NAME

lvmthin — LVM thin provisioning

DESCRIPTION

Blocks in a standard logical volume are allocated when the LV is created, but blocks in a thin provisioned logical volume are allocated as they are written. Because of this, a thin provisioned LV is given a virtual size, and can then be much larger than physically available storage. The amount of physical storage provided for thin provisioned LVs can be increased later as the need arises.

Blocks in a standard LV are allocated (during creation) from the VG, but blocks in a thin LV are allocated (during use) from a special "thin pool LV". The thin pool LV contains blocks of physical storage, and blocks in thin LVs just reference blocks in the thin pool LV.

A thin pool LV must be created before thin LVs can be created within it. A thin pool LV is created by combining two standard LVs: a large data LV that will hold blocks for thin LVs, and a metadata LV that will hold metadata. The metadata tracks which data blocks belong to each thin LV.

Snapshots of thin LVs are efficient because the data blocks common to a thin LV and any of its snapshots are shared. Snapshots may be taken of thin LVs or of other thin snapshots. Blocks common to recursive snapshots are also shared in the thin pool. There is no limit to or degradation from sequences of snapshots.

As thin LVs or snapshot LVs are written to, they consume data blocks in the thin pool. As free data blocks in the pool decrease, more free blocks may need to be supplied. This is done by extending the thin pool data LV with additional physical space from the VG. Removing thin LVs or snapshots from the thin pool can also free blocks in the thin pool. However, removing LVs is not always an effective way of freeing space in a thin pool because the amount is limited to the number of blocks not shared with other LVs in the pool.

Incremental block allocation from thin pools can cause thin LVs to become fragmented. Standard LVs generally avoid this problem by allocating all the blocks at once during creation.

Thin Terms

ThinDataLV

thin data LV
large LV created in a VG
used by thin pool to store ThinLV blocks

ThinMetaLV

thin metadata LV
small LV created in a VG
used by thin pool to track data block usage

ThinPoolLV

thin pool LV
combination of ThinDataLV and ThinMetaLV
contains ThinLVs and SnapLVs

ThinLV

thin LV
created from ThinPoolLV
appears blank after creation

SnapLV
snapshot LV
created from ThinPoolLV
appears as a snapshot of another LV after creation

Thin Usage

The primary method for using lvm thin provisioning:

1. create ThinDataLV

Create an LV that will hold thin pool data.

lvcreate −n ThinDataLV −L LargeSize VG

Example

```
# lvcreate −n pool0 −L 10G vg
```

2. create ThinMetaLV

Create an LV that will hold thin pool metadata.

lvcreate −n ThinMetaLV −L SmallSize VG

Example

```
# lvcreate −n pool0meta −L 1G vg

# lvs
  LV        VG Attr       LSize
  pool0     vg -wi-a----- 10.00g
  pool0meta vg -wi-a----- 1.00g
```

3. create ThinPoolLV

Combine the data and metadata LVs into a thin pool LV.
ThinDataLV is renamed to hidden ThinPoolLV_tdata.
ThinMetaLV is renamed to hidden ThinPoolLV_tmeta.
The new ThinPoolLV takes the previous name of ThinDataLV.

lvconvert −−type thin-pool −−poolmetadata VG/ThinMetaLV VG/ThinDataLV

Example

```
# lvconvert −−type thin-pool −−poolmetadata vg/pool0meta vg/pool0

# lvs vg/pool0
  LV    VG Attr       LSize  Pool Origin Data% Meta%
  pool0 vg twi-a-tz-- 10.00g      0.00   0.00

# lvs −a
  LV            VG Attr       LSize
  pool0         vg twi-a-tz-- 10.00g
  [pool0_tdata] vg Twi-ao---- 10.00g
  [pool0_tmeta] vg ewi-ao---- 1.00g
```

4. create ThinLV

Create a new thin LV from the thin pool LV.
The thin LV is created with a virtual size.
Multiple new thin LVs may be created in the thin pool.
Thin LV names must be unique in the VG.
The '--type thin' option is inferred from the virtual size option.
The --thinpool argument specifies which thin pool will
contain the ThinLV.

lvcreate −n ThinLV −V VirtualSize −−thinpool VG/ThinPoolLV

Example
Create a thin LV in a thin pool:
```
# lvcreate −n thin1 −V 1T −−thinpool vg/pool0
```

