dm-raid

The device-mapper RAID (dm-raid) target provides a bridge from DM to MD. It allows the MD RAID drivers to be accessed using a device-mapper interface.

Mapping Table Interface

The target is named "raid" and it accepts the following parameters:

```
<raid_type> <#raid_params> <raid_params> \
  <#raid_devs> <metadata_dev0> <dev0> [.. <metadata_devN> <devN>]
```

<raid_type>:

raid0	RAID0 striping (no resilience)
raid1	RAID1 mirroring
raid4	RAID4 with dedicated last parity disk
raid5_n	RAID5 with dedicated last parity disk supporting takeover Same as raid4 • Transitory layout
raid5_la	RAID5 left asymmetric • rotating parity 0 with data continuation
raid5_ra	RAID5 right asymmetric • rotating parity N with data continuation
raid5_ls	RAID5 left symmetric • rotating parity 0 with data restart
raid5_rs	RAID5 right symmetric • rotating parity N with data restart
raid6_zr	RAID6 zero restart • rotating parity zero (left-to-right) with data restart
raid6_nr	RAID6 N restart • rotating parity N (right-to-left) with data restart
raid6_nc	RAID6 N continue • rotating parity N (right-to-left) with data continuation
raid6_n_6	RAID6 with dedicate parity disks • parity and Q-syndrome on the last 2 disks; layout for takeover from/to raid4/raid5_n
raid6_la_6	Same as "raid_la" plus dedicated last Q-syndrome disk • layout for takeover from raid5_la from/to raid6
raid6_ra_6	Same as "raid5_ra" dedicated last Q-syndrome disk • layout for takeover from raid5_ra from/to raid6
raid6_ls_6	Same as "raid5_ls" dedicated last Q-syndrome disk • layout for takeover from raid5_ls from/to raid6
raid6_rs_6	Same as "raid5_rs" dedicated last Q-syndrome disk • layout for takeover from raid5_rs from/to raid6
raid10	Various RAID10 inspired algorithms chosen by additional params (see raid10_format and raid10_copies below) • RAID10: Striped Mirrors (aka 'Striping on top of mirrors') • RAID1E: Integrated Adjacent Stripe Mirroring • RAID1E: Integrated Offset Stripe Mirroring • and other similar RAID10 variants

Reference: Chapter 4 of https://www.snia.org/sites/default/files/SNIA_DDF_Technical_Position_v2.0.pdf

<#raid_params>: The number of parameters that follow.

<raid_params> consists of

Mandatory parameters:

<chunk_size>:

Chunk size in sectors. This parameter is often known as "stripe size". It is the only mandatory parameter and is placed first.

followed by optional parameters (in any order):

[sync|nosync]

Force or prevent RAID initialization.

[rebuild <idx>]

Rebuild drive number 'idx' (first drive is 0).

[daemon_sleep <ms>]

Interval between runs of the bitmap daemon that clear bits. A longer interval means less bitmap I/O but resyncing after a failure is likely to take longer.

[min_recovery_rate <kB/sec/disk>]

Throttle RAID initialization

[max_recovery_rate <kB/sec/disk>]

Throttle RAID initialization

[write_mostly <idx>]

Mark drive index 'idx' write-mostly.

[max_write_behind <sectors>]

See '--write-behind=' (man mdadm)

[stripe_cache <sectors>]

Stripe cache size (RAID 4/5/6 only)

[region_size <sectors>]

The region_size multiplied by the number of regions is the logical size of the array. The bitmap records the device synchronisation state for each region.

