Red Hat Enterprise Linux 9

Device Mapper Multipath の設定

Device Mapper Multipath 機能の設定および管理

Last Updated: 2024-06-26

Red Hat Enterprise Linux 9 Device Mapper Multipath の設定

Device Mapper Multipath 機能の設定および管理

法律上の通知

概要

Device Mapper のマルチパス (DM Multipath) を使用すると、サーバーノードとストレージアレイとの間の複数の I/O パスを 1 つのデバイスに設定できます。これらの I/O パスは、個別のケーブル、スイッチ、コントローラーを含むことができる物理的なストレージエリアネットワーク (SAN) 接続です。 マルチパスは I/O パスを集約し、集約されたパスで設定される新しいデバイスを作成します。

目次

RED HAT ドキュメントへのフィードバック (英語のみ)

Red Hat ドキュメントに関するご意見やご感想をお寄せください。また、改善点があればお知らせください。

Jira からのフィードバック送信 (アカウントが必要)

1. Jira の Web サイトにログインします。
2. 上部のナビゲーションバーで **Create** をクリックします。
3. **Summary** フィールドにわかりやすいタイトルを入力します。
4. **Description** フィールドに、ドキュメントの改善に関するご意見を記入してください。ドキュメントの該当部分へのリンクも追加してください。
5. ダイアログの下部にある **Create** をクリックします。

第1章 デバイスマッパーマルチパスの概要

DM Multipath は以下を提供します。

冗長性

DM Multipath は、アクティブ/パッシブ設定でフェイルオーバーを提供できます。アクティブ/パッシブ設定では、常にパスのサブセットのみが I/O に使用されます。ケーブル、スイッチ、コントローラーなどの I/O パスの要素に障害が発生した場合、DM Multipath は代替パスに切り替わります。

注記

パスの数はセットアップによって異なります。通常、DM Multipath 設定にはストレージへのパスが 2、4、または 8 個ありますが、これは一般的な設定であり、他の数となる可能性もあります。

パフォーマンスの向上

DM Multipath は、アクティブ/アクティブモードで設定できます。このモードでは、I/O はラウンドロビン方式でパスに分散されます。一部の設定では、DM Multipath は I/O パスの負荷を検出し、負荷を動的に再調整できます。

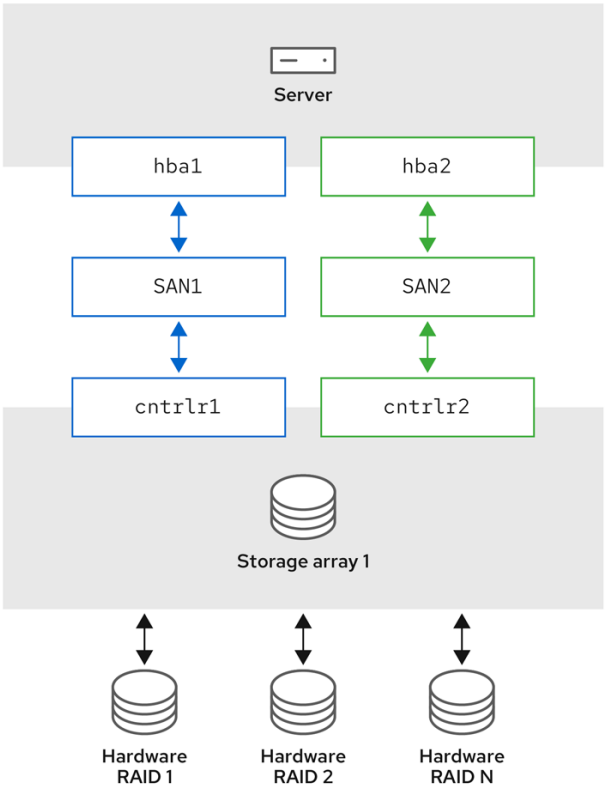
1.1. 1 つの RAID デバイスを使用したアクティブ/パッシブのマルチパス設定

この設定では、サーバー上に 2 つのホストバスアダプター (HBA)、2 つの SAN スイッチ、および 2 つの RAID コントローラーがあります。この設定では、次のような障害が発生する可能性があります。

- HBA の障害
- ファイバーチャネルケーブルの障害
- SAN スイッチの障害
- アレイコントローラーポートの障害

DM Multipath が設定されると、上記のポイントのいずれかで障害が発生すると、DM Multipath は別の I/O パスに切り替わります。以下の図は、サーバーから RAID デバイスへの 2 つの I/O パスを使用した設定を説明します。ここでは、**hba1**、**SAN1**、および **cntrlr1** を通る 1 つの I/O パス
と、**hba2**、**SAN2**、および **cntrlr2** を通る別の I/O パスがあります。

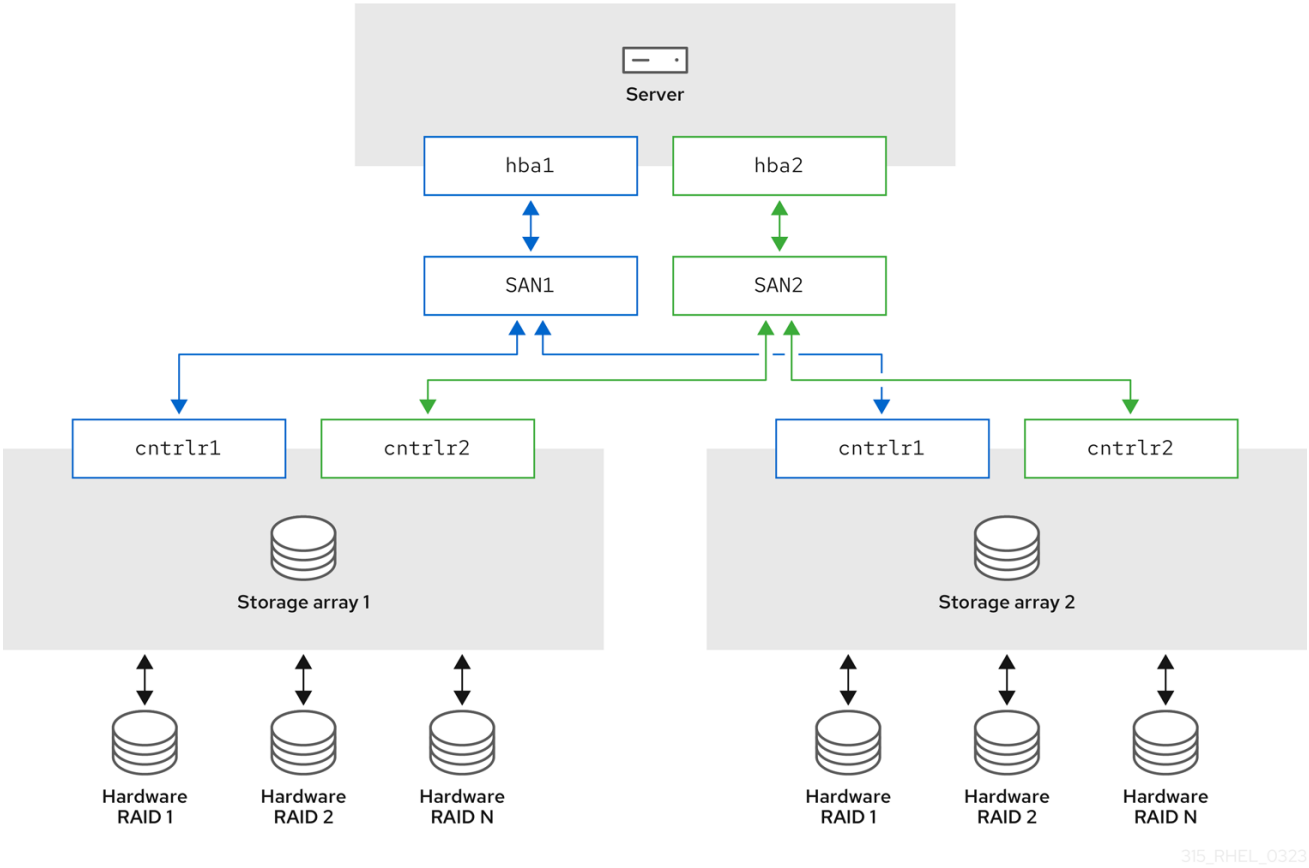
図1.1 1つの RAID デバイスを使用したアクティブ/パッシブのマルチパス設定



1.2. 2 つの RAID デバイスを使用したアクティブ/パッシブのマルチパス設定

この設定では、サーバー上に 2 つの HBA、2 つの SAN スイッチ、およびそれぞれ 2 つの RAID コントローラーを備えた 2 つの RAID デバイスがあります。DM Multipath が設定されている場合には、どちらかの RAID デバイスへの I/O パスのどこかのポイントで障害が発生すると、DM Multipath はそのデバイスの別 I/O パスに切り替わります。以下の図は、各 RAID デバイスへの 2 つの I/O パスを使用した設定を説明します。ここでは、各 RAID デバイスへの 2 つの I/O パスがあります。

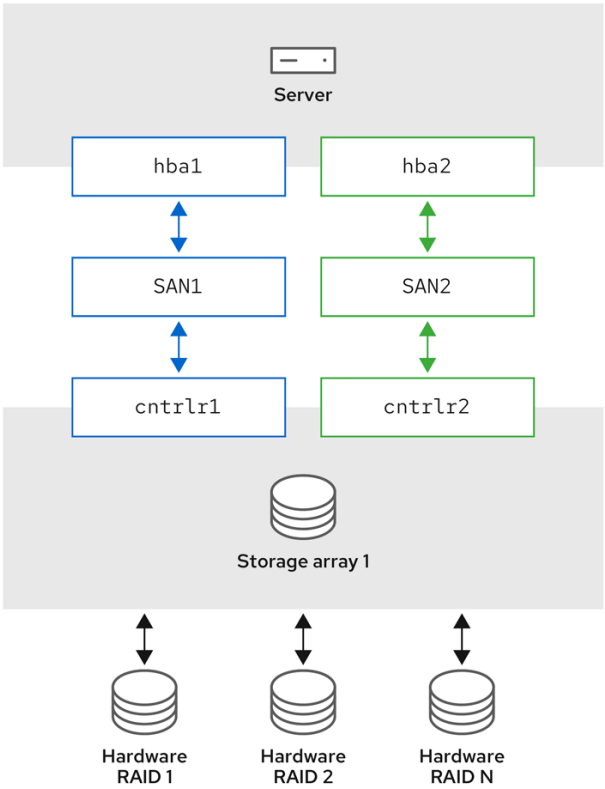
図1.2 2 つの RAID デバイスを使用したアクティブ/パッシブのマルチパス設定



1.3. 1 つの RAID デバイスを使用したアクティブ/アクティブのマルチパス設定

この設定では、サーバー上に HBA、2 つの SAN スイッチ、および 2 つの RAID コントローラーがあります。以下の図は、サーバーからストレージデバイスへの 2 つの I/O パスを使用した設定について説明します。ここで、I/O は、これら 2 つのパスに分散できます。

図1.3 1つの RAID デバイスを使用したアクティブ/アクティブのマルチパス設定



1.4. DM MULTIPATH コンポーネント

以下の表は、DM Multipath コンポーネントを示しています。

表1.1 DM Multipath のコンポーネント

コンポーネント	説明
dm_multipath カーネルモジュール	I/O を再ルーティングし、パスとパスグループのフェイルオーバーに対応します。
mpathconf ユーティリティー	デバイスマッパーマルチパスを設定して有効にします。
multipath コマンド	マルチパスデバイスをリスト表示して設定します。これは、ブロックデバイスが追加されるたびに **udev** により実行され、デバイスがマルチパスデバイスの一部であるかどうかを判断します。
multipathd デーモン	マルチパスデバイスを自動的に作成および削除し、パスを監視します。パスが失敗して戻ってくると、マルチパスデバイスが更新される場合があります。マルチパスデバイスへのインタラクティブな変更を許可します。**/etc/multipath.conf** ファイルに変更がある場合は、サービスを再読み込みします。

kpartx コマンド	デバイス上のパーティションのデバイスマッパーデバイスを作成します。このコマンドは、マルチパスデバイスが作成され、その上にパーティションデバイスが作成されると、**udev** により自動的に実行されます。**kpartx** コマンドは独自のパッケージで提供されますが、**device-mapper-multipath** パッケージはそれに依存しています。
mpathpersist	マルチパスデバイスに **SCSI-3** 永続予約を設定します。このコマンドは、**sg_persist** が SCSI デバイスに対して行う方法と似ていますが、マルチパスデバイスのすべてのパスで永続的な予約の設定を処理します。**multipathd** と調整して、後で追加されるパスに予約が正しく設定されるようにします。この機能を使用するには、**reservation_key** 属性を **/etc/multipath.conf** ファイルで定義する必要があります。定義しないと、**multipathd** デーモンは、新しく検出されたパスまたは復元されたパスの永続的な予約を確認しません。

1.5. MULTIPATH コマンド

multipath コマンドは、デバイスの複数のパスを検出および結合するために使用されます。マルチパス化されたデバイスの管理に使用できるさまざまなオプションを提供します。

次の表に、役立つと思われる **multipath** コマンドのいくつかのオプションを示します。

表1.2 便利な multipath コマンドのオプション

オプション	説明
-l	**sysfs** から収集した現在のマルチパストポロジーとデバイスマッパーを表示します。
-ll	**sysfs** から収集した現在のマルチパストポロジー、デバイスマッパー、使用可能な全システムコンポーネントを表示します。
-f device	指定したマルチパスデバイスを削除します。
-F	不要なマルチパスデバイスをすべて削除します。
-w device	**wwids** ファイルから指定デバイスの **wwid** を削除します。
-W	現在のマルチパスデバイスのみが含まれるように、**wwids** ファイルをリセットします。

オプション	説明
-r	マルチパスデバイスを強制的にリロードします。

1.6. マルチパストポロジーの表示

パスを効果的に監視したり、マルチパスの問題をトラブルシューティングしたり、マルチパス設定が正しく設定されているかどうかを確認したりするために、マルチパストポロジーを表示できます。

手順

1. マルチパスデバイストポロジーを表示します。

```
# 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
```

出力は 3 つの部分に分けることができます。各部分には、次のグループの情報が表示されます。

- マルチパスデバイス情報:
  - **mpatha (3600d0230000000000e13954ed5f89300)**: エイリアス (エイリアスと異なる場合は wwid)
  - **dm-4**: dm デバイス名
  - **WINSYS,SF2372**: ベンダー、製品
  - **size=233G**: サイズ
  - **features='1 queue_if_no_path'**: 機能
  - **hwhandler='0'**: ハードウェアハンドラー
  - **wp=rw** : 書き込み権限
- パスグループ情報:
  - **policy='service-time 0'**: スケジュールポリシー
  - **prio=1**: パスグループの優先度
  - **status=active**: パスグループステータス
- パス情報:
  - **6:0:0:0**: ホスト:チャネル:id:lun
  - **sdf**: devnode
  - **8:80** : メジャー:マイナー番号

- **active**: dm ステータス
- **ready**: パスステータス
- **running**: オンライン状態
  dm、パス、オンラインのステータスの詳細は、パスステータス を参照してください。

マルチパスデバイスのリスト表示、作成、またはリロードに使用される他のマルチパスコマンドでも、デバイストポロジーは表示されます。ただし、一部の情報が不明で、出力に **undef** と表示される場合があります。これは通常の動作です。正しい状態を表示するには、**multipath -ll** コマンドを使用します。

