Red Hat Enterprise Linux 8

Deduplicating and compressing logical volumes on RHEL

Using VDO to increase LVM storage capacity

Last Updated: 2022-08-30

Red Hat Enterprise Linux 8 Deduplicating and compressing logical volumes on RHEL

Using VDO to increase LVM storage capacity

Abstract

This document explains how to use the Virtual Data Optimizer (VDO) feature in LVM to manage deduplicated and compressed logical volumes on RHEL.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message .

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your feedback on our documentation. Let us know how we can improve it.

Submitting comments on specific passages

1. View the documentation in the **Multi-page HTML** format and ensure that you see the **Feedback** button in the upper right corner after the page fully loads.
2. Use your cursor to highlight the part of the text that you want to comment on.
3. Click the **Add Feedback** button that appears near the highlighted text.
4. Add your feedback and click **Submit**.

Submitting feedback through Bugzilla (account required)

1. Log in to the Bugzilla website.
2. Select the correct version from the **Version** menu.
3. Enter a descriptive title in the **Summary** field.
4. Enter your suggestion for improvement in the **Description** field. Include links to the relevant parts of the documentation.
5. Click **Submit Bug**.

CHAPTER 1. INTRODUCTION TO VDO ON LVM

The Virtual Data Optimizer (VDO) feature provides inline block-level deduplication, compression, and thin provisioning for storage. You can manage VDO as a type of LVM logical volumes (LVs), similar to LVM thinly provisioned volumes.

VDO volumes on LVM (LVM-VDO) are composed of the following LVs:

VDO pool LV

This is the backing physical device that stores, deduplicates, and compresses data for the VDO LV. The VDO pool LV sets the physical size of the VDO volume, which is the amount of data that VDO can store on the disk.
Currently, each VDO pool LV can hold only one VDO LV. As a result, VDO deduplicated and compresses each VDO LV separately. In other words, VDO cannot deduplicate or compress a piece of data that is shared between several VDO LV.

VDO LV

This is the virtual, provisioned device on top of the VDO pool LV. The VDO LV sets the provisioned, logical size of the VDO volume, which is the amount of data that applications can write to the volume before deduplication and compression occurs.

Table 1.1. A comparison of components in VDO on LVM and LVM thin provisioning

	Physical device	Provisioned device
VDO on LVM	VDO pool LV	VDO LV
LVM thin provisioning	Thin pool	Thin volume

Since VDO is thinly provisioned, the file system and applications only see the logical space in use and are not aware of the actual physical space available. Use scripting to monitor the actual available space and generate an alert if use exceeds a threshold: for example, when the VDO pool LV is 80% full.

Additional resources

- Deduplicating and compressing storage
- Creating and managing thinly-provisioned volumes (thin volumes)

CHAPTER 2. LVM-VDO REQUIREMENTS

VDO on LVM has certain requirements on its placement and your system resources.

2.1. VDO MEMORY REQUIREMENTS

Each VDO volume has two distinct memory requirements:

The VDO module

VDO requires a fixed 38 MB of RAM and several variable amounts:

- 1.15 MB of RAM for each 1 MB of configured block map cache size. The block map cache requires a minimum of 150MB RAM.
- 1.6 MB of RAM for each 1 TB of logical space.
- 268 MB of RAM for each 1 TB of physical storage managed by the volume.

The UDS index

The Universal Deduplication Service (UDS) requires a minimum of 250 MB of RAM, which is also the default amount that deduplication uses. You can configure the value when formatting a VDO volume, because the value also affects the amount of storage that the index needs.
The memory required for the UDS index is determined by the index type and the required size of the deduplication window:

Index type	Deduplication window	Note
Dense	1 TB per 1 GB of RAM	A 1 GB dense index is generally sufficient for up to 4 TB of physical storage.
Sparse	10 TB per 1 GB of RAM	A 1 GB sparse index is generally sufficient for up to 40 TB of physical storage.

NOTE

The minimal disk usage for a VDO volume using default settings of 2 GB slab size and 0.25 dense index, requires approx 4.7 GB. This provides slightly less than 2 GB of physical data to write at 0% deduplication or compression.

Here, the minimal disk usage is the sum of the default slab size and dense index.

The UDS Sparse Indexing feature is the recommended mode for VDO. It relies on the temporal locality of data and attempts to retain only the most relevant index entries in memory. With the sparse index, UDS can maintain a deduplication window that is ten times larger than with dense, while using the same amount of memory.

