

```
[student@serverb ~]$ sudo -i  
[sudo] password for student: student  
[root@serverb ~]#
```

1.3. 使用 **parted** 创建 512 MiB 的分区，并将其类型设置为 Linux LVM。

```
[root@serverb ~]# parted -s /dev/vdb mkpart primary 514MiB 1026MiB  
[root@serverb ~]# parted -s /dev/vdb set 2 lvm on
```

1.4. 使用 **udevadm settle** 让系统注册新分区。

```
[root@servera ~]# udevadm settle
```

1.5. 使用 **pvcreate** 将该分区初始化为 PV。

```
[root@serverb ~]# pvcreate /dev/vdb2  
Physical volume "/dev/vdb2" successfully created.
```

1.6. 使用新的 **/dev/vdb2** PV，通过 **vgextend** 扩展名为 **servera\_01\_vg** 的 VG。

```
[root@serverb ~]# vgextend servera_01_vg /dev/vdb2  
Volume group "servera_01_vg" successfully extended
```

2. 将 **servera\_01\_lv** 逻辑卷扩展到 768 MiB，包括文件系统。

2.1. 使用 **lvextend** 将 **servera\_01\_lv** LV 扩展到 768 MiB。

```
[root@serverb ~]# lvextend -L 768M /dev/servera_01_vg/servera_01_lv  
Size of logical volume servera_01_vg/servera_01_lv changed from 256.00 MiB (64  
extents) to 768.00 MiB (192 extents).  
Logical volume servera_01_vg/servera_01_lv successfully resized.
```



### 注意

或者，也可以使用 **-L +512M** 选项来调整 LV 的大小。

2.2. 使用 **xfs\_growfs** 将 XFS 文件系统扩展到 LV 上的其余可用空间。

```
[root@serverb ~]# xfs_growfs /storage/data1  
meta-data=/dev/mapper/servera_01_vg-servera_01_lv isize=512 agcount=4  
agsize=16384 blks  
...output omitted...
```



### 注意

此示例显示通过 **xfs\_growfs** 步骤扩展文件系统。另一种方法是向 **lvextend** 命令添加 **-r** 选项。

3. 在现有卷组中，创建名为 `serverb_02_lv` 且大小为 128 MiB 的一个新逻辑卷。添加 XFS 文件系统，并将其永久挂载于 `/storage/data2`。

3.1. 使用 `lvcreate` 从 `serverb_01_vg` VG 创建一个名为 `serverb_02_lv` 的 128 MiB LV。

```
[root@serverb ~]# lvcreate -n serverb_02_lv -L 128M serverb_01_vg
Logical volume "serverb_02_lv" created
```

3.2. 使用 `mkfs` 在 `serverb_02_lv` LV 上放置 `xfs` 文件系统。使用 LV 设备名称。

```
[root@serverb ~]# mkfs -t xfs /dev/serverb_01_vg/serverb_02_lv
meta-data=/dev/serverb_01_vg/serverb_02_lv isize=512    agcount=4, agsize=8192
blks
...output omitted...
```

3.3. 使用 `mkdir` 在 `/storage/data2` 创建挂载点。

```
[root@serverb ~]# mkdir /storage/data2
```

3.4. 在 `serverb` 上，将下面这一行添加到 `/etc/fstab` 的末尾：

```
/dev/serverb_01_vg/serverb_02_lv /storage/data2 xfs defaults 1 2
```

3.5. 使用 `systemctl daemon-reload` 以新的 `/etc/fstab` 配置来更新 `systemd`。

```
[root@servera ~]# systemctl daemon-reload
```

3.6. 使用 `mount` 验证 `/etc/fstab` 条目，并且挂载新的 `serverb_02_lv` LV 设备。

```
[root@serverb ~]# mount /storage/data2
```

4. 完成后，重新启动 `serverb` 计算机，并从 `workstation` 计算机运行命令 `lab lvm-review grade` 以验证工作。

```
[root@serverb ~]# systemctl reboot
```

等待 `serverb` 完全启动，然后继续执行评估。

## 评估

在 `workstation` 上，运行 `lab lvm-review grade` 脚本来确认是否成功完成本练习。

```
[student@workstation ~]$ lab lvm-review grade
```

## 完成

在 `workstation` 上，运行 `lab lvm-review finish` 脚本来完成实验。

```
[student@workstation ~]$ lab lvm-review finish
```

本实验到此结束。

## 总结

---

在本章中，您学到了：

- 通过在多个存储设备上分配空间，LVM 可以让您创建灵活的存储。
- 物理卷、卷组和逻辑卷可由各种工具（如**pvcreate**、**vgreduce** 和 **lvextend**）进行管理。
- 逻辑卷可以格式化为文件系统或交换空间，并可以持久挂载。
- 可以为卷组增加存储空间，也可以动态扩展逻辑卷。