Create another thin LV in the same thin pool:
```
# lvcreate −n thin2 −V 1T −−thinpool vg/pool0
```

```
# lvs vg/thin1 vg/thin2
  LV    VG Attr       LSize Pool  Origin Data%
  thin1 vg Vwi-a-tz-- 1.00t pool0        0.00
  thin2 vg Vwi-a-tz-- 1.00t pool0        0.00
```

5. create SnapLV

Create snapshots of an existing ThinLV or SnapLV.
Do not specify **−L**, **−−size** when creating a thin snapshot.
A size argument will cause an old COW snapshot to be created.

lvcreate −n SnapLV −s VG/ThinLV
lvcreate −n SnapLV −s VG/PrevSnapLV

Example
Create first snapshot of an existing ThinLV:
```
# lvcreate −n thin1s1 −s vg/thin1
```

Create second snapshot of the same ThinLV:
```
# lvcreate −n thin1s2 −s vg/thin1
```

Create a snapshot of the first snapshot:
```
# lvcreate −n thin1s1s1 −s vg/thin1s1
```

```
# lvs vg/thin1s1 vg/thin1s2 vg/thin1s1s1
  LV        VG Attr       LSize Pool  Origin
  thin1s1   vg Vwi---tz-k 1.00t pool0 thin1
  thin1s2   vg Vwi---tz-k 1.00t pool0 thin1
  thin1s1s1 vg Vwi---tz-k 1.00t pool0 thin1s1
```

6. activate SnapLV

Thin snapshots are created with the persistent "activation skip" flag, indicated by the "k" attribute. Use −K with lvchange or vgchange to activate thin snapshots with the "k" attribute.

lvchange −ay −K VG/SnapLV

Example

```
# lvchange −ay −K vg/thin1s1

# lvs vg/thin1s1
  LV      VG Attr       LSize Pool  Origin
  thin1s1 vg Vwi-a-tz-k 1.00t pool0 thin1
```

Thin Topics

Alternate syntax for specifying type thin−pool
Automatic pool metadata LV
Specify devices for data and metadata LVs
Tolerate device failures using raid
Spare metadata LV
Metadata check and repair
Activation of thin snapshots
Removing thin pool LVs, thin LVs and snapshots
Manually manage free data space of thin pool LV
Manually manage free metadata space of a thin pool LV
Using fstrim to increase free space in a thin pool LV
Automatically extend thin pool LV
Data space exhaustion
Metadata space exhaustion
Zeroing
Discard
Chunk size
Size of pool metadata LV
Create a thin snapshot of an external, read only LV
Convert a standard LV to a thin LV with an external origin
Single step thin pool LV creation
Single step thin pool LV and thin LV creation
Merge thin snapshots
XFS on snapshots

Alternate syntax for specifying type thin−pool

The fully specified syntax for creating a thin pool LV shown above is:

lvconvert −−type thin-pool −−poolmetadata VG/ThinMetaLV VG/ThinDataLV

An existing LV is converted to a thin pool by changing its type to thin-pool. An alternate syntax may be used for the same operation:

lvconvert −−thinpool VG/ThinDataLV −−poolmetadata VG/ThinMetaLV

The thin-pool type is inferred by lvm; the --thinpool option is not an alias for --type thin-pool. The use of the --thinpool option here is different from the use of the --thinpool option when creating a thin LV, where it specifies the pool in which the thin LV is created.

Automatic pool metadata LV

A thin data LV can be converted to a thin pool LV without specifying a thin pool metadata LV. LVM automatically creates a metadata LV from the same VG.

lvcreate −n ThinDataLV −L LargeSize VG
lvconvert −−type thin−pool VG/ThinDataLV

Example
```
# lvcreate −n pool0 −L 10G vg
# lvconvert −−type thin−pool vg/pool0

# lvs −a
  pool0           vg          twi-a-tz--  10.00g
  [pool0_tdata]   vg          Twi-ao----  10.00g
  [pool0_tmeta]   vg          ewi-ao----  16.00m
```

Specify devices for data and metadata LVs

The data and metadata LVs in a thin pool are best created on separate physical devices. To do that, specify the device name(s) at the end of the lvcreate line. It can be especially helpful to use fast devices for the metadata LV.

lvcreate −n ThinDataLV −L LargeSize VG LargePV
lvcreate −n ThinMetaLV −L SmallSize VG SmallPV
lvconvert −−type thin−pool −−poolmetadata VG/ThinMetaLV VG/ThinDataLV

Example
```
# lvcreate −n pool0 −L 10G vg /dev/sdA
# lvcreate −n pool0meta −L 1G vg /dev/sdB
# lvconvert −−type thin−pool −−poolmetadata vg/pool0meta vg/pool0
```

lvm.conf(5) **thin_pool_metadata_require_separate_pvs**
controls the default PV usage for thin pool creation.

