Red Hat Enterprise Linux 8

配置设备映射器多路径

配置和管理设备映射器多路径功能

Last Updated: 2025-08-07

Red Hat Enterprise Linux 8 配置设备映射器多路径

配置和管理设备映射器多路径功能

Legal Notice

Abstract

使用设备映射器多路径（DM Multipath），您可以将服务器节点和存储阵列间的多个 I/O 路径配置为单一设备。这些 I/O 路径是可包含独立电缆、交换机和控制器的物理 SAN 连接。 多路径聚合了 I/O 路径并生成由聚合路径组成的新设备。

Table of Contents

对红帽文档提供反馈

我们感谢您对我们文档的反馈。让我们了解如何改进它。

通过 Jira 提交反馈（需要帐户）

1. 登录到 Jira 网站。
2. 单击顶部导航栏中的 **Create**。
3. 在 **Summary** 字段中输入描述性标题。
4. 在 **Description** 字段中输入您对改进的建议。包括文档相关部分的链接。
5. 点对话框底部的 **Create**。

第 1 章 设备映射器多路径概述

DM 多路径提供：

冗余

DM 多路径可在主动/被动（active/passive）配置中提供故障切换。在主动/被动配置中，任何时候路径的子集都只用于 I/O 。如果 I/O 路径的任何元素（如电缆、交换机或控制器）出现故障，DM 多路径会切换到备用路径。

注意

路径的数量取决于设置。通常，DM 多路径设置有 2、4 或 8 个到存储的路径，但这是一个常见设置，也可以对路径使用其他数字。

改进的性能

可将 DM 多路径配置为主动/主动模式，其中将 I/O 以轮循（round-robin）方式分布到所有路径中。在某些配置中，DM-Multipath 可以检测 I/O 路径上的加载，并动态重新平衡负载。

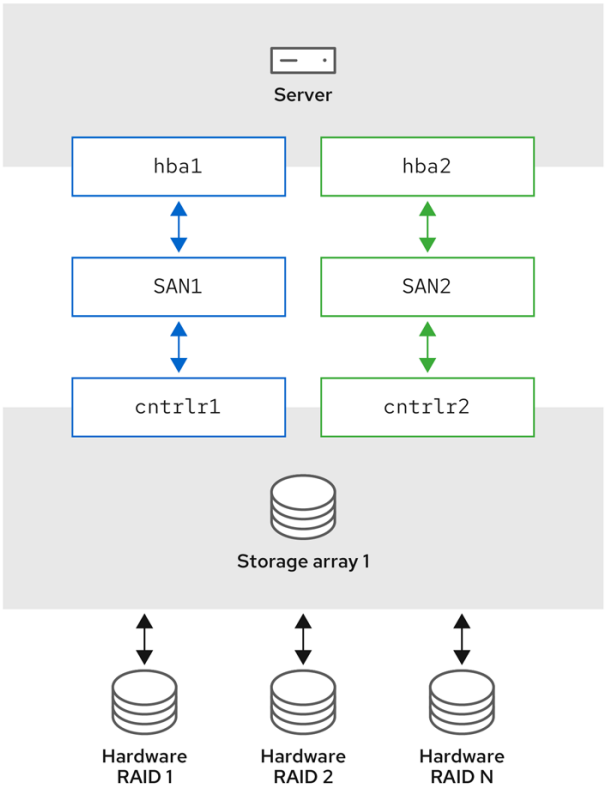
1.1. 带一个 RAID 设备的主动/被动多路径配置

在此配置中，服务器上有两个主机总线适配器(HBA)，两个 SAN 交换机和两个 RAID 控制器。以下是在这个配置中可能出现的故障：

- HBA 故障
- 光纤通道电缆失败
- SAN 交换机故障
- 阵列控制器端口故障

配置 DM 多路径后，任何这些点会导致 DM 多路径切换到备用 I/O 路径。以下镜像描述了服务器到 RAID 设备的两个 I/O 路径的配置。在这里，有一个 I/O 路径通过 **hba1**、**SAN1** 和 **cntrlr1**，第二个 I/O 路径则经过 **hba2**、**SAN2** 和 **cntrlr2**。

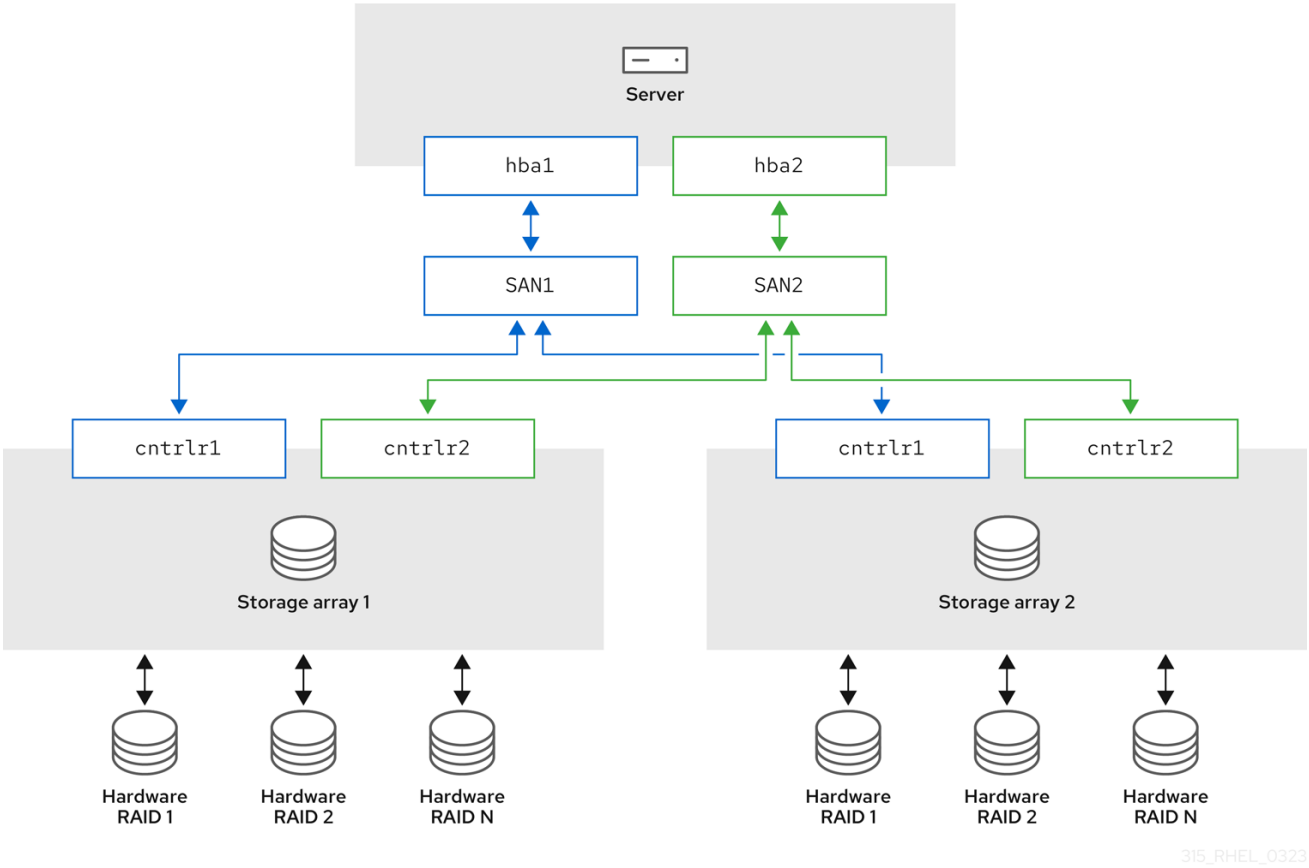
图 1.1. 带一个 RAID 设备的主动/被动多路径配置



1.2. 带两个 RAID 设备的主动/被动多路径配置

在此配置中，服务器中存在两个 HBA，两个 SAN 交换机，每个有两个 RAID 控制器。配置 DM 多路径后，在任意 RAID 设备的 I/O 路径点会导致 DM 多路径切换到该设备的备用 I/O 路径。下图展示了一个配置，每个 RAID 设备有两个 I/O 路径。在这里，每个 RAID 设备有两个 I/O 路径。

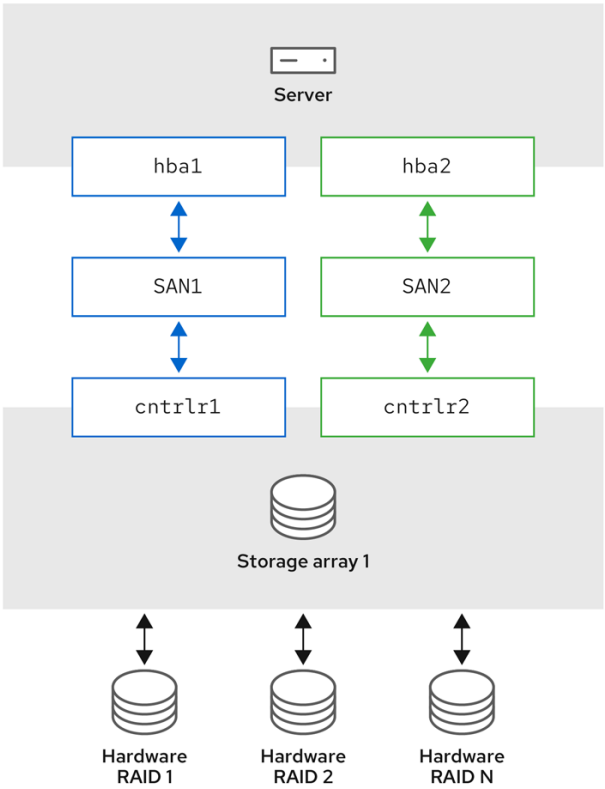
图 1.2. 带两个 RAID 设备的主动/被动多路径配置



1.3. 带一个 RAID 设备的主动/主动多路径配置

在此配置中，服务器中有两个 HBA、两个 SAN 交换机和两个 RAID 控制器。以下镜像描述了从服务器到存储设备的两个 I/O 路径的配置。在这里，可将 I/O 分布到这两个路径中。

图 1.3. 带一个 RAID 设备的主动/主动多路径配置



1.4. DM 多路径组件

下表描述了 DM 多路径组件。

表 1.1. DM 多路径的组件

组件	描述
dm_multipath 内核模块	为路径和路径组群重新路由 I/O 并支持故障切换。
mpathconf 工具	配置并启用设备映射器多路径。
multipath 命令	列出并配置多路径设备。每当添加块设备时，它也由 **udev** 执行，以确定该设备是否是多路径设备的一部分。
multipathd 守护进程	自动创建和删除多路径设备并监控路径；作为路径失败，可以更新多路径设备。允许对多路径设备进行交互式的修改。如果 **/etc/multipath.conf** 文件有任何更改，请重新加载该服务。
kpartx 命令	为设备中的分区创建设备映射器设备。当创建了多路径设备以便在其之上创建分区设备时，该命令将由 **udev** 自动执行。**kpartx** 命令在其自己的软件包中提供，但 **device-mapper-multipath** 软件包依赖于它。

mpathpersist	在多路径设备中设置 **SCSI-3** 持久预留。这个命令的工作方式与 **sg_persist** 对于 SCSI 设备的工作方式相似，它们不是多路径的，但它会处理在多路径设备的所有路径中设置持久性保留的方法。它与 **multipathd** 协调，以确保在稍后添加的路径上正确设置保留。要使用此功能，必须在 **/etc/multipath.conf** 文件中定义 **reservation_key** 属性。否则 **multipathd** 守护进程将不会检查新发现的路径或恢复的路径。

1.5. 显示多路径拓扑

要有效地监控路径，对多路径问题进行故障排除，或者检查多路径配置是否被正确设置，您可以显示多路径拓扑。

步骤

1. 显示多路径设备拓扑：

```
# multipath -ll
mpatha (3600d0230000000000e13954ed5f89300) dm-4 WINSYS,SF2372
size=233G features='1 queue_if_no_path' hwhandler='0' wp=rw
`-+- policy='service-time 0' prio=1 status=active
  `- 6:0:0:0 sdf 8:80 active ready running
```

输出可以被分成三个部分。每个部分都显示以下组的信息：

- 多路径设备信息：
  - **mpatha (3600d0230000000000e13954ed5f89300)**: alias (如果与别名不同，则为 wwid)
  - **dm-4**: dm 设备名称
  - **WINSYS,SF2372**: 厂商，产品
  - **size=233G**: 大小
  - **features='1 queue_if_no_path'**: 特性
  - **hwhandler='0'**: 硬件处理程序
  - **wp=rw**: 写权限
- 路径组信息：
  - **policy='service-time 0'**: 调度策略
  - **prio=1** ：路径组优先级
  - **status=active**: 路径组状态
- 路径信息：

- **6:0:0:0**: host:channel:id:lun
- **Sdf**: devnode
- **8:80**: major:minor 号
- **active**: dm 状态
- **ready** ：路径状态
- **running** ：在线状态
  有关 dm、路径和在线状态的更多信息，请参阅 路径状态。

用于列出、创建或重新载入多路径设备的其他多路径命令也会显示设备拓扑。但是，某些信息可能未知，并在输出中显示为 **undef**。这是正常行为。使用 **multipath -ll** 命令查看正确的状态。

注意

在某些情况下，如创建多路径设备，多路径拓扑会显示一个参数，其表示是否采取了任何措施。例如，以下命令输出显示 **create:** 参数，来表示创建了一个多路径设备：

```
create: mpatha (3600d0230000000000e13954ed5f89300) undef WINSYS,SF2372
size=233G features='1 queue_if_no_path' hwhandler='0' wp=undef
`-+- policy='service-time 0' prio=1 status=undef
  `- 6:0:0:0 sdf 8:80 undef ready running
```

1.6. 路径状态

multipathd 守护进程会根据 **/etc/multipath.conf** 文件中定义的轮询间隔定期对路径状态进行更新。就内核而言，**dm** 状态与路径状态类似。**dm** 状态将保持其当前状态，直到路径检查程序完成为止。