海量视频题库 myitpub.com QQ:5565462  
www.52myit.com

## 章 8

# 实施高级存储功能

### 目标

使用 Stratis 本地存储管理系统管理存储，并使用 VDO 卷优化使用中的存储空间。

### 培训目标

- 使用 Stratis 本地存储管理系统管理多个存储层。
- 使用 VDO 压缩存储设备上的数据并进行重复删除，以此来优化存储空间的使用。

### 章节

- 使用 Stratis 管理分层存储（及引导式练习）
- 使用 VDO 压缩存储和删除重复数据（及引导式练习）

### 实验

实施高级存储功能

# 使用 STRATIS 管理分层存储

## 培训目标

学完本节后，您应能够使用 Stratis 本地存储管理系统管理多个存储层。

## 描述 STRATIS 架构

红帽企业 Linux (RHEL) 当前的本地存储解决方案中包含许多成熟、稳定的技术，其中包括设备映射器 (dm)、逻辑卷管理器 (LVM) 及 XFS 文件系统。这些组件提供的功能包括可大规模扩展的文件系统、快照、冗余 (RAID) 逻辑设备、多路径、精简配置、缓存、重复数据删除，以及对虚拟机和容器的支持。每个存储堆栈层 (dm、LVM 和 XFS) 均使用专门面向层的命令和实用程序进行管理，这就要求系统管理员将物理设备、固定大小的卷和文件系统作为独立的存储组件进行管理。

近年来出现了新一代的存储管理解决方案，称为卷管理文件系统，它可以在创建文件系统及调整其大小时以动态、透明的方式来管理卷层。不过，尽管这些文件系统的社区开发已经持续了很多年，但仍未达到成为红帽企业 Linux 主要本地存储所需的功能支持和稳定性水平。

在 RHEL 8 中，红帽推出了 Stratis 存储管理解决方案。与其他存储项目的尝试一样，Stratis 的开发并不是从零开始，而是使用现有的 RHEL 存储组件。Stratis 以管理物理存储设备池的服务形式运行，并透明地为所创建的文件系统创建和管理卷。由于 Stratis 使用现有的存储驱动程序和工具，因此 Stratis 也支持当前在 LVM、XFS 和设备映射器中使用的所有高级存储功能。

在卷管理文件系统中，文件系统借助一个名为精简配置的概念内置于磁盘设备的共享池中。Stratis 文件系统没有固定大小，也不再预分配未使用的块空间。尽管文件系统仍构建在隐藏的 LVM 卷上，但 Stratis 会为您管理基础卷，并可在需要时对其进行扩展。文件系统的“使用中”大小可视作所含文件占用的实际块数量。文件系统的可用空间就是它所驻留的池设备中仍未使用的空间量。多个文件系统可以驻留在同一磁盘设备池中，共享可用空间，但文件系统也可以保留池空间，以便在需要时保证可用性。

Stratis 使用存储的元数据来识别所管理的池、卷和文件系统。因此，绝不应该对 Stratis 创建的文件系统进行手动重新格式化或重新配置；只应使用 Stratis 工具和命令对它们进行管理。手动配置 Stratis 文件系统可能会导致该元数据丢失，并阻止 Stratis 识别它已创建的文件系统。

您可以使用不同组的块设备来创建多个池。在每个池中，您可以创建一个或多个文件系统。目前，每个池最多可以创建  $2^{24}$  个文件系统。下图说明了 Stratis 存储管理解决方案的元素是如何定位的。

海量视频题库 myitpub.com QQ:5565462  
www.52myit.com

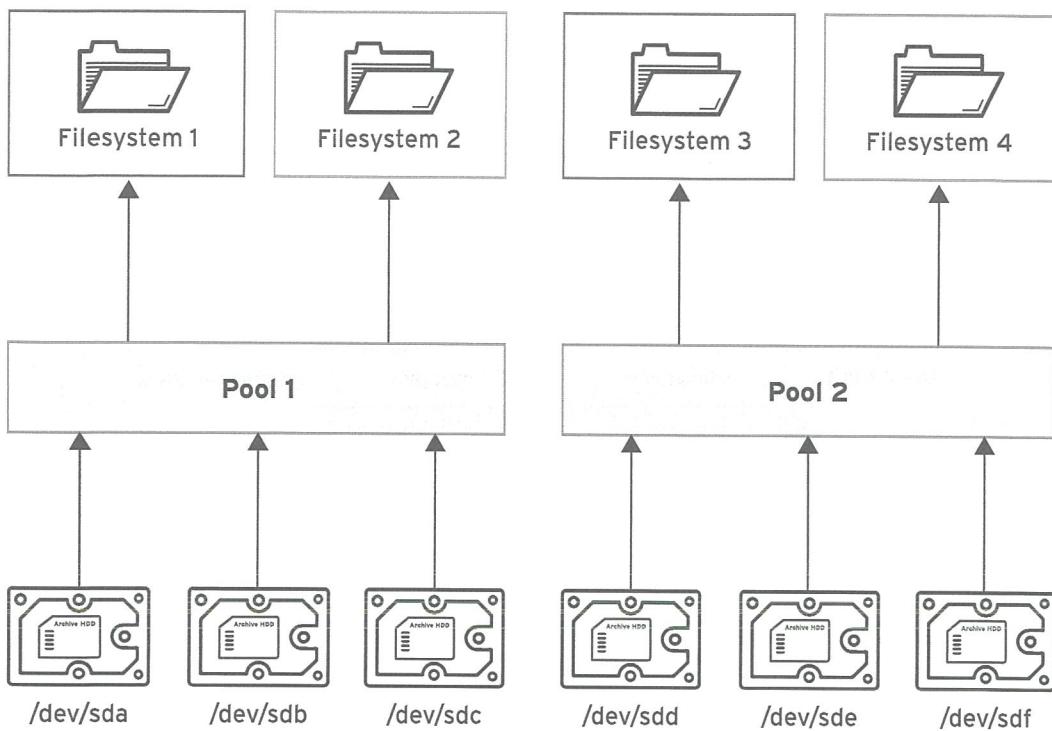


图 8.1: Stratis 的元素

池可将块设备分组到数据层，或分组到缓存层（可选）。数据层侧重于灵活性和完整性，而缓存层则侧重于提高性能。由于缓存层旨在提高性能，因此应使用具有更高每秒输入/输出操作次数 (IOPS) 的块设备，如 SSD。