Tolerate device failures using raid

To tolerate device failures, use raid for the pool data LV and pool metadata LV. This is especially recommended for pool metadata LVs.

lvcreate −−type raid1 −m 1 −n ThinMetaLV −L SmallSize VG PVA PVB
lvcreate −−type raid1 −m 1 −n ThinDataLV −L LargeSize VG PVC PVD
lvconvert −−type thin−pool −−poolmetadata VG/ThinMetaLV VG/ThinDataLV

Example
```
# lvcreate −−type raid1 −m 1 −n pool0 −L 10G vg /dev/sdA /dev/sdB
# lvcreate −−type raid1 −m 1 −n pool0meta −L 1G vg /dev/sdC /dev/sdD
# lvconvert −−type thin−pool −−poolmetadata vg/pool0meta vg/pool0
```

Spare metadata LV

The first time a thin pool LV is created, lvm will create a spare metadata LV in the VG. This behavior can be controlled with the option −−poolmetadataspare y|n. (Future thin pool creations will also attempt to create the pmspare LV if none exists.)

To create the pmspare ("pool metadata spare") LV, lvm first creates an LV with a default name, e.g. lvol0, and then converts this LV to a hidden LV with the _pmspare suffix, e.g. lvol0_pmspare.

One pmspare LV is kept in a VG to be used for any thin pool.

The pmspare LV cannot be created explicitly, but may be removed explicitly.

Example

```
# lvcreate −n pool0 −L 10G vg
# lvcreate −n pool0meta −L 1G vg
# lvconvert −−type thin−pool −−poolmetadata vg/pool0meta vg/pool0

# lvs −a
  [lvol0_pmspare] vg          ewi-------
  pool0           vg          twi---tz--
  [pool0_tdata]   vg          Twi-------
  [pool0_tmeta]   vg          ewi-------
```

The "Metadata check and repair" section describes the use of the pmspare LV.

Metadata check and repair

If thin pool metadata is damaged, it may be repairable. Checking and repairing thin pool metadata is analagous to running fsck on a file system.

When a thin pool LV is activated, lvm runs the thin_check command to check the correctness of the metadata on the pool metadata LV.

lvm.conf(5) **thin_check_executable**
can be set to an empty string ("") to disable the thin_check step. This is not recommended.

lvm.conf(5) **thin_check_options**
controls the command options used for the thin_check command.

If the thin_check command finds a problem with the metadata, the thin pool LV is not activated, and the thin pool metadata needs to be repaired.

Simple repair commands are not always successful. Advanced repair may require editing thin pool metadata and lvm metadata. Newer versions of the kernel and lvm tools may be more successful at repair. Report the details of damaged thin metadata to get the best advice on recovery.

Command to repair a thin pool:
lvconvert −−repair VG/ThinPoolLV

Repair performs the following steps:

1. Creates a new, repaired copy of the metadata.
lvconvert runs the thin_repair command to read damaged metadata from the existing pool metadata LV, and writes a new repaired copy to the VG's pmspare LV.

2. Replaces the thin pool metadata LV.
If step 1 is successful, the thin pool metadata LV is replaced with the pmspare LV containing the corrected metadata. The previous thin pool metadata LV, containing the damaged metadata, becomes visible with the new name ThinPoolLV_tmetaN (where N is 0,1,...).

If the repair works, the thin pool LV and its thin LVs can be activated, and the LV containing the damaged thin pool metadata can be removed. It may be useful to move the new metadata LV (previously pmspare) to a better PV.

If the repair does not work, the thin pool LV and its thin LVs are lost.

If metadata is manually restored with thin_repair directly, the pool metadata LV can be manually swapped with another LV containing new metadata:

lvconvert −−thinpool VG/ThinPoolLV −−poolmetadata VG/NewThinMetaLV

Activation of thin snapshots

When a thin snapshot LV is created, it is by default given the "activation skip" flag. This flag is indicated by the "k" attribute displayed by lvs:

```
# lvs vg/thin1s1
  LV         VG  Attr       LSize Pool  Origin
  thin1s1    vg  Vwi---tz-k 1.00t pool0 thin1
```

This flag causes the snapshot LV to be skipped, i.e. not activated, by normal activation commands. The skipping behavior does not apply to deactivation commands.

