Red Hat Enterprise Linux 8

論理ボリュームの設定および管理

LVM 論理ボリュームの設定および管理ガイド

Last Updated: 2019-05-10

Red Hat Enterprise Linux 8 論理ボリュームの設定および管理

LVM 論理ボリュームの設定および管理ガイド

法律上の通知

概要

本書は、Red Hat Enterprise Linux 8 で、LVM 論理ボリュームを管理する方法を説明します。

目次

RED HAT ドキュメントへのフィードバック

ドキュメントの改善に関するご意見やご要望をお聞かせください。

- 特定の文章に簡単なコメントを記入する場合は、ドキュメントが Multi-page HTML 形式になっているのを確認してください。コメントを追加する部分を強調表示し、そのテキストの下に表示される **Add Feedback** ポップアップをクリックし、表示された手順に従ってください。
- より詳細なフィードバックを行う場合は、Bugzilla のチケットを作成します。
    1. Bugzilla の Web サイトにアクセスします。
    2. Component で **Documentation** を選択します。
    3. **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記入してください。
    4. **Submit Bug** をクリックします。

第1章 論理ボリューム

ボリューム管理により、物理ストレージ上に抽象化レイヤーが作成され、論理ストレージボリュームを作成できるようになりました。さまざまな面で、物理ストレージを直接使用するよりも柔軟性が高くなります。また、ハードウェアストレージ設定はソフトウェアから見えなくなるため、アプリケーションを停止したりファイルシステムをアンマウントしたりせずに、サイズ変更や移動が可能になります。したがって、運用コストが削減できます。

物理ストレージを直接使用する場合と比較して、論理ボリュームには、以下のような利点があります。

- 容量の柔軟性
  論理ボリュームを使用すると、ディスクとパーティションを 1 つの論理ボリュームに集約できるため、ファイルシステムを複数のディスクに渡って拡張できます。

- サイズ変更可能なストレージプール
  基礎となるディスクデバイスを再フォーマットしたり、パーティションを再作成したりせずに、簡単なソフトウェアコマンドを使用して論理ボリュームのサイズを拡大または縮小できます。

- オンラインデータ移動
  より新しく、迅速で、障害耐性の高いストレージサブシステムを導入するために、システムがアクティブな状態でもデータを移動できます。データは、ディスクが使用中の場合でもディスクに再配置できます。たとえば、ホットスワップ可能なディスクを削除する前に空にできます。

- 便利なデバイスの命名
  論理ストレージボリュームは、ユーザー定義のカスタム命名されたグループで管理できます。

- ディスクのストライピング
  2 つ以上のディスクにまたがってデータをストライプ化する論理ボリュームを作成できます。これにより、スループットが大幅に向上します。

- ボリュームのミラーリング
  論理ボリュームは、データのミラーを設定する際に便利な方法を提供します。

- ボリュームスナップショット
  論理ボリュームを使用すると、一貫したバックアップが可能なデバイススナップショットを撮ったり、実際のデータに影響を及ぼすことなく変更の影響をテストしたりできます。

- シンボリューム
  論理ボリュームは、シンプロビジョニングにできます。これにより、利用可能なエクステントよりも大きな論理ボリュームを作成できます。

- キャッシュボリューム
  キャッシュ論理ボリュームでは高速なブロックデバイス (SSD ドライブなど) から構成される小さい論理ボリュームが使用されるため、頻繁に使用されるブロックを小さい高速な論理ボリュームに格納することにより、大きくて低速な論理ボリュームのパフォーマンスが向上します。

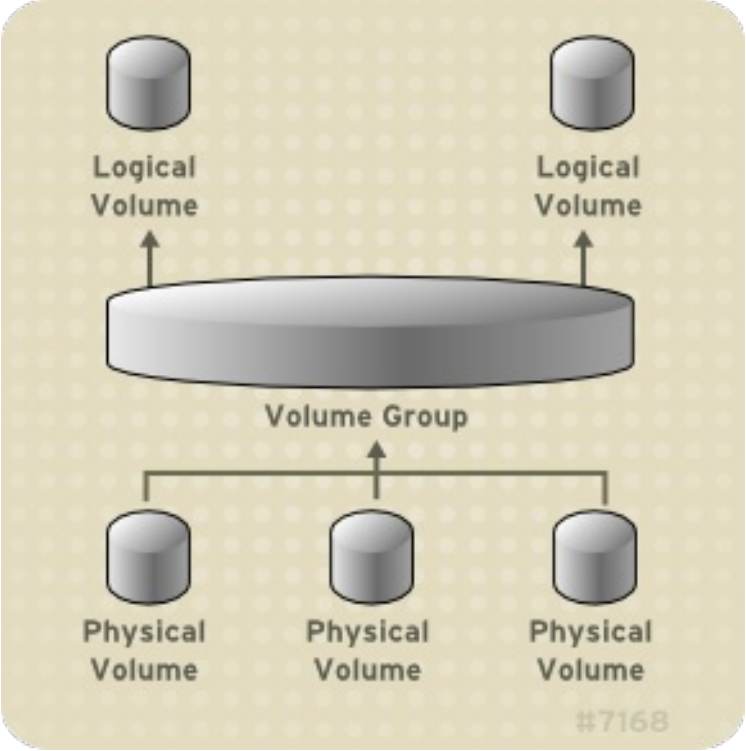
1.1. LVM アーキテクチャーの概要

LVM 論理ボリュームの基礎となる物理ストレージユニットは、パーティション、ディスク全体などのブロックデバイスです。このデバイスは、LVM **物理ボリューム** (Physical Volume: PV) として初期化されます。

LVM 論理ボリュームを作成するために、物理ボリュームを**ボリュームグループ** (Volume Group: VG) に統合します。これによりディスク領域のプールが作成され、そこから LVM 論理ボリューム (Logical Volume: LV) を割り当てます。このプロセスは、ディスクをパーティションに分割する方法に類似しています。論理ボリュームは、ファイルシステムやアプリケーション (データベースなど) に使用されます。

図1.1「LVM 論理ボリュームのコンポーネント」 は、LVM 論理ボリュームのコンポーネントを簡略に示しています。

図1.1 LVM 論理ボリュームのコンポーネント



1.2. 物理ボリューム

LVM 論理ボリュームの基礎となる物理ストレージユニットは、パーティションやディスク全体のようなブロックデバイスです。LVM 論理ボリューム用にデバイスを使用するには、デバイスを物理ボリューム (PV) として初期化する必要があります。ブロックデバイスを物理ボリュームとして初期化すると、デバイスの先頭位置にラベルが付けられます。

LVM ラベルは、デフォルトでは 2 番目の 512 バイトセクターに配置されます。物理ボリュームを作成する場合は、先頭の 4 つのセクターのいずれかにラベルを配置することにより、このデフォルト設定を書き換えることができます。これにより、必要に応じて LVM ボリュームを、このセクターを利用する他のシステムと併用できるようになります。

デバイスがシステムの起動時に任意の順序で初期化されても、LVM ラベルにより物理デバイスの識別とデバイスの順序付けが提供されます。LVM ラベルは、再起動してもクラスター全体で維持されます。

LVM ラベルは、デバイスを LVM 物理ボリュームとして識別するものです。これには、物理ボリューム用のランダムな一意識別子 (UUID) が含まれます。また、ブロックデバイスのサイズもバイト単位で保存し、LVM メタデータがデバイスのどこに保存されているかも記録します。

LVM メタデータには、システムにある LVM ボリュームグループの設定詳細が含まれています。デフォルトでは、メタデータの複製コピーが、ボリュームグループ内で、すべての物理ボリュームの、すべてのメタデータ領域に保存されています。LVM メタデータのサイズは小さく、ASCII 形式が使用されま

す。

現在、LVM では、各物理ボリュームにメタデータのコピーを 1 つまたは 2 つ保存できます。コピーをゼロにすることもできます。デフォルトでは 1 つ保存されます。物理ボリューム上に保存するメタデータのコピー数を一度設定したら、その数を後で変更することはできません。最初のコピーはデバイスの先頭にあるラベルの後に保存されます。2 つ目のコピーがある場合は、デバイスの最後に配置されます。意図したものとは別のディスクに誤って書き込みを行い、ディスクの先頭領域を上書きしてしまった場合でも、デバイス後部にある 2 つ目のコピーでメタデータを復元できます。

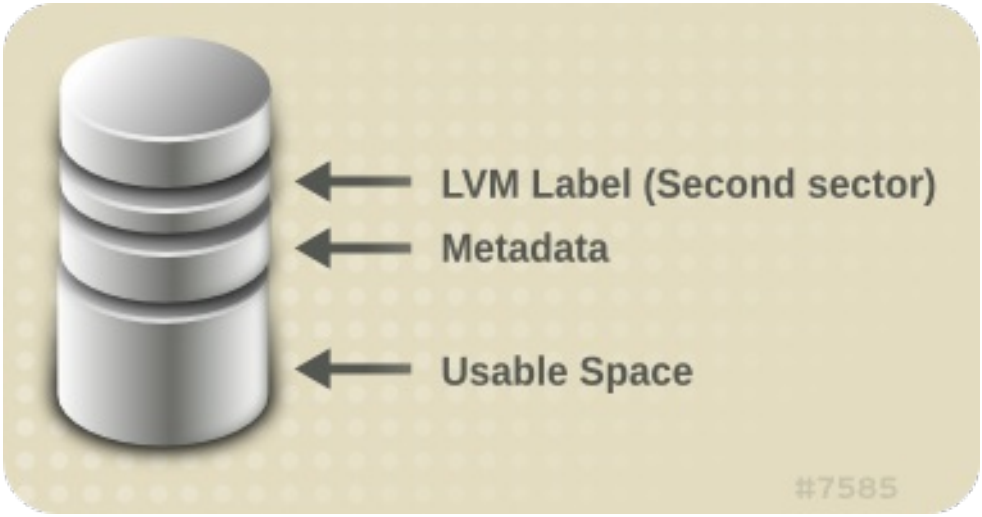
1.2.1. LVM 物理ボリュームのレイアウト

図1.2「物理ボリュームのレイアウト」 は、LVM 物理ボリュームのレイアウトを示しています。LVM ラベルが 2 番目のセクターにあり、その後にメタデータ領域、使用可能なデバイス領域と続きます。

注記

Linux カーネル (および本書) では、セクターのサイズを 512 バイトとしています。

図1.2 物理ボリュームのレイアウト



1.2.2. ディスク上の複数パーティション

LVM の使用により、ディスクパーティションから物理ボリュームを作成できます。Red Hat では、以下のような理由により、ディスク全体に対応するパーティションを 1 つ作成し、1 つの LVM 物理ボリュームとしてラベルを付けることを推奨しています。

- 管理上の利便性
  各ディスクが一度だけ表示されると、システム内のハードウェアの追跡が簡単になります。これは、特にディスクに障害が発生した場合に役に立ちます。さらに、1 つのディスクに物理ボリュームが複数あると、起動時に、不明なパーティションに関するカーネルの警告が発生する可能性があります。

- ストライピングのパフォーマンス
  LVM は、2 つの物理ボリュームが同じ物理ディスクにあるかどうかを認識しません。2 つの物理ボリュームが同じ物理ディスクにあるときに、ストライプ化された論理ボリュームを作成すると、作成されたボリュームのディスクは同じでも、パーティションは異なる可能性があります。このとき、パフォーマンスは向上せず、低下します。

1 つのディスクを、複数の LVM 物理ボリュームに分割しないといけない場合があります (推奨はされません)。たとえば、ディスクがほとんどないシステムで、既存システムを LVM ボリュームに移行する場

合に、パーティション間でデータを移動しなければならない場合があります。さらに、大容量のディスクが存在し、管理目的で複数のボリュームグループを必要とする場合は、そのディスクにパーティションを設定する必要があります。ディスクに複数のパーティションがあり、そのパーティションがいずれも同じボリュームグループにある場合に、ストライプ化ボリュームを作成するときは、論理ボリュームに追加するパーティションを注意して指定してください。

1.3. ボリュームグループ

物理ボリュームはボリュームグループ (VG) に統合されます。これにより、論理ボリュームに割り当て可能なディスク領域のプールが作成されます。

ボリュームグループ内で、割り当て可能なディスク領域は、エクステントと呼ばれる固定サイズの単位に分割されます。割り当て可能な領域の最小単位は、1 エクステントです。エクステントは、物理ボリュームでは物理エクステントと呼ばれます。

論理ボリュームには、物理エクステントと同じサイズの論理エクステントが割り当てられます。そのため、エクステントのサイズは、ボリュームグループ内のすべての論理ボリュームで同じになります。ボリュームグループは、論理エクステントを物理エクステントにマッピングします。

1.4. LVM 論理ボリューム

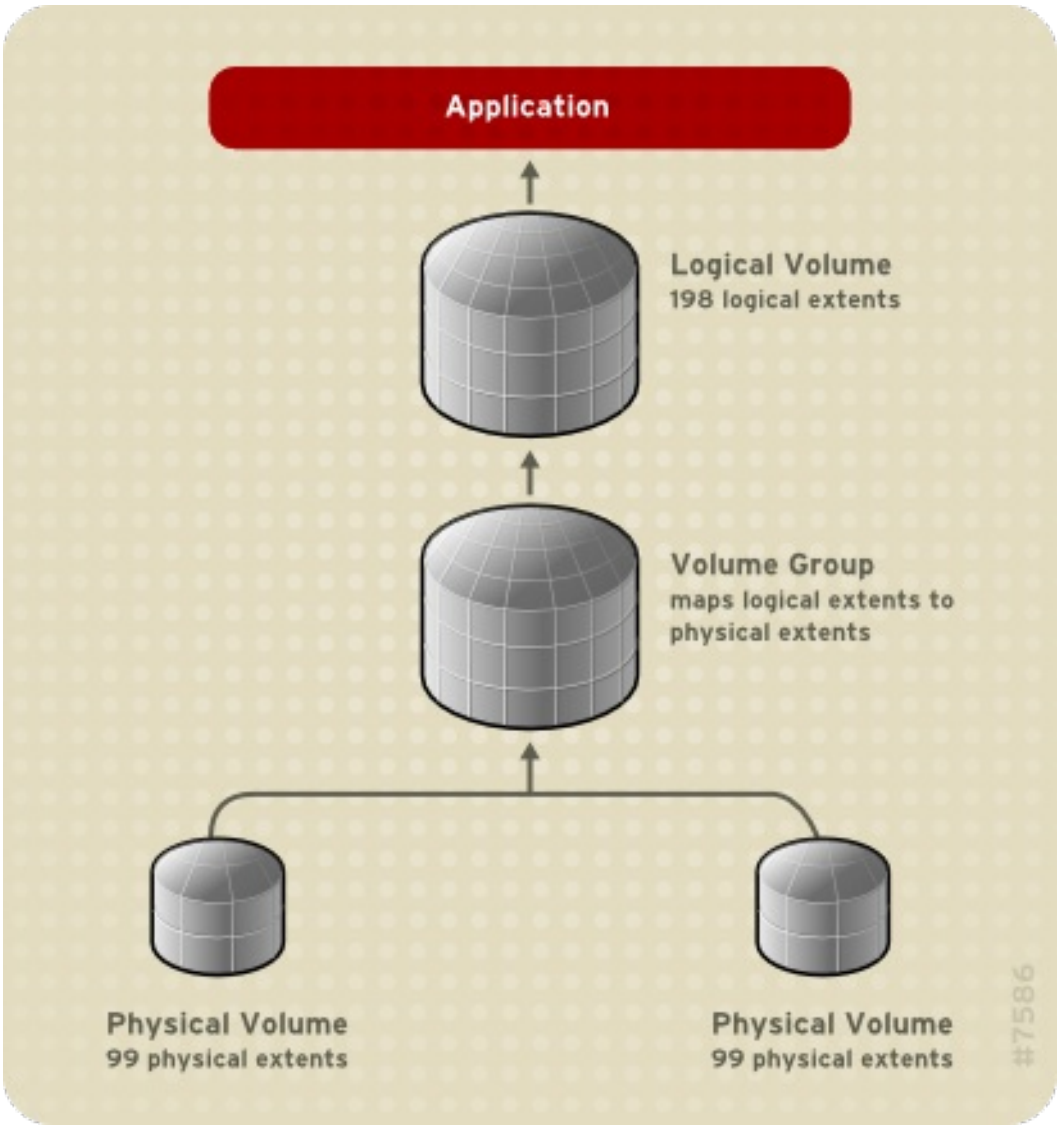
LVM では、ボリュームグループは論理ボリュームに分割されます。以下のセクションでは、論理ボリュームのタイプを説明します。

1.4.1. リニアボリューム

リニアボリュームは、複数の物理ボリュームの領域を 1 つの論理ボリュームに統合します。たとえば、60GB ディスクが 2 つある場合は、120GB の論理ボリュームを作成できます。物理ストレージは連結されます。

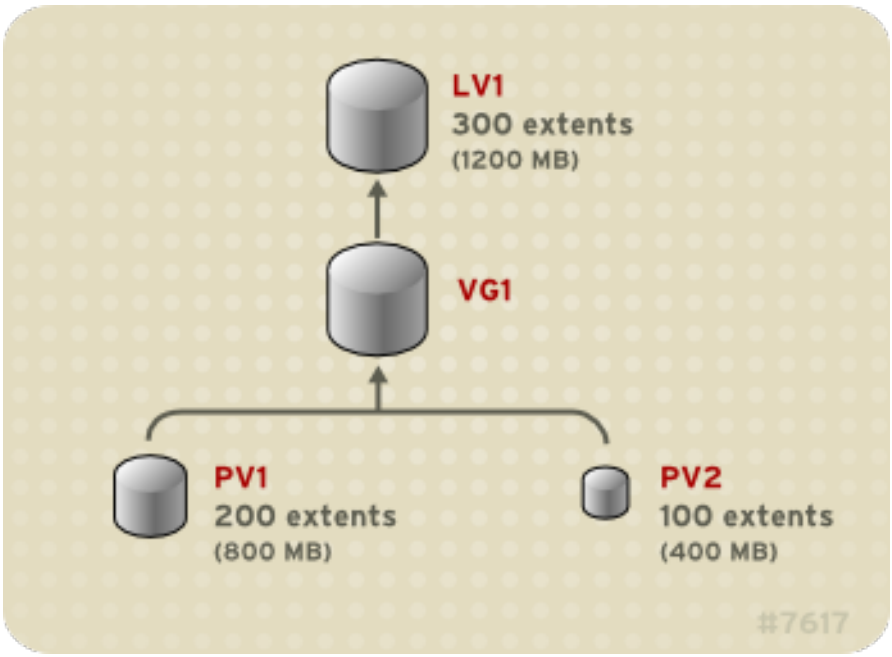
リニアボリュームを作成すると、論理ボリュームの領域に、物理エクステントの範囲が順番に割り当てられます。たとえば、図1.3「エクステントのマッピング」 にあるように、 1 から 99 までの論理エクステントが 1 つ目の物理ボリュームにマッピングされ、 100 から 198 までの論理エクステントが 2 つ目の物理ボリュームにマッピングされます。アプリケーションからは、サイズが 198 エクステントのデバイスが 1 つあるように見えます。

図1.3 エクステントのマッピング



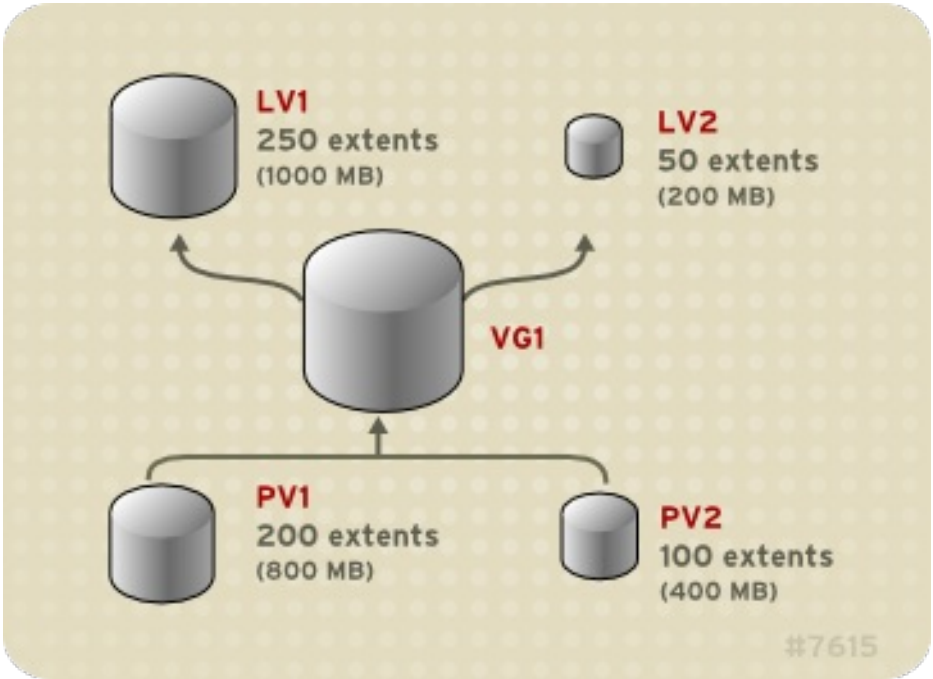
論理ボリュームを構成している各物理ボリュームのサイズは、すべて同じである必要はありません。図1.4「サイズの異なる物理ボリュームを用いたリニアボリューム」 は、物理エクステントサイズが 4MB のボリュームグループ **VG1** を示しています。このボリュームグループには、**PV1** と **PV2** という 2 つの物理ボリュームがあります。1 エクステントは 4MB なので、物理ボリュームが分割される単位は 4MB になります。この例では、**PV1** のエクステントサイズは 200 (800MB) で、 **PV2** のエクステントサイズは 100 (400MB) です。リニアボリュームは、エクステントサイズ 1 から 300 (4MB から 1200MB) の間で作成できます。この例では、300 エクステントのリニアボリューム **LV1** を作成しました。

図1.4 サイズの異なる物理ボリュームを用いたリニアボリューム



物理エクステントのプールから、任意のサイズで、複数のリニア論理ボリュームを設定できます。図1.5「複数の論理ボリューム」 は、図1.4「サイズの異なる物理ボリュームを用いたリニアボリューム」と同じボリュームグループを示していますが、ここでは、そのボリュームグループから論理ボリュームを 2 つ作成します。250 エクステント (1000MB) の **LV1** と、50 エクステント (200MB) の **LV2** です。

図1.5 複数の論理ボリューム



1.4.2. ストライプ化論理ボリューム

LVM 論理ボリュームにデータを書き込む際に、ファイルシステムは、基礎となる物理ボリューム全体にデータを分配します。このとき、ストライプ化論理ボリュームを作成すると、データを物理ボリュームに書き込む方法を制御できます。順次読み取りと書き込みが大量に行われる場合には、これによりデータ I/O の効率を向上できます。

ストライピングは、ラウンドロビン式で、指定した数の物理ボリュームにデータを書き込んでいくことで、パフォーマンスを向上させます。I/O は、ストライピングでは並行して実行されます。これによ

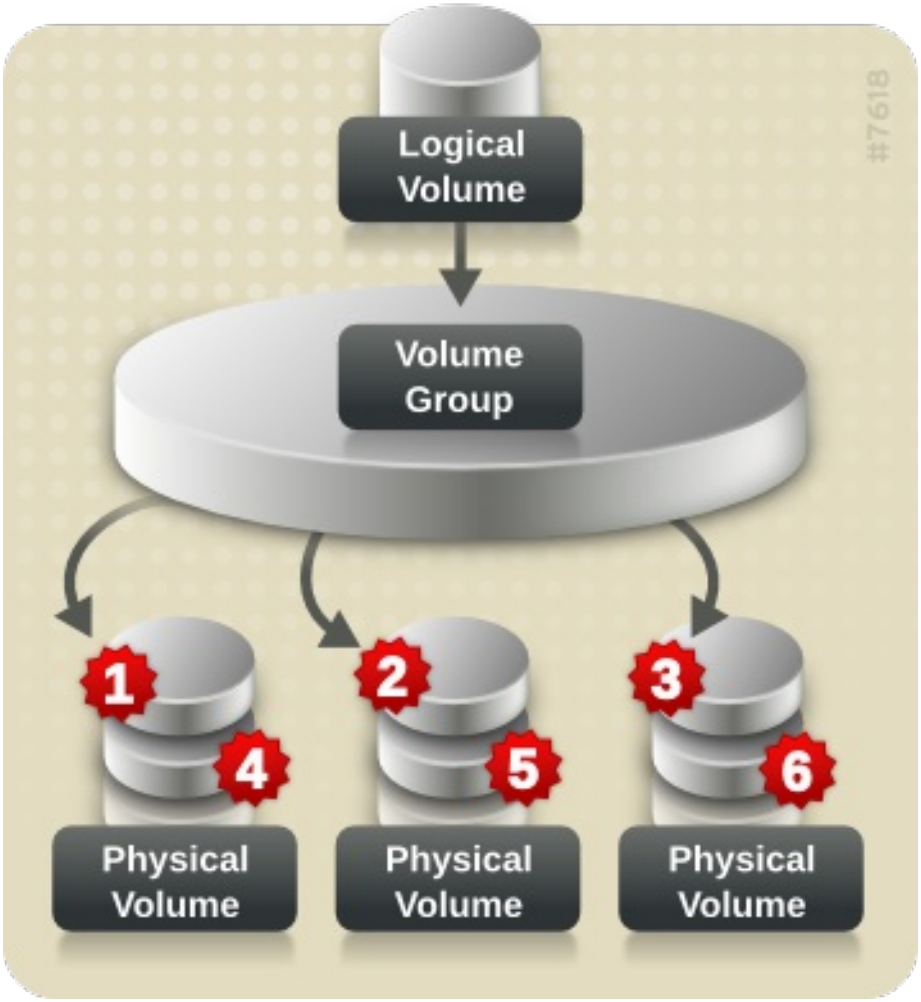
り、ストライプで追加される各物理ボリュームでは、ほぼ直線的なパフォーマンスの向上が期待できます。

以下の図では、3 つの物理ボリュームにデータがストライプ化されている状態を示しています。

- データの 1 番目のストライプは、1 番目の物理ボリュームに書き込まれます。
- データの 2 番目のストライプは、2 番目の物理ボリュームに書き込まれます。
- データの 3 番目のストライプは、3 番目の物理ボリュームに書き込まれます。
- データの 4 番目のストライプは、1 番目の物理ボリュームに書き込まれます。

ストライプ化された論理ボリュームでは、ストライプのサイズは、エクステントのサイズを越えることはできません。

図1.6 3 つの PV にまたがるデータのストライピング

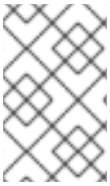


ストライプ化論理ボリュームは、別のデバイスセットを最初のセットの末端に連結すれば拡張できます。ストライプ化論理ボリュームを拡張するには、ストライプに対応するボリュームグループを構成する、基礎となる物理ボリュームセットに、十分な空き領域が必要です。たとえば、ボリュームグループ全域を使用している 2 way ストライプがある場合は、そのボリュームグループに物理ボリュームを 1 つだけ追加しても、ストライプは拡張できません。ボリュームグループには物理ボリュームを 2 つ以上追加する必要があります。

1.4.3. RAID 論理ボリューム

LVM は RAID0/1/4/5/6/10 に対応します。LVM の RAID ボリュームには以下の特徴があります。

- LVM で作成および管理される RAID 論理ボリュームは、MD カーネルドライバーを使用します。
- RAID1 イメージはアレイから一時的に切り離して、後でアレイにマージし直すことができます。
- LVM RAID ボリュームはスナップショットに対応します。



注記

RAID 論理ボリュームはクラスターには対応していません。RAID 論理ボリュームは 1 台のマシンに作成でき、かつ排他的にアクティブにできますが、複数のマシンで同時にアクティブにすることはできません。

1.4.4. シンプロビジョニングされた論理ボリューム (シンボリューム)

論理ボリュームのシンプロビジョニングが可能になりました。これにより、利用可能なエクステントよりも大きい論理ボリュームを作成できます。シンプロビジョニングを使用すると、空き領域のストレージプール (シンプールと呼ばれる) を管理して、アプリケーションで必要なときに、任意の数のデバイスに割り当てることができます。シンプールにバインドされ、アプリケーションが実際に論理ボリュームに書き込むときにあとで割り当てるデバイスを作成できます。シンプールは、ストレージ領域をコスト効率よく割り当てる必要がある場合は、動的に拡張できます。

注記

シンボリュームは、クラスターのノード間ではサポートされません。シンプールとそのすべてのシンボリュームは、1 つのクラスターノードでのみ排他的にアクティブにする必要があります。