路径状态

ready, ghost
: 路径已启动，并准备好进行 I/O 操作。

faulty, shaky
: 路径停用。

i/o pending
: 检查程序正在主动检查此路径，状态将很快被更新。

i/o timeout
: 检查程序在超时前没有返回 **success/failure**。这被视为与 **faulty** 一样。

removed
: 路径已从系统中删除，很快将从多路径设备中删除。这被视为与 **faulty** 一样。

wild
: **multipathd** 无法运行路径检查程序，因为内部错误或配置问题。这被视为与 **faulty** 一样，但多路径将跳过对路径的多个操作。

unchecked
: 还没有对此路径运行路径检查程序，因为它刚刚被发现，它没有分配的路径检查程序，或者路径检查程序遇到了错误。这与 **wild** 相同。

delayed

路径检查程序返回路径已启动，但多路径会延迟路径的恢复，因为路径最近失败了多次，且多路径在此情况下已被配置为延迟路径。这被视为与 **faulty** 一样。

Dm 状态

Active

映射到 **ready** 和 **ghost** 路径状态。

Failed

映射到所有其他路径状态，但 **i/o pending** 除外，其没有一个对等的 **dm** 状态。

在线状态

Running

设备已启用。

Offline

设备已被禁用。

1.7. 其他资源

- 系统中的 **multipath (8)** 和 **multipathd (8)** man page
- **/etc/multipath.conf** 文件

第 2 章 多路径设备

DM 多路径提供了一种逻辑地整理 I/O 路径的方法，方法是在基础设备上创建单一多路径设备。如果没有 DM 多路径，系统会将服务器节点中的每个路径都把一个存储控制器视为单独的设备，即使 I/O 路径将相同的服务器节点连接到同一存储控制器。

2.1. 多路径设备识别符

当新设备受 DM 多路径控制时，这些设备会在 **/dev/mapper/** 和 **/dev/** 目录中创建。

注意

任何格式为 **/dev/dm-X** 的设备都仅供内部使用，且不应该被管理员直接使用。

下面描述了多路径设备名称：

- 当 **user_friendly_names** 配置选项设置为 **no** 时，多路径设备的名称被设置为 World Wide Identifier(WWID)。默认情况下，多路径设备的名称被设置为它的 WWID。设备名称应为 **/dev/mapper/WWID**。它还在 **/dev/** 目录中创建，名为 **/dev/dm-X**。

- 或者，您可以在 **/etc/multipath.conf** 文件中将 **user_friendly_names** 选项设置为 **yes**。这会将 **multipath** 部分中的 **alias** 设置为 **mpathN** 格式的节点唯一名称。该设备名称应该是 **/dev/mapper/mpathN** 和 **/dev/dm-X**。但不能保证，在所有使用多路径设备的节点中的设备名称都是一致的。同样，如果您在 **/etc/multipath.conf** 文件中设置了 **alias** 选项，该名称不会自动在集群中的所有节点中保持一致。

注意

如果您使用 LVM 在多路径设备中创建逻辑设备，这不应造成问题。为了使您的多路径设备名称在每个节点上一致，红帽建议禁用 **user_friendly_names** 选项。

例如：一个带有两个 HBA 的节点，通过一个没有区的 FC 交换机就可以看到四个设备：**/dev/sda**, **/dev/sdb**, **/dev/sdc**, 和 **/dev/sdd**。DM 多路径会创建一个唯一 WWID 设备，它根据多路径配置将 I/O 重新路由到这四个底层设备。

除了 **user_friendly_names** 和 **alias** 选项外，多路径设备还具有其他属性。您可以通过在 **/etc/multipath.conf** 文件的 **multipaths** 部分中为该设备创建条目来修改特定多路径设备的这些属性。

其他资源

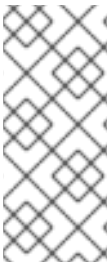
- 您系统上的 **multipath (8)** 和 **multipath.conf (8)** 手册页
- **/etc/multipath.conf** 文件
- DM 多路径组件

2.2. 逻辑卷中的多路径设备

创建多路径设备后，您可以使用多路径设备名称，因为在创建逻辑卷管理器(LVM)物理卷时使用物理设备名称。例如，如果 **/dev/mapper/mpatha** 是多路径设备的名称，则 **pvcreate /dev/mapper/mpatha** 命令将 **/dev/mapper/mpatha** 标记为物理卷。

在创建 LVM 卷组时，您可以使用生成的 LVM 物理设备，就像使用其它 LVM 物理设备一样。

要过滤 **/etc/lvm/lvm.conf** 文件中的所有 **sd** 设备，在文件的 **devices** 部分添加 **filter = ["r/block/", "r/disk/", "r/sd./", "a/./"]** 过滤。



注意

如果您试图在配置的分区的整个设备中创建 LVM 物理卷，则 **pvcreate** 命令会失败。如果您不具体指定每个块设备，Anaconda 和 Kickstart 安装程序会创建空分区表。如果您要使用整个设备而不是创建分区，请从该设备中删除现有分区。您可以使用 **kpartx -d** device 命令和 **fdisk** 实用程序删除现有分区。如果您的系统有大于 2Tb 的块设备，使用 **parted** 工具删除分区。

当您创建使用 **active/passive** 多路径设备作为基础物理设备的 LVM 逻辑卷时，您可以选择在 **/etc/lvm/lvm.conf** 文件中包含过滤器，以排除多路径设备下的磁盘。这是因为如果阵列在收到 I/O 时自动更改被动路径，则当没有过滤这些设备时，多路径都会在 LVM 扫描被动路径时进行故障转移。

内核通过自动检测要使用的正确硬件处理程序来更改主动/被动状态。对于需要干预以改变其状态的主动/被动路径，多路径会自动使用这个硬件处理器根据需要进行操作。如果内核没有自动检测要使用的正确硬件处理程序，您可以使用"hardware_handler"选项配置 multipath.conf 文件中要使用的硬件处理程序。对于需要命令使被动路径被为主动的 **active/passive** 阵列，LVM 会在发生这种情况时输出警告信息。

根据您的配置，LVM 可能会输出以下任何信息：

- LUN 未就绪：

```
end_request: I/O error, dev sdc, sector 0
sd 0:0:0:3: Device not ready: <6>: Current: sense key: Not Ready
    Add. Sense: Logical unit not ready, manual intervention required
```

- 读失败：

```
/dev/sde: read failed after 0 of 4096 at 0: Input/output error
```

以下是上述错误的原因：

- 在为机器提供主动/被动路径的存储设备中设置多路径。
- 路径是直接访问的，而不是通过多路径设备访问。

其他资源

- 系统中的 **lvm.conf** 手册页
- DM 多路径组件

第 3 章 配置 DM 多路径

您可以使用 **mpathconf** 工具设置 DM 多路径。这个工具会根据以下情况创建或编辑 **/etc/multipath.conf** 多路径配置文件：

- 如果 **/etc/multipath.conf** 文件已存在，则 **mpathconf** 实用程序将编辑该文件。
- 如果 **/etc/multipath.conf** 文件不存在，则 **mpathconf** 实用程序将从头开始创建 **/etc/multipath.conf** 文件。

3.1. 检查 DEVICE-MAPPER-MULTIPATH 软件包

在您的系统中设置 DM 多路径前，请确定您的系统是最新的，并包含 **device-mapper-multipath** 软件包。

步骤

1. 检查您的系统是否包含 **device-mapper-multipath** 软件包：

```
# rpm -q device-mapper-multipath
device-mapper-multipath-current-package-version
```

   如果您的系统没有包括这个软件包，它会输出以下内容：

```
package device-mapper-multipath is not installed
```

2. 如果您的系统没有包括这个软件包，请运行以下命令安装它：

```
# yum install device-mapper-multipath
```

3.2. 使用 DM 多路径设置基本故障切换配置

您可以为基本故障切换配置设置 DM 多路径，并在启动 multipathd 守护进程前编辑 **/etc/multipath.conf** 文件。

先决条件

- 管理访问权限.

步骤

1. 启用并初始化多路径配置文件：

```
# mpathconf --enable
```

2. 可选：编辑 **/etc/multipath.conf** 文件。
   大多数默认设置都已配置，包括设置为 **failover** 的 **path_grouping_policy** 。
3. 可选：多路径设备的默认命名格式被设置为 **/dev/mapper/mpathn** 格式。如果您希望使用不同的命名格式：

a. 将 DM 多路径配置为使用多路径设备 WWID 作为其名称，而不是 mpath_n_ user 友好命名方案：

```
# mpathconf --enable --user_friendly_names n
```

b. 重新载入 DM 多路径守护进程的配置：

```
# systemctl reload multipathd.service
```

4. 启动 DM 多路径守护进程：

```
# systemctl start multipathd.service
```

验证

- 确认 DM 多路径守护进程正在运行，且没有问题：

```
# systemctl status multipathd.service
```

- 验证多路径设备的命名格式：

```
# ls /dev/mapper/
```

3.3. 在生成多路径设备时忽略本地磁盘

有些机器在其内部磁盘中使用本地 SCSI 卡，我们不建议在这些设备中使用 DM 多路径。如果将 **find_multipaths** 配置参数设置为 **on**，则不必在这些设备上禁用多路径。

如果您没有将 **find_multipaths** 配置参数设置为 **on**，您可以使用以下步骤修改 DM 多路径配置文件，以便在配置多路径时忽略本地磁盘。

步骤

1. 使用任何已知参数（如设备的型号、路径或厂商）识别内部磁盘，并使用以下选项之一确定其 WWID：

- 显示现有的多路径设备：

```
# multipath -v2 -l

mpatha (WDC_WD800JD-75MSA3_WD-WMAM9FU71040) dm-2 ATA,WDC WD800JD-
75MS
size=33 GB features="0" hwhandler="0" wp=rw
`-+- policy='round-robin 0' prio=0 status=active
  |- 0:0:0:0 sda 8:0 active undef running
```

- 显示 DM 多路径可能创建的附加多路径设备：

```
# multipath -v2 -d

: mpatha (WDC_WD800JD-75MSA3_WD-WMAM9FU71040) dm-2 ATA,WDC
WD800JD-75MS
```

```
size=33 GB features="0" hwhandler="0" wp=undef
`-+- policy='round-robin 0' prio=1 status=undef
  |- 0:0:0:0 sda 8:0  undef ready running
```

- 显示设备信息：

```
# multipathd show paths raw format "%d %w" | grep sda
sda WDC_WD800JD-75MSA3_WD-WMAM9FU71040
```

在这个示例中，**/dev/sda** 是内部磁盘，其 WWID 是 **WDC_WD800JD-75MSA3_WD-WMAM9FU71040**。

2. 编辑 **/etc/multipath.conf** 文件的 **blacklist** 部分，以使用其 WWID 属性忽略此设备：

```
blacklist {
    wwid WDC_WD800JD-75MSA3_WD-WMAM9FU71040
}
```

警告

虽然您可以使用其 **devnode** 参数，如 **sda** 来识别设备，但它不能是一个安全的流程，因为无法保证 **/dev/sda** 在重启时引用同一设备。

3. 检查 **/etc/multipath.conf** 文件中的任何配置错误：

```
# multipath -t > /dev/null
```

要查看完整报告，请不要丢弃命令输出：

```
# multipath -t
```

4. 如果 initramfs 中包含磁盘，则重新建立 **initramfs**。如需更多信息，请参阅在 initramfs 中配置多路径。

5. 通过重新配置 **multipathd** 守护进程来重新载入 **/etc/multipath.conf** 文件：

```
# systemctl reload multipathd
```

注意

当在使用时，无法删除本地磁盘之上的多路径设备。要忽略这样的设备，请停止该设备的所有用户。例如，通过卸载其上的任何文件系统，并停用使用它的任何逻辑卷。如果这是不可能的，您可以重启系统来删除多路径设备。

验证

1. 验证内部磁盘是否被忽略了，且其没有在多路径输出中显示：

- 列出多路径设备：

```
# multipath -v2 -l
```

- 列出 DM 多路径可以创建的额外设备：

```
# multipath -v2 -d
```

其他资源

- 您系统上的 **multipath.conf (5)** 手册页

3.4. 使用 DM 多路径配置附加存储

默认情况下，DM 多路径包括支持 DM 多路径的最常见存储阵列的内置配置。如果您的存储阵列还没有配置，您可以通过编辑 **/etc/multipath.conf** 文件来添加一个。

注意

在初始配置中添加附加存储设备，使设置与您的预期需求保持一致。DM 多路径允许以后为可扩展性或升级添加设备，但此方法可能需要调整配置以确保兼容性。

步骤

1. 查看默认配置值和支持的设备：

```
# multipathd show config
```

2. 编辑 **/etc/multipath.conf** 文件以设置多路径配置。

例 3.1. HP OPEN-V 存储设备的 DM 多路径配置

```
# Set default configurations for all devices managed by DM Multipath

defaults {
    # Enable user-friendly names for devices
    user_friendly_names yes
}

devices {
    # Define configuration for HP OPEN-V storage
    device {
        vendor "HP"
        pproduct "OPEN-V"
        no_path_retry 18
    }
}
```

3. 保存更改并关闭编辑器。

4. 通过扫描新设备来更新多路径设备列表：

```
# multipath -r
```

验证

- 确认多路径设备是否被正确识别：

```
# multipath -ll
```

3.5. 在 INITRAMFS 中配置多路径

在 **initramfs** 文件系统中设置多路径对于无缝存储功能至关重要，特别是在需要冗余和负载均衡的情况下。这个设置可确保多路径设备在引导过程的早期可用，这对于维护存储设置的完整性至关重要，并可防止潜在的问题。

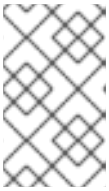
先决条件

- 在您的系统上配置 DM 多路径。

步骤

1. 使用多路径配置文件重建 **initramfs** 文件系统：

```
# dracut --force --add multipath
```



注意

在 **initramfs** 中使用多路径并修改其配置文件时，请记住重建 **initramfs** 以使更改生效。如果您的 root 设备使用多路径，则 **dracut** 命令将在 **initramfs** 中自动包含多路径模块。

2. 可选：如果 **initramfs** 中的多路径不再需要：

   a. 删除多路径配置文件：

```
# rm /etc/dracut.conf.d/multipath.conf
```

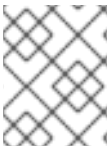
   b. 使用添加的多路径配置重建 **initramfs** ：

```
# dracut --force --omit multipath
```

验证

- 检查与多路径相关的文件和配置是否存在：

```
# lsinitrd /path/to/initramfs.img -m | grep multipath
```