注記

マルチパスデバイスの作成など、特定の場合には、マルチパストポロジーに、アクションが実行されたかどうかを示すパラメーターが表示されます。たとえば、次のコマンド出力は、マルチパスデバイスが作成されたことを表す **create:** パラメーターを示しています。

```
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
```

1.7. パスステータス

パスの状態は、**/etc/multipath.conf** ファイルに定義されているポーリング間隔に応じ、**multipathd** デーモンによって定期的に更新されます。カーネルの観点から見ると、**dm** ステータスはパスステータスと似ています。**dm** ステータスは、パスチェッカーが完了するまで現在のステータスを保持します。

パスステータス

ready、ghost
: パスが有効であり、I/O の準備ができています。

faulty、shaky
: パスが無効です。

i/o pending
: チェッカーがアクティブにこのパスを確認し、ステータスはまもなく更新されます。

i/o timeout
: チェッカーがタイムアウト期間経過前に **success/failure** を返しませんでした。これは **faulty** と同じように処理されます。

removed
: システムからパスが削除され、マルチパスデバイスからまもなく削除されます。これは **faulty** と同じように処理されます。

wild
: 内部エラーまたは設定の問題によって、**multipathd** がパスチェッカーを実行できませんでした。これは、マルチパスがパス上の多くのアクションをスキップすることを除いて、**failed** と同じように処理されます。

unchecked
: パスが今検出された、割り当てられたパスチェッカーがない、パスチェッカーにエラーが発生したなどの理由で、このパスでパスチェッカーが実行していません。これは **wild** と同じように処理されます。

delayed

パスチェッカーはパスが有効であると返しますが、マルチパスがパスの回復を遅らせています。このパスで最近、複数回障害が発生しており、パスを遅延するようにマルチパスが設定されているためです。これは **faulty** と同じように処理されます。

dm ステータス

Active

ready および **ghost** のパスステータスにマップされます。

Failed

同等の **dm** 状態を持たない **i/o pending** を除く、他のすべてのパスステータスにマップされます。

オンラインステータス

Running

デバイスが有効です。

オフライン

デバイスが無効です。

1.8. 関連情報

- man ページの **multipath(8)** および **multipathd(8)**
- **/etc/multipath.conf** ファイル

第2章 マルチパスデバイス

DM Multipath は、基礎となるデバイスの上に 1 つのマルチパスデバイスを作成することにより、I/O パスを論理的に整理する方法を提供します。DM Multipath を使用しない場合は、I/O パスが同じサーバーノードを同じストレージコントローラーに接続している場合でも、システムはサーバーノードからストレージコントローラーへの各パスを個別のデバイスとして扱います。

2.1. マルチパスデバイス識別子

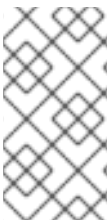
新しいデバイスが DM Multipath の制御下にある場合に、これらのデバイスは **/dev/mapper/** ディレクトリーおよび **/dev/** ディレクトリーに作成されます。

注記

/dev/dm-X という形式のデバイスは内部使用専用であるため、管理者が直接使用するものではありません。

以下は、マルチパスデバイス名について説明しています。

- **user_friendly_names** 設定オプションが **no** に設定されている場合は、マルチパスデバイスの名前が World Wide Identifier (WWID) に設定されます。デフォルトでは、マルチパスデバイスの名前はその WWID に設定されます。デバイス名は **/dev/mapper/WWID** になります。また、**/dev/dm-X** という名前の **/dev/** ディレクトリーにも作成されます。
- または、**/etc/multipath.conf** ファイルで、**user_friendly_names** オプションを **yes** に設定できます。これにより、**multipath** セクションの **alias** が、**mpathN** 形式のノード固有の名前に設定されます。デバイス名は、**/dev/mapper/mpathN** および **/dev/dm-X** になります。ただし、マルチパスデバイスを使用するすべてのノードでデバイス名が同じであるとは限りません。同様に、**/etc/multipath.conf** ファイルで **alias** オプションを設定した場合は、クラスター内のすべてのノードで自動的に名前が一致しません。



注記

LVM を使用してマルチパスデバイスから論理デバイスを作成する場合は、これにより問題が発生することはありません。すべてのノードでマルチパスデバイス名の一貫性を保つために、Red Hat は、**user_friendly_names** オプションを無効にすることを推奨します。

たとえば、ゾーンに分けられていない 1 つの FC スイッチにより、2 つのポートを持つストレージコントローラーに接続された 2 つの HBA を持つノードは、**/dev/sda**、**/dev/sdb**、**/dev/sdc**、および **/dev/sdd** の 4 つのデバイスを認識します。DM Multipath は、マルチパス設定に従って、I/O を基本となるこれらの 4 つのデバイスにルーティングしなおす一意の WWID を持つシングルデバイスを作成します。

user_friendly_names オプションおよび **alias** オプションの他に、マルチパスデバイスには他の属性もあります。**/etc/multipath.conf** ファイルの **multipaths** セクションに、デバイスのエントリーを作成することにより、特定のマルチパスデバイスのこれらの属性を変更できます。

関連情報

- man ページの **multipath(8)** および **multipath.conf(8)**
- **/etc/multipath.conf** ファイル

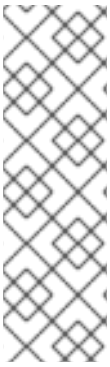
- DM Multipath コンポーネント

2.2. 論理ボリューム内のマルチパスデバイス

マルチパスデバイスを作成したら、論理ボリュームマネージャー (LVM) 物理ボリュームを作成する際に物理デバイス名を使用するのと同様に、マルチパスデバイス名を使用できます。たとえば、**/dev/mapper/mpatha** がマルチパスデバイスの名前である場合、**pvcreate /dev/mapper/mpatha** コマンドは、**/dev/mapper/mpatha** を物理ボリュームとしてマークします。

他の LVM 物理デバイスを使用するのと同じように、LVM ボリュームグループを作成するときに、作成された LVM 物理デバイスを使用できます。

/etc/lvm/lvm.conf ファイル内のすべての **sd** デバイスをフィルタリングするには、そのファイルの **devices** セクションに **filter = ["r/block/", "r/disk/", "r/sd./", "a/./"]** を追加します。



注記

パーティションを設定したデバイス全体に LVM 物理ボリュームを作成しようとすると、**pvcreate** コマンドは失敗します。Anaconda および Kickstart のインストールプログラムは、すべてのブロックデバイスに特に指定しない限り、空のパーティションテーブルを作成します。パーティションを作成する代わりにデバイス全体を使用する場合は、デバイスから既存のパーティションを削除します。**kpartx -d** デバイスコマンドと **fdisk** ユーティリティーを使用して、既存のパーティションを削除できます。システムに 2Tb を超えるブロックデバイスがある場合は、**parted** ユーティリティーを使用してパーティションを削除します。

active/passive マルチパスアレイを基礎となる物理デバイスとして使用する LVM 論理ボリュームを作成する場合は、必要に応じて、**/etc/lvm/lvm.conf** ファイルにフィルターを追加して、マルチパスデバイスの基礎となるディスクを除外できます。これは、I/O の受信時にアレイがアクティブパスをパッシブパスに自動的に変更する場合に、このようなデバイスにフィルターが設定されていないと、LVM がパッシブパスをスキャンするたびにマルチパスがフェイルオーバーおよびフェイルバックするためです。

カーネルは、使用する正しいハードウェアハンドラーを自動的に検出してアクティブ/パッシブ状態を変更します。状態を変更するために介入を必要とするアクティブ/パッシブパスの場合、マルチパスは自動的にこのハードウェアハンドラーを使用して、必要に応じて介入します。カーネルが使用する正しいハードウェアハンドラーを自動的に検出しない場合は、multipath.conf ファイルで "hardware_handler" オプションを使用して、使用するハードウェアハンドラーを設定できます。パッシブパスをアクティブにするコマンドを必要とする **アクティブ/パッシブ** アレイでこの問題が発生すると、LVM が警告メッセージを出力します。

設定によっては、LVM が以下のいずれかのメッセージを出力することがあります。

- LUN の準備ができていません:

```
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
```

- 読み取りに失敗しました:

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

以下は、上記のエラーの理由です。

- マルチパスは、マシンにアクティブ/パッシブパスを提供するストレージデバイスでは設定されません。
- パスはマルチパスデバイスではなく、直接アクセスします。

関連情報

- **lvm.conf** の man ページ
- DM Multipath コンポーネント

第3章 DM MULTIPATH の設定

mpathconf ユーティリティーを使用して DM Multipath を設定できます。このユーティリティーは、次のシナリオに基づいて **/etc/multipath.conf** マルチパス設定ファイルを作成または編集します。

- **/etc/multipath.conf** ファイルがすでに存在する場合は、**mpathconf** ユーティリティーでファイルを編集します。
- **/etc/multipath.conf** ファイルが存在しない場合は、**mpathconf** ユーティリティーにより **/etc/multipath.conf** ファイルが新たに作成されます。

3.1. DEVICE-MAPPER-MULTIPATH パッケージの確認

システムに DM-Multipath をセットアップする前に、システムが最新であり、**device-mapper-multipath** パッケージがインストールされていることを確認してください。

手順

1. システムに **device-mapper-multipath** パッケージが含まれているか確認します。

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

   システムにパッケージが含まれていない場合は、次のように出力されます。

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

2. システムにパッケージが含まれていない場合は、次のコマンドを実行してパッケージをインストールします。

```
# dnf install device-mapper-multipath
```

3.2. DM MULTIPATH を使用した基本的なフェイルオーバー設定のセットアップ

基本的なフェイルオーバー設定用の DM Multipath をセットアップし、multipathd デーモンを起動する前に **/etc/multipath.conf** ファイルを編集できます。

前提条件

- 管理アクセスがある。

手順

1. マルチパス設定ファイルを有効にして初期化します。

```
# mpathconf --enable
```

2. オプション: **/etc/multipath.conf** ファイルを編集します。
   ほとんどのデフォルト設定はすでに設定されています。たとえば、**path_grouping_policy** は **failover** に設定されています。

3. オプション: マルチパスデバイスのデフォルトの命名形式は、**/dev/mapper/mpathn** 形式に設定されます。別の命名形式を使用する場合は、次のようにします。

   a. ユーザーフレンドリーな命名スキーム (mpath_n_) の代わりに、マルチパスデバイスの WWID を名前として使用するように DM Multipath を設定します。

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

   b. DM Multipath デーモンの設定をリロードします。

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

4. DM Multipath デーモンを起動します。

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

検証

- DM Multipath デーモンが問題なく実行されていることを確認します。

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

- マルチパスデバイスの命名形式を確認します。

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

3.3. マルチパスデバイスの作成時にローカルのディスクを無視

一部のマシンには内部ディスク用のローカル SCSI カードがあり、DM Multipath をこのようなデバイスで使用することは推奨されません。**find_multipaths** 設定パラメーターを **yes** に設定すれば、このようなデバイスでマルチパスを無効にする必要はありません。

find_multipaths 設定パラメーターを **yes** に設定しない場合は、以下の手順に従って DM Multipath 設定ファイルを変更すると、マルチパスの設定時にローカルのディスクを無視できます。

手順

1. デバイスのモデル、パス、ベンダーなどの既知のパラメーターを使用して内部ディスクを識別し、次のオプションのいずれかを使用してその WWID を決定します。

   - 既存のマルチパスデバイスを表示します。

     ```
     # 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
     ```

   - DM Multipath が作成できるマルチパスデバイスをさらに表示します。

     ```
     # 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
```

- デバイス情報を表示します。

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

この例では、**/dev/sda** は内部ディスクで、その WWID は **WDC_WD800JD-75MSA3_WD-WMAM9FU71040** です。

2. **/etc/multipath.conf** ファイルの **blacklist** セクションを編集し、WWID 属性を使用してこのデバイスを無視します。

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

警告

sda などの **devnode** パラメーターを使用してデバイスを識別することはできますが、**/dev/sda** が再起動時に同じデバイスを参照することが保証されていないため、この手順は安全ではありません。

3. **/etc/multipath.conf** ファイルに設定エラーがないか確認します。

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

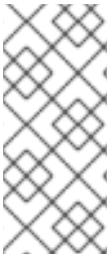
完全なレポートを表示するには、コマンド出力を破棄しないでください。

```
# multipath -t
```

4. ディスクが **initramfs** に含まれている場合は、initramfs を再作成します。詳細は、initramfs でのマルチパスの設定 を参照してください。

5. **multipathd** デーモンを再設定して、**/etc/multipath.conf** ファイルをリロードします。

```
# systemctl reload multipathd
```



注記

ローカルディスク上のマルチパスデバイスは、使用中に削除することができません。このようなデバイスを無視するには、デバイスのすべてのユーザーを停止します。たとえば、デバイス上のファイルシステムをアンマウントし、それを使用している論理ボリュームを非アクティブ化します。これが不可能な場合は、システムを再起動してマルチパスデバイスを削除できます。

検証

1. 内部ディスクが無視され、マルチパス出力に表示されないことを確認します。
   - マルチパスデバイスをリスト表示します。

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

   - DM Multipath が作成できる追加デバイスをリスト表示します。

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

関連情報

- **multipath.conf(5)** man ページ

3.4. DM MULTIPATH での追加のストレージの設定

デフォルトでは、DM Multipath には、DM Multipath をサポートする最も一般的なストレージアレイの組み込み設定が含まれています。ストレージアレイに設定がない場合は、**/etc/multipath.conf** ファイルを編集して追加できます。

注記

初期設定中にストレージデバイスを追加して、予想されるニーズに合わせてセットアップを調整します。DM Multipath では、スケーラビリティーまたはアップグレードのために後でデバイスを追加できますが、この方法では互換性を確保するために設定の調整が必要になる場合があります。

前提条件

- 管理アクセスがある。

手順

1. デフォルトの設定値とサポートされているデバイスを表示します。

   ```
   # multipathd show config
   ```

2. **/etc/multipath.conf** ファイルを編集して、マルチパスを設定します。

   例3.1 HP OPEN-V ストレージデバイス用の DM Multipath 設定

   ```
   # 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. 変更を保存してエディターを閉じます。
4. 新しいデバイスをスキャンして、マルチパスデバイスの一覧を更新します。

```
# multipath -r
```

検証

- マルチパスデバイスが正しく認識されていることを確認します。

```
# multipath -ll
```

3.5. INITRAMFS でのマルチパスの設定

initramfs ファイルシステムでのマルチパスの設定は、特に冗長性と負荷分散が必要なシナリオにおいて、シームレスなストレージ機能を実現するのに不可欠です。この設定により、ブートプロセスの早い段階でマルチパスデバイスが利用可能になります。これは、ストレージ設定の整合性を維持し、潜在的な問題を防ぐうえで非常に重要です。

前提条件

- 管理アクセスがある。
- システムに DM Multipath が設定されている。

手順

1. マルチパス設定ファイルを使用して **initramfs** ファイルシステムを再構築します。

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

注記

initramfs でマルチパスを使用し、その設定ファイルを変更する場合は、変更を反映するために必ず **initramfs** を再構築してください。ルートデバイスがマルチパスを使用している場合、**dracut** コマンドは自動的にマルチパスモジュールを **initramfs** に組み込みます。