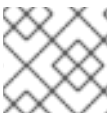
ストレージ管理者は、シンプロビジョニングを使用することで物理ストレージをオーバーコミットできるため、多くの場合は、追加のストレージを購入する必要がなくなります。たとえば、10 人のユーザーから、各自のアプリケーションに使用するファイルシステムをそれぞれ 100GB 要求された場合、ストレージ管理者は各ユーザーに 100GB のファイルシステムを作成します (ただし、実際には 100GB 未満のストレージが、必要に応じて使用されます) 。シンプロビジョニングを使用する場合は、ストレージ管理者がストレージプールを監視し、容量が満杯になり始めたら容量を追加することが重要です。

利用可能な領域をすべて使用できるようにするために、LVM はデータの破棄に対応します。これにより、破棄されたファイルや、その他のブロック範囲で以前に使用された領域を再利用できます。

シンボリュームは、新たに実装されたコピーオンライト (COW) スナップショット論理ボリュームに対応します。これにより、多くの仮想デバイスでシンプール内の同一データを共有できます。

1.4.5. スナップショットボリューム

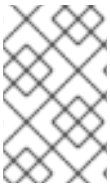
LVM スナップショット機能により、サービスを中断せずに任意の時点でデバイスの仮想イメージを作成できます。スナップショットの取得後に複製元のデバイスに変更が加えられると、データが変更される前に、これから変更される部分のコピーがスナップショット機能により作成されるため、このコピーを使用して、デバイスの状態を再構築できます。



注記

LVM は、シンプロビジョニングされたスナップショットをサポートします。

スナップショットは、スナップショットの作成後に変更されたデータ部分のみをコピーするため、スナップショット機能に必要なストレージは最小限になります。たとえば、コピー元がほとんど更新されない場合は、複製元の 3 ~ 5 % の容量があれば十分にスナップショットを維持できます。



注記

ファイルシステムのスナップショットコピーは仮想コピーであり、ファイルシステムのメディアバックアップを実際に作成するわけではありません。スナップショットは、バックアップの代替手順にはなりません。

複製元のボリュームへの変更を保管するために確保するスペース量は、スナップショットのサイズによって異なります。たとえば、スナップショットを作成してから複製元を完全に上書きした場合に、その変更を保管するのに必要なスナップショットのサイズは、複製元のボリュームと同じか、それ以上になります。スナップショットのサイズは、予想される変更レベルに応じて決定する必要があります。たとえば、**/usr** など、その大部分が読み取り用に使用されるボリュームの短期的なスナップショットに必要なスペースは、**/home** のように大量の書き込みが行われるボリュームの長期的なスナップショットに必要なスペースよりも小さくなります。

スナップショットが満杯になると、そのスナップショットは無効になります。複製元のボリュームの変更をトラッキングできなくなるためです。スナップショットのサイズは常時監視する必要があります。ただし、スナップショットのサイズは完全に変更することが可能なため、ストレージに余裕があれば、スナップショットが停止しないように、ボリュームサイズを拡大できます。逆に、スナップショットボリュームサイズが必要以上に大きければ、そのボリュームのサイズを縮小して、他の論理ボリュームで必要となる領域を確保できます。

スナップショットのファイルシステムを作成しても、複製元への完全な読み取り/書き込みのアクセスは引き続き可能です。スナップショット上のチャンクを変更した場合は、そのチャンクにマークが付けられ、そこには、複製元のボリュームのコピーは入りません。

スナップショット機能にはいくつかの用途があります。

- 最も一般的な用途は、継続的にデータを更新している稼動中のシステムを停止せずに、論理ボリューム上でバックアップを実行する必要がある場合にスナップショットを撮ることです。
- スナップショットファイルシステムで **fsck** コマンドを実行してファイルシステムの整合性をチェックし、複製元のファイルシステムを修復する必要があるかどうかを判断できます。
- スナップショットは読み取りおよび書き込み用なので、スナップショットを撮ってそのスナップショットにテストを実行することにより、実際のデータに触れることなく、実稼働データにアプリケーションのテストを実行できます。
- LVM ボリュームを作成して、Red Hat の仮想化と併用することが可能です。LVM スナップショットを使用して、仮想ゲストイメージのスナップショットを作成できます。このスナップショットは、最小限のストレージを使用して、既存のゲストの変更や新規ゲストの作成を行う上で利便性の高い方法を提供します。

lvconvert コマンドの **--merge** オプションを使用して、スナップショットを複製元のボリュームにマージすることが可能です。この機能の用途の 1 つがシステムロールバックの実行で、データやファイルを紛失した場合や、システムを以前の状態に復元する必要がある場合に行います。スナップショットボリュームのマージ後の論理ボリュームには、複製元のボリューム名、マイナー番号、UUID が付けられ、マージされたスナップショットは削除されます。

1.4.6. シンプロビジョニングされたスナップショットボリューム

Red Hat Enterprise Linux は、シンプロビジョニングされたスナップショットボリュームに対応します。シンプロビジョニングされたスナップショットボリュームにより、多くの仮想デバイスを同じデー

タボリュームに格納できます。これにより管理が簡略化され、スナップショットボリューム間でのデータ共有が可能になります。

シンボリュームや、LVM スナップショットボリュームの場合、シンプロビジョニングされたスナップショットボリュームは、クラスターのノード間ではサポートされていません。スナップショットボリュームは、1つのクラスターノードで排他的にアクティブにする必要があります。

シンプロビジョニングされたスナップショットボリュームの利点は以下のとおりです。

- 同じボリュームからのスナップショットが複数ある場合に、シンプロビジョニングされたスナップショットボリュームを使用すれば、ディスクの使用量を減らすことができます。
- 複製元が同じスナップショットが複数ある場合は、複製元に1回書き込むことにより1回の COW 操作でデータを保存できます。複製元のスナップショットの数を増やしても、速度が大幅に低下することはありません。
- シンプロビジョニングされたスナップショットボリュームは、別のスナップショットの作成元の論理ボリュームとして使用できます。これにより、再帰的スナップショット (スナップショットのスナップショットのそのまたスナップショットなど) の深度を任意に決定できます。
- シン論理ボリュームのスナップショットにより、シン論理ボリュームを作成することもできます。COW 操作が必要になるまで、あるいはスナップショット自体が書き込まれるまで、データ領域は消費されません。
- シンスナップショットボリュームは、複製元とともにアクティブにしておく必要はありません。そのため、スナップショットボリュームが多数ある場合は、複製元のみをアクティブにし、スナップショットボリュームはアクティブにしないでおくことができます。
- シンプロビジョニングされたスナップショットボリュームの複製元を削除すると、そのボリュームのスナップショットは、それぞれ独立したシンプロビジョニングボリュームになります。したがって、スナップショットとその複製元のボリュームをマージする代わりに、複製元のボリュームを削除し、その独立したボリュームを新たな複製元ボリュームにして、シンプロビジョニングされたスナップショットを新たに作成できます。

シンスナップショットボリュームを使用する利点は数多くありますが、古い LVM スナップショットボリューム機能の方がニーズに沿うケースもあります。

- シンプールのチャンクサイズは変更できません。シンプールのチャンクサイズが大きい場合 (1MB など) や、チャンクサイズが短時間のスナップショットには効率的でない場合は、代わりに以前のスナップショット機能を使用できます。
- シンスナップショットボリュームのサイズを制限することはできません。スナップショットは、必要な場合はシンプール内の全領域を使用するため、ニーズに沿っていない場合があります。

一般的には、使用するスナップショットの形式を決定する際に、使用しているサイトの特定要件を考慮に入れるようにしてください。

1.4.7. キャッシュボリューム

LVM は高速ブロックデバイス (SSD ドライブなど) を、大規模な低速ブロックデバイスのライトバックまたはライトスルーキャッシュとして使用することをサポートします。既存の論理ボリュームのパフォーマンスを改善するためにキャッシュ論理ボリュームを作成したり、大規模で低速なデバイスと共に小規模で高速なデバイスで構成される新規のキャッシュ論理ボリュームを作成したりできます。

第2章 LVM 論理ボリュームの設定

以下の手順では、基本的な LVM 管理タスクの例を示します。

2.1. 3 つのディスク上での LVM 論理ボリューム作成

この手順例では、**/dev/sda1**、**/dev/sdb1**、および **/dev/sdc1** のディスクで構成される **mylv** という名前の LVM 論理ボリュームを作成します。

1. ボリュームグループのディスクを使用するには、**pvcreate** コマンドで、そのディスクに LVM 物理ボリュームラベルを付けます。

警告

このコマンドは、**/dev/sda1**、**/dev/sdb1**、および **/dev/sdc1** にあるデータを破棄します。

```
# pvcreate /dev/sda1 /dev/sdb1 /dev/sdc1
  Physical volume "/dev/sda1" successfully created
  Physical volume "/dev/sdb1" successfully created
  Physical volume "/dev/sdc1" successfully created
```

2. 作成した LVM 物理ボリュームで構成されるボリュームグループを作成します。以下のコマンドを使用すると、ボリュームグループ **myvg** が作成されます。

```
# vgcreate myvg /dev/sda1 /dev/sdb1 /dev/sdc1
  Volume group "myvg" successfully created
```

vgs コマンドを使用すると、作成したボリュームグループの属性を表示できます。

```
# vgs
  VG   #PV #LV #SN Attr   VSize  VFree
  myvg   3   0   0 wz--n- 51.45G 51.45G
```

3. 作成したボリュームグループから論理ボリュームを作成します。以下のコマンドを使用して、ボリュームグループ **myvg** から、論理ボリューム **mylv** を作成します。この例では、ボリュームグループの 2 ギガバイトを使用する論理ボリュームが作成されます。

```
# lvcreate -L 2G -n mylv myvg
  Logical volume "mylv" created
```

4. 論理ボリュームにファイルシステムを作成します。以下のコマンドを使用すると、論理ボリュームに **ext4** ファイルシステムが作成されます。

```
# mkfs.ext4 /dev/myvg/mylv
mke2fs 1.44.3 (10-July-2018)
Creating filesystem with 524288 4k blocks and 131072 inodes
Filesystem UUID: 616da032-8a48-4cd7-8705-bd94b7a1c8c4
```

```
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

以下のコマンドは、論理ボリュームをマウントして、ファイルシステムディスクの領域使用率を報告します。

```
# mount /dev/myvg/mylv /mnt
# df
Filesystem            1K-blocks     Used  Available Use% Mounted on
/dev/mapper/myvg-mylv    1998672     6144    1871288   1% /mnt
```

2.2. RAID0 (ストライピング) 論理ボリュームの作成

RAID0 論理ボリュームは、論理ボリュームデータをストライプサイズ単位で複数のサブボリューム全体に分散します。

RAID0 ボリュームを作成するコマンドの書式は以下のとおりです。

```
lvcreate --type raid0[_meta] --stripes Stripes --stripesize StripeSize VolumeGroup
[PhysicalVolumePath ...]
```

表2.1 RAID0 コマンドの作成に関するパラメーター

パラメーター	説明
--type raid0[_meta]	**raid0** を指定すると、メタデータボリュームなしで RAID0 ボリュームが作成されます。**raid0_meta** を指定すると、メタデータボリュームとともに RAID0 ボリュームが作成されます。RAID0 には耐障害性がないため、RAID1/10 の場合のようにミラーリングされたすべてのデータブロックを格納したり、RAID4/5/6 の場合のようにすべてのパリティーブロックを格納したりする必要はありません。したがって、ミラーリングされたブロックまたはパリティーブロックの再同期の進行状態を把握するメタデータボリュームは必要ありません。ただし、RAID0 から RAID4/5/6/10 に変換するには、メタデータボリュームが必要です。**raid0_meta** を指定すると、割り当ての失敗を防ぐためにこれらのメタデータが事前に割り当てられます。
--stripes Stripes	論理ボリュームを分散するデバイスの数を指定します。
--stripesize StripeSize	各ストライプのサイズをキロバイト単位で指定します。これは、次のデバイスに移動する前にデバイスに書き込まれるデータの量です。

パラメーター	説明
VolumeGroup	使用するボリュームグループを指定します。
PhysicalVolumePath ...	使用するデバイスを指定します。指定しない場合は、LVM により、**Stripes** オプションに指定されているデバイスの数が、各ストライプに 1 つずつ選択されます。

この手順例では、**/dev/sda1**、**/dev/sdb1**、および **/dev/sdc1** のディスクで構成される **mylv** という名前の LVM RAID0 論理ボリュームを作成します。

1. **pvcreate** コマンドを使用し、LVM 物理ボリュームとして使用するディスクにラベルを付けます。

警告

このコマンドは、**/dev/sda1**、**/dev/sdb1**、および **/dev/sdc1** にあるデータを破棄します。

```
# pvcreate /dev/sda1 /dev/sdb1 /dev/sdc1
  Physical volume "/dev/sda1" successfully created
  Physical volume "/dev/sdb1" successfully created
  Physical volume "/dev/sdc1" successfully created
```

2. ボリュームグループ **myvg** を作成します。以下のコマンドを使用すると、ボリュームグループ **myvg** が作成されます。

```
# vgcreate myvg /dev/sda1 /dev/sdb1 /dev/sdc1
  Volume group "myvg" successfully created
```

vgs コマンドを使用すると、作成したボリュームグループの属性を表示できます。

```
# vgs
  VG   #PV #LV #SN Attr   VSize  VFree
  myvg   3   0   0 wz--n- 51.45G 51.45G
```

3. 作成したボリュームグループから、RAID0 論理ボリュームを作成します。以下のコマンドを使用すると、ボリュームグループ **myvg** から、RAID0 ボリューム **mylv** が作成されます。この例では、ストライプサイズが 4 キロバイトで 3 つのストライプがある、サイズが 2 ギガバイトの論理ボリュームが作成されます。

```
# lvcreate --type raid0 -L 2G --stripes 3 --stripesize 4 -n mylv myvg
  Rounding size 2.00 GiB (512 extents) up to stripe boundary size 2.00 GiB(513 extents).
  Logical volume "mylv" created.
```

4. RAID0 論理ボリュームにファイルシステムを作成します。以下のコマンドを使用すると、論理ボリュームに **ext4** ファイルシステムが作成されます。

```
# mkfs.ext4 /dev/myvg/mylv
mke2fs 1.44.3 (10-July-2018)
Creating filesystem with 525312 4k blocks and 131376 inodes
Filesystem UUID: 9d4c0704-6028-450a-8b0a-8875358c0511
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

以下のコマンドは、論理ボリュームをマウントして、ファイルシステムディスクの領域使用率を報告します。

```
# mount /dev/myvg/mylv /mnt
# df
Filesystem            1K-blocks     Used  Available Use% Mounted on
/dev/mapper/myvg-mylv    2002684     6168    1875072   1% /mnt
```

2.3. ボリュームグループの分割

この手順例では、3 つの物理ボリュームから構成されるボリュームグループを使用します。この物理ボリュームに未使用領域が十分にあれば、新たにディスクを追加しなくてもボリュームグループを作成できます。

はじめに、ボリュームグループ **myvg** から論理ボリューム **mylv** を作成します。そのボリュームグループは、**/dev/sda1**、**/dev/sdb1**、および **/dev/sdc1** の 3 つの物理ボリュームで構成されます。

その後、**/dev/sda1** と **/dev/sdb1** を使用したボリュームグループ **myvg** と、**/dev/sdc1** を使用したボリュームグループ **yourvg** を使用します。

1. **pvscan** コマンドを使用すると、現在ボリュームグループで利用可能な空き領域の容量を確認できます。

```
# pvscan
  PV /dev/sda1   VG myvg   lvm2 [17.15 GB / 0    free]
  PV /dev/sdb1   VG myvg   lvm2 [17.15 GB / 12.15 GB free]
  PV /dev/sdc1   VG myvg   lvm2 [17.15 GB / 15.80 GB free]
  Total: 3 [51.45 GB] / in use: 3 [51.45 GB] / in no VG: 0 [0   ]
```

2. **pvmove** コマンドを使用して、**/dev/sdc1** で使用中の物理エクステントをすべて **/dev/sdb1** に移動できます。**pvmove** コマンドの実行には時間がかかる場合があります。
クラスター内で、**pvmove** コマンドは、1 つのノードで排他的にアクティブになる論理ボリュームだけを移動できます。

```
# pvmove /dev/sdc1 /dev/sdb1
  /dev/sdc1: Moved: 14.7%
  /dev/sdc1: Moved: 30.3%
  /dev/sdc1: Moved: 45.7%
  /dev/sdc1: Moved: 61.0%
```

```
  /dev/sdc1: Moved: 76.6%
  /dev/sdc1: Moved: 92.2%
  /dev/sdc1: Moved: 100.0%
```

データを移動したら、**/dev/sdc1** 上のすべての領域が空き領域になっていることが確認できます。

```
# pvscan
  PV /dev/sda1   VG myvg   lvm2 [17.15 GB / 0    free]
  PV /dev/sdb1   VG myvg   lvm2 [17.15 GB / 10.80 GB free]
  PV /dev/sdc1   VG myvg   lvm2 [17.15 GB / 17.15 GB free]
  Total: 3 [51.45 GB] / in use: 3 [51.45 GB] / in no VG: 0 [0   ]
```

3. 新規ボリュームグループ **yourvg** を作成するために、**vgsplit** コマンドを使用して、ボリュームグループ **myvg** を分割します。
   以下のコマンドは、ボリュームグループ **myvg** からボリュームグループ **yourvg** を分割し、物理ボリューム **/dev/sdc1** を新しいボリュームグループ **yourvg** に移動します。

```
# lvchange -a n /dev/myvg/mylv
# vgsplit myvg yourvg /dev/sdc1
  Volume group "yourvg" successfully split from "myvg"
```

   vgs コマンドを使用すると、2 つのボリュームグループの属性を確認できます。

```
# vgs
  VG     #PV #LV #SN Attr   VSize  VFree
  myvg     2   1   0 wz--n- 34.30G 10.80G
  yourvg   1   0   0 wz--n- 17.15G 17.15G
```

4. ボリュームグループを新しく作成したら、新規の論理ボリューム **yourlv** を作成します。

```
# lvcreate -L 5G -n yourlv yourvg
  Logical volume "yourlv" created
```

5. 新規の論理ボリュームにファイルシステムを作成し、その論理ボリュームをマウントします。

```
# mkfs.ext4 /dev/yourvg/yourlv
mke2fs 1.44.3 (10-July-2018)
Creating filesystem with 524288 4k blocks and 131072 inodes
Filesystem UUID: 616da032-8a48-4cd7-8705-bd94b7a1c8c4
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

# mount /dev/yourvg/yourlv /mnt
```

2.4. 論理ボリュームからのディスクの削除

この手順例では、ディスクを交換するか、または別のボリュームで使用するために、既存の論理ボリュームからディスクを削除する方法を示しています。ディスクを削除する前に、LVM 物理ボリュームのエクステントを、別のディスクまたはディスクセットに移動する必要があります。

2.4.1. 既存の物理ボリュームへのエクステントの移動

この例では、論理ボリュームが、ボリュームグループ **myvg** の 4 つの物理ボリュームに分配されています。

```
# pvs -o+pv_used
  PV         VG   Fmt  Attr PSize  PFree  Used
  /dev/sda1  myvg lvm2 a-   17.15G 12.15G  5.00G
  /dev/sdb1  myvg lvm2 a-   17.15G 12.15G  5.00G
  /dev/sdc1  myvg lvm2 a-   17.15G 12.15G  5.00G
  /dev/sdd1  myvg lvm2 a-   17.15G  2.15G 15.00G
```

この例では、物理ボリューム **/dev/sdb1** からエクステントを移動して、この物理ボリュームをボリュームグループから削除できるようにします。

1. ボリュームグループの別の物理ボリュームに空きのエクステントが十分にある場合は、他のオプションを使用せずに、削除するデバイス上で **pvmove** コマンドを実行すると、エクステントは他のデバイスに分配されます。
   クラスター内で、**pvmove** コマンドは、1 つのノードで排他的にアクティブになる論理ボリュームだけを移動できます。

   ```
   # pvmove /dev/sdb1
     /dev/sdb1: Moved: 2.0%
    ...
     /dev/sdb1: Moved: 79.2%
    ...
     /dev/sdb1: Moved: 100.0%
   ```

   pvmove コマンドの実行が終了すると、エクステントの分配は次のようになります。

   ```
   # pvs -o+pv_used
     PV         VG   Fmt  Attr PSize  PFree  Used
     /dev/sda1  myvg lvm2 a-   17.15G  7.15G 10.00G
     /dev/sdb1  myvg lvm2 a-   17.15G 17.15G     0
     /dev/sdc1  myvg lvm2 a-   17.15G 12.15G  5.00G
     /dev/sdd1  myvg lvm2 a-   17.15G  2.15G 15.00G
   ```

2. **vgreduce** コマンドを使用して、ボリュームグループから物理ボリューム **/dev/sdb1** を削除します。

   ```
   # vgreduce myvg /dev/sdb1
     Removed "/dev/sdb1" from volume group "myvg"
   # pvs
     PV         VG   Fmt  Attr PSize  PFree
     /dev/sda1  myvg lvm2 a-   17.15G  7.15G
     /dev/sdb1       lvm2 --   17.15G 17.15G
     /dev/sdc1  myvg lvm2 a-   17.15G 12.15G
     /dev/sdd1  myvg lvm2 a-   17.15G  2.15G
   ```

これでディスクは物理的に削除できるようになり、他のユーザーに割り当てることも可能になります。

2.4.2. 新規ディスクへのエクステントの移動

この例では、以下のように、ボリュームグループ **myvg** の 3 つの物理ボリュームに、論理ボリュームが分配されています。

```
# pvs -o+pv_used
  PV         VG   Fmt  Attr PSize  PFree  Used
  /dev/sda1  myvg lvm2 a-   17.15G  7.15G 10.00G
  /dev/sdb1  myvg lvm2 a-   17.15G 15.15G  2.00G
  /dev/sdc1  myvg lvm2 a-   17.15G 15.15G  2.00G
```

以下の手順では、**/dev/sdb1** のエクステントを、新しいデバイス **/dev/sdd1** に移動します。

1. **/dev/sdd1** から、物理ボリュームを作成します。

   ```
   # pvcreate /dev/sdd1
     Physical volume "/dev/sdd1" successfully created
   ```

2. 新たに作成した物理ボリューム **/dev/sdd1** を、既存のボリュームグループ **myvg** に追加します。

   ```
   # vgextend myvg /dev/sdd1
     Volume group "myvg" successfully extended
   # pvs -o+pv_used
     PV         VG   Fmt  Attr PSize  PFree  Used
     /dev/sda1  myvg lvm2 a-   17.15G  7.15G 10.00G
     /dev/sdb1  myvg lvm2 a-   17.15G 15.15G  2.00G
     /dev/sdc1  myvg lvm2 a-   17.15G 15.15G  2.00G
     /dev/sdd1  myvg lvm2 a-   17.15G 17.15G     0
   ```

3. **pvmove** を使用して、データを **/dev/sdb1** から **/dev/sdd1** へ移動します。

   ```
   # pvmove /dev/sdb1 /dev/sdd1
     /dev/sdb1: Moved: 10.0%
   ...
     /dev/sdb1: Moved: 79.7%
   ...
     /dev/sdb1: Moved: 100.0%

   # pvs -o+pv_used
     PV         VG   Fmt  Attr PSize  PFree  Used
     /dev/sda1  myvg lvm2 a-   17.15G  7.15G 10.00G
     /dev/sdb1  myvg lvm2 a-   17.15G 17.15G     0
     /dev/sdc1  myvg lvm2 a-   17.15G 15.15G  2.00G
     /dev/sdd1  myvg lvm2 a-   17.15G 15.15G  2.00G
   ```

4. データを **/dev/sdb1** から移動したら、この物理ボリュームをボリュームグループから削除できます。

   ```
   # vgreduce myvg /dev/sdb1
     Removed "/dev/sdb1" from volume group "myvg"
   ```

これで、このディスクを別のボリュームグループに再度割り当てたり、システムから削除できるようになりました。

2.5. ボリュームグループの統合

2 つのボリュームグループを統合して 1 つのボリュームグループにするには、**vgmerge** コマンドを使用します。ボリュームの物理エクステントサイズが同じで、かつ両ボリュームグループの物理および論理ボリュームのサマリーが「マージ先」ボリュームグループの制限内に収まる場合は、非アクティブな「マージ元」のボリュームを、アクティブまたは非アクティブの「マージ先」ボリュームにマージができます。

以下のコマンドは、非アクティブなボリュームグループ **my_vg** をアクティブまたは非アクティブなボリュームグループ **databases** にマージして、詳細なランタイム情報を提供します。

```
# vgmerge -v databases my_vg
```

2.6. ボリュームグループを別のシステムへ移動

LVM ボリュームグループ全体を、別のシステムに移動できます。これを実行するには、**vgexport** コマンドと **vgimport** コマンドの使用が推奨されます。

注記

vgimport コマンドの **--force** 引数を使用できます。これで物理ボリュームがないボリュームグループをインポートし、**vgreduce --removemissing** コマンドを実行します。

vgexport コマンドは、非アクティブのボリュームグループにシステムがアクセスできないようにするため、物理ボリュームの割り当て解除が可能になります。**vgimport** コマンドは、**vgexport** コマンドで非アクティブにしていたボリュームグループに、マシンが再度アクセスできるようにします。

ボリュームグループを 2 つのシステム間で移行するには、以下の手順に従います。

1. ボリュームグループ内のアクティブなボリュームのファイルにアクセスしているユーザーがいないことを確認してから、論理ボリュームをアンマウントします。

2. **vgchange** コマンドで **-a n** 引数を使用して、そのボリュームグループを非アクティブとしてマークします。これによりボリュームグループでこれ以上の動作が発生しないようにします。

3. **vgexport** コマンドを使用してボリュームグループをエクスポートします。これにより、削除中のシステムからボリュームグループへアクセスできなくなります。
   ボリュームグループをエクスポートして **pvscan** コマンドを実行すると、以下の例のように、エクスポート先のボリュームグループに物理ボリュームが表示されます。

   ```
   # pvscan
     PV /dev/sda1    is in exported VG myvg [17.15 GB / 7.15 GB free]
     PV /dev/sdc1    is in exported VG myvg [17.15 GB / 15.15 GB free]
     PV /dev/sdd1   is in exported VG myvg [17.15 GB / 15.15 GB free]
     ...
   ```

   次にシステムがシャットダウンする時に、ボリュームグループを構成していたディスクを外して、新しいシステムに接続できます。

4. ディスクが新しいシステムに接続したら、**vgimport** コマンドを使用してボリュームグループをインポートし、新しいシステムからアクセスできるようにします。

5. **vgchange** コマンドで **-a y** 引数を使用して、ボリュームグループをアクティブにします。

6. ファイルシステムをマウントして使用可能にします。

2.7. 永続的なデバイス番号の設定

メジャーデバイス番号とマイナーデバイス番号は、モジュールのロード時に動的に割り当てられます。一部のアプリケーションは、ブロックデバイスが常に同じデバイス (メジャーとマイナー) 番号でアクティブにされている場合に、最も効果的に機能します。これらは **lvcreate** コマンドと **lvchange** コマンドで、以下の引数を使用して指定できます。

```
--persistent y --major major --minor minor
```

別のデバイスにすでに動的に割り当てられている番号を使用しないように、マイナー番号は大きくします。

NFS を使用してファイルシステムをエクスポートする場合は、そのエクスポートファイルで **fsid** パラメーターを指定すると、LVM 内で永続的なデバイス番号を設定する必要がなくなります。

2.8. LVM エクステントサイズの指定

ボリュームグループの作成に物理ボリュームが使用されるとき、ディスク領域はデフォルトで 4MB のエクステントに分割されます。このエクステントは、論理ボリュームのサイズを拡張/縮小する最小単位です。エクステントの数が多くても、論理ボリュームの I/O パフォーマンスに影響を与えることはありません。

エクステントサイズのデフォルト設定が適切でない場合は、**vgcreate** コマンドに **-s** オプションを使用して、エクステントのサイズを指定できます。**vgcreate** コマンドに **-p** 引数と **-l** 引数を使用すると、ボリュームグループに追加可能な物理ボリュームまたは論理ボリュームの数に制限をかけることができます。

第3章 論理ボリュームのサイズ変更

論理ボリュームを作成したら、ボリュームのサイズを変更できます。

3.1. 論理ボリュームの拡張

論理ボリュームのサイズを拡張するには、**lvextend** コマンドを使用します。

論理ボリュームを拡張する場合は、追加するボリュームの容量、または拡張後のボリュームのサイズを指定できます。

以下のコマンドは、論理ボリューム **/dev/myvg/homevol** を 12 ギガバイトに拡張します。

```
# lvextend -L12G /dev/myvg/homevol
lvextend -- extending logical volume "/dev/myvg/homevol" to 12 GB
lvextend -- doing automatic backup of volume group "myvg"
lvextend -- logical volume "/dev/myvg/homevol" successfully extended
```