## 描述简化的存储堆栈

Stratis 简化了各种红帽产品之间本地存储部署和配置的诸多方面。例如，在早期版本的 Anaconda 安装程序中，系统管理员必须对磁盘管理的各个方面进行分层。现在，安装程序使用 Stratis，简化了磁盘设置。其他使用 Stratis 的产品包括 Cockpit、红帽虚拟化和红帽企业 Linux Atomic 主机。对于所有这些产品，Stratis 令存储空间和快照的管理变得更为简单，且更不容易出错。相较于以编程方式使用 CLI，Stratis 更容易与高级管理工具进行集成。

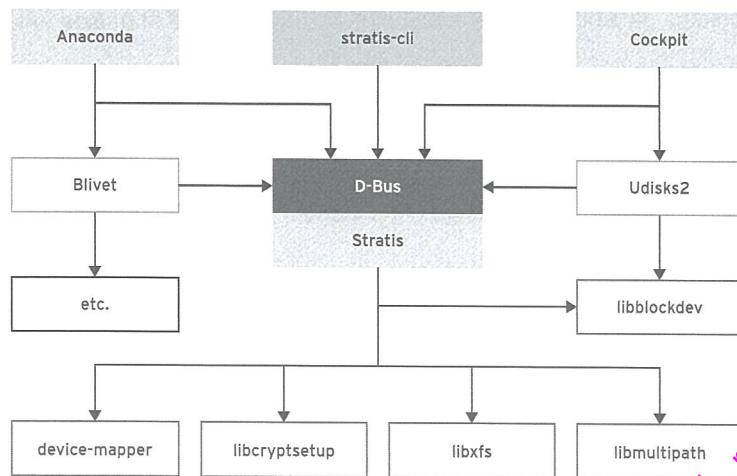


图 8.2: Linux 存储管理堆栈中的 Stratis

## 描述 Stratis 层

在内部，Stratis 使用 **Backstore** 子系统来管理块设备，并使用 **Thinpool** 子系统来管理池。**Backstore** 有一个数据层，负责维护块设备磁盘上的元数据，以及检测和纠正数据损坏。缓存层使用高性能块设备，作为数据层之上的缓存。**Thinpool** 子系统管理与 Stratis 文件系统关联的精简部署卷。该子系统使用 **dm-thin** 设备映射器驱动程序取代 LVM 进行虚拟卷大小调整和管理。**dm-thin** 可以创建虚拟大小比较大、采用 XFS 格式，但物理大小比较小的卷。当物理大小快要满时，Stratis 会自动将其扩大。

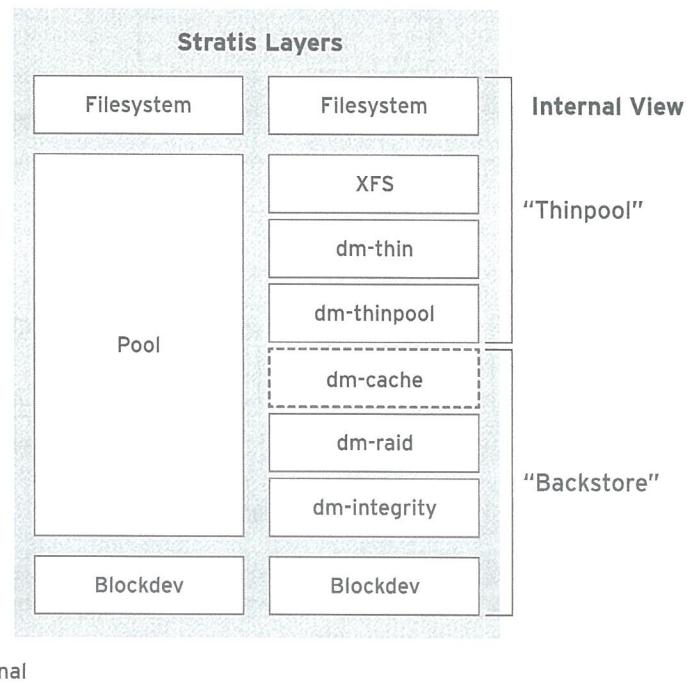


图 8.3: Stratis 层

## 管理精简配置的文件系统

要使用 Stratis 存储管理解决方案来管理精简配置的文件系统，请安装 stratis-cli 和 stratisd 软件包。stratis-cli 软件包中提供了 **stratis** 命令，它通过 D-Bus API 将用户请求转换为 **stratisd** 服务。stratisd 软件包中提供了 **stratisd** 服务，它实现 D-Bus 接口并管理和监控 Stratis 的元素，如块设备、池和文件系统。当 **stratisd** 服务处于运行状态时，可以使用 D-Bus API。

使用常用工具安装并激活 Stratis：

- 使用 **yum install** 命令安装 stratis-cli 和 stratisd。

```
[root@host ~]# yum install stratis-cli stratisd
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

- 使用 **systemctl** 命令激活 stratisd 服务。

```
[root@host ~]# systemctl enable --now stratisd
```

以下是使用 Stratis 存储管理解决方案执行的常见管理操作。

- 使用 **stratis pool create** 命令来创建包含一个或多个块设备的池。

```
[root@host ~]# stratis pool create pool1 /dev/vdb
```

每个池都是 **/stratis** 目录下的一个子目录。

- 使用 **stratis pool list** 命令查看可用池的列表。