[raid10_copies <# copies>], [raid10_format <near|far|offset>]

These two options are used to alter the default layout of a RAID10 configuration. The number of copies is can be specified, but the default is 2. There are also three variations to how the copies are laid down - the default is "near". Near copies are what most people think of with respect to mirroring. If these options are left unspecified, or 'raid10_copies 2' and/or 'raid10_format near' are given, then the layouts for 2, 3 and 4 devices are:

2 drives	**3 drives**	**4 drives**
A1 A1	A1 A1 A2	A1 A1 A2 A2
A2 A2	A2 A3 A3	A3 A3 A4 A4
A3 A3	A4 A4 A5	A5 A5 A6 A6
A4 A4	A5 A6 A6	A7 A7 A8 A8

The 2-device layout is equivalent 2-way RAID1. The 4-device layout is what a traditional RAID10 would look like. The 3-device layout is what might be called a 'RAID1E - Integrated Adjacent Stripe Mirroring'.

If 'raid10_copies 2' and 'raid10_format far', then the layouts for 2, 3 and 4 devices are:

2 drives	**3 drives**	**4 drives**
A1 A2	A1 A2 A3	A1 A2 A3 A4
A3 A4	A4 A5 A6	A5 A6 A7 A8
A5 A6	A7 A8 A9	A9 A10 A11 A12
A2 A1	A3 A1 A2	A2 A1 A4 A3
A4 A3	A6 A4 A5	A6 A5 A8 A7
A6 A5	A9 A7 A8	A10 A9 A12 A11

If 'raid10_copies 2' and 'raid10_format offset', then the layouts for 2, 3 and 4 devices are:

2 drives	**3 drives**	**4 drives**
A1 A2	A1 A2 A3	A1 A2 A3 A4

2 drives	3 drives	4 drives
A2 A1	A3 A1 A2	A2 A1 A4 A3
A3 A4	A4 A5 A6	A5 A6 A7 A8
A4 A3	A6 A4 A5	A6 A5 A8 A7
A5 A6	A7 A8 A9	A9 A10 A11 A12
A6 A5	A9 A7 A8	A10 A9 A12 A11

Here we see layouts closely akin to 'RAID1E - Integrated Offset Stripe Mirroring'.

[delta_disks <N>]

The delta_disks option value (-251 < N < +251) triggers device removal (negative value) or device addition (positive value) to any reshape supporting raid levels 4/5/6 and 10. RAID levels 4/5/6 allow for addition of devices (metadata and data device tuple), raid10_near and raid10_offset only allow for device addition. raid10_far does not support any reshaping at all. A minimum of devices have to be kept to enforce resilience, which is 3 devices for raid4/5 and 4 devices for raid6.

[data_offset <sectors>]

This option value defines the offset into each data device where the data starts. This is used to provide out-of-place reshaping space to avoid writing over data while changing the layout of stripes, hence an interruption/crash may happen at any time without the risk of losing data. E.g. when adding devices to an existing raid set during forward reshaping, the out-of-place space will be allocated at the beginning of each raid device. The kernel raid4/5/6/10 MD personalities supporting such device addition will read the data from the existing first stripes (those with smaller number of stripes) starting at data_offset to fill up a new stripe with the larger number of stripes, calculate the redundancy blocks (CRC/Q-syndrome) and write that new stripe to offset 0. Same will be applied to all N-1 other new stripes. This out-of-place scheme is used to change the RAID type (i.e. the allocation algorithm) as well, e.g. changing from raid5_ls to raid5_n.

[journal_dev <dev>]

This option adds a journal device to raid4/5/6 raid sets and uses it to close the 'write hole' caused by the non-atomic updates to the component devices which can cause data loss during recovery. The journal device is used as writethrough thus causing writes to be throttled versus non-journaled raid4/5/6 sets. Takeover/reshape is not possible with a raid4/5/6 journal device; it has to be deconfigured before requesting these.

[journal_mode <mode>]

This option sets the caching mode on journaled raid4/5/6 raid sets (see 'journal_dev <dev>' above) to 'writethrough' or 'writeback'. If 'writeback' is selected the journal device has to be resilient and must not suffer from the 'write hole' problem itself (e.g. use raid1 or raid10) to avoid a single point of failure.

<#raid_devs>: The number of devices composing the array.