注意

虽然提供的验证步骤可以为您提供成功的指示，建议进行最终的测试引导，以确保配置按预期工作。

- 重启后，确认多路径设备是否被正确识别：

```
# multipath -ll
```

第 4 章 在 NVME 设备中启用多路径

您可以将通过光纤传输（如光纤通道(FC)）连接到您系统的 Non-volatile Memory Express™(NVMe™)设备进行多路径。您可以在多个多路径解决方案之间进行选择。

4.1. 本地 NVME 多路径和 DM 多路径

Non-volatile Memory Express™ (NVMe™)设备支持原生多路径功能。当在 NVMe 中配置多路径时，您可以在标准 DM 多路径和原生 NVMe 多路径之间进行选择。

DM 多路径和原生 NVMe 多路径都支持 NVMe 设备的 Asymmetric Namespace Access(ANA)多路径方案。ANA 识别控制器和主机之间的优化路径，并提高性能。

当启用原生 NVMe 多路径时，它会全局地应用于所有 NVMe 设备。它可以提供更高的性能，但不包含 DM 多路径提供的所有功能。例如，原生 NVMe 多路径只支持 **numa** 和 **round-robin** 路径选择方法。

红帽建议您在 Red Hat Enterprise Linux 8 中使用 DM 多路径作为默认多路径解决方案。

4.2. 启用原生 NVME 多路径

nvme_core.multipath 选项的默认内核设置被设置为 **N**，这意味着原生 Non-volatile Memory Express™(NVMe™)多路径被禁用。您可以使用原生 NVMe 多路径解决方案启用原生 NVMe 多路径。

先决条件

- NVMe 设备连接到您的系统。如需更多信息，请参阅 通过光纤设备概述 NVMe。

步骤

1. 检查内核中是否启用了原生 NVMe 多路径：

```
# cat /sys/module/nvme_core/parameters/multipath
```

这个命令显示以下之一：

N
禁用原生 NVMe 多路径。

Y
启用原生 NVMe 多路径。

2. 如果原生 NVMe 多路径被禁用，请使用以下方法之一启用它：

- 使用内核选项：

a. 在命令行中添加 **nvme_core.multipath=Y** 选项：

```
# grubby --update-kernel=ALL --args="nvme_core.multipath=Y"
```

b. 在 64 位 IBM Z 构架中更新引导菜单：

```
# zipl
```

c. 重启系统。

- 使用内核模块配置文件：

a. 使用以下内容创建 **/etc/modprobe.d/nvme_core.conf** 配置文件：

```
options nvme_core multipath=Y
```

b. 备份 **initramfs** 文件：

```
# cp /boot/initramfs-$(uname -r).img /boot/initramfs-$(uname -r).bak.$(date
+%m-%d-%H%M%S).img
```

c. 重建 **initramfs** ：

```
# dracut --force --verbose
```

d. 重启系统。

3. 可选： 在运行的系统中，更改 NVMe 设备中的 I/O 策略，以便在所有可用路径中分发 I/O：

```
# echo "round-robin" > /sys/class/nvme-subsystem/nvme-subsys0/iopolicy
```

4. 可选：使用 **udev** 规则永久设置 I/O 策略。使用以下内容创建 **/etc/udev/rules.d/71-nvme-io-policy.rules** 文件：

```
ACTION=="add|change", SUBSYSTEM=="nvme-subsystem", ATTR{iopolicy}="round-robin"
```

验证

1. 验证您的系统是否识别 NVMe 设备。以下示例假设您有一个通过光纤存储子系统连接的 NVMe，它有两个 NVMe 命名空间：

```
# nvme list

Node             SN                   Model                                    Namespace Usage
Format           FW Rev
---------------- -------------------- ---------------------------------------- --------- --------------------------
- ---------------- --------
/dev/nvme0n1     a34c4f3a0d6f5cec     Linux                                    1         250.06  GB /
250.06  GB    512   B +  0 B   4.18.0-2
/dev/nvme0n2     a34c4f3a0d6f5cec     Linux                                    2         250.06  GB /
250.06  GB    512   B +  0 B   4.18.0-2
```

2. 列出所有连接的 NVMe 子系统：

```
# nvme list-subsys

nvme-subsys0 - NQN=testnqn
\
 +- nvme0 fc traddr=nn-0x20000090fadd597a:pn-0x10000090fadd597a host_traddr=nn-
0x20000090fac7e1dd:pn-0x10000090fac7e1dd live
 +- nvme1 fc traddr=nn-0x20000090fadd5979:pn-0x10000090fadd5979 host_traddr=nn-
```

```
0x20000090fac7e1dd:pn-0x10000090fac7e1dd live
 +- nvme2 fc traddr=nn-0x20000090fadd5979:pn-0x10000090fadd5979 host_traddr=nn-
0x20000090fac7e1de:pn-0x10000090fac7e1de live
 +- nvme3 fc traddr=nn-0x20000090fadd597a:pn-0x10000090fadd597a host_traddr=nn-
0x20000090fac7e1de:pn-0x10000090fac7e1de live
```

检查活动传输类型。例如，**nvme0 fc** 表示设备通过光纤通道传输连接，**nvme tcp** 则表示设备通过 TCP 连接。

3. 如果您编辑了内核选项，请验证内核命令行上是否启用了原生 NVMe 多路径：

```
# cat /proc/cmdline

BOOT_IMAGE=[...] nvme_core.multipath=Y
```

4. 如果您更改了 I/O 策略，请验证 **round-robin** 是否是 NVMe 设备上活跃的 I/O 策略：

```
# cat /sys/class/nvme-subsystem/nvme-subsys0/iopolicy

round-robin
```

其他资源

- 配置内核命令行参数

4.3. 在 NVME 设备中启用 DM 多路径

您可以通过禁用原生 NVMe 多路径在连接的 NVMe 设备上启用 DM 多路径。

先决条件

- NVMe 设备连接到您的系统。如需更多信息，请参阅 通过光纤设备概述 NVMe。

步骤

1. 检查原生 NVMe 多路径是否被禁用：

```
# cat /sys/module/nvme_core/parameters/multipath
```

这个命令显示以下之一：

N
禁用原生 NVMe 多路径。

Y
启用原生 NVMe 多路径。

2. 如果启用了原生 NVMe 多路径，请使用以下方法之一禁用它：

- 使用内核选项：

a. 在内核命令行中删除 **nvme_core.multipath=Y** 选项：

```
# grubby --update-kernel=ALL --remove-args="nvme_core.multipath=Y"
```

-

b. 在 64 位 IBM Z 构架中更新引导菜单：

```
# zipl
```

c. 重启系统。

- 使用内核模块配置文件：

a. 如果存在，请从 **/etc/modprobe.d/nvme_core.conf** 文件中删除 **nvme_core multipath=Y** 选项行。

b. 备份 **initramfs** 文件：

```
# cp /boot/initramfs-$(uname -r).img /boot/initramfs-$(uname -r).bak.$(date +%m%d-%H%M%S).img
```

c. 重建 **initramfs** ：

```
# cp /boot/initramfs-$(uname -r).img /boot/initramfs-$(uname -r).bak.$(date +%m-%d-%H%M%S).img
# dracut --force --verbose
```

d. 重启系统。

3. 启用 DM 多路径：

```
# systemctl enable --now multipathd.service
```

4. 在所有可用路径中分发 I/O。在 **/etc/multipath.conf** 文件中添加以下内容：

```
devices {
        device {
                vendor "NVME"
                product ".*"
                path_grouping_policy    group_by_prio
        }
}
```

注意

当 DM 多路径管理 NVMe 设备时，**/sys/class/nvme-subsys0/iopolicy** 配置文件不会影响 I/O 分发。

5. 重新载入 **multipathd** 服务以应用配置更改：

```
# multipath -r
```

验证

- 验证原生 NVMe 多路径是否已禁用：

```
# cat /sys/module/nvme_core/parameters/multipath
N
```

- 验证 DM 多路径是否可以识别 nvme 设备：

```
# multipath -l

eui.00007a8962ab241100a0980000d851c8 dm-6 NVME,NetApp E-Series
size=20G features='0' hwhandler='0' wp=rw
`-+- policy='service-time 0' prio=0 status=active
  |- 0:10:2:2 nvme0n2 259:3 active undef running
`-+- policy='service-time 0' prio=0 status=enabled
  |- 4:11:2:2 nvme4n2 259:28 active undef running
`-+- policy='service-time 0' prio=0 status=enabled
  |- 5:32778:2:2 nvme5n2 259:38 active undef running
`-+- policy='service-time 0' prio=0 status=enabled
  |- 6:32779:2:2 nvme6n2 259:44 active undef running
```

其他资源

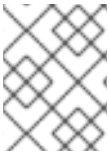
- 配置内核命令行参数
- 配置 DM 多路径

第 5 章 修改 DM 多路径配置文件

默认情况下，DM 多路径会为多数常见的多路径用例提供配置值。另外，DM 多路径包括对自己支持 DM 多路径的最常见存储阵列的支持。

您可以通过编辑 **/etc/multipath.conf** 配置文件来覆盖 DM 多路径的默认配置值。如果需要，您还可以在配置文件中添加不支持的默认存储阵列。在多路径配置文件中，您只需要指定配置所需的部分，或者需要更改默认值的部分。如果文件中没有与您的环境相关的部分，或者不需要覆盖默认值，您可以将其注释掉，因为它们位于初始文件中。

在配置文件中，您也可以使用正则表达式。



注意

如果您从 **initramfs** 文件系统运行多路径并对多路径配置文件进行任何更改，则必须重建 **initramfs** 文件系统以使更改生效

5.1. 配置文件概述

多路径配置文件可分为以下几个部分：

黑名单

不视为多路径的特定设备列表。

blacklist_exceptions

根据 **blacklist** 部分的参数，列出其他将被忽略的多路径设备。

defaults

DM 多路径的常规默认设置。

multipaths

各个多路径设备特性的设置。这些值会覆盖在配置文件中的 **overrides**, **devices**, 和 **defaults** 部分指定的值。

devices

各个存储控制器的设置。这些值覆盖了在 配置文件的 **defaults** 部分中指定的内容。如果您使用默认不支持的存储阵列，您可能需要为阵列创建 **devices** 子部分。

overrides

适用于所有设备的设置。这些值覆盖了在配置文件的 **devices** 和 **defaults** 部分中指定的值。

当系统决定多路径设备的属性时，它会按照以下顺序检查 **multipath.conf** 文件中的单独部分的设置：

1. **multipaths** 部分
2. **overrides** 部分
3. **devices** 部分
4. **defaults** 部分

以下是查看默认配置的方法：

- 如果在多路径设备上安装机器，则默认多路径配置会自动应用。默认配置包括以下内容：
  - 如需默认配置值的完整列表，请执行 **multipath -t** 或 **multipathd show config** 命令。

- 有关描述的配置选项列表，请查看系统中的 **multipath.conf** 手册页。

- 如果您没有在安装过程中设置多路径，请执行 **mpathconf --enable** 命令来获取默认配置。

5.2. 配置文件默认设置

/etc/multipath.conf 配置文件包含一个 **defaults** 部分。本节包含设备映射器 (DM) 多路径的默认配置。默认值根据您的初始设备设置可能会有所不同。

下表描述了 **multipath.conf** 配置文件的 **defaults** 部分中设置的可选属性。如果没有设置它们，则应用来自 **overrides**, 或 **devices** 部分中的默认值。

表 5.1. 多路径配置默认设置

å±€æ€§	描述
polling_interval	指定两个路径检查间隔（以秒为单位）。对于可正常工作的路径，检查间的间隔会逐渐增加到 **max_polling_interval**。
	默认值为 **5**。
max_polling_interval	指定两个路径检查间隔的最大长度（以秒为单位）。
	默认值为 **4 * polling_interval**。
find_multipaths	定义设置多路径设备的模式。可用值包括：
	off: 如果 **find_multipaths** 设为 **off**，则**多路径** 将规则应用于 **strict** 值，**multipathd** 守护进程将规则应用为 **greedy** 值。
	在 : **上** 至少有两个设备不在具有相同 World Wide Identifier (WWID)的**黑名单** 中，或者多路径创建了设备 WWID 的多路径设备（即使该多路径设备不再存在），则该设备被视为多路径设备路径。
	greedy ： **multipathd** 和 **multipath** 多会将每个非黑名单的设备视为多路径设备路径。
	smart ：多路径自动认为，每个非黑名单的设备都是多路径设备路径。如果有一个带有相同 WWID 的第二个路径没有出现在 **find_multipaths_timeout** 指定的时间内，多路径会释放该设备并使其可以被系统其余使用。**multipathd** 守护进程像使用 **on** 值一样应用规则。
	strict ：如果创建使用设备 WWID 的多路径设备，这个值只会将设备视为多路径路径。
	默认值为 **off**。默认 **multipath.conf** 文件将 **find_multipaths** 设置为 **上的**。

å±€æ€§	描述
find_multipaths_timeout	这代表超时（以秒为单位），在检测到第一个路径后等待附加路径（如果设置了 **find_multipaths smart** ）。可能的值包括：
	正值 ：如果使用正值设置，则超时适用于所有非黑名单设备。
	负值: 如果设置了负值，则超时时间只适用于在多路径硬件表中有一个条目的已知设备，这些设备位于内置表中，或者在 **设备** 部分中。其他未知设备使用 1 秒的超时以避免引导延迟。
	0 ：系统应用此属性的内置默认值。
	已知硬件的默认值为 **-10**。这意味着已知设备有 10 秒超时。未知设备有一个 1 秒的超时。如果 **find_multipaths** 属性具有 **Smart** 之外的值，则此属性没有影响。
uxsock_timeout	以毫秒为单位设置 **multipathd** 互动命令的超时。
	对于有大量设备的系统，**multipathd** 互动命令可能会超时并失败。如果发生这种情况，请增加这个超时时间来解决这个问题。
	默认值为 **4000**。
reassign_maps	启用重新分配设备映射器映射。使用这个选项时，**multipathd** 守护进程会重新映射现有的设备映射器映射，以便始终指向多路径设备，而不是底层块设备。可能的值有 **yes**，**no**。默认值为 **no**。
verbosity	默认详细程度值为 **2**。数值越大详细程度越高。有效级别在 **0** 与 **4** 之间。
path_selector	指定用于确定用于下一个 I/O 操作的路径的默认算法。可能的值包括：
	round-robin 0 ：通过路径组中的每个路径进行循环，将相同数量的 I/O 请求发送由 **rr_min_io** 或 **rr_min_io_rq** 确定的。
	queue-length 0 ：发送下一个 I/O 请求组，将路径缩减为预期最少的 I/O 请求数。
	service-time 0 ：发送下一组 I/O 请求，并在预计服务时间最短的情况下降低路径。这通过将未完成的 I/O 的总大小除除以相对吞吐量的每个路径来决定。
	默认值为 **service-time 0**。
path_grouping_policy	指定要应用到未指定多路径的默认路径分组策略。可能的值包括：

