Red Hat Enterprise Linux 9

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のガイド

Last Updated: 2022-07-10

Red Hat Enterprise Linux 9 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のガイド

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

本書は、Red Hat Enterprise Linux 9 で、LVM 論理ボリュームを管理する方法を説明します。

目次

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、弊社の CTO、Chris Wright のメッセージ を参照してください。

RED HAT ドキュメントへのフィードバックの提供

ご意見ご要望をお聞かせください。ドキュメントの改善点はございませんか。

- 特定の部分についての簡単なコメントをお寄せいただく場合は、以下をご確認ください。
  1. ドキュメントの表示が **Multi-page HTML** 形式になっていて、ドキュメントの右上隅に **Feedback** ボタンがあることを確認してください。
  2. マウスカーソルで、コメントを追加する部分を強調表示します。
  3. 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ます。
  4. 表示される手順に従ってください。
- Bugzilla を介してフィードバックを送信するには、新しいチケットを作成します。
  1. Bugzilla の Web サイトに移動します。
  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 論理ボリューム管理の概要

論理ボリューム管理(LVM)により、物理ストレージ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するのに役立ちます。様々な面で、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が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。

1.1. LVM のアーキテクチャー

以下は、LVM のコンポーネントです。

物理ボリューム

物理ボリューム(PV)は、LVM 使用用に指定されたパーティションまたはディスク全体です。詳細は、LVM 物理ボリュームの管理 を参照してください。

ボリュームグループ

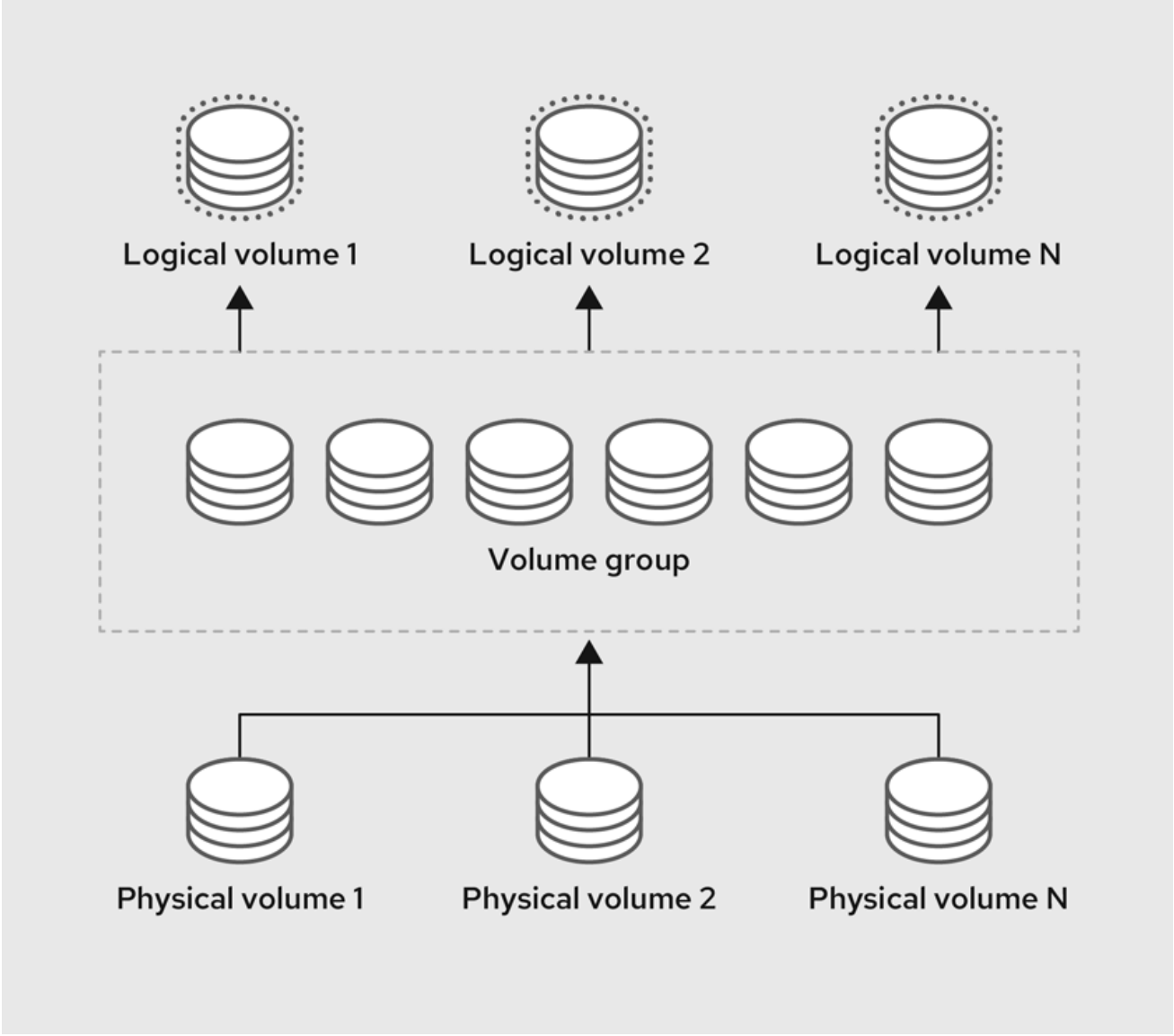
ボリュームグループ(VG)は、物理ボリューム(PV)の集合です。これにより、論理ボリュームに割り当て可能なディスク領域のプールが作成されます。詳細は、「LVM ボリュームグループの管理」 を参照してください。

論理ボリューム

論理ボリュームは、マウント可能なストレージデバイスを表します。詳細は、「LVM 論理ボリュームの管理」 を参照してください。

以下の図は、LVM のコンポーネントを示しています。

図1.1 LVM 論理ボリュームのコンポーネント



1.2. LVM の利点

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ます。この機能を使用すると、ファイルシステムを複数のデバイスにまたがって拡張でき、1 つの大きなファイルシステムとして扱うことができます。

サイズ変更可能なストレージボリューム

基になる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マイズした名前で管理できます。

ストライプ化ボリューム

2 つ以上のデバイス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。

RAID ボリューム

論理ボリュームは、データのRAIDを設定する際に便利な方法を提供します。これにより、デバイス障害に対する保護が可能になり、パフォーマンスが向上します。

ボリュームスナップショット

論理ボリュームの特定の時点のコピーであるスナップショットを作成して、一貫性のあるバックアップを作成したり、実際のデータに影響を与えずに変更の影響をテストしたりすることができます。

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、利用可能な物理容量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は、SSD ドライブなどの高速なブロックデバイスを使用して、大規模で低速なブロックデバイスのパフォーマンスを向上させます。

第2章 LVM 物理ボリュームの管理

物理ボリューム(PV)は、LVM 使用用に指定されたパーティションまたはディスク全体です。LVM 論理ボリューム用にデバイスを使用する場合は、デバイスを物理ボリュームとして初期化する必要があります。

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のみ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。**dd if=/dev/zero of=<PhysicalVolume> bs=512 count=1** コマンドを root として使用し、最初のセクターをゼロにして、既存のパーティションテーブルを削除できます。

2.1. 物理ボリュームの概要

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。以下は、LVM ラベルについて説明しています。

- LVM ラベルにより物理デバイスの正しい識別とデバイスの順序付けが行われます。ラベルが付けられていない、LVM 以外のデバイスは、起動時にシステムが検出した順序に応じて、再起動後に名前が変更される場合があります。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。

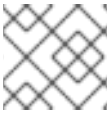
- LVM ラベルは、デフォルトでは 2 番目の 512 バイトセクターに配置されます。物理ボリュームを作成する場合は、先頭の 4 つのセクターのいずれかにラベルを配置することにより、このデフォルト設定を書き換えることができます。これにより、必要に応じて LVM ボリュームを、このセクターを利用する他のユーザーと併用できるようになります。