Although the sparse index provides the greatest coverage, the dense index provides more deduplication advice. For most workloads, given the same amount of memory, the difference in deduplication rates between dense and sparse indexes is negligible.

Additional resources

- Examples of VDO requirements by physical size

2.2. VDO STORAGE SPACE REQUIREMENTS

You can configure a VDO volume to use up to 256 TB of physical storage. Only a certain part of the physical storage is usable to store data. This section provides the calculations to determine the usable size of a VDO-managed volume.

VDO requires storage for two types of VDO metadata and for the UDS index:

- The first type of VDO metadata uses approximately 1 MB for each 4 GB of *physical storage* plus an additional 1 MB per slab.
- The second type of VDO metadata consumes approximately 1.25 MB for each 1 GB of *logical storage*, rounded up to the nearest slab.
- The amount of storage required for the UDS index depends on the type of index and the amount of RAM allocated to the index. For each 1 GB of RAM, a dense UDS index uses 17 GB of storage, and a sparse UDS index will use 170 GB of storage.

Additional resources

- Examples of VDO requirements by physical size
- Slab size in VDO

2.3. EXAMPLES OF VDO REQUIREMENTS BY PHYSICAL SIZE

The following tables provide approximate system requirements of VDO based on the physical size of the underlying volume. Each table lists requirements appropriate to the intended deployment, such as primary storage or backup storage.

The exact numbers depend on your configuration of the VDO volume.

Primary storage deployment

In the primary storage case, the UDS index is between 0.01% to 25% the size of the physical size.

Table 2.1. Storage and memory requirements for primary storage

Physical size	RAM usage: UDS	RAM usage: VDO	Disk usage	Index type
10GB–1TB	250MB	472MB	2.5GB	Dense
2–10TB	1GB	3GB	10GB	Dense
	250MB		22GB	Sparse
11–50TB	2GB	14GB	170GB	Sparse
51–100TB	3GB	27GB	255GB	Sparse

Physical size	RAM usage: UDS	RAM usage: VDO	Disk usage	Index type
101–256TB	12GB	69GB	1020GB	Sparse

Backup storage deployment

In the backup storage case, the UDS index covers the size of the backup set but is not bigger than the physical size. If you expect the backup set or the physical size to grow in the future, factor this into the index size.

Table 2.2. Storage and memory requirements for backup storage

Physical size	RAM usage: UDS	RAM usage: VDO	Disk usage	Index type
10GB–1TB	250MB	472MB	2.5 GB	Dense
2–10TB	2GB	3GB	170GB	Sparse
11–50TB	10GB	14GB	850GB	Sparse
51–100TB	20GB	27GB	1700GB	Sparse
101–256TB	26GB	69GB	3400GB	Sparse

2.4. PLACEMENT OF LVM-VDO IN THE STORAGE STACK

You must place certain storage layers under a VDO logical volume and others above it.

You can place thick-provisioned layers on top of VDO, but you cannot rely on the guarantees of thick provisioning in that case. Because the VDO layer is thin-provisioned, the effects of thin provisioning apply to all layers above it. If you do not monitor the VDO volume, you might run out of physical space on thick-provisioned volumes above VDO.

The supported placement of the following layers is under VDO. Do not place them above VDO:

- DM Multipath
- DM Crypt
- Software RAID (LVM or MD RAID)

The following configurations are **not supported**:

- VDO on top of a loopback device
- Encrypted volumes on top of VDO
- Partitions on a VDO volume
- RAID, such as LVM RAID, MD RAID, or any other type, on top of a VDO volume

- Deploying Ceph Storage on LVM-VDO

Additional resources

- Stacking LVM volumes knowledgebase article

CHAPTER 3. CREATING A DEDUPLICATED AND COMPRESSED LOGICAL VOLUME

You can create an LVM logical volume that uses the VDO feature to deduplicate and compress data.