å±€æ€§	描述
	failover ：每个优先级组 1 个路径。
	multibus ：在 1 优先级组中的所有有效路径。
	group_by_serial ：每个检测到的序列号 1 优先级组。
	group_by_prio ：每个路径优先级值 1 优先级组。优先级由 **prio** 属性决定。
	group_by_node_name ：每个目标节点名称 1 优先级组。**/sys/class/fc_transport/target*/node_name** 目录包含目标节点名称。
	默认值为 **failover**。
uid_attrs	将这个选项设置为按 WWID 激活合并 **uevents**。此操作可能提高了 uevent 处理效率。也是配置 udev 属性用来确定唯一路径标识符 (WWID) 的替代方法。
	这个选项的值是以空格分开的记录列表，如 **type:ATTR**，其中 **type** 与设备节点名称匹配，**ATTR** 是匹配的 udev 属性的名称。
	如果您配置这个选项，且与设备的设备名称匹配，它会覆盖其他配置的方法以确定这个设备的 WWID。
	您可以通过将该值设置为 **sd:ID_SERIAL dasd:ID_UID nvme:ID_WWN** 来启用 **uevent** 合并。
	默认为 **unset**。
prio	指定调用以获取路径优先级值的默认功能。例如，SPC-3 中的 ALUA 位提供了可被利用的 **prio** 值。可能的值包括：
	const ：对所有路径设置优先级 1。
	EMC ：为 EMC 阵列生成路径优先级。
	sysfs ：从 **sysfs** 生成路径优先级。这个优先级程序接受可选的 **prio_arg** 值 **exclusive_pref_bit**。**sysfs** 值使用 **sysfs** 属性 **access_state** 和 **preferred_path**。

å±€æ€§	描述
	alua ：根据 SCSI-3 ALUA 设置生成路径优先级。如果您在设备配置中指定 **prio alua** 和 **prio_args exclusive_pref_bit**，多路径会创建一个路径，该路径只包含 **exclusive_pref_bit** 设置的路径，并为该路径分配最高优先级最高的路径。有关此情况的更多信息，请参阅 **multipath.conf (5)** 手册页。
	ontap ：为 NetApp 阵列生成路径优先级。
	rdac ：为 LSI/Engenio RDAC 控制器生成路径优先级。
	hp_sw ：在 active/standby 模式中生成 Compaq/HP 控制器的路径优先级。
	hds ：为 Hitachi HDS Modular 存储阵列生成路径优先级。
	random ：在 1 到 10 之间生成随机优先级。
	weightedpath ：根据正则表达式和所提供的优先级作为参数生成路径优先级。需要一个 **prio_args** 关键字。
	path_latency ：根据延迟算法生成路径优先级。需要一个 **prio_args** 关键字。
	ana ：根据 NVMe ANA 设置生成路径优先级。这个优先级例程是硬件依赖的。
	datacore ：为某些 DataCore 存储阵列生成路径优先级。需要一个 **prio_args** 关键字。这个优先级例程是硬件依赖的。
	iet ：根据其 IP 地址为 iSCSI 目标生成路径优先级。需要一个**prio_args** 关键字。此优先级例程仅通过 iSCSI 提供。
	默认值取决于 **detect_prio** 设置。如果将 **detect_prio** 设置为 **yes**，则默认优先级算法为 **sysfs**。唯一的例外是 NetAPP E-Series，其默认为 **alua**。如果 **detect_prio** 设为 **no**，则默认优先级算法为 **const**。
prio_args	传递给 **prio** 功能的参数。这只适用于以下优先级：
	weighted: 格式需要是 **<hbtl,devname,serial,wwn> <regex1> <prio1> <regex2> <prio2>**
	HBTL ：Regex 值可以是 SCSI H:B:T:L 格式。例如：**1:0:.:. , *:0:0:**。
	devname ：Regex 值可以是设备名称格式。例如：**sda**, **sd.e**。

å±€æ€§	描述
	serial ：Regex 值可以是序列号格式。通过 **sysfs** 查找 **串行** 设备，或者运行命令 **multipathd show paths format "%z"**。
	wwn: Regex 值可采用 **host_wwnn:host_wwpn:target_wwn:target_wwpn** 形式。这些值可以通过 **sysfs** 或命令 **multipathd show paths format %N:%R:%n:%r"** 查找。
	path_latency: 需要的值，格式为 **io_num= <integer> base_num= <integer>**。
	io_num ：读取 IO 的数量，持续发送到当前路径。这个值有助于计算平均路径延迟。有效值包括 **Integer**, **[2, 200]**。
	base_num: logarithmic scale 的基本数字值。这个值有助于对不同的优先级进行分区。有效值包括 **Integer**, **[2, 10]**。最大的延迟值是 **100s**，最低平均延迟值为 **1us**。
	alua ：如果设置了 **exclusive_pref_bit** 值，则设置了 **preferred_path_bit** 的路径始终创建自己的路径组。
	sysfs ：如果设置了 **exclusive_pref_bit** 值，则设置了 **preferred_path_bit** 的路径始终创建自己的路径组。
	datacore: 需要的值，格式为 **timeout=<milliseconds> preferredsds=<name>**。
	preferredsds ：必需的值，它代表首选的 SDS 名称。
	timeout: 这个值是可选的。以毫秒为单位设置查询超时。
	iet: 一个需要的值，格式为 **preferredip=<ip_address>**。
	preferredip ：此值是必需的。这是 iSCSI 目标的首选 IP 地址，使用点十进制表示法。
	默认值为 **unset**。
功能	多路径设备的默认额外功能，格式为 *"number_of_features_plus_arguments feature1 ..."*。
	features 的可能值包括：
	queue_if_no_path ：与将 **no_path_retry** 设置为 **queue** 的效果相同。

å±€æ€§	描述
	pg_init_retries *n*: 在失败前，重新尝试路径组初始化最多 *n* 次。这个值必须在 1 到 50 之间。
	pg_init_delay_msecs *msecs*: **pg_init** 重新尝试初始化前的时间（毫秒）。这个值必须在 0 到 60000 之间。
	queue_ *mode* ：选择每个多路径设备的队列模式。*mode* 值选项包括 **bio**、**rq** 或 **mq**。它们分别对应于基于 bio、request 和 block-multiqueue 请求 (**blk-mq**)。
	默认值为 *unset*。默认也可以依赖于内核参数 **dm_mod.use_blk_mq**。如果已在 参数中设置了两个选项，则这两个选项为 **mq**，否则为 **rq**。
path_checker	指定确定路径状态的默认方法。可能的值包括：
	readsector0 ：读该设备的第一个扇区。
	tur ：向设备发生一个 **TEST UNIT READY** 命令。
	emc_clariion: 查询 EMC Clariion 特定的 EVPD 页 0xC0 以确定路径。
	hp_sw ：检查使用 Active/Standby 固件的 HP 存储阵列的路径状态。
	rdac ：检查 LSI/Engenio RDAC 存储控制器的路径状态。
	directio ：阅读带有直接 I/O 的第一个扇区。
	cciss_tur ：检查 HP/COMPAQ Smart Array (CCISS) 控制器的路径状态。这依赖于具体硬件。
	none ：不检查该设备。退回使用从 **sysfs** 检索的值。
	默认值为 **tur**。
alias_prefix	此属性代表 **user_friendly_names** 前缀。
	默认值为 **mpath**。
failback	管理路径组故障恢复。可能的值包括：
	immediate ：指定包含主动路径的最高优先级路径组的即时故障。
	manual ：指定没有立即故障恢复，但可以通过操作员的干预来进行故障恢复。

å±€æ€§	描述
	followover ：指定只有路径组的第一个路径处于活跃状态时，才能执行自动故障恢复。当另一个节点请求故障切换时，这会让节点自动进行故障恢复。
	大于零的数字值指定延迟故障恢复，以秒为单位表示。
	默认值为 **manual**。
rr_min_io	指定在切换到当前路径组中的下一个路径前要路由到路径的 I/O 请求数量。此设置仅适用于运行超过 2.6.31 的内核。较新的系统应使用 **rr_min_io_rq**。默认值为 **1000**。
rr_min_io_rq	指定在切换到当前路径组中的下一个路径前要路由到路径的 I/O 请求数量。使用基于请求的 device-mapper-multipath。此设置可用于运行当前内核的系统。在运行早于 2.6.31 的内核的系统上，请使用 **rr_min_io**。默认值为：**1**。
no_path_retry	此属性的数字值指定在禁用排队前，路径检查程序必须针对多路径设备中的所有路径失败的次数。
	值 **fail** 表示立即失败，而不排队。
	值 **queue** 表示在路径修复前排队不应停止。
	默认值为 **fail**。
user_friendly_names	可能的值包括：
	yes ：指定系统可以使用 **/etc/multipath/bindings** 文件为多路径分配永久唯一的别名，格式为 **mpath<n>**。
	no ：系统使用 WWID 作为多路径的别名。在配置文件的 **multipaths** 部分中设置的所有特定于设备的别名，可覆盖此名称。
	默认值为 **no**。
queue_without_daemon	如果设置为 **no**，则 **multipathd** 守护进程会禁用所有设备的队列，在关闭时。默认值为 **no**。
flush_on_last_del	如果设置为 **yes**，则 **multipathd** 守护进程会在删除最后的路径到设备时禁用排队。默认值为 no。
max_fds	设置多路径和 **multipathd** 守护进程可打开的最大打开文件描述符数量。这等同于 **ulimit -n** 命令。默认值为 **max**，它从 **/proc/sys/fs/nr_open** 被设置为系统限制。

å±€æ€§	描述
checker_timeout	用来使用优先级优先级和路径检查器的超时时间（以秒为单位）发出带有显式超时的 SCSI 命令。**sys/block/sd<x>/device/timeout** 目录包括默认值。
fast_io_fail_tmo	当在一个 FC 远程端口中出现问题后 SCSI 层需要等待的时间（秒）才使到远程端口上的设备 I/O 失败。这个值需要小于 **dev_loss_tmo** 的值。把它设置为 **off** 会禁用超时。默认值为 **5**。**fast_io_fail_tmo** 选项会覆盖底层路径设备的 **recovery_tmo** 和 **replacement_timeout** 选项的值。
dev_loss_tmo	SCSI 层在 FC 远程端口上检测到问题后等待的秒数，然后再从系统中删除。把它设置为 infinity 将会将其设置为 2147483647 秒（68 年）。OS 决定默认值。
eh_deadline	指定 SCSI 层在 SCSI 设备失败时执行错误处理的最大秒数。在这个超时后，scsi 层会执行完整的 HBA 重置。在 **rport** 永不丢失的情况下，需要设置此项，因此 **fast_io_fail_tmo** 和 **dev_loss_tmo** 不会触发，但 **scsi** 命令仍然挂起。当 SCSI 错误处理器执行 HBA 重置时，这会影响那个 HBA 上的所有目标路径。只有当受影响 HBA 中的所有目标都被多路径时，才应设置 **eh_deadline** 值。
	默认值为 **unset**。
detect_prio	如果它被设置为 **yes**，则多路径会检测该设备是否是支持 Asymmetric Logical Unit Access (ALUA)的 SCSI 设备，或者支持 Asymmetric Namespace Access (ANA)的 NVMe 设备。如果设备支持 ALUA，则多路径会自动为其分配 **alua** prioritizer。如果设备支持 ANA，则多路径会自动为其分配 **ana** prioritizer。
	如果 **detect_prio** 设置为 **no**，或者设备不支持 ALUA 或 ANA，则 **prio** 属性会设置 prioritizer。
	默认值为 **yes**。
uid_attribute	指定用于设备 WWID 的 **udev** 属性。
	默认值取决于具体设备：SCSI 设备为 **ID_SERIAL**，DASD 设备为 **ID_UID**，NVMe 设备为 **ID_WWN**。
force_sync	如果设置为 **yes**，则此参数可防止路径检查程序在 async 模式下运行。这意味着一次仅运行一个检查程序。当多个 **multipathd** checkers 并行运行，并可能导致大量 CPU 压力时，这非常有用。
	默认值为 **no**。

å±€æ€§	描述
strict_timing	如果设置为 **yes**，**multipathd** 守护进程会在正好 1 秒后启动一个新的路径检查程序循环，以便每个路径检查在 **polling_interval** 的确切设置秒内进行。在忙碌的系统上，路径检查所需的时间可能长于一秒。接下来的轮循中会考虑缺少的 tick。如果路径检查的时间超过 **polling_interval** 设置的秒，则发出警告。
	默认值为 **no**。
retrigger_tries,**retrigger_delay**	结合使用 **retrigger_tries** 和 **retrigger_delay** 参数，使 **multipathd** retrigger uevents。如果 **udev** 无法完全处理原始的 **uevents**，则会导致多路径无法使用该设备。**retrigger_tries** 参数设置多路径试图在未完全设置时重新触发 **uevent** 的次数。**retrigger_delay** 参数设置重试间隔秒数。这两个选项都接受大于或等于 **0** 的数字。将 **retrigger_tries** 参数设置为 **0** 可禁用重试。将 **retrigger_delay** 参数设置为 **0** 会导致在路径检查程序的下一个循环中重新发出 **uevent**。
	retrigger_tries 的默认值为 **3**。**retrigger_delay** 的默认值为 10。
missing_uev_wait_timeout	此属性控制 **multipathd** 守护进程等待为新创建的多路径设备接收来自 **udev** 的更改事件的秒数。之后，它会自动启用设备重新载入。在大多数情况下，**multipathd** 延迟会在设备上重新加载，直到它从初始表负载接收更改 **uevent**。
	默认值为 **30**。
deferred_remove	如果设置为 **yes**，**multipathd** 会执行延迟删除，而不是在删除最后一个路径设备时进行常规删除。这样可确保如果执行常规删除，且删除失败，则当最后一个用户关闭该设备时，设备会被自动删除。默认值为 **no**。
san_path_err_threshold, **san_path_err_forget_rate**, **san_path_err_recovery_time**	如果将所有三个属性设置为大于零的整数，则它们可让 **multipathd** 守护进程使 multipathd 守护进程无法重新生成路径，从而监控路径检查程序失败的频率。如果路径检查程序失败的次数多于 **san_path_err_threshold** 属性的值，但在 **san_path_err_forget_rate** 检查内，则 **multipathd** 守护进程不会重新恢复路径，直到 **san_path_err_recovery_time** 属性的值返回，且没有任何路径检查失败。
	如需更多信息，请参阅 **multipath.conf (5)** 中的 Shaky paths **检测** 部分。
	默认值为 **no**。