以下は、LVM メタデータについて説明しています。

- 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。

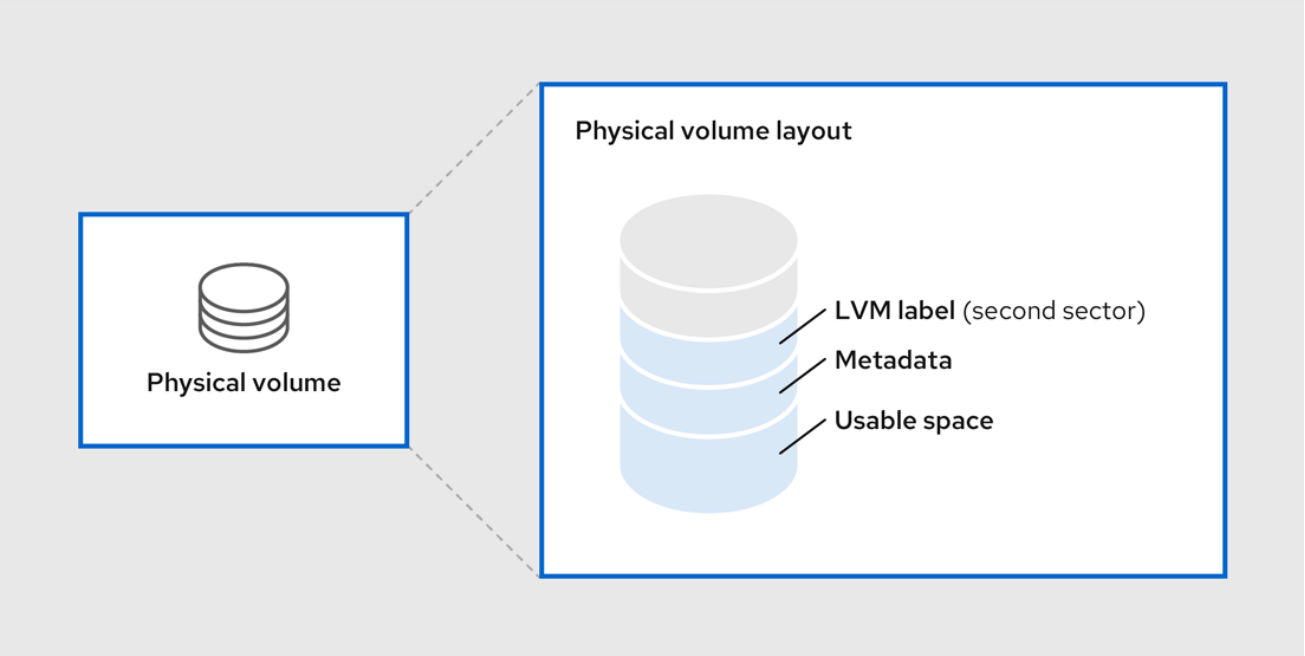
次の図は、LVM 物理ボリュームのレイアウトを示しています。LVM ラベルが 2 番目のセクターにあり、その後にメタデータ領域、使用可能なデバイス領域と続きます。



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2.1 物理ボリュームのレイアウト



関連情報

- ディスク上の複数パーティション

2.2. ディスク上の複数パーティション

LVM を使用して、ディスクパーティションから物理ボリューム(PV)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。

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、改善ではなく低下します。

RAID の冗長性

LVM は、2 つの物理ボリュームが同じデバイスにあるかどうかを判断できません。2 つの物理ボリュームが同じデバイスにある場合に RAID 論理ボリュームを作成すると、パフォーマンスとフォールトトレランスが失われる可能性があります。

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、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。

LVM は、パーティション化していないディスクを物理ボリュームとして使用することをサポートしま

すが、パーティションなしで PV を作成すると、混合オペレーティングシステム環境で問題が発生する可能性があるため、ディスク全体を単一のパーティションとして作成することが推奨されます。他のオペレーティングシステムはデバイスを空き状態として解釈し、ドライブの先頭にある PV ラベルを上書きする可能性があります。

2.3. LVM 物理ボリュームの作成

この手順では、LVM 物理ボリューム (PV) を作成し、ラベルを付ける方法を説明します。

この手順では、**/dev/vdb1**、**/dev/vdb2**、および **/dev/vdb3** を、システムで利用可能なストレージデバイスに置き換えます。

前提条件

- **lvm2** パッケージがインストールされている。

手順

1. **pvcreate** コマンドに、スペースで区切られたデバイス名を引数として使用して、複数の物理ボリュームを作成します。

```
# pvcreate /dev/vdb1 /dev/vdb2 /dev/vdb3
  Physical volume "/dev/vdb1" successfully created.
  Physical volume "/dev/vdb2" successfully created.
  Physical volume "/dev/vdb3" successfully created.
```

   これにより、ラベルが **/dev/vdb1**、**/dev/vdb2**、および **/dev/vdb3** に配置され、それらが LVM に属する物理ボリュームとしてマークされます。

2. 要件に応じて、以下のコマンドのいずれかを使用して、作成した物理ボリュームを表示します。

   a. **pvdisplay** コマンド: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) および他のオプションが、決められた形式で表示されます。

```
# pvdisplay
--- NEW Physical volume ---
  PV Name               /dev/vdb1
  VG Name
  PV Size               1.00 GiB
[..]
--- NEW Physical volume ---
  PV Name               /dev/vdb2
  VG Name
  PV Size               1.00 GiB
[..]
--- NEW Physical volume ---
  PV Name               /dev/vdb3
  VG Name
  PV Size               1.00 GiB
[..]
```

   b. **pvs** コマンド:物理ボリュームの情報を設定可能な形式で出力します。

```
# pvs
  PV         VG  Fmt    Attr    PSize      PFree
/dev/vdb1        lvm2           1020.00m   0
/dev/vdb2        lvm2           1020.00m   0
/dev/vdb3        lvm2           1020.00m   0
```

c. **pvscan** コマンド:システムにある物理ボリュームで対応している LVM ブロックデバイスをすべてスキャンします。このコマンドで、特定の物理ボリュームがスキャンされないように、**lvm.conf** ファイルでフィルターを定義することができます。

```
# pvscan
  PV  /dev/vdb1                  lvm2 [1.00 GiB]
  PV  /dev/vdb2                  lvm2 [1.00 GiB]
  PV  /dev/vdb3                  lvm2 [1.00 GiB]
```

関連情報

- **pvcreate (8)**、**pvdisplay (8)**、**pvs (8)**、**pvscan (8)**、および **lvm (8)** の man ページ

2.4. LVM 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場合、**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。

手順

1. 物理ボリュームを削除します。

```
# pvremove /dev/vdb3
Labels on physical volume "/dev/vdb3" successfully wiped.
```

2. 既存の物理ボリュームを表示し、必要なボリュームが削除されているかどうかを確認します。

```
# pvs
  PV         VG   Fmt    Attr    PSize      PFree
/dev/vdb1        lvm2           1020.00m   0
/dev/vdb2        lvm2           1020.00m   0
```

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。詳細は、「ボリュームグループからの物理ボリュームの削除」を参照してください。

関連情報

- **pvremove(8)** の man ページ

第3章 LVM ボリュームグループの管理

ボリュームグループ(VG)は、物理ボリューム(PV)の集合です。これにより、論理ボリューム(LV)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。

3.1. LVM ボリュームグループの作成

この手順では、物理ボリューム**/dev/vdb1**および**/dev/vdb2**を使用して、LVM ボリュームグループ(VG) **myvg**を作成する方法を説明します。