2. オプション: **initramfs** のマルチパスが必要なくなった場合は、以下を実行します。
   a. マルチパス設定ファイルを削除します。

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

b. 追加したマルチパス設定を使用して **initramfs** を再構築します。

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

検証

- マルチパス関連のファイルと設定が存在するかどうかを確認します。

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

注記

上記の検証手順により設定の成功を確認できますが、設定が期待どおりに機能することを確認するために、最終的なテスト起動を行うことを推奨します。

- 再起動したら、マルチパスデバイスが正しく認識されていることを確認します。

```
# multipath -ll
```

第4章 NVME デバイスでのマルチパスの有効化

ファイバーチャネル (FC) などのファブリックトランスポートを介して、システムに接続されている Non-volatile Memory Express™ (NVMe™) デバイスをマルチパスすることができます。複数のマルチパスソリューションを選択することができます。

4.1. ネイティブ NVME マルチパスと DM MULTIPATH

Non-volatile Memory Express™ (NVMe™) デバイスは、ネイティブなマルチパス機能をサポートしています。NVMe にマルチパスを設定する場合、標準の DM Multipath フレームワークと NVMe のネイティブなマルチパスのどちらかを選択できます。

DM Multipath と NVMe のネイティブマルチパスは、どちらも NVMe デバイスのマルチパス方式である ANA(Asymmetric Namespace Access) に対応しています。ANA は、コントローラーとホスト間の最適化されたパスを特定し、パフォーマンスを向上させます。

ネイティブ NVMe マルチパスを有効にすると、すべての NVMe デバイスにグローバルに適用されます。より高いパフォーマンスを提供できますが、DM Multipath が提供するすべての機能は含まれていません。例えば、ネイティブの NVMe マルチパスは、**numa** と **round-robin** のパス選択方法のみをサポートしています。

デフォルトでは、NVMe マルチパスは Red Hat Enterprise Linux 9 で有効になっており、これは推奨されるマルチパスソリューションです。

4.2. NVME デバイスでの DM MULTIPATH の有効化

nvme_core.multipath オプションのデフォルトのカーネル設定は **Y** に設定されています。これは、ネイティブ Non-volatile Memory Express™ (NVMe™) マルチパスが有効であることを意味します。ネイティブ NVMe マルチパスを無効にすることで、接続された NVMe デバイスで DM マルチパスを有効にできます。

前提条件

- NVMe デバイスがシステムに接続されていることを確認します。詳細は、NVMe over Fabric デバイスの概要 を参照してください。

手順

1. ネイティブ NVMe マルチパスが有効になっているかどうかを確認します。

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

   コマンドは以下のいずれかを表示します。

   N
   ネイティブ NVMe マルチパスは無効です。

   Y
   ネイティブ NVMe マルチパスは有効です。

2. ネイティブ NVMe マルチパスが有効になっている場合は、次のいずれかの方法を使用して無効にします。

   - カーネルオプションの使用

a. **nvme_core.multipath=N** オプションをコマンドラインに追加します。

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

b. 64 ビットの IBM Z アーキテクチャーでは、ブートメニューを更新します。

```
# zipl
```

c. システムを再起動します。

- カーネルモジュール設定ファイルの使用

a. 以下の内容で **/etc/modprobe.d/nvme_core.conf** 設定ファイルを作成します。

```
options nvme_core multipath=N
```

b. **initramfs** ファイルをバックアップします。

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

c. **initramfs** を再構築します。

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

d. システムを再起動します。

3. DM マルチパスを有効にします。

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

4. 利用可能なすべてのパスに I/O を分配します。**/etc/multipath.conf** ファイルに以下の内容を追加します。

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

注記

DM Multipath が NVMe デバイスを管理する場合、**/sys/class/nvme-subsystem/nvme-subsys0/iopolicy** 設定ファイルは I/O ディストリビューションには影響を与えません。

5. 設定の変更を適用するために、**multipathd** サービスをリロードします。

```
# multipath -r
```

検証

- ネイティブ NVMe マルチパスが無効になっているかどうかを確認します。

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

- DM マルチパスが NVMe デバイスを認識しているかどうかを確認します。

```
# 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
```

関連情報

- カーネルコマンドラインパラメーターの設定
- DM Multipath の設定

4.3. ネイティブ NVME マルチパスの実現

ネイティブ NVMe マルチパスが無効になっている場合は、次の解決策を使用して有効にできます。

前提条件

- NVMe デバイスがシステムに接続されていることを確認します。詳細は、NVMe over Fabric デバイスの概要 を参照してください。

手順

1. カーネルでネイティブ NVMe マルチパスが有効になっているかどうかを確認します。

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

コマンドは以下のいずれかを表示します。

N
ネイティブ NVMe マルチパスは無効です。

Y
ネイティブ NVMe マルチパスは有効です。

2. ネイティブ NVMe マルチパスが無効になっている場合は、次のいずれかの方法を使用して有効にします。

   - カーネルオプションの使用

     a. カーネルコマンドラインから **nvme_core.multipath=N** オプションを削除します。

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

     b. 64 ビットの IBM Z アーキテクチャーでは、ブートメニューを更新します。

     ```
     # zipl
     ```

     c. システムを再起動します。

   - カーネルモジュール設定ファイルの使用

     a. **/etc/modprobe.d/nvme_core.conf** 設定ファイルを削除します。

     ```
     # rm /etc/modprobe.d/nvme_core.conf
     ```

     b. **initramfs** ファイルをバックアップします。

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

     c. **initramfs** を再構築します。

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

     d. システムを再起動します。

3. オプション: 実行中のシステムで、NVMe デバイスの I/O ポリシーを変更して、利用可能なすべてのパスに I/O を分散させます。

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

4. オプション: **udev** ルールを使用して I/O ポリシーを永続的に設定します。以下の内容で **/etc/udev/rules.d/71-nvme-io-policy.rules** ファイルを作成します。

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

検証

1. システムが NVMe デバイスを認識しているかどうかを確認します。次の例は、2 つの NVMe 名前空間を持つ NVMe over fabrics ストレージサブシステムが接続されていることを想定しています:

   ```
   # 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. 接続されているすべての NVMe サブシステムをリストアップします。

```
# 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
```

アクティブトランスポートタイプを確認します。例えば、**nvme0 fc** はファイバーチャネルトランスポートで接続されていることを示し、**nvme tcp** は TCP で接続されていることを示しています。

3. カーネルオプションを編集した場合は、カーネルコマンドラインでネイティブ NVMe マルチパスが有効になっているかどうかを確認します。