å±€æ€§	描述
marginal_path_double_failed_time, **marginal_path_err_sample_time**, **marginal_path_err_rate_threshold**, **marginal_path_err_recheck_gap_time**	如果 **marginal_path_double_failed_time**,**marginal_path_err_rate_threshold**, 和 **marginal_path_err_recheck_gap_time** 设置为大于 **0** 的整数，则 **marginal_path_err_sample_time** 设置为大于 **120** 的整数， 它们通过测试重复失败的路径的 I/O 故障率，使 **multipathd** 守护进程可以防止 reky 路径。
	如果路径在 **marginal_path_double_failed_time** 属性中设置的值内失败，则 **multipathd** 守护进程不会立即重新恢复它，当路径检查程序决定备份它时。相反，**multipathd** 会将读取 I/O 的稳定流添加到 **marginal_path_err_sample_time** 属性中设置的值（以秒为单位）。如果每千个 I/O 属性的 **marginal_path_err_rate_threshold** 属性中设定了值，**multipathd** 会等待 **marginal_path_err_recheck_gap_time** 秒，然后启动另一个以读取 I/O 测试路径的测试路径。否则，**multipathd** 会重新声明路径。
	如需更多信息，请参阅 **multipath.conf (5)** 中的 Shaky paths **检测** 部分。
	默认值为 **no**。
marginal_pathgroups	可能的值包括：
	on ：当一个边缘路径检测方法决定路径边缘时，系统会重新输入路径并将其放置在单独的 pathgroup 中。只有在首先尝试所有非可能路径组后，此组才会生效。这可以防止在系统仍可使用一些边缘路径时出现 IO 错误。该路径会在配置的时间通过监控后立即返回到常规路径组。
	off: **delay_*_checks**,**marginal_path_***, 和 **san_path_err_*** 属性使系统无法重新生成任何 marginal 或 shaky 路径，直到它们被监控了配置的时间。
	fpin ： **multipathd** 守护进程收到 **fpin** 通知，将路径状态设置为 **marginal** 和重新组路径，如 **on** 值所述。
	marginal_path_* 和 **san_path_err_*** 属性被隐式设置为 **no**。
	如需更多信息，请参阅 **multipath.conf (5)** 中的 Shaky paths **检测** 部分。
	默认值为 **no**。
log_checker_err	如果设置为 **once**，则**multipathd** 会在详细程度 2 中记录第一个路径检查程序错误。在恢复设备之前，系统会在详细程度 3 中记录进一步的错误。如果将 **log_checker_err** 参数设置为 **always**，**multipathd** 始终会在详细程度 2 中记录路径检查程序错误。默认值为 **always**。

å±€æ€§	描述
skip_kpartx	如果设置为 **yes**，则**kpartx** 不会自动在该设备中创建分区。这可让您在没有创建分区的情况下创建多路径设备，即使该设备有分区表。这个选项的默认值为 **no**。
max_sectors_kb	使用此选项，您可以在多路径设备首次激活前，将 **max_sectors_kb** 设备队列参数设置为多路径设备的所有底层路径上的指定的值。每当系统创建新的多路径设备时，设备都会继承路径设备的 **max_sectors_kb** 值。为多路径设备手动增大这个值或降低路径设备的值可能会导致多路径创建大于路径设备的 I/O 操作。使用 **max_sectors_kb** 参数是在路径设备之上创建多路径设备前设置这些值的简单方法，并防止传递任何无效的 I/O 操作。如果您没有设置此参数，path devices 驱动程序会自动设置它，多路径设备会从路径设备中继承它。
ghost_delay	此属性设置在仅使用 ghost 路径创建设备后，多路径在将其准备好在 **systemd** 中使用的秒数。这提供了多路径运行硬件处理程序之前显示的主动路径时间，以便将 ghost 路径切换到活跃路径。
	把它设置为 **0** 或 **no** 使多路径立即将带有 ghost 路径的设备标记为就绪。
	默认值为 **no**。
enable_foreign	此属性启用或禁用外部库。
	该值是一个正则表达式。如果外部库的名称与表达式匹配，则会加载。
	默认情况下，所有库都启用。但是，默认配置文件还会将此属性设置为"^**$"**，这将禁用所有外部库。
recheck_wwid	如果设置为 **yes**，当恢复失败路径时，**multipathd** 守护进程会重新检查路径 WWID。如果 WWID 中有变化，则路径会从当前多路径设备中删除，并作为新路径再次添加。如果手动重新添加，**multipathd** 守护进程还会再次检查路径 WWID。
	这个选项只适用于带有配置的 SCSI 设备，以使用默认的 **uid_attribute**、**ID_SERIAL** 或 **sysfs** 来获取其 WWID。
	默认值为 **no**。
remove_retries	这个选项设定多路径重试删除正在使用的设备的次数。在每次尝试之间，多路径都会不活跃 1 秒。默认值为 **0**，这意味着多路径不会重试删除。
detect_checker	如果设置为 **yes**，则多路径会检查设备是否支持 ALUA 或冗余磁盘阵列控制器 (RDAC)。如果设备支持 ALUA，则多路径会为其分配 **tur path_checker**。如果设备支持 RDAC，**multipathd** 守护进程会为其分配 **rdac path_checker**。如果设备不支持 ALUA 或 RDAC，或者 **detect_checker** 被设置为 **no**，**path_checker** 属性会设置路径检查程序。

å±€æ€§	描述
	默认值为 **yes**。
reservation_key	**mpathpersist** 参数使用此服务操作保留密钥。必须为使用持久保留的所有多路径设备设置它，并且它必须与 **PERSISTENT RESERVE OUT** 参数类别中的 **RESERVATION KEY** 字段相同，其中包含应用客户端提供给设备服务器的 8 字节值。如果您在使用 **mpathpersist** 注册密钥时使用 **--param-aptpl** 选项，您必须将 **:aptpl** 附加到保留密钥的末尾。
	这个参数也可以设置为 **file**，这会导致 **mpathpersist** 自动存储用于在 **prkeys** 文件中注册多路径设备的 **RESERVATION KEY**。然后 **multipathd** 守护进程使用这个密钥在出现时注册额外的路径。当您删除注册时，这会自动从 **prkeys** 文件中删除 **RESERVATION KEY**。它默认是 **unset**。如果需要持久性保留，建议将此属性设置为 **file**。
all_tg_pt	如果此选项在 **mpathpersist** 注册密钥时设为 **yes**，它会将注册的密钥从一个主机视为一个目标端口，就像从一个主机移动到所有目标端口一样。这必须设置为 **yes**，以便在主机中的所有目标端口上自动设置和清除注册密钥，而不是每个主机的每个目标端口成功使用 **mpathpersist**。默认值为 **no**。

其他资源

- 您系统上的 **multipath.conf (5)** 手册页

5.3. 修改多路径配置文件默认设置

multipath.conf 文件 的 defaults 部分中设置的默认值由 DM 多路径使用，除非被 **multipath.conf** 文件的设备、多路径或覆盖部分中指定的属性所覆盖。

流程

1. 查看 **/etc/multipath.conf** 配置文件，其中包含配置默认值模板：

```
#defaults {
#       polling_interval        10
#       path_selector           "round-robin 0"
#       path_grouping_policy    multibus
#       uid_attribute           ID_SERIAL
#       prio                    alua
#       path_checker            readsector0
#       rr_min_io               100
#       max_fds                 8192
#       rr_weight               priorities
#       failback                immediate
#       no_path_retry           fail
#       user_friendly_names     yes
#}
```

2. 覆盖任何配置参数的默认值。您可以从此模板将相关行复制到 **defaults** 部分，并取消注释它。

例如，要将 **path_grouping_policy** 参数覆盖为 **multibus**，而不是默认值 **failover**，请将模板中的相应行复制到配置文件的初始默认值部分，然后取消对它的注释，如下所示：

```
defaults {
        user_friendly_names     yes
        path_grouping_policy    multibus
}
```

3. 通过运行以下命令之一修改多路径配置文件后，验证 **/etc/multipath.conf** 文件：

   - 要显示任何配置错误，请运行：

     ```
     # multipath -t > /dev/null
     ```

   - 要显示使用添加的更改显示新配置，请运行：

     ```
     # multipath -t
     ```

4. 重新载入 **/etc/multipath.conf** 文件并重新配置 **multipathd** 守护进程以使更改生效：

   ```
   # service multipathd reload
   ```

其他资源

- 您系统上的 **multipath.conf (5)** 和 **multipathd (8)** man page

5.4. 配置文件 MULTIPATHS 部分

使用 **/etc/multipath.conf** 配置文件的 **multipaths** 部分设置单个多路径设备的属性。设备映射器(DM)多路径使用这些属性覆盖所有其他配置设置，包括 **overrides** 部分中的设置。

multipaths 部分仅将 **multipath** 子部分识别为属性。下表显示了您可以在 **multipath** 子部分中为每个特定多路径设备设置的属性。这些属性仅适用于指定的多路径。如果几个 **multipath** 子部分与特定的设备全球识别符(WWID)匹配，则这些小节的内容会合并。与之前版本相比，来自最新条目的设置均具有优先权。

表 5.2. multipath 子部分属性

å±€æ€§	描述
wwid	指定多路径设备的 WWID，多路径属性应用到其中。这个参数对于 **multipath.conf** 文件的这个部分是必需的。
alias	指定多路径设备的符号名称，多路径属性应用到其中。如果您使用 **user_friendly_names**，请不要将此值设置为 **mpath <n>**。这可能导致与自动分配的用户友好名称冲突，并为您提供不正确的设备节点名称。

以下示例显示了在配置文件中为两个特定多路径设备指定的多路径属性。第一个设备的 WWID 为 **3600508b4000156d70001200000b0000**，符号链接名为 **yellow**。

示例中的第二个多路径设备的 WWID 为 **1DEC_321816758474**，符号链接名为 **red**。

例 5.1. 多路径属性规格

```
multipaths {
        multipath {
                wwid                    3600508b4000156d70001200000b0000
                alias                   yellow
                path_grouping_policy    multibus
                path_selector           "round-robin 0"
                failback                manual
                no_path_retry           5
        }
        multipath {
                wwid                    1DEC_321816758474
                alias                   red
        }
}
```

其他资源

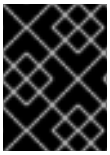
- 您系统上的 **multipath.conf (5)** 手册页
- 配置文件默认设置
- 配置文件覆盖部分

5.5. 配置文件 DEVICES 部分

使用 **multipath.conf** 配置文件的 **devices** 部分为单个存储控制器类型定义设置。本节中设置的值覆盖 **defaults** 部分中的指定的值。

系统根据 **vendor**, **product**, 和 **revision** 关键字标识存储控制器类型。这些关键字是正则表达式，必须与有关特定设备的 **sysfs** 信息匹配。

devices 部分仅将 **device** 子部分识别为属性。如果某个设备有多个关键字匹配，则所有匹配条目的属性都将应用到其中。如果在多个匹配的 **device** 子部分中指定属性，则后续版本的条目优先于任何之前条目。



重要

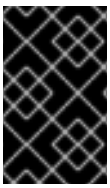
最新版本的 **device** 子部分中的配置属性会覆盖任何之前 **devices** 子部分中的属性，以及 **defaults** 部分中的属性。

下表显示了您可以在 **device** 子部分中设置的属性。

表 5.3. devices 部分属性

å±€æ€§	描述
vendor	指定与设备厂商名称匹配的正则表达式。这是一个必需属性。

å±€æ€§	描述
产品	指定与设备产品名称匹配的正则表达式。这是一个必需属性。
revision	指定与设备产品修订匹配的正则表达式。如果缺少 revision 属性，则所有设备修订都匹配。
product_blacklist	多路径使用此属性创建具有 **vendor** 属性的设备 **blacklist** 项，它与这个设备项的 **vendor** 属性匹配，以及一个 **product** 属性，它与这个 **product_blacklist** 属性匹配。
vpd_vendor	使用 VPD 页缩写来显示特定于供应商的 Vital 产品数据(VPD) 页面信息。
	multipathd 守护进程使用此信息来收集设备特定信息。目前只支持 **hp3par** VPD 页面。
hardware_handler	指定用于特定设备类型的硬件处理器。所有可能的值都依赖于硬件，包括：
	emc：DGC 类数组的硬件处理程序，如 CLARiiON CX/AX 和 EMC VNX 和 unity 系列。
	rdac ：LSI/Engenio/NetApp RDAC 类的硬件处理器，如 NetApp SANtricity E/EF 系列，以及 IBM DELL SGI STK 和 SUN 的 OEM 阵列。
	hp_sw: HP/COMPAQ/DEC HSG80 和 MSA/HSV 阵列的硬件处理程序，只使用 Active/Standby 模式。
	alua：SCSI-3 ALUA 兼容阵列的硬件处理程序.
	ana: NVMe ANA 兼容阵列的硬件处理程序。
	默认值为 **unset**。



重要

Linux 内核、版本 4.3 及更新版本会自动将设备处理程序关联到已知设备。这包括支持 SCSI-3 ALUA 的所有设备。之后，内核不会启用更改处理程序。在这些内核上设置此类设备的 hardware_handler 属性不会起作用。