前提条件

- **lvm2** パッケージがインストールされている。
- 物理ボリュームが作成されます。物理ボリュームの作成方法は、「LVM 物理ボリュームの作成」を参照してください。

手順

1. ボリュームグループを作成します。

```
# vgcreate myvg /dev/vdb1 /dev/vdb2
 Volume group "myvg" successfully created.
```

これにより、**myvg** という名前の VG が作成されます。物理ボリュームの **/dev/vdb1** および **/dev/vdb2** は、ボリュームグループ **myvg** のベースストレージレベルです。

2. 要件に応じて、以下のコマンドのいずれかを使用して、作成したボリュームグループを表示します。

   a. **vgs** コマンド: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
  VG   #PV #LV #SN Attr   VSize   VFree
  myvg   4   1   0 wz--n-   3.98g 1008.00m
```

   b. **vgdisplay** コマンド: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) およびその他のオプションを表示します。以下の例は、ボリュームグループ **myvg** に関する **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 myvg _ --- Volume group --- VG Name _myvg
  System ID
  Format                lvm2
  Metadata Areas        4
```

```
  Metadata Sequence No  6
  VG Access             read/write
[..]
```

c. **vgscan** コマンド:ボリュームグループ用に、システムにあるサポートされるすべての LVM ブロックデバイスをスキャンします。

```
# vgscan
  Found volume group "myvg" using metadata type lvm2
```

3. オプション:オプション: 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。

```
# vgextend myvg /dev/vdb3
Physical volume "/dev/vdb3" successfully created.
Volume group "myvg" successfully extended
```

関連情報

- **pvcreate(8)**、**vgextend(8)**、**vgdisplay(8)**、**vgs(8)**、**vgscan(8)**、および **lvm(8)** のmanページ

3.2. LVM 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ボリューム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。

手順

- 非アクティブなボリュームグループ **databases** 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 **myvg** にマージして、詳細なランタイム情報を提供します。

```
# vgmerge -v myvg databases
```

関連情報

- **vgmerge(8)** の man ページ

3.3. 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。

手順

1. 物理ボリュームがまだ使用中の場合は、データを同じボリュームグループから別の物理ボリュームに移行します。

```
# pvmove /dev/vdb3
  /dev/vdb3: Moved: 2.0%
 ...
  /dev/vdb3: Moved: 79.2%
 ...
  /dev/vdb3: Moved: 100.0%
```

2. 既存のボリュームグループ内の他の物理ボリュームに空きエクステントが十分にない場合は、以下を行います。

   a. **/dev/vdb4** から、物理ボリュームを新規作成します。

   ```
   # pvcreate /dev/vdb4
     Physical volume "/dev/vdb4" successfully created
   ```

   b. 新規作成した物理ボリュームを **myvg** ボリュームグループに追加します。

   ```
   # vgextend myvg /dev/vdb4
     Volume group "myvg" successfully extended
   ```

   c. データを **/dev/vdb3** から **/dev/vdb4** に移動します。

   ```
   # pvmove /dev/vdb3 /dev/vdb4
     /dev/vdb3: Moved: 33.33%
     /dev/vdb3: Moved: 100.00%
   ```

3. ボリュームグループから物理ボリューム **/dev/vdb3** を削除します。

```
# vgreduce myvg /dev/vdb3
Removed "/dev/vdb3" from volume group "myvg"
```

検証

- **/dev/vdb3** 物理ボリュームが **myvg** ボリュームグループから削除されているかどうかを確認します。

```
# pvs
  PV         VG    Fmt   Attr   PSize       PFree      Used
  /dev/vdb1 myvg  lvm2   a--    1020.00m    0         1020.00m
  /dev/vdb2 myvg  lvm2   a--    1020.00m    0         1020.00m
  /dev/vdb3       lvm2   a--    1020.00m   1008.00m    12.00m
```

関連情報

- **vgreduce(8)**、**pvmove(8)**、および **pvs(8)** の manページ

3.4. LVM ボリュームグループの分割

この手順では、既存のボリュームグループを分割する方法を説明します。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。

初期設定では、ボリュームグループ **myvg** は/dev/vdb1、/dev/vdb2、および /dev/vdb3 で構成されます。この手順を完了すると、ボリュームグループ **myvg** は /dev/vdb1 および /dev/vdb2 で構成され、2 番目のボリュームグループ **yourvg** は /dev/vdb3 で構成されます。

前提条件

- ボリュームグループに十分な空き領域がある。**vgscan** 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。
- 既存の物理ボリュームの空き容量に応じて、**pvmove** コマンドを使用して、使用されている物理エクステントをすべて他の物理ボリュームに移動します。詳細は、ボリュームグループからの物理ボリュームの削除 を参照してください。

手順

1. 既存のボリュームグループ **myvg** を新しいボリュームグループ **yourvg** に分割します。

```
# vgsplit myvg yourvg /dev/vdb3
  Volume group "yourvg" successfully split from "myvg"
```

注記

既存のボリュームグループを使用して論理ボリュームを作成した場合は、次のコマンドを実行して論理ボリュームを非アクティブにします。

```
# lvchange -a n /dev/myvg/mylv
```

論理ボリュームを作成する方法は、LVM 論理ボリュームの管理」を参照してください。

2. 2 つのボリュームグループの属性を表示します。

```
# 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
```

検証

- 新規作成したボリュームグループ **yourvg** が、/dev/vdb3 物理ボリュームで構成されているかどうかを確認します。

```
# pvs
  PV         VG     Fmt  Attr PSize    PFree    Used
  /dev/vdb1 myvg   lvm2 a--  1020.00m    0     1020.00m
  /dev/vdb2 myvg   lvm2 a--  1020.00m    0     1020.00m
  /dev/vdb3 yourvg lvm2 a--  1020.00m 1008.00m  12.00m
```

関連情報

- **vgsplit(8)**、**vgs(8)**、および **pvs(8)** の manページ

3.5. LVM ボリュームグループの名前変更

この手順では、既存のボリュームグループ **myvg** の名前を **myvg1** に変更します。

手順

1. ボリュームグループを非アクティブにします。クラスター化ボリュームグループの場合は、アクティブになっているすべてのノードでボリュームグループを非アクティブにします。そのために、各ノードで以下のコマンドを使用します。

```
# vgchange --activate n myvg
```

2. 既存のボリュームグループの名前を変更します。

```
# vgrename myvg myvg1
Volume group "myvg" successfully renamed to "myvg1"
```

また、デバイスへの完全パスを指定して、ボリュームグループの名前を変更することもできます。

```
# vgrename /dev/myvg /dev/myvg1
```

関連情報

- **vgrename(8)** の man ページ

第4章 LVM 論理ボリュームの管理

論理ボリュームは、ファイルシステム、データベース、またはアプリケーションが使用できる仮想のブロックストレージデバイスです。LVM 論理ボリュームを作成する場合は、物理ボリューム(PV)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。

4.1. 論理ボリュームの概要

管理者は、標準のディスクパーティションとは異なり、データを破棄せずに論理ボリュームを拡大または縮小できます。ボリュームグループの物理ボリュームが別のドライブまたは RAID アレイにある場合は、ストレージデバイスに論理ボリュームを分散することもできます。

論理ボリュームを、ボリュームに必要なデータよりも小さい容量に縮小すると、データが失われる可能性があります。さらに、ファイルシステムの中には縮小できないものもあります。柔軟性を最大限にするために、現在のニーズに合わせて論理ボリュームを作成し、過剰なストレージ容量を未割り当ての状態にします。必要に応じて、未割り当ての領域を使用するように、論理ボリュームを安全に拡張できます。