A snapshot LV with the "k" attribute can be activated using the −K (or −−ignoreactivationskip) option in addition to the standard −ay (or −−activate y) option.

Command to activate a thin snapshot LV:
lvchange −ay −K VG/SnapLV

The persistent "activation skip" flag can be turned off during lvcreate, or later with lvchange using the −kn (or −−setactivationskip n) option. It can be turned on again with −ky (or −−setactivationskip y).

When the "activation skip" flag is removed, normal activation commands will activate the LV, and the −K activation option is not needed.

Command to create snapshot LV without the activation skip flag:
lvcreate −kn −n SnapLV −s VG/ThinLV

Command to remove the activation skip flag from a snapshot LV:
lvchange −kn VG/SnapLV

lvm.conf(5) **auto_set_activation_skip**
controls the default activation skip setting used by lvcreate.

Removing thin pool LVs, thin LVs and snapshots

Removing a thin LV and its related snapshots returns the blocks it used to the thin pool LV. These blocks will be reused for other thin LVs and snapshots.

Removing a thin pool LV removes both the data LV and metadata LV and returns the space to the VG.

lvremove of thin pool LVs, thin LVs and snapshots cannot be reversed with vgcfgrestore.

vgcfgbackup does not back up thin pool metadata.

Manually manage free data space of thin pool LV

The available free space in a thin pool LV can be displayed with the lvs command. Free space can be added by extending the thin pool LV.

Command to extend thin pool data space:
lvextend −L Size VG/ThinPoolLV

Example
1. A thin pool LV is using 26.96% of its data blocks.
```
# lvs
  LV    VG    Attr       LSize   Pool  Origin Data%
  pool0 vg    twi-a-tz--  10.00g               26.96
```

2. Double the amount of physical space in the thin pool LV.
```
# lvextend −L+10G vg/pool0
```

3. The percentage of used data blocks is half the previous value.
```
# lvs
  LV    VG    Attr       LSize   Pool  Origin Data%
  pool0 vg    twi-a-tz--  20.00g               13.48
```

Other methods of increasing free data space in a thin pool LV include removing a thin LV and its related snapsots, or running fstrim on the file system using a thin LV.

Manually manage free metadata space of a thin pool LV

The available metadata space in a thin pool LV can be displayed with the lvs −o+metadata_percent command.

Command to extend thin pool metadata space:
lvextend −L Size VG/ThinPoolLV_tmeta

Example
1. A thin pool LV is using 12.40% of its metadata blocks.
```
# lvs −oname,size,data_percent,metadata_percent vg/pool0
  LV    LSize  Data%  Meta%
  pool0 20.00g 13.48  12.40
```

2. Display a thin pool LV with its component thin data LV and thin metadata LV.

```
# lvs −a −oname,attr,size vg
  LV              Attr       LSize
  pool0           twi-a-tz--  20.00g
  [pool0_tdata]   Twi-ao----  20.00g
  [pool0_tmeta]   ewi-ao----  12.00m
```

3. Double the amount of physical space in the thin metadata LV.

```
# lvextend −L+12M vg/pool0_tmeta
```

4. The percentage of used metadata blocks is half the previous value.

```
# lvs −a −oname,size,data_percent,metadata_percent vg
  LV              LSize   Data%  Meta%
  pool0           20.00g  13.48   6.20
  [pool0_tdata]   20.00g
  [pool0_tmeta]   24.00m
```

Using fstrim to increase free space in a thin pool LV

Removing files in a file system on top of a thin LV does not generally add free space back to the thin pool. Manually running the fstrim command can return space back to the thin pool that had been used by removed files. fstrim uses discards and will not work if the thin pool LV has discards mode set to ignore.