```
[root@host ~]# stratis pool list
Name      Total Physical Size  Total Physical Used
pool1        5 GiB             52 MiB
```

- 使用 **stratis pool add-data** 命令向池中添加额外的块设备。

```
[root@host ~]# stratis pool add-data pool1 /dev/vdc
```

- 使用 **stratis blockdev list** 命令查看池的块设备。

```
[root@host ~]# stratis blockdev list pool1
Pool Name  Device Node    Physical Size   State   Tier
pool1       /dev/vdb          5 GiB     In-use   Data
pool1       /dev/vdc          5 GiB     In-use   Data
```

- 使用 **stratis filesystem create** 命令为池创建动态、灵活的文件系统。

```
[root@host ~]# stratis filesystem create pool1 filesystem1
```

Stratis 文件系统的链接位于 **/stratis/pool1** 目录中。

- Stratis 支持通过 **stratis filesystem snapshot** 命令创建文件系统快照。快照独立于源文件系统。

```
[root@host ~]# stratis filesystem snapshot pool1 filesystem1 snapshot1
```

- 使用 **stratis filesystem list** 命令查看可用文件系统的列表。

```
[root@host ~]# stratis filesystem list
...output omitted...
```

为了确保持久挂载 Stratis 文件系统, 请编辑 **/etc/fstab** 并指定文件系统的详细信息。以下命令显示文件系统的 UUID, 在 **/etc/fstab** 中应使用该 UUID 来识别文件系统。

```
[root@host ~]# lsblk --output=UUID /stratis/pool1/filesystem1
UUID
31b9363b-add8-4b46-a4bf-c199cd478c55
```

以下是 **/etc/fstab** 文件中用于持久挂载 Stratis 文件系统的条目示例。

```
UUID=31b9...8c55 /dir1 xfs defaults,x-systemd.requires=stratisd.service 0 0
```

**x-systemd.requires=stratisd.service** 挂载选项可延迟挂载文件系统，直到 **systemd** 在启动过程中启动 **stratisd.service** 为止。



### 注意

如果在 **/etc/fstab** 中未对 Stratis 文件系统使用 **x-systemd.requires=stratisd.service** 挂载选项，将会导致计算机在下一次重启时引导至 **emergency.target**。



### 参考文献

如需更多信息，请参阅《红帽企业 Linux 8 配置和管理文件系统指南》中的使用 **Straits** 管理分层本地存储一章，网址为：

[https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/8/html-single/configuring\\_and\\_managing\\_file\\_systems/](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_and_managing_file_systems/)

### Stratis 存储

<https://stratis-storage.github.io/>

### Stratis 从 ZFS、Btrfs 和 Linux 卷管理器借鉴的内容

<https://opensource.com/article/18/4/stratis-lessons-learned>

## ► 指导练习

# 使用 STRATIS 管理分层存储

在本练习中，您将使用 Stratis 存储管理解决方案来创建协同工作的池、卷和文件系统。

## 成果

您应能够：

- 使用 Stratis 存储管理解决方案来创建精简配置的文件系统。
- 验证 Stratis 卷是否会动态增长，以支持实时数据增加。
- 通过精简配置的文件系统的快照访问数据。

## 在你开始之前

以 `student` 身份并使用 `student` 作为密码登录 `workstation`。

在 `workstation` 上，运行 `lab advstorage-stratis start` 开始本练习。此脚本可正确设置环境并确保将 `servera` 上的其他磁盘清理干净。

```
[student@workstation ~]$ lab advstorage-stratis start
```

- 1. 从 `workstation`，以 `student` 用户身份打开连接 `servera` 的 SSH 会话。

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- 2. 切换到 `root` 用户。

```
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 3. 使用 `yum` 安装 `stratisd` 和 `stratis-cli` 软件包。

```
[root@servera ~]# yum install stratisd stratis-cli
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

- 4. 使用 `systemctl` 命令激活 `stratisd` 服务。

```
[root@servera ~]# systemctl enable --now stratisd
```

► 5. 确保 stratispool1 Stratis 池与块设备 /dev/vdb 共存。

- 5.1. 使用 **stratis pool create** 命令创建一个名为 stratispool1 的 Stratis 池。

```
[root@servera ~]# stratis pool create stratispool1 /dev/vdb
```

- 5.2. 使用 **stratis pool list** 命令验证 stratispool1 的可用性。

[root@servera ~]# stratis pool list			
Name	Total Physical Size	Total Physical Used	
stratispool1	5 GiB	52 MiB	

请注意上述输出中池的大小。

► 6. 使用 /dev/vdc 块设备扩展 stratispool1 的容量。

- 6.1. 使用 **stratis pool add-data** 命令将块设备 /dev/vdc 添加到 stratispool1 中。

```
[root@servera ~]# stratis pool add-data stratispool1 /dev/vdc
```

- 6.2. 使用 **stratis pool list** 命令验证 stratispool1 的大小。

[root@servera ~]# stratis pool list			
Name	Total Physical Size	Total Physical Used	
stratispool1	10 GiB	56 MiB	

如上所示，在添加块设备时，stratispool1 的池大小会相应增加。

- 6.3. 使用 **stratis blockdev list** 命令，验证当前为 stratispool1 成员的块设备。

[root@servera ~]# stratis blockdev list stratispool1					
Pool Name	Device Node	Physical Size	State	Tier	
stratispool1	/dev/vdb	5 GiB	In-use	Data	
stratispool1	/dev/vdc	5 GiB	In-use	Data	

► 7. 在池 stratispool1 中添加一个精简配置的文件系统 stratis-filesystem1。在 stratisvol 上挂载文件系统在 stratis-filesystem1 文件系统上创建一个名为 file1 的文件，其中包含文本 Hello World!。

- 7.1. 使用 **stratis filesystem create** 命令在 stratispool1 上创建精简配置的文件系统 stratis-filesystem1。此命令最多可能需要一分钟完成。