```
# cat /proc/cmdline

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

4. I/O ポリシーを変更した場合は、NVMe デバイス上で **round-robin** がアクティブな I/O ポリシーであるかどうかを確認します。

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

round-robin
```

関連情報

- カーネルコマンドラインパラメーターの設定

第5章 DM MULTIPATH 設定ファイルの編集

DM Multipath では、マルチパスで最も一般的に使用する設定値がデフォルトで提供されています。また、DM Multipath に対応する最も一般的なストレージアレイへのサポートも DM Multipath に含まれています。DM-Multipath のデフォルトの設定値は、**/etc/multipath.conf** 設定ファイルを編集するとオーバーライドできます。必要に応じて、サポートされていないデフォルトのストレージアレイを設定ファイルに追加することもできます。

対応しているデバイスなどのデフォルトの設定値は、以下のいずれかのコマンドを実行して確認してください。

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

注記

initramfs ファイルシステムからマルチパスを実行してマルチパス設定ファイルに変更を加える場合、変更を有効にするには **initramfs** ファイルシステムを再構築する必要があります。

マルチパス設定ファイルでは、設定に必要なセクション、またはデフォルト値から変更する必要があるセクションのみを指定する必要があります。使用環境には無関係なファイルのセクションや、デフォルト値を無効にする必要がないファイルのセクションでは、初期ファイルに指定されているコメントアウトを削除する必要はありません。

設定ファイルでは、正規表現の記述構文を使用できます。

5.1. 設定ファイルの概要

マルチパス設定ファイルは以下のセクションに分かれています。

blacklist
: マルチパス設定の対象として考慮しないデバイスのリスト。

blacklist_exceptions
: 指定されていなければ、**blacklist** セクションのパラメーターに従って無視されるマルチパスのデバイスのリスト。

defaults
: DM Multipath のデフォルトの全般設定。

multipaths
: マルチパスデバイスの特性に関する個別設定。ここで指定する値は、設定ファイルの **overrides**、**devices**、および **defaults** のセクションで指定されている値より優先されます。

devices
: ストレージコントローラーの個別設定。ここで指定する値は、設定ファイル内の **defaults** セクションで指定されている値より優先されます。デフォルトでは対応していないストレージアレイを使用している場合は、そのアレイ用の **devices** サブセクションを作成する必要があります。

overrides
: すべてのデバイスに適用される設定。ここで指定する値は、設定ファイルの **devices** セクションおよび **defaults** セクションで指定されている値より優先されます。

システムがマルチパスデバイスの属性を決定するとき、次の順序で **multipath.conf** ファイルから個別のセクションの設定をチェックします。

1. **multipaths** セクション
2. **overrides** セクション
3. **devices** セクション
4. **defaults** セクション

5.2. 設定ファイルの DEFAULTS セクション

/etc/multipath.conf 設定ファイルには、**デフォルト** セクションが含まれています。このセクションには、デバイスマッパー (DM) マルチパスのデフォルト設定が含まれています。デフォルト値は、デバイスの初期設定により異なる場合があります。

デフォルト設定を表示する方法は次のとおりです。

- マシンをマルチパスデバイスにインストールすると、デフォルトのマルチパス設定が自動的に適用されます。デフォルトの設定には、以下が含まれます。
  - デフォルト設定値の完全なリストを表示するには、**multipath -t** または **multipathd show config** コマンドを実行します。
  - 設定オプションのリストと説明については、**multipath.conf** の man ページを参照してください。
- インストール中にマルチパスをセットアップしなかった場合は、**mpathconf --enable** コマンドを実行してデフォルト設定を取得します。

次の表では、**multipath.conf** 設定ファイルの **defaults** セクションで設定される属性について説明します。**multipaths** セクションで指定された属性は、**devices** セクションの値よりも優先されます。**devices** セクションで指定された属性は、デフォルト値よりも優先されます。**overrides** セクションを使用して、すべてのデバイスタイプの属性値を設定します。デバイスタイプの **devices** セクションにビルトインの設定エントリーがある場合も例外ではありません。**overrides** セクションに必須の属性はありません。ただし、このセクションで設定された属性は、**devices** または **defaults** セクションの値よりも優先されます。

表5.1 マルチパス設定の defults セクション

属性	説明
polling_interval	パスチェックが行われる間隔を秒数で指定します。適正に機能するパスでは、チェックの間隔は **max_polling_interval** まで徐々に増加します。デフォルト値は **5** です。
max_polling_interval	2 つのパスチェック間の最大間隔を秒単位で指定します。
	デフォルト値は **4 * polling_interval** です。
find_multipaths	マルチパスデバイスのセットアップモードを定義します。利用可能な値は次のとおりです。

属性	説明
	no: **find_multipaths** が **no** に設定されている場合、**multipath** は **strict** 値と同様にルールを適用し、**multipathd** デーモンは **greedy** 値と同様にルールを適用します。
	yes: 同じ World Wide Identifier (WWID) を持つ**blacklist** に登録されていないデバイスが少なくとも 2 つある場合、または以前にマルチパスがデバイス WWID を持つマルチパスデバイスを作成している場合 (そのマルチパスデバイスがすでに存在しない場合を含む)、デバイスはマルチパスデバイスパスとして扱われます。
	greedy: **multipathd** と **multipath** の両方が、ブラックリストに登録されていないすべてのデバイスをマルチパスデバイスパスとして扱います。
	smart: マルチパスは、ブラックリストに登録されていないすべてのデバイスを自動的にマルチパスデバイスパスと認識します。同じ WWID を持つ 2 番目のパスが **find_multipaths_timeout** に設定された時間内に表示されない場合、マルチパスはデバイスを解放し、システム内で使用できるようにします。**multipathd** デーモンは、**yes** 値と同様にルールを適用します。
	strict: デバイス WWID を使用してマルチパスデバイスを作成する場合にのみ、デバイスをマルチパスパスとして扱います。
	デフォルト値は **off** です。デフォルトの **multipath.conf** ファイルでは、**find_multipaths** が **yes** に設定されます。
find_multipaths_timeout	**find_multipaths smart** が設定されている場合に、最初のパスを検出した後に追加のパスを待機するためのタイムアウトを秒単位で表します。使用できる値を以下に示します。
	正の値: 正の値を設定すると、ブラックリストに登録されていないすべてのデバイスにタイムアウトが適用されます。
	負の値: 負の値を設定すると、マルチパスハードウェアテーブル (ビルトインテーブルまたは **デバイス** セクションのいずれか) にエントリーがある既知のデバイスにのみタイムアウトが適用されます。その他の不明なデバイスは、起動の遅延を避けるために 1 秒だけのタイムアウトを使用します。
	0: システムは、この属性にビルトインのデフォルトを適用します。
	既知のハードウェアのデフォルト値は **-10** です。これは、既知のデバイスのタイムアウトが 10 秒であることを意味します。不明なデバイスのタイムアウトは 1 秒です。**find_multipaths** 属性の値が **smart** 以外の場合、この属性は影響しません。

属性	説明
uxsock_timeout	**multipathd** 対話型コマンドのタイムアウトをミリ秒単位で設定します。
	多数のデバイスがあるシステムでは、**multipathd** 対話型コマンドがタイムアウトして失敗する場合があります。失敗する場合、このタイムアウトを増やして問題を解決してください。
	デフォルト値は **4000** です。
reassign_maps	デバイスマッパーマップの再割り当てを有効にします。このオプションを使用すると、**multipathd** デーモンは、既存のデバイスマッパーのマップを、基本的なブロックデバイスではなく、常にマルチパスデバイスに向けるように再マップします。使用可能な値は、**yes** または **no** です。デフォルト値は **no** です。
verbosity	詳細度のデフォルト値は **2** です。値が高いほど詳細レベルが高くなります。使用できるレベルは **0** から **4** の間です。
path_selector	次回の I/O 動作に使用するパスを決定する、デフォルトのアルゴリズムを指定します。使用できる値を以下に示します。
	round-robin 0: パスグループ内のすべてのパスをループし、**rr_min_io** または **rr_min_io_rq** により決定された同数の I/O 要求をそれぞれに送信します。
	queue-length 0: I/O 要求の次のグループを、未処理の I/O 要求の数が最も少ないパスに送信します。
	service-time 0: I/O 要求の次のグループを、推定サービス時間が最短のパスに送信します。これは、各パスへの未処理 I/O の合計サイズを相対スループットで割ることによって決定されます。
	デフォルト値は **service-time 0** です。
path_grouping_policy	未指定のマルチパスに適用する、デフォルトのパスグルーピングポリシーを指定します。使用できる値を以下に示します。
	failover: 優先グループごとに 1 つのパス。
	multibus: 1 つの優先グループで有効なすべてのパス。
	group_by_serial: 検出されたシリアル番号ごとに 1 つの優先グループ。
	group_by_prio: パス優先値ごとに 1 つの優先グループ。優先順位は、**prio** 属性によって決定されます。

属性	説明
	group_by_node_name: ターゲットノード名ごとに 1 つの優先グループ。**/sys/class/fc_transport/target*/node_name** ディレクトリーには、ターゲットノード名が含まれています。
	デフォルト値は **failover** です。
uid_attrs	WWID による **uevent** のマージを有効にするには、このオプションを設定します。このアクションにより、uevent の処理効率が向上する場合があります。これは、一意のパス識別子 (WWID) を決定するために使用する udev プロパティーを設定する別の方法でもあります。
	このオプションの値は、**type:ATTR** のようなスペースで区切られたレコードのリストです。**type** はデバイスノード名の先頭と一致し、**ATTR** はデバイスの照合に使用する udev プロパティーの名前です。
	このオプションを設定し、それがデバイスのデバイスノード名と一致する場合、このデバイスの WWID を決定するために設定された他の方法が上書きされます。
	この値を **sd:ID_SERIAL dasd:ID_UID nvme:ID_WWN** に設定することで、**uevent** マージを有効にできます。
	デフォルトは **unset** です。
prio	パスの優先値を得るために呼び出すデフォルトの関数を指定します。例えば、SPC-3 の ALUA ビットは悪用可能な **prio** 値を提供します。使用できる値を以下に示します。
	const: すべてのパスに優先度 1 を設定します。
	emc: EMC アレイのパス優先度を生成します。
	sysfs: **sysfs** からパス優先度を生成します。この prioritizer は、オプションの **prio_arg** 値 **exclusive_pref_bit** を受け入れます。**sysfs** 値は、**sysfs** 属性の **access_state** および **preferred_path** を使用します。
	alua: SCSI-3 ALUA 設定に基づいてパス優先度を生成します。デバイス設定で **prio alua** および **prio_args exclusive_pref_bit** を指定すると、マルチパスにより **exclusive_pref_bit** が設定されたパスのみを含むパスグループが作成され、そのパスグループに最高の優先度が割り当てられます。このタイプのケースについて、詳しくは **multipath.conf (5)** の man ページを参照してください。
	ontap: NetApp アレイのパスの優先度を生成します。

属性	説明
	rdac: LSI/Engenio RDAC コントローラーのパスの優先度を生成します。
	hp_sw: active/standby モードにおける Compaq/HP コントローラー用パスの優先度を生成します。
	hds: Hitachi HDS Modular ストレージアレイのパスの優先度を生成します。
	random: 1 から 10 までのランダムな優先度を生成します。
	weightedpath: 正規表現と引数として指定された優先度に基づいて、パスの優先度を生成します。**prio_args** キーワードが必要です。
	path_latency: レイテンシーアルゴリズムに基づいてパスの優先度を生成します。**prio_args** キーワードが必要です。
	ana: NVMe ANA 設定に基づいてパス優先度を生成します。この優先度ルーチンはハードウェアに依存します。
	datacore: 一部の DataCore ストレージアレイのパス優先度を生成します。**prio_args** キーワードが必要です。この優先度ルーチンはハードウェアに依存します。
	iet: IP アドレスに基づいて iSCSI ターゲットのパス優先度を生成します。**prio_args** キーワードが必要です。この優先度ルーチンは、iSCSI でのみ使用できます。
	デフォルト値は、**detect_prio** の設定により異なります。**detect_prio** が **yes** に設定されている場合、デフォルトの優先度アルゴリズムは **sysfs** です。唯一の例外は NetAPP E シリーズで、デフォルトは**alua** です。**detect_prio** が **no** に設定されている場合、デフォルトの優先度アルゴリズムは **const** です。
prio_args	関数 **prio** に渡す引数。これは、次の prioritizer にのみ適用されます。
	weighted: **<hbtl,devname,serial,wwn> <regex1> <prio1> <regex2> <prio2>** 形式の値が必要です。
	hbtl: Regex 値は、SCSI H:B:T:L 形式にできます。例:**1:0:.:. , *:0:0:**
	devname: Regex 値はデバイス名形式にできます。例:**sda**、**sd.e**
	serial: Regex 値はシリアル番号形式にできます。**serial** を、**sysfs** を介して検索するか、コマンド **multipathd show paths format "%z"** を実行して検索します。

属性	説明
	wwn: Regex 値 は、**host_wwnn:host_wwpn:target_wwnn:target_wwpn** の形式にできます。これらの値は、**sysfs** を介して検索するか、コマンド **multipathd show paths format %N:%R:%n:%r"** を実行して検索できます。
	path_latency: **io_num= <integer> base_num=<integer>** の形式の値が必要です。
	io_num: 現在のパスに継続的に送信される読み取り IO の数。この値は、平均パスレイテンシーの計算に役立ちます。有効な値には、**Integer**、**[2, 200]** が含まれます。
	base_num: 対数スケールの基本数値。この値は、異なる優先順位を分割するのに役立ちます。有効な値には、**Integer**、**[2, 10]** が含まれます。平均レイテンシーの最大値は **100s** で、平均レイテンシーの最小値は **1us** です。
	alua: **exclusive_pref_bit** 値が設定されている場合、**preferred_path_bit** が設定されたパスは常に独自のパスグループを作成します。
	sysfs: **exclusive_pref_bit** 値が設定されている場合、**preferred_path_bit** が設定されたパスは常に独自のパスグループを作成します。
	datacore: **timeout=<milliseconds> preferredsds=<name>** の形式の値が必要です。
	preferredsds: この値は必須であり、優先される SDS 名を表します。
	timeout: この値はオプションです。問い合わせのタイムアウトをミリ秒単位で設定します。
	iet: **preferredip=<ip_address>** の形式の値が必要です。
	preferredip: この値は必須です。これは、iSCSI ターゲットの優先 IP アドレス (ドット付き十進表記) です。
	デフォルト値は **unset** です。
features	マルチパスデバイスの追加デフォルト機能です。形式は "number_of_features_plus_arguments feature1 ..." となります。
	features に使用できる値:

属性	説明
	queue_if_no_path: **no_path_retry** を **queue** に設定するのと同じです。
	pg_init_retries n: 失敗するまでパスグループの初期化を最高 **n** 回再試行します。数値は 1 から 50 の間でなければなりません。
	pg_init_delay_msecs msecs: **pg_init** の再試行が開始されるまでのミリ秒数。数値は 0 から 60000 の間でなければなりません。
	queue_mode mode: マルチパスデバイスごとにキューイングモードを選択します。**mode** 値のオプションは、**bio**、**rq**、または **mq** です。それぞれ bio-based、request-based、block-multiqueue request-based (**blk-** mq) に対応しています。
	デフォルトでは、この値は **設定** されません。
path_checker	パスの状態を判断するためのデフォルトのメソッドを指定します。使用できる値を以下に示します。
	readsector0: デバイスの最初のセクターを読み取ります。
	tur: デバイスに対して **TEST UNIT READY** コマンドを発行します。
	emc_clariion: EMC Clariion 固有の EVPD ページ 0xC0 の問い合わせを行い、パスを特定します。
	hp_sw: Active/Standby のファームウェアを搭載した HP ストレージアレイのパスの状態をチェックします。
	rdac: LSI/Engenio RDAC ストレージコントローラーのパス状態をチェックします。
	directio: 直接 I/O を使用する最初のセクターを読み取ります。
	cciss_tur: HP/COMPAQ Smart Array (CCISS) コントローラーのパス状態を確認します。これはハードウェアに依存します。
	none: デバイスはチェックしません。**sysfs** から取得した値を使用するようにフォールバックします。
	デフォルト値は **tur** です。
alias_prefix	この属性は、**user_friendly_names** プレフィックスを表します。
	デフォルト値は **mpath** です。

属性	説明
failback	パスグループのフェイルバックを管理します。使用できる値を以下に示します。
	immediate: アクティブなパスを含む最も優先順位の高いパスグループへの即時フェイルバックを指定します。
	manual: フェイルバックはすぐに実行されず、オペレーターの介入によってのみ発生することを指定します。
	followover: パスグループの最初のパスがアクティブになったときにのみ、自動フェイルバックを実行できるように指定します。これにより、別のノードがフェイルオーバーを要求しているときは、ノードが自動的にフェイルバックしなくなります。
	0 より大きい数値で、フェイルバックの遅延を秒単位で指定します。
	デフォルト値は **manual** です。
rr_min_io	現在のパスグループで、次のパスに切り替えるまでにルーティングする I/O 要求数を指定します。この設定は、2.6.31 より前のカーネルを実行しているシステムにのみ適用されます。2.6.31 以降のシステムには、**rr_min_io_rq** を使用してください。デフォルト値は **1000** です。
rr_min_io_rq	現在のパスグループで、次のパスに切り替えるまでにルーティングする I/O 要求数を指定します。request-base の device-mapper-multipath を使用します。この設定は、現在のカーネルを実行しているシステムで使用できます。2.6.31 より前のバージョンのカーネルを実行しているシステムの場合は **rr_min_io** を使用してください。デフォルト値は **1** です。
no_path_retry	この属性の数値は、キューイングを無効にする前に、パスチェッカーがマルチパスデバイス内のすべてのパスに対して何回失敗するとキューイングが無効になるかを指定します。
	fail を指定すると、キュー待ちはせず直ちに失敗します。
	queue を指定すると、パスが修復されるまでキュー待ちは停止しません。
	デフォルト値は **fail** です。
user_friendly_names	使用できる値を以下に示します。
	yes: システムが **/etc/multipath/bindings** ファイルを使用して、永続的で一意のエイリアスを **mpath<n>** の形式でマルチパスに割り当てることができることを指定します。

属性	説明
	no: システムは WWID をマルチパスのエイリアスとして使用します。設定ファイルの **multipaths** セクションで設定したデバイス固有のエイリアスは、この名前をオーバーライドします。
	デフォルト値は **no** です。
queue_without_daemon	**no** に設定すると、**multipathd** デーモンはシャットダウン時にすべてのデバイスのキューイングを無効にします。デフォルト値は **no** です。
flush_on_last_del	**yes** に設定すると、**multipathd** デーモンは、デバイスへの最後のパスが削除されるとキューイングを無効にします。デフォルト値は no です。
max_fds	マルチパスおよび **multipathd** デーモンで開くことが可能な、オープンファイル記述子の最大数をセットします。これは、**ulimit -n** コマンドに相当します。デフォルト値は **max** で、これは **/proc/sys/fs/nr_open** からのシステム制限に設定されます。
checker_timeout	明示的なタイムアウトで、SCSI コマンドを発行するパス checker および prioritizer に使用するタイムアウト (秒) です。**sys/block/sd<x>/device/timeout** ディレクトリーにはデフォルト値が含まれています。
fast_io_fail_tmo	FC リモートポートで問題が検出されてから、そのリモートポート上のデバイスへの I/O が失敗するまでに SCSI レイヤーが待機する秒数。この値は **dev_loss_tmo** の値より小さくなければなりません。これを **オフ** に設定すると、タイムアウトが無効になります。デフォルト値は **5** です。**fast_io_fail_tmo** オプションは、基礎となるパスデバイスの **recovery_tmo** および **replacement_timeout** オプションの値をオーバーライドします。
dev_loss_tmo	FC リモートポートで問題が検出された後、それをシステムから削除するまで SCSI レイヤーが待機する秒数。無限に設定する場合は、2147483647 秒または 68 年に設定します。OS がデフォルト値を決定します。
eh_deadline	SCSI デバイスに障害が発生した場合に、SCSI レイヤーがエラー処理の実行に費やす最大秒数を指定します。このタイムアウトの後、SCSI レイヤーは HBA のフルリセットを実行します。**rport** が失われることがなく、そのため **fast_io_fail_tmo** と **dev_loss_tmo** はトリガーされないが、**scsi** コマンドは引き続きハングする場合、これを設定する必要があります。SCSI エラーハンドラーが HBA のリセットを実行すると、その HBA 上のすべてのターゲットパスに影響します。**eh_deadline** 値は、影響を受ける HBA 上のすべてのターゲットがマルチパス化されている場合にのみ設定する必要があります。
	デフォルト値は **unset** です。

属性	説明
detect_prio	これが **yes** に設定されている場合、マルチパスは、デバイスが非対称論理ユニットアクセス (ALUA) をサポートする SCSI デバイスであるか、非対称名前空間アクセス (ANA) をサポートする NVMe デバイスであるかを検出します。デバイスが ALUA をサポートしている場合、マルチパスは自動的に **alua** prioritizer を割り当てます。デバイスが ANA をサポートしている場合、マルチパスは自動的に **ana** prioritizer を割り当てます。
	detect_prio が **no** に設定されている場合、またはデバイスが ALUA または ANA をサポートしていない場合、**prio** 属性は prioritizer を設定します。
	デフォルト値は **yes** です。
uid_attribute	デバイスの WWID に使用する **udev** 属性を指定します。
	デフォルト値はデバイスに依存し、SCSI デバイスの場合は **ID_SERIAL**、DASD デバイスの場合は **ID_UID**、NVMe デバイスの場合は **ID_WWN** です。
force_sync	**yes** に設定すると、このパラメーターはパスチェッカーが非同期モードで実行されないようにします。これは、一度に 1 つのチェッカーのみが実行されることを意味します。これは、多数の **multipathd** チェッカーが並行して実行され、CPU に大きな負荷がかかる可能性がある場合に役立ちます。
	デフォルト値は **no** です。
strict_timing	**yes** に設定すると、**multipathd** デーモンはちょうど 1 秒後に新しいパスチェッカーループを開始し、各パスチェックが **polling_interval** に設定された秒数ちょうどで実行されるようにします。ビジー状態のシステムでは、パスチェックにかかる時間が 1 秒を超える場合があります。欠落した時間は、次のラウンドで考慮されます。パスチェックにかかる時間が **polling_interval** に設定された秒数よりを超える場合は、警告が出力されます。
	デフォルト値は **no** です。
retrigger_tries、**retrigger_delay**	**retrigger_tries** パラメーターと **retrigger_delay** パラメーターを組み合わせて使用し、**multipathd** が uevent を再トリガーするようにします。**udev** が元の **uevents** を完全に処理できない場合、マルチパスはデバイスを使用できなくなります。**retrigger_tries** パラメーターは、デバイスが完全にセットアップされていない場合に、マルチパスが **uevent** の再トリガーを試行する回数を設定します。**retrigger_delay** パラメーターは、再試行の間隔 (秒) を設定します。これら両方のオプションでは **0** 以上の数値を使用できます。**retrigger_tries** パラメーターを **0** に設定すると、再試行が無効になります。**retrigger_delay** パラメーターを **0** に設定すると、パスチェッカーの次のループで **uevent** が再発行されます。

属性	説明
	retrigger_tries のデフォルト値は **3** です。**retrigger_delay** のデフォルト値は 10 です。
missing_uev_wait_timeout	この属性は、**multipathd** デーモンが新しく作成されたマルチパスデバイスの **udev** から変更イベントを受信するまで待機する秒数を制御します。その後、デバイスのリロードが自動的に有効になります。ほとんどの場合、**multipathd** は、最初のテーブルロードから変更 **uevent** を受け取るまで、デバイスでのリロードを遅らせます。
	デフォルト値は **30** です。
deferred_remove	**yes** に設定すると、**multipathd** は、最後のパスデバイスが削除されたときに、通常の削除ではなく遅延削除を実行します。これにより、通常の削除が行われ、削除に失敗したときに multipathed デバイスが使用中である場合、デバイスは最終ユーザーがデバイスを終了したときに自動的に削除されます。デフォルト値は **no** です。
san_path_err_threshold, **san_path_err_forget_rate**, **san_path_err_recovery_time**	これら 3 つの属性すべてを 0 より大きい整数に設定すると、**multipathd** デーモンは、パスチェッカーが失敗する頻度を監視することによって不安定なパスが元に戻らないようにすることができます。**san_path_err_forget_rate** チェック内で、パスチェッカーの失敗回数が **san_path_err_threshold** 属性の値を超える場合、**multipathd** デーモンは、パスチェッカーが失敗することなく **san_path_err_recovery_time** 属性の値 (秒単位) が経過するまでパスを復元しません。
	詳細については、**multipath.conf (5)** の **不安定なパスの検出** セクションを参照してください。
	デフォルト値は **no** です。
marginal_path_double_failed_time、**marginal_path_err_sample_time**、**marginal_path_err_rate_threshold**、**marginal_path_err_recheck_gap_time**	**marginal_path_double_failed_time**、**marginal_path_err_rate_threshold**、および **marginal_path_err_recheck_gap_time** が **0** より大きい整数に設定され、**marginal_path_err_sample_time** が **120** より大きい整数に設定されている場合、失敗を繰り返すパスの I/O 失敗率をテストして、**multipathd** デーモンが不安定なパスを元に戻さないようにすることができます。
	marginal_path_double_failed_time 属性に秒単位で設定された値の範囲内でパスが 2 回失敗した場合、パスがバックアップされているとパスチェッカーが判断すると、**multipathd** デーモンはパスをすぐには復元しません。代わりに、**multipathd** は、**marginal_path_err_sample_time** 属性に秒単位で設定された値のパスに対して、読み取り I/O の安定したストリームを発行します。I/O 1000 回あたりのエラー数が **marginal_path_err_rate_threshold** 属性に設定された値より多い場合、**multipathd** は **marginal_path_err_recheck_gap_time** 秒待機してから、読み取り I/O でパスをテストする別のサイクルを開始します。それ以外の場合、**multipathd** はパスを復元します。

属性	説明
	詳細については、**multipath.conf (5)** の **不安定なパスの検出** セクションを参照してください。
	デフォルト値は **no** です。
marginal_pathgroups	使用できる値を以下に示します。
	on: 1 つのマージナルパス検出方法によりパスがマージナルであると判断された場合、システムはそのパスを復元し、別のパスグループに配置します。このグループは、すべての非マージナルパスグループが最初に試行された後にのみ有効になります。これにより、システムは引き続きいくつかのマージナルパスを使用でき、IO エラーが発生する可能性もなくなります。パスは、設定された監視時間が経過すると、すぐに通常のパスグループに戻ります。
	off: **delay_*_checks**、**marginal_path_***、および **san_path_err_*** 属性は、設定された監視時間が経過するまで、システムによる **マージナル** パスまたは **不安定な** パスの復元を阻止します。
	fpin: **multipathd** デーモンは **fpin** 通知を受信し、パスの状態を **marginal** に設定し、**on** 値で記述されているようにパスを再グループ化します。
	marginal_path_* および **san_path_err_*** 属性は、暗黙的に **no** に設定されます。
	詳細については、**multipath.conf (5)** の **不安定なパスの検出** セクションを参照してください。
	デフォルト値は **no** です。
log_checker_err	**once** に設定すると、**multipathd** は、最初のパスチェッカーエラーを詳細レベル 2 でログに記録します。システムは、デバイスが復元されるまで、それ以降のエラーを詳細レベル 3 でログに記録します。**log_checker_err** パラメーターが **always** に設定されている場合、**multipathd** は常に詳細レベル 2 でパスチェッカーエラーをログに記録します。デフォルト値は **always** です。
skip_kpartx	**yes** に設定すると、**kpartx** はデバイスにパーティションを自動的に作成しません。これにより、デバイスにパーティションテーブルがある場合でも、パーティションを作成せずにマルチパスデバイスを作成することができます。このオプションのデフォルト値は **no** です。

属性	説明
max_sectors_kb	このオプションを使用すると、マルチパスデバイスを最初にアクティブ化する前に、マルチパスデバイスの基になるすべてのパスで **max_sectors_kb** デバイスキューパラメーターを指定された値に設定できます。システムが新しいマルチパスデバイスを作成するたびに、デバイスはパスデバイスから **max_sectors_kb** 値を継承します。手動でこの値をマルチパスデバイス向けに高めたり、パスデバイス向けにこの値を低くすると、マルチパスデバイスはパスデバイスが許可するよりも大きな I/O 操作を作成する場合があります。**max_sectors_kb** パラメーターを使用すると、パスデバイス上にマルチパスデバイスを作成する前にこれらの値を簡単に設定でき、無効なサイズの I/O 操作が渡されることを回避できます。このパラメーターを設定しない場合、パスデバイスドライバーが自動的に設定し、マルチパスデバイスはパスデバイスから継承します。
ghost_delay	この属性は、ゴーストパスのみでデバイスを作成した後、**systemd** で使用できるようにマークするまで、マルチパスが待機する秒数を設定します。これにより、マルチパスがハードウェアハンドラーを実行してゴーストパスをアクティブパスに切り替える前に、アクティブパスが表示される時間が与えられます。
	これを **0** または **no** に設定すると、マルチパスはすぐにゴーストパスのみを持つデバイスを準備完了としてマークします。
	デフォルト値は **no** です。
enable_foreign	この属性は、外部ライブラリーを有効または無効にします。
	値は正規表現です。名前が式と一致する場合、外部ライブラリーがロードされます。
	デフォルトでは、外部ライブラリーは有効になっていません。**nvme** を使用して NVMe ネイティブマルチパスのサポートを有効にするか、**.*** を使用してすべての外部ライブラリーを有効にします。
recheck_wwid	**yes** に設定すると、失敗したパスが復元されたときに、**multipathd** デーモンがパスの WWID を再チェックします。WWID に変更がある場合、パスは現在のマルチパスデバイスから削除され、新しいパスとして再度追加されます。**multipathd** デーモンは、パス WWID が手動で再追加された場合にも再度チェックします。
	このオプションは、WWID を取得するためにデフォルトの **uid_attribute**、**ID_SERIAL**、または **sysfs** を使用するように設定された SCSI デバイスに対してのみ機能します。
	デフォルト値は **no** です。

属性	説明
remove_retries	このオプションは、マルチパスが使用中のデバイスの削除を再試行する回数を設定します。各試行の間に、マルチパスは 1 秒間非アクティブになります。デフォルトの値は **0** で、マルチパスは削除を試行しません。
detect_checker	**yes** に設定すると、マルチパスは、デバイスが ALUA または Redundant Disk Array Controller (RDAC) をサポートしているかどうかをチェックします。デバイスが ALUA をサポートしている場合、マルチパスはそのデバイスに **tur path_checker** を割り当てます。デバイスが RDAC をサポートしている場合、**multipathd** デーモンはそのデバイスに **rdac path_checker** を割り当てます。デバイスが ALUA または RDAC をサポートしていない場合、または **detect_checker** が **no** に設定されている場合、**path_checker** 属性はパスチェッカーを設定します。
	デフォルト値は **yes** です。
reservation_key	**mpathpersist** パラメーターは、このサービスアクション予約キーを使用します。永続予約を使用するすべてのマルチパスデバイスに設定する必要があり、**PERSISTENT RESERVE OUT** パラメーターリストの **RESERVATION KEY** フィールドと同じである必要があります。これには、I_T ネクサスを特定するために、アプリケーションクライアントがデバイスサーバーに提供した 8 バイトの値が含まれます。キーを **mpathpersist** に登録するときに **--param-aptpl** オプションを使用する場合は、予約キーの末尾に **:aptpl** を追加する必要があります。
	このパラメーターは **file** に設定することもできます。これにより、**mpathpersist** は、マルチパスデバイスの登録に使用される **RESERVATION KEY** を **prkeys** ファイルに自動的に格納します。**multipathd** デーモンは、このキーを使用して、表示される追加のパスを登録します。登録を削除すると、**RESERVATION KEY** が **prkeys** ファイルから自動的に削除されます。これは、デフォルトで **unset** になっています。永続的な予約が必要な場合は、この属性を **file** に設定することを推奨します。
all_tg_pt	**mpathpersist** がキーを登録するときにこのオプションが **yes** に設定されていると、1 つのホストから 1 つのターゲットポートに登録されたキーは、1 つのホストからすべてのターゲットポートに送信されるものとして扱われます。1 つのターゲットに 1 つのホストではなく、1 つのホストからすべてのターゲットポートで登録キーを自動的に設定および消去するアレイで **mpathpersist** を適切に使用するには、**yes** に設定する必要があります。デフォルト値は **no** です。

関連情報

- **multipath.conf(5)** man ページ

5.3. 設定ファイルの MULTIPATHS セクション

multipath.conf 設定ファイルの **multipaths** セクションを使用して、個々のマルチパスデバイスの属性を設定します。デバイスマッパー (DM) マルチパスは、これらの属性を使用して、**overrides** セクショ

ンの設定を含む、他のすべての設定オプションをオーバーライドします。**overrides** セクションの属性リストについては、設定ファイルの overrides セクション 参照してください。

multipaths セクションでは、**multipath** サブセクションのみが属性として認識されます。次の表は、特定のマルチパスデバイスごとに、**multipath** サブセクションで設定できる属性を示しています。これらの属性は、指定された 1 つのマルチパスにのみ適用されます。複数の **multipath** サブセクションが特定のデバイスの World Wide Identifier (WWID) と一致する場合、それらのサブセクションの内容がマージされます。最新エントリーの設定は、以前のバージョンよりも優先されます。

表5.2 multipath サブセクションの属性

属性	説明
wwid	multipath 属性を適用するマルチパスデバイスの WWID を指定します。このパラメーターは、**multipath.conf** ファイルのこのセクションに必須となります。
alias	multipath 属性が適用されるマルチパスデバイスのシンボリック名を指定します。**user_friendly_names** を使用している場合は、この値を **mpath <n>** に設定しないでください。自動的に割り当てられたユーザーフレンドリー名との競合が発生し、不正なデバイスノード名がになる可能性があります。

次のリストの属性はオプションです。これらを設定しない場合、**overrides**、**devices**、または **defaults** セクションのデフォルト値が適用されます。これらの属性の完全な説明については、設定ファイルのデフォルト を参照してください。

- **path_grouping_policy**
- **path_selector**
- **prio**
- **prio_args**
- **failback**
- **no_path_retry**
- **rr_min_io**
- **rr_min_io_rq**
- **flush_on_last_del**
- **features**
- **reservation_key**
- **user_friendly_names**
- **deferred_remove**
- **san_path_err_threshold**

- **san_path_err_forget_rate**
- **san_path_err_recovery_time**
- **marginal_path_err_sample_time**
- **marginal_path_err_rate_threshold**
- **marginal_path_err_recheck_gap_time**
- **marginal_path_double_failed_time**
- **delay_watch_checks**
- **delay_wait_checks**
- **skip_kpartx**
- **max_sectors_kb**
- **ghost_delay**

設定ファイルで、2 つの特定マルチパスデバイスに対して multipath 属性を指定している例を以下に示します。1 つ目のデバイスの WWID は **3600508b4000156d70001200000b0000** で、シンボリック名は **yellow** です。

2 つ目のマルチパスデバイスの WWID は **1DEC_321816758474** で、シンボリック名は **red** です。

例5.1 マルチパス属性の仕様

