity 0 with data restart

raid6_nr

? RAID6 N restart (aka right symmetric)

? Rotating parity N with data restart

raid6_nc

? RAID6 N continue

? Rotating parity N with data continuation

raid6_n_6

? RAID6 last parity devices

? Fixed dedicated last devices (P-Syndrome N-1 and Q-Syndrome N)

with striped data used for striped/raid0 conversions

? Used for RAID Takeover

raid6_{ls,rs,la,ra}_6

? RAID6 last parity device

? Dedicated last parity device used for conversions from/to

raid5_{ls,rs,la,ra}

raid6_ls_6

? RAID6 N continue

? Same as raid5_ls for N-1 devices with fixed Q-Syndrome N

? Used for RAID Takeover

raid6_la_6

? RAID6 N continue

? Same as raid5_la for N-1 devices with fixed Q-Syndrome N

? Used forRAID Takeover

raid6_rs_6

? RAID6 N continue

? Same as raid5_rs for N-1 devices with fixed Q-Syndrome N

? Used for RAID Takeover

raid6_ra_6

? RAID6 N continue

? ame as raid5_ra for N-1 devices with fixed Q-Syndrome N

? Used for RAID Takeover

History

The 2.6.38-rc1 version of the Linux kernel introduced a device-mapper target to interface with the software RAID (MD) personalities. This provided device-mapper with RAID 4/5/6

capabilities and a larger development community. Later, support for RAID1, RAID10, and RAID1E (RAID 10 variants) were added. Support for these new kernel RAID targets was added to LVM version 2.02.87. The capabilities of the LVM raid1 type have surpassed the old mirror type. raid1 is now recommended instead of mirror. raid1 became the default for mirroring in LVM version 2.02.100.