```
[root@servera ~]# stratis filesystem create stratispool1 stratis-filesystem1
```

- 7.2. 使用 **stratis filesystem list** 命令验证 stratis-filesystem1 的可用性。

```
[root@servera ~]# stratis filesystem list
Pool Name      Name          Used     Created        Device
                  UUID
stratispool1  stratis-filesystem1  546 MiB  Mar 29 2019 07:48  /stratis/
stratispool1/stratis-filesystem1  8714...e7db
```

请注意 **stratis-filesystem1** 的当前使用情况。在以下步骤中，文件系统的使用会按需增加。

7.3. 使用 **mkdir** 命令创建名为 **/stratisvol** 的目录。

```
[root@servera ~]# mkdir /stratisvol
```

7.4. 使用 **mount** 命令将 **stratis-filesystem1** 挂载于 **/stratisvol**。

```
[root@servera ~]# mount /stratis/stratispool1/stratis-filesystem1 /stratisvol
```

7.5. 使用 **mount** 命令，验证 Stratis 文件系统 **stratis-filesystem1** 是否挂载于 **/stratisvol**。

```
[root@servera ~]# mount
...output omitted...
/dev/mapper/stratis-1-5c0e...12b9-thin-fs-8714...e7db on /stratisvol type xfs
(rw,relatime,seclabel,attr2,inode64,sunit=2048,swidth=2048,noquota)
```

7.6. 使用 **echo** 命令创建文本文件 **/stratisvol/file1**。

```
[root@servera ~]# echo "Hello World!" > /stratisvol/file1
```

► 8. 验证精简配置的文件系统 **stratis-filesystem1** 是否随着文件系统上数据的增加而动态增长。

8.1. 使用 **stratis filesystem list** 命令查看 **stratis-filesystem1** 的当前使用情况。

```
[root@servera ~]# stratis filesystem list
Pool Name      Name          Used     Created        Device
                  UUID
stratispool1  stratis-filesystem1  546 MiB  Mar 29 2019 07:48  /stratis/
stratispool1/stratis-filesystem1  8714...e7db
```

8.2. 使用 **dd** 命令在 **stratis-filesystem1** 上创建一个 2 GiB 文件。此命令最多可能需要一分钟完成。

```
[root@servera ~]# dd if=/dev/urandom of=/stratisvol/file2 bs=1M count=2048
```

8.3. 使用 **stratis filesystem list** 命令验证 **stratis-filesystem1** 的使用情况。

```
[root@servera ~]# stratis filesystem list
Pool Name      Name          Used      Created      Device
                           UUID
stratispool1   stratis-filesystem1  2.53 GiB  Mar 29 2019 07:48  /stratis/
stratispool1/stratis-filesystem1  8714...e7db
```

上述输出显示 **stratis-filesystem1** 的使用量有所增加。使用量的增加确认了精简配置的文件系统已发生动态扩展，以适应因创建 **/stratisvol/file2** 而带来的实时数据增长。

- 9. 创建 **stratis-filesystem1** 的快照 **stratis-filesystem1-snap**。此快照将允许您访问从 **stratis-filesystem1** 中删除的任何文件。

9.1. 使用 **stratis filesystem snapshot** 命令创建 **stratis-filesystem1** 的一个快照。此命令最多可能需要一分钟完成。

```
[root@servera ~]# stratis filesystem snapshot stratispool1 \
stratis-filesystem1 stratis-filesystem1-snap
```

9.2. 使用 **stratis filesystem list** 命令验证快照的可用性。

```
[root@servera ~]# stratis filesystem list
...output omitted...
stratispool1  stratis-filesystem1-snap  2.53 GiB  Mar 29 2019 10:28  /stratis/
stratispool1/stratis-filesystem1-snap  291d...8a16
```

9.3. 删除 **/stratisvol/file1** 文件。

```
[root@servera ~]# rm /stratisvol/file1
rm: remove regular file '/stratisvol/file1'? y
```

9.4. 使用 **mkdir** 命令创建 **/stratisvol-snap** 目录。

```
[root@servera ~]# mkdir /stratisvol-snap
```

9.5. 使用 **mount** 命令将 **stratis-filesystem1-snap** 的快照挂载于 **/stratisvol-snap**。

```
[root@servera ~]# mount /stratis/stratispool1/stratis-filesystem1-snap \
/stratisvol-snap
```

9.6. 确认是否仍能使用 **stratis-filesystem1-snap** 快照来访问从 **stratis-filesystem1** 中删除的文件。

```
[root@servera ~]# cat /stratisvol-snap/file1
Hello World!
```

- 10. 使用 **umount** 命令卸载 **/stratisvol** 和 **/stratisvol-snap**。

```
[root@servera ~]# umount /stratisvol-snap  
[root@servera ~]# umount /stratisvol
```

- 11. 从系统中删除精简配置的文件系统 stratis-filesystem1 及其快照 stratis-filesystem1-snap。

11.1. 使用 **stratis filesystem destroy** 命令销毁 stratis-filesystem1-snap。