其他资源

- 您系统上的 **multipath.conf (5)** 手册页
- 配置文件默认设置

5.6. 配置文件覆盖部分

overrides 部分可以识别可选的 **protocol** 子部分，并可包含多个 **protocol** 子部分。系统使用强制 **type** 属性将路径设备与 **protocol** 子匹配。匹配 **protocol** 子部分中的属性优先于其它 **overrides** 部分中的属性。如果有多个匹配的 **protocol** 子部分，则后续条目具有更高的优先级。

protocol 子部分识别以下强制属性：

表 5.4. multipath 子部分属性

å±€æ€§	描述
type	指定路径设备的协议字符串。可能的值包括：
	scsi:fcp, **scsi:spi**, **scsi:ssa**, **scsi:sbp**, **scsi:srp**, **scsi:iscsi**, **scsi:sas**, **scsi:adt**, **scsi:ata**, **scsi:unspec**, **ccw**, **cciss**, **nvme**, **undef**
	此属性不是正则表达式。路径设备协议字符串必须完全匹配。

以下列表中的属性对于 **protocol** 子部分是可选的。如果没有设置它们，则应用来自 **overrides**, **devices** 或 **defaults** 部分中的默认值。

- **fast_io_fail_tmo**
- **dev_loss_tmo**
- **eh_deadline**

其他资源

- 您系统上的 **multipath.conf (5)** 手册页
- 配置文件默认设置

5.7. DM 多路径覆盖设备超时

restore_tmo sysfs 选项控制一个特定 iSCSI 设备的超时时间。以下选项全局覆盖 **recovery_tmo** 值：

- **replacement_timeout** 配置选项会全局覆盖所有 iSCSI 设备的 **recovery_tmo** 值。
- 对于由 DM 多路径管理的所有 iSCSI 设备，DM 多路径中的 **fast_io_fail_tmo** 选项会全局覆盖 **recovery_tmo** 值。
  DM 多路径中的 **fast_io_fail_tmo** 选项会覆盖光纤通道设备的 **fast_io_fail_tmo** 选项。

DM 多路径 **fast_io_fail_tmo** 选项优先于 **replacement_timeout**。每次重新载入 **multipathd** 服务时，它会将 **recovery_tmo** 重置为 **fast_io_fail_tmo** 配置选项的值。使用 DM 多路径 **fast_io_fail_tmo** 配置选项覆盖由 DM 多路径管理的设备中的 **recovery_tmo**。

5.8. 修改具体设备的多路径设置

在 **multipath.conf** 配置文件的 **multipaths** 部分中，您可以添加特定于单个多路径设备的配置，由强制 WWID 参数引用。

这些默认设置由 DM 多路径使用，并覆盖 **multipath.conf** 文件的 **overrides**、**default** 和 **devices** 部分设置的属性。**multipaths** 部分可能存在任意数量的多路径子部分。

流程

1. 修改特定多路径设备的 **multipaths** 部分。以下示例显示了在配置文件中为两个特定多路径设备指定的多路径属性：

   - 第一个设备的 WWID 为 **3600508b4000156d70001200000b0000**，符号链接名为 **yellow**。
   - 示例中的第二个多路径设备的 WWID 为 **1DEC_321816758474**，符号链接名为 **red**。

   在本例中，**rr_weight** 属性设置为 **priorities**。

```
multipaths {
        multipath {
                wwid                    3600508b4000156d70001200000b0000
                alias                   yellow
                path_grouping_policy    multibus
                path_selector           "round-robin 0"
                failback                manual
                rr_weight               priorities
                no_path_retry           5
        }
        multipath {
                wwid                    1DEC_321816758474
                alias                   red
                rr_weight               priorities
        }
}
```

2. 通过运行以下命令之一修改多路径配置文件后，验证 **/etc/multipath.conf** 文件：

   - 要显示任何配置错误，请运行：

```
# multipath -t > /dev/null
```

   - 要显示使用添加的更改显示新配置，请运行：

```
# multipath -t
```

3. 重新载入 **/etc/multipath.conf** 文件并重新配置 **multipathd** 守护进程以使更改生效：

```
# service multipathd reload
```

其他资源

- 您系统上的 **multipath.conf (5)** 手册页

5.9. 使用协议修改特定设备的多路径配置

您可以根据其传输协议配置多路径设备路径。通过使用 **/etc/multipath.conf** 文件中的 **overrides** 部分中的 **protocol** 子部分，您可以覆盖特定路径上的多路径配置设置。这可让通过多种传输协议访问多路径设备，如 Fiber Channel (FC)或互联网小型计算机系统接口(iSCSI)。

protocol 子部分中设定的选项会覆盖覆盖 **overrides**, **devices** 和 **defaults** 部分中的值。这些选项只适用于使用匹配小节的 **type** 参数的传输协议的设备。

先决条件

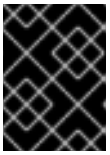
- 您已在系统中配置了设备映射器 (DM) 多路径。
- 您有多路径设备，其中并非所有路径都使用相同的传输协议。

流程

1. 运行以下命令查看具体路径协议：

```
# multipathd show paths format "%d %P"
dev protocol
sda scsi:ata
sdb scsi:fcp
sdc scsi:fcp
```

2. 通过为每个多路径类型添加 **protocol** 子部分，编辑 **/etc/multipath.conf** 文件的 **overrides** 部分。
overrides 部分可以包含多个 **protocol** 部分。



重要

protocol 部分必须包含 **type** 参数。然后，使用匹配 **type** 参数配置所有路径，然后使用 **protocol** 子部分中列出的其余参数进行更新。

- 路径设备的设置，使用 **scsi:fcp** 协议：

```
overrides {
        dev_loss_tmo 60
        fast_io_fail_tmo 8
        protocol {
                type "scsi:fcp"
                dev_loss_tmo 70
                fast_io_fail_tmo 10
                eh_deadline 360
        }
}
```

- 使用 **scsi:iscsi** 协议的路径设备设置：

```
overrides {
        dev_loss_tmo 60
        fast_io_fail_tmo 8
        protocol {
                type "scsi:iscsi"
                dev_loss_tmo 60
                fast_io_fail_tmo 120
```

```
        }
}
```

- 路径设备的设置，使用所有其他协议：

```
overrides {
        dev_loss_tmo 60
        fast_io_fail_tmo 8
        protocol {
                type "<type of protocol>"
                dev_loss_tmo 60
                fast_io_fail_tmo 8
        }
}
```

其他资源

- 您系统上的 **multipath.conf (5)** 手册页

5.10. 修改存储控制器的多路径设置

multipath.conf 配置文件的 **devices** 部分为独立的存储设备设置属性。这些属性可由 DM 多路径使用，除非被包括该设备的路径的 **multipath.conf** 文件的 **multipaths** 或 **overrides** 部分的内容覆盖。这些属性覆盖 **multipath.conf** 文件的 **defaults** 部分中设置的属性。

流程

1. 查看默认配置值的信息，包括支持的设备：

```
# multipathd show config
# multipath -t
```

在多路径配置中，默认包括支持多路径的许多设备。

2. 可选：如果需要修改默认配置值，您可以通过在配置文件中包含覆盖这些值的设备的条目来覆盖默认值。您可以复制 **multipathd show config** 命令显示的设备的设备配置默认值，并覆盖您要更改的值。

3. 通过设置 **vendor** 和 **product** 参数，将没有被默认自动配置的设备添加到配置文件的 **devices** 部分。打开 **/sys/block/*device_name*/device/vendor** 和 **/sys/block/*device_name*/device/model** 文件，其中 *device_name* 是多路径的设备，如下例所示：

```
# cat /sys/block/sda/device/vendor
WINSYS
# cat /sys/block/sda/device/model
SF2372
```

4. 可选： 根据您的具体设备指定附加参数：

主动/主动 设备

通常，在这种情况下不需要设置附加参数。如果需要，您可以将 **path_grouping_policy** 设置为 **multibus**。其他可能需要设置的参数为 **no_path_retry** 和 **rr_min_io**。

主动/被动 设备

如果它自动将 I/O 的路径切换到被动路径，您需要将检查程序功能更改为不会将 I/O 发送到路径路径，以测试其是否工作，否则您的设备会保持故障。这意味着，您已将 **path_checker** 设置为 **tur**，它适用于支持 Test unit Ready 命令的所有 SCSI 设备。

如果设备需要特殊命令来切换路径，则为多路径配置这个设备需要硬件处理器内核模块。当前可用的硬件处理器是 **emc**。如果您的设备不够，您可能无法为多路径配置设备。

以下示例显示了多路径配置文件中的 **device** 条目：

```
# }
# device {
#  vendor   "COMPAQ  "
#  product   "MSA1000         "
#  path_grouping_policy multibus
#  path_checker  tur
#  rr_weight  priorities
# }
#}
```

5. 通过运行以下命令之一修改多路径配置文件后，验证 **/etc/multipath.conf** 文件：

   - 要显示任何配置错误，请运行：

     ```
     # multipath -t > /dev/null
     ```

   - 要显示使用添加的更改显示新配置，请运行：

     ```
     # multipath -t
     ```

6. 重新载入 **/etc/multipath.conf** 文件并重新配置 **multipathd** 守护进程以使更改生效：

   ```
   # service multipathd reload
   ```

其他资源

- 您系统上的 **multipath.conf (5)** 和 **multipathd (8)** man page

5.11. 为所有设备设定多路径值

使用 **multipath.conf** 配置文件的 **overrides** 部分，您可以为所有设备设置配置值。这部分支持 **multipath.conf** 配置文件的 **devices** 和 **defaults** 部分支持的所有属性，这是除 **vendor**, **product**, 和 **revision** 以外的所有 **devices** 项属性。

DM 多路径为所有设备使用这些属性，除非被 multipath.conf 文件的 **multipath.conf** 文件的 **multipaths** 部分中指定的属性覆盖。这些属性覆盖 **multipath.conf** 文件的 **devices** 和 **defaults** 部分中设置的属性。

流程

1. 覆盖特定于设备的设置。例如，您可能希望所有设备都将 **no_path_retry** 设置为 **fail**。当所有路径都失败时，使用以下命令关闭队列。这会覆盖任何特定于设备的设置。

```
overrides {
        no_path_retry fail
}
```

2. 通过运行以下命令之一修改多路径配置文件后，验证 **/etc/multipath.conf** 文件：

   - 要显示任何配置错误，请运行：

     ```
     # multipath -t > /dev/null
     ```

   - 要显示使用添加的更改显示新配置，请运行：

     ```
     # multipath -t
     ```

3. 重新载入 **/etc/multipath.conf** 文件并重新配置 **multipathd** 守护进程以使更改生效：

   ```
   # service multipathd reload
   ```

其他资源

- 您系统上的 **multipath.conf (5)** 手册页

第 6 章 防止设备多路径

您可以将 DM 多路径配置为在配置多路径设备时忽略所选设备。DM 多路径不会将这些忽略的设备分组到多路径设备中。

6.1. DM 多路径为路径创建多路径设备的条件

DM 多路径有一组默认规则，用于决定是否为路径创建多路径设备还是忽略路径。您可以配置行为。

如果将 **find_multipaths** 配置参数设定为 **off**，则多路径总是会尝试为每个未明确禁用的路径创建一个多路径设备。如果 **find_multipaths** 配置参数被设置为 **on**，则只在满足以下条件之一时，多路径会创建一个设备：

- 至少有两个路径有相同的全局-Wide Identification(WWID)没有禁用。
- 您可以使用 **multipath** 命令指定设备来手动强制创建设备。
- 一个路径的 WWID 与之前创建的多路径设备相同，即使那个多路径设备目前还不存在。每当创建多路径设备时，多路径都会记住设备的 WWID，以便在看到该 WWID 的路径时立即自动创建该设备。这可让您让多路径自动选择到多路径设备的正确路径，而无需在其它设备中禁用多路径。

如果您之前使用 **find_multipaths** 参数创建了多路径设备，然后稍后将 参数设置为 **on**，您可能需要从 **/etc/multipath/wwids** 文件中删除您不想作为多路径设备创建的 WWID。以下示例显示了示例 **/etc/multipath/wwids** 文件。WWID 用斜杠(/)括起：

```
# Multipath wwids, Version : 1.0
# NOTE: This file is automatically maintained by multipath and multipathd.
# You should not need to edit this file in normal circumstances.
#
# Valid WWIDs:
/3600d0230000000000e13955cc3757802/
/3600d0230000000000e13955cc3757801/
/3600d0230000000000e13955cc3757800/
/3600d02300069c9ce09d41c31f29d4c00/
/SWINSYS  SF2372        0E13955CC3757802/
/3600d0230000000000e13955cc3757803/
```

除了 **on** 和 **off** 之外，您还可以将 **find_multipaths** 设置为以下值：

strict

多路径永远不会接受之前没有多路径的路径，因此不在 **/etc/multipath/wwids** 文件中。

smart

多路径会在出现时立即接受 **udev** 中的非禁用设备。如果 **multipathd** 没有在使用 **find_multipaths_timeout** 参数设置的超时中创建设备，它将在该设备中释放其声明。

find_multipaths 的内置默认值为 **off**。但是，**mpathconf** 创建的默认 **multipath.conf** 文件会将 **find_multipaths** 的值设置为 **on**。

当 **find_multipaths** 参数设置为 **on** 时，仅在带有您不想使用多路径的设备中禁用多路径。因此，通常不需要在设备中禁用多路径。

如果您将之前创建的多路径设备添加到 **黑名单** 中，通过使用 **-w** 选项从 **/etc/multipath/wwids** 文件中删除该设备的 WWID 有助于避免与其他程序出现问题。例如，要从 **/etc/multipath/wwids** 文件中删除 WWID 为**3600d0230000000000e13954ed5f89300** 的设备 **/dev/sdb**。

- 使用设备名称删除多路径设备。

```
# multipath -w /dev/sdb
wwid '3600d0230000000000e13954ed5f89300' removed
```

- 使用设备的 WWID 删除多路径设备。

```
# multipath -w 3600d0230000000000e13954ed5f89300
wwid '3600d0230000000000e13954ed5f89300' removed
```

您也可以使用 **-W** 选项来更新 **/etc/multipath/wwids** 文件。这会将 **/etc/multipath/wwids** 文件重置为仅包含当前多路径设备的 WWID。要重置文件，请运行以下命令：

```
# multipath -W
successfully reset wwids
```

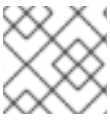
其他资源

- 您系统上的 **multipath.conf (5)** 手册页

6.2. 在某些设备中禁用多路径的条件

您可以根据以下标准在设备中禁用多路径：

- WWID
- 设备名称
- 设备类型
- 属性
- 协议



注意

默认情况下，禁用了各种设备类型，即使您注释掉了配置文件的初始 **黑名单** 部分。

对于每个设备，DM 多路径会按照以下顺序评估这些条件：

1. **属性**
2. **devnode**
3. **device**
4. **protocol**
5. **wwid**

如果某个设备被任何上述条件所禁用，DM 多路径会将它排除在 **multipathd** 处理之外，不会评估后续标准。对于每个条件，如果设备同时匹配，则异常列表优先于禁用的设备列表。