Example

A thin pool has 10G of physical data space, and a thin LV has a virtual size of 100G. Writing a 1G file to the file system reduces the free space in the thin pool by 10% and increases the virtual usage of the file system by 1%. Removing the 1G file restores the virtual 1% to the file system, but does not restore the physical 10% to the thin pool. The fstrim command restores the physical space to the thin pool.

```
# lvs −a −oname,attr,size,pool_lv,origin,data_percent,metadata_percent vg
LV              Attr       LSize   Pool  Origin Data%  Meta%
pool0           twi-a-tz--  10.00g              47.01  21.03
thin1           Vwi-aotz-- 100.00g pool0         2.70

# df −h /mnt/X
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/vg-thin1   99G  1.1G   93G   2% /mnt/X

# dd if=/dev/zero of=/mnt/X/1Gfile bs=4096 count=262144; sync

# lvs
pool0           vg   twi-a-tz--  10.00g              57.01  25.26
thin1           vg   Vwi-aotz-- 100.00g pool0         3.70

# df −h /mnt/X
/dev/mapper/vg-thin1   99G  2.1G   92G   3% /mnt/X

# rm /mnt/X/1Gfile

# lvs
pool0           vg   twi-a-tz--  10.00g              57.01  25.26
thin1           vg   Vwi-aotz-- 100.00g pool0         3.70
```

```
# df −h /mnt/X
/dev/mapper/vg-thin1   99G  1.1G   93G   2% /mnt/X

# fstrim −v /mnt/X

# lvs
pool0          vg   twi-a-tz--  10.00g             47.01  21.03
thin1          vg   Vwi-aotz-- 100.00g pool0         2.70
```

The "Discard" section covers an option for automatically freeing data space in a thin pool.

Automatically extend thin pool LV

The lvm daemon dmeventd (lvm2-monitor) monitors the data usage of thin pool LVs and extends them when the usage reaches a certain level. The necessary free space must exist in the VG to extend thin pool LVs. Monitoring and extension of thin pool LVs are controlled independently.

monitoring

When a thin pool LV is activated, dmeventd will begin monitoring it by default.

Command to start or stop dmeventd monitoring a thin pool LV:
lvchange −−monitor {y|n} VG/ThinPoolLV

The current dmeventd monitoring status of a thin pool LV can be displayed with the command lvs -o+seg_monitor.

autoextend

dmeventd should be configured to extend thin pool LVs before all data space is used. Warnings are emitted through syslog when the use of a thin pool reaches 80%, 85%, 90% and 95%. (See the section "Data space exhaustion" for the effects of not extending a thin pool LV.) The point at which dmeventd extends thin pool LVs, and the amount are controlled with two configuration settings:

lvm.conf(5) **thin_pool_autoextend_threshold**
is a percentage full value that defines when the thin pool LV should be extended. Setting this to 100 disables automatic extention. The minimum value is 50.

lvm.conf(5) **thin_pool_autoextend_percent**
defines how much extra data space should be added to the thin pool LV from the VG, in percent of its current size.

disabling

There are multiple ways that extension of thin pools could be prevented:

- If the dmeventd daemon is not running, no monitoring or automatic extension will occur.

- Even when dmeventd is running, all monitoring can be disabled with the lvm.conf monitoring setting.

- To activate or create a thin pool LV without interacting with dmeventd, the --ignoremonitoring option can be used. With this option, the command will not ask dmeventd to monitor the thin pool LV.

- Setting thin_pool_autoextend_threshould to 100 disables automatic extension of thin pool LVs, even if they are being monitored by dmeventd.

Example
If thin_pool_autoextend_threshold is 70 and thin_pool_autoextend_percent is 20, whenever a pool exceeds 70% usage, it will be extended by another 20%. For a 1G pool, using 700M will trigger a resize to 1.2G. When the usage exceeds 840M, the pool will be extended to 1.44G, and so on.

Data space exhaustion

When properly managed, thin pool data space should be extended before it is all used (see the section "Automatically extend thin pool LV"). If thin pool data space is already exhausted, it can still be extended (see the section "Manually manage free data space of thin pool LV".)

The behavior of a full thin pool is configurable with the --errorwhenfull y|n option to lvcreate or lvchange. The errorwhenfull setting applies only to writes; reading thin LVs can continue even when data space is exhausted.

Command to change the handling of a full thin pool:
lvchange --errorwhenfull {y|n} VG/ThinPoolLV

lvm.conf(5) **error_when_full**
controls the default error when full behavior.

The current setting of a thin pool LV can be displayed with the command: lvs -o+lv_when_full.