以下のコマンドは、論理ボリューム **/dev/myvg/homevol** に 1 ギガバイト追加します。

```
# lvextend -L+1G /dev/myvg/homevol
lvextend -- extending logical volume "/dev/myvg/homevol" to 13 GB
lvextend -- doing automatic backup of volume group "myvg"
lvextend -- logical volume "/dev/myvg/homevol" successfully extended
```

lvcreate コマンドと同様に、**lvextend** コマンドの **-l** 引数を使用して、論理ボリュームの拡張サイズをエクステント数で指定できます。また、この引数を使用してボリュームグループのパーセンテージ、またはボリュームグループに残ってる空き領域をパーセンテージで指定することもできます。以下のコマンドは、**testlv** という論理ボリュームを拡張して、ボリュームグループ **myvg** の空き領域をすべて使用するようにします。

```
# lvextend -l +100%FREE /dev/myvg/testlv
  Extending logical volume testlv to 68.59 GB
  Logical volume testlv successfully resized
```

論理ボリュームを拡張したら、それに合わせてファイルシステムのサイズも拡張する必要があります。

多くのファイルシステムサイズ変更ツールは、デフォルトで、ファイルシステムのサイズを、その下の論理ボリュームのサイズまで拡大するため、2 つのコマンドに同じサイズを指定する必要はありません。

3.2. 論理ボリュームのファイルシステムの拡張

論理ボリュームのファイルシステムを拡張するには、以下の手順を実行します。

1. 既存のボリュームグループに、論理ボリュームを拡張するのに十分な空き領域があるかどうかを調べます。空き領域が足りない場合は、次の手順を実行します。

   a. **pvcreate** コマンドを使用して、新しい物理ボリュームを作成します。

   b. **vgextend** コマンドを使用して、その物理ボリュームを組み込む、拡張対象の論理ボリュームが含まれるボリュームグループを拡張します。

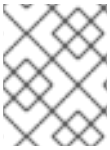
2. ボリュームグループを拡張できたら、**lvresize** コマンドで論理ボリュームを拡張して、ファイルシステムが拡張できるようにします。

3. 論理ボリュームでファイルシステムのサイズを変更します。

lvresize コマンドの **-r** オプションを使用すれば、1つのコマンドで、論理ボリュームを拡張し、基礎となるファイルシステムのサイズを変更できることに注意してください。

3.3. 論理ボリュームの縮小

論理ボリュームのサイズを縮小するには、**lvreduce** コマンドを使用します。



注記

縮小は GFS2 および XFS ファイルシステムではサポートされないため、GFS2 または XFS ファイルシステムが含まれる論理ボリュームのサイズは縮小できません。

縮小する論理ボリュームにファイルシステムが含まれている場合は、データの損失を防ぐため、ファイルシステムが、縮小する論理ボリュームにあるスペースを使用しないようにしてください。そのため、論理ボリュームにファイルシステムが含まれている場合は **lvreduce** コマンドの **--resizefs** オプションを使用することが推奨されます。このオプションを使用すると、**lvreduce** コマンドは論理ボリュームを縮小する前にファイルシステムの縮小を試みます。ファイルシステムの縮小に失敗した場合 (ファイルシステムが満杯であったり、ファイルシステムが縮小をサポートしない場合に失敗します)、**lvreduce** コマンドの実行に失敗し、論理ボリュームを縮小しません。

警告

ほとんどの場合、**lvreduce** コマンドはデータ損失の可能性を警告し、確認を要求します。しかし、論理ボリュームが非アクティブな状態であったり、**--resizefs** オプションが使用されなかった場合など、警告が表示されない場合があるため、データの損失を防ぐのに確認プロンプトのみを信頼しないようにしてください。

lvreduce コマンドの **--test** オプションは、ファイルシステムのチェックやファイルシステムのサイズ変更のテストを行わないため、操作が安全な場所を示しません。

以下のコマンドは、ボリュームグループ **vg00** の論理ボリューム **lvol1** を64 メガバイトに縮小します。この例では、**lvol1** にはファイルシステムが含まれ、このコマンドにより、論理ボリュームとともにサイズが変更されます。次の例は、コマンドの出力を示しています。

```
# lvreduce --resizefs -L 64M vg00/lvol1
fsck from util-linux 2.23.2
/dev/mapper/vg00-lvol1: clean, 11/25688 files, 8896/102400 blocks
resize2fs 1.42.9 (28-Dec-2013)
Resizing the filesystem on /dev/mapper/vg00-lvol1 to 65536 (1k) blocks.
The filesystem on /dev/mapper/vg00-lvol1 is now 65536 blocks long.

  Size of logical volume vg00/lvol1 changed from 100.00 MiB (25 extents) to 64.00 MiB (16 extents).
  Logical volume vg00/lvol1 successfully resized.
```

サイズ変更値の前に「-」記号を指定すると、その値が論理ボリュームの実際のサイズから減算されます。以下の例は、論理ボリュームの絶対サイズを 64 メガバイトに縮小する代わりに、ボリュームを 64 メガバイト分縮小する場合に使用するコマンドを示しています。

```
# lvreduce --resizefs -L -64M vg00/lvol1
```

3.4. ストライプ化論理ボリュームの拡張

ストライプ化論理ボリュームのサイズを拡大するには、ボリュームグループを構成している物理ボリュームに、ストライプをサポートする十分な空き領域が必要です。たとえば、ボリュームグループ全域を使用する 2 way ストライプがある場合は、ボリュームグループに物理ボリュームを 1 つ追加しただけでは、ストライプを拡張することはできません。拡張するには少なくとも 2 つの物理ボリュームを、ボリュームグループに追加する必要があります。

たとえば、以下の **vgs** コマンドで表示された、2 つの物理ボリュームで構成されるボリュームグループ **vg** について考えてみましょう。

```
# vgs
  VG   #PV #LV #SN Attr   VSize   VFree
  vg     2   0   0 wz--n- 271.31G 271.31G
```

ボリュームグループの全領域を使用して、ストライプを作成できます。

```
# lvcreate -n stripe1 -L 271.31G -i 2 vg
  Using default stripesize 64.00 KB
  Rounding up size to full physical extent 271.31 GB
  Logical volume "stripe1" created
# lvs -a -o +devices
  LV      VG   Attr   LSize   Origin Snap%  Move Log Copy%  Devices
  stripe1 vg   -wi-a- 271.31G                             /dev/sda1(0),/dev/sdb1(0)
```

ボリュームグループの空き領域がなくなっていることに注意してください。

```
# vgs
  VG   #PV #LV #SN Attr   VSize   VFree
  vg     2   1   0 wz--n- 271.31G    0
```

以下のコマンドで、ボリュームグループに物理ボリュームをもう 1 つ追加します。これで、135 ギガバイトの領域が追加されます。

```
# vgextend vg /dev/sdc1
  Volume group "vg" successfully extended
# vgs
  VG   #PV #LV #SN Attr   VSize   VFree
  vg     3   1   0 wz--n- 406.97G 135.66G
```

この時点では、ストライプ化論理ボリュームを、ボリュームグループの最大サイズまで拡大することはできません。データをストライプ化するには、2 つの物理デバイスが必要です。

```
# lvextend vg/stripe1 -L 406G
  Using stripesize of last segment 64.00 KB
  Extending logical volume stripe1 to 406.00 GB
```

```
  Insufficient suitable allocatable extents for logical volume stripe1: 34480
more required
```

ストライプ化論理ボリュームを拡張するには、もう 1 つの物理ボリュームを追加してから、論理ボリュームを拡張します。この例では、ボリュームグループに物理ボリュームを 2 つ追加することにより、ボリュームグループの最大サイズまで、論理ボリュームを拡張できるようになっています。

```
# vgextend vg /dev/sdd1
  Volume group "vg" successfully extended
# vgs
  VG   #PV #LV #SN Attr   VSize   VFree
  vg     4   1   0 wz--n- 542.62G 271.31G
# lvextend vg/stripe1 -L 542G
  Using stripesize of last segment 64.00 KB
  Extending logical volume stripe1 to 542.00 GB
  Logical volume stripe1 successfully resized
```

ストライプ化論理ボリュームを拡張するのに十分な物理デバイスがない場合でも、その拡張部分がストライプ化されなくても問題がないならば、ボリュームの拡張は可能です。ただし、これによりパフォーマンスが一定ではなくなる可能性があります。論理ボリュームに領域を追加する場合、デフォルトの動作では、既存論理ボリュームの最後のセグメントと同じストライピングパラメーターを使用するようになっていますが、このパラメーターはオーバーライドできます。以下の例では、初回の **lvextend** コマンドが失敗した後に、既存のストライプ化論理ボリュームを拡張して残りの空き領域を使用するようにしています。

```
# lvextend vg/stripe1 -L 406G
  Using stripesize of last segment 64.00 KB
  Extending logical volume stripe1 to 406.00 GB
  Insufficient suitable allocatable extents for logical volume stripe1: 34480
more required
# lvextend -i1 -l+100%FREE vg/stripe1
```

第4章 LVM 物理ボリュームの管理

LVM 物理ボリュームを管理するために使用できるコマンドや手順には、さまざまなものがあります。

4.1. 物理ボリュームとして使用するブロックデバイスのスキャン

以下の例のように、**lvmdiskscan** コマンドを使用して、物理ボリュームに使用できるブロックデバイスをスキャンできます。

```
# lvmdiskscan
  /dev/ram0                 [       16.00 MB]
  /dev/sda                  [       17.15 GB]
  /dev/root                 [       13.69 GB]
  /dev/ram                  [       16.00 MB]
  /dev/sda1                 [       17.14 GB] LVM physical volume
  /dev/VolGroup00/LogVol01     [      512.00 MB]
  /dev/ram2                 [       16.00 MB]
  /dev/new_vg/lvol0            [       52.00 MB]
  /dev/ram3                 [       16.00 MB]
  /dev/pkl_new_vg/sparkie_lv   [        7.14 GB]
  /dev/ram4                 [       16.00 MB]
  /dev/ram5                 [       16.00 MB]
  /dev/ram6                 [       16.00 MB]
  /dev/ram7                 [       16.00 MB]
  /dev/ram8                 [       16.00 MB]
  /dev/ram9                 [       16.00 MB]
  /dev/ram10                [       16.00 MB]
  /dev/ram11                [       16.00 MB]
  /dev/ram12                [       16.00 MB]
  /dev/ram13                [       16.00 MB]
  /dev/ram14                [       16.00 MB]
  /dev/ram15                [       16.00 MB]
  /dev/sdb                  [       17.15 GB]
  /dev/sdb1                 [       17.14 GB] LVM physical volume
  /dev/sdc                  [       17.15 GB]
  /dev/sdc1                 [       17.14 GB] LVM physical volume
  /dev/sdd                  [       17.15 GB]
  /dev/sdd1                 [       17.14 GB] LVM physical volume
  7 disks
  17 partitions
  0 LVM physical volume whole disks
  4 LVM physical volumes
```

4.2. 物理ボリュームのパーティションタイプの設定

ディスクデバイス全体を物理ボリュームに使用している場合は、そのディスクにはパーティションテーブルを含めないでください。ディスクパーティションが DOS の場合は、**fdisk**、**cfdisk** などのコマンドを使用して、パーティション ID を 0x8e に設定している必要があります。ディスクデバイス全体に物理ボリュームがある場合はパーティションテーブルを消去する必要がありますが、このとき、そのディスクにあるデータはすべて効果的に破棄されます。以下のコマンドを使用すれば、最初のセクターをゼロで初期化し、既存のパーティションテーブルを削除できます。

```
# dd if=/dev/zero of=PhysicalVolume bs=512 count=1
```

4.3. LVM 物理ボリュームのサイズ変更

なんらかの理由で基礎となるブロックデバイスのサイズを変更する必要がある場合は、**pvresize** コマンドを使用して LVM のサイズを更新します。このコマンドは、LVM が物理ボリュームを使用しているときに実行できます。

4.4. 物理ボリュームの削除

デバイスを LVM で使用する必要がなくなったら、**pvremove** コマンドを使用して LVM ラベルを削除できます。**pvremove** コマンドを実行すると、空の物理ボリュームにある LVM メタデータをゼロにします。

削除する物理ボリュームがボリュームグループの一部になっている場合は、**vgreduce** コマンドで、ボリュームグループから物理ボリュームを取り除く必要があります。

```
# pvremove /dev/ram15
  Labels on physical volume "/dev/ram15" successfully wiped
```

4.5. ボリュームグループへの物理ボリュームの追加

物理ボリュームを既存ボリュームグループに追加するには、**vgextend** コマンドを使用します。**vgextend** コマンドは、空き物理ボリュームを 1 つまたは複数追加して、ボリュームグループの容量を増やします。

以下のコマンドは、物理ボリューム **/dev/sdf1** をボリュームグループ **vg1** に追加します。

```
# vgextend vg1 /dev/sdf1
```

4.6. ボリュームグループからの物理ボリュームの削除

ボリュームグループから未使用の物理ボリュームを削除するには、**vgreduce** コマンドを使用します。**vgreduce** コマンドは、空の物理ボリュームを 1 つまたは複数削除して、ボリュームグループの容量を縮小します。これにより、物理ボリュームが解放され、異なるボリュームグループで使用したり、システムから削除できるようになります。

ボリュームグループから物理ボリュームを削除する前に、**pvdisplay** コマンドを使用して、その物理ボリュームが論理ボリュームで使用されていないことを確認できます。

```
# pvdisplay /dev/hda1

-- Physical volume ---
PV Name               /dev/hda1
VG Name               myvg
PV Size               1.95 GB / NOT usable 4 MB [LVM: 122 KB]
PV#                   1
PV Status             available
Allocatable           yes (but full)
Cur LV                1
PE Size (KByte)       4096
Total PE              499
Free PE               0
Allocated PE          499
PV UUID               Sd44tK-9IRw-SrMC-MOkn-76iP-iftz-OVSen7
```

物理ボリュームが使用中の場合は、**pvmove** コマンドを使用して、データを別の物理ボリュームに移行する必要があります。その後、**vgreduce** コマンドを使用してその物理ボリュームを削除します。

以下のコマンドは、物理ボリューム **/dev/hda1** を、ボリュームグループ **my_volume_group** から取り除きます。

```
# vgreduce my_volume_group /dev/hda1
```

論理ボリュームに、障害のある物理ボリュームが含まれる場合は、その論理ボリュームを使用することはできません。見つからない物理ボリュームをボリュームグループから削除します。その物理ボリュームに論理ボリュームが割り当てられていない場合は、**vgreduce** コマンドの **--removemissing** パラメーターを使用できます。

障害が発生した物理ボリュームに、セグメントタイプが **mirror** の論理ボリュームのミラーイメージが含まれる場合は、**vgreduce --removemissing --mirrorsonly --force** コマンドを使用して、ミラーからイメージを削除できます。これにより、物理ボリュームのミラーイメージである論理ボリュームのみが削除されます。

第5章 LVM コンポーネントの表示

LVM は、LVM コンポーネントを表示し、その表示をカスタマイズするさまざまな方法を提供します。本セクションでは、基本的な LVM 表示コマンドの使用をまとめています。

5.1. LVM コマンドによる LVM 情報の表示

lvm コマンドは、LVM サポートおよび設定に関する情報を表示するのに使用できる、いくつかのビルトインオプションを提供します。

- **lvm devtypes**
  認識されている組み込みブロックデバイスを表示します。
- **lvm formats**
  認識されているメタデータ形式を表示します。
- **lvm help**
  LVM ヘルプテキストを表示します。
- **lvm segtypes**
  認識されている論理ボリュームセグメントタイプを表示します。
- **lvm tags**
  このホストに定義したタグを表示します。
- **lvm version**
  現在のバージョン情報を表示します。

5.2. 物理ボリュームの表示

LVM 物理ボリュームのプロパティーを表示するのに使用できるコマンドは、**pvs**、**pvdisplay**、および **pvscan** の 3 つです。

pvs コマンドは、物理ボリュームの情報を設定可能な形式で提供し、1 物理ボリュームにつき 1 行ずつ表示します。**pvs** コマンドでは、形式をかなり自由に制御できるため、スクリプト作成時に役に立ちます。

pvdisplay コマンドは、各物理ボリュームの詳細をそれぞれ複数行出力します。物理プロパティー (サイズ、エクステント、ボリュームグループなど) が、決められた形式で表示されます。

以下の例は、1 つの物理ボリュームについて、**pvdisplay** コマンドで出力した情報です。

```
# pvdisplay
  --- Physical volume ---
  PV Name               /dev/sdc1
  VG Name               new_vg
  PV Size               17.14 GB / not usable 3.40 MB
  Allocatable           yes
  PE Size (KByte)       4096
  Total PE              4388
  Free PE               4375
  Allocated PE          13
  PV UUID               Joqlch-yWSj-kuEn-IdwM-01S9-XO8M-mcpsVe
```

pvscan コマンドは、システムにある、物理ボリュームでサポートされている LVM ブロックデバイスをすべてスキャンします。

以下のコマンドでは、検出された物理デバイスがすべて表示されます。

```
# pvscan
 PV /dev/sdb2   VG vg0   lvm2 [964.00 MB / 0   free]
 PV /dev/sdc1   VG vg0   lvm2 [964.00 MB / 428.00 MB free]
 PV /dev/sdc2            lvm2 [964.84 MB]
 Total: 3 [2.83 GB] / in use: 2 [1.88 GB] / in no VG: 1 [964.84 MB]
```

このコマンドが、特定の物理ボリュームをスキャンしないように、**lvm.conf** ファイルにフィルターを定義できます。

5.3. ボリュームグループの表示

LVM ボリュームグループのプロパティーを表示するのに使用できるコマンドは 2 つあります。**vgs** および **vgdisplay** です。

vgs コマンドは、ボリュームグループの情報を設定可能な形式で提供し、1 ボリュームグループにつき 1 行ずつ表示します。**vgs** コマンドでは、形式をかなり自由に制御できるため、スクリプト作成時に役に立ちます。

vgdisplay コマンドは、決められた形式でボリュームグループのプロパティー (サイズ、エクステント、物理ボリュームの数など) を表示します。以下の例は、ボリュームグループ **new_vg** に対する **vgdisplay** コマンドの出力を示しています。ボリュームグループを指定しないと、既存のボリュームグループがすべて表示されます。

```
# vgdisplay new_vg
  --- Volume group ---
  VG Name               new_vg
  System ID
  Format                lvm2
  Metadata Areas        3
  Metadata Sequence No  11
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               0
  Max PV                0
  Cur PV                3
  Act PV                3
  VG Size               51.42 GB
  PE Size               4.00 MB
  Total PE              13164
  Alloc PE / Size       13 / 52.00 MB
  Free  PE / Size       13151 / 51.37 GB
  VG UUID               jxQJ0a-ZKk0-OpMO-0118-nlwO-wwqd-fD5D32
```

5.4. 論理ボリュームの表示

LVM 論理ボリュームのプロパティーを表示するのに使用できるコマンドは、**lvs**、**lvdisplay**、および **lvscan** の 3 つです。

lvs コマンドは、論理ボリュームの情報を設定可能な形式で提供し、1論理ボリュームにつき1行ずつ表示します。**lvs** コマンドでは、形式をかなり自由に制御できるため、スクリプト作成時に役に立ちます。

lvdisplay コマンドは、決められた形式で、論理ボリュームのプロパティー (サイズ、レイアウト、マッピングなど) を表示します。

以下のコマンドは、**vg00** 内にある **lvol2** の属性を示しています。スナップショット論理ボリュームがこの元の論理ボリューム用に作成されている場合、このコマンドはすべてのスナップショット論理ボリュームとそのステータス (アクティブまたは非アクティブ) の一覧を表示します。

```
# lvdisplay -v /dev/vg00/lvol2
```

lvscan コマンドは、システム内のすべての論理ボリュームをスキャンし、以下の例のようにそれらを一覧表示します。

```
# lvscan
  ACTIVE            '/dev/vg0/gfslv' [1.46 GB] inherit
```

第6章 LVM 用のカスタム報告

LVM では、カスタマイズされたレポートを生成したり、レポートの出力をフィルタリングしたりするためのさまざまな設定およびコマンドラインオプションが提供されます。LVM レポート機能の完全な説明は、man ページの **lvmreport**(7) を参照してください。

pvs、**lvs**、および **vgs** コマンドを使用して、LVM オブジェクトについての簡潔でカスタマイズ可能なレポートを作成できます。これらのコマンドが生成するレポートには、オブジェクトごとに 1 行の出力が含まれます。各行には、オブジェクトに関連するプロパティーのフィールドの順序付けられた一覧が含まれます。レポートするオブジェクトを選択する方法には、物理ボリューム、ボリュームグループ、論理ボリューム、物理ボリュームセグメント、および論理ボリュームセグメント別の 5 つの方法があります。

lvm fullreport コマンドを使用して、物理ボリューム、ボリュームグループ、論理ボリューム、物理ボリュームセグメント、および論理ボリュームセグメントに関する情報を一度に報告できます。このコマンドとその機能については、man ページの **lvm-fullreport**(8) を参照してください。

LVM は、LVM コマンドの実行中に収集された操作、メッセージ、およびオブジェクトごとのステータス (完全なオブジェクト ID 付き) が含まれるメッセージログレポートをサポートします。LVM ログレポートの詳細は、man ページの **lvmreport**(7) を参照してください。

6.1. LVM 表示の形式の制御

pvs、**lvs**、または **vgs** コマンドのどれを使用するかによって、表示されるデフォルトのフィールドセットとソート順序が決定されます。これらのコマンドの出力は、以下の引数を使用して制御できます。

- **-o** 引数を使用すると、表示するフィールドをデフォルト以外に変更できます。たとえば、以下の出力は、物理ボリュームの名前とサイズのみを表示します。

```
# pvs -o pv_name,pv_size
PV PSize
/dev/sdb1 17.14G
/dev/sdc1 17.14G
/dev/sdd1 17.14G
```

- -o 引数との組み合わせで使用するプラス記号 (+) を使用して、出力にフィールドを追加できます。
  以下の例は、デフォルトフィールドに加えて、物理ボリュームの UUID を表示しています。

```
# pvs -o +pv_uuid
PV VG Fmt Attr PSize PFree PV UUID
/dev/sdb1 new_vg lvm2 a- 17.14G 17.14G onFF2w-1fLC-ughJ-D9eB-M7iv-6XqA-dqGeXY
/dev/sdc1 new_vg lvm2 a- 17.14G 17.09G Joqlch-yWSj-kuEn-IdwM-01S9-X08M-mcpsVe
/dev/sdd1 new_vg lvm2 a- 17.14G 17.14G yvfvZK-Cf31-j75k-dECm-0RZ3-0dGW-UqkCS
```

- コマンドに **-v** 引数を追加すると、追加のフィールドが含まれます。たとえば、**pvs -v** コマンドは、デフォルトフィールドに加えて、**DevSize** と **PV UUID** のフィールドも表示します。

```
# pvs -v
Scanning for physical volume names
PV VG Fmt Attr PSize PFree DevSize PV UUID
/dev/sdb1 new_vg lvm2 a- 17.14G 17.14G 17.14G onFF2w-1fLC-ughJ-D9eB-M7iv-6XqA-
dqGeXY
```

```
/dev/sdc1 new_vg lvm2 a- 17.14G 17.09G 17.14G Joqlch-yWSj-kuEn-ldwM-01S9-XO8M-mcpsVe
/dev/sdd1 new_vg lvm2 a- 17.14G 17.14G 17.14G yvfvZK-Cf31-j75k-dECm-0RZ3-0dGW-tUqkCS
```

- **--noheadings** 引数は、見出し行を表示しません。これはスクリプトを作成する際に便利です。以下の例は、**pv_name** 引数と共に **--noheadings** 引数を使用して、すべての物理ボリュームの一覧を生成しています。

```
# pvs --noheadings -o pv_name
/dev/sdb1
/dev/sdc1
/dev/sdd1
```

- **--separator separator** 引数は、**区切り文字** を使用して、各フィールドを区切ります。次の例は、**pvs** コマンドのデフォルト出力フィールドを等号 (=) で分割しています。

```
# pvs --separator =
PV=VG=Fmt=Attr=PSize=PFree
/dev/sdb1=new_vg=lvm2=a-=17.14G=17.14G
/dev/sdc1=new_vg=lvm2=a-=17.14G=17.09G
/dev/sdd1=new_vg=lvm2=a-=17.14G=17.14G
```

separator 引数の使用時にフィールドを配置するには、**--aligned** 引数とともに **separator** 引数を使用します。

```
# pvs --separator = --aligned
PV =VG =Fmt =Attr=PSize =PFree
/dev/sdb1 =new_vg=lvm2=a- =17.14G=17.14G
/dev/sdc1 =new_vg=lvm2=a- =17.14G=17.09G
/dev/sdd1 =new_vg=lvm2=a- =17.14G=17.14G
```

lvs コマンドまたは **vgs** コマンドの **-P** 引数を使用して、通常の出力では表示されない、障害が発生したボリュームの情報を表示します

表示引数の完全なリストは、man ページの **pvs**(8)、**vgs**(8)、および **lvs**(8) を参照してください。

ボリュームグループフィールドは、物理ボリューム (および物理ボリュームセグメント) フィールド、または論理ボリューム (および論理ボリュームセグメント) フィールドと混在させることができますが、物理ボリュームフィールドと論理ボリュームフィールドは混在させることはできません。たとえば、以下のコマンドは、1つの物理ボリュームつき1行の出力を表示します。

```
# vgs -o +pv_name
  VG     #PV #LV #SN Attr   VSize  VFree  PV
  new_vg   3   1   0 wz--n- 51.42G 51.37G /dev/sdc1
  new_vg   3   1   0 wz--n- 51.42G 51.37G /dev/sdd1
  new_vg   3   1   0 wz--n- 51.42G 51.37G /dev/sdb1
```

6.2. LVM オブジェクト表示フィールド

このセクションでは、コマンド **pvs**、**vgs**、および **lvs** を使用して、LVM オブジェクトについて表示できる情報を一覧表示する一連の表を提供します。

便宜上、フィールド名の接頭辞は、コマンドのデフォルトと一致する場合は省略できます。たとえば、**pvs** コマンドでは、**name** は **pv_name**、**vgs** コマンドでは、**name** は **vg_name** と解釈されます。

以下のコマンドの実行は、**pvs -o pv_free** の実行に相当します。

```
# pvs -o free
  PFree
  17.14G
  17.09G
  17.14G
```

注記

pvs、**vgs**、および **lvs** 出力の属性フィールドにある文字数は、以降のリリースで増える可能性があります。既存の文字フィールドの位置は変更しませんが、新しいフィールドが末尾に追加される可能性があります。相対的な位置を使用して特定の属性文字を検索するスクリプトを作成する場合は、このことを考慮して、フィールドの終点ではなく、フィールドの始点を基点として文字検索を行います。たとえば、**lv_attr** フィールドの 9 番目のビットの文字 **p** を検索する場合は、文字列 "^/.....p/" で指定できます。文字列 "/*p$/" は使用しないでください。

表6.1「pvs コマンド表示フィールド」 は、**pvs** コマンドの表示引数、ヘッダーに表示されるフィールド名、フィールドの説明を一覧にまとめています。

表6.1 pvs コマンド表示フィールド

引数	ヘッダー	説明
dev_size	DevSize	物理ボリュームを作成する基となる配下のデバイスのサイズ
pe_start	1st PE	配下のデバイス内の最初の物理エクステントの開始点までのオフセット
pv_attr	Attr	物理ボリュームのステータス - (a)llocatable または e(x)ported
pv_fmt	Fmt	物理ボリュームのメタデータ形式 (**lvm2** または **lvm1**)
pv_free	PFree	物理ボリュームにある残りの空き領域
pv_name	PV	物理ボリュームの名前
pv_pe_alloc_count	Alloc	使用される物理エクステントの数
pv_pe_count	PE	物理エクステントの数
pvseg_size	SSize	物理ボリュームのセグメントサイズ
pvseg_start	Start	物理ボリュームセグメントの最初の物理エクステント

引数	ヘッダー	説明
pv_size	PSize	物理ボリュームのサイズ
pv_tags	PV Tags	物理ボリュームに割り当てられた LVM タグ
pv_used	Used	物理ボリューム上で現在使用中の領域の量
pv_uuid	PV UUID	物理ボリュームの UUID