```
[root@servera ~]# stratis filesystem destroy stratispool1 stratis-filesystem1-snap
```

11.2. 使用 **stratis filesystem destroy** 命令销毁 stratis-filesystem1。

```
[root@servera ~]# stratis filesystem destroy stratispool1 stratis-filesystem1
```

11.3. 退出 root 用户的 shell。

```
[root@servera ~]# exit  
logout  
[student@servera ~]$
```

11.4. 从 servera 注销。

```
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$
```

## 完成

在 workstation 上，运行 **lab advstorage-stratis finish** 来完成本练习。此脚本将删除在实验过程中创建的分区和文件，并确保环境清理干净。

```
[student@workstation ~]$ lab advstorage-stratis finish
```

本引导式练习到此结束。

# 使用 VDO 压缩存储和删除重复数据

## 培训目标

学完本节后，您应能够使用 VDO 压缩存储设备上的数据并进行重复删除，以此来优化存储空间的使用。

## 描述虚拟数据优化器

红帽企业 Linux 8 包含虚拟数据优化器 (VDO) 驱动程序，可以优化块设备上数据的空间占用。VDO 是一个 Linux 设备映射器驱动程序，它可以减少块设备上的磁盘空间使用，同时最大限度减少数据重复，从而节省磁盘空间，甚至提高数据吞吐量。VDO 包括两个内核模块：**kvdo** 模块用于以透明的方式控制数据压缩，**uds** 则可用于重复数据删除。

VDO 层位于现有块存储设备（如 RAID 设备或本地磁盘）的顶部。这些块设备也可以是加密设备。存储层（如 LVM 逻辑卷和文件系统）位于 VDO 设备之上。下图显示了在一个由使用优化存储设备的 KVM 虚拟机构成的基础架构中，VDO 所处的位置。

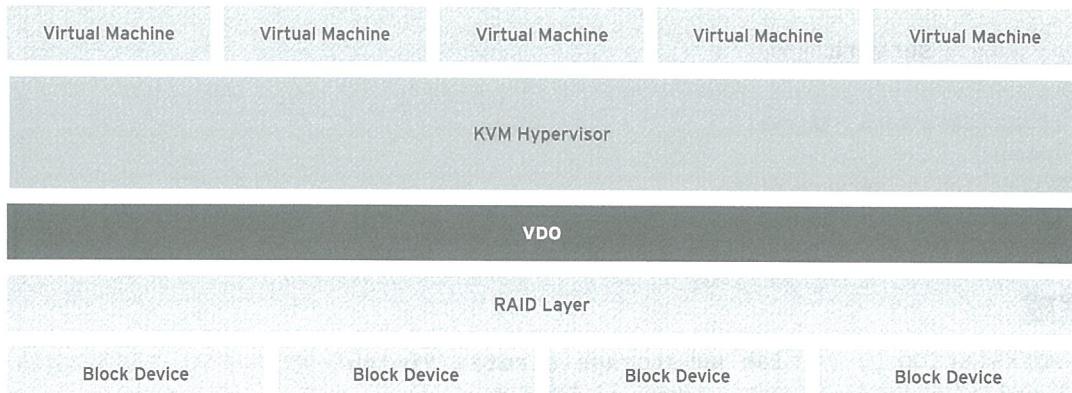


图 8.4: 基于 VDO 的虚拟机

VDO 会按以下顺序对数据实施三个阶段的处理，以减少存储设备上的空间占用：

1. 零块消除将过滤掉仅包含零 (0) 的数据块，且仅在元数据中记录这些块的信息。非零数据块随即被传递到下一个处理阶段。该阶段将启用 VDO 设备中的精简配置功能。
2. 重复数据删除将去除冗余的数据块。在创建相同数据的多个副本时，VDO 会检测重复数据块并更新元数据，以便使用这些重复块来引用原始数据块，而不会创建冗余数据块。通用重复数据删除服务 (UDS) 内核模块将通过其维护的元数据来检查数据的冗余。该内核模块是作为 VDO 的一部分而提供的。
3. 最后一个阶段是压缩。**kvdo** 内核模块使用 LZ4 压缩对块进行压缩，并以 4 KB 块进行分组。

## 实施虚拟数据优化器

利用 VDO 创建的逻辑设备被称为 VDO 卷。VDO 卷与磁盘分区类似；您可以将这些卷格式化为所需的文件系统类型，并像常规文件系统那样进行挂载。此外，您还可以将 VDO 卷用作 LVM 物理卷。

要创建 VDO 卷，请指定块设备以及 VDO 向用户显示的逻辑设备的名称。您可以指定 VDO 卷的逻辑大小（可选）。VDO 卷的逻辑大小可以大于实际块设备的物理大小。

由于 VDO 卷采用了精简配置，因此用户只能看到正在使用的逻辑空间，而无法了解实际可用的物理空间。如果在创建卷时未指定逻辑大小，则 VDO 会将实际物理大小视为卷的逻辑大小。这种采用 1:1 的比率映射逻辑大小与物理大小的方式有利于提高性能，但同时也会降低存储空间的使用效率。应根据您的基础架构要求来确定是优先考虑性能还是空间效率。

当 VDO 卷的逻辑大小超过实际物理大小时，应使用 **vdostats --verbose** 命令主动监控卷统计信息，以查看实际使用情况。

## 启用 VDO

安装 VDO 和 kmod-kvdo 软件包，以便在系统中启用 VDO。

```
[root@host ~]# yum install vdo kmod-kvdo
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