重要

AMD システム、Intel システム、ARM システム、および IBM Power Systems サーバーで、ブートローダーは LVM ボリュームを読み取ることができません。このため、**/boot** パーティションは、LVM ではなく標準のパーティションで作成してください。IBM Z の場合は、**zipl** ブートローダーによりリニアマッピングを使用した LVM 論理ボリューム上の **/boot** に対応しています。デフォルトのインストールプロセスでは、/ パーティションと swap パーティションは常に LVM ボリューム内に、**/boot** パーティションは別途、物理ボリューム上に作成されます。

以下は、論理ボリュームの種類になります。

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。

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、基に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。順次の読み取りおよび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。

RAID 論理ボリューム

LVM は、RAID レベル 0、1、4、5、6、10 に対応します。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。

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)

シンプロビジョニングされた論理ボリュームを使用すると、利用可能な物理ストレージよりも大きな論理ボリュームを作成できます。シンプロビジョニングされたボリュームセットを作成すると、

システムは要求されるストレージの全量を割り当てる代わりに、実際に使用する容量を割り当てることができます。

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作成元のデバイスに変更が加えられると、データが変更する前に、これから変更す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。

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

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ると、同じデータボリュームにより多くの仮想デバイスを格納できます。シンプロビジョニングされたスナップショットは、特定のタイミングでキャプチャーする際にすべてのデータをコピーしていないため、便利です。

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、大規模で低速なブロックデバイスのライトバックまたはライトスルーのキャッシュとして使用することに対応します。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。

4.2. LVM 論理ボリュームの作成

この手順では、**/dev/vdb1**、**/dev/vdb2**、および **/dev/vdb3** 物理ボリュームを使用して作成された **myvg** ボリュームグループから **mylv** LVM 論理ボリューム (LV)を作成する方法を説明します。

前提条件

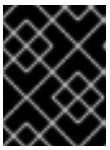
- **lvm2** パッケージがインストールされている。
- ボリュームグループが作成されます。詳細は、「LVM ボリュームグループの作成」を参照してください。

手順

1. 論理ボリュームを作成します。

```
# lvcreate -n mylv -L 500M myvg
```

-n オプションを使用して LV 名を **mylv** に設定し、**-L** オプションを使用して、Mb 単位で LV のサイズを設定しますが、他の単位を使用することもできます。デフォルトでは、論理ボリュームのタイプはリニアですが、**--type** オプションを使用して必要なタイプを指定できます。



重要

ボリュームグループに要求されるサイズとタイプの空き物理エクステントが十分にない場合、このコマンドは失敗します。

2. 要件に応じて、以下のコマンドのいずれかを使用して、作成した論理ボリュームを表示します。
   a. **lvs** コマンド:論理ボリューム情報を設定可能な形式で提供して、1 つの論理ボリュームにつき 1 行ずつ表示します。

```
# lvs
  LV   VG   Attr       LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
 mylv myvg -wi-ao----   500.00m
```

b. **lvdisplay** コマンド: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。

```
# lvdisplay -v /dev/myvg/mylv
  --- Logical volume ---
  LV Path                /dev/myvg/mylv
  LV Name                mylv
  VG Name                myvg
  LV UUID                YTnAk6-kMlT-c4pG-HBFZ-Bx7t-ePMk-7YjhaM
  LV Write Access        read/write
[..]
```

c. **lvscan** コマンド:システム内のすべての論理ボリュームをスキャンし、それらを一覧表示します。

```
# lvscan
  ACTIVE                 '/dev/myvg/mylv' [500.00 MiB] inherit
```

3. 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 **xfs** ファイルシステムが作成されます。

```
# mkfs.xfs /dev/myvg/mylv
meta-data=/dev/myvg/mylv       isize=512    agcount=4, agsize=32000 blks
         =                     sectsz=512   attr=2, projid32bit=1
         =                     crc=1        finobt=1, sparse=1, rmapbt=0
         =                     reflink=1
data     =                       bsize=4096   blocks=128000, imaxpct=25
         =                     sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=1368, version=2
         =                     sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
Discarding blocks...Done.
```

4. 論理ボリュームをマウントして、ファイルシステムのディスクの領域使用率を報告します。

```
# mount /dev/myvg/mylv /mnt

# df -h
Filesystem              1K-blocks  Used   Available Use% Mounted on

/dev/mapper/myvg-mylv   506528    29388  477140     6%   /mnt
```

関連情報

- **lvcreate(8)**、**lvdisplay(8)**、**lvs(8)**、**lvscan(8)**、**lvm(8)**、および **mkfs.xfs(8)** の man ページ

4.3. LVM 論理ボリュームの名前の変更

この手順では、既存の論理ボリューム **mylv** の名前を **mylv1** に変更する方法を説明します。

手順

1. 論理ボリュームが現在マウントされている場合は、ボリュームをアンマウントします。

```
# umount /mnt
```

   /mnt は、マウントポイントに置き換えます。

2. クラスター環境に論理ボリュームが存在する場合は、アクティブになっているすべてのノードで、論理ボリュームを非アクティブにします。アクティブになっている各ノードで、次のコマンドを実行します。

```
# lvchange --activate n myvg/mylv
```

3. 既存の論理ボリュームの名前を変更します。

```
# lvrename myvg mylv mylv1
Logical volume "mylv" successfully renamed to "mylv1"
```

   また、デバイスへの完全パスを指定して、論理ボリュームの名前を変更することもできます。

```
# lvrename /dev/myvg/mylv /dev/myvg/mylv1
```

関連情報

- **lvrename(8)** の man ページ

4.4. 論理ボリュームからのディスクの削除

この手順では、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説明します。

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。

手順

1. LV を使用する際に、物理ボリュームの使用済み容量と空き容量を表示します。

```
# pvs -o+pv_used
  PV         VG    Fmt    Attr   PSize      PFree     Used
 /dev/vdb1 myvg  lvm2   a--    1020.00m    0        1020.00m
 /dev/vdb2 myvg  lvm2   a--    1020.00m    0        1020.00m
 /dev/vdb3 myvg  lvm2   a--    1020.00m   1008.00m   12.00m
```

2. データを他の物理ボリュームに移動します。

   a. 既存のボリュームグループ内の他の物理ボリュームに空きエクステントが十分にある場合は、以下のコマンドを使用してデータを移動します。

```
# pvmove /dev/vdb3
```

```
  /dev/vdb3: Moved: 2.0%
 ...
  /dev/vdb3: Moved: 79.2%
 ...
  /dev/vdb3: Moved: 100.0%
```

b. 既存のボリュームグループ内の他の物理ボリュームに空きエクステントが十分にない場合は、以下のコマンドを使用して新しい物理ボリュームを追加し、新たに作成した物理ボリュームを使用してボリュームグループを拡張し、この物理ボリュームにデータを移動します。

```
# pvcreate /dev/vdb4
  Physical volume "/dev/vdb4" successfully created

# vgextend myvg /dev/vdb4
  Volume group "myvg" successfully extended

# pvmove /dev/vdb3 /dev/vdb4
  /dev/vdb3: Moved: 33.33%
  /dev/vdb3: Moved: 100.00%
```

3. 物理ボリュームを削除します。

```
# vgreduce myvg /dev/vdb3
Removed "/dev/vdb3" from volume group "myvg"
```

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。

```
# vgreduce --removemissing myvg
```

関連情報

- **pvmove(8)**、**vgextend(8)**、**vereduce(8)**、および **pvs(8)** の manページ

4.5. LVM 論理ボリュームの削除

この手順では、既存の論理ボリューム **/dev/myvg/mylv1** をボリュームグループ **myvg** から削除する方法を説明します。

手順

1. 論理ボリュームが現在マウントされている場合は、ボリュームをアンマウントします。

```
# umount /mnt
```

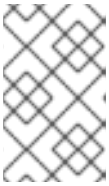
2. クラスター環境に論理ボリュームが存在する場合は、アクティブになっているすべてのノードで、論理ボリュームを非アクティブにします。アクティブになっている各ノードで、次のコマンドを実行します。

```
# lvchange --activate n vg-name/lv-name
```