デフォルトで **pvs** コマンドが表示するフィールドは、**pv_name**、**vg_name**、**pv_fmt**、**pv_attr**、**pv_size**、および **pv_free** です。この表示は、**pv_name** でソートされます。

```
# pvs
  PV         VG     Fmt  Attr PSize  PFree
  /dev/sdb1  new_vg lvm2 a-   17.14G 17.14G
  /dev/sdc1  new_vg lvm2 a-   17.14G 17.09G
  /dev/sdd1  new_vg lvm2 a-   17.14G 17.13G
```

pvs コマンドに **-v** 引数を使用すると、デフォルトの表示に、**dev_size** フィールドおよび **pv_uuid** フィールドが追加されます。

```
# pvs -v
    Scanning for physical volume names
  PV         VG     Fmt  Attr PSize  PFree  DevSize PV UUID
  /dev/sdb1  new_vg lvm2 a-   17.14G 17.14G  17.14G onFF2w-1fLC-ughJ-D9eB-M7iv-6XqA-
dqGeXY
  /dev/sdc1  new_vg lvm2 a-   17.14G 17.09G  17.14G Joqlch-yWSj-kuEn-IdwM-01S9-XO8M-mcpsVe
  /dev/sdd1  new_vg lvm2 a-   17.14G 17.13G  17.14G yvfvZK-Cf31-j75k-dECm-0RZ3-0dGW-tUqkCS
```

pvs コマンドに **--segments** 引数を使用すると、各物理ボリュームセグメントの情報を表示します。セグメントはエクステントの集合です。セグメントの表示は、論理ボリュームがフラグメント化 (断片化) しているかどうかを確認するのに役立ちます。

デフォルトで **pvs --segments** コマンドが表示するフィールドは、**pv_name**、**vg_name**、**pv_fmt**、**pv_attr**、**pv_size**、**pv_free**、**pvseg_start**、および **pvseg_size** です。この表示は、物理ボリューム内では **pv_name** および **pvseg_size** でソートされます。

```
# pvs --segments
  PV         VG         Fmt  Attr PSize  PFree  Start SSize
  /dev/hda2  VolGroup00 lvm2 a-   37.16G 32.00M     0  1172
  /dev/hda2  VolGroup00 lvm2 a-   37.16G 32.00M  1172    16
  /dev/hda2  VolGroup00 lvm2 a-   37.16G 32.00M  1188     1
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G     0    26
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G    26    24
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G    50    26
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G    76    24
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G   100    26
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G   126    24
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G   150    22
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G   172  4217
```

```
  /dev/sdb1  vg       lvm2 a-   17.14G 17.14G     0  4389
  /dev/sdc1  vg       lvm2 a-   17.14G 17.14G     0  4389
  /dev/sdd1  vg       lvm2 a-   17.14G 17.14G     0  4389
  /dev/sde1  vg       lvm2 a-   17.14G 17.14G     0  4389
  /dev/sdf1  vg       lvm2 a-   17.14G 17.14G     0  4389
  /dev/sdg1  vg       lvm2 a-   17.14G 17.14G     0  4389
```

pvs -a コマンドを使用して、LVM が検出した、LVM 物理ボリュームとして初期化していないデバイスを確認できます。

```
# pvs -a
  PV                         VG     Fmt  Attr PSize  PFree
  /dev/VolGroup00/LogVol01                --       0      0
  /dev/new_vg/lvol0                       --       0      0
  /dev/ram                                --       0      0
  /dev/ram0                               --       0      0
  /dev/ram2                               --       0      0
  /dev/ram3                               --       0      0
  /dev/ram4                               --       0      0
  /dev/ram5                               --       0      0
  /dev/ram6                               --       0      0
  /dev/root                               --       0      0
  /dev/sda                                --       0      0
  /dev/sdb                                --       0      0
  /dev/sdb1                  new_vg lvm2 a-   17.14G 17.14G
  /dev/sdc                                --       0      0
  /dev/sdc1                  new_vg lvm2 a-   17.14G 17.09G
  /dev/sdd                                --       0      0
  /dev/sdd1                  new_vg lvm2 a-   17.14G 17.14G
```

表6.2「vgs 表示フィールド」 は、**vgs** コマンドの表示引数、ヘッダーに表示されるフィールド名、およびフィールドの説明を一覧にまとめています。

表6.2 vgs 表示フィールド

引数	ヘッダー	説明
lv_count	#LV	ボリュームグループに含まれる論理ボリュームの数
max_lv	MaxLV	ボリュームグループで許容される論理ボリュームの最大数 (無制限の場合は 0)
max_pv	MaxPV	ボリュームグループで許容される物理ボリュームの最大数 (無制限の場合は 0)
pv_count	#PV	ボリュームグループを定義する物理ボリューム数
snap_count	#SN	ボリュームグループに含まれるスナップショット数
vg_attr	Attr	ボリュームグループのステータス - (w)riteable (書き込み可能)、(r)eadonly (読み取りのみ)、resi(z)eable (サイズ変更可能)、e(x)ported (エクスポート済)、(p)artial (部分的)、および (c)lustered (クラスター化)

引数	ヘッダー	説明
vg_extent_count	#Ext	ボリュームグループの物理エクステントの数
vg_extent_size	Ext	ボリュームグループの物理エクステントのサイズ
vg_fmt	Fmt	ボリュームグループのメタデータ形式 (**lvm2** または **lvm1**)
vg_free	VFree	ボリュームグループの残りの空き領域のサイズ
vg_free_count	Free	ボリュームグループの空き物理エクステントの数
vg_name	VG	ボリュームグループ名
vg_seqno	Seq	ボリュームグループの改訂を示す番号
vg_size	VSize	ボリュームグループのサイズ
vg_sysid	SYS ID	LVM1 システム ID
vg_tags	VG Tags	ボリュームグループに割り当てられた LVM タグ
vg_uuid	VG UUID	ボリュームグループの UUID

デフォルトで **vgs** コマンドが表示するフィールドは、**vg_name**、**pv_count**、**lv_count**、**snap_count**、**vg_attr**、**vg_size**、および **vg_free** です。この表示は、**vg_name** でソートされます。

```
# vgs
  VG     #PV #LV #SN Attr   VSize  VFree
  new_vg   3   1   1 wz--n- 51.42G 51.36G
```

vgs コマンドに **-v** 引数を使用すると、デフォルトの表示に **vg_extent_size** および **vg_uuid** のフィールドが追加されます。

```
# vgs -v
    Finding all volume groups
    Finding volume group "new_vg"
  VG     Attr   Ext   #PV #LV #SN VSize  VFree  VG UUID
  new_vg wz--n- 4.00M   3   1   1 51.42G 51.36G jxQJ0a-ZKk0-OpMO-0118-nlwO-wwqd-fD5D32
```

表6.3「lvs 表示フィールド」 は、**lvs** コマンドの表示引数、ヘッダーに表示されるフィールド名、およびフィールドの説明を一覧にまとめています。

注記

Red Hat Enterprise Linux の最近のリリースでは、**lvs** コマンドの出力に追加のフィールドが表示される場合がありますが、フィールドの順番は変わらず、追加のフィールドは最後に表示されます。

表6.3 lvs 表示フィールド

引数	ヘッダー	説明
* **chunksize** * **chunk_size**	Chunk	スナップショットボリュームのユニットサイズ
copy_percent	Copy%	ミラー化論理ボリュームの同期のパーセンテージ。さらに **pv_move** コマンドで物理エクステントを移動する時にも使用されます。
devices	Devices	論理ボリュームを構成する配下のデバイス - 物理ボリューム、論理ボリューム、および物理エクステントと論理エクステントの開始点
lv_ancestors	Ancestors	シンプールスナップショットにおける、論理ボリュームの先祖 (ancestor)
lv_descendants	Descendants	シンプールスナップショットにおける、論理ボリュームの子孫 (descendant)
lv_attr	Attr	論理ボリュームのステータス。論理ボリュームの属性ビットは以下のようになります。 * ビット 1 - ボリュームタイプ - (m)irrored (ミラー化)、(M)irrored without initial sync (初期同期なしのミラー化)、(o)rigin (複製元)、(O)rigin with merging snapshot (マージするスナップショットがある複製元)、(r)aid (RAID)、(R)aid without initial sync (初期同期なしの RAID)、(s)napshot (スナップショット)、merging (S)napshot (マージするスナップショット)、(p)vmove (物理ボリュームの移動)、(v)irtual (仮想)、mirror or raid (i)mage (ミラーまたは RAID イメージ)、mirror or raid (I)mage out-of-sync (ミラーまたは RAID イメージの非同期)、mirror (l)og device (ミラーログデバイス)、under (c)onversion (変換中)、thin (V)olume (シンボリューム)、(t)hin pool (シンプール)、(T)hin pool data (シンプールデータ)、raid or thin pool m(e)tadata or pool metadata spare (RAID またはシンプールメタデータまたはプールメタデータのスペア) * ビット 2 - パーミッション - (w)riteable (書き込み可能)、(r)ead-only (読み取り専用)、(R)ead-only activation of non-read-only volume (読み取り専用でないボリュームを読み取り専用にアクティブ化) * ビット 3 - 割り当てポリシー - (a)nywhere (どこでも)、(c)ontiguous (連続的)、(i)nherited (継承)、c(l)ing (膠着)、(n)ormal (通常)。これは、たとえば **pvmove** コマンドの実行時など、割り当ての変更に対してボリュームが現在ロックされている場合に大文字になります。 * ビット 4 - 固定されたマイナー番号 * ビット 5 - 状態 - (a)ctive (アクティブ)、(s)uspended (サス

引数	ヘッダー	説明
		ペンド)、(I)nvalid snapshot (無効なスナップショット)、(S)uspended snapshot (無効なサスペンドされたスナップショット)、snapshot (m)erge failed (スナップショットのマージが失敗)、suspended snapshot (M)erge failed (サスペンドされたスナップショットのマージが失敗)、mapped (d)evice present without tables (テーブルのないマッピングされたデバイス)、mapped device present with (i)nactive table (非アクティブのテーブルを持つマッピングされたデバイス) * ビット 6 - デバイス開放(o) * ビット 7 - ターゲットタイプ - (m)irror (ミラー)、(r)aid (RAID)、(s)napshot (スナップショット)、(t)hin (シン)、(u)nknown (不明)、(v)irtual (仮想)。これは、同じターゲットのカーネルに関連する論理ボリュームをグループにまとめます。たとえば、ミラーイメージ、ミラーログ、ミラー自体が元のデバイスマッパーのミラーカーネルドライバーを使用する場合、それらは (m) と表示されます。md raid カーネルドライバーを使用する同等の raid はすべて (r) と表示されます。元のデバイスマッパードライバーを使用するスナップショットは (s) と表示され、シンプロビジョニングドライバーを使用するシンボリュームのスナップショットは (t) と表示されます。 * ビット 8 - 新しく割り当てられたデータブロックは使用前に、ゼロ(z) のブロックで上書きされます。 * ビット 9 - ボリュームの正常性 - (p)artial (部分的)、(r)efresh needed (更新が必要)、(m)ismatches exist (不一致が存在)、(w)ritemostly (書き込み多発)。部分的 (p) は、この論理ボリュームが使用する 1 つ以上の物理ボリュームがシステムから欠落していることを表します。更新 (r) は、この RAID 論理ボリュームが使用する 1 つ以上の物理ボリュームが書き込みエラーが生じたことを表します。書き込みエラーは、その物理ボリュームの一時的な障害により引き起こされたか、または物理ボリュームに障害があることを示すかのいずれかの可能性があります。デバイスは更新するか、または置き換える必要があります。不一致 (m) は、RAID 論理ボリュームのアレイに一貫していない部分があることを表します。不整合は、RAID 論理ボリューム上で **check** 操作を開始すると検出されます。(スクラビング操作 **check** および **repair** は、**lvchange** コマンドにより RAID 論理ボリューム上で実行できます。) 書き込み多発 (w) は write-mostly とマークが付けられた RAID 1 論理ボリュームのデバイスを表します。 * ビット 10 - s(k)ip activation (アクティブ化のスキップ - このボリュームには、アクティブ化の実行時にスキップされるようにフラグが設定されます。
lv_kernel_major	KMaj	論理ボリュームの実際のメジャーデバイス番号 (非アクティブの場合は -1)

引数	ヘッダー	説明
lv_kernel_minor	KMIN	論理ボリュームの実際のマイナーデバイス番号 (非アクティブの場合は -1)
lv_major	Maj	論理ボリュームの永続的なメジャーデバイス番号 (未指定の場合は -1)
lv_minor	Min	論理ボリュームの永続的なマイナーデバイス番号 (未指定の場合は -1)
lv_name	LV	論理ボリュームの名前
lv_size	LSize	論理ボリュームのサイズ
lv_tags	LV Tags	論理ボリュームに割り当てられた LVM タグ
lv_uuid	LV UUID	論理ボリュームの UUID
mirror_log	Log	ミラーログが存在するデバイス
modules	モジュール	この論理ボリュームを使用するのに必要な対応するカーネルデバイスマッパーターゲット
move_pv	Move	**pvmove** コマンドで作成された一時的な論理ボリュームの元となる物理ボリューム
origin	Origin	スナップショットボリュームの複製元のデバイス
* **regionsize** * **region_size**	Region	ミラー化論理ボリュームのユニットサイズ
seg_count	#Seg	論理ボリュームのセグメント数
seg_size	SSize	論理ボリュームのセグメントサイズ
seg_start	Start	論理ボリュームのセグメントのオフセット
seg_tags	Seg Tags	論理ボリュームのセグメントに割り当てられた LVM タグ
segtype	タイプ	論理ボリュームのセグメントタイプ (例: ミラー、ストライプ、リニア)
snap_percent	Snap%	使用中スナップショットボリュームの現在のパーセンテージ
stripes	#Str	論理ボリュームのストライプ、またはミラーの数

引数	ヘッダー	説明
* **stripesize** * **stripe_size**	Stripe	ストライプ化論理ボリュームのストライプのユニットサイズ

デフォルトで **lvs** コマンドが表示するのは以下になります。デフォルトの表示は、ボリュームグループ内では **vg_name** および **lv_name** でソートされます。

```
# lvs
  LV     VG            Attr       LSize    Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  origin VG            owi-a-s---    1.00g
  snap   VG            swi-a-s---  100.00m      origin 0.00
```

lvs コマンドの一般的な用途は、論理ボリュームを構成する基本的なデバイスを表示するコマンドに、**devices** を追加することです。また、この例では、**-a** オプションを指定して、RAID ミラーなどの論理ボリュームのコンポーネントである内部ボリュームを、括弧で囲んで表示します。この例には、RAID ボリューム、ストライプのボリューム、シンプールのボリュームが含まれます。

```
# lvs -a -o +devices
  LV              VG           Attr       LSize   Pool   Origin Data%  Meta%  Move Log Cpy%Sync Convert
Devices
  raid1           VG           rwi-a-r---   1.00g                                    100.00
raid1_rimage_0(0),raid1_rimage_1(0)
  [raid1_rimage_0] VG          iwi-aor---   1.00g                                                  /dev/sde1(7041)
  [raid1_rimage_1] VG          iwi-aor---   1.00g                                                  /dev/sdf1(7041)
  [raid1_rmeta_0]  VG          ewi-aor---   4.00m                                                   /dev/sde1(7040)
  [raid1_rmeta_1]  VG          ewi-aor---   4.00m                                                   /dev/sdf1(7040)
  stripe1         VG           -wi-a-----  99.95g                                                 /dev/sde1(0),/dev/sdf1(0)
  stripe1         VG           -wi-a-----  99.95g                                                 /dev/sdd1(0)
  stripe1         VG           -wi-a-----  99.95g                                                 /dev/sdc1(0)
  [lvol0_pmspare]  rhel_host-083 ewi-------   4.00m                                                    /dev/vda2(0)
  pool00          rhel_host-083 twi-aotz--  <4.79g              72.90  54.69
pool00_tdata(0)
  [pool00_tdata]   rhel_host-083 Twi-ao----  <4.79g                                                   /dev/vda2(1)
  [pool00_tmeta]   rhel_host-083 ewi-ao----   4.00m                                                   /dev/vda2(1226)
  root            rhel_host-083 Vwi-aotz--  <4.79g pool00        72.90
  swap            rhel_host-083 -wi-ao---- 820.00m                                                  /dev/vda2(1227)
```

lvs コマンドで **-v** 引数を使用して、デフォルトの表示に、**seg_count**、**lv_major**、**lv_minor**、**lv_kernel_major**、**lv_kernel_minor**、**lv_uuid** のフィールドを追加します。

```
# lvs -v
    Finding all logical volumes
  LV        VG     #Seg Attr   LSize  Maj Min KMaj KMin Origin Snap%  Move Copy%  Log Convert LV
UUID
  lvol0     new_vg    1 owi-a- 52.00M  -1  -1 253  3                                       LBy1Tz-sr23-OjsI-LT03-
nHLC-y8XW-EhCl78
  newvgsnap1 new_vg    1 swi-a-  8.00M  -1  -1 253  5    lvol0    0.20                   1ye1OU-1clu-
o79k-20h2-ZGF0-qCJm-CfbsIx
```

デフォルトで **lvs --segments** コマンドが表示するフィールド

は、**lv_name**、**vg_name**、**lv_attr**、**stripes**、**segtype**、および **seg_size** です。デフォルトの表示は、ボリュームグループでは **vg_name**、**lv_name** でソートされ、論理ボリュームでは **seg_start** でソートされます。論理ボリュームがフラグメント化されている場合、このコマンドの出力は以下を表示します。

```
# lvs --segments
  LV       VG         Attr   #Str Type   SSize
  LogVol00 VolGroup00 -wi-ao    1 linear  36.62G
  LogVol01 VolGroup00 -wi-ao    1 linear 512.00M
  lv       vg         -wi-a-    1 linear 104.00M
  lv       vg         -wi-a-    1 linear 104.00M
  lv       vg         -wi-a-    1 linear 104.00M
  lv       vg         -wi-a-    1 linear  88.00M
```

lvs --segments コマンドで **-v** 引数を使用すると、デフォルトの表示に **seg_start**、**stripesize**、**chunksize** のフィールドが追加されます。

```
# lvs -v --segments
    Finding all logical volumes
  LV         VG     Attr   Start SSize  #Str Type   Stripe Chunk
  lvol0      new_vg owi-a-    0  52.00M    1 linear     0     0
  newvgsnap1 new_vg swi-a-    0   8.00M    1 linear     0  8.00K
```

以下の1つ目の例は、設定された論理ボリュームが1つあるシステムで実行した **lvs** コマンドのデフォルト出力を示しています。その次の例は、**segments** 引数を指定した **lvs** コマンドのデフォルト出力を表示しています。

```
# lvs
  LV    VG     Attr   LSize  Origin Snap%  Move Log Copy%
  lvol0 new_vg -wi-a- 52.00M
# lvs --segments
  LV    VG     Attr   #Str Type   SSize
  lvol0 new_vg -wi-a-    1 linear 52.00M
```

6.3. LVM 報告のソート

通常、**lvs** コマンド、**vgs** コマンド、または **pvs** コマンドの出力全体をソートして、コラムを正しく配置するには、まずそれを生成して内部に保管する必要があります。**--unbuffered** 引数を指定すると、生成直後にソートされていないままの出力で表示できます。

別の順列のコラム一覧のソートを指定するには、報告コマンドのいずれかと一緒に **-O** 引数を使用します。出力自体の中にこれらのフィールドを含める必要はありません。

以下の例は、物理ボリュームの名前、サイズ、および空き領域を表示する **pvs** コマンドの出力を示しています。

```
# pvs -o pv_name,pv_size,pv_free
  PV         PSize  PFree
  /dev/sdb1  17.14G 17.14G
  /dev/sdc1  17.14G 17.09G
  /dev/sdd1  17.14G 17.14G
```

以下の例では、空き領域のフィールドでソートされた同じ出力を示しています。

```
# pvs -o pv_name,pv_size,pv_free -O pv_free
  PV         PSize  PFree
  /dev/sdc1  17.14G 17.09G
  /dev/sdd1  17.14G 17.14G
  /dev/sdb1  17.14G 17.14G
```

以下の例では、ソートするフィールドを表示する必要がないことを示しています。

```
# pvs -o pv_name,pv_size -O pv_free
  PV         PSize
  /dev/sdc1  17.14G
  /dev/sdd1  17.14G
  /dev/sdb1  17.14G
```

逆順でソートするには、**-O** 引数の後で指定するフィールドの先頭に **-** 印を付けます。

```
# pvs -o pv_name,pv_size,pv_free -O -pv_free
  PV         PSize  PFree
  /dev/sdd1  17.14G 17.14G
  /dev/sdb1  17.14G 17.14G
  /dev/sdc1  17.14G 17.09G
```

6.4. LVM レポート表示への単位の指定

LVM 報告表示用の単位を指定するには、報告コマンドに **--units** 引数を使用します。バイト(b)、キロバイト(k)、メガバイト(m)、ギガバイト(g)、テラバイト(t)、エクサバイト(e)、ペタバイト(p)、および人間が読める表示(h) を指定できます。デフォルトは人間が読める表示です。このデフォルト設定を上書きするには、**/etc/lvm/lvm.conf** ファイルの **global** セクション内の **units** パラメーターを設定します。

以下の例は、**pvs** コマンドの出力をデフォルトのギガバイトでなく、メガバイトで指定しています。

```
# pvs --units m
  PV         VG     Fmt  Attr PSize     PFree
  /dev/sda1         lvm2 --   17555.40M 17555.40M
  /dev/sdb1  new_vg lvm2 a-   17552.00M 17552.00M
  /dev/sdc1  new_vg lvm2 a-   17552.00M 17500.00M
  /dev/sdd1  new_vg lvm2 a-   17552.00M 17552.00M
```

デフォルトでは、単位は 2 の累乗 (1024 の倍数) で表示されます。単位を 1000 の倍数で表示するには、大文字 (B、K、M、G、T、H) で単位を指定できます。

以下のコマンドは、1024 の倍数 (デフォルト) で出力を表示します。

```
# pvs
  PV         VG     Fmt  Attr PSize  PFree
  /dev/sdb1  new_vg lvm2 a-   17.14G 17.14G
  /dev/sdc1  new_vg lvm2 a-   17.14G 17.09G
  /dev/sdd1  new_vg lvm2 a-   17.14G 17.14G
```

以下のコマンドは、1000 の倍数で出力を表示します。

```
#  pvs --units G
```

```
  PV         VG     Fmt  Attr PSize  PFree
  /dev/sdb1  new_vg lvm2 a-   18.40G 18.40G
  /dev/sdc1  new_vg lvm2 a-   18.40G 18.35G
  /dev/sdd1  new_vg lvm2 a-   18.40G 18.40G
```

セクター (512 バイトとして定義) またはカスタム単位も指定できます。

以下の例は、**pvs** コマンドの出力をセクター数として表示します。

```
# pvs --units s
  PV         VG     Fmt  Attr PSize     PFree
  /dev/sdb1  new_vg lvm2 a-   35946496S 35946496S
  /dev/sdc1  new_vg lvm2 a-   35946496S 35840000S
  /dev/sdd1  new_vg lvm2 a-   35946496S 35946496S
```

以下の例は、**pvs** コマンドの出力を 4 MB 単位で表示しています。

```
# pvs --units 4m
  PV         VG     Fmt  Attr PSize    PFree
  /dev/sdb1  new_vg lvm2 a-   4388.00U 4388.00U
  /dev/sdc1  new_vg lvm2 a-   4388.00U 4375.00U
  /dev/sdd1  new_vg lvm2 a-   4388.00U 4388.00U
```

6.5. JSON 形式で LVM コマンド結果の表示

LVM 表示コマンドで**--reportformat** オプションを使用して JSON 形式で出力を表示できます。

以下の例は、標準的なデフォルト形式の **lvs** の出力を示しています。

```
# lvs
  LV      VG            Attr       LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  my_raid my_vg         Rwi-a-r---  12.00m                                    100.00
  root    rhel_host-075 -wi-ao----   6.67g
  swap    rhel_host-075 -wi-ao---- 820.00m
```

以下のコマンドは、JSON 形式を指定する場合と同じ LVM 設定の出力を表示します。

```
# lvs --reportformat json
  {
      "report": [
          {
              "lv": [
                  {"lv_name":"my_raid", "vg_name":"my_vg", "lv_attr":"Rwi-a-r---", "lv_size":"12.00m",
"pool_lv":"", "origin":"", "data_percent":"", "metadata_percent":"", "move_pv":"", "mirror_log":"",
"copy_percent":"100.00", "convert_lv":""},
                  {"lv_name":"root", "vg_name":"rhel_host-075", "lv_attr":"-wi-ao----", "lv_size":"6.67g",
"pool_lv":"", "origin":"", "data_percent":"", "metadata_percent":"", "move_pv":"", "mirror_log":"",
"copy_percent":"", "convert_lv":""},
                  {"lv_name":"swap", "vg_name":"rhel_host-075", "lv_attr":"-wi-ao----", "lv_size":"820.00m",
"pool_lv":"", "origin":"", "data_percent":"", "metadata_percent":"", "move_pv":"", "mirror_log":"",
"copy_percent":"", "convert_lv":""}
              ]
```

```
        }
      ]
  }
```

また、**/etc/lvm/lvm.conf** ファイルで **output_format** 設定を使用して、レポート形式を設定オプションとして設定することもできます。ただし、コマンドラインの **--reportformat** 設定はこの設定よりも優先されます。

6.6. LVM コマンドログの表示

レポート指向および処理指向の LVM コマンドを使用して、コマンドログを報告できます (これが **log/report_command_log** 設定で有効になっている場合)。このレポートで表示およびソートするフィールドセットを決定できます。

以下の例では、LVM コマンド向けの完全なログレポートを生成するように LVM を設定します。この例では、論理ボリューム **lvol0** と **lvol1** の両方が、それらの論理ボリュームを含むボリュームグループ **VG** とともに正常に処理されたことを確認できます。

```
# lvmconfig --type full log/command_log_selection
command_log_selection="all"

# lvs
  Logical Volume
  ==============
  LV    LSize Cpy%Sync
  lvol1 4.00m 100.00
  lvol0 4.00m

  Command Log
  ===========
  Seq LogType Context    ObjType ObjName ObjGrp  Msg     Errno RetCode
    1 status  processing lv      lvol0   vg      success     0       1
    2 status  processing lv      lvol1   vg      success     0       1
    3 status  processing vg      vg              success     0       1

# lvchange -an vg/lvol1
  Command Log
  ===========
  Seq LogType Context    ObjType ObjName ObjGrp  Msg     Errno RetCode
    1 status  processing lv      lvol1   vg      success     0       1
    2 status  processing vg      vg              success     0       1
```

LVM レポートおよびコマンドログの設定の詳細については、man ページの **lvmreport** を参照してください。

第7章 RAID 論理ボリュームの設定

LVM は RAID0/1/4/5/6/10 をサポートします。

注記

RAID 論理ボリュームはクラスターには対応していません。RAID 論理ボリュームは 1 台のマシンに作成でき、かつ排他的にアクティブにできますが、複数のマシンで同時にアクティブにすることはできません。

RAID 論理ボリュームを作成するには、raid の種類を **lvcreate** コマンドの **--type** 引数として指定します。表7.1「RAID のセグメントタイプ」 では、可能な RAID セグメントのタイプを説明しています。LVM で RAID 論理ボリュームを作成した後、他の LVM 論理ボリュームと同じようにボリュームをアクティブ化、変更、削除、表示、および使用することができます。

表7.1 RAID のセグメントタイプ

セグメントタイプ	説明
raid1	RAID1 ミラーリング。**-m** を指定し、ストライピングを指定しない場合、これは **lvcreate** コマンドの **--type** 引数のデフォルト値になります。
raid4	RAID4 専用パリティーディスク
raid5	**raid5_ls** と同じ
raid5_la	* RAID5 left asymmetric * ローテートパリティー 0 + データ継続
raid5_ra	* RAID5 right asymmetric *ローテートパリティー N + データ継続
raid5_ls	* RAID5 left symmetric * ローテートパリティー 0 +データ再起動
raid5_rs	* RAID5 right symmetric * ローテートパリティー N + データ再起動
raid6	**raid6_zr** と同じ
raid6_zr	* RAID6 zero restart * ローテートパリティーゼロ (左から右) + データ再起動

セグメントタイプ	説明
raid6_nr	* RAID6 N restart * ローテートパリティー N (左から右) + データ再起動
raid6_nc	* RAID6 N continue ローテートパリティー N (左から右) + データを継続
raid10	ストライピング + ミラーリング。**-m** を指定し、1 よりも大きい数をストライプの数として指定すると、これは **lvcreate** コマンドの **--type** 引数のデフォルト値になります。 * ミラーセットのストライピング
raid0/raid0_meta	ストライピング。RAID0 では、ストライプ単位で、複数のデータサブボリュームに論理ボリュームデータが分散されます。これは、パフォーマンスを向上させるために使用します。論理ボリュームのデータは、いずれかのデータサブボリュームで障害が発生すると失われます。

ほとんどのユーザーでは、5 つのプライマリータイプ (**raid1**、**raid4**、**raid5**、**raid6**、**raid10**) の 1 つを指定すれば対応できるはずです。RAID 5/6 が使用する各種アルゴリズムの詳細は、http://www.snia.org/sites/default/files/SNIA_DDF_Technical_Position_v2.0.pdf に記載されている『**Common RAID Disk Data Format Specification**』の第 4 章を参照してください。