The errorwhenfull setting does not effect the monitoring and autoextend settings, and the monitoring/autoextend settings do not effect the errorwhenfull setting. It is only when monitoring/autoextend are not effective that the thin pool becomes full and the errorwhenfull setting is applied.

errorwhenfull n

This is the default. Writes to thin LVs are accepted and queued, with the expectation that pool data space will be extended soon. Once data space is extended, the queued writes will be processed, and the thin pool will return to normal operation.

While waiting to be extended, the thin pool will queue writes for up to 60 seconds (the default). If data space has not been extended after this time, the queued writes will return an error to the caller, e.g. the file system. This can result in file system corruption for non-journaled file systems that may require fsck. When a thin pool returns errors for writes to a thin LV, any file system is subject to losing unsynced user data.

The 60 second timeout can be changed or disabled with the dm−thin−pool kernel module option **no_space_timeout.** This option sets the number of seconds that thin pools will queue writes. If set to 0, writes will not time out. Disabling timeouts can result in the system running out of resources, memory exhaustion, hung tasks, and deadlocks. (The timeout applies to all thin pools on the system.)

errorwhenfull y

Writes to thin LVs immediately return an error, and no writes are queued. In the case of a file system, this can result in corruption that may require fsck (the specific consequences depend on the thin LV user.)

data percent

When data space is exhausted, the lvs command displays 100 under Data% for the thin pool LV:

```
# lvs vg/pool0
  LV     VG        Attr       LSize   Pool  Origin Data%
  pool0  vg        twi-a-tz-- 512.00m              100.00
```

causes

A thin pool may run out of data space for any of the following reasons:

- Automatic extension of the thin pool is disabled, and the thin pool is not manually extended. (Disabling automatic extension is not recommended.)
- The dmeventd daemon is not running and the thin pool is not manually extended. (Disabling dmeventd is not recommended.)
- Automatic extension of the thin pool is too slow given the rate of writes to thin LVs in the pool. (This can be addressed by tuning the thin_pool_autoextend_threshold and thin_pool_autoextend_percent.)
- The VG does not have enough free blocks to extend the thin pool.

Metadata space exhaustion

If thin pool metadata space is exhausted (or a thin pool metadata operation fails), errors will be returned for IO operations on thin LVs.

When metadata space is exhausted, the lvs command displays 100 under Meta% for the thin pool LV:

```
# lvs −o lv_name,size,data_percent,metadata_percent vg/pool0
  LV    LSize Data%  Meta%
  pool0              100.00
```

The same reasons for thin pool data space exhaustion apply to thin pool metadata space.

Metadata space exhaustion can lead to inconsistent thin pool metadata and inconsistent file systems, so the response requires offline checking and repair.

1. Deactivate the thin pool LV, or reboot the system if this is not possible.

2. Repair thin pool with lvconvert −−repair.
   See "Metadata check and repair".

3. Extend pool metadata space with lvextend VG/ThinPoolLV_tmeta.

See "Manually manage free metadata space of a thin pool LV".

4. Check and repair file system with fsck.

Zoering

When a thin pool provisions a new data block for a thin LV, the new block is first overwritten with zeros. The zeroing mode is indicated by the "z" attribute displayed by lvs. The option −Z (or −−zero) can be added to commands to specify the zeroing mode.

Command to set the zeroing mode when creating a thin pool LV:
lvconvert −−type thin−pool −Z{y|n}
−−poolmetadata VG/ThinMetaLV VG/ThinDataLV

Command to change the zeroing mode of an existing thin pool LV:
lvchange −Z{y|n} VG/ThinPoolLV

If zeroing mode is changed from "n" to "y", previously provisioned blocks are not zeroed.

Provisioning of large zeroed chunks impacts performance.

lvm.conf(5) **thin_pool_zero**
controls the default zeroing mode used when creating a thin pool.

Discard

The discard behavior of a thin pool LV determines how discard requests are handled. Enabling discard under a file system may adversely affect the file system performance (see the section on fstrim for an alternative.) Possible discard behaviors:

ignore: Ignore any discards that are received.

nopassdown: Process any discards in the thin pool itself and allow the no longer needed extends to be overwritten by new data.

passdown: Process discards in the thin pool (as with nopassdown), and pass the discards down the the underlying device. This is the default mode.