3.1. LVM-VDO DEPLOYMENT SCENARIOS

You can deploy VDO on LVM (LVM-VDO) in a variety of ways to provide deduplicated storage for:

- block access
- file access
- local storage
- remote storage

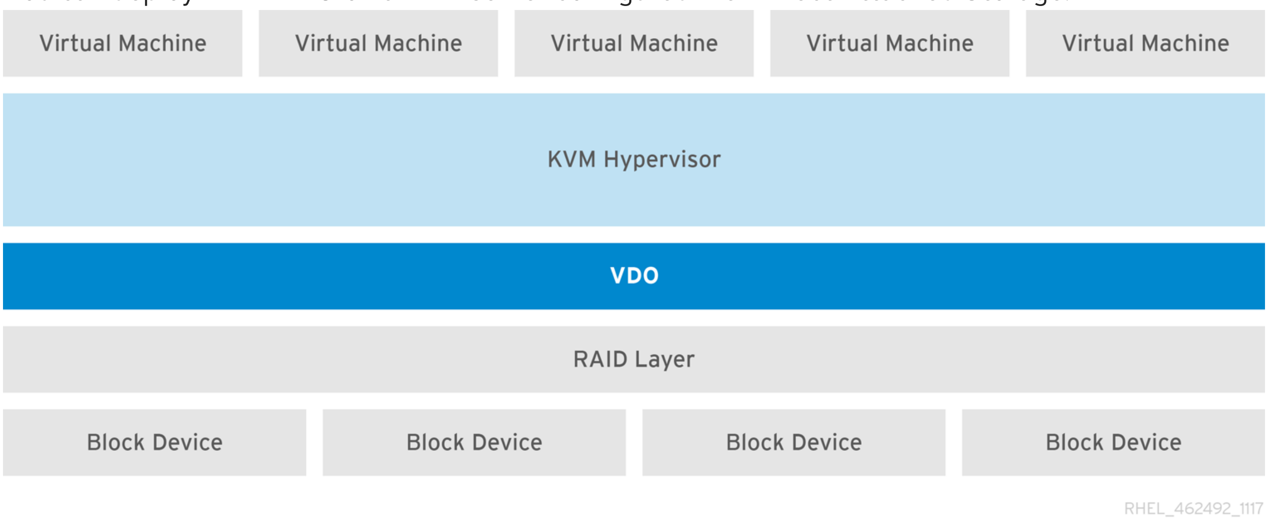
Because LVM-VDO exposes its deduplicated storage as a regular logical volume (LV), you can use it with standard file systems, iSCSI and FC target drivers, or as unified storage.

NOTE

Deploying Ceph Storage on LVM-VDO is currently not supported.

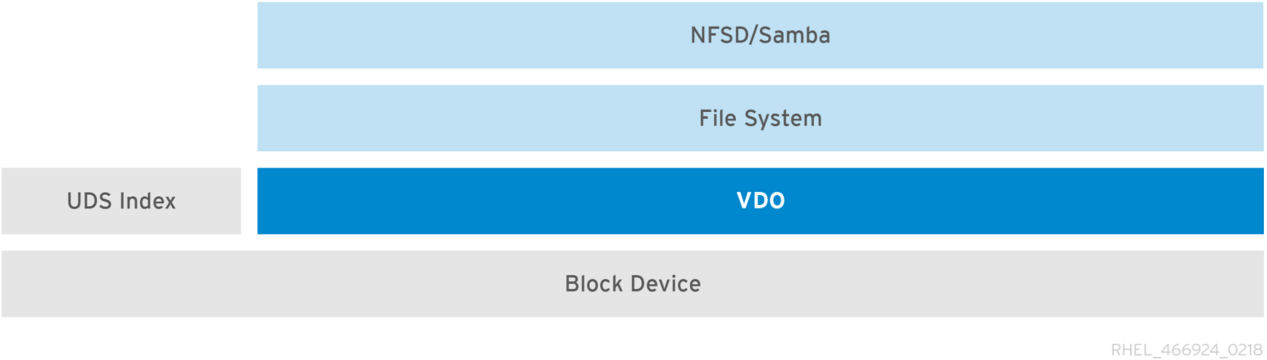
KVM

You can deploy LVM-VDO on a KVM server configured with Direct Attached Storage.



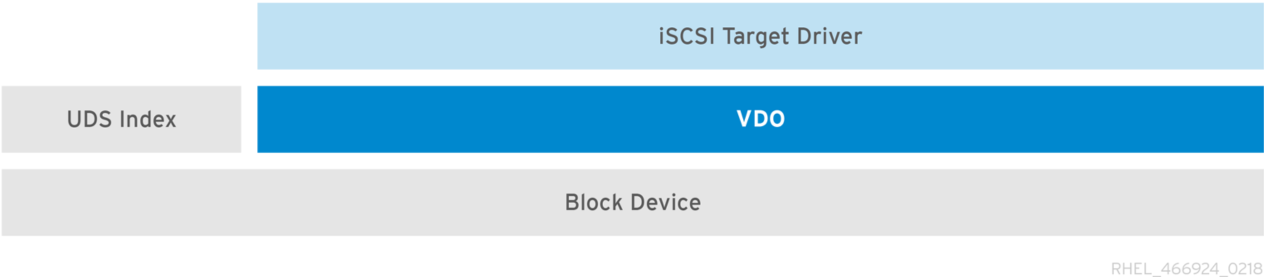
File systems

You can create file systems on top of a VDO LV and expose them to NFS or CIFS users with the NFS server or Samba.