RAID 論理ボリュームを作成するとき、LVM は、データまたはアレイ内のパリティーサブボリュームごとに、サイズが 1 エクステントのメタデータサブボリュームを作成します。たとえば、2 方向の RAID1 アレイを作成すると、メタデータサブボリュームが 2 つ (**lv_rmeta_0** および **lv_rmeta_1**) と、データサブボリュームが 2 つ (**lv_rimage_0** および **lv_rimage_1**) 作成されます。同様に、3 way ストライプ (および暗黙的なパリティーデバイスが 1 つ) の RAID4 を作成すると、メタデータサブボリュームが 4 つ (**lv_rmeta_0**、**lv_rmeta_1**、**lv_rmeta_2**、および **lv_rmeta_3**)、データサブボリュームが 4 つ (**lv_rimage_0**、**lv_rimage_1**、**lv_rimage_2**、および **lv_rimage_3**) が作成されます。

注記

LVM RAID Calculator を使用すると、RAID ストレージで論理ボリュームを作成するコマンドを生成できます。このアプリケーションは、現在のストレージまたは作成されるストレージについて入力した情報を使用してコマンドを生成します。LVM RAID Calculator アプリケーションは、https://access.redhat.com/labs/lvmraidcalculator/ で使用できます。

7.1. RAID 論理ボリュームの作成

ここでは、異なる種類の RAID 論理ボリュームを作成するコマンドの例を取り上げます。

-m 引数にコピー数を指定して、RAID1 アレイを作成できます。同様に、**-i argument** オプションに、RAID 4/5/6 論理ボリュームのストライプ数を指定します。**-I** 引数で、ストライプのサイズを指定することもできます。

以下のコマンドは、ボリュームグループ **my_vg** 内に、1 ギガバイトの 2 方向 RAID1 アレイ **my_lv** を作成します。

```
# lvcreate --type raid1 -m 1 -L 1G -n my_lv my_vg
```

以下のコマンドは、ボリュームグループ **my_vg** に、サイズが 1 ギガバイトで、名前が **my_lv** の RAID5 アレイ (ストライプ 3 つ + 暗黙的なパリティードライブ 1 つ) を作成します。ストライプ数の指定は、LVM ストライプ化ボリュームの場合と同じように行います。パリティードライブは、正確な数だけ自動的に追加されます。

```
# lvcreate --type raid5 -i 3 -L 1G -n my_lv my_vg
```

以下のコマンドは、ボリュームグループ **my_vg** に、サイズが 1 ギガバイトで、名前が **my_lv** の RAID6 アレイ (ストライプ 3 つ + 暗黙的なパリティードライブ 2 つ) を作成します。

```
# lvcreate --type raid6 -i 3 -L 1G -n my_lv my_vg
```

7.2. RAID0 (ストライピング) 論理ボリュームの作成

RAID0 論理ボリュームは、論理ボリュームデータをストライプサイズ単位で複数のサブボリューム全体に分散します。

RAID0 ボリュームを作成するコマンドの書式は以下のとおりです。

```
lvcreate --type raid0[_meta] --stripes Stripes --stripesize StripeSize VolumeGroup
[PhysicalVolumePath ...]
```

表7.2 RAID0 コマンドの作成に関するパラメーター

パラメーター	説明
--type raid0[_meta]	**raid0** を指定すると、メタデータボリュームなしで RAID0 ボリュームが作成されます。**raid0_meta** を指定すると、メタデータボリュームとともに RAID0 ボリュームが作成されます。RAID0 には耐障害性がないため、RAID1/10 の場合のようにミラーリングされたすべてのデータブロックを格納したり、RAID4/5/6 の場合のようにすべてのパリティーブロックを格納したりする必要はありません。したがって、ミラーリングされたブロックまたはパリティーブロックの再同期の進行状態を把握するメタデータボリュームは必要ありません。ただし、RAID0 から RAID4/5/6/10 に変換するには、メタデータボリュームが必要です。**raid0_meta** を指定すると、割り当ての失敗を防ぐためにこれらのメタデータが事前に割り当てられます。

パラメーター	説明
--stripes Stripes	論理ボリュームを分散するデバイスの数を指定します。
--stripesize StripeSize	各ストライプのサイズをキロバイト単位で指定します。これは、次のデバイスに移動する前にデバイスに書き込まれるデータの量です。
VolumeGroup	使用するボリュームグループを指定します。
PhysicalVolumePath ...	使用するデバイスを指定します。指定しない場合は、LVM により、**Stripes** オプションに指定されているデバイスの数が、各ストライプに 1 つずつ選択されます。

この手順例では、**/dev/sda1**、**/dev/sdb1**、および **/dev/sdc1** のディスクで構成される **mylv** という名前の LVM RAID0 論理ボリュームを作成します。

1. **pvcreate** コマンドを使用し、LVM 物理ボリュームとして使用するディスクにラベルを付けます。

警告

このコマンドは、**/dev/sda1**、**/dev/sdb1**、および **/dev/sdc1** にあるデータを破棄します。

```
# pvcreate /dev/sda1 /dev/sdb1 /dev/sdc1
  Physical volume "/dev/sda1" successfully created
  Physical volume "/dev/sdb1" successfully created
  Physical volume "/dev/sdc1" successfully created
```

2. ボリュームグループ **myvg** を作成します。以下のコマンドを使用すると、ボリュームグループ **myvg** が作成されます。

```
# vgcreate myvg /dev/sda1 /dev/sdb1 /dev/sdc1
  Volume group "myvg" successfully created
```

vgs コマンドを使用すると、作成したボリュームグループの属性を表示できます。

```
# vgs
  VG   #PV #LV #SN Attr   VSize  VFree
  myvg   3   0   0 wz--n- 51.45G 51.45G
```

3. 作成したボリュームグループから、RAID0 論理ボリュームを作成します。以下のコマンドを使用すると、ボリュームグループ **myvg** から、RAID0 ボリューム **mylv** が作成されます。この例

では、ストライプサイズが 4 キロバイトで 3 つのストライプがある、サイズが 2 ギガバイトの論理ボリュームが作成されます。

```
# lvcreate --type raid0 -L 2G --stripes 3 --stripesize 4 -n mylv myvg
  Rounding size 2.00 GiB (512 extents) up to stripe boundary size 2.00 GiB(513 extents).
  Logical volume "mylv" created.
```

4. RAID0 論理ボリュームにファイルシステムを作成します。以下のコマンドを使用すると、論理ボリュームに **ext4** ファイルシステムが作成されます。

```
# mkfs.ext4 /dev/myvg/mylv
mke2fs 1.44.3 (10-July-2018)
Creating filesystem with 525312 4k blocks and 131376 inodes
Filesystem UUID: 9d4c0704-6028-450a-8b0a-8875358c0511
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

以下のコマンドは、論理ボリュームをマウントして、ファイルシステムディスクの領域使用率を報告します。

```
# mount /dev/myvg/mylv /mnt
# df
Filesystem              1K-blocks     Used  Available Use% Mounted on
/dev/mapper/myvg-mylv     2002684     6168    1875072   1% /mnt
```

7.3. RAID ボリュームを初期化する速度の制御

RAID10 論理ボリュームを作成する際に、**sync** 操作で論理ボリュームを初期化するのに必要なバックグラウンド I/O は、その他の I/O 操作 (ボリュームグループメタデータへの更新など) を LVM デバイスに押し出す可能性があります。これはとくに RAID 論理ボリュームを多数作成している場合に生じる可能性があります。これにより、他の LVM 操作の速度が遅くなる場合があります。

RAID 論理ボリュームが初期化される速度は、復旧スロットルを実装することで制御できます。**sync** 操作が実行される速度は、**lvcreate** コマンドの **--minrecoveryrate** および **--maxrecoveryrate** オプションでこれらの操作の最小および最大 I/O 速度を設定すると制御できます。これらのオプションは次のように指定します。

- **--maxrecoveryrate Rate[bBsSkKmMgG]**
  RAID 論理ボリュームの最大復旧速度を設定し、通常の I/O 操作が押し出されないようにします。**Rate** には、アレイ内の各デバイスに対する 1 秒あたりのデータ通信量を指定します。接尾辞を指定しない場合は、kiB/sec/device (デバイスごとに kiB/秒) と見なされます。復旧速度を 0 に設定すると無制限になります。

- **--minrecoveryrate Rate[bBsSkKmMgG]**
  RAID 論理ボリュームの最小復旧速度を設定し、負荷の高い通常の I/O がある場合でも、**sync** 操作の I/O が最小スループットを達成できるようにします。**Rate** には、アレイ内の各デバイスに対する 1 秒あたりのデータ通信量を指定します。接尾辞を指定しない場合は、kiB/sec/device (デバイスごとに kiB/秒) と見なされます。

以下のコマンドは、最大速度が 128 kiB/sec/device で、サイズが 10 ギガバイトのストライプが 3 つある、2 方向の RAID10 アレイを作成します。このアレイは名前は **my_lv** で、ボリュームグループは **my_vg** になります。

```
# lvcreate --type raid10 -i 2 -m 1 -L 10G --maxrecoveryrate 128 -n my_lv my_vg
```

RAID のスクラブ操作の最小および最大復旧速度を指定することもできます。

7.4. リニアデバイスの RAID デバイスへの変換

既存のリニア論理ボリュームを RAID デバイスに変換するには、**lvconvert** コマンドの **--type** 引数を使用します。

以下のコマンドは、ボリュームグループ **my_vg** のリニア論理ボリューム **my_lv** を、2 方向の RAID1 アレイに変換します。

```
# lvconvert --type raid1 -m 1 my_vg/my_lv
```

RAID 論理ボリュームは、メタデータとデータサブボリュームのペアで構成されているため、リニアデバイスを RAID1 アレイに変換すると、メタデータサブボリュームが作成され、リニアボリュームが存在する物理ボリューム (のいずれか) にある、複製元の論理ボリュームに関連付けられます。イメージは、メタデータ/データサブボリュームのペアに追加されます。たとえば、複製元のデバイスは以下のとおりです。

```
# lvs -a -o name,copy_percent,devices my_vg
  LV     Copy%  Devices
  my_lv         /dev/sde1(0)
```

2 方向の RAID1 アレイに変換すると、デバイスには、以下のデータとメタデータサブボリュームのペアが含まれます。

```
# lvconvert --type raid1 -m 1 my_vg/my_lv
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv           6.25   my_lv_rimage_0(0),my_lv_rimage_1(0)
  [my_lv_rimage_0]       /dev/sde1(0)
  [my_lv_rimage_1]       /dev/sdf1(1)
  [my_lv_rmeta_0]        /dev/sde1(256)
  [my_lv_rmeta_1]        /dev/sdf1(0)
```

複製元の論理ボリュームとペアのメタデータイメージを同じ物理ボリュームに配置できないと、**lvconvert** は失敗します。

7.5. LVM RAID1 論理ボリュームの LVM リニア論理ボリュームへの変換

lvconvert コマンドを使用して、既存の RAID1 LVM 論理ボリュームを LVM リニア論理ボリュームに変換するには **-m0** 引数を指定します。これにより、すべての RAID データサブボリュームおよび RAID アレイを構成するすべての RAID メタデータサブボリュームが削除され、最高レベルの RAID1 イメージがリニア論理ボリュームとして残されます。

以下の例は、既存の LVM RAID1 論理ボリュームを表示しています。

```
# lvs -a -o name,copy_percent,devices my_vg
```

```
  LV              Copy%  Devices
  my_lv             100.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
  [my_lv_rimage_0]        /dev/sde1(1)
  [my_lv_rimage_1]        /dev/sdf1(1)
  [my_lv_rmeta_0]         /dev/sde1(0)
  [my_lv_rmeta_1]         /dev/sdf1(0)
```

以下のコマンドは、LVM RAID1 論理ボリューム **my_vg/my_lv** を LVM リニアデバイスに変換します。

```
# lvconvert -m0 my_vg/my_lv
# lvs -a -o name,copy_percent,devices my_vg
  LV      Copy%  Devices
  my_lv          /dev/sde1(1)
```

LVM RAID1 論理ボリューム を LVM リニアボリュームに変換する場合は、削除する物理ボリュームを指定できます。以下の例は、**/dev/sda1** と **/dev/sdb1** の 2 つのイメージで構成される LVM RAID1 論理ボリュームのレイアウトを表示しています。この例で、**lvconvert** コマンドは **/dev/sda1** を削除して、**/dev/sdb1** をリニアデバイスを構成する物理ボリュームとして残すように指定します。

```
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv             100.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
  [my_lv_rimage_0]        /dev/sda1(1)
  [my_lv_rimage_1]        /dev/sdb1(1)
  [my_lv_rmeta_0]         /dev/sda1(0)
  [my_lv_rmeta_1]         /dev/sdb1(0)
# lvconvert -m0 my_vg/my_lv /dev/sda1
# lvs -a -o name,copy_percent,devices my_vg
  LV    Copy%  Devices
  my_lv        /dev/sdb1(1)
```

7.6. ミラー化 LVM デバイスの RAID1 デバイスへの変換

lvconvert コマンドを使用して、セグメントタイプが **mirror** の既存のミラー化 LVM デバイスを RAID1 LVM デバイスに変換するには、**--type raid1** 引数を指定します。これにより、ミラーサブボリューム (**mimage**) の名前が、RAID サブボリューム (**rimage**) に変更します。また、ミラーログは削除され、対応するデータサブボリュームと同じ物理ボリュームのデータサブボリューム用に、メタデータサブボリューム (**rmeta**) が作成されます。

以下の例は、ミラー化論理ボリューム **my_vg/my_lv** のレイアウトを示しています。

```
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv             15.20 my_lv_mimage_0(0),my_lv_mimage_1(0)
  [my_lv_mimage_0]        /dev/sde1(0)
  [my_lv_mimage_1]        /dev/sdf1(0)
  [my_lv_mlog]            /dev/sdd1(0)
```

以下のコマンドは、ミラー化論理ボリューム **my_vg/my_lv** を、RAID1 論理ボリュームに変換します。

```
# lvconvert --type raid1 my_vg/my_lv
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv             100.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
```

```
[my_lv_rimage_0]       /dev/sde1(0)
[my_lv_rimage_1]       /dev/sdf1(0)
[my_lv_rmeta_0]        /dev/sde1(125)
[my_lv_rmeta_1]        /dev/sdf1(125)
```

7.7. RAID 論理ボリュームのサイズ変更

RAID 論理ボリュームのサイズ変更は、以下の方法でできます。

- いずれのタイプの RAID 論理ボリュームのサイズも、**lvresize** コマンドまたは **lvextend** コマンドで増やすことができます。これは、RAID イメージの数を変更するものではありません。ストライプ化 RAID 論理ボリュームでは、ストライプ化 RAID 論理ボリュームの作成時と同じストライプを丸める制約が適用されます。

- いずれのタイプの RAID 論理ボリュームのサイズも、**lvresize** または **lvreduce** コマンドで減らすことができます。これは、RAID イメージの数を変更するものではありません。**lvextend** コマンドでは、ストライプ化 RAID 論理ボリュームの作成時と同じストライプを丸める制約が適用されます。

- **lvconvert** コマンドで **--stripes N** パラメーターを使用すると、ストライプ化 RAID 論理ボリューム (**raid4**/**5**/**6**/**10**) のストライプの数を変更できます。このように、ストライプを追加または削除することで、RAID 論理ボリュームのサイズを増減できます。**raid10** ボリュームにはストライプを追加することしかできないため注意してください。この機能は、同じ RAID レベルを維持しながら、RAID 論理ボリュームの属性を変更できる、RAID の *再成形* 機能の一部になります。RAID 再成形の詳細と、**lvconvert** コマンドを使用して RAID 論理ボリュームを再成形する例は、man ページの **lvmraid**(7) を参照してください。

7.8. 既存の RAID1 デバイスのイメージ数を変更

既存の RAID1 アレイ内のイメージ数は、LVM ミラーリングの初期実装でイメージ数を変更する場合と同様に変更できます。**lvconvert** コマンドを使用して、追加または削除するメタデータ/データサブボリュームのペアの数を指定します。

lvconvert コマンドを使用して RAID1 デバイスにイメージを追加する際、追加後のイメージ数を指定できます。または、デバイスに追加するイメージ数を指定できます。メタデータ/データイメージのペアを置く物理ボリュームを指定することもできます。

メタデータサブボリューム (**rmeta** と呼ばれる) は、対応するデータサブボリューム (**rimage**) と同じ物理デバイスに常に存在します。メタデータ/データのサブボリュームのペアは、**--alloc anywhere** を指定しない限り、RAID アレイにある別のメタデータ/データサブボリュームのペアと同じ物理ボリュームには作成されません。

RAID1 ボリュームにイメージを追加するコマンドの形式は、以下のとおりです。

```
lvconvert -m new_absolute_count vg/lv [removable_PVs]
lvconvert -m +num_additional_images vg/lv [removable_PVs]
```

たとえば、以下のコマンドは、2 方向の RAID1 アレイである LVM デバイス **my_vg**/**my_lv** を表示します。

```
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv             6.25    my_lv_rimage_0(0),my_lv_rimage_1(0)
  [my_lv_rimage_0]        /dev/sde1(0)
```

```
  [my_lv_rimage_1]        /dev/sdf1(1)
  [my_lv_rmeta_0]         /dev/sde1(256)
  [my_lv_rmeta_1]         /dev/sdf1(0)
```

以下のコマンドは、2 方向の RAID1 デバイス **my_vg/my_lv** を、3 方向の RAID1 デバイスに変換します。

```
# lvconvert -m 2 my_vg/my_lv
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv             6.25 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
  [my_lv_rimage_0]       /dev/sde1(0)
  [my_lv_rimage_1]       /dev/sdf1(1)
  [my_lv_rimage_2]       /dev/sdg1(1)
  [my_lv_rmeta_0]        /dev/sde1(256)
  [my_lv_rmeta_1]        /dev/sdf1(0)
  [my_lv_rmeta_2]        /dev/sdg1(0)
```

イメージを RAID1 アレイに追加する場合は、イメージに使用する物理ボリュームを指定できます。以下のコマンドは、2 方向の RAID1 デバイス **my_vg/my_lv** を、3 方向の RAID1 デバイスに変換し、物理ボリューム **/dev/sdd1** がアレイに使用されるようにします。

```
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv             56.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
  [my_lv_rimage_0]       /dev/sda1(1)
  [my_lv_rimage_1]       /dev/sdb1(1)
  [my_lv_rmeta_0]        /dev/sda1(0)
  [my_lv_rmeta_1]        /dev/sdb1(0)
# lvconvert -m 2 my_vg/my_lv /dev/sdd1
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv             28.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
  [my_lv_rimage_0]       /dev/sda1(1)
  [my_lv_rimage_1]       /dev/sdb1(1)
  [my_lv_rimage_2]       /dev/sdd1(1)
  [my_lv_rmeta_0]        /dev/sda1(0)
  [my_lv_rmeta_1]        /dev/sdb1(0)
  [my_lv_rmeta_2]        /dev/sdd1(0)
```

RAID1 アレイからイメージを削除するには、以下のコマンドを使用します。**lvconvert** コマンドを使用して RAID1 デバイスからイメージを削除する場合は、削除後のイメージの合計数を指定できます。または、デバイスから削除するイメージ数を指定することもできます。また、デバイスを削除する物理ボリュームを指定することもできます。

```
lvconvert -m new_absolute_count vg/lv [removable_PVs]
lvconvert -m -num_fewer_images vg/lv [removable_PVs]
```

また、イメージとその関連付けられたメタデータサブボリュームを削除すると、それよりも大きな番号のイメージがそのスロットを引き継ぎます。たとえば、**lv_rimage_0**、**lv_rimage_1**、および **lv_rimage_2** で構成される 3 方向の RAID1 アレイから **lv_rimage_1** を削除すると、RAID1 アレイを構成するイメージの名前は **lv_rimage_0** と **lv_rimage_1** になります 。サブボリューム **lv_rimage_2** の名前が、空のスロットを引き継いで **lv_rimage_1** になります。

以下の例は、3 方向の RAID1 論理ボリューム **my_vg/my_lv** のレイアウトを示しています。

```
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv           100.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
  [my_lv_rimage_0]       /dev/sde1(1)
  [my_lv_rimage_1]       /dev/sdf1(1)
  [my_lv_rimage_2]       /dev/sdg1(1)
  [my_lv_rmeta_0]        /dev/sde1(0)
  [my_lv_rmeta_1]        /dev/sdf1(0)
  [my_lv_rmeta_2]        /dev/sdg1(0)
```

以下のコマンドは、3 方向の RAID1 論理ボリュームを、2 方向の RAID1 論理ボリュームに変換します。

```
# lvconvert -m1 my_vg/my_lv
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv           100.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
  [my_lv_rimage_0]       /dev/sde1(1)
  [my_lv_rimage_1]       /dev/sdf1(1)
  [my_lv_rmeta_0]        /dev/sde1(0)
  [my_lv_rmeta_1]        /dev/sdf1(0)
```

以下のコマンドは、3 方向の RAID1 論理ボリュームを、2 方向の RAID1 論理ボリュームに変換し、物理ボリューム **/dev/sde1** からイメージを削除することを指定します。

```
# lvconvert -m1 my_vg/my_lv /dev/sde1
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv           100.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
  [my_lv_rimage_0]       /dev/sdf1(1)
  [my_lv_rimage_1]       /dev/sdg1(1)
  [my_lv_rmeta_0]        /dev/sdf1(0)
  [my_lv_rmeta_1]        /dev/sdg1(0)
```

7.9. RAID イメージを複数の論理ボリュームに分割

RAID 論理ボリュームのイメージを分割して新しい論理ボリュームを形成できます。

RAID イメージを分割するコマンドの形式は、以下のとおりです。

```
lvconvert --splitmirrors count -n splitname vg/lv [removable_PVs]
```

既存の RAID1 論理ボリュームから RAID イメージを削除する場合と同様に、RAID データのサブボリューム (およびその関連付けられたメタデータのサブボリューム) をデバイスから削除する場合、それより大きい番号のイメージは、そのスロットを埋めるために番号が変更になります。そのため、RAID アレイを構成する論理ボリュームのインデックス番号は連続する整数となります。

注記

RAID1 アレイがまだ同期していない場合は、RAID イメージを分割できません。

以下の例は、2 方向の RAID1 論理ボリューム **my_lv** を、**my_lv** と **new** の 2 つのリニア論理ボリュームに分割します。

```
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv           12.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
  [my_lv_rimage_0]       /dev/sde1(1)
  [my_lv_rimage_1]       /dev/sdf1(1)
  [my_lv_rmeta_0]        /dev/sde1(0)
  [my_lv_rmeta_1]        /dev/sdf1(0)
# lvconvert --splitmirror 1 -n new my_vg/my_lv
# lvs -a -o name,copy_percent,devices my_vg
  LV     Copy%  Devices
  my_lv         /dev/sde1(1)
  new           /dev/sdf1(1)
```

以下の例は、3 方向の RAID1 論理ボリューム **my_lv** を、2 方向の RAID1 論理ボリューム **my_lv** と、リニア論理ボリューム **new** に分割します。

```
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv           100.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
  [my_lv_rimage_0]       /dev/sde1(1)
  [my_lv_rimage_1]       /dev/sdf1(1)
  [my_lv_rimage_2]       /dev/sdg1(1)
  [my_lv_rmeta_0]        /dev/sde1(0)
  [my_lv_rmeta_1]        /dev/sdf1(0)
  [my_lv_rmeta_2]        /dev/sdg1(0)
# lvconvert --splitmirror 1 -n new my_vg/my_lv
# lvs -a -o name,copy_percent,devices my_vg
  LV           Copy%  Devices
  my_lv           100.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
  [my_lv_rimage_0]       /dev/sde1(1)
  [my_lv_rimage_1]       /dev/sdf1(1)
  [my_lv_rmeta_0]        /dev/sde1(0)
  [my_lv_rmeta_1]        /dev/sdf1(0)
  new                    /dev/sdg1(1)
```

7.10. RAID イメージの分割とマージ

lvconvert コマンドで、**--splitmirrors** 引数とともに **--trackchanges** 引数を使用すると、すべての変更を追跡しながら、RAID1 アレイのイメージを一時的に読み取り専用に分割できます。これにより、イメージの分割後に変更になったアレイの部分のみを再同期する一方で、後でそのイメージをアレイにマージし直すことができます。

RAID イメージを分割する **lvconvert** コマンドの形式は、以下のとおりです。

```
lvconvert --splitmirrors count --trackchanges vg/lv [removable_PVs]
```

--trackchanges 引数を使用して RAID イメージを分割する場合、分割するイメージを指定することはできますが、分割されるボリューム名を変更することはできません。また、分割して作成されたボリュームには以下の制限があります。

- 作成された新規ボリュームは読み取り専用です。
- 新規ボリュームのサイズは変更できません。

- 残りのアレイの名前は変更できません。
- 残りのアレイのサイズは変更できません。
- 新規のボリュームと、残りのアレイを個別にアクティブにすることはできません。

指定した **--trackchanges** 引数を使用して分割したイメージをマージするには、**lvconvert** コマンドに **--merge** 引数を指定して実行します。イメージをマージすると、イメージが分割されてから変更したアレイの部分のみが再同期されます。

RAID イメージをマージする **lvconvert** コマンドの形式は、以下のとおりです。

```
lvconvert --merge raid_image
```

以下の例は、残りのアレイへの変更を追跡する一方で、RAID1 論理ボリュームを作成し、そのボリュームからイメージを分割します。

```
# lvcreate --type raid1 -m 2 -L 1G -n my_lv .vg
  Logical volume "my_lv" created
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv           100.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
  [my_lv_rimage_0]       /dev/sdb1(1)
  [my_lv_rimage_1]       /dev/sdc1(1)
  [my_lv_rimage_2]       /dev/sdd1(1)
  [my_lv_rmeta_0]        /dev/sdb1(0)
  [my_lv_rmeta_1]        /dev/sdc1(0)
  [my_lv_rmeta_2]        /dev/sdd1(0)
# lvconvert --splitmirrors 1 --trackchanges my_vg/my_lv
  my_lv_rimage_2 split from my_lv for read-only purposes.
  Use 'lvconvert --merge my_vg/my_lv_rimage_2' to merge back into my_lv
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv           100.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
  [my_lv_rimage_0]       /dev/sdb1(1)
  [my_lv_rimage_1]       /dev/sdc1(1)
  my_lv_rimage_2         /dev/sdd1(1)
  [my_lv_rmeta_0]        /dev/sdb1(0)
  [my_lv_rmeta_1]        /dev/sdc1(0)
  [my_lv_rmeta_2]        /dev/sdd1(0)
```

以下の例は、残りのアレイへの変更を追跡する一方で、RAID1 ボリュームからイメージを分割します。その後、ボリュームをアレイにマージし直しています。

```
# lvconvert --splitmirrors 1 --trackchanges my_vg/my_lv
  lv_rimage_1 split from my_lv for read-only purposes.
  Use 'lvconvert --merge my_vg/my_lv_rimage_1' to merge back into my_lv
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv           100.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
  [my_lv_rimage_0]       /dev/sdc1(1)
  my_lv_rimage_1         /dev/sdd1(1)
  [my_lv_rmeta_0]        /dev/sdc1(0)
  [my_lv_rmeta_1]        /dev/sdd1(0)
# lvconvert --merge my_vg/my_lv_rimage_1
```

```
  my_vg/my_lv_rimage_1 successfully merged back into my_vg/my_lv
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv           100.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
  [my_lv_rimage_0]       /dev/sdc1(1)
  [my_lv_rimage_1]       /dev/sdd1(1)
  [my_lv_rmeta_0]        /dev/sdc1(0)
  [my_lv_rmeta_1]        /dev/sdd1(0)
```

RAID1 ボリュームからイメージを分割後、その分割を永続的にするには **lvconvert --splitmirrors** コマンドを実行します。ここでは **--trackchanges** 引数は指定せず、イメージを分割する最初の **lvconvert** コマンドを繰り返します。これにより、**--trackchanges** 引数が作成したリンクが機能しなくなります。

追跡されるイメージを永久に分割する目的である場合を除き、**--trackchanges** 引数を使用してイメージを分割してから、アレイで **lvconvert --splitmirrors** コマンドを実行することはできません。

以下の一連のコマンドは、イメージを分割してこれを追跡してから、追跡されるイメージを永久に分割します。

```
# lvconvert --splitmirrors 1 --trackchanges my_vg/my_lv
  my_lv_rimage_1 split from my_lv for read-only purposes.
  Use 'lvconvert --merge my_vg/my_lv_rimage_1' to merge back into my_lv
# lvconvert --splitmirrors 1 -n new my_vg/my_lv
# lvs -a -o name,copy_percent,devices my_vg
  LV   Copy%  Devices
  my_lv       /dev/sdc1(1)
  new         /dev/sdd1(1)
```

ただし、以下の一連のコマンドは失敗する点に注意してください。