Command to display the current discard mode of a thin pool LV:
lvs −o+discards VG/ThinPoolLV

Command to set the discard mode when creating a thin pool LV:
lvconvert −−discards {ignore|nopassdown|passdown}
−−type thin−pool −−poolmetadata VG/ThinMetaLV VG/ThinDataLV

Command to change the discard mode of an existing thin pool LV:
lvchange −−discards {ignore|nopassdown|passdown} VG/ThinPoolLV

Example
lvs −o name,discards vg/pool0

pool0 passdown

lvchange −−discards ignore vg/pool0

lvm.conf(5) **thin_pool_discards**
controls the default discards mode used when creating a thin pool.

Chunk size

The size of data blocks managed by a thin pool can be specified with the −−chunksize option when the thin pool LV is created. The default unit is kilobytes and the default value is 64KiB. The value must be a power of two between 4KiB and 1GiB.

When a thin pool is used primarily for the thin provisioning feature, a larger value is optimal. To optimize for a lot of snapshotting, a smaller value reduces copying time and consumes less space.

Command to display the thin pool LV chunk size:
lvs −o+chunksize VG/ThinPoolLV

Example
lvs −o name,chunksize
 pool0 64.00k

lvm.conf(5) **thin_pool_chunk_size**
controls the default chunk size used when creating a thin pool.

Size of pool metadata LV

The amount of thin metadata depends on how many blocks are shared between thin LVs (i.e. through snapshots). A thin pool with many snapshots may need a larger metadata LV.

The range of supported metadata LV sizes is 2MiB to 16GiB.
The default size is estimated with the formula:
ThinPoolLVSize / ThinPoolLVChunkSize * 64b.

When creating a thin metadata LV explicitly, the size is specified in the lvcreate command. When a command automatically creates a thin metadata LV, the −−poolmetadatasize option can be used specify a non-default size. The default unit is megabytes.

Create a thin snapshot of an external, read only LV

Thin snapshots are typically taken of other thin LVs or other thin snapshot LVs within the same thin pool. It is also possible to take thin snapshots of external, read only LVs. Writes to the snapshot are stored in the thin pool, and the external LV is used to read unwritten parts of the thin snapshot.

lvcreate −n SnapLV −s VG/ExternalOriginLV −−thinpool VG/ThinPoolLV

Example

```
# lvchange −an vg/lve
# lvchange −−permission r vg/lve
# lvcreate −n snaplve −s vg/lve −−thinpool vg/pool0

# lvs vg/lve vg/snaplve
  LV      VG  Attr       LSize  Pool  Origin Data%
  lve     vg  ori------- 10.00g
  snaplve vg  Vwi-a-tz-- 10.00g pool0 lve      0.00
```

Convert a standard LV to a thin LV with an external origin

A new thin LV can be created and given the name of an existing standard LV. At the same time, the existing LV is converted to a read only external LV with a new name. Unwritten portions of the thin LV are read from the external LV. The new name given to the existing LV can be specified with −−originname, otherwise the existing LV will be given a default name, e.g. lvol#.

Convert ExampleLV into a read only external LV with the new name NewExternalOriginLV, and create a new thin LV that is given the previous name of ExampleLV.

lvconvert −−type thin −−thinpool VG/ThinPoolLV
−−originname NewExternalOriginLV −−thin VG/ExampleLV

Example

```
# lvcreate −n lv_example −L 10G vg

# lvs
  lv_example      vg          -wi-a-----  10.00g

# lvconvert −−type thin −−thinpool vg/pool0
        −−originname lv_external −−thin vg/lv_example

# lvs
  LV              VG          Attr       LSize   Pool  Origin
  lv_example      vg          Vwi-a-tz--  10.00g pool0 lv_external
  lv_external     vg          ori-------  10.00g
```

Single step thin pool LV creation

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

lvcreate −−type thin−pool −L LargeSize −n ThinPoolLV VG

Example

```
# lvcreate −−type thin−pool −L8M -n pool0 vg

# lvs vg/pool0
  LV    VG  Attr       LSize Pool Origin Data%
  pool0 vg  twi-a-tz-- 8.00m             0.00
```

```
# lvs −a
  pool0           vg          twi-a-tz--   8.00m
  [pool0_tdata]   vg          Twi-ao----   8.00m
  [pool0_tmeta]   vg          ewi-ao----   8.00m
```

Single step thin pool LV and thin LV creation

A thin pool LV and a thin LV can be created with a single lvcreate command. This one command creates a thin data LV, a thin metadata LV, combines the two into a thin pool LV, and creates a thin LV in the new pool.
−L LargeSize specifies the physical size of the thin pool LV.
−V VirtualSize specifies the virtual size of the thin LV.