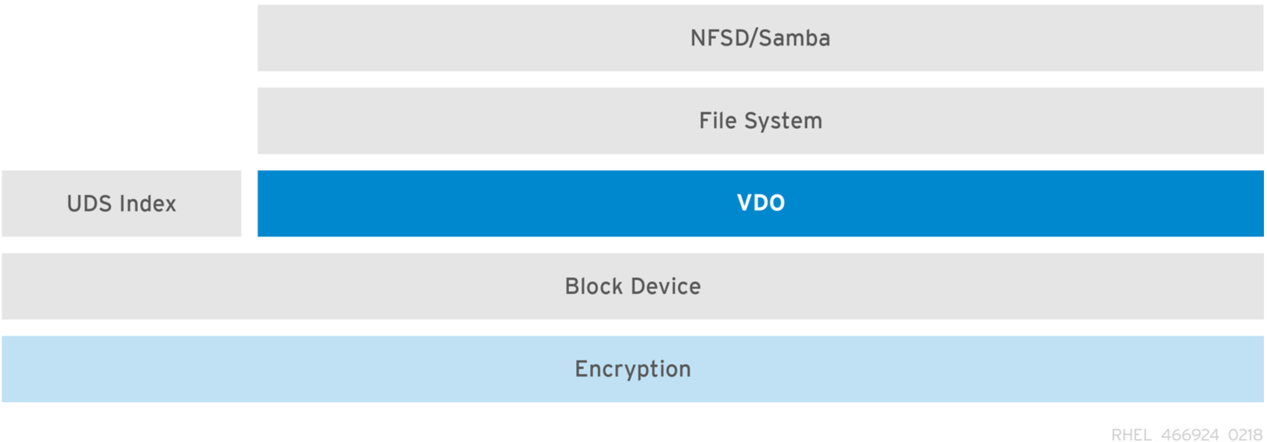
iSCSI target

You can export the entirety of the VDO LV as an iSCSI target to remote iSCSI initiators.



Encryption

Device Mapper (DM) mechanisms such as DM Crypt are compatible with VDO. Encrypting a VDO LV volumes helps ensure data security, and any file systems above the VDO LV are still deduplicated.



IMPORTANT

Applying the encryption layer above the VDO LV results in little if any data deduplication. Encryption makes duplicate blocks different before VDO can deduplicate them.

Always place the encryption layer below the VDO LV.

3.2. THE PHYSICAL AND LOGICAL SIZE OF AN LVM-VDO VOLUME

This section describes the physical size, available physical size, and logical size that VDO can utilize.

Physical size

This is the same size as the physical extents allocated to the VDO pool LV. VDO uses this storage for:

- User data, which might be deduplicated and compressed
- VDO metadata, such as the UDS index

Available physical size

This is the portion of the physical size that VDO is able to use for user data.
It is equivalent to the physical size minus the size of the metadata, rounded down to a multiple of the slab size.

Logical Size

This is the provisioned size that the VDO LV presents to applications. It is usually larger than the available physical size. VDO currently supports any logical size up to 254 times the size of the physical volume with an absolute maximum logical size of 4 PB.
When you set up a VDO logical volume (LV), you specify the amount of logical storage that the VDO LV presents. When hosting active VMs or containers, Red Hat recommends provisioning storage at a 10:1 logical to physical ratio, that is, if you are utilizing 1 TB of physical storage, you would present it as 10 TB of logical storage.

If you do not specify the **--virtualsize** option, VDO provisions the volume to a **1:1** ratio. For example, if you put a VDO LV on top of a 20 GB VDO pool LV, VDO reserves 2.5 GB for the UDS index, if the default index size is used. The remaining 17.5 GB is provided for the VDO metadata and user data. As a result, the available storage to consume is not more than 17.5 GB, and can be less due to metadata that makes up the actual VDO volume.