3. **lvremove** ユーティリティーを使用して、論理ボリュームを削除します。

```
# lvremove /dev/myvg/mylv1

Do you really want to remove active logical volume "mylv1"? [y/n]: y
Logical volume "mylv1" successfully removed
```



注記

この場合は、論理ボリュームが非アクティブになっていません。削除する前に論理ボリュームを明示的に非アクティブにした場合は、アクティブな論理ボリュームを削除するかどうかを確認するプロンプトが表示されません。

関連情報

- **lvremove(8)** の man ページ

4.6. LVM ボリュームグループの削除

この手順では、既存のボリュームグループを削除する方法を説明します。

前提条件

- ボリュームグループには論理ボリュームがありません。ボリュームグループから論理ボリュームを削除するには、LVM 論理ボリュームの削除を参照してください。

手順

1. クラスター環境にボリュームグループが存在する場合は、その他のすべてのノードで、ボリュームグループの**lockspace**を停止します。削除するノード以外の全ノードで、次のコマンドを実行します。

```
# vgchange --lockstop vg-name
```

ロックが停止するのを待ちます。

2. ボリュームグループを削除します。

```
# vgremove vg-name
  Volume group "vg-name" successfully removed
```

関連情報

- **vgremove(8)** の man ページ

第5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。

5.1. 論理ボリュームとファイルシステムの拡張

この手順では、論理ボリュームを拡張し、同じ論理ボリューム上のファイルシステムを拡張する方法を説明します。

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。

前提条件

1. ファイルシステムを持つ既存の論理ボリューム(LV)がある。**df -Th** コマンドを使用して、ファイルシステムのタイプを確認します。
   LV およびファイルシステムの作成に関する詳細は、「LVM 論理ボリュームの作成」を参照してください。

2. LV およびファイルシステムを拡張するのに十分な領域がボリュームグループにある。**vgs -o name,vgfree** コマンドを使用して、利用可能な領域を確認します。

手順

1. オプション:オプション: ボリュームグループに LV を拡張するのに十分な領域がない場合は、以下のコマンドを使用してボリュームグループに新しい物理ボリュームを追加します。

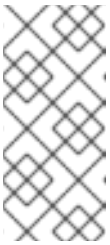
   ```
   # vgextend myvg /dev/vdb3
   Physical volume "/dev/vdb3" successfully created.
   Volume group "myvg" successfully extended
   ```

   詳細は、「LVM ボリュームグループの作成」を参照してください。

2. ボリュームグループが十分に大きくなったので、要件に応じて以下のいずれかの手順を実行します。

   a. 指定されたサイズで LV を拡張するには、次のコマンドを使用します。

   ```
   # lvextend -L 3G /dev/myvg/mylv
   Size of logical volume myvg/mylv changed from 2.00 GiB (512 extents) to 3.00 GiB (768 extents).
   Logical volume myvg/mylv successfully resized.
   ```



注記

lvextend コマンドで **-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ます。

```
# lvextend -r -L 3G /dev/myvg/mylv
```

警告

lvresize コマンドを使用して、同じ引数で論理ボリュームを拡張することもできますが、このコマンドでは、誤って縮小されない保証はありません。

b. **mylv** 論理ボリュームを拡張して、**myvg** ボリュームグループの未割り当て領域をすべて埋めるには、次のコマンドを使用します。

```
# lvextend -l +100%FREE /dev/myvg/mylv
 Size of logical volume myvg/mylv changed from 10.00 GiB (2560 extents) to 6.35 TiB (1665465 extents).
 Logical volume myvg/mylv successfully resized.
```

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。

3. **lvextend**コマンドで **r** オプションを使用して1つのコマンドでLVを拡張してファイルシステムのサイズを変更しない場合は、以下のコマンドを使用して論理ボリューム上のファイルシステムのサイズを変更します。

```
xfs_growfs /mnt/mnt1/
meta-data=/dev/mapper/myvg-mylv  isize=512    agcount=4, agsize=65536 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1
data     =                       bsize=4096   blocks=262144, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
data blocks changed from 262144 to 524288
```

注記

xfs_growfs は、**-D** オプションを指定しないと、基となるデバイスがサポートする最大サイズまでファイルシステムを拡張します。詳細は「Increasing the size of an XFS file system」を参照してください。

ext4 ファイルシステムのサイズを変更する場合は、「Resizing an ext4 file system 」を参照してください。

検証

- 以下のコマンドを使用して、ファイルシステムが拡大しているかどうかを確認します。

```
# df -Th
Filesystem            Type      Size  Used Avail Use% Mounted on
devtmpfs              devtmpfs  1.9G     0  1.9G   0% /dev
tmpfs                 tmpfs     1.9G     0  1.9G   0% /dev/shm
tmpfs                 tmpfs     1.9G  8.6M  1.9G   1% /run
tmpfs                 tmpfs     1.9G     0  1.9G   0% /sys/fs/cgroup
/dev/mapper/rhel-root xfs        45G  3.7G   42G   9% /
/dev/vda1             xfs      1014M  369M  646M  37% /boot
tmpfs                 tmpfs     374M     0  374M   0% /run/user/0
/dev/mapper/myvg-mylv xfs       2.0G   47M  2.0G   3% /mnt/mnt1
```

関連情報

- **vgextend(8)**、**lvextend(8)**、および **xfs_growfs(8)** の manページ

5.2. 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

GFS2 および XFS のファイルシステムでは縮小に対応していないため、GFS2 または XFS の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領域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確認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。

手順

- **myvg** ボリュームグループの **mylv** 論理ボリュームを 64 メガバイトに縮小するには、次のコマンドを使用します。

```
# lvreduce --resizefs -L 64M myvg/mylv
fsck from util-linux 2.37.2
/dev/mapper/myvg-mylv: clean, 11/25688 files, 4800/102400 blocks
resize2fs 1.46.2 (28-Feb-2021)
Resizing the filesystem on /dev/mapper/myvg-mylv to 65536 (1k) blocks.
The filesystem on /dev/mapper/myvg-mylv is now 65536 (1k) blocks long.

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

この例では、**mylv** にはファイルシステムが含まれ、このコマンドによって論理ボリュームとともにサイズが変更されます。

- 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。論理ボリュームを絶対サイズ 64 メガバイトに縮小するには、次のコマンドを使用します。

```
# lvreduce --resizefs -L -64M myvg/mylv
```

関連情報

- **lvreduce(8)** の man ページ

第6章 論理ボリュームのスナップショット

LVM スナップショット機能を使用すると、サービスを中断することなく、ある時点でのボリュームの仮想イメージ (**/dev/sda** など) を作成できます。

6.1. スナップショットボリュームの概要

スナップショットを作成した後で元のボリューム (スナップショットの元になるボリューム) を変更すると、スナップショット機能は、ボリュームの状態を再構築できるように、変更前の変更されたデータ領域のコピーを作成します。スナップショットを作成しても、作成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。

スナップショットは、スナップショットの作成後に変更し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、作成元の 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。バックアップ手順に代わるものではありません。スナップショットコピーは仮想コピーであり、実際のメディアバックアップではありません。

作成元のボリュームへの変更を保管するために確保する領域は、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作成元を完全に上書きする場合に、その変更の保管に必要なスナップショットのサイズは、作成元のボリュームと同等か、それ以上になります。スナップショットのサイズは定期的に監視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領域は、**/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領域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、作成元のボリュームの変更を追跡できなくなるため、そのスナップショットは無効になります。ただし、スナップショットが無効になるのを防ぐために、使用量が **snapshot_autoextend_threshold** 値を超えるたびにスナップショットを自動的に拡張するように LVM を設定できます。スナップショットは完全にサイズ変更可能で、次の操作を実行できます。

- ストレージ容量に余裕がある場合は、スナップショットボリュームのサイズを大きくして、削除されないようにすることができます。
- スナップショットの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。

スナップショットボリュームには、次の利点があります。

- 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、ファイルシステムの整合性をチェックすれば、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であるため、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。

6.2. 元のボリュームのスナップショット作成

-s または **--size** 引数の後に必要なサイズを指定して **lvcreate** コマンドを使用し、元のボリューム (作成元) のスナップショットを作成します。ボリュームのスナップショットは書き込み可能です。デフォルトでは、シンプロビジョニングされたスナップショットと比較すると、通常のアクティベーションコマンドを実行中に、作成元のボリュームを使用して、スナップショットのボリュームを有効にします。LVM は、元のボリュームのサイズとボリュームに必要なメタデータサイズの合計よりも大きいスナップショットボリュームの作成をサポートしていません。これより大きいスナップショットボリュームを指定すると、LVM は作成元のサイズに必要なスナップショットボリュームを作成します。

注記

クラスター内のノードは LVM スナップショットをサポートしていません。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。

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。

前提条件

- ボリュームグループ **vg001** を作成している。詳細は、「LVM ボリュームグループの作成」を参照してください。

手順

1. ボリュームグループ **vg001** から、論理ボリューム **origin** を作成します。