## 创建 VDO 卷

要创建 VDO 卷，请运行 **vdo create** 命令。

```
[root@host ~]# vdo create --name=vdo1 --device=/dev/vdd --vdoLogicalSize=50G
...output omitted...
```

如果省略逻辑大小，则生成的 VDO 卷将与其物理设备的大小相同。

当 VDO 卷就位时，您可以将它格式化为所选的文件系统类型并挂载于系统的文件系统层次结构下。

## 分析 VDO 卷

要分析 VDO 卷，请运行 **vdo status** 命令。此命令将以 YAML 格式显示有关 VDO 系统的报告以及 VDO 卷的状态。此外，它还显示 VDO 卷的属性。使用 **--name=** 选项可指定特定卷的名称。如果省略特定卷的名称，**vdo status** 命令的输出中将显示所有 VDO 卷的状态。

```
[root@host ~]# vdo status --name=vdo1
...output omitted...
```

**vdo list** 命令显示当前启动的 VDO 卷的列表。您可以分别使用 **vdo start** 和 **vdo stop** 命令来启动和停止 VDO 卷。



### 参考文献

如需更多信息，请参阅《红帽企业 Linux 8 重复数据删除和存储压缩指南》中的 VDO 入门章节，网址为：

[https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/8/html-single/deduplicating\\_and\\_compressing\\_storage/](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/deduplicating_and_compressing_storage/)

### 虚拟数据优化器简介

<https://rhelblog.redhat.com/2018/04/11/introducing-virtual-data-optimizer-to-reduce-cloud-and-on-premise-storage-costs/>

## ► 指导练习

# 使用 VDO 压缩存储和删除重复数据

在本练习中，您将创建一个 VDO 卷，将其格式化为文件系统，进行挂载，在上面存储数据，以及调查压缩和重复数据删除对实际使用的存储空间的影响。

## 成果

您应能够：

- 使用虚拟数据优化器创建卷，将其格式化为相应的文件系统类型，并在上面挂载文件系统。
- 调查删除重复数据和压缩对虚拟数据优化器卷的影响。

## 在你开始之前

以 student 身份并使用 student 作为密码登录 workstation。

在 workstation 上，运行 **lab advstorage-vdo start** 开始本练习。此脚本将确保 /dev/vdd 磁盘上没有分区并正确设置环境。

```
[student@workstation ~]$ lab advstorage-vdo start
```

- 1. 从 workstation，以 student 用户身份打开连接 servera 的 SSH 会话。

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$
```

- 2. 使用 /dev/vdd 设备创建 VDO 卷 vdo1。将其逻辑大小设置为 50 GB。

2.1. 切换到 root 用户。

```
[student@servera ~]$ sudo -i  
[sudo] password for student: student  
[root@servera ~]#
```

2.2. 使用 rpm 命令，确认已安装 vdo 软件包。

```
[root@servera ~]# yum list installed vdo  
vdo-6.2.0.293-10.el8.x86_64
```

2.3. 使用 vdo create 命令创建 vdo1 卷。

```
[root@servera ~]# vdo create --name=vdo1 --device=/dev/vdd --vdoLogicalSize=50G  
...output omitted...
```

2.4. 使用 **vdo list** 命令验证 vdo1 卷的可用性。

```
[root@servera ~]# vdo list  
vdo1
```

► 3. 验证 vdo1 卷上是否启用了压缩和重复数据删除功能。

- 3.1. 使用 **grep**, 在 **vdo status --name=vdo1** 命令的输出中搜索包含字符串 **Deduplication** 的行。

```
[root@servera ~]# vdo status --name=vdo1 | grep Deduplication  
Deduplication: enabled
```

- 3.2. 使用 **grep**, 在 **vdo status --name=vdo1** 命令的输出中搜索包含字符串 **Compression** 的行。

```
[root@servera ~]# vdo status --name=vdo1 | grep Compression  
Compression: enabled
```

► 4. 将 vdo1 卷格式化为 XFS 文件系统类型并挂载于 **/mnt/vdo1**。

- 4.1. 使用 **mkfs** 命令将 vdo1 卷格式化为 XFS 文件系统。

```
[root@servera ~]# mkfs.xfs -K /dev/mapper/vdo1  
...output omitted...
```

上述 **mkfs.xfs** 命令中的 **-K** 选项可以防止文件系统中未使用的块被立即丢弃, 这样可以让命令更快返回。

- 4.2. 使用 **udevadm** 命令注册新设备节点。

```
[root@servera ~]# udevadm settle
```

- 4.3. 使用 **mkdir** 命令创建 **/mnt/vdo1** 目录。

```
[root@servera ~]# mkdir /mnt/vdo1
```

- 4.4. 使用 **mount** 命令将 vdo1 卷挂载于 **/mnt/vdo1**。

```
[root@servera ~]# mount /dev/mapper/vdo1 /mnt/vdo1
```

- 4.5. 使用 **mount** 命令, 验证 vdo1 卷是否挂载成功。

```
[root@servera ~]# mount  
...output omitted...  
/dev/mapper/vdo1 on /mnt/vdo1 type xfs  
(rw,relatime,seclabel,attr2,inode64,noquota)
```

- 5. 在 vdo1 卷上创建同一文件 /root/install.img 的三个副本。比较卷的统计信息，验证卷上是否发生数据重复删除和压缩的情况。上述输出可能在您的系统上有所不同。

5.1. 使用 **vdostats** 命令查看卷的初始统计信息和状态。