Additional resources

- Examples of VDO requirements by physical size

3.3. SLAB SIZE IN VDO

The physical storage of the VDO volume is divided into a number of slabs. Each slab is a contiguous region of the physical space. All of the slabs for a given volume have the same size, which can be any power of 2 multiple of 128 MB up to 32 GB.

The default slab size is 2 GB to facilitate evaluating VDO on smaller test systems. A single VDO volume can have up to 8192 slabs. Therefore, in the default configuration with 2 GB slabs, the maximum allowed physical storage is 16 TB. When using 32 GB slabs, the maximum allowed physical storage is 256 TB. VDO always reserves at least one entire slab for metadata, and therefore, the reserved slab cannot be used for storing user data.

Slab size has no effect on the performance of the VDO volume.

Table 3.1. Recommended VDO slab sizes by physical volume size

Physical volume size	Recommended slab size
10–99 GB	1 GB
100 GB – 1 TB	2 GB
2–256 TB	32 GB

NOTE

The minimal disk usage for a VDO volume using default settings of 2 GB slab size and 0.25 dense index, requires approx 4.7 GB. This provides slightly less than 2 GB of physical data to write at 0% deduplication or compression.

Here, the minimal disk usage is the sum of the default slab size and dense index.

You can control the slab size by providing the **--config 'allocation/vdo_slab_size_mb=*size-in-megabytes*'** option to the **lvcreate** command.

3.4. INSTALLING VDO

This procedure installs software necessary to create, mount, and manage VDO volumes.

Procedure

- Install the **vdo** and **kmod-kvdo** packages:

```
# yum install vdo kmod-kvdo
```

3.5. CREATING AN LVM-VDO VOLUME

This procedure creates an VDO logical volume (LV) on a VDO pool LV.

Prerequisites

- Install the VDO software. For more information, see Installing VDO.
- An LVM volume group with free storage capacity exists on your system.

Procedure

1. Pick a name for your VDO LV, such as **vdo1**. You must use a different name and device for each VDO LV on the system.
   In all the following steps, replace *vdo-name* with the name.

2. Create the VDO LV:

```
# lvcreate --type vdo \
        --name vdo-name
        --size physical-size
```

```
        --virtualsize logical-size \
        vg-name
```

- Replace *vg-name* with the name of an existing LVM volume group where you want to place the VDO LV.
- Replace *logical-size* with the amount of logical storage that the VDO LV will present.
- If the physical size is larger than 16TiB, add the following option to increase the slab size on the volume to 32GiB:

  ```
  --config 'allocation/vdo_slab_size_mb=32768'
  ```

  If you use the default slab size of 2GiB on a physical size larger than 16TiB, the **lvcreate** command fails with the following error:

  ```
  ERROR - vdoformat: formatVDO failed on '/dev/device': VDO Status: Exceeds maximum number of slabs supported
  ```

  Example 3.1. Creating a VDO LV for container storage

  For example, to create a VDO LV for container storage on a 1TB VDO pool LV, you can use:

  ```
  # lvcreate --type vdo \
          --name vdo1
          --size 1T
          --virtualsize 10T \
          vg-name
  ```

  IMPORTANT

  If a failure occurs when creating the VDO volume, remove the volume to clean up.

3. Create a file system on the VDO LV:
   - For the XFS file system:

     ```
     # mkfs.xfs -K /dev/vg-name/vdo-name
     ```

   - For the ext4 file system:

     ```
     # mkfs.ext4 -E nodiscard /dev/vg-name/vdo-name
     ```

Additional resources

- **lvmvdo(7)** man page

3.6. MOUNTING AN LVM-VDO VOLUME

This procedure mounts a file system on an LVM-VDO volume, either manually or persistently.

Prerequisites

- An LVM-VDO volume exists on your system. For more information, see Creating an LVM-VDO volume.

Procedure

- To mount the file system on the LVM-VDO volume manually, use:

```
# mount /dev/vg-name/vdo-name mount-point
```

- To configure the file system to mount automatically at boot, add a line to the **/etc/fstab** file:
  - For the XFS file system:

    ```
    /dev/vg-name/vdo-name mount-point xfs defaults 0 0
    ```

  - For the ext4 file system:

    ```
    /dev/vg-name/vdo-name mount-point ext4 defaults 0 0
    ```

  If the LVM-VDO volume is located on a block device that requires network, such as iSCSI, add the **_netdev** mount option. For iSCSI and other block devices requiring network, see the **systemd.mount(5)** man page for information on the **_netdev** mount option.