lvcreate −V VirtualSize −L LargeSize
−n ThinLV −−thinpool VG/ThinPoolLV

Equivalent to:
lvcreate −−type thin−pool −L LargeSize VG/ThinPoolLV
lvcreate −n ThinLV −V VirtualSize −−thinpool VG/ThinPoolLV

Example

```
# lvcreate −L8M −V2G −n thin1 −−thinpool vg/pool0

# lvs −a
  pool0           vg          twi-a-tz--   8.00m
  [pool0_tdata]   vg          Twi-ao----   8.00m
  [pool0_tmeta]   vg          ewi-ao----   8.00m
  thin1           vg          Vwi-a-tz--   2.00g pool0
```

Merge thin snapshots

A thin snapshot can be merged into its origin thin LV using the lvconvert −−merge command. The result of a snapshot merge is that the origin thin LV takes the content of the snapshot LV, and the snapshot LV is removed. Any content that was unique to the origin thin LV is lost after the merge.

Because a merge changes the content of an LV, it cannot be done while the LVs are open, e.g. mounted. If a merge is initiated while the LVs are open, the effect of the merge is delayed until the origin thin LV is next activated.

lvconvert −−merge VG/SnapLV

Example

```
# lvs vg
  LV      VG Attr       LSize   Pool  Origin
  pool0   vg twi-a-tz--  10.00g
  thin1   vg Vwi-a-tz-- 100.00g pool0
  thin1s1 vg Vwi-a-tz-k 100.00g pool0 thin1

# lvconvert −−merge vg/thin1s1
```

```
# lvs vg
  LV     VG Attr       LSize   Pool  Origin
  pool0   vg twi-a-tz--  10.00g
  thin1   vg Vwi-a-tz-- 100.00g pool0
```

Example

Delayed merging of open LVs.

```
# lvs vg
  LV     VG Attr       LSize   Pool  Origin
  pool0   vg twi-a-tz--  10.00g
  thin1   vg Vwi-aotz-- 100.00g pool0
  thin1s1 vg Vwi-aotz-k 100.00g pool0 thin1

# df
/dev/mapper/vg-thin1            100G   33M  100G   1% /mnt/X
/dev/mapper/vg-thin1s1          100G   33M  100G   1% /mnt/Xs

# ls /mnt/X
file1 file2 file3
# ls /mnt/Xs
file3 file4 file5

# lvconvert −−merge vg/thin1s1
Logical volume vg/thin1s1 contains a filesystem in use.
Delaying merge since snapshot is open.
Merging of thin snapshot thin1s1 will occur on next activation.

# umount /mnt/X
# umount /mnt/Xs

# lvs −a vg
  LV              VG   Attr       LSize   Pool  Origin
  pool0           vg   twi-a-tz--  10.00g
  [pool0_tdata]   vg   Twi-ao----  10.00g
  [pool0_tmeta]   vg   ewi-ao----   1.00g
  thin1           vg   Owi-a-tz-- 100.00g pool0
  [thin1s1]       vg   Swi-a-tz-k 100.00g pool0 thin1

# lvchange −an vg/thin1
# lvchange −ay vg/thin1

# mount /dev/vg/thin1 /mnt/X

# ls /mnt/X
file3 file4 file5
```

XFS on snapshots

Mounting an XFS file system on a new snapshot LV requires attention to the file system's log state and uuid. On the snapshot LV, the xfs log will contain a dummy transaction, and the xfs uuid will match the uuid from the file system on the origin LV.

If the snapshot LV is writable, mounting will recover the log to clear the dummy transaction, but will require skipping the uuid check:

mount /dev/VG/SnapLV /mnt −o nouuid

Or, the uuid can be changed on disk before mounting:

xfs_admin −U generate /dev/VG/SnapLV
mount /dev/VG/SnapLV /mnt

If the snapshot LV is readonly, the log recovery and uuid check need to be skipped while mounting read-only:

mount /dev/VG/SnapLV /mnt −o ro,nouuid,norecovery

SEE ALSO

lvm(8), **lvm.conf**(5), **lvcreate**(8), **lvconvert**(8), **lvchange**(8), **lvextend**(8), **lvremove**(8), **lvs**(8), **thin_dump**(8), **thin_repair**(8) **thin_restore**(8)