```
[root@servera ~]# vdostats --human-readable
Device          Size     Used   Available  Use% Space saving%
/dev/mapper/vdo1    5.0G    3.0G      2.0G   60%        99%
```

请注意，卷中有 3 GB 的空间已被占用，因为 VDO 卷在创建时会为自己保留 3-4 GB。此外还要注意，**Space saving%** 字段中的值 **99%** 表示您尚未在卷中创建任何内容，构成了所有节省的卷空间。

5.2. 将 /root/install.img 复制到 /mnt/vdo1/install.img.1 并验证卷统计信息。  
复制文件最多可能需要一分钟。

```
[root@servera ~]# cp /root/install.img /mnt/vdo1/install.img.1
[root@servera ~]# vdostats --human-readable
Device          Size     Used   Available  Use% Space saving%
/dev/mapper/vdo1    5.0G    3.4G      1.6G   68%        5%
```

请注意，**Used** 字段的值从 **3.0G** 增加至 **3.4G**，因为您向卷中复制了一个文件，这会占用一些空间。此外还要注意，**Space saving%** 字段的值从 **99%** 减少至 **5%**，因为最初卷中并没有内容，由此导致在创建文件前，卷空间利用率较低，而卷空间节省率较高。现在，由于您在卷中创建了该文件的唯一副本，里面没有要删除的重复数据，因此卷空间的节省率非常低。

5.3. 将 /root/install.img 复制到 /mnt/vdo1/install.img.2 并验证卷统计信息。  
复制文件最多可能需要一分钟。

```
[root@servera ~]# cp /root/install.img /mnt/vdo1/install.img.2
[root@servera ~]# vdostats --human-readable
Device          Size     Used   Available  Use% Space saving%
/dev/mapper/vdo1    5.0G    3.4G      1.6G   68%        51%
```

请注意，已用的卷空间并没有改变，而节省的卷空间百分比还有所增加，这说明发生了数据重复删除以减少同一文件的冗余副本所占用的空间。上述输出中的 **Space saving%** 值可能在您的系统上有所不同。

5.4. 退出 root 用户的 shell。

```
[root@servera ~]# exit
logout
[student@servera ~]$
```

5.5. 从 servera 注销。

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

## 完成

在 workstation 上，运行 **lab advstorage-vdo finish** 来完成本练习。此脚本将删除在实验过程中创建的文件，并确保环境清理干净。

```
[student@workstation ~]$ lab advstorage-vdo finish
```

本引导式练习到此结束。

## ► 开放研究实验

# 实施高级存储功能

在本练习中，您将使用 Stratis 存储管理解决方案来创建可根据数据需求的增加而增长的文件系统，并使用虚拟数据优化器创建卷，以高效利用存储空间。

## 成果

您应能够：

- 使用 Stratis 存储管理解决方案来创建精简配置的文件系统。
- 验证 Stratis 卷是否会动态增长，以支持实时数据增加。
- 通过精简配置的文件系统的快照访问数据。
- 使用虚拟数据优化器创建卷并将其挂载到文件系统上。
- 调查删除重复数据和压缩对虚拟数据优化器卷的影响。

## 在你开始之前

以 `student` 身份并使用 `student` 作为密码登录 `workstation`。

在 `workstation` 上，运行 `lab advstorage-review start` 以开始实验。此脚本可正确设置环境并确保将 `serverb` 上的其他磁盘清理干净。

```
[student@workstation ~]$ lab advstorage-review start
```

1. 从 `workstation`，以 `student` 用户身份打开连接 `serverb` 的 SSH 会话。
2. 切换到 `root` 用户。
3. 使用 `yum` 安装 `stratisd` 和 `stratis-cli` 软件包。
4. 使用 `systemctl` 命令启动并启用 `stratisd` 服务。
5. 创建包含块设备 `/dev/vdb` 的 Stratis 池 `labpool`。
6. 利用系统中可用的磁盘 `/dev/vdc` 扩展 `labpool` 的容量。
7. 在 `labpool` 池中创建精简配置的文件系统 `labfs`。将此文件系统挂载于 `/labstratisvol`，以实现在重启后持久保留。在 `labfs` 文件系统上创建一个名为 `labfile1` 的文件，其中包含文本 `Hello World!`。别忘了在 `/etc/fstab` 中使用 `x-systemd.requires=stratisd.service` 挂载选项。
8. 验证精简配置的文件系统 `labfs` 是否随着文件系统上数据的增加而动态增长。
9. 创建 `labfs` 文件系统的快照 `labfs-snap`。此快照允许您访问从 `labfs` 中删除的任何文件。
10. 使用设备 `/dev/vdd` 创建 VDO 卷 `labvdo`。将其逻辑大小设置为 `50 GB`。
11. 采用 XFS 文件系统将卷 `labvdo` 挂载于 `/labvdovol`，以实现在重启后持久保留。别忘了在 `/etc/fstab` 中使用 `x-systemd.requires=vdo.service` 挂载选项。

12. 在 labvdo 卷上创建文件 **/root/install.img** 的三个副本。比较卷的统计信息，验证卷上是否发生数据重复删除和压缩的情况。

## 评估

在 workstation 上，运行 **lab advstorage-review grade** 命令以确认本练习是否成功。

```
[student@workstation ~]$ lab advstorage-review grade
```

## 完成

在 workstation 上，运行 **lab advstorage-review finish** 来完成本练习。此脚本将删除在实验过程中创建的分区和文件，并确保环境清理干净。

```
[student@workstation ~]$ lab advstorage-review finish
```

本实验到此结束。