```
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
        }
}
```

関連情報

- **multipath.conf(5)** man ページ
- 設定ファイルの defaults セクション
- 設定ファイルの overrides セクション

5.4. 設定ファイルの DEVICES セクション

multipath.conf 設定ファイルの **devices** セクションを使用して、個々のストレージコントローラータイプの設定を定義します。このセクションで設定された値は、**defaults** セクションで指定された値をオーバーライドします。

システムは、**vendor**、**product**、および **revision** キーワードによってストレージコントローラーのタイプを識別します。これらのキーワードは正規表現であり、特定のデバイスに関する **sysfs** 情報と一致する必要があります。

devices セクションは、**device** サブセクションのみを属性として認識します。デバイスに一致するキーワードが複数ある場合は、一致するすべてのエントリーの属性がデバイスに適用されます。属性が複数の一致する **device** サブセクションで指定されている場合、新しいバージョンのエントリーが以前のエントリーよりも優先されます。

重要

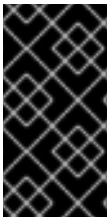
device サブセクションの最新バージョンの設定属性は、以前の **devices** サブセクションおよび **defaults** セクションの属性をオーバーライドします。

次の表に、**device** サブセクションで設定できる属性を示します。

表5.3 devices セクションの属性

属性	説明
vendor	デバイスベンダー名と一致する正規表現を指定します。これは必須属性です。
product	デバイスの製品名に一致する正規表現を指定します。これは必須属性です。
revision	デバイスの製品リビジョンに一致する正規表現を指定します。リビジョン属性がない場合、すべてのデバイスリビジョンが一致します。
product_blacklist	マルチパスはこの属性を使用して、このデバイスエントリーの **vendor** 属性と一致する **vendor** 属性と、この **product_blacklist** 属性と一致する **product** 属性を持つデバイス **blacklist** エントリーを作成します。
vpd_vendor	VPD ページの省略形を使用して、ベンダー固有の Vital Product Data (VPD) ページ情報を表示します。
	multipathd デーモンは、この情報を使用してデバイス固有の情報を収集します。現在、**hp3par** VPD ページのみがサポートされています。
hardware_handler	特定のデバイスタイプに使用するハードウェアハンドラーを指定します。可能な値はすべてハードウェアに依存し、以下が含まれます。
	emc: CLARiX CX/AX や EMC VNX などの DGC クラスアレイおよび Unity ファミリーのハードウェアハンドラー。
	rdac: NetApp SANtricity E/EF Series などの LSI/Engenio/NetApp RDAC クラスと、IBM DELL SGI STK および SUN の OEM アレイのハードウェアハンドラー。

属性	説明
	hp_sw: アクティブ/スタンバイモード専用の HP/COMPAQ/DEC HSG80 および MSA/HSV アレイのハードウェアハンドラー。
	alua: SCSI-3 ALUA 互換アレイのハードウェアハンドラー。
	ana: NVMe ANA 互換アレイのハードウェアハンドラー。
	デフォルト値は **unset** です。



重要

バージョン 4.3 以降の Linux カーネルでは、デバイスハンドラーが既知のデバイスに自動的にアタッチされます。これには、SCSI-3 ALUA をサポートするすべてのデバイスが含まれます。カーネルでは、後でハンドラーを変更できません。カーネルでそのようなデバイスに hardware_handler 属性を設定しても効果はありません。

次のリストの属性はオプションです。設定しない場合は、**defaults** セクションのデフォルト値が適用されます。これらの属性の完全な説明については、設定ファイルのデフォルト を参照してください。

- **path_grouping_policy**
- **uid_attribute**
- **getuid_callout**
- **path_selector**
- **path_checker**
- **prio**
- **prio_args**
- **failback**
- **alias_prefix**
- **no_path_retry**
- **rr_min_io**
- **rr_min_io_rq**
- **flush_on_last_del**
- **features**

- **reservation_key**
- **user_friendly_names**
- **deferred_remove**
- **san_path_err_threshold**
- **san_path_err_forget_rate**
- **san_path_err_recovery_time**
- **marginal_path_err_sample_time**
- **marginal_path_err_rate_threshold**
- **marginal_path_err_recheck_gap_time**
- **marginal_path_double_failed_time**
- **delay_watch_checks**
- **delay_wait_checks**
- **skip_kpartx**
- **max_sectors_kb**
- **ghost_delay**
- **all_tg_pt**

関連情報

- **multipath.conf(5)** man ページ
- 設定ファイルの defaults セクション

5.5. 設定ファイルの OVERRIDES セクション

overrides セクションは、オプションの **protocol** サブセクションを認識し、複数の **protocol** サブセクションを含めることができます。システムは、必須の **type** 属性を使用して、パスデバイスを **protocol** サブセクションと照合します。一致する **protocol** サブセクションの属性は、残りの **overrides** セクションの属性よりも優先されます。一致する **protocol** サブセクションが複数ある場合は、後のエントリーほど優先度が高くなります。

次のリストの属性はオプションです。これらを設定しない場合、**devices** または **defaults** セクションのデフォルト値が適用されます。

- **path_grouping_policy**
- **uid_attribute**
- **getuid_callout**
- **path_selector**

- path_checker
- alias_prefix
- features
- prio
- prio_args
- failback
- no_path_retry
- rr_min_io
- rr_min_io_rq
- flush_on_last_del
- fast_io_fail_tmo
- dev_loss_tmo
- eh_deadline
- user_friendly_names
- retain_attached_hw_handler
- detect_prio
- detect_checker
- deferred_remove
- san_path_err_threshold
- san_path_err_forget_rate
- san_path_err_recovery_time
- marginal_path_err_sample_time
- marginal_path_err_rate_threshold
- marginal_path_err_recheck_gap_time
- marginal_path_double_failed_time
- delay_watch_checks
- delay_wait_checks
- skip_kpartx
- max_sectors_kb

- **ghost_delay**
- **all_tg_pt**

protocol サブセクションは、次の必須属性を認識します。

表5.4 protocol サブセクションの属性

属性	説明
type	パスデバイスのプロトコル文字列を指定します。使用できる値を以下に示します。
	scsi:fcp、**scsi:spi**、**scsi:ssa**、**scsi:sbp**、**scsi:srp**、**scsi:iscsi**、**scsi:sas**、**scsi:adt**、**scsi:ata**、**scsi:unspec**、**ccw**、**cciss**、**nvme**、**undef**
	この属性は正規表現ではありません。パスデバイスプロトコル文字列は正確に一致する必要があります。

次のリストの属性は、**protocol** サブセクションのオプションです。これらを設定しない場合、**overrides**、**devices** または **defaults** セクションのデフォルト値が適用されます。

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

関連情報

- **multipath.conf(5)** man ページ
- 設定ファイルの defaults セクション

5.6. DM MULTIPATH がデバイスタイムアウトの上書き

recovery_tmo sysfs オプションは、特定の iSCSI デバイスのタイムアウトを制御します。次のオプションは、システム全体の **recovery_tmo** 値を上書きします。

- **replacement_timeout** 設定オプションは、システム全体で全 iSCSI デバイスの **recovery_tmo** 値を上書きします。
- DM Multipath が管理するすべての iSCSI デバイスで、DM Multipath の **fast_io_fail_tmo** オプションは、システム全体の **recovery_tmo** 値を上書きします。
  DM Multipath の **fast_io_fail_tmo** オプションは、ファイバーチャネルデバイスの **fast_io_fail_tmo** オプションを上書きします。

DM Multipath の **fast_io_fail_tmo** オプションは **replacement_timeout** よりも優先します。Red Hat では、**replacement_timeou** を使用して、DM Multipath が管理するデバイスの **recovery_tmo** を上書きすることは推奨しません。これは、**multipathd** サービスが再読み込みを行うと、DM Multipath が常に **recovery_tmo** をリセットするためです。

5.7. マルチパス設定ファイルのデフォルトの編集

/etc/multipath.conf 設定ファイルには **defaults** セクションがあり、以下のように **user_friendly_names** パラメーターを **yes** に設定できます。