Additional resources

- **systemd.mount(5)** man page

3.7. CHANGING THE COMPRESSION AND DEDUPLICATION SETTINGS ON AN LVM-VDO VOLUME

This procedure enables or disables the compression and deduplication of a VDO pool logical volume (LV).

> **NOTE**
>
> Compression and deduplication are enabled by default.

Prerequisites

- An LVM-VDO volume exists on your system.

Procedure

1. To find out if the compression and deduplication is enabled or disabled on your logical volumes:

```
# lvs -o+vdo_compression,vdo_deduplication
```

2. Find out status of the compression and status of the deduplication index of your running active VDOPoolLV:

```
# lvs -o+vdo_compression_state,vdo_index_state
```

The **vdo_index_state** can show as **error**, **close**, **opening**, **closing**, **online**, and **offline**.

3. To enable or disable the compression for VDOPoolLV:

```
# lvchange --compression y|n  vg-name/vdopoolname
```

4. To enable or disable the deduplication for VDOPoolLV:

```
# lvchange --deduplication y|n vg-name/vdopoolname
```

Additional resources

- **lvmvdo(7)** man page

3.8. MANAGING THIN PROVISIONING WITH VIRTUAL DATA OPTIMIZER

It is possible to configure a thin-provisioned VDO volume to prepare for future expansion of the physical space, in order to address a condition where the physical space usage of the VDO volume is approaching 100%. Instead of using **-l 100%FREE** in the **lvcreate** operation, for example, use '95%FREE' to ensure that there is some reserved space for recovery later on if needed. This procedure describes how to resolve the following issues:

- The volume runs out of space
- The file system enters read-only mode
- ENOSPC reported by the volume

NOTE

The best way to address high physical space usage on a VDO volume is to delete unused files, and discard the blocks used by these unused files either by using online discard or the **fstrim** program. The physical space of a VDO volume can only be grown to 8192 slabs that is 16 TB for a VDO volume with the default slab size of 2 GB, or 256 TB for a VDO volume with the maximal slab size of 32 GB.

In all of the following steps, replace *myvg* and *myvdo* with the volume group and VDO name respectively.

Prerequisites

1. You have installed the **lvm2**, **vdo**, and **kmod-kvdo** packages.
2. An LVM volume group with free storage capacity exists on your system.

3. A thin-provisioned VDO volume using the **lvcreate --type vdo --name myvdo myvg -L logical-size-of-pool --virtualsize virtual-size-of-vdo** command. For more information, see Creating an LVM-VDO volume.

Procedure

1. Determine the optimal logical size for a thin-provisioned VDO volume

```
# vdostats myvg-vpool0-vpool

Device              1K-blocks Used      Available  Use% Space saving%
myvg-vpool0-vpool   104856576  29664088 75192488   28%   69%
```

To calculate the space savings ratio, use the following formula:

```
Savings ratio = 1 / (1 - Space saving%)
```

In this example,

- there is approximately a **3.22:1** space savings ratio on a data set of about 80 GB.
- Multiplying the data set size by the ratio would yield a potential logical size of 256 GB if more data with the same space savings were written to the VDO volume.
- Adjusting this number downward to 200 GB yields a logical size with a safe margin of free physical space, given the same space savings ratio.

2. Monitor the free physical space in a VDO volume:

```
# vdostats myvg-vpool0-vpool
```

This command can be executed periodically to provide monitoring of the used and free physical space of the VDO volume.

3. Optional: View the warnings on physical space usage on a VDO volume by using the available **/usr/share/doc/vdo/examples/monitor/monitor_check_vdostats_physicalSpace.pl** script:

```
# /usr/share/doc/vdo/examples/monitor/monitor_check_vdostats_physicalSpace.pl myvg-vpool0-vpool
```

4. When creating a VDO volume, the **dmeventd** monitoring service monitors the usage of physical space in a VDO volume. This is enabled by default when a VDO volume is created or started. Use the **journalctl** command to view the output of **dmeventd** in the logs while monitoring a VDO volume:

```
lvm[8331]: Monitoring VDO pool myvg-vpool0-vpool.
...

lvm[8331]: WARNING: VDO pool myvg-vpool0-vpool is now 84.63% full.
lvm[8331]: WARNING: VDO pool myvg-vpool0-vpool is now 91.01% full.
lvm[8331]: WARNING: VDO pool myvg-vpool0-vpool is now 97.34% full.
```