Each device consists of two entries. The first is the device containing the metadata (if any); the second is the one containing the data. A Maximum of 64 metadata/data device entries are supported up to target version 1.8.0. 1.9.0 supports up to 253 which is enforced by the used MD kernel runtime.

If a drive has failed or is missing at creation time, a '-' can be given for both the metadata and data drives for a given position.

Example Tables

```
# RAID4 - 4 data drives, 1 parity (no metadata devices)
# No metadata devices specified to hold superblock/bitmap info
# Chunk size of 1MiB
# (Lines separated for easy reading)

0 1960893648 raid \
        raid4 1 2048 \
        5 - 8:17 - 8:33 - 8:49 - 8:65 - 8:81

# RAID4 - 4 data drives, 1 parity (with metadata devices)
# Chunk size of 1MiB, force RAID initialization,
#       min recovery rate at 20 kiB/sec/disk

0 1960893648 raid \
        raid4 4 2048 sync min_recovery_rate 20 \
        5 8:17 8:18 8:33 8:34 8:49 8:50 8:65 8:66 8:81 8:82
```

Status Output

'dmsetup table' displays the table used to construct the mapping. The optional parameters are always printed in the order listed above with "sync" or "nosync" always output ahead of the other arguments, regardless of the order used when originally loading the table. Arguments that can be repeated are ordered by value.

'dmsetup status' yields information on the state and health of the array. The output is as follows (normally a single line, but expanded here for clarity):

```
1: <s> <l> raid \
2:      <raid_type> <#devices> <health_chars> \
3:      <sync_ratio> <sync_action> <mismatch_cnt>
```

Line 1 is the standard output produced by device-mapper.

Line 2 & 3 are produced by the raid target and are best explained by example:

```
0 1960893648 raid raid4 5 AAAAA 2/490221568 init 0
```

Here we can see the RAID type is raid4, there are 5 devices - all of which are 'A'live, and the array is 2/490221568 complete with its initial recovery. Here is a fuller description of the individual fields:

<raid_type>	Same as the <raid_type> used to create the array.
<health_chars>	One char for each device, indicating: • 'A' = alive and in-sync • 'a' = alive but not in-sync • 'D' = dead/failed.
<sync_ratio>	The ratio indicating how much of the array has undergone the process described by 'sync_action'. If the 'sync_action' is "check" or "repair", then the process of "resync" or "recover" can be considered complete.
<sync_action>	One of the following possible states: idle • No synchronization action is being performed. frozen • The current action has been halted. resync • Array is undergoing its initial synchronization or is resynchronizing after an unclean shutdown (possibly aided by a bitmap). recover • A device in the array is being rebuilt or replaced. check • A user-initiated full check of the array is being performed. All blocks are read and checked for consistency. The number of discrepancies found are recorded in <mismatch_cnt>. No changes are made to the array by this action. repair • The same as "check", but discrepancies are corrected. reshape • The array is undergoing a reshape.
<mismatch_cnt>	The number of discrepancies found between mirror copies in RAID1/10 or wrong parity values found in RAID4/5/6. This value is valid only after a "check" of the array is performed. A healthy array has a 'mismatch_cnt' of 0.
<data_offset>	The current data offset to the start of the user data on each component device of a raid set (see the respective raid parameter to support out-of-place reshaping).
<journal_char>	• 'A' - active write-through journal device. • 'a' - active write-back journal device. • 'D' - dead journal device. • '-' - no journal device.

Message Interface

The dm-raid target will accept certain actions through the 'message' interface. ('man dmsetup' for more information on the message interface.) These actions include:

"idle"	Halt the current sync action.
"frozen"	Freeze the current sync action.
"resync"	Initiate/continue a resync.
"recover"	Initiate/continue a recover process.

"check"	Initiate a check (i.e. a "scrub") of the array.
"repair"	Initiate a repair of the array.