```
defaults {
        user_friendly_names yes
}
```

上記は、**user_friendly_names** パラメーターのデフォルト値を上書きします。**multipath.conf file** の defaults セクションで設定されたデフォルト値は、devices、multipath で指定された属性によって上書きされない限り、または **multipath.conf** ファイルのセクションをオーバーライドしない限り、DMMultipath によって使用されます。

手順

1. **/etc/multipath.conf** 設定ファイルを表示します。このファイルには、設定のデフォルトのテンプレートが含まれています。

```
#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. 設定パラメーターのデフォルト値を上書きします。このテンプレートから **defaults** のセクションに関連する行をコピーして、コメントを外すことができます。
たとえば、**path_grouping_policy** パラメーターを、デフォルト値の **failover** ではなく **multibus** に上書きするには、以下のように、テンプレートで該当行を見つけて設定ファイルの defaults セクションにコピーし、そのコメントを外します。

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

3. 次のいずれかのコマンドを実行してマルチパス設定ファイルを変更した後、**/etc/multipath.conf** ファイルを検証します。

   - 設定エラーを表示するには、以下のコマンドを実行します。

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

   - 変更が追加された新しい設定を表示するには、以下のコマンドを実行します。

```
# multipath -t
```

4. **/etc/multipath.conf** ファイルを再読み込みし、**multipathd** デーモンを再設定して変更を反映します。

```
# service multipathd reload
```

関連情報

- **multipath.conf(5)** and **multipathd(8)** man pages

5.8. 特定デバイスのマルチパス設定の編集

multipath.conf 設定ファイルの **multipaths** セクションで、必須の WWID パラメーターによって参照される個々のマルチパスデバイスに固有の設定を追加できます。

このデフォルトは DM Multipath により使用され、**multipath.conf** ファイルの **overrides** セクション、**defaults** セクション、および **devices** セクションに設定された属性を上書きします。**multipaths** セクションには、任意の数のマルチパスサブセクションを含めることができます。

手順

1. 特定のマルチパスデバイスの **multipaths** セクションを変更します。設定ファイルで、2 つの特定マルチパスデバイスに対して multipath 属性を指定している例を以下に示します。
   - 1 つ目のデバイスの WWID は **3600508b4000156d70001200000b0000** で、シンボリック名は **yellow** です。
   - 2 番目のマルチパスデバイスの WWID は **1DEC_321816758474** で、シンボリック名 **red** があります。

   rr_weight 属性は **priorities** に設定されています。

```
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. 次のいずれかのコマンドを実行してマルチパス設定ファイルを変更した後、**/etc/multipath.conf** ファイルを検証します。
   - 設定エラーを表示するには、以下のコマンドを実行します。

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

- 変更が追加された新しい設定を表示するには、以下のコマンドを実行します。

```
# multipath -t
```

3. **/etc/multipath.conf** ファイルを再読み込みし、**multipathd** デーモンを再設定して変更を反映します。

```
# service multipathd reload
```

関連情報

- **multipath.conf(5)** man ページ

5.9. プロトコルを使用した特定デバイスのマルチパス設定の変更

トランスポートプロトコルに基づいてマルチパスデバイスパスを設定できます。**/etc/multipath.conf** ファイルの **overrides** セクションの **protocol** サブセクションを使用すると、特定のパスでマルチパス設定を上書きできます。これにより、Fibre Channel (FC) や Internet Small Computer Systems Interface (iSCSI) などの複数のトランスポートプロトコルを介したマルチパスデバイスへのアクセスが可能になります。

protocol サブセクションで設定したオプションは、**overrides**、**devices**、および **defaults** セクションの値を上書きします。これらのオプションは、サブセクションの **type** パラメーターに一致するトランスポートプロトコルを使用するデバイスにのみ適用されます。

前提条件

- システムで Device Mapper (DM) マルチパスを設定している。
- すべてのパスが同じトランスポートプロトコルを使用するのではなく、マルチパスデバイスがある。

手順

1. 以下のコマンドを実行して、特定のパスプロトコルを表示します。

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

2. 各マルチパスタイプに **protocol** サブセクションを追加して、**/etc/multipath.conf** ファイルの **overrides** セクションを編集します。

- **scsi:fcp** プロトコルを使用するパスデバイスの設定。

```
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
        }
}
```

- **scsi:iscsi** プロトコルを使用するパスデバイスの設定。

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

- 他のすべてのプロトコルを使用するパスデバイスの設定。

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

overrides セクションには、複数の **protocol** サブセクションを含めることができます。

重要

protocol サブセクションには **type** パラメーターが含める必要があります。次に、**タイプ** パラメーターが一致するすべてのパスの設定が、**プロトコル** サブセクションにリストされている残りのパラメーターで更新されます。

関連情報

- **multipath.conf(5)** man ページ

5.10. ストレージコントローラーのマルチパス設定の編集

multipath.conf 設定ファイルの **devices** セクションは、各ストレージデバイスの属性を設定します。デバイスを含むパスの **multipath.conf** ファイルの **multipaths** セクションまたは **overrides** セクションに指定された属性により上書きされた場合を除き、これらの値は DM Multipath により使用されます。これらの属性は、**multipath.conf** ファイルの **defaults** セクションに設定された属性を上書きします。

手順

1. サポートされているデバイスを含む、デフォルトの設定値に関する情報を表示します。

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

マルチパスに対応しているデバイスの多くは、デフォルトでマルチパスの設定に含まれています。

2. オプション: デフォルトの設定値を変更する必要がある場合は、それらの値を上書きするデバイスの設定ファイルにエントリーを含めることで、デフォルト値を上書きできます。**multipathd show config** コマンドが表示する、目的のデバイスのデフォルト値をコピーして、変更したい値に書き換えることができます。

3. **vendor** と **product** のパラメーターを設定して、デフォルトで自動的に設定されないデバイスを設定ファイルの **devices** セクションに追加します。次の例に示すように、**/sys/block/device_name/device/vendor** および **/sys/block/device_name/device/model** ファイルを開いてこれらの値を見つけます。ここで、device_name はマルチパスされるデバイスです。

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

4. オプション: 特定のデバイスに応じて、追加のパラメーターを指定します。

   active/active デバイス

   通常、この場合、追加のパラメーターを設定する必要はありません。必要に応じて、**path_grouping_policy** を **multibus** に設定できます。この他に、設定が必要となる可能性があるパラメーターは **no_path_retry** と **rr_min_io** です。

   active/passive デバイス

   I/O を持つパスが自動的に passive パスに切り替えられる場合は、チェッカーの関数を、I/O をパスに送信しない関数に変更し、それが正しく動作するかどうかを検証する必要があります (これを行わないとデバイスはフェイルオーバーし続けます)。これは、**path_checker** を **tur** に設定したことを意味します。これは、ほとんどの場合、Test Unit Ready コマンドをサポートするすべての SCSI デバイスで機能します。

   パスの切り替えに特殊なコマンドを必要とするデバイスにマルチパスを設定するには、ハードウェアハンドラーカーネルモジュールが必要になります。現在、利用可能なハードウェアハンドラーは **emc** です。このハンドラーが目的のデバイスに使用できない場合は、そのデバイスにマルチパスを設定できない可能性があります。

   マルチパス設定ファイルの **device** エントリーの例を以下に示します。

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

5. 次のいずれかのコマンドを実行してマルチパス設定ファイルを変更した後、**/etc/multipath.conf** ファイルを検証します。

- 設定エラーを表示するには、以下のコマンドを実行します。

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

- 変更が追加された新しい設定を表示するには、以下のコマンドを実行します。

```
# multipath -t
```

6. **/etc/multipath.conf** ファイルを再読み込みし、**multipathd** デーモンを再設定して変更を反映します。

```
# service multipathd reload
```

関連情報

- **multipath.conf(5)** and **multipathd(8)** man pages

5.11. すべてのデバイスへのマルチパス値の設定

multipath.conf 設定ファイルの **overrides** セクションを使用すると、すべてのデバイスの設定値を設定できます。このセクションは、**multipath.conf** 設定ファイルの **devices** セクションおよび **defaults** セクションの両方で対応しているすべての属性に対応します。これは、**vendor**、**product**、および **revision** 以外のすべての **devices** セクション属性になります。

DM Multipath は、デバイスを含むパスの **multipath.conf** ファイルの **multipaths** セクションで指定された属性によって上書きされない限り、すべてのデバイスに対してこれらの属性を使用します。この属性は、**multipath.conf** ファイルの **devices** セクションおよび **defaults** セクションに設定された属性を上書きします。

手順

1. デバイス固有の設定を上書きします。たとえば、すべてのデバイスで **no_path_retry** を **fail** に設定できます。すべてのパスに障害が発生した場合は、次のコマンドを使用してキューイングをオフにします。これにより、デバイス固有の設定が上書きされます。

```
overrides {
        no_path_retry fail
}
```

2. 次のいずれかのコマンドを実行してマルチパス設定ファイルを変更した後、**/etc/multipath.conf** ファイルを検証します。

- 設定エラーを表示するには、以下のコマンドを実行します。

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

- 変更が追加された新しい設定を表示するには、以下のコマンドを実行します。

```
# multipath -t
```

3. **/etc/multipath.conf** ファイルを再読み込みし、**multipathd** デーモンを再設定して変更を反映します。

```
# service multipathd reload
```

関連情報

- **multipath.conf(5)** man ページ

第6章 デバイスのマルチパスの防止

マルチパスデバイスを設定する際に、選択したデバイスを無視するように DM Multipath を設定できます。DM Multipath は、これらの無視されたデバイスをマルチパスデバイスにグループ化しません。

6.1. DM MULTIPATH がパスのマルチパスデバイスを作成する際の条件

DM Multipath には、パス用のマルチパスデバイスを作成するか、パスを無視するかどうかを判断するためのデフォルトルールのセットがあります。動作を設定できます。

find_multipaths 設定パラメーターを **off** に設定すると、マルチパスは明示的に無効にされていないパスに対して常にマルチパスデバイスを作成しようとします。**find_multipaths** 設定パラメーターを **on** に設定すると、以下の条件のいずれかが満たされている場合に限り、マルチパスによりデバイスが作成されます。

- 無効にされていない、同じ World-Wide Identification (WWID) を持つパスが少なくとも 2 つあります。
- **multipath** コマンドでデバイスを指定して、デバイスの作成を手動で強制する場合。
- 任意のパスが、以前に作成されたマルチパスデバイスと同一の WWID を持っている場合 (そのマルチパスデバイスがその時点で存在していない場合も含む)。マルチパスデバイスを作成すると、常にそのデバイスの WWID が記憶されるため、同一の WWID を持つパスが検出されると、そのデバイスが自動的に再作成されます。これにより、他のデバイスでマルチパスを無効にしなくても、マルチパスが自動的に正しいパスを選択してマルチパスデバイスにすることができます。

find_multipaths パラメーターを使用せずにマルチパスデバイスを作成してから、パラメーターを **on** に設定した場合は、**/etc/multipath/wwids** ファイルから、マルチパスデバイスとして作成しないデバイスの WWID を削除しなければならない場合があります。次の例は、サンプルの **/etc/multipath/wwids** ファイルを示しています。WWID は、スラッシュ (/) で囲まれています。