其他资源

- 为禁用多路径的设备添加例外

6.3. 使用 WWID 禁用多路径

您可以通过其全局识别(WWID)禁用独立设备上的多路径。

流程

1. 查找设备的 WWID：

```
# multipathd show paths raw format "%d %w" | grep sdb
sdb 3600508b4001080520001e00011700000
```

2. 使用 **wwid** 条目禁用 **/etc/multipath.conf** 配置文件中的设备。
以下示例显示了 DM 多路径配置文件中禁用 WWID 为 **3600508b4001080520001e00011700000** 的设备的行：

```
blacklist {
        wwid 3600508b4001080520001e00011700000
}
```

3. 通过运行以下命令之一修改多路径配置文件后，验证 **/etc/multipath.conf** 文件：

   - 要显示任何配置错误，请运行：

```
# multipath -t > /dev/null
```

   - 要显示使用添加的更改显示新配置，请运行：

```
# multipath -t
```

4. 重新载入 **/etc/multipath.conf** 文件并重新配置 **multipathd** 守护进程以使更改生效：

```
# service multipathd reload
```

6.4. 使用设备名称禁用多路径

您可以使用设备名称在设备类型中禁用多路径，以便 DM 多路径不会将其分组到多路径设备中。

流程

1. 显示设备信息：

```
# udevadm info --query=all -n /dev/mapper/sd*
```

2. 使用 **devnode** 条目禁用 **/etc/multipath.conf** 配置文件中的设备。

下面的例子显示，DM 多路径配置文件中禁用所有 SCSI 设备的行，因为它也禁用所有 **sd*** 设备：

```
blacklist {
        devnode "^sd[a-z]"
}
```

您可以使用 **devnode** 条目禁用单个设备，而不是禁用特定类型的所有设备。但不建议这样做，因为除非由 **udev** 规则静态映射，否则无法保证重启后特定设备的名称相同。例如：重启后，设备名称可以从 **/dev/sda** 改为 **/dev/sdb**。

默认情况下，DM 多路径会禁用所有不是 SCSI、NVMe 或者 DASD 的设备，使用以下 **devnode** 条目：

```
blacklist {
        devnode "!^(sd[a-z]|dasd[a-z]|nvme[0-9])"
}
```

这个条目禁用的设备通常不支持 DM 多路径。

3. 通过运行以下命令之一修改多路径配置文件后，验证 **/etc/multipath.conf** 文件：

   - 要显示任何配置错误，请运行：

     ```
     # multipath -t > /dev/null
     ```

   - 要显示使用添加的更改显示新配置，请运行：

     ```
     # multipath -t
     ```

4. 重新载入 **/etc/multipath.conf** 文件并重新配置 **multipathd** 守护进程以使更改生效：

   ```
   # service multipathd reload
   ```

其他资源

- 为禁用多路径的设备添加例外

6.5. 根据设备类型禁用多路径

您可以使用 **device** 部分在设备中禁用多路径。

流程

1. 显示设备类型：

   ```
   # multipathd show paths raw format "%d %s" | grep sdb
   sdb HP,HSV210
   ```

2. 使用 **device** 部分，禁用 **/etc/multipath.conf** 配置文件中的设备。
   以下示例禁用所有 IBM DS4200 和 HP 设备的多路径：

   ```
   blacklist {
   ```

```
        device {
                vendor  "IBM"
                product "3S42"       #DS4200 Product 10
        }
        device {
                vendor  "HP"
                product ".*"
        }
}
```

3. 通过运行以下命令之一修改多路径配置文件后，验证 **/etc/multipath.conf** 文件：

   - 要显示任何配置错误，请运行：

     ```
     # multipath -t > /dev/null
     ```

   - 要显示使用添加的更改显示新配置，请运行：

     ```
     # multipath -t
     ```

4. 重新载入 **/etc/multipath.conf** 文件并重新配置 **multipathd** 守护进程以使更改生效：

   ```
   # service multipathd reload
   ```

6.6. 使用 UDEV 属性禁用多路径

您可以通过其 **udev** 属性参数禁用对设备的多路径。

流程

1. 显示设备的 **udev** 变量：

   ```
   # udevadm info --query=all -n /dev/sdb
   ```

2. 使用 **property** 参数禁用 **/etc/multipath.conf** 配置文件中的设备。此参数是一个正则表达式字符串，与设备的 **udev** 环境变量名称匹配。
   以下示例禁用了所有使用 **udev** 属性 **ID_ATA** 的设备上的多路径：

   ```
   blacklist {
         property "ID_ATA"
   }
   ```

3. 通过运行以下命令之一修改多路径配置文件后，验证 **/etc/multipath.conf** 文件：

   - 要显示任何配置错误，请运行：

     ```
     # multipath -t > /dev/null
     ```

   - 要显示使用添加的更改显示新配置，请运行：

     ```
     # multipath -t
     ```

4. 重新载入 **/etc/multipath.conf** 文件并重新配置 **multipathd** 守护进程以使更改生效：

```
# service multipathd reload
```

6.7. 使用设备协议禁用多路径

您可以使用 **设备** 协议禁用设备上的多路径。

流程

1. 可选：查看路径使用的协议：

```
# multipathd show paths raw format "%d %P" | grep sdb
sdb scsi:fcp
```

2. 使用 **protocol** 参数禁用 **/etc/multipath.conf** 配置文件中的设备。
protocol 参数使用正则表达式，并将具有匹配协议字符串的所有设备列入黑名单。例如，要在所有 nvme 设备中禁用多路径，请使用：

```
blacklist {
        protocol "nvme"
}
```

DM 多路径识别 **协议** 字符串，如 **scsi:fcp**,**scsi:spi**,**scsi:ssa**,**scsi:sbp**,**scsi:srp**,**scsi:iscsi**,**scsi:sas**,**scsi:adt**,**scsi:ata**,**scsi:unspec**,**ccw**,**cciss**,**nvme:pcie**,**nvme:** rdma**, nvme:** fc**, nvme:** tcp**, nvme:** loop**, nvme:apple-** nvme**, nvme:** unspec , **和** undef。

3. 通过运行以下命令之一修改多路径配置文件后，验证 **/etc/multipath.conf** 文件：

- 要显示任何配置错误，请运行：

```
# multipath -t > /dev/null
```

- 要显示使用添加的更改显示新配置，请运行：

```
# multipath -t
```

4. 重新载入 **/etc/multipath.conf** 文件并重新配置 **multipathd** 守护进程以使更改生效：

```
# service multipathd reload
```

6.8. 为禁用多路径的设备添加例外

您可以通过在当前禁用多路径的设备中添加例外来启用多路径。

先决条件

- 在某些设备中禁用多路径。

流程

1. 使用 **/etc/multipath.conf** 配置文件的 **blacklist_exceptions** 部分在设备上启用多路径。
当在配置文件的 **blacklist_exceptions** 部分中指定设备时，您必须使用与 **黑名单** 部分中指定的相同标准指定例外。例如： WWID 异常不适用于 **devnode** 条目禁用的设备，即使禁用的设备与该 WWID 关联。同样，**devnode** 例外仅适用于 **devnode** 条目，**device** 例外则仅适用于设备条目。

例 6.1. WWID 异常

如果您有大量设备，且希望仅多路径 WWID 为 **3600d0230000000000e13955cc3757803**，而不是逐一禁用每个设备，您可以禁用所有这些设备，然后禁用所有这些设备，然后通过将以下几行添加到 **/etc/multipath.conf** 文件中来只启用其中一个。

```
blacklist {
        wwid ".*"
}

blacklist_exceptions {
        wwid "3600d0230000000000e13955cc3757803"
}
```

另外，您可以使用感叹号(**!**)来反转 **黑名单** 条目，该条目会禁用除指定 WWID 之外的所有设备：

```
blacklist {
        wwid "!3600d0230000000000e13955cc3757803"
}
```

例 6.2. udev 属性的例外

property 参数的工作方式与其他 **blacklist_exception** 参数不同。**property** 参数的值必须与 **udev** 数据库中变量名称匹配。否则，设备会被禁用。使用这个参数，您可以在某些 SCSI 设备中禁用多路径，如 USB 盘和本地硬盘。

要只在可能进行多路径的 SCSI 设备中启用多路径，请将此参数设置为 **(SCSI_IDENT_|ID_WWN)**，如下例所示：

```
blacklist_exceptions {
        property "(SCSI_IDENT_|ID_WWN)"
}
```

2. 通过运行以下命令之一修改多路径配置文件后，验证 **/etc/multipath.conf** 文件：

- 要显示任何配置错误，请运行：

```
# multipath -t > /dev/null
```

- 要显示使用添加的更改显示新配置，请运行：

```
# multipath -t
```

3. 重新载入 **/etc/multipath.conf** 文件并重新配置 **multipathd** 守护进程以使更改生效：

```
# service multipathd reload
```

第 7 章 管理多路径卷

您可以使用 **multipath**,**dmsetup**, 和 **multipathd** 命令（由 DM 多路径提供）来管理多路径卷。

7.1. 重新定义在线多路径设备大小

您可以使用 **多** 板和 **resize2fs** 命令调整在线多路径设备和文件系统大小。

流程

1. 重新定义您的物理设备大小。
2. 查找逻辑单元号(LUN)的路径：

```
# multipath -l
```

3. 重新定义您的路径大小。对于 SCSI 设备，在 **rescan** 文件中写入 1 以便重新扫描 SCSI 驱动程序，如下命令所示：

```
# echo 1 > /sys/block/path_device/device/rescan
```

请确定您为每个路径设备运行这个命令。例如：如果您的路径设备是 **sda**、**sdb**、**sde**、sde 和 **sdf**，请运行以下命令：

```
# echo 1 > /sys/block/sda/device/rescan
# echo 1 > /sys/block/sdb/device/rescan
# echo 1 > /sys/block/sde/device/rescan
# echo 1 > /sys/block/sdf/device/rescan
```

4. 重新定义多路径设备大小：

```
# multipathd resize map multipath_device
```

5. 重新定义文件系统大小，假设没有使用 LVM 或 DOS 分区：

```
# resize2fs /dev/mapper/mpatha
```

7.2. 将 ROOT 文件系统从单一路径设备移动到多路径设备中

如果您在单一路径设备中安装了您的系统，且稍后向 root 文件系统添加了另一个路径，请将您的 root 文件系统移到多路径设备中。

先决条件

- 已安装 **device-mapper-multipath** 软件包。

流程

1. 创建 **/etc/multipath.conf** 配置文件：

```
# mpathconf --enable
```

2. 启用 **multipathd** 服务：

```
# systemctl enable multipathd.service
```

3. 如果 **find_multipaths** 配置参数没有设置为 **on**，请编辑 **/etc/multipath.conf** 文件的 **blacklist** 和 **blacklist_exceptions** 部分，如从多路径中阻止设备 中所述。

4. 将设备的 WWID 添加到 **/etc/multipath/wwids** 文件中：

```
# multipath -a /dev/sdb
wwid '3600d02300069c9ce09d41c4ac9c53200' added
```

使用 root 设备名称替换 */dev/sdb*。

5. 确认您的配置文件设置是否正确：

```
# multipath -d 3600d02300069c9ce09d41c4ac9c53200
: mpatha (3600d02300069c9ce09d41c4ac9c53200) undef 3PARdata,VV
size=446M features='1 queue_if_no_path' hwhandler='1 alua' wp=undef
`-+- policy='service-time 0' prio=50 status=undef
  `- 5:0:0:0 sdb 8:16 undef ready running
```

将 3600d02300069c9ce09d41c4ac9c53200 替换为交换设备的 WWID。

6. 使用 **multipath** 重建 **initramfs** 文件系统：

```
# dracut --force --add multipath
```

7. 关闭机器。

8. 引导机器。

9. 使其他路径对机器可见。

验证

- 运行以下命令，检查多路径设备是否已创建：

```
# multipath -l | grep 3600d02300069c9ce09d41c4ac9c53200
mpatha (3600d02300069c9ce09d41c4ac9c53200) dm-0 3PARdata,VV
```

7.3. 将 SWAP 文件系统从单一路径设备移动到多路径设备中

默认情况下将 swap 设备设定为逻辑卷。如果您在组成逻辑卷的物理卷中设置了多路径，则不需要将它们配置为多路径设备。如果您的 swap 设备不是 LVM 卷，且使用设备名称挂载，您可能需要编辑 **/etc/fstab** 文件以切换到适当的多路径设备名称。

流程

1. 创建 **/etc/multipath.conf** 配置文件：

```
# mpathconf --enable
```

2. 启用 **multipathd** 服务：

```
# systemctl enable multipathd.service
```

3. 如果 **find_multipaths** 配置参数没有设置为 **on**，请编辑 **/etc/multipath.conf** 文件的 **blacklist** 和 **blacklist_exceptions** 部分，如从多路径中阻止设备 中所述。

4. 将设备的 WWID 添加到 **/etc/multipath/wwids** 文件中：

```
# multipath -a /dev/sdb
wwid '3600d02300069c9ce09d41c4ac9c53200' added
```

使用 swap 设备名称替换 */dev/sdb*。

5. 确认您的配置文件设置是否正确：

```
# multipath -d 3600d02300069c9ce09d41c4ac9c53200
: mpatha (3600d02300069c9ce09d41c4ac9c53200) undef 3PARdata,VV
size=446M features='1 queue_if_no_path' hwhandler='1 alua' wp=undef
`-+- policy='service-time 0' prio=50 status=undef
  `- 5:0:0:0 sdb 8:16 undef ready running
```

将 3600d02300069c9ce09d41c4ac9c53200 替换为交换设备的 WWID。

6. 在 **/etc/multipath.conf** 文件中为交换设备设置别名：

```
multipaths {
    multipath {
        wwid WWID_of_swap_device
        alias swapdev
    }
}
```

7. 编辑 **/etc/fstab** 文件，并使用多路径设备替换到 root 设备的旧设备路径。
例如，如果您在 **/etc/fstab** 文件中有以下条目：

```
/dev/sdb2 swap                    swap    defaults        0 0
```