```
# lvcreate -L 1G -n origin vg001
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 **/dev/vg001/origin** を作成します。

```
# lvcreate --size 100M --name snap --snapshot /dev/vg001/origin
  Logical volume "snap" created.
```

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。

3. 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。

```
# lvs -a -o +devices
  LV      VG    Attr       LSize  Pool Origin Data% Meta% Move Log Cpy%Sync Convert
Devices
 origin vg001  owi-a-s---  1.00g                                        /dev/sde1(0)
  snap vg001  swi-a-s--- 100.00m     origin 0.00                            /dev/sde1(256)
```

lvdisplay /dev/vg001/origin コマンドを使用して、論理ボリューム **/dev/vg001/origin** のステータスと、すべてのスナップショット論理ボリュームおよびそれらのステータス (アクティブまたは非アクティブなど) を表示することもできます。

警告

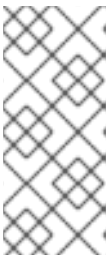
複製元ボリュームが変更されると、スナップショットのサイズが拡大されるため、**lvs** コマンドを使用して、スナップショットボリュームのパーセンテージを定期的に監視して、満杯にならないように確認することが重要です。スナップショットは、100% になると完全に消失します。これは、作成元ボリュームの変更されていない部分に書き込みが行われるため、スナップショットが必ず破損するためです。

4. ただし、使用量が 100% の場合にスナップショットが無効になるのを回避するために、使用量が **snapshot_autoextend_threshold** 値を超えるたびにスナップショットを自動的に拡張するように LVM を設定できます。**/etc/lvm.conf** ファイルから **snapshot_autoextend_threshold** および **snapshot_autoextend_percent** オプションの既存の値を表示し、要件に従ってそれらを編集します。
次の例では、**snapshot_autoextend_threshold** オプションを 100 未満の値に設定し、**snapshot_autoextend_percent** オプションをスナップショットボリュームを拡張するための要件に応じた値に設定します。

```
# vi /etc/lvm.conf
snapshot_autoextend_threshold = 70
snapshot_autoextend_percent = 20
```

次のコマンドを実行して、このスナップショットを手動で拡張することもできます。

```
# lvextend -L+100M /dev/vg001/snap
```



注記

この機能には、ボリュームグループに未割り当てのスペースが必要です。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。

関連情報

- **lvcreate (8)**、**lvextend (8)**、および **lvs (8)** の man ページ
- **/etc/lvm/lvm.conf** file

6.3. スナップショットと元のボリュームのマージ

--merge オプションを指定して **lvconvert** コマンドを使用し、スナップショットを元のボリューム (作成元) にマージします。データやファイルを失った場合や、システムを以前の状態に復元する必要がある場合に、システムのロールバックを実行できます。スナップショットボリュームをマージすると、作成された論理ボリュームには、元のボリュームの名前、マイナー番号、および UUID が含まれます。マージの進行中、作成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。

作成元のボリュームとスナップショットボリュームの両方が起動されておらず、アクティブでない場合、マージはすぐに開始されます。それ以外の場合は、作成元またはスナップショットのいずれかがアクティブ化され、両方が閉じられた後にマージが開始されます。スナップショットを閉じることができない作成元 (**root** ファイルシステムなど) にマージできるのは、作成元のボリュームがアクティブ化された後です。

手順

1. スナップショットボリュームをマージします。以下のコマンドは、スナップショットボリューム **vg001/snap** をその **作成元** にマージします。

```
# lvconvert --merge vg001/snap
Merging of volume vg001/snap started.
  vg001/origin: Merged: 100.00%
```

2. 元のボリュームを表示します。

```
# lvs -a -o +devices
  LV      VG    Attr       LSize  Pool Origin Data% Meta% Move Log Cpy%Sync Convert
Devices
  origin vg001  owi-a-s---  1.00g                                             /dev/sde1(0)
```

関連情報

- **lvconvert(8)** の man ページ

第7章 シンプロビジョニングされたボリューム (シンボリューム) の作成および管理

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と論理ボリュームをサポートします。

論理ボリュームとスナップショットボリュームは、シンプロビジョニングできます。

- シンプロビジョニングされた論理ボリュームを使用すると、利用可能な物理ストレージよりも大きな論理ボリュームを作成できます。
- 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を使用すると、同じデータボリュームにより多くの仮想デバイスを格納できます。

7.1. シンプロビジョニングの概要

最新のストレージスタックの多くは、シックプロビジョニングとシンプロビジョニングのどちらかを選択できるようになりました。

- シックプロビジョニングは、ブロックストレージの従来の動作を実行でき、実際の使用状況に関係なくブロックが割り当てられます。
- シンプロビジョニングは、ブロックストレージのプールよりも大きいサイズ (データを保存する物理デバイスよりもサイズが大きくなる可能性のあるブロックストレージ) をプロビジョニングできるので、過剰にプロビジョニングされる可能性があります。個々のブロックは実際に使用されるまで割り当てられないため、オーバープロビジョニングが発生する可能性があります。同じプールを共有する複数のシンプロビジョニングされたデバイスがある場合に、これらのデバイスは過剰にプロビジョニングされる可能性があります。

シンプロビジョニングを使用すると、物理ストレージをオーバーコミットでき、代わりにシンプールと呼ばれる空き領域のプールを管理できます。アプリケーションで必要な場合は、このシンプールを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に、動的に拡張できます。

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 要求された場合には、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)。

注記

シンプロビジョニングを使用する場合は、ストレージプールを監視して、使用可能な物理スペースが不足したときに容量を追加することが重要です。

以下は、シンプロビジョニングされたデバイスを使用する利点です。

- 使用可能な物理ストレージよりも大きい論理ボリュームを作成できます。
- 同じデータボリュームに保存する仮想デバイスを増やすことができます。
- データ要件をサポートするために論理的かつ自動的に拡張できるファイルシステムを作成できます。未使用のブロックはプールに戻され、プール内の任意のファイルシステムで使用できます。

シンプロビジョニングされたデバイスを使用する場合に発生する可能性のある欠点は次のとおりです。

- シンプロビジョニングされたボリュームには、使用可能な物理ストレージが不足するという固有のリスクがあります。基盤となるストレージを過剰にプロビジョニングした場合に、使用可能な物理ストレージが不足しているために停止する可能性があります。たとえば、バッキング用に 1T の物理ストレージのみを使用して、10T のシンプロビジョニングストレージを作成する場合に、この 1T が使い果たされると、ボリュームは使用不可または書き込み不能になります。