```
# 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/
```

on と **off** に加えて、**find_multipaths** も以下の値に設定することができます。

strict

マルチパスは、以前にマルチパス化されていないパスを許可しないため、そのパスは **/etc/multipath/wwids** ファイルに記載されていません。

smart

マルチパスは、**udev** で無効になっていないデバイスが表示されるとすぐに、それらを常に受け入れます。**multipathd** が **find_multipaths_timeout** パラメーターで設定されたタイムアウト内にデバイスを作成しない場合、デバイスに対するクレームを解放します。

find_multipaths の組み込みデフォルト値は **off** です。ただし、**mpathconf** により作成されたデフォルトの **multipath.conf** ファイルは、**find_multipaths** の値を **on** に設定します。

find_multipaths パラメーターが **on** に設定されている場合、マルチパスを使用したくない複数のパスを持つデバイスでのみマルチパスを無効にします。このため、通常、デバイスでマルチパスを無効にする必要はありません。

以前に作成したマルチパスデバイスを **blacklist** に追加する場合は、**-w** オプションを使用して **/etc/multipath/wwids** ファイルからそのデバイスの WWID を削除すると、他のプログラムとの問題を回避するのに役立つことがあります。たとえば、WWID **3600d0230000000000e13954ed5f89300** のデバイス **/dev/sdb** を **/etc/multipath/wwids** ファイルから削除する場合は、以下のいずれかの方法を使用することができます。

- デバイス名でマルチパスデバイスを削除する。

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

- マルチパスデバイスの WWID を使用して削除する。

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

また、**-W** オプションで **/etc/multipath/wwids** ファイルを更新することも可能です。これにより、**/etc/multipath/wwids** ファイルがリセットされ、現在のマルチパスデバイスの WWID のみが含まれるようになります。リセットする場合は、以下を実行してください。

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

関連情報

- **multipath.conf(5)** man ページ

6.2. 特定のデバイスでマルチパスを無効にする基準

以下のいずれかの基準により、デバイスでマルチパスを無効にできます。

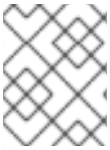
- WWID
- デバイス名
- デバイスの種別
- プロパティー
- プロトコル

すべてのデバイスに対して、DM Multipath は以下の順番でこの基準を評価します。

1. **プロパティー**
2. **devnode**
3. **device**

4. **プロトコル**
5. **wwid**

上記の基準のいずれかによってデバイスが無効になっていることが判明した場合、DM Multipath はそのデバイスを **multipathd** による処理から除外し、後の基準を評価しません。各基準において、デバイスが例外リストと無効なデバイスリストの両方に一致する場合は、例外リストが優先されます。



注記

デフォルトでは、設定ファイルの初期 **blacklist** セクションをコメントアウトした後でも、さまざまなデバイス種別が無効化されます。

関連情報

- マルチパスを無効にしたデバイスに対する例外の追加

6.3. WWID によるマルチパスの無効化

WWID (World-Wide Identification) を使用して、個々のデバイスでマルチパスを無効にできます。

手順

1. デバイスの WWID を検索します。

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

2. **wwid** エントリーを使用して、**/etc/multipath.conf** 設定ファイルのデバイスを無効にします。次の例は、WWID が **3600508b4001080520001e00011700000** のデバイスを無効にする DM マルチパス設定ファイル内の行を示します。

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

3. 次のいずれかのコマンドを実行してマルチパス設定ファイルを変更した後、**/etc/multipath.conf** ファイルを検証します。
   - 設定エラーを表示するには、以下のコマンドを実行します。

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

   - 変更が追加された新しい設定を表示するには、以下のコマンドを実行します。

```
# multipath -t
```

4. **/etc/multipath.conf** ファイルを再読み込みし、**multipathd** デーモンを再設定して変更を反映します。

```
# service multipathd reload
```

6.4. デバイス名によるマルチパスの無効化

DM Multipath がマルチパスデバイスにグループ化しないように、デバイス名でデバイスタイプのマルチパスを無効にできます。

手順

1. デバイス情報を表示します。

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

2. **devnode** エントリーを使用して、**/etc/multipath.conf** 設定ファイルのデバイスを無効にします。
以下の例は、すべての **sd*** デバイスを無効にするため、すべての SCSI デバイスを無効にする DM Multipath 設定ファイル内の行を示しています。

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

devnode エントリーを使用すると、特定タイプのすべてのデバイスではなく、個別のデバイスを無効にできます。ただし、**udev** ルールで静的にマッピングされていない限り、再起動時に特定のデバイス名が同じ名前になる保証がないため、この方法は推奨されません。たとえば、システムが再起動するとデバイス名が **/dev/sda** から **/dev/sdb** に変わる可能性があります。

デフォルトでは、DM Multipath は以下の **devnode** エントリーを使用して、SCSI、NVMe、または DASD 以外のすべてのデバイスを無効にします。

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

このエントリーを無効にするデバイスは、通常、 DM Multipath に対応していません。

3. 次のいずれかのコマンドを実行してマルチパス設定ファイルを変更した後、**/etc/multipath.conf** ファイルを検証します。

- 設定エラーを表示するには、以下のコマンドを実行します。

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

- 変更が追加された新しい設定を表示するには、以下のコマンドを実行します。

```
# multipath -t
```

4. **/etc/multipath.conf** ファイルを再読み込みし、**multipathd** デーモンを再設定して変更を反映します。

```
# service multipathd reload
```

関連情報

- マルチパスを無効にしたデバイスに対する例外の追加

6.5. デバイスの種別によるマルチパスの無効化

デバイスセクションを使用して、デバイスのマルチパスを無効にできます。

手順

1. 表示デバイスの種類:

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

2. **device** セクションを使用して、**/etc/multipath.conf** 設定ファイルのデバイスを無効にします。以下の例は、IBM DS4200 および HP のすべてのデバイスでのマルチパスを無効にします。

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

3. 次のいずれかのコマンドを実行してマルチパス設定ファイルを変更した後、**/etc/multipath.conf** ファイルを検証します。

   - 設定エラーを表示するには、以下のコマンドを実行します。

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

   - 変更が追加された新しい設定を表示するには、以下のコマンドを実行します。

     ```
     # multipath -t
     ```

4. **/etc/multipath.conf** ファイルを再読み込みし、**multipathd** デーモンを再設定して変更を反映します。

```
# service multipathd reload
```

6.6. UDEV プロパティーによるマルチパスの無効化

udev プロパティーパラメーターを使用して、デバイスのマルチパスを無効にできます。

手順

1. デバイスの **udev** 変数を表示します。

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

2. **property** パラメーターを使用して、**/etc/multipath.conf** 設定ファイルのデバイスを無効にします。このパラメーターは、デバイスの **udev** 環境変数名と一致する正規表現の文字列です。
以下の例は、**udev** プロパティー **ID_ATA** を持つすべてのデバイスでマルチパスを無効にします。

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

3. 次のいずれかのコマンドを実行してマルチパス設定ファイルを変更した後、**/etc/multipath.conf** ファイルを検証します。

   - 設定エラーを表示するには、以下のコマンドを実行します。

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

   - 変更が追加された新しい設定を表示するには、以下のコマンドを実行します。

     ```
     # multipath -t
     ```

4. **/etc/multipath.conf** ファイルを再読み込みし、**multipathd** デーモンを再設定して変更を反映します。

```
# service multipathd reload
```

6.7. デバイスプロトコルによるマルチパスの無効化

デバイスプロトコルを使用して、デバイスのマルチパスを無効にすることができます。

手順

1. オプション: パスが使用するプロトコルを表示します。

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

2. **protocol** パラメーターを使用して、**/etc/multipath.conf** 設定ファイルのデバイスを無効にします。
protocol パラメーターは正規表現を取り、プロトコル文字列が一致するすべてのデバイスをブラックリストに指定します。たとえば、すべての nvme デバイスでマルチパスを無効にするには、以下を使用します。

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

DM Multipath は以下のプロトコル文字列を認識します。

- **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**
- **undef**

3. 次のいずれかのコマンドを実行してマルチパス設定ファイルを変更した後、**/etc/multipath.conf** ファイルを検証します。
   - 設定エラーを表示するには、以下のコマンドを実行します。

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

   - 変更が追加された新しい設定を表示するには、以下のコマンドを実行します。

     ```
     # multipath -t
     ```

4. **/etc/multipath.conf** ファイルを再読み込みし、**multipathd** デーモンを再設定して変更を反映します。

   ```
   # service multipathd reload
   ```

6.8. マルチパスを無効にしたデバイスに対する例外の追加

マルチパスが現在無効になっているデバイスに例外を追加することで、マルチパスを有効にできます。

前提条件

- 特定のデバイスでマルチパスが無効になっている。

手順

1. **/etc/multipath.conf** 設定ファイルの **blacklist_exceptions** セクションを使用して、デバイスでマルチパスを有効にします。
   設定ファイルの **blacklist_exceptions** セクションでデバイスを指定する場合は、**blacklist** セクションで指定したものと同じ基準を使用して例外を指定する必要があります。たとえば、無効にしたデバイスがその WWID に関連付けられている場合でも、WWID 例外は **devnode** エントリーで無効になっているデバイスには適用されません。同様に、**devnode** 例外は **devnode** エントリーにしか適用されず、**device** 例外はデバイスエントリーにしか適用されません。

   例6.1 WWID による例外

   たとえば、デバイスが多数あり、その中の 1 つのデバイス (以下の例では WWID が **3600d0230000000000e13955cc3757803** のデバイス) でのみマルチパスを有効にする場合は、有効にするデバイス以外のものを 1 つ 1 つ無効にするのではなく、一旦すべてのデバイスを無効にしてから、**/etc/multipath.conf** ファイルに以下の行を追加し、必要なデバイスのみを有効にします。

   ```
   blacklist {
         wwid ".*"
   }

   blacklist_exceptions {
         wwid "3600d0230000000000e13955cc3757803"
   }
   ```

   または、感嘆符 (**!**) を使用して **blacklist** エントリーを反転することもできます。これにより、指定した WWID を除くすべてのデバイスを無効にできます。

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

   例6.2 udev プロパティーによる例外

   property パラメーターの挙動は、他の **blacklist_exception** パラメーターとは異なります。**property** パラメーターの値は、**udev** データベース内の変数の名前と一致する必要があります。それ以外の場合は、デバイスは無効になります。このパラメーターを使用すると、USB スティックやローカルハードドライブなどの特定の SCSI デバイスでマルチパスを無効にできます。

   合理的にマルチパス化できる SCSI デバイスでのみマルチパスを有効にするには、以下の例のようにこのパラメーターを **SCSI_IDENT_|ID_WWN)** に設定します。

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

2. 次のいずれかのコマンドを実行してマルチパス設定ファイルを変更した後、**/etc/multipath.conf** ファイルを検証します。

- 設定エラーを表示するには、以下のコマンドを実行します。

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

- 変更が追加された新しい設定を表示するには、以下のコマンドを実行します。

```
# multipath -t
```

3. **/etc/multipath.conf** ファイルを再読み込みし、**multipathd** デーモンを再設定して変更を反映します。

```
# service multipathd reload
```

第7章 マルチパス化されたボリュームの管理

DM Multipath が提供するコマンドのうち、マルチパスボリュームを管理するコマンドを以下に示します。

- **multipath**
- **dmsetup**
- **multipathd**

7.1. オンラインのマルチパスデバイスのサイズ変更

オンラインのマルチパスデバイスのサイズを変更する必要がある場合は、以下の手順に従ってください。

手順

1. 物理デバイスのサイズを変更します。
2. 以下のコマンドを実行し、論理ユニット番号 (LUN) までのパスを検索します。

```
# multipath -l
```

3. パスのサイズを変更します。SCSI デバイスの場合は、デバイスの **rescan** ファイルに 1 と書き込むと、SCSI ドライバーによる再スキャンが行われます。以下にコマンド例を示します。

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

各パスデバイスに対してこのコマンドを実行します。たとえば、パスデバイスが **sda**、**sdb**、**sde**、および **sdf** の場合は、次のコマンドを実行します。

```
# 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. マルチパスデバイスのサイズを変更します。

```
# multipathd resize map multipath_device
```

5. ファイルシステムのサイズを変更します (LVM または DOS のパーティションが使用されていないことを前提とします)。

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

7.2. ROOT ファイルシステムをシングルパスデバイスからマルチパスデバイスへ移動

シングルパスのデバイスにシステムをインストールしてから、別のパスを root ファイルシステムに追加する場合は、root ファイルシステムをマルチパスのデバイスに移行する必要があります。シングルパスからマルチパスへの移行については、以下の手順を参照してください。

前提条件

- **device-mapper-multipath** パッケージがインストールされている。

手順

1. **/etc/multipath.conf** 設定ファイルを作成し、multipath モジュールをロードし、**multipathd systemd** サービスを有効にします。

```
# dnf install device-mapper-multipath
```

2. 以下のコマンドを実行して **/etc/multipath.conf** 設定ファイルを作成、マルチパスモジュールをロードして **multipathd** の **chkconfig** を **on** に設定します。

```
# mpathconf --enable
```

3. **find_multipaths** 設定パラメーターが **yes** に設定されていない場合は、Preventing devices from multipathing の説明にしたがって **/etc/multipath.conf** の **blacklist** および **blacklist_exceptions** セクションを編集します。

4. 検出され次第、root デバイスにマルチパスデバイスを構築させるため、次のコマンドを実行します。また、このコマンドを実行すると、パスが 1 つしかない場合でも必ず **find_multipaths** がデバイスを許可するようになります。

```
# multipath -a root_devname
```

たとえば、root デバイスが **/dev/sdb** の場合は、次のコマンドを実行します。

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

5. **multipath** コマンドを実行して設定ファイルの設定が正しく行われたことを確認し、次のような行が出力されていることを検索します。これは、コマンドがマルチパスデバイスの作成に失敗したことを示しています。

```
date  wwid: ignoring map
```

たとえば、デバイスの WWID が **3600d02300069c9ce09d41c4ac9c53200** の場合は、次のような行が出力に表示されます。

```
# multipath
Oct 21 09:37:19 | 3600d02300069c9ce09d41c4ac9c53200: ignoring map
```

6. **multipath** を使用して **initramfs** ファイルシステムを再構築します。

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

7. マシンをシャットダウンします。

8. マシンを起動します。

9. 他のパスがマシンから見えるようにする。

検証手順

- 以下のコマンドを実行して、マルチパスデバイスが作成されているかどうかを確認します。

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

7.3. SWAP ファイルシステムをシングルパスデバイスからマルチパスデバイスへ移動

デフォルトでは、swap デバイスは論理ボリュームとして設定されます。論理ボリュームグループを設定する物理ボリュームでマルチパスを設定している限り、このようなデバイスをマルチパスデバイスとして設定する特別な手順は必要ありません。ただし、swap デバイスが LVM ボリュームではなく、デバイス名でマウントする場合には、**/etc/fstab** ファイルに、適切なマルチパスデバイス名を設定しないといけない場合があります。

手順

1. **/etc/multipath/wwids** ファイルにデバイスの WWID を追加します。

```
# multipath -a swap_devname
```

たとえば、root デバイスが **/dev/sdb** の場合は、次のコマンドを実行します。

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

2. 設定ファイルの設定が正しく行われたことを確認するため、**multipath** コマンドを実行して、次のような行が出力されていることを検索します。

```
date  wwid: ignoring map
```

これは、コマンドがマルチパスデバイスの作成に失敗したことを示しています。

たとえば、デバイスの WWID が 3600d02300069c9ce09d41c4ac9c53200 の場合は、次のような行が出力に表示されます。

```
# multipath
Oct 21 09:37:19 | 3600d02300069c9ce09d41c4ac9c53200: ignoring map
```

3. **/etc/multipath.conf** ファイルに、swap デバイスのエイリアスを設定します。

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

4. **/etc/fstab** ファイルで、root デバイスへの古いデバイスパスを、マルチパスデバイスに置き換えます。
たとえば、**/etc/fstab** ファイルに、以下のようなエントリーがあるとします。