```
# lvconvert --splitmirrors 1 --trackchanges my_vg/my_lv
  my_lv_rimage_1 split from my_lv for read-only purposes.
  Use 'lvconvert --merge my_vg/my_lv_rimage_1' to merge back into my_lv
# lvconvert --splitmirrors 1 --trackchanges my_vg/my_lv
  Cannot track more than one split image at a time
```

同様に、以下の一連のコマンドも失敗します。分割されたイメージが追跡されていないためです。

```
# lvconvert --splitmirrors 1 --trackchanges my_vg/my_lv
  my_lv_rimage_1 split from my_lv for read-only purposes.
  Use 'lvconvert --merge my_vg/my_lv_rimage_1' to merge back into my_lv
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv           100.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
  [my_lv_rimage_0]       /dev/sdc1(1)
  my_lv_rimage_1         /dev/sdd1(1)
  [my_lv_rmeta_0]        /dev/sdc1(0)
  [my_lv_rmeta_1]        /dev/sdd1(0)
# lvconvert --splitmirrors 1 -n new my_vg/my_lv /dev/sdc1
  Unable to split additional image from my_lv while tracking changes for my_lv_rimage_1
```

7.11. RAID 障害ポリシーの設定

LVM RAID は、**lvm.conf** ファイルの **raid_fault_policy** フィールドで定義されている詳細設定に基づいて、デバイス障害を自動で処理します。

- **raid_fault_policy** フィールドが **allocate** に設定されている場合、システムは障害が発生したデバイスをボリュームグループの予備のデバイスに置き換えようとします。予備のデバイスがないと、システムログにレポートが送信されます。
- **raid_fault_policy** フィールドが **warn** に設定されている場合、システムは警告を生成して、ログにはデバイスが失敗したことが示されます。これにより、ユーザーは取るべき一連の動作を決めることができます。

該当するポリシーを使用するデバイスが残っている限り、RAID 論理ボリュームは操作を続行します。

7.11.1. 「allocate」RAID 障害ポリシー

以下の例では、**raid_fault_policy** フィールドは **lvm.conf** ファイルで **allocate** に設定されています。RAID 論理ボリュームは、以下のように配置されます。

```
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv          100.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
  [my_lv_rimage_0]       /dev/sde1(1)
  [my_lv_rimage_1]       /dev/sdf1(1)
  [my_lv_rimage_2]       /dev/sdg1(1)
  [my_lv_rmeta_0]        /dev/sde1(0)
  [my_lv_rmeta_1]        /dev/sdf1(0)
  [my_lv_rmeta_2]        /dev/sdg1(0)
```

/dev/sde デバイスが失敗すると、システムログはエラーメッセージを表示します。

```
# grep lvm /var/log/messages
Jan 17 15:57:18 bp-01 lvm[8599]: Device #0 of raid1 array, my_vg-my_lv, has failed.
Jan 17 15:57:18 bp-01 lvm[8599]: /dev/sde1: read failed after 0 of 2048 at
250994294784: Input/output error
Jan 17 15:57:18 bp-01 lvm[8599]: /dev/sde1: read failed after 0 of 2048 at
250994376704: Input/output error
Jan 17 15:57:18 bp-01 lvm[8599]: /dev/sde1: read failed after 0 of 2048 at 0:
Input/output error
Jan 17 15:57:18 bp-01 lvm[8599]: /dev/sde1: read failed after 0 of 2048 at
4096: Input/output error
Jan 17 15:57:19 bp-01 lvm[8599]: Couldn't find device with uuid
3lugiV-3eSP-AFAR-sdrP-H20O-wM2M-qdMANy.
Jan 17 15:57:27 bp-01 lvm[8599]: raid1 array, my_vg-my_lv, is not in-sync.
Jan 17 15:57:36 bp-01 lvm[8599]: raid1 array, my_vg-my_lv, is now in-sync.
```

raid_fault_policy フィールドが **allocate** に設定されているため、障害が発生したデバイスは、ボリュームグループの新しいデバイスに置き換わります。

```
# lvs -a -o name,copy_percent,devices vg
  Couldn't find device with uuid 3lugiV-3eSP-AFAR-sdrP-H20O-wM2M-qdMANy.
  LV            Copy%  Devices
  lv           100.00 lv_rimage_0(0),lv_rimage_1(0),lv_rimage_2(0)
  [lv_rimage_0]       /dev/sdh1(1)
  [lv_rimage_1]       /dev/sdf1(1)
  [lv_rimage_2]       /dev/sdg1(1)
```

```
[lv_rmeta_0]       /dev/sdh1(0)
[lv_rmeta_1]       /dev/sdf1(0)
[lv_rmeta_2]       /dev/sdg1(0)
```

障害が発生したデバイスを交換しても、LVM は、障害が発生したデバイスが見つけられないと示すことに注意してください。これは、障害が発生したデバイスが、RAID 論理ボリュームからは削除されても、ボリュームグループからは削除されていないためです。**vgreduce --removemissing VG** を実行すると、障害が発生したデバイスをボリュームグループから削除できます。

raid_fault_policy を **allocate** に設定したにもかかわらず、予備のデバイスがない場合は、割り当てが失敗し、論理ボリュームがそのままの状態になります。割り当てに失敗すると、ドライブが修復され、**lvchange** コマンドの **--refresh** オプションで、障害が発生したデバイスの復元を開始できます。もしくは、障害が発生したデバイスを交換することもできます。

7.11.2. 「warn」RAID 障害ポリシー

以下の例では、**raid_fault_policy** フィールドは、**lvm.conf** ファイルで **warn** に設定されています。RAID 論理ボリュームは、以下のように配置されます。

```
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv          100.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
  [my_lv_rimage_0]       /dev/sdh1(1)
  [my_lv_rimage_1]       /dev/sdf1(1)
  [my_lv_rimage_2]       /dev/sdg1(1)
  [my_lv_rmeta_0]        /dev/sdh1(0)
  [my_lv_rmeta_1]        /dev/sdf1(0)
  [my_lv_rmeta_2]        /dev/sdg1(0)
```

/dev/sdh デバイスに障害が発生すると、システムログはエラーメッセージを表示します。ただし、この場合、LVM はイメージの 1 つを置き換えて RAID デバイスを自動的に修復しようとはしません。したがって、デバイスに障害が発生したら、**lvconvert** コマンドの **--repair** 引数を使用してデバイスを置き換えることができます。

7.12. 論理ボリュームで RAID デバイスの交換

論理ボリュームの RAID デバイスは、**lvconvert** コマンドで置き換えることができます。

- RAID デバイスに障害が何も発生しなかった場合は、**lvconvert** コマンドの **--replace** 引数を使用して、デバイスを置き換えます。
- RAID デバイスに障害が発生した場合は、**lvconvert** コマンドの **--repair** 引数を使用して、障害が発生したデバイスを交換します。

7.12.1. 障害のない RAID デバイスの交換

論理ボリュームの RAID デバイスを交換するには、**lvconvert** コマンドの **--replace** 引数を使用します。このコマンドは、RAID デバイスが失敗しないと有効ではありません。

lvconvert --replace コマンドの形式は、以下のとおりです。

```
lvconvert --replace dev_to_remove vg/lv [possible_replacements]
```

以下の例は、RAID1 論理ボリュームを作成した後に、そのボリューム内のデバイスを交換しています。

```
# lvcreate --type raid1 -m 2 -L 1G -n my_lv my_vg
  Logical volume "my_lv" created
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv           100.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
  [my_lv_rimage_0]       /dev/sdb1(1)
  [my_lv_rimage_1]       /dev/sdb2(1)
  [my_lv_rimage_2]       /dev/sdc1(1)
  [my_lv_rmeta_0]        /dev/sdb1(0)
  [my_lv_rmeta_1]        /dev/sdb2(0)
  [my_lv_rmeta_2]        /dev/sdc1(0)
# lvconvert --replace /dev/sdb2 my_vg/my_lv
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv           37.50 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
  [my_lv_rimage_0]       /dev/sdb1(1)
  [my_lv_rimage_1]       /dev/sdc2(1)
  [my_lv_rimage_2]       /dev/sdc1(1)
  [my_lv_rmeta_0]        /dev/sdb1(0)
  [my_lv_rmeta_1]        /dev/sdc2(0)
  [my_lv_rmeta_2]        /dev/sdc1(0)
```

以下の例は、RAID1 論理ボリュームを作成した後に、そのボリューム内のデバイスを交換し、交換したデバイスに使用する物理ボリュームを指定しています。

```
# lvcreate --type raid1 -m 1 -L 100 -n my_lv my_vg
  Logical volume "my_lv" created
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv           100.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
  [my_lv_rimage_0]       /dev/sda1(1)
  [my_lv_rimage_1]       /dev/sdb1(1)
  [my_lv_rmeta_0]        /dev/sda1(0)
  [my_lv_rmeta_1]        /dev/sdb1(0)
# pvs
  PV         VG       Fmt  Attr PSize    PFree
  /dev/sda1  my_vg    lvm2 a--  1020.00m  916.00m
  /dev/sdb1  my_vg    lvm2 a--  1020.00m  916.00m
  /dev/sdc1  my_vg    lvm2 a--  1020.00m 1020.00m
  /dev/sdd1  my_vg    lvm2 a--  1020.00m 1020.00m
# lvconvert --replace /dev/sdb1 my_vg/my_lv /dev/sdd1
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv           28.00 my_lv_rimage_0(0),my_lv_rimage_1(0)
  [my_lv_rimage_0]       /dev/sda1(1)
  [my_lv_rimage_1]       /dev/sdd1(1)
  [my_lv_rmeta_0]        /dev/sda1(0)
  [my_lv_rmeta_1]        /dev/sdd1(0)
```

1 度に 2 つ以上の RAID デバイスを交換するには、以下の例のように複数の **replace** 引数を指定します。

```
# lvcreate --type raid1 -m 2 -L 100 -n my_lv my_vg
  Logical volume "my_lv" created
# lvs -a -o name,copy_percent,devices my_vg
```

```
  LV              Copy%  Devices
  my_lv           100.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
  [my_lv_rimage_0]       /dev/sda1(1)
  [my_lv_rimage_1]       /dev/sdb1(1)
  [my_lv_rimage_2]       /dev/sdc1(1)
  [my_lv_rmeta_0]        /dev/sda1(0)
  [my_lv_rmeta_1]        /dev/sdb1(0)
  [my_lv_rmeta_2]        /dev/sdc1(0)
# lvconvert --replace /dev/sdb1 --replace /dev/sdc1 my_vg/my_lv
# lvs -a -o name,copy_percent,devices my_vg
  LV              Copy%  Devices
  my_lv           60.00 my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
  [my_lv_rimage_0]       /dev/sda1(1)
  [my_lv_rimage_1]       /dev/sdd1(1)
  [my_lv_rimage_2]       /dev/sde1(1)
  [my_lv_rmeta_0]        /dev/sda1(0)
  [my_lv_rmeta_1]        /dev/sdd1(0)
  [my_lv_rmeta_2]        /dev/sde1(0)
```

7.12.2. 論理ボリュームに障害が発生した RAID デバイスの交換

RAID は従来の LVM ミラーリングとは異なります。LVM ミラーリングでは、障害が発生したデバイスを削除する必要がありました。削除しないと、ミラー化論理ボリュームがハングしたためです。RAID アレイは、障害があるデバイスがあっても稼働し続けることができます。RAID1 以外の RAID タイプでデバイスを削除すると、レベルが低い RAID に変換されます (たとえば、RAID6 から RAID5、もしくは RAID4 または RAID5 から RAID0)。そのため、障害のあるデバイスを無条件に削除してから交換するのではなく、**lvconvert** コマンドで **--repair** 引数を使用して、RAID ボリュームのデバイスを 1 回で置き換えることができます。

次の例では、RAID 論理ボリュームが以下のように配置されます。

```
# lvs -a -o name,copy_percent,devices my_vg
  LV              Cpy%Sync Devices
  my_lv           100.00   my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
  [my_lv_rimage_0]         /dev/sde1(1)
  [my_lv_rimage_1]         /dev/sdc1(1)
  [my_lv_rimage_2]         /dev/sdd1(1)
  [my_lv_rmeta_0]          /dev/sde1(0)
  [my_lv_rmeta_1]          /dev/sdc1(0)
  [my_lv_rmeta_2]          /dev/sdd1(0)
```

/dev/sdc デバイスに障害が発生した場合、**lvs** コマンドの出力は以下のようになります。

```
# lvs -a -o name,copy_percent,devices my_vg
  /dev/sdc: open failed: No such device or address
  Couldn't find device with uuid A4kRl2-vIzA-uyCb-cci7-bOod-H5tX-IzH4Ee.
  WARNING: Couldn't find all devices for LV my_vg/my_lv_rimage_1 while checking used and
assumed devices.
  WARNING: Couldn't find all devices for LV my_vg/my_lv_rmeta_1 while checking used and
assumed devices.
  LV              Cpy%Sync Devices
  my_lv           100.00   my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
  [my_lv_rimage_0]         /dev/sde1(1)
  [my_lv_rimage_1]         [unknown](1)
  [my_lv_rimage_2]         /dev/sdd1(1)
```

```
  [my_lv_rmeta_0]         /dev/sde1(0)
  [my_lv_rmeta_1]         [unknown](0)
  [my_lv_rmeta_2]         /dev/sdd1(0)
```

次のコマンドを実行して、障害が発生したコマンドを交換して、論理ボリュームを表示します。

```
# lvconvert --repair my_vg/my_lv
  /dev/sdc: open failed: No such device or address
  Couldn't find device with uuid A4kRl2-vIzA-uyCb-cci7-bOod-H5tX-IzH4Ee.
  WARNING: Couldn't find all devices for LV my_vg/my_lv_rimage_1 while checking used and
assumed devices.
  WARNING: Couldn't find all devices for LV my_vg/my_lv_rmeta_1 while checking used and
assumed devices.
Attempt to replace failed RAID images (requires full device resync)? [y/n]: y
  Faulty devices in my_vg/my_lv successfully replaced.
# lvs -a -o name,copy_percent,devices my_vg
  /dev/sdc: open failed: No such device or address
  /dev/sdc1: open failed: No such device or address
  Couldn't find device with uuid A4kRl2-vIzA-uyCb-cci7-bOod-H5tX-IzH4Ee.
  LV              Cpy%Sync Devices
  my_lv           43.79    my_lv_rimage_0(0),my_lv_rimage_1(0),my_lv_rimage_2(0)
  [my_lv_rimage_0]         /dev/sde1(1)
  [my_lv_rimage_1]         /dev/sdb1(1)
  [my_lv_rimage_2]         /dev/sdd1(1)
  [my_lv_rmeta_0]          /dev/sde1(0)
  [my_lv_rmeta_1]          /dev/sdb1(0)
  [my_lv_rmeta_2]          /dev/sdd1(0)
```

障害が発生したデバイスを交換しても、LVM は、障害が発生したデバイスが見つけられないと示すことに注意してください。これは、障害が発生したデバイスが、RAID 論理ボリュームからは削除されても、ボリュームグループからは削除されていないためです。**vgreduce --removemissing VG** を実行すると、障害が発生したデバイスをボリュームグループから削除できます。

デバイス障害が一時的であったり、障害が発生したデバイスの修復が可能な場合は、**lvchange** コマンドの **--refresh** オプションを使用して、障害が発生したデバイスの復旧を開始できます。

以下のコマンドは論理ボリュームを更新します。

```
# lvchange --refresh my_vg/my_lv
```

7.13. RAID 論理ボリュームでのデータ整合性の確認 (RAID スクラビング)

LVM は RAID 論理ボリュームのスクラビングサポートを提供します。RAID スクラビングは、すべてのデータおよびアレイ内のパリティーブロックを読み込み、それらが一貫しているかどうかを確認するプロセスです。

lvchange コマンドの **--syncaction** オプションを使用して、RAID スクラビングの操作を開始します。**check** または **repair** のいずれかの操作を指定します。**check** 操作はアレイ全体を対象に、アレイ内の不一致の数を記録しますが、不一致を修復することはありません。**repair** 操作は不一致を発見した時点で修正を行います。

RAID 論理ボリュームのスクラビングを実行するコマンドの形式は以下のとおりです。

```
lvchange --syncaction {check|repair} vg/raid_lv
```

注記

lvchange --syncaction repair vg/raid_lv 操作は、**lvconvert --repair vg/raid_lv** と同じ機能を実行しません。**lvchange --syncaction repair** は、アレイでバックグラウンドの同期操作を開始しますが、**lvconvert --repair** は、ミラーまたは RAID 論理ボリュームで障害が発生したデバイスを修復または交換します。

RAID スクラビング操作をサポートするため、**lvs**コマンドは、**raid_sync_action** と **raid_mismatch_count** の 2 つの新しい出力可能なフィールドに対応します。これらのフィールドは、デフォルトでは出力されません。これらのフィールドを表示するには、以下のように、**lvs** の **-o** パラメーターを使用して指定します。

```
lvs -o +raid_sync_action,raid_mismatch_count vg/lv
```

raid_sync_action フィールドは、raid ボリュームが現在実行している同期操作を表示します。これには、以下のいずれかの値を使用できます。

- **idle** - すべての同期操作が完了しています (何も実行していません)
- **resync** - アレイを初期化、またはマシン障害後の復旧を実行します
- **recover** - アレイ内のデバイスを置き換えます
- **check** - アレイの不一致を検索します
- **repair** - 不一致を検索し、修復します

raid_mismatch_count フィールドは、**check** 操作時に検出された不一致の数を表示します。

lvs コマンドの **Cpy%Sync** フィールドは、**check** および **repair** を含む **raid_sync_action** 操作のいずれかの進捗を出力するようになりました。

lvs コマンドの **lv_attr** フィールドは、RAID スクラビング操作をサポートする追加のインジケーターを提供するようになりました。このフィールドのビット 9 は、論理ボリュームの正常性を示し、以下のインジケーターがサポートされるようになりました。

- 「(**m**) ismatches (不一致)」は、RAID 論理ボリュームに不一致があることを示します。この文字は、スクラビング操作で RAID に一貫性がない部分があることを検出した後に表示されます。
- 「(**r**)efresh (更新)」は、LVM がデバイスラベルを読み取り、デバイスを稼働できると認識した場合でも、RAID アレイのデバイスに障害が発生し、カーネルがこれを障害と認識していることを示します。この論理ボリュームを、デバイスが利用可能になったことをカーネルに通知するために更新する (refresh) か、もしくはデバイスに障害が発生したと思われる場合はデバイスを置き換える (replace) 必要があります。

RAID スクラビング操作を実行する際に、**sync** 操作で必要になるバックグラウンド I/O は、その他の I/O 操作 (ボリュームグループメタデータへの更新など) を LVM デバイスに押し出す可能性があります。これにより、他の LVM 操作が遅くなる可能性があります。復旧スロットルを実装して RAID 論理ボリュームのスクラビングを実行する速度を制御できます。

sync 操作の実行される速度は、**lvchange** コマンドの **--minrecoveryrate** および **--maxrecoveryrate** オプションを使用して、それらの操作の最小および最大 I/O 速度を設定することにより制御できます。これらのオプションは以下のように指定します。

- **--maxrecoveryrate Rate[bBsSkKmMgG]**

RAID 論理ボリュームの最大復旧速度を設定し、通常の I/O 操作が押し出されないようにします。**Rate** には、アレイ内の各デバイスに対する 1 秒あたりのデータ通信量を指定します。接尾辞を指定しない場合は、kiB/sec/device (デバイスごとに kiB/秒) と見なされます。復旧速度を 0 に設定すると無制限になります。

- **--minrecoveryrate Rate[bBsSkKmMgG]**
  RAID 論理ボリュームの最小復旧速度を設定し、負荷の高い通常の I/O がある場合でも、**sync** 操作の I/O が最小スループットを達成できるようにします。**Rate** には、アレイ内の各デバイスに対する 1 秒あたりのデータ通信量を指定します。接尾辞を指定しない場合は、kiB/sec/device (デバイスごとに kiB/秒) と見なされます。

7.14. RAID レベルの変更 (RAID テイクオーバー)

LVM は、Raid **テイクオーバー** をサポートします。これは、RAID 論理ボリュームの RAID レベルを別のレベル (たとえば RAID 5 から RAID 6) へ変えることを意味します。RAID レベルの変更は、通常、デバイスの耐障害性を増減したり、論理ボリュームのストライプ化をやり直すために行われます。RAID テイクオーバーには **lvconvert** を使用します。RAID テイクオーバーの詳細と、**lvconvert** を使用して RAID 論理ボリュームを変換する例は、man ページの **lvmraid**(7) を参照してください。

7.15. RAID ボリュームの属性の変更 (RAID 再成形)

RAID **再成形** とは、同じ RAID レベルを維持しつつ、RAID 論理ボリュームの属性を変更することを言います。変更できる属性には、RAID レイアウト、ストライプのサイズ、ストライプの数などがあります。RAID 再成形と、**lvconvert** コマンドを使用して RAID 論理ボリュームを再生成する例は、man ページの **lvmraid**(7) を参照してください。

7.16. RAID1 論理ボリュームでの I/O 操作の制御

lvchange コマンドの **--writemostly** および **--writebehind** パラメーターを使用して、RAID1 論理ボリュームのデバイスに対する I/O 操作を制御できます。これらのパラメーターを使用する形式は以下のとおりです。

- **--[raid]writemostly PhysicalVolume[:{t|y|n}]**
  RAID1 論理ボリュームのデバイスを **write-mostly** とマークします。これらのドライブのすべての読み取りは、必要でない限り回避されます。このパラメーターを設定することにより、ドライブに対する I/O 操作の回数を最小限に抑えることができます。デフォルトでは、論理ボリュームに指定された物理ボリュームに対して、**write-mostly** 属性を yes に設定します。**:n** を物理ボリュームに追加して **write-mostly** フラグを削除したり、**:t** を指定して値を切り替えたりできます。**--writemostly** 引数は、1 つのコマンドで複数回指定できるため、1 回で論理ボリュームのすべての物理ボリュームで、write-mostly 属性を切り替えることができます。

- **--[raid]writebehind IOCount**
  write-mostly というマークが付いている RAID1 論理ボリュームのデバイスに許可される、未処理の書き込みの最大数を指定します。この値を上回ると書き込みは同期され、構成要素になっているデバイスへの書き込みがすべて、アレイが書き込みの完了を知らせる前に完了してしまいます。この値をゼロに設定すると、設定はクリアになり、システムが値を任意に選択できるようになります。

7.17. RAID 論理ボリュームのリージョンサイズの変更

RAID 論理ボリュームを作成すると、論理ボリュームのリージョンサイズは、**/etc/lvm/lvm.conf** ファイルの **raid_region_size** パラメーターの値になります。このデフォルト値は、**lvcreate** コマンドの **-R** オプションで上書きできます。

RAID 論理ボリュームを作成したら、**lvconvert** コマンドの **-R** オプションで、ボリュームのリージョンサイズを変更できます。以下の例では、論理ボリューム **vg/raidlv** のリージョンサイズを 4096K に変更します。リージョンサイズを変更するには、RAID ボリュームを同期する必要があります。

```
# lvconvert -R 4096K vg/raid1
Do you really want to change the region_size 512.00 KiB of LV vg/raid1 to 4.00 MiB? [y/n]: y
  Changed region size on RAID LV vg/raid1 to 4.00 MiB.
```

第8章 スナップショット論理ボリューム

LVM スナップショット機能は、サービスを中断せずに、特定の時点でデバイスの仮想イメージを作成する機能を提供します。

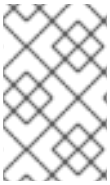
8.1. スナップショットボリューム

LVM スナップショット機能により、サービスを中断せずに任意の時点でデバイスの仮想イメージを作成できます。スナップショットの取得後に複製元のデバイスに変更が加えられると、データが変更される前に、これから変更される部分のコピーがスナップショット機能により作成されるため、このコピーを使用して、デバイスの状態を再構築できます。

注記

LVM は、シンプロビジョニングされたスナップショットをサポートします。

スナップショットは、スナップショットの作成後に変更されたデータ部分のみをコピーするため、スナップショット機能に必要なストレージは最小限になります。たとえば、コピー元がほとんど更新されない場合は、複製元の 3 ~ 5 % の容量があれば十分にスナップショットを維持できます。



注記

ファイルシステムのスナップショットコピーは仮想コピーであり、ファイルシステムのメディアバックアップを実際に作成するわけではありません。スナップショットは、バックアップの代替手順にはなりません。

複製元のボリュームへの変更を保管するために確保するスペース量は、スナップショットのサイズによって異なります。たとえば、スナップショットを作成してから複製元を完全に上書きした場合に、その変更を保管するのに必要なスナップショットのサイズは、複製元のボリュームと同じか、それ以上になります。スナップショットのサイズは、予想される変更レベルに応じて決定する必要があります。たとえば、**/usr** など、その大部分が読み取り用に使用されるボリュームの短期的なスナップショットに必要なスペースは、**/home** のように大量の書き込みが行われるボリュームの長期的なスナップショットに必要なスペースよりも小さくなります。

スナップショットが満杯になると、そのスナップショットは無効になります。複製元のボリュームの変更をトラッキングできなくなるためです。スナップショットのサイズは常時監視する必要があります。ただし、スナップショットのサイズは完全に変更することが可能なため、ストレージに余裕があれば、スナップショットが停止しないように、ボリュームサイズを拡大できます。逆に、スナップショットボリュームサイズが必要以上に大きければ、そのボリュームのサイズを縮小して、他の論理ボリュームで必要となる領域を確保できます。

スナップショットのファイルシステムを作成しても、複製元への完全な読み取り/書き込みのアクセスは引き続き可能です。スナップショット上のチャンクを変更した場合は、そのチャンクにマークが付けられ、そこには、複製元のボリュームのコピーは入りません。

スナップショット機能にはいくつかの用途があります。

- 最も一般的な用途は、継続的にデータを更新している稼動中のシステムを停止せずに、論理ボリューム上でバックアップを実行する必要がある場合にスナップショットを撮ることです。
- スナップショットファイルシステムで **fsck** コマンドを実行してファイルシステムの整合性をチェックし、複製元のファイルシステムを修復する必要があるかどうかを判断できます。

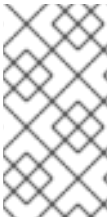
- スナップショットは読み取りおよび書き込み用なので、スナップショットを撮ってそのスナップショットにテストを実行することにより、実際のデータに触れることなく、実稼働データにアプリケーションのテストを実行できます。

- LVM ボリュームを作成して、Red Hat の仮想化と併用することが可能です。LVM スナップショットを使用して、仮想ゲストイメージのスナップショットを作成できます。このスナップショットは、最小限のストレージを使用して、既存のゲストの変更や新規ゲストの作成を行う上で利便性の高い方法を提供します。

lvconvert コマンドの **--merge** オプションを使用して、スナップショットを複製元のボリュームにマージすることが可能です。この機能の用途の 1 つがシステムロールバックの実行で、データやファイルを紛失した場合や、システムを以前の状態に復元する必要がある場合に行います。スナップショットボリュームのマージ後の論理ボリュームには、複製元のボリューム名、マイナー番号、UUID が付けられ、マージされたスナップショットは削除されます。

8.2. スナップショットボリュームの作成

スナップショットボリュームを作成するには、**lvcreate** コマンドで **-s** 引数を使用します。スナップショットボリュームは書き込み可能です。



注記

LVM スナップショットは、クラスターのノード間ではサポートされていません。共有ボリュームグループ内にスナップショットボリュームは作成できません。ただし、共有論理ボリューム上でデータの一貫したバックアップ作成が必要な場合は、ボリュームを排他的にアクティブにした上で、スナップショットを作成できます。

注記

RAID 論理ボリュームを対象としたスナップショットに対応しています。

LVM では、複製元のボリュームのサイズよりも大きく、そのボリュームのメタデータを必要とするスナップショットを作成できません。これよりも大きなスナップショットボリュームを指定しても、システムには、複製元のサイズに必要な大きさのスナップショットボリュームのみが作成されます。

デフォルトでは、通常の起動コマンド時に、スナップショットボリュームがスキップされます。

次の手順は、**origin** という名前の論理ボリュームを作成し、その論理ボリュームから、**snap** という名前のスナップショットボリュームを作成します。

1. ボリュームグループ **VG** を使用して、論理ボリューム **origin** を作成します。

```
# lvcreate -L 1G -n origin VG
  Logical volume "origin" created.
```

2. 名前が **snap** で、サイズが 100 MB のスナップショット論理ボリュームを作成します。これは、**/dev/VG/origin** のスナップショット論理ボリュームになります。元の論理ボリュームにファイルシステムが含まれている場合は、任意のディレクトリー上でスナップショット論理ボリュームをマウントしてから、そのファイルシステムのコンテンツにアクセスし、元のファイルシステムが更新を継続している間にバックアップを実行できます。