- ボリュームが、シンプロビジョニングデバイスの後の階層に破棄するように送信していない場合には、使用量の計算が正確でなくなります。たとえば、**-o discard mount** オプションを指定せずにファイルシステムを配置し、シンプロビジョニングされたデバイス上で **fstrim** を定期的に実行しないと、以前に使用されたストレージの割り当てが解除されることはありません。このような場合に、実際に使用していなくても、時間の経過とともにプロビジョニングされた量をすべて使用することになります。

- 使用可能な物理スペースが不足しないように、論理的および物理的な使用状況を監視する必要があります。

- スナップショットのあるファイルシステムでは、コピーオンライト (CoW) 操作が遅くなる可能性があります。

- データブロックは複数のファイルシステム間で混在する可能性があり、エンドユーザーにそのように表示されない場合でも、基盤となるストレージのランダムアクセス制限につながります。

7.2. シンプロビジョニングされた論理ボリュームの作成

シンプロビジョニングされた論理ボリュームを使用すると、利用可能な物理ストレージよりも大きな論理ボリュームを作成できます。シンプロビジョニングされたボリュームセットを作成すると、システムは要求されるストレージの全量を割り当てる代わりに、実際に使用する容量を割り当てることができます。

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。この手順では、シンプロビジョニングされた論理ボリュームを作成および拡張する方法について説明します。

前提条件

- ボリュームグループを作成している。詳細は、「LVM ボリュームグループの作成」を参照してください。

手順

1. シンプールを作成します。

```
# lvcreate -L 100M -T vg001/mythinpool
  Thin pool volume with chunk size 64.00 KiB can address at most 15.81 TiB of data.
  Logical volume "mythinpool" created.
```

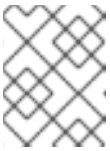
物理領域のプールを作成しているため、プールのサイズを指定する必要があります。**lvcreate** コマンドの **-T** オプションでは引数を使用できません。コマンドで指定する他のオプションをもとに、作成するデバイスのタイプを決定します。次の例に示すように、追加のパラメーターを使用してシンプールを作成することもできます。

- 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** パラメーターでは、作成するシン

プール論理ボリュームの名前を指定する必要があります。次の例では、**-thinpool** パラメーターを使用して、サイズが 100M のボリュームグループ vg001 に mythinpool という名前のシンプールを作成します。

```
# lvcreate -L 100M --thinpool mythinpool vg001
  Thin pool volume with chunk size 64.00 KiB can address at most 15.81 TiB of data.
  Logical volume "mythinpool" created.
```

- プールの作成ではストライピングがサポートされているため、**-i** および **-I** オプションを使用してストライプを作成できます。次のコマンドは、2 つの 64kB ストライプと 256kB のチャンクサイズを持つボリュームグループ vg001 に thinpool という名前で、サイズが 100M のシンプールを作成します。また、vg001/thinvolume という名前でサイズが 1T のシンボリュームを作成します。



注記

ボリュームグループに十分な空き領域がある 2 つの物理ボリュームがあることを確認してください。そうでない場合にはシンプールを作成できません。

```
# lvcreate -i 2 -I 64 -c 256 -L 100M -T vg001/thinpool -V 1T --name thinvolume
```

2. シンボリュームを作成します。

```
# lvcreate -V 1G -T vg001/mythinpool -n thinvolume
  WARNING: Sum of all thin volume sizes (1.00 GiB) exceeds the size of thin pool
vg001/mythinpool (100.00 MiB).
  WARNING: You have not turned on protection against thin pools running out of space.
  WARNING: Set activation/thin_pool_autoextend_threshold below 100 to trigger automatic
extension of thin pools before they get full.
  Logical volume "thinvolume" created.
```

このような場合には、ボリュームを含むプールのサイズよりも大きい、ボリュームの仮想サイズを指定しています。次の例に示すように、追加のパラメーターを使用してシンボリュームを作成することもできます。

- シンボリュームとシンプールの両方を作成するには、**lvcreate** コマンドの **-T** オプションを使用して、サイズと仮想サイズの両方の引数を指定します。

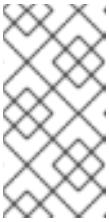
```
# lvcreate -L 100M -T vg001/mythinpool -V 1G -n thinvolume
  Thin pool volume with chunk size 64.00 KiB can address at most 15.81 TiB of data.
  WARNING: Sum of all thin volume sizes (1.00 GiB) exceeds the size of thin pool
vg001/mythinpool (100.00 MiB).
  WARNING: You have not turned on protection against thin pools running out of space.
  WARNING: Set activation/thin_pool_autoextend_threshold below 100 to trigger
automatic extension of thin pools before they get full.
  Logical volume "thinvolume" created.
```

- 残りの空き領域を使用してシンボリュームとシンプールを作成するには、**100%FREE** オプションを使用します。

```
# lvcreate -V 1G -l 100%FREE -T vg001/mythinpool -n thinvolume
Thin pool volume with chunk size 64.00 KiB can address at most <15.88 TiB of data.
  Logical volume "thinvolume" created.
```

- 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には、**lvconvert** コマンドの **--thinpool** パラメーターを使用します。また、**-poolmetadata** パラメーターを **--thinpool** パラメーターと組み合わせて使用して、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。
  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.
```



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。**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。

- デフォルトでは、**lvcreate** コマンドは、次の式に従って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。

```
Pool_LV_size / Pool_LV_chunk_size * 64
```

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対応している値は 2MiB ~ 16GiB です。

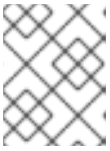
次の例は、シンプールのメタデータボリュームのデフォルト値を増やす方法を示しています。

```
# lvcreate -V 1G -l 100%FREE -T vg001/mythinpool --poolmetadatasize 16M -n
thinvolume
Thin pool volume with chunk size 64.00 KiB can address at most 15.81 TiB of data.
  Logical volume "thinvolume" created.
```

3. 作成されたシンプールとシンボリュームを表示します。

```
# lvs -a -o +devices
  LV              VG    Attr       LSize   Pool       Origin Data%  Meta%  Move Log Cpy%Sync
Convert Devices
  [lvol0_pmspare]    vg001 ewi-------   4.00m                                                  /dev/sda(0)
  mythinpool         vg001 twi-aotz-- 100.00m                 0.00   10.94
mythinpool_tdata(0)
  [mythinpool_tdata] vg001 Twi-ao---- 100.00m
/dev/sda(1)
  [mythinpool_tmeta] vg001 ewi-ao----   4.00m
/dev/sda(26)
  thinvolume         vg001 Vwi-a-tz--   1.00g mythinpool        0.00
```

4. オプション:**lvextend** コマンドを使用して、シンプールのサイズを拡張します。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。



注記

シンプールとシンボリュームの作成中に **-l100%FREE** 引数を使用すると、このコマンドは失敗します。

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、さらに 100M 分を拡張します。

```
# lvextend -L+100M vg001/mythinpool
  Size of logical volume vg001/mythinpool_tdata changed from 100.00 MiB (25 extents) to
200.00 MiB (50 extents).
  WARNING: Sum of all thin volume sizes (1.00 GiB) exceeds the size of thin pool
vg001/mythinpool (200.00 MiB).
  WARNING: You have not turned on protection against thin pools running out of space.
  WARNING: Set activation/thin_pool_autoextend_threshold below 100 to trigger automatic
extension of thin pools before they get full.