5. Remediate VDO volumes that are almost out of available physical space. When it is possible to add a physical space to a VDO volume, but the volume space is full before it can be grown, it may be necessary to temporarily stop I/O to the volume.
   To temporarily stop I/O to the volume, execute the following steps, where VDO volume *myvdo* contains a file system mounted on the */users/homeDir* path:

   a. Freeze the filesystem:

   ```
   # xfs_freeze -f /users/homeDir

   # vgextend myvg /dev/vdc2

   # lvextend -l new_size myvg/vpool0-name

   # xfs_freeze -u /users/homeDir
   ```

   b. Unmount the filesystem:

   ```
   # umount /users/homeDir

   # vgextend myvg /dev/vdc2

   # lvextend -l new_size myvg/vpool0-name

   # mount -o discard /dev/myvg/myvdo /users/homeDir
   ```

   NOTE

   Unmounting or freezing a filesystem with cached data will incur a write of the cached data, which may fill the physical space of the VDO volume. Consider the maximum amount of cached filesystem data when setting a monitoring threshold for free physical space on a VDO volume.

6. Blocks that are no longer used by a file system can be cleaned up by using the **fstrim** utility. Executing **fstrim** against a mounted file system on top of a VDO volume may result in increased free physical space for that volume. The **fstrim** utility will send discards to the VDO volume, which are then used to remove references to the previously used blocks. If any of those blocks are single-referenced, the physical space will be available to use.

   a. Check VDO stats to see what the current amount of free space is:

   ```
   # vdostats --human-readable myvg-vpool0-vpool

    Device            Size  Used  Available Use%  Space saving%
   myvg-vpool0-vpool  100.0G 95.0G 5.0G      95%   73%
   ```

   b. Discard unused blocks:

   ```
   # fstrim /users/homeDir
   ```

   c. View the free physical space of the VDO volume:

   ```
   # vdostats --human-readable myvg-vpool0-vpool
   ```

```
Device              Size    Used   Available Use%  Space saving%
myvg-vpool0-vpool  100.0G  30.0G  70.0G     30%    43%
```

In this example, after executing **fstrim** on the file system, the discards were able to return 65G of physical space to use in the VDO volume.

NOTE

Discarding volumes with lower levels of deduplication and compression will have a possibility of reclaiming physical space than discarding volumes with higher levels of deduplication and compression. A volume that has high levels of deduplication and compression may potentially require a more extensive cleanup to reclaim physical space than just simply discarding already unused blocks.

CHAPTER 4. TRIM OPTIONS ON AN LVM-VDO VOLUME

You can mount your file system with the **discard** option, which informs the VDO volume of the unused space. Another option is to use the **fstrim** application, which is an on-demand discarding, or **mount -o discard** command for immediate discarding.

When using the **fstrim** application, the admin needs to schedule and monitor an additional process, while using **mount -o discard** command allows for immediate recovery of space when possible.

Note that it is currently recommended to use **fstrim** application to discard unused blocks rather than the **discard** mount option because the performance impact of this option can be quite severe. For this reason, **nodiscard** is the default.

4.1. ENABLING DISCARD MOUNT OPTION ON VDO

This procedure enables the **discard** option on your VDO volume.

Prerequisites

- An LVM-VDO volume exists on your system.

Procedure

- Enable the **discard** on your volume:

```
# mount -o discard /dev/vg-name/vdo-name mount-point
```

Additional resources

- **xfs(5)**, **mount(8)**, and **lvmvdo(7)** man pages

4.2. SETTING UP PERIODIC TRIM OPERATION

This procedure enables a scheduled TRIM operation on your system.

Prerequisites

- An LVM-VDO volume exists on your system.

Procedure

- Enable and start the timer:

```
# systemctl enable --now fstrim.timer
```

Verification

- Verify that the timer is enabled:

```
# systemctl list-timers fstrim.timer
```

Example 4.1. Possible output of the verification procedure

```
# systemctl list-timers fstrim.timer
NEXT                         LEFT         LAST  PASSED  UNIT         ACTIVATES
Mon 2021-05-10 00:00:00 EDT  5 days left  n/a   n/a     fstrim.timer fstrim.service
```

NOTE

You will not see any reference to a VDO volume, because the **fstrim.timer** runs across all mounted filesystems.

Additional resources

- **fstrim(8)** man page