将条目改为以下内容：

```
/dev/mapper/swapdev swap          swap    defaults        0 0
```

8. 使用多路径重建 initramfs 文件系统：

```
# dracut --force --add multipath
```

9. 关闭机器。

10. 引导机器。

11. 使其他路径对机器可见。

验证

- 验证 swap 设备是否在多路径设备中：

```
# swapon -s

Filename                Type        Size Used    Priority
/dev/dm-3               partition   4169724 0    -2
```

文件名应与多路径交换设备匹配。

```
# readlink -f /dev/mapper/swapdev
/dev/dm-3
```

7.4. 为多路径设备确定设备映射器条目

您可以使用 **multipathd** 命令发现哪个设备映射器条目与多路径设备匹配。

流程

- 显示所有设备映射器设备：

```
# multipathd show maps format "%n %d"

name   sysfs
mpathd dm-4
mpathb dm-3
mpatha dm-2
mpathh dm-9
```

7.5. 管理 MULTIPATHD 守护进程

multipathd 命令可用于管理 **multipathd** 守护进程。

流程

- 查看 **multipathd show maps** 命令输出的默认格式：

```
# multipathd show maps
name sysfs uuid
mpathc dm-0 360a98000324669436c2b45666c567942
```

- 有些 **multipathd** 命令包括 **format** 选项，后跟通配符。使用以下命令显示可用通配符列表：

```
# multipathd show wildcards
multipath format wildcards:
%n  name
%w  uuid
%d  sysfs
...
```

- 显示 **multipathd** 监控的多路径设备。使用通配符指定显示的字段：

```
# multipathd show maps format "%n %w %d %s"
name   uuid                              sysfs vend/prod/rev
mpathc 360a98000324669436c2b45666c567942 dm-0  NETAPP,LUN
```

- 显示 **multipathd** 监控的路径。使用通配符指定显示的字段：

```
# multipathd show paths format "%n %w %d %s"
target WWNN        uuid                              dev vend/prod/rev
0x50001fe1500d2250 3600508b4001080520001e00011700000 sdb HP,HSV210
```

- 以原始格式显示数据：

```
# multipathd show maps raw format "%n %w %d %s"
mpathc 360a98000324669436c2b45666c567942 dm-0 NETAPP,LUN
```

在原始格式中，不会打印标头，且不会添加字段来与标头匹配。此输出更易于用来编写脚本。

其他资源

- 系统中 **multipathd**(8)手册页

第 8 章 删除存储设备

您可以从正在运行的系统中安全地删除存储设备，这有助于防止系统内存过载和数据丢失。不要删除系统中的存储设备：

- 空闲内存低于内存总量的 5%，每 100 个超过 10 个样本。
- 交换是活跃的（在 **vmstat** 命令的输出中非零的 **si** 和 **so** 列）。

先决条件

- 在删除存储设备前，请确保在 I/O 刷新过程中由于系统内存负载增加而有足够的可用内存。使用以下命令查看系统的当前内存负载和可用内存：

```
# vmstat 1 100
# free
```

8.1. 安全删除存储设备

从正在运行的系统中安全地删除存储设备需要顶级的方法。从顶层（通常是应用程序或文件系统）开始，并在底层（即物理设备）上工作。

您可以通过多种方式使用存储设备，它们可以在物理设备之上有不同的虚拟配置。例如：您可以将设备的多个实例分组到多路径设备中，使其成为 RAID 的一部分，或者您可以将其成为 LVM 组的一部分。此外，设备可以通过文件系统访问，或者可以直接访问设备，如"原始"设备。

使用 top-to-bottom 方法时，您必须确保：

- 要删除的设备没有被使用
- 对该设备的所有待处理的 I/O 都会被清除
- 操作系统无法引用存储设备

8.2. 删除块设备和相关元数据

要从正在运行的系统中安全地删除块设备，以帮助防止系统内存过载和数据丢失，您需要首先从它们中删除元数据。从文件系统开始，处理堆栈中的每个层，然后继续到磁盘。这些操作可防止将您的系统处于不一致的状态。

根据您要删除的设备类型，使用可能不同的特定命令：

- **lvremove**、**vgremove** 和 **pvremove** 特定于 LVM。
- 对于软件 RAID，请运行 **mdadm** 以删除该阵列。如需更多信息，请参阅管理 RAID。
- 对于使用 LUKS 加密的块设备，有特定的额外步骤。以下流程不适用于使用 LUKS 加密的块设备。如需更多信息，请参阅使用 LUKS 加密块设备。

警告

重新扫描 SCSI 总线或执行更改操作系统状态的其他操作，而无需遵循这个流程，因为 I/O 超时、设备被意外删除或数据丢失。

先决条件

- 您有一个现有的块设备堆栈，其中包含文件系统、逻辑卷和卷组。
- 您确定没有其他应用程序或服务正在使用您要删除的设备。
- 您从您要删除的设备备份了数据。
- 可选：如果要删除多路径设备，且您无法访问其路径设备，请运行以下命令来禁用多路径设备的队列：

```
# multipathd disablequeueing map multipath-device
```

这可让设备的 I/O 失败，允许使用该设备的应用程序关闭。

注意

一次删除其元数据层的设备，确保不会在磁盘上保留过时的签名。

流程

1. 卸载文件系统：

```
# umount /mnt/mount-point
```

2. 删除文件系统：

```
# wipefs -a /dev/vg0/myvol
```

如果您已在 **/etc/fstab** 文件中添加了一个条目，以便在文件系统和挂载点之间建立持久关联，请在此时编辑 **/etc/fstab** 以删除该条目。

根据您要删除的设备类型，继续执行以下步骤：

3. 删除包含文件系统的逻辑卷(LV)：

```
# lvremove vg0/myvol
```

4. 如果卷组中没有剩余的其他逻辑卷(VG)，您可以安全地删除包含该设备的 VG：

```
# vgremove vg0
```

5. 从 PV 设备中删除物理卷(PV)元数据：

```
# pvremove /dev/sdc1
```

```
# wipefs -a /dev/sdc1
```

6. 删除包含 PV 的分区：

```
# parted /dev/sdc rm 1
```

7. 如果要完全擦除该设备，请删除分区表：

```
# wipefs -a /dev/sdc
```

8. 只有在您要物理删除该设备时才执行以下步骤：

- 如果您要删除多路径设备，请执行以下命令：

a. 查看该设备的所有路径：

```
# multipath -l
```

稍后需要这个命令的输出。

b. 清除 I/O 并删除多路径设备：

```
# multipath -f multipath-device
```

- 如果该设备没有配置为多路径设备，或者设备配置为多路径设备，并且您之前将 I/O 传递给单个路径，请将任何未完成的 I/O 刷新到所有使用的设备路径：

```
# blockdev --flushbufs device
```

对于直接访问的设备非常重要，**umount** 或 **vgreduce** 命令不会清除 I/O。

- 如果您要删除 SCSI 设备，请执行以下命令：

a. 删除对基于路径的设备名称的任何引用，如 ***/dev/sd***、***/dev/disk/by-path*** 或 **major:minor** number（在系统上的应用程序、脚本或工具中）。这样可保证以后添加的不同设备不会为当前的设备错误。

b. 从 SCSI 子系统中删除该设备的每个路径：

```
# echo 1 > /sys/block/device-name/device/delete
```

此处，如果设备之前用作多路径设备，则 ***device-name*** 从 **multipath -l** 命令的输出中检索。

9. 从正在运行的系统中删除物理设备。请注意，当您删除此设备时，I/O 到其它设备不会停止。

验证

- 验证您要删除的设备是否没有显示 **lsblk** 命令的输出。以下是一个输出示例：

```
# lsblk
```

```
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0    5G  0 disk
sr0     11:0    1 1024M  0 rom
vda    252:0    0   10G  0 disk
|-vda1 252:1    0    1M  0 part
|-vda2 252:2    0  100M  0 part /boot/efi
`-vda3 252:3    0  9.9G  0 part /
```

其他资源

- **multipath (8)**, **pvremove (8)**, **vgremove (8)**, **lvremove (8)**, **wipefs (8)**, **parted (8)**, **blockdev (8)**, 和 **umount (8)** man page

第 9 章 DM 多路径故障排除

如果您在进行多路径配置时遇到问题，您可以检查这些问题。以下问题可能会导致多路径配置缓慢或无法正常工作：

多路径守护进程没有运行

如果您在实施多路径配置时遇到问题，请确保 **multipathd** 守护进程正在运行，如 配置 DM 多路径 中所述。**multipathd** 守护进程必须正在运行才能使用多路径设备。

queue_if_no_path 功能的问题

如果使用 **"1 queue_if_no_path"** 选项配置多路径设备，那么在恢复一个或多个路径前，任何问题 I/O 的进程都会挂起。

9.1. 对 QUEUE_IF_NO_PATH 功能的问题进行故障排除

如果使用 **"1 queue_if_no_path"** 选项配置多路径设备，那么在恢复一个或多个路径前，任何问题 I/O 的进程都会挂起。要避免这种情况，请在 **/etc/multipath.conf** 文件中设置 **no_path_retry N** 参数，其中 *N* 是系统应该重试路径的次数。

要在使用 **特性 "1 queue_if_no_path"** 选项时没有所描述的问题，您可以在运行时为特定 LUN 禁用队列策略，因此所有路径都不可用。

流程

1. 禁用队列：
   - 对于特定设备：

     ```
     # multipathd disablequeueing map device
     ```

   - 对于所有设备：

     ```
     # multipathd disablequeueing maps
     ```

     禁用队列后，它将保持禁用状态，直到您重启或重新加载 **multipathd**。
2. 将队列重置为以前的值：
   - 对于特定设备：

     ```
     # multipathd restorequeueing map device
     ```

   - 对于所有设备：

     ```
     # multipathd restorequeueing maps
     ```

9.2. 使用 MULTIPATHD 互动控制台进行故障排除

multipathd -k 命令是 **multipathd** 守护进程的互动接口。执行此命令将进入互动的多路径控制台。执行此命令后，您可以输入 **help** 来获取可用命令列表，**Ctrl+D** 退出。

使用 **multipathd** 互动控制台来对与您的系统相关的问题进行故障排除。

流程

1. 在退出控制台前显示多路径配置，包括默认值：

```
# multipathd -k
multipathd> show config
multipathd> Ctrl+D
```

2. 确定多路径获取对 **multipath.conf** 文件的所有更改：

```
# multipathd -k
multipathd> reconfigure
multipathd> Ctrl+D
```

3. 确保路径检查程序正常工作：

```
# multipathd -k
multipathd> show paths
multipathd> Ctrl+D
```

4. 您也可以直接从命令行运行单个 **multipathd** 交互式命令，而无需启动交互式控制台。例如，要检查多路径是否获取了对 **multipath.conf** 文件的所有更改，请运行以下命令：

```
# multipathd reconfigure
```

第 10 章 使用 EH_DEADLINE 配置存储错误恢复的最大时间

您可以配置最大允许的时间来恢复失败的 SCSI 设备。这个配置保证了 I/O 响应时间，即使存储硬件因为失败而变得无响应。

10.1. EH_DEADLINE 参数

SCSI 错误处理(EH)机制尝试在失败的 SCSI 设备上执行错误恢复。SCSI 主机对象 **eh_deadline** 参数允许您配置恢复的最大时间。配置的时间过期后，SCSI EH 会停止并重置整个主机总线适配器(HBA)。

使用 **eh_deadline** 可以缩短时间：

- 关闭失败的路径，
- 切换路径，或者
- 禁用 RAID 分片。

警告

当 **eh_deadline** 过期时，SCSI EH 会重置 HBA，这会影响那个 HBA 中的所有目标路径，而不仅仅是故障。如果由于其他原因无法使用冗余路径，则可能会出现 I/O 错误。如果在所有目标上都配置了多路径，请只启用 **eh_deadline**。另外，如果您的多路径设备没有完全冗余，您应该验证 **no_path_retry** 是否设置为足够大，以允许路径恢复。

eh_deadline 参数的值以秒为单位指定。默认设置为 **off**，它会禁用时间限制并允许进行所有错误恢复。

eh_deadline 很有用的情况

在大多数情况下，您不需要启用 **eh_deadline**。在某些特定场景中，使用 **eh_deadline** 非常有用。例如，如果在光纤通道(FC)交换机和目标端口之间发生链接丢失，且 HBA 没有收到 Registered State Change Notifications(RSCN)。在这种情况下，I/O 请求和错误恢复命令会超时，而不是遇到错误。在这个环境中设置 **eh_deadline** 会针对恢复时间设置上限。这可让失败的 I/O 在由 DM 多路径的另一个可用路径中检索。

在以下条件下，**eh_deadline** 参数不提供额外的好处，因为 I/O 和错误恢复命令会立即失败，这会导致 DM 多路径重试：

- 如果启用了 RSCN
- 如果 HBA 没有注册链接不可用

10.2. 设置 EH_DEADLINE 参数

您可以配置 **eh_deadline** 参数的值来限制最大 SCSI 恢复时间。

流程

- 您可以使用以下方法之一配置 **eh_deadline** ：

- **multipath.conf** 文件的 **defaults** 部分
  在 **multipath.conf** 文件的 defaults 部分，将 **eh_deadline** 参数设置为所需的秒数：

  ```
  # eh_deadline 300
  ```

  注意

  在 RHEL 8.4 中，使用 **multipath.conf** 文件的 defaults 部分设置 **eh_deadline** 参数是首选的方法。

  要使用此方法关闭 **eh_deadline** 参数，请将 **eh_deadline** 设置为 **off**。

- **sysfs**
  将秒数写入 **/sys/class/scsi_host/host<host-number>/eh_deadline** 文件中。例如，要在 SCSI 主机 6 上通过 **sysfs** 设置 **eh_deadline** 参数：

  ```
  # echo 300 > /sys/class/scsi_host/host6/eh_deadline
  ```

  要使用此方法关闭 **eh_deadline** 参数，请使用 echo **off**。

- 内核参数
  使用 **scsi_mod.eh_deadline** 内核参数为所有 SCSI HBA 设置默认值。

  ```
  # echo 300 > /sys/module/scsi_mod/parameters/eh_deadline
  ```

  要使用此方法关闭 **eh_deadline** 参数，请使用 echo **-1**。

其他资源

- 如何使用 udev 规则（红帽知识库）永久设置 eh_deadline 和 eh_timeout