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

エントリーを以下のように変更します。

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

5. multipath を使用して initramfs ファイルシステムを再構築します。

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

6. マシンをシャットダウンします。
7. マシンを起動します。
8. 他のパスがマシンから見えるようにする。

検証手順

- スワップデバイスがマルチパスデバイス上にあるかどうかを確認します。

```
# swapon -s
```

以下に例を示します。

```
# swapon -s

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

ファイル名は、マルチパススワップデバイスと一致させる必要があります。

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

7.4. DMSETUP コマンドでデバイスマッパーエントリーの特定

dmsetup コマンドを使用して、マルチパスを設定したデバイスに一致するデバイスマッパーのエントリーを検索できます。

手順

- デバイスマッパーの全デバイスとそのメジャー番号、マイナー番号を表示します。dm デバイス名は、マイナー番号で特定できます。たとえば、マイナー番号 3 は、マルチパスを設定したデバイス **/dev/dm-3** に対応します。

```
# dmsetup ls
mpathd  (253:4)
mpathep1        (253:12)
mpathfp1        (253:11)
mpathb  (253:3)
mpathgp1        (253:14)
mpathhp1        (253:13)
```

```
mpatha  (253:2)
mpathh  (253:9)
mpathg  (253:8)
VolGroup00-LogVol01     (253:1)
mpathf  (253:7)
VolGroup00-LogVol00     (253:0)
mpathe  (253:6)
mpathbp1        (253:10)
mpathd  (253:5)
```

7.5. MULTIPATHD デーモンの管理

multipathd コマンドを使用して、**multipathd** デーモンを管理できます。

手順

- **multipathd show maps** コマンドの出力のデフォルトフォーマットを表示します。

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

- 一部の **multipathd** コマンドには、後にワイルドカードが付いた **format** オプションを含むものがあります。次のコマンドを実行すると、使用できるワイルドカードのリストを表示できます。

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

- **multipathd** が監視しているマルチパスデバイスを表示します。ワイルドカードを使用して、表示されるフィールドを指定します。

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

- **multipathd** が監視しているパスを表示します。ワイルドカードを使用して、表示されるフィールドを指定します。

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

- データを raw 形式で表示します。

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

raw 形式ではヘッダーは出力されず、フィールドがパッディングされていないため、列とヘッダーが調整されていません。このため、出力はスクリプトで使いやすくなります。

関連情報

- **multipathd**(8) の man ページ

第8章 ストレージデバイスの削除

実行中のシステムからストレージデバイスを安全に削除できるので、システムメモリーのオーバーロードやデータ損失を防ぐことができます。

前提条件

- I/O フラッシュ中にシステムメモリーの読み込みが増加するため、ストレージデバイスを削除する前に、システムのメモリーが十分にあることを確認する必要がある。次のコマンドを使用して、システムの現在のメモリー負荷および空きメモリーを表示する。

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

- Red Hat では、以下のシステムでのストレージデバイスの削除は推奨していない。
  - 空きメモリーが合計メモリーの 5 % 未満 (サンプル 100 件の内 10 件以上)。
  - スワップが有効になっている (**vmstat** コマンドの出力で **si** と **so** のコラムが 0 以外の値)。

8.1. ストレージデバイスの安全な削除

稼働中のシステムからストレージデバイスを安全に取り外すには、上から下へのアプローチが必要です。アプリケーションやファイルシステムなどの最上位層から始め、物理デバイスなどの最下位層に向かって作業を進めます。

ストレージデバイスは複数の方法で使用でき、物理デバイスの上層に別の仮想設定を指定できます。例えば、デバイスの複数のインスタンスをマルチパスデバイスにグループ化したり、RAID の一部にしたり、LVM グループの一部にしたりすることが可能です。さらに、デバイスはファイルシステムを介してアクセスすることもできるし、raw デバイスのように直接アクセスすることもできます。

上から下へのアプローチを用いながら、次のことを確認する必要があります。

- 削除したいデバイスが使用中でないこと
- デバイスへの保留中の I/O がすべてフラッシュされる
- オペレーティングシステムがストレージデバイスを参照していない

8.2. ブロックデバイスと関連メタデータの削除

実行中のシステムからブロックデバイスを安全に削除するには、システムメモリーのオーバーロードとデータ損失を防ぐために、最初にブロックデバイスからメタデータを削除する必要があります。ファイルシステムから始めて、スタック内の各レイヤーに対処し、ディスクに進みます。これらのアクションにより、システムが不整合な状態になるのを防ぎます。

削除するデバイスのタイプに応じて異なる特定のコマンドを使用します。

- **lvremove**、**vgremove**、および **pvremove** は LVM に固有です。
- ソフトウェア RAID の場合、**mdadm** を実行してアレイを削除します。詳細は、RAID の管理 を参照してください。

- LUKS を使用して暗号化されたブロックデバイスの場合、特定の追加手順があります。次の手順は、LUKS を使用して暗号化されたブロックデバイスでは機能しません。詳細は、LUKS を使用したブロックデバイスの暗号化 を参照してください。

警告

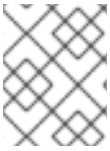
SCSI バスを再びスキャンしたり、ここで説明されている手順に従わずにオペレーティングシステムを変更する別のアクションを実行すると、I/O タイムアウトが原因で遅延が発生したり、デバイスやデータが予期せず削除されたりする可能性があります。

前提条件

- ファイルシステム、論理ボリューム、およびボリュームグループを含む既存のブロックデバイススタックがある。
- 削除するデバイスを他のアプリケーションやサービスが使用していないことを確認した。
- 削除するデバイスからデータをバックアップした。
- オプション: マルチパスデバイスを削除する必要があり、そのパスデバイスにアクセスできない場合は、次のコマンドを実行してマルチパスデバイスのキューイングを無効にしておく。

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

  無効にすることで、デバイスの I/O が失敗し、デバイスを使用しているアプリケーションがシャットダウンできるようになります。



注記

メタデータを含むデバイスを一度に 1 レイヤーずつ削除することで、ディスクに古い署名が残らないようにします。

手順

1. ファイルシステムをアンマウントします。

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

2. ファイルシステムを削除します。

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

注記

/etc/fstab ファイルにエントリーを追加して、ファイルシステムとマウントポイントの間の永続的な関連付けを作成した場合は、この時点で **/etc/fstab** を編集してそのエントリーを削除する必要もあります。

削除するデバイスのタイプに応じて、次の手順に進みます。

3. ファイルシステムを含む論理ボリューム (LV) を削除します。

```
# lvremove vg0/myvol
```

4. ボリュームグループ (VG) に他の論理ボリュームが残っていない場合は、デバイスを含む VG を安全に削除できます。

```
# vgremove vg0
```

5. 物理ボリューム (PV) メタデータを PV デバイスから削除します。

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

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

6. PV が含まれていたパーティションを削除します。

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

注記

デバイスを完全にワイプする場合にのみ、次の手順に従います。

7. パーティションテーブルを削除します。

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

注記

デバイスを物理的に取り外す場合にのみ、次の手順に従います。

- マルチパスデバイスを削除する場合は，次のコマンドを実行します。

  a. デバイスへの全パスを表示します。

```
# multipath -l
```

  このコマンドの出力は、後のステップで必要になります。

  i. I/O をフラッシュして、マルチパスデバイスを削除します。

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

- デバイスがマルチパスデバイスとして設定されていない場合や、デバイスがマルチパスデバイスとして設定されていて、過去に I/O を個別のパスに渡している場合は、未処理の I/O を、使用されている全デバイスパスにフラッシュします。

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

この操作は、**umount** コマンドまたは **vgreduce** コマンドで I/O がフラッシュされないデバイスに直接アクセスする場合に重要になります。

- SCSI デバイスを取り外す場合は、以下のコマンドを実行します。

  a. システム上のアプリケーション、スクリプト、またはユーティリティーで、**/dev/sd**、**/dev/disk/by-path**、または **major:minor** 番号など、デバイスのパスベースの名前への参照をすべて削除します。参照を削除することで、今後追加される別のデバイスが現在のデバイスと混同されないようにします。

  b. SCSI サブシステムからデバイスへの各パスを削除します。

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

  デバイスが以前にマルチパスデバイスとして使用されていた場合、**device-name** は、**multipath -l** コマンドの出力からの内容に置き換えます。

8. 稼働中のシステムから物理デバイスを削除します。このデバイスを削除しても、他のデバイスへの I/O は停止しないことに注意してください。

検証

- 削除したデバイスが **lsblk** コマンドの出力に表示されないことを確認します。出力例を以下に示します。

```
# 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 /
```

関連情報

- **multipath (8)**、**pvremove (8)**、**vgremove (8)**、**lvremove (8)**、**wipefs (8)**、**parted (8)**、**blockdev (8)**、および **umount (8)** man ページ。

第9章 DM MULTIPATH のトラブルシューティング

マルチパス設定の実装に問題がある場合は、さまざまな項目を確認できます。マルチパス設定の反応が遅かったり、機能しない場合は、以下の問題が原因として考えられます。

マルチパスデーモンが実行していない

マルチパス設定の実装に問題がある場合は、DM Multipath の設定 で説明するように、**multipathd** デーモンが実行されていることを確認してください。マルチパスが設定されているデバイスを使用するには、**multipathd** デーモンを実行しておく必要があります。

queue_if_no_path 機能に関する問題

マルチパスデバイスが **features "1 queue_if_no_path"** オプションで設定されている場合は、1 つ以上のパスが復元されるまで、I/O を発行するすべてのプロセスがハングアップします。

9.1. QUEUE_IF_NO_PATH 機能に関する問題のトラブルシューティング

マルチパスデバイスが **features "1 queue_if_no_path"** オプションで設定されている場合は、1 つ以上のパスが復元されるまで、I/O を発行するすべてのプロセスがハングアップします。これを回避するには、**/etc/multipath.conf** ファイルに **no_path_retry N** パラメーターを設定します (**N** はシステムでパスを再試行する回数に置き換えます)。

上記の問題なしで **features "1 queue_if_no_path"** オプションを使用する場合は、すべてのパスが使用できない特定の LUN に対して、実行時にキューイングポリシーを無効にすることができます。

手順

1. キューイングを無効にします。
   - 特定のデバイスの場合:

     ```
     # multipathd disablequeueing map device
     ```

   - すべてのデバイスの場合:

     ```
     # multipathd disablequeueing maps
     ```

     キューイングを無効にすると、**multipathd** を再起動または再読み込みするまで、無効のままになります。
2. キューイングを以前の値にリセットします。
   - 特定のデバイスの場合:

     ```
     # multipathd restorequeueing map device
     ```

   - すべてのデバイスの場合:

     ```
     # multipathd restorequeueing maps
     ```

9.2. MULTIPATHD 対話式コンソールでトラブルシューティング

multipathd -k コマンドは、**multipathd** デーモンに対する対話式のインターフェイスです。このコマンドを実行すると対話式のマルチパスコンソールが立ち上がります。このコマンドを実行した後、**help** を入力すると、利用可能なコマンドのリストが表示され、**Ctrl**+**D** を押すと終了できます。

multipathd 対話型コンソールを使用して、システムで発生した問題のトラブルシューティングを行います。

手順

1. コンソールを終了する前に、デフォルト値を含むマルチパス設定を表示します。

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

2. マルチパスが **multipath.conf** ファイルへの変更をすべて反映するようにします。

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

3. パスチェッカーが正常に動作していることを確認します。

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

4. また、対話型コンソールを起動せずに、コマンドラインから直接 1 つの **multipathd** 対話型コマンドを実行することも可能です。たとえば、マルチパスが **multipath.conf** ファイルへの変更をすべて反映することを確認するには、次のコマンドを実行します。

```
# multipathd reconfigure
```

第10章 EH_DEADLINE を使用したストレージエラーからの回復における最大時間の設定

障害が発生した SCSI デバイスを復旧するのに許容できる最大時間を設定できます。この設定は、ストレージハードウェアが不具合により応答しなくなっても、I/O 応答時間を保証します。

10.1. EH_DEADLINE パラメーター

SCSI エラー処理 (EH) メカニズムは、障害が発生した SCSI デバイスでエラーからの復旧の実行を試行します。SCSI ホストオブジェクト **eh_deadline** パラメーターでは、復旧時間の最大量を設定できます。設定した時間が過ぎると、SCSI EH は、ホストバスアダプター (HBA) 全体を停止してリセットします。

eh_deadline を使用すると、以下のいずれかの時間を短縮できます。

- エラーのあるパスのシャットオフ
- パスの切り替え
- RAID スライスの無効化

警告

eh_deadline が過ぎると、SCSI EH は HBA をリセットします。これは、エラーが発生しているものだけでなく、HBA 上のすべてのターゲットパスに影響します。一部の冗長パスがその他の理由により利用できない場合は、I/O エラーが発生する可能性があります。すべてのターゲットでマルチパスが設定されている場合にのみ、**eh_deadline** を有効にします。また、マルチパスデバイスが完全に冗長でない場合は、**no_path_retry** がパスの回復を可能にするのに十分な大きさに設定されていることを確認する必要があります。

eh_deadline パラメーターの値は秒単位で指定されます。デフォルト設定は **off** で、時間制限が無効になり、すべてのエラー復旧が行われるようになります。

eh_deadline が便利なシナリオ

多くの場合、**eh_deadline** を有効にする必要はありません。**eh_deadline** を使用すると、特定のシナリオで役立つ場合があります。たとえば、ファイバーチャネル (FC) スイッチとターゲットポート間でリンクが失われ、HBA が Registered State Change Notifications (RSCN) を受信しない場合などです。このような場合、I/O 要求やエラーからの復旧コマンドは、エラーに遭遇することなく、すべてタイムアウトになります。この環境で **eh_deadline** を設定すると、リカバリー時間に上限が課せられます。これにより、DM Multipath により、利用できる別のパスで不具合の発生した I/O の再試行が可能になります。

以下の条件下では、**eh_deadline** パラメーターは、これ以上のメリットをもたらしません。その理由は、DM Multipath の再試行を可能にする I/O とエラー復旧コマンドがすぐに失敗するためです。

- RSCN が有効になっている場合
- HBA が利用できなくなっているリンクを登録しない場合

10.2. EH_DEADLINE パラメーターの設定

この手順では、SCSI を復旧する最大時間を制限する **eh-daedline** パラメーターの値を設定します。

手順

- **eh_deadline** は、以下のいずれかの方法で設定できます。
  - **multipath.conf** ファイルの **defaults** セクション
    multipath.conf ファイルの defaults セクションから、**eh_deadline** パラメーターを必要な秒数に設定します。

    ```
    # eh_deadline 300
    ```

    注記

    RHEL 8.4 以降、**multpath.conf** ファイルの defaults セクションを使用して **eh_deadline** パラメーターを設定することが推奨されます。

    このメソッドで **eh_deadline** パラメーターをオフにするには、**eh_deadline** を **off** に設定します。

  - **sysfs**
    /sys/class/scsi_host/host<host-number>/eh_deadline ファイルに秒数を書き込みます。たとえば、SCSI ホスト 6 の **sysfs** を介して **eh_deadline** パラメーターを設定するには、次のようにします。

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

    このメソッドで **eh_deadline** パラメーターをオフにするには、echo **off** を使用します。

  - カーネルパラメーター
    すべての SCSI HBA のデフォルト値は **scsi_mod.eh_deadline** カーネルパラメーターを使用して設定します。

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

    このメソッドで **eh_deadline** パラメーターをオフにするには、echo **-1** を使用します。

関連情報

- How to set eh_deadline and eh_timeout persistently, using a udev rule