```
# lvcreate --size 100M --snapshot --name snap /dev/VG/origin
  Logical volume "snap" created.
```

3. 論理ボリューム **/dev/VG/origin** のステータスを表示します。ここでは、スナップショットのすべての論理ボリュームとそのステータス (アクティブかどうか) を表示します。

```
# lvdisplay /dev/VG/origin
  --- Logical volume ---
  LV Path                /dev/VG/origin
  LV Name                origin
  VG Name                VG
  LV UUID                EsFoBp-CB9H-Epl5-pUO4-Yevi-EdFS-xtFnaF
  LV Write Access        read/write
  LV Creation host, time host-083.virt.lab.msp.redhat.com, 2019-04-11 14:45:06 -0500
  LV snapshot status     source of
                         snap [active]
  LV Status              available
  # open                 0
  LV Size                1.00 GiB
  Current LE             256
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     8192
  Block device           253:6
```

4. デフォルトでは **lvs** コマンドは、複製元のボリュームと、使用されているスナップショットボリュームの現在の割合を表示します。以下の例は、スナップショットボリュームを作成したあとの **lvs** コマンドのデフォルト出力を示しています。これには、論理ボリュームを構成するデバイスを含む表示が含まれます。

```
# lvs -a -o +devices
  LV     VG Attr       LSize   Pool   Origin Data%  Meta%  Move Log Cpy%Sync Convert
Devices
  origin VG owi-a-s---   1.00g                                             /dev/sde1(0)
  snap   VG swi-a-s--- 100.00m       origin 0.00                           /dev/sde1(256)
```

警告

複製元ボリュームが変更すると、スナップショットのサイズが拡大されるため、**lvs** コマンドを使用して、スナップショットボリュームのパーセンテージを定期的に監視して、満杯にならないように確認することが重要です。スナップショットは、100% になると完全に消失します。これは、複製元ボリュームの変更されていない部分に書き込みが行われるため、スナップショットが必ず破損するためです。

スナップショットが満杯になると、スナップショット自体が無効になるだけでなく、そのスナップショットデバイスにマウントされているすべてのファイルシステムのマウントが強制的に解除されます。これにより、マウントポイントへのアクセス時に必ず発生するファイルシステムエラーを回避できます。さらに、**lvm.conf** ファイルで **snapshot_autoextend_threshold** オプションを指定できます。このオプションにより、スナップショットの残りの領域が設定されたしきい値を下回ると、常にスナップショットを自動的に拡張できるようになりました。この機能の利用に際しては、ボリュームグループ内に未割り当ての領域があることが条件になります

LVM では、複製元ボリュームのサイズよりも大きく、そのボリュームのメタデータを必要とするスナップショットボリュームを作成できません。同様に、スナップショットの自動拡張を実行しても、スナップショットに必要なサイズとして計算される最大サイズを超えて拡張されることはありません。スナップショットのサイズが複製元のボリュームを包含できるまで拡大されると、スナップショットの自動拡張はモニターされなくなります。

snapshot_autoextend_threshold および **snapshot_autoextend_percent** の設定の詳細は、**/etc/lvm/lvm.conf** ファイルを参照してください。

8.3. スナップショットボリュームのマージ

lvconvert コマンドの **--merge** オプションを使用して、スナップショットを複製元のボリュームにマージできます。複製元とスナップショットボリュームの両方が閉じている状態だと、マージはただちに開始します。そうでない場合は、複製元またはスナップショットのいずれかがアクティブになり、かつ両方が閉じられている状態に初めてなったときにマージが開始します。root ファイルシステムのように、閉じることができない複製元へのスナップショットのマージは、次に複製元ボリュームがアクティブになるまで行われません。マージが開始すると、マージ後の論理ボリュームには、複製元の名前、マイナー番号、UUID が入ります。マージの進行中、複製元に対する読み取りまたは書き込みは、マージ中のスナップショットに対して実行されているかのように見えます。マージが完了すると、マージされたスナップショットは削除されます。

以下のコマンドは、スナップショットボリューム **vg00/lvol1_snap** をその複製元にマージします。

```
# lvconvert --merge vg00/lvol1_snap
```

コマンドライン上で複数のスナップショットを指定したり、LVM オブジェクトタグを使用して複数のスナップショットをそれぞれの複製元にマージしたりすることが可能です。以下の例では、論理ボリューム **vg00/lvol1**、**vg00/lvol2**、および **vg00/lvol3** にはすべて **@some_tag** タグが付きます。以下のコマンドは、この 3 つのボリュームのスナップショット論理ボリュームを連続的にマージします。マージは **vg00/lvol1**、**vg00/lvol2**、**vg00/lvol3** の順で行われます。**--background** オプションを使用している場合は、すべてのスナップショット論理ボリュームのマージが並行して開始されます。

```
# lvconvert --merge @some_tag
```

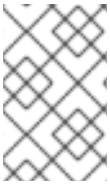
lvconvert --merge コマンドの詳細は、man ページの **lvconvert**(8) を参照してください。

第9章 シンプロビジョニングされた論理ボリューム (シンボリューム) の作成および管理

論理ボリュームは、シンプロビジョニングにできます。これにより、利用可能なエクステントよりも大きな論理ボリュームを作成できます。

9.1. シンプロビジョニングされた論理ボリューム (シンボリューム)

論理ボリュームのシンプロビジョニングが可能になりました。これにより、利用可能なエクステントよりも大きい論理ボリュームを作成できます。シンプロビジョニングを使用すると、空き領域のストレージプール (シンプールと呼ばれる) を管理して、アプリケーションで必要なときに、任意の数のデバイスに割り当てることができます。シンプールにバインドされ、アプリケーションが実際に論理ボリュームに書き込むときにあとで割り当てるデバイスを作成できます。シンプールは、ストレージ領域をコスト効率よく割り当てる必要がある場合は、動的に拡張できます。



注記

シンボリュームは、クラスターのノード間ではサポートされません。シンプールとそのすべてのシンボリュームは、1 つのクラスターノードでのみ排他的にアクティブにする必要があります。

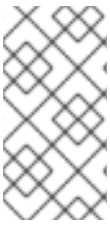
ストレージ管理者は、シンプロビジョニングを使用することで物理ストレージをオーバーコミットできるため、多くの場合は、追加のストレージを購入する必要がなくなります。たとえば、10 人のユーザーから、各自のアプリケーションに使用するファイルシステムをそれぞれ 100GB 要求された場合、ストレージ管理者は各ユーザーに 100GB のファイルシステムを作成します (ただし、実際には 100GB 未満のストレージが、必要に応じて使用されます)。シンプロビジョニングを使用する場合は、ストレージ管理者がストレージプールを監視し、容量が満杯になり始めたら容量を追加することが重要です。

利用可能な領域をすべて使用できるようにするために、LVM はデータの破棄に対応します。これにより、破棄されたファイルや、その他のブロック範囲で以前に使用された領域を再利用できます。

シンボリュームは、新たに実装されたコピーオンライト (COW) スナップショット論理ボリュームに対応します。これにより、多くの仮想デバイスでシンプール内の同一データを共有できます。

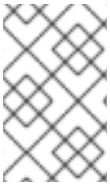
9.2. シンプロビジョニングされた論理ボリュームの作成

論理ボリュームのシンプロビジョニングが可能になりました。これにより、利用可能なエクステントよりも大きい論理ボリュームを作成できます。シンプロビジョニングを使用すると、空き領域のストレージプール (シンプールと呼ばれる) を管理して、アプリケーションで必要なときに、任意の数のデバイスに割り当てることができます。シンプールにバインドされ、アプリケーションが実際に論理ボリュームに書き込むときにあとで割り当てるデバイスを作成できます。シンプールは、ストレージ領域をコスト効率よく割り当てる必要がある場合は、動的に拡張できます。



注記

このセクションでは、シンプロビジョニングされた論理ボリュームを作成し、拡張するために使用する基本的なコマンドの概要を説明します。LVM シンプロビジョニングの詳細情報と、シンプロビジョニングされた論理ボリュームと共に LVM コマンドおよびユーティリティーを使用する方法は、man ページの **lvmthin**(7) を参照してください。



注記

シンボリュームは、クラスターのノード間ではサポートされません。シンプールとそのすべてのシンボリュームは、1つのクラスターノードでのみ排他的にアクティブにする必要があります。

シンボリュームを作成するには、以下のタスクを実行します。

1. **vgcreate** コマンドを使用して、ボリュームグループを作成します。
2. **lvcreate** コマンドを使用して、シンプールを作成します。
3. **lvcreate** コマンドを使用して、シンプール内にシンボリュームを作成します。

lvcreate コマンドに **-T** (または **--thin**) オプションを付けて、シンプールまたはシンボリュームを作成できます。また、**lvcreate** の **-T** オプションを使用して、1つのコマンドで同時にプール内にシンプールとシンプロビジョニングされたボリュームの両方を作成することもできます。

以下のコマンドは、**lvcreate** コマンドの **-T** オプションを使用して、**mythinpool** という名前のシンプールを作成します。これは、ボリュームグループ **vg001** 内にあり、サイズは 100M です。物理領域のプールを作成しているため、プールのサイズを指定する必要があります。**lvcreate** コマンドの **-T** オプションは引数を取りません。コマンドで指定する他のオプションから、作成されるデバイスのタイプが推定されます。

```
# lvcreate -L 100M -T vg001/mythinpool
  Rounding up size to full physical extent 4.00 MiB
  Logical volume "mythinpool" created
# lvs
  LV           VG     Attr     LSize   Pool Origin Data%  Move Log Copy% Convert
  my mythinpool vg001  twi-a-tz 100.00m             0.00
```

以下のコマンドは、**lvcreate** コマンドに **-T** オプションを使用して、シンプール **vg001/mythinpool** に **thinvolume** という名前のシンボリュームを作成します。ここでは、仮想サイズを指定して、ボリュームを含むプールよりも大きなボリュームの仮想サイズを指定している点に注意してください。

```
# lvcreate -V 1G -T vg001/mythinpool -n thinvolume
  Logical volume "thinvolume" created
# lvs
  LV         VG       Attr     LSize   Pool       Origin Data%  Move Log Copy%  Convert
  mythinpool  vg001    twi-a-tz 100.00m                      0.00
  thinvolume  vg001    Vwi-a-tz   1.00g mythinpool          0.00
```

以下のコマンドは、**lvcreate** コマンドに **-T** オプションを使用して、プール内にシンプールとシンプロビジョニングされたボリュームを作成します。その際、**lvcreate** コマンドでサイズと仮想サイズの引数を指定します。また、このコマンドは、ボリュームグループ **vg001** にシンプール **mythinpool** を作成し、そのプールにシンプロビジョニングされたボリューム **thinvolume** も作成します。

```
# lvcreate -L 100M -T vg001/mythinpool -V 1G -n thinvolume
  Rounding up size to full physical extent 4.00 MiB
  Logical volume "thinvolume" created
# lvs
  LV          VG       Attr     LSize   Pool     Origin Data%  Move Log Copy%  Convert
  mythinpool   vg001    twi-a-tz 100.00m                 0.00
  thinvolume   vg001    Vwi-a-tz   1.00g mythinpool       0.00
```

また、**lvcreate** コマンドの **--thinpool** パラメーターを指定して、シンプールを作成することもできます。**-T** オプションとは異なり、**--thinpool** パラメーターには、作成しているシンプール論理ボリューム名の引数が必要です。以下の例は、**lvcreate** で **--thinpool** パラメーターを指定して、**mythinpool** という名前のシンプールを作成します。これは、ボリュームグループ **vg001** 内にあり、サイズは 100M です。

```
# lvcreate -L 100M --thinpool mythinpool vg001
  Rounding up size to full physical extent 4.00 MiB
  Logical volume "mythinpool" created
# lvs
  LV         VG     Attr     LSize   Pool Origin Data%  Move Log Copy% Convert
  mythinpool  vg001  twi-a-tz 100.00m             0.00
```

ストライピングはプールを作成するためにサポートされています。以下のコマンドは、2 つの 64 kB のストライプがあり、チャンクサイズが 256 kB のボリュームグループ **vg001** 内に **pool** という名前の 100M のシンプールを作成します。また、1T のシンボリューム **vg00/thin_lv** も作成します。

```
# lvcreate -i 2 -I 64 -c 256 -L 100M -T vg00/pool -V 1T --name thin_lv
```

lvextend コマンドを使用して、シンボリュームのサイズを拡張できます。ただし、シンプールのサイズを縮小することはできません。

以下のコマンドは、既存のシンプールのサイズ (100M) を変更し、100M 拡張します。

```
# lvextend -L+100M vg001/mythinpool
  Extending logical volume mythinpool to 200.00 MiB
  Logical volume mythinpool successfully resized
# lvs
  LV          VG       Attr     LSize   Pool     Origin Data%  Move Log Copy%  Convert
  mythinpool   vg001    twi-a-tz 200.00m                 0.00
  thinvolume   vg001    Vwi-a-tz   1.00g mythinpool          0.00
```

他の論理ボリュームのタイプと同様に、**lvrename** を使用してボリューム名を変更したり、**lvremove** を使用してボリュームを削除したりできます。また、**lvs** コマンドおよび **lvdisplay** コマンドを使用して、ボリュームの情報を表示することもできます。

デフォルトでは、**lvcreate** コマンドは、計算式 (Pool_LV_size / Pool_LV_chunk_size * 64) から、シンプールのメタデータ論理ボリュームのサイズを設定します。スナップショットが大量にある場合や、シンプールのサイズが小さく、後でシンプールのサイズが急激に大きくなることが予測される場合は、**lvcreate** コマンドの **--poolmetadatasize** パラメーターで、シンプールのメタデータボリュームのデフォルト値を大きくしないといけない場合があります。シンプールのメタデータ論理ボリュームでサポートされる値は 2MiB ~ 16GiB です。

lvconvert コマンドの **--thinpool** パラメーターを使用して、既存の論理ボリュームをシンプールボリュームに変換できます。既存の論理ボリュームをシンプールボリュームに変換する場合は、**lvconvert** コマンドの **--thinpool** パラメーターとともに **--poolmetadata** パラメーターを使用して、既存の論理ボリュームをシンプールボリュームのメタデータボリュームに変換する必要があります。

注記

論理ボリュームをシンプールボリュームまたはシンプールメタデータボリュームに変換すると、論理ボリュームのコンテンツが破棄されます。この場合、**lvconvert** はデバイスのコンテンツを保存するのではなく、コンテンツを上書きするためです。

以下の例は、ボリュームグループ **vg001** の既存の論理ボリューム **lv1** を、シンプールボリュームに変換します。また、ボリュームグループ **vg001** の既存の論理ボリューム **lv2** を、そのシンプールボリュームのメタデータボリュームに変換します。

```
# lvconvert --thinpool vg001/lv1 --poolmetadata vg001/lv2
  Converted vg001/lv1 to thin pool.
```

9.3. シンプロビジョニングされたスナップショットボリューム

Red Hat Enterprise Linux は、シンプロビジョニングされたスナップショットボリュームに対応します。シンプロビジョニングされたスナップショットボリュームにより、多くの仮想デバイスを同じデータボリュームに格納できます。これにより管理が簡略化され、スナップショットボリューム間でのデータ共有が可能になります。

シンボリュームや、LVM スナップショットボリュームの場合、シンプロビジョニングされたスナップショットボリュームは、クラスターのノード間ではサポートされていません。スナップショットボリュームは、1つのクラスターノードで排他的にアクティブにする必要があります。

シンプロビジョニングされたスナップショットボリュームの利点は以下のとおりです。

- 同じボリュームからのスナップショットが複数ある場合に、シンプロビジョニングされたスナップショットボリュームを使用すれば、ディスクの使用量を減らすことができます。
- 複製元が同じスナップショットが複数ある場合は、複製元に1回書き込むことにより1回の COW 操作でデータを保存できます。複製元のスナップショットの数を増やしても、速度が大幅に低下することはありません。
- シンプロビジョニングされたスナップショットボリュームは、別のスナップショットの作成元の論理ボリュームとして使用できます。これにより、再帰的スナップショット (スナップショットのスナップショットのそのまたスナップショットなど) の深度を任意に決定できます。
- シン論理ボリュームのスナップショットにより、シン論理ボリュームを作成することもできます。COW 操作が必要になるまで、あるいはスナップショット自体が書き込まれるまで、データ領域は消費されません。
- シンスナップショットボリュームは、複製元とともにアクティブにしておく必要はありません。そのため、スナップショットボリュームが多数ある場合は、複製元のみをアクティブにし、スナップショットボリュームはアクティブにしないでおくことができます。
- シンプロビジョニングされたスナップショットボリュームの複製元を削除すると、そのボリュームのスナップショットは、それぞれ独立したシンプロビジョニングボリュームになります。したがって、スナップショットとその複製元のボリュームをマージする代わりに、複製元のボリュームを削除し、その独立したボリュームを新たな複製元ボリュームにして、シンプロビジョニングされたスナップショットを新たに作成できます。

シンスナップショットボリュームを使用する利点は数多くありますが、古い LVM スナップショットボリューム機能の方がニーズに沿うケースもあります。

- シンプールのチャンクサイズは変更できません。シンプールのチャンクサイズが大きい場合 (1MB など) や、チャンクサイズが短時間のスナップショットには効率的でない場合は、代わりに以前のスナップショット機能を使用できます。
- シンスナップショットボリュームのサイズを制限することはできません。スナップショットは、必要な場合はシンプール内の全領域を使用するため、ニーズに沿っていない場合があります。

一般的には、使用するスナップショットの形式を決定する際に、使用しているサイトの特定要件を考慮に入れるようにしてください。

9.4. シンプロビジョニングされたスナップショットボリュームの作成

Red Hat Enterprise Linux は、シンプロビジョニングのスナップショットボリュームのサポートを提供します。

注記

このセクションでは、シンプロビジョニングされたスナップショットボリュームを作成し、拡張するために使用する基本的なコマンドの概要を説明します。LVM シンプロビジョニングの詳細情報と、シンプロビジョニングされた論理ボリュームと共に LVM コマンドおよびユーティリティーを使用する方法は、man ページの **lvmthin**(7) を参照してください。

重要

シンプロビジョニングされたスナップショットボリュームを作成する場合、ボリュームのサイズは指定しません。サイズパラメーターを指定すると、作成されるスナップショットはシンプロビジョニングされたスナップショットボリュームにはならず、データを保管するためにシンプールを使用することもありません。たとえば、**lvcreate -s vg/thinvolume -L10M** コマンドは、作成元ボリュームがシンボリュームであっても、シンプロビジョニングされたスナップショット (シンスナップショット) を作成しません。

シンスナップショットは、シンプロビジョニングされた作成元ボリューム用に作成するか、またはシンプロビジョニングされない作成元ボリューム用にも作成できます。

lvcreate コマンドで、**--name** オプションを使用してスナップショットボリューム名を指定できます。以下のコマンドは、シンプロビジョニングされた論理ボリューム **vg001/thinvolume** で、シンプロビジョニングされたスナップショットボリューム (**mysnapshot1**) を作成します。

```
# lvcreate -s --name mysnapshot1 vg001/thinvolume
  Logical volume "mysnapshot1" created
# lvs
  LV          VG      Attr     LSize   Pool       Origin     Data%  Move Log Copy%  Convert
  mysnapshot1 vg001   Vwi-a-tz   1.00g mythinpool thinvolume   0.00
  mythinpool  vg001   twi-a-tz 100.00m                        0.00
  thinvolume  vg001   Vwi-a-tz   1.00g mythinpool             0.00
```

シンスナップショットボリュームには、他のシンボリュームと同じ特性があります。ボリュームのアクティブ化、拡張、名前変更、削除、さらにはスナップショット作成も個別に行うことができます。

デフォルトで、スナップショットボリュームは、通常のアクティブ化コマンドの実行時に省略されます。スナップショットボリュームのアクティブ化を制御する方法は、「論理ボリュームのアクティブ化」 を参照してください。

シンプロビジョニングされていない論理ボリュームの、シンプロビジョニングされたスナップショットを作成することもできます。シンプロビジョニングされていない論理ボリュームはシンプール内に含まれていないため、**外部の複製元** と呼ばれます。外部の複製元ボリュームは、複数の異なるシンプールの、多くのシンプロビジョニングされたスナップショットボリュームにより使用され、共有されることも可能です。外部の複製元は、シンプロビジョニングされたスナップショットが作成される際に非アクティブであり、かつ読み取り専用である必要があります。

外部の複製元のシンプロビジョニングされたスナップショットを作成するには、**--thinpool** オプションを指定する必要があります。以下のコマンドは、読み取り専用の非アクティブなボリューム **origin_volume** のシンスナップショットボリュームを作成します。このシンスナップショットボリュームの名前は **mythinsnap** です。論理ボリューム **origin_volume** は、既存のシンプール **vg001/pool** を使用する、ボリュームグループ **vg001** 内のシンスナップショットボリューム **mythinsnap** に対する外部の複製元になります。複製元ボリュームは、スナップショットボリュームと同じボリュームグループ内に存在する必要があるため、複製元の論理ボリュームを指定する場合にボリュームグループを指定する必要はありません。

```
# lvcreate -s --thinpool vg001/pool origin_volume --name mythinsnap
```

以下のコマンドにあるように、最初のスナップショットボリュームの 2 番目のシンプロビジョニングされたスナップショットボリュームを作成できます。

```
# lvcreate -s vg001/mythinsnap --name my2ndthinsnap
```

lvs コマンドのレポートフィールド **lv_ancestors** および **lv_descendants** 指定すると、シンスナップショット論理ボリュームのすべての先祖 (ancestor) と子孫 (descendant) をそれぞれ表示できます。

以下の例では、下記の点を前提としています。

- **stack1** は、ボリュームグループ **vg001** で元となるボリュームです。
- **stack2** は、**stack1** のスナップショットです。
- **stack3** は、**stack2** のスナップショットです。
- **stack4** は、**stack3** のスナップショットです。

さらに

- **stack5** も、**stack2** のスナップショットです。
- **stack6** は、**stack5** のスナップショットです。

```
$ lvs -o name,lv_ancestors,lv_descendants vg001
  LV      Ancestors             Descendants
  stack1                        stack2,stack3,stack4,stack5,stack6
  stack2  stack1                stack3,stack4,stack5,stack6
  stack3  stack2,stack1         stack4
  stack4  stack3,stack2,stack1
  stack5  stack2,stack1         stack6
  stack6  stack5,stack2,stack1
  pool
```

注記

lv_ancestors フィールドおよび **lv_descendants** フィールドは、既存の依存関係を表示しますが削除されたエントリーは追跡しません。このチェーンの最中にエントリーが削除されると、依存関係チェーンが壊れるからです。たとえば、この設定例から論理ボリューム **stack3** を削除すると、以下のように表示されます。

```
$ lvs -o name,lv_ancestors,lv_descendants vg001
  LV      Ancestors             Descendants
  stack1                        stack2,stack5,stack6
  stack2  stack1                stack5,stack6
  stack4
  stack5  stack2,stack1         stack6
  stack6  stack5,stack2,stack1
  pool
```

ただし、削除した論理ボリュームを追跡して表示するようにシステムを設定できます。**lv_ancestors_full** フィールドおよび **lv_descendants_full** フィールドを指定することで、このボリュームを含む完全依存チェーンを表示できます。

9.5. 削除されているシンスナップショットボリュームの追跡および表示

lvm.conf 設定ファイルで **record_lvs_history** メタデータオプションを有効にして、削除したシンスナップショットとシン論理ボリュームを追跡するように設定します。これにより、元の依存関係チェーンから削除し、**過去** の論理ボリュームになった論理ボリュームを含む、完全シンスナップショット依存関係を表示できます。

lvm.conf 設定ファイルで **lvs_history_retention_time** メタデータオプションを使用し、保持時間 (秒) を指定して、決められた期間、システムに過去のボリュームを保持するように設定できます。

過去の論理ボリュームは、削除された論理ボリュームを単純化したものを保持し、以下の、ボリュームのレポートフィールドを含みます。

- **lv_time_removed** - 論理ボリュームの削除時間
- **lv_time** - 論理ボリュームの作成時間
- **lv_name** - 論理ボリュームの名前
- **lv_uuid** - 論理ボリュームの UUID
- **vg_name** - 論理ボリュームを含むボリュームグループ

ボリュームを削除すると、過去の論理ボリューム名には頭にハイフンが付きます。たとえば、論理ボリューム **lvol1** を削除すると、過去のボリューム名は **-lvol1** となります。過去の論理ボリュームは再アクティベートすることができません。

record_lvs_history メタデータオプションを有効にしていても、**lvremove** コマンドの **--nohistory** オプションを指定して論理ボリュームを削除すれば、過去の論理ボリュームを個別に保持しないようにできます。

ボリューム表示に過去の論理ボリュームを含むには、LVM 表示コマンドに **-H|--history** オプションを指定します。**-H** オプションとともに、レポートフィールド **lv_full_ancestors** および **lv_full_descendants** を指定すると、過去のボリュームを含む完全なシンスナップショット依存関係チェーンを表示できます。

以下のコマンド群は、過去の論理ボリュームを表示および管理する例を示します。

1. **lvm.conf** ファイルに **record_lvs_history=1** を設定して、過去の論理ボリュームを保持します。このメタデータオプションは、デフォルトでは有効ではありません。

2. 以下のコマンドを入力して、シンプロビジョニングスナップショットチェーンを表示します。この例では、以下のように設定されています。

   - **lvol1** は元となるボリュームで、チェーンの中で最初のボリュームになります。
   - **lvol2** は、**lvol1** のスナップショットです。
   - **lvol3** は、**lvol2** のスナップショットです。
   - **lvol4** は、**lvol3** のスナップショットです。
   - **lvol5** は、**lvol3** のスナップショットです。
     例の **lvs** 表示コマンドに **-H** オプションを追加しても、シンスナップショットボリュームは削除されていないため、過去の論理ボリュームは表示されません。

     ```
     # lvs -H -o name,full_ancestors,full_descendants
       LV    FAncestors        FDescendants
       lvol1                   lvol2,lvol3,lvol4,lvol5
       lvol2 lvol1             lvol3,lvol4,lvol5
       lvol3 lvol2,lvol1       lvol4,lvol5
       lvol4 lvol3,lvol2,lvol1
       lvol5 lvol3,lvol2,lvol1
       pool
     ```

3. スナップショットチェーンから論理ボリューム **lvol3** を削除してから再度 **lvs** コマンドを実行し、過去の論理ボリュームが、先祖 (ancestor) と子孫 (descendant) とともに、どのように表示されるかを確認します。

   ```
   # lvremove -f vg/lvol3
     Logical volume "lvol3" successfully removed
   # lvs -H -o name,full_ancestors,full_descendants
     LV     FAncestors         FDescendants
     lvol1                     lvol2,-lvol3,lvol4,lvol5
     lvol2  lvol1               -lvol3,lvol4,lvol5
     -lvol3 lvol2,lvol1         lvol4,lvol5
     lvol4  -lvol3,lvol2,lvol1
     lvol5  -lvol3,lvol2,lvol1
     pool
   ```

4. 過去のボリュームが削除された時間を表示するには、**lv_time_removed** レポートフィールドを使用できます。

   ```
   # lvs -H -o name,full_ancestors,full_descendants,time_removed
     LV     FAncestors         FDescendants               RTime
     lvol1                     lvol2,-lvol3,lvol4,lvol5
     lvol2  lvol1               -lvol3,lvol4,lvol5
     -lvol3 lvol2,lvol1         lvol4,lvol5                2016-03-14 14:14:32 +0100
     lvol4  -lvol3,lvol2,lvol1
     lvol5  -lvol3,lvol2,lvol1
     pool
   ```

5. **vgname/lvname** フォーマットを以下の例のように指定すると、表示コマンドで過去の論理ボリュームを個別に参照できます。**lv_attr** フィールドの 5 番目の例を **h** に設定して、ボリュームが過去のボリュームであることを示すように設定されています。

```
# lvs -H vg/-lvol3
  LV     VG   Attr       LSize
  -lvol3 vg   ----h-----    0
```

6. ボリュームにライブの子孫がないと、LVM は過去の論理ボリュームを保持しません。これは、スナップショットチェーンの最後に論理ボリュームを削除すると、論理ボリュームが過去の論理ボリュームとして保持されないことを示しています。

```
# lvremove -f vg/lvol5
  Automatically removing historical logical volume vg/-lvol5.
  Logical volume "lvol5" successfully removed
# lvs -H -o name,full_ancestors,full_descendants
  LV     FAncestors          FDescendants
  lvol1                      lvol2,-lvol3,lvol4
  lvol2  lvol1                -lvol3,lvol4
  -lvol3 lvol2,lvol1          lvol4
  lvol4  -lvol3,lvol2,lvol1
  pool
```

7. 以下のコマンドを実行して、ボリューム **lvol1** および **lvol2** を削除します。次に **lvs** コマンドを実行して、ボリュームが削除されるとどのように表示されるかを確認します。