Discard Support

The implementation of discard support among hardware vendors varies. When a block is discarded, some storage devices will return zeroes when the block is read. These devices set the 'discard_zeroes_data' attribute. Other devices will return random data. Confusingly, some devices that advertise 'discard_zeroes_data' will not reliably return zeroes when discarded blocks are read! Since RAID 4/5/6 uses blocks from a number of devices to calculate parity blocks and (for performance reasons) relies on 'discard_zeroes_data' being reliable, it is important that the devices be consistent. Blocks may be discarded in the middle of a RAID 4/5/6 stripe and if subsequent read results are not consistent, the parity blocks may be calculated differently at any time; making the parity blocks useless for redundancy. It is important to understand how your hardware behaves with discards if you are going to enable discards with RAID 4/5/6.

Since the behavior of storage devices is unreliable in this respect, even when reporting 'discard_zeroes_data', by default RAID 4/5/6 discard support is disabled -- this ensures data integrity at the expense of losing some performance.

Storage devices that properly support 'discard_zeroes_data' are increasingly whitelisted in the kernel and can thus be trusted.

For trusted devices, the following dm-raid module parameter can be set to safely enable discard support for RAID 4/5/6:

'devices_handle_discards_safely'

Version History

```
1.0.0  Initial version.  Support for RAID 4/5/6
1.1.0  Added support for RAID 1
1.2.0  Handle creation of arrays that contain failed devices.
1.3.0  Added support for RAID 10
1.3.1  Allow device replacement/rebuild for RAID 10
1.3.2  Fix/improve redundancy checking for RAID10
1.4.0  Non-functional change.  Removes arg from mapping function.
1.4.1  RAID10 fix redundancy validation checks (commit 55ebbb5).
1.4.2  Add RAID10 "far" and "offset" algorithm support.
1.5.0  Add message interface to allow manipulation of the sync_action.
       New status (STATUSTYPE_INFO) fields: sync_action and mismatch_cnt.
1.5.1  Add ability to restore transiently failed devices on resume.
1.5.2  'mismatch_cnt' is zero unless [last_]sync_action is "check".
1.6.0  Add discard support (and devices_handle_discard_safely module param).
1.7.0  Add support for MD RAID0 mappings.
1.8.0  Explicitly check for compatible flags in the superblock metadata
       and reject to start the raid set if any are set by a newer
       target version, thus avoiding data corruption on a raid set
       with a reshape in progress.
1.9.0  Add support for RAID level takeover/reshape/region size
       and set size reduction.
1.9.1  Fix activation of existing RAID 4/10 mapped devices
1.9.2  Don't emit '- -' on the status table line in case the constructor
       fails reading a superblock. Correctly emit 'maj:min1 maj:min2' and
       'D' on the status line.  If '- -' is passed into the constructor, emit
       '- -' on the table line and '-' as the status line health character.
1.10.0 Add support for raid4/5/6 journal device
1.10.1 Fix data corruption on reshape request
1.11.0 Fix table line argument order
       (wrong raid10_copies/raid10_format sequence)
1.11.1 Add raid4/5/6 journal write-back support via journal_mode option
1.12.1 Fix for MD deadlock between mddev_suspend() and md_write_start() available
1.13.0 Fix dev_health status at end of "recover" (was 'a', now 'A')
1.13.1 Fix deadlock caused by early md_stop_writes().  Also fix size an
       state races.
1.13.2 Fix raid redundancy validation and avoid keeping raid set frozen
1.14.0 Fix reshape race on small devices.  Fix stripe adding reshape
       deadlock/potential data corruption.  Update superblock when
       specific devices are requested via rebuild.  Fix RAID leg
       rebuild errors.
1.15.0 Fix size extensions not being synchronized in case of new MD bitmap
       pages allocated;  also fix those not occurring after previous reductions
1.15.1 Fix argument count and arguments for rebuild/write_mostly/journal_(dev|mode)
       on the status line.
```