```
# lvremove -f vg/lvol1 vg/lvol2
  Logical volume "lvol1" successfully removed
  Logical volume "lvol2" successfully removed
# lvs -H -o name,full_ancestors,full_descendants
  LV     FAncestors          FDescendants
  -lvol1                     -lvol2,-lvol3,lvol4
  -lvol2 -lvol1               -lvol3,lvol4
  -lvol3 -lvol2,-lvol1        lvol4
  lvol4  -lvol3,-lvol2,-lvol1
  pool
```

8. 過去の論理ボリュームを完全に削除したら、**lvremove** コマンドを再度実行します。以下の例のように、ハイフンが追加された、過去のボリューム名を指定します。

```
# lvremove -f vg/-lvol3
Historical logical volume "lvol3" successfully removed
# lvs -H -o name,full_ancestors,full_descendants
  LV     FAncestors    FDescendants
  -lvol1               -lvol2,lvol4
  -lvol2 -lvol1        lvol4
  lvol4  -lvol2,-lvol1
  pool
```

9. 子孫にライブボリュームが含まれるチェーンがある場合に限り、過去の論理ボリュームは保持されます。これは、以下の例のように、そのボリュームにリンクされている子孫がない場合に、過去の論理ボリュームを削除すると、チェーンの論理ボリュームがすべて削除されることを意味します。

```
# lvremove -f vg/lvol4
  Automatically removing historical logical volume vg/-lvol1.
  Automatically removing historical logical volume vg/-lvol2.
  Automatically removing historical logical volume vg/-lvol4.
  Logical volume "lvol4" successfully removed
```

第10章 LVM キャッシュ論理ボリューム

LVM により、LVM キャッシュ論理ボリュームが完全にサポートされます。キャッシュ論理ボリュームでは、高速なブロックデバイス (SSD ドライブなど) で構成される小さい論理ボリュームが使用されるため、頻繁に使用されるブロックを、小さい高速な論理ボリュームに格納することにより、大きくて低速な論理ボリュームのパフォーマンスが向上します。

10.1. キャッシュボリュームタイプ

LVM キャッシュは、LVM 論理ボリュームタイプを使用します。関連するこれらすべての論理ボリュームは、同じボリュームグループ内に存在する必要があります。

- 複製元論理ボリューム - 大きくて低速な論理ボリューム
- キャッシュプール論理ボリューム - 小さい高速な論理ボリューム。キャッシュデータ論理ボリュームとキャッシュメタデータ論理ボリュームの 2 つのデバイスから構成されます。
- キャッシュデータ論理ボリューム - キャッシュプール論理ボリューム用のデータブロックを含む論理ボリューム
- キャッシュメタデータ論理ボリューム - キャッシュプール論理ボリューム用のメタデータを含む論理ボリューム。データブロックが保存された場所を指定するアカウンティング情報を保持します (たとえば、複製元の論理ボリューム上またはキャッシュデータ論理ボリューム上)。
- キャッシュ論理ボリューム - 作成元の論理ボリュームとキャッシュプール論理ボリュームを含む論理ボリューム。これは、さまざまなキャッシュボリュームコンポーネントをカプセル化する使用可能なデバイスです。

10.2. LVM キャッシュ論理ボリュームの作成

以下の手順に従うと、LVM キャッシュ論理ボリュームを作成できます。

1. 低速な物理ボリュームと高速な物理ボリュームを含むボリュームグループを作成します。この例では、**/dev/sde1** は低速なデバイスであり、**/dev/sdf1** は高速なデバイスです。両方のデバイスは、ボリュームグループ **VG** に含まれます。

```
# pvcreate /dev/sde1
# pvcreate /dev/sdf1
# vgcreate VG /dev/sde1 /dev/sdf1
```

2. 作成元のボリュームを作成します。この例では、サイズが 10 ギガバイトであり、低速な物理ボリュームである **/dev/sde1** から構成される **lv** という名前の作成元ボリュームが作成されます。

```
# lvcreate -L 10G -n lv VG /dev/sde1
```

3. キャッシュプール論理ボリュームを作成します。この例では、ボリュームグループ **VG** に含まれる高速なデバイス **/dev/sdf1** に、**cpool** という名前のキャッシュプール論理ボリュームが作成されます。このコマンドにより作成されるキャッシュプール論理ボリュームは、隠しキャッシュデータ論理ボリューム **cpool_cdata** と、隠しキャッシュメタデータ論理ボリューム **cpool_cmeta** から構成されます。

```
# lvcreate --type cache-pool -L 5G -n cpool VG /dev/sdf1
  Using default stripesize 64.00 KiB.
  Logical volume "cpool" created.
```

```
# lvs -a -o name,size,attr,devices VG
  LV              LSize  Attr       Devices
  cpool             5.00g Cwi---C--- cpool_cdata(0)
  [cpool_cdata]     5.00g Cwi------- /dev/sdf1(4)
  [cpool_cmeta]     8.00m ewi------- /dev/sdf1(2)
  lv               10.00g -wi-a----- /dev/sde1(0)
  [lvol0_pmspare]  8.00m ewi------- /dev/sdf1(0)
```

さらに複雑な設定の場合は、キャッシュデータとキャッシュメタデータ論理ボリュームを個別に作成し、それらのボリュームをキャッシュプール論理ボリュームに結合しないといけない場合があります。この手順は、man ページの **lvmcache**(7) を参照してください。

4. キャッシュプール論理ボリュームを作成元論理ボリュームにリンクして、キャッシュ論理ボリュームを作成します。作成された、ユーザー使用可能なキャッシュ論理ボリュームには、作成元の論理ボリュームの名前が付けられます。作成元の論理ボリュームは、**_corig** が元の名前に追加された状態で隠し論理ボリュームになります。これはライブ変換することが可能ですが、必ず最初にバックアップを実行してください。

```
# lvconvert --type cache --cachepool cpool VG/lv
  Logical volume cpool is now cached.
# lvs -a -o name,size,attr,devices VG
    LV              LSize  Attr       Devices
    [cpool]           5.00g Cwi---C--- cpool_cdata(0)
    [cpool_cdata]     5.00g Cwi-ao---- /dev/sdf1(4)
    [cpool_cmeta]     8.00m ewi-ao---- /dev/sdf1(2)
    lv               10.00g Cwi-a-C--- lv_corig(0)
    [lv_corig]       10.00g owi-aoC--- /dev/sde1(0)
    [lvol0_pmspare]  8.00m ewi------- /dev/sdf1(0)
```

5. 必要に応じて、キャッシュされた論理ボリュームをシンプール論理ボリュームに変換できます。プールから作成されたすべてのシン論理ボリュームがキャッシュを共有することに注意してください。
以下のコマンドでは、シンプールメタデータ (**lv_tmeta**) を割り当てるのに高速なデバイス **/dev/sdf1** を使用します。これは、キャッシュプールボリュームで使用されるのと同じデバイスです。つまり、シンプールメタデータボリュームはキャッシュデータ論理ボリューム **cpool_cdata** とキャッシュメタデータ論理ボリューム **cpool_cmeta** の両方とそのデバイスを共有します。

```
# lvconvert --type thin-pool VG/lv /dev/sdf1
    WARNING: Converting logical volume VG/lv to thin pool's data volume with metadata
wiping.
    THIS WILL DESTROY CONTENT OF LOGICAL VOLUME (filesystem etc.)
 Do you really want to convert VG/lv? [y/n]: y
    Converted VG/lv to thin pool.
# lvs -a -o name,size,attr,devices vg
    LV              LSize  Attr       Devices
    [cpool]           5.00g Cwi---C--- cpool_cdata(0)
    [cpool_cdata]     5.00g Cwi-ao---- /dev/sdf1(4)
    [cpool_cmeta]     8.00m ewi-ao---- /dev/sdf1(2)
    lv               10.00g twi-a-tz-- lv_tdata(0)
    [lv_tdata]       10.00g Cwi-aoC--- lv_tdata_corig(0)
    [lv_tdata_corig] 10.00g owi-aoC--- /dev/sde1(0)
    [lv_tmeta]       12.00m ewi-ao---- /dev/sdf1(1284)
    [lvol0_pmspare]  12.00m ewi------- /dev/sdf1(0)
    [lvol0_pmspare]  12.00m ewi------- /dev/sdf1(1287)
```

その他の管理例を含む LVM キャッシュボリュームの詳細は、man ページの **lvmcache**(7) を参照してください。

第11章 論理ボリュームのアクティブ化

アクティブ状態の論理ボリュームは、ブロックデバイスを介して使用できます。アクティブになっている論理ボリュームにはアクセスにでき、変更できます。論理ボリュームを作成すると、デフォルトでアクティブになります。

個々の論理ボリュームを非アクティブにしてカーネルが認識しないようにする必要がある状況はさまざまです。個々の論理ボリュームは、**lvchange** コマンドの **-a** オプションを使用してアクティブまたは非アクティブにできます。

個々の論理ボリュームを非アクティブにするには、以下のコマンドを実行します。

```
lvchange -an vg/lv
```

個々の論理ボリュームをアクティブにするには、以下のコマンドを実行します。

```
lvchange -ay vg/lv
```

vgchange コマンドの **-a** オプションを使用して、ボリュームグループの論理ボリュームをすべてアクティブまたは非アクティブにできます。これは、ボリュームグループの個々の論理ボリュームに **lvchange -a** コマンドを実行するのと同じです。

ボリュームグループの論理ボリュームをすべて非アクティブにするには、以下のコマンドを実行します。

```
vgchange -an vg
```

ボリュームグループの論理ボリュームをすべてアクティブにするには、以下のコマンドを実行します。

```
vgchange -ay vg
```

11.1. 論理ボリュームの自動アクティブ化の制御

論理ボリュームの自動アクティブ化は、システム起動時に論理ボリュームイベントベースで自動的にアクティブにすることを指します。システムでデバイスが利用可能になると (デバイスのオンラインイベント)、**systemd/udev** は、各デバイスに **lvm2-pvscan** サービスを実行します。このサービスは、named デバイスを読み込む **pvscan --cache -aay device** コマンドを実行します。デバイスがボリュームグループに属している場合、**pvscan** コマンドは、そのボリュームグループに対する物理ボリュームのすべてがそのシステムに存在するかどうかを確認します。存在する場合は、このコマンドが、そのボリュームグループにある論理ボリュームをアクティブにします。

/etc/lvm/lvm.conf 設定ファイルで以下の設定オプションを使用して、論理ボリュームの自動アクティブ化を制御できます。

- **global/event_activation**
  event_activation が無効になっている場合、**systemd/udev** は、システムの起動時に存在する物理ボリュームでのみ論理ボリュームを自動アクティブにします。すべての物理ボリュームが表示されていないと、一部の論理ボリュームが自動的にアクティブにならない場合もあります。

- **activation/auto_activation_volume_list**
  auto_activation_volume_list を空のリストに設定すると、自動アクティベーションは完全に無効になります。特定の論理ボリュームとボリュームグループに **auto_activation_volume_list** を設定すると、自動アクティベーションは設定した論理ボリュームに制限されます。

このオプションの設定は **/etc/lvm/lvm.conf** 設定ファイルを参照してください。

11.2. 論理ボリュームのアクティブ化の制御

以下の方法で、論理ボリュームのアクティブ化を制御できます。

- **/etc/lvm/conf** ファイルの **activation/volume_list** 設定で行います。これにより、どの論理ボリュームをアクティブにするかを指定できます。このオプションの使用方法は **/etc/lvm/lvm.conf** 設定ファイルを参照してください。
- 論理ボリュームのアクティブ化スキップフラグで行います。このフラグが論理ボリュームに設定されていると、通常のアクティベーションコマンド時にそのボリュームがスキップされます。

以下の方法で、論理ボリュームのアクティブ化スキップフラグを設定できます。

- **lvcreate** コマンドの **-kn** オプションまたは **--setactivationskip n** オプションを指定して、論理ボリュームの作成時にアクティブ化スキップフラグをオフにできます。
- **lvchange** コマンドの **-kn** オプションまたは **--setactivationskip n** オプションを指定して、既存の論理ボリュームのアクティブ化スキップフラグをオフにできます。
- **lvchange** コマンドの **-ky** オプションまたは **--setactivationskip y** オプションを指定して、オフになっているボリュームに対して、アクティベーションスキップフラグを再度オンにします。

このアクティブ化スキップフラグが論理ボリュームに設定されているかを確認するには、**lvs** コマンドを実行します。以下のような **k** 属性が表示されます。

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

標準オプション **-ay** または **--activate y** の他に、**-K** オプションまたは **--ignoreactivationskip** オプションを使用して、**k** 属性セットで論理ボリュームをアクティブにできます。

デフォルトでは、新スナップショットボリュームは、作成時にアクティブ化スキップのフラグが付いています。**/etc/lvm/lvm.conf** ファイルの **auto_set_activation_skip** 設定でこの新しいシンスナップショットボリュームのデフォルトのアクティブ化スキップ設定を制御できます。

以下のコマンドは、アクティブ化スキップフラグが設定されているシンスナップショット論理ボリュームをアクティベートします。

```
# lvchange -ay -K VG/SnapLV
```

以下のコマンドは、アクティブ化スキップフラグがない、シンスナップショットを作成します。

```
# lvcreate --type thin -n SnapLV -kn -s ThinLV --thinpool VG/ThinPoolLV
```

以下のコマンドは、スナップショット論理ボリュームから、アクティブ化スキップフラグを削除します。

```
# lvchange -kn VG/SnapLV
```

11.3. 共有論理ボリュームのアクティベーション

以下のように、**lvchange** コマンドおよび **vgchange** コマンドの **-a** オプションを使用して、共有論理ボリュームの論理ボリュームのアクティブ化を制御できます。

コマンド	アクティベーション
lvchange -ay|e	共有論理ボリュームを排他モードでアクティベートし、1台のホストのみが論理ボリュームをアクティベートできるようにします。アクティベーションが失敗 (論理ボリュームが別のホストでアクティブの場合に発生する可能性あり) すると、エラーが報告されます。
lvchange -asy	共有モードで共有論理ボリュームをアクティブにし、複数のホストが論理ボリュームを同時にアクティブにできるようにします。アクティブできない場合は (その論理ボリュームは別のホストで排他的にアクティブになっている場合に発生する可能性あり)、エラーが報告されます。スナップショットなど、論理タイプが共有アクセスを禁止している場合は、コマンドがエラーを報告して失敗します。複数のホストから同時に使用できない論理ボリュームタイプには、シン、キャッシュ、RAID、およびスナップショットがあります。
lvchange -an	論理ボリュームを非アクティブにします。

11.4. 欠落しているデバイスを含む論理ボリュームのアクティブ化

lvchange コマンドで、**activation_mode** パラメーターを以下のいずれかの値に設定することで、デバイスが見つからない論理ボリュームをアクティブにする方法を設定できます。

アクティベーションモード	意味
complete	物理ボリュームが見つからない論理ボリュームのみをアクティベートすることを許可します。これが最も制限的なモードです。
degraded	物理ボリュームが見つからない RAID 論理ボリュームをアクティベートすることを許可します。
partial	物理ボリュームが見つからない任意の論理ボリュームをアクティベートすることを許可します。このオプションは、回復または修復にのみ使用してください。

activation_mode のデフォルト値は、**/etc/lvm/lvm.conf** ファイルの **activation_mode** 設定により決まります。詳細情報は、man ページの **lvmraid**(7) を参照してください。

第12章 LVM デバイススキャンの制御

/etc/lvm/lvm.conf ファイルにフィルターを設定することで、LVM デバイスのスキャンを制御できます。**lvm.conf** ファイルのフィルターは、**/dev** ディレクトリーのデバイス名に適用される一連の単純な正規表現で構成され、見つかった各ブロックデバイスを許可するかどうかを指定します。

12.1. デバイスのスキャンを制御するフィルターの設定

以下の例は、LVM がスキャンするデバイスを制御するフィルターの使用を示しています。正規表現は完全なパス名に対して自由に照合されるため、これらの例の一部は必ずしも推奨される方法ではないことに注意してください。たとえば、**a/loop/** は **a/.*loop.*/** と同等であり、**/dev/solooperation/lvol1** に一致します。

以下のフィルターは、検出されたすべてのデバイスを追加します。これは、設定ファイル内で設定されているフィルターはないため、デフォルトの動作になります。

```
filter = [ "a/.*/" ]
```

以下のフィルターは、ドライブにメディアが入っていない場合の遅延を回避するために cdrom デバイスを削除します。

```
filter = [ "r|/dev/cdrom|" ]
```

以下のフィルターはすべてのループを追加して、その他のすべてのブロックデバイスを削除します。

```
filter = [ "a/loop.*/", "r/.*/" ]
```

以下のフィルターはすべてのループと IDE を追加して、その他のすべてのブロックデバイスを削除します。

```
filter =[ "a|loop.*|", "a|/dev/hd.*|", "r|.*|" ]
```

以下のフィルターは 1 番目の IDE ドライブ上にパーティション 8 のみを追加して、他のすべてのブロックデバイスを削除します。

```
filter = [ "a|^/dev/hda8$|", "r/.*/" ]
```

12.2. LVM コマンドが論理ボリュームをスキャンするかどうかの制御

デフォルトでは、LVM コマンドは、システムの論理ボリュームをスキャンしません。このデフォルトの動作には、次の利点があります。

- システムにアクティブな論理ボリュームが多数ある場合は、LVM コマンドごとに追加の時間が必要になり、パフォーマンスに悪影響を及ぼし、望まない遅延やタイムアウトが発生します。
- 論理ボリュームにゲスト VM イメージの物理ボリュームが含まれている場合、ホストは通常、ゲストに属する階層化された物理ボリュームをスキャンまたは使用しません。ただし、ゲスト VM の物理ボリュームが、ホストから見える SCSI デバイスに直接存在する場合は、ホストの LVM がその物理ボリュームにアクセスできないようにするために、「デバイスのスキャンを制御するフィルターの設定」のようにフィルターを設定する必要があります。

まれに、意図的に論理ボリューム層に物理ボリューム層を載せる場合に、論理ボリュームをスキャンすることが必要になる場合があります。LVM を設定してすべての論理ボリュームをスキャンするに

は、**/etc/lvm/lvm.conf** ファイルの **scan_lvs** 設定オプションを **scan_lvs=1** に設定します。LPM コマンドがどの論理ボリュームをスキャンするかを制限するには、「Configuring filters to control device scanning」 の説明通りに、**/etc/lvm/lvm.conf** 設定ファイルにデバイスフィルターを設定できます。

第13章 LVM の割り当ての制御

デフォルトでは、ボリュームグループは、同じ物理ボリューム上に並行ストライプを配置しないなど、常識的な規則に従って物理エクステントを割り当てます。これは、**normal** 割り当てポリシーです。**vgcreate** コマンドで **--alloc** 引数を使用して、**contiguous**、**anywhere**、または **cling** の割り当てポリシーを指定します。一般的に、**normal** 以外の割り当てポリシーが必要となるのは、通常とは異なる、標準外のエクステント割り当てを必要とする特別なケースのみです。

13.1. LVM の割り当てポリシー

LVM の操作で物理エクステントを単一または複数の論理ボリュームに割り当てる必要がある場合、割り当ては以下のように行われます。

- ボリュームグループ上で割り当てられていない物理エクステントが、割り当ての候補になります。コマンドラインの末尾に物理エクステントの範囲を指定すると、指定した物理ボリュームの中で、その範囲内で割り当てられていない物理エクステントだけが、割り当て用エクステントとして考慮されます。

- 割り当てポリシーは順番に試行されます。最も厳格なポリシー (**contiguous**) から始まり、最後は **--alloc** オプションで指定した割り当てポリシーか、特定の論理ボリュームやボリュームグループにデフォルトとして設定されている割り当てポリシーが試行されます。割り当てポリシーでは、埋める必要がある空の論理ボリューム領域の最小番号の論理エクステントから始まり、割り当てポリシーによる制限に沿って、できるだけ多くの領域の割り当てを行います。領域が足りなくなると、LVM は次のポリシーに移動します。

割り当てポリシーの制限は以下のとおりです。

- **contiguous** の割り当てポリシーでは、論理ボリュームの最初の論理エクステントを除いたすべての論理エクステントは、その直前の論理エクステントに物理的に隣接している必要があります。
  論理ボリュームがストライプ化またはミラー化されている場合は、**contiguous** の割り当て制限が、領域を必要とする各ストライプまたはミラーイメージ (レッグ) に個別に適用されます。

- **cling** の割り当てポリシーでは、既存の論理ボリュームに追加する論理エクステントに使用される物理ボリュームが、その論理ボリュームにある別の (1 つ以上の) 論理エクステントですでに使用されている必要があります。**allocation/cling_tag_list** の設定パラメーターが定義されており、一覧表示されているいずれかのタグが 2 つの物理ボリュームにある場合、この 2 つの物理ボリュームは一致すると見なされます。これにより、割り当てのために、同様のプロパティー (物理的な場所など) を持つ物理ボリュームのグループにタグを付け、その物理ボリュームを同等なものとして処理できます。
  論理ボリュームがストライプ化またはミラー化されると、**cling** の割り当て制限が、領域を必要とする各ストライプまたはミラーイメージ (レッグ) に個別に適用されます。

- **normal** の割り当てポリシーは、並列の論理ボリューム (異なるストライプまたはミラーイメージ/レッグ) 内の同じオフセットで、その並列の論理ボリュームにすでに割り当てられている論理エクステントと同じ物理ボリュームを共有する物理エクステントは選択しません。
  ミラーデータを保持するために、論理ボリュームと同時にミラーログを割り当てる場合、**normal** の割り当てポリシーでは、最初にログとデータに対して、それぞれ別の物理ボリュームを選択しようとします。異なる物理ボリュームを選択できず、かつ **allocation/mirror_logs_require_separate_pvs** 設定パラメーターが 0 に設定されている場合は、データの一部とログが物理ボリュームを共有できるようになります。

  また、シンプールメタデータを割り当てる場合も、**normal** の割り当てポリシーはミラーログを割り当てる場合と同じようになりますが、設定パラメーターは **allocation/thin_pool_metadata_require_separate_pvs** の値が適用されます。

- 割り当て要求を満たすのに十分な空きエクステントがあっても、**normal** の割り当てポリシーで使用されない場合は、たとえ同じ物理ボリュームに 2 つのストライプを配置することでパフォーマンスが低下しても、**anywhere** 割り当てポリシーがその空きエクステントを使用します。

割り当てポリシーは、**vgchange** コマンドで変更できます。

注記

定義された割り当てポリシーに沿って、このセクションで説明されている以上のレイアウトの操作が必要な場合は、今後のバージョンでコードが変更する可能性があることに注意してください。たとえば、割り当て可能な空き物理エクステントの数が同じ 2 つの空の物理ボリュームをコマンドラインで指定する場合、現行バージョンの LVM では、それが表示されている順番に使用が検討されます。ただし、今後のリリースで、そのプロパティーが変更されない保証はありません。特定の論理ボリューム用に特定のレイアウトを取得することが重要な場合は、各手順に適用される割り当てポリシー基づいて LVM がレイアウトを決定することがないように、**lvcreate** と **lvconvert** を順に使用してレイアウトを構築してください。

特定のケースで、割り当てプロセスがどのように行われているかを確認するには、コマンドに **-vvvv** オプションを追加するなどして、デバッグロギングの出力を表示します。

13.2. 物理ボリューム上での割り当て防止

pvchange コマンドを使用すると、単一または複数の物理ボリュームの空き領域で物理エクステントが割り当てられないように設定できます。これは、ディスクエラーが発生した場合や、物理ボリュームを取り除く場合に必要となる可能性があります。

以下のコマンドは、**/dev/sdk1** での物理エクステントの割り当てを無効にします。

```
# pvchange -x n /dev/sdk1
```

pvchange コマンドで **-xy** 引数を使用すると、無効にされていた割り当てを許可できます。

13.3. CLING 割り当てポリシーを使用した論理ボリュームの拡張

LVM ボリュームを拡張する際には、**lvextend** コマンドの **--alloc cling** オプションを使用して、**cling** 割り当てポリシーを指定できます。このポリシーにより、同一の物理ボリュームのスペースが、既存の論理ボリュームの最終セグメントとして選択されます。物理ボリューム上に十分な領域がなく、タグの一覧が **/etc/lvm/lvm.conf** ファイル内で定義されている場合には、LVM は、その物理ボリュームにいずれかのタグが付けられているかを確認し、既存エクステントと新規エクステント間で、物理ボリュームのタグを適合させようとします。

たとえば、ご使用の論理ボリュームが単一のボリュームグループ内の 2 サイト間でミラー化されている場合は、**@site1** タグと **@site2** タグを使用し、サイトの場所に応じて物理ボリュームにタグを付けることができます。この場合は、**lvm.conf** ファイル内に以下の行を指定します。

```
cling_tag_list = [ "@site1", "@site2" ]
```

以下の例では、**lvm.conf** ファイルが変更されて、次のような行が追加されています。

```
cling_tag_list = [ "@A", "@B" ]
```

また、この例では、**/dev/sdb1**、**/dev/sdc1**、**/dev/sdd1**、**/dev/sde1**、**/dev/sdf1**、**/dev/sdg1**、および **/dev/sdh1** の物理ボリュームで構成されるボリュームグループ **taft** が作成されています。これらの物理ボリュームには、**A**、**B** および **C** のタグが付けられています。この例では、**C** のタグは使用されていませんが、LVM がタグを使用して、ミラーレッグに使用する物理ボリュームを選択することを示しています。

```
# pvs -a -o +pv_tags /dev/sd[bcdefgh]
  PV         VG   Fmt  Attr PSize  PFree  PV Tags
  /dev/sdb1  taft lvm2 a--  15.00g 15.00g A
  /dev/sdc1  taft lvm2 a--  15.00g 15.00g B
  /dev/sdd1  taft lvm2 a--  15.00g 15.00g B
  /dev/sde1  taft lvm2 a--  15.00g 15.00g C
  /dev/sdf1  taft lvm2 a--  15.00g 15.00g C
  /dev/sdg1  taft lvm2 a--  15.00g 15.00g A
  /dev/sdh1  taft lvm2 a--  15.00g 15.00g A
```

以下のコマンドは、ボリュームグループ **taft** から 10 ギガバイトのミラー化ボリュームを作成します。

```
# lvcreate --type raid1 -m 1 -n mirror --nosync -L 10G taft
  WARNING: New raid1 won't be synchronised. Don't read what you didn't write!
  Logical volume "mirror" created
```

以下のコマンドは、ミラーレッグおよび RAID メタデータのサブボリュームに使用されるデバイスを表示します。

```
# lvs -a -o +devices
  LV              VG   Attr       LSize  Log Cpy%Sync Devices
  mirror          taft Rwi-a-r--- 10.00g       100.00 mirror_rimage_0(0),mirror_rimage_1(0)
  [mirror_rimage_0] taft iwi-aor--- 10.00g             /dev/sdb1(1)
  [mirror_rimage_1] taft iwi-aor--- 10.00g             /dev/sdc1(1)
  [mirror_rmeta_0]  taft ewi-aor---  4.00m             /dev/sdb1(0)
  [mirror_rmeta_1]  taft ewi-aor---  4.00m             /dev/sdc1(0)
```

以下のコマンドは、ミラー化ボリュームのサイズを拡張します。**cling** 割り当てポリシーで、同じタグが付いた物理ボリュームを使用して、ミラーレッグが拡張される必要があることを示します。

```
# lvextend --alloc cling -L +10G taft/mirror
  Extending 2 mirror images.
  Extending logical volume mirror to 20.00 GiB
  Logical volume mirror successfully resized
```

以下に表示したコマンドは、レッグとして同一のタグが付いた物理ボリュームを使用してミラーレッグが拡張されているのを示しています。**C** のタグが付いた物理ボリュームは無視される点に注意してください。

```
# lvs -a -o +devices
  LV              VG   Attr       LSize  Log Cpy%Sync Devices
  mirror          taft Rwi-a-r--- 20.00g       100.00 mirror_rimage_0(0),mirror_rimage_1(0)
  [mirror_rimage_0] taft iwi-aor--- 20.00g             /dev/sdb1(1)
  [mirror_rimage_0] taft iwi-aor--- 20.00g             /dev/sdg1(0)
  [mirror_rimage_1] taft iwi-aor--- 20.00g             /dev/sdc1(1)
  [mirror_rimage_1] taft iwi-aor--- 20.00g             /dev/sdd1(0)
  [mirror_rmeta_0]  taft ewi-aor---  4.00m             /dev/sdb1(0)
  [mirror_rmeta_1]  taft ewi-aor---  4.00m             /dev/sdc1(0)
```