  Logical volume vg001/mythinpool successfully resized
```

```
# lvs -a -o +devices
  LV              VG    Attr       LSize   Pool       Origin Data%  Meta%  Move Log Cpy%Sync
Convert Devices
  [lvol0_pmspare]    vg001 ewi-------   4.00m                                                  /dev/sda(0)
  mythinpool         vg001 twi-aotz-- 200.00m                 0.00   10.94
mythinpool_tdata(0)
  [mythinpool_tdata] vg001 Twi-ao---- 200.00m
/dev/sda(1)
  [mythinpool_tdata] vg001 Twi-ao---- 200.00m
/dev/sda(27)
  [mythinpool_tmeta] vg001 ewi-ao----   4.00m
/dev/sda(26)
  thinvolume         vg001 Vwi-a-tz--   1.00g mythinpool        0.00
```

5. オプション:シンプールとシンボリュームの名前を変更するには、次のコマンドを使用します。

```
# lvrename vg001/mythinpool vg001/mythinpool1
  Renamed "mythinpool" to "mythinpool1" in volume group "vg001"

# lvrename vg001/thinvolume vg001/thinvolume1
  Renamed "thinvolume" to "thinvolume1" in volume group "vg001"
```

名前を変更した後に、シンプールとシンボリュームを表示します。

```
# lvs
  LV          VG       Attr     LSize   Pool       Origin Data%  Move Log Copy%  Convert
mythinpool1 vg001    twi-a-tz 100.00m                    0.00
thinvolume1 vg001    Vwi-a-tz   1.00g mythinpool1        0.00
```

6. オプション:シンプールを削除するには、次のコマンドを使用します。

```
# lvremove -f vg001/mythinpool1
  Logical volume "thinvolume1" successfully removed.
  Logical volume "mythinpool1" successfully removed.
```

関連情報

- **lvcreate (8)**、**lvrename (8)**、**lvs (8)**、および **lvconvert (8)** の man ページ

7.3. シンプロビジョニングの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をサポートします。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 (LV) を作成することもできます。シンプロビジョニングの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。

注記

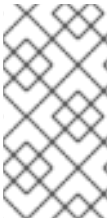
すべてのシンボリュームや、LVM スナップショットボリュームと同様に、シンプロビジョニングのスナップショットボリュームは、クラスターのノード間では対応していません。スナップショットボリュームは、1 つのクラスターノードで排他的にアクティブにする必要があります。

従来のスナップショットでは、作成されたスナップショットごとに新しい領域を割り当てる必要があります。この領域では、作成元に変更が加えられてもデータが保持されます。ただし、シンプロビジョニングスナップショットは、作成元と同じ領域を共有します。シン LV のスナップショットは、シン LV とそのスナップショットのいずれかに共通のデータブロックが共有されるので効率的です。シン LV のスナップショットを作成することも、他のシンスナップショットから作成することもできます。再帰スナップショットに共通のブロックもシンプールで共有されます。

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利点は以下のとおりです。

- オリジンのスナップショットの数を増やしても、パフォーマンスへの影響はほとんどありません。
- シンスナップショットボリュームは、新しいデータのみが書き込まれ、各スナップショットにコピーされないため、ディスク使用量を減らすことができます。
- 従来のスナップショットの要件でしたが、シンスナップショットボリュームを作成元と同時にアクティブにする必要はありません。
- スナップショットからオリジンを復元する場合、シンスナップショットをマージする必要はありません。オリジンを削除して、代わりにスナップショットを使用できます。従来のスナップショットには、コピーバックする必要がある変更を保存する別のボリュームがあります。つまり、元のスナップショットにマージしてリセットする必要があります。
- 従来のスナップショットと比較して、許可されるスナップショットの上限数がはるかに増えています。

シンプロビジョニングの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、従来の LVM スナップショットボリューム機能の方がニーズに適している場合もあります。すべてのタイプのボリュームで従来のスナップショットを使用できます。ただし、シンスナップショットを使用するには、シンプロビジョニングを使用する必要があります。



注記

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します。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するようにしてください。

デフォルトで、シン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。

7.4. 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作成

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ると、同じデータボリュームにより多くの仮想デバイスを格納できます。

重要

シンプロビジョニングの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の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のスナップショットを作成しません。

シンプロビジョニングの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ていない作成元ボリューム用にも作成できます。次の手順では、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するさまざまな方法について説明します。

前提条件

- シンプロビジョニングされた論理ボリュームを作成している。詳細は、シンプロビジョニングされた論理ボリュームの作成 を参照してください。

手順

- 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のスナップショットボリューム (名前: **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
```

注記

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。シンボリュームのサイズを拡張する方法は、シンプロビジョニングされた論理ボリュームの作成 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。外部の作成元ボリュームは、複数の異なるシンプールからであっても、多くのシンプロビジョニングのスナップショットボリュームで使用でき、共有できます。外部の作成元は、シンプロビジョニングの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。次の例では、**origin_volume** という名前の読み取り専用の非アクティブな論理ボリュームのシンスナップショットボリュームを作成します。このシンプロビジョニングの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** 内のシンプロビジョニングのスナップショットボリューム **mythinsnap** に対する外部の作成元になります。作成元のボリュームは、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に属している必要があります。作成元の論理ボリュームを指定するときは、ボリュームグループを指定しないでください。

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

- 以下のコマンドを実行して、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のスナップショットボリュームを作成できます。

```
# lvcreate -s vg001/mysnapshot1 --name mysnapshot2
Logical volume "mysnapshot2" created.
```

3 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するには、次のコマンドを使用します。

```
# lvcreate -s vg001/mysnapshot2 --name mysnapshot3
Logical volume "mysnapshot3" created.
```

検証

- シンスナップショット論理ボリュームのすべての祖先と子孫のリストを表示します。

```
$ lvs -o name,lv_ancestors,lv_descendants vg001
  LV          Ancestors                       Descendants
  mysnapshot2 mysnapshot1,thinvolume          mysnapshot3
  mysnapshot1 thinvolume                      mysnapshot2,mysnapshot3
  mysnapshot3 mysnapshot2,mysnapshot1,thinvolume
  mythinpool
  thinvolume                                  mysnapshot1,mysnapshot2,mysnapshot3
```

ここでは、以下のようになります。

- **thinvolume** は、ボリュームグループ **vg001** で元となるボリュームです。
- **mysnapshot1** は **thinvolume** のスナップショットです。
- **mysnapshot2** は **mysnapshot1** のスナップショットです。

- mysnapshot3 は mysnapshot2 のスナップショットです、

注記

lv_ancestors フィールドと **lv_descendants** フィールドには、既存の依存関係が表示されます。ただし、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ためです。

関連情報

- **lvcreate(8)** の man ページ

第8章 LVM のトラブルシューティング

LVM ツールを使用して、LVM ボリュームおよびグループのさまざまな問題のトラブルシューティングを行うことができます。

8.1. LVM での診断データの収集

LVM コマンドが想定どおりに機能しない場合は、以下の方法で診断情報を収集できます。

手順

- 以下の方法を使用して、さまざまな診断データを収集します。
  - **-v** 引数を LVM コマンドに追加して、コマンドの出力の詳細レベルを増やします。**v** を追加すると、詳細度をさらに増やすことができます。**v** は最大 4 つ許可されます（例：**-vvvv**）。
  - **/etc/lvm/lvm.conf** 設定ファイルの **log** セクションで、**level** オプションの値を増やします。これにより、LVM がシステムログにより多くの情報を提供します。
  - 問題が論理ボリュームのアクティブ化に関連する場合は、アクティブ化中に LVM がログメッセージをログに記録できるようにします。
    i. **/etc/lvm/lvm.conf** 設定ファイルの **log** セクションで **activation = 1** オプションを設定します。
    ii. LVM コマンドに **-vvvv** オプションを付けて実行します。
    iii. コマンドの出力を確認します。
    iv. **activation** オプションを **0** にリセットします。
    オプションを **0** にリセットしないと、メモリー不足の状況でシステムが応答しなくなる可能性があります。
  - 診断目的で情報ダンプを表示します。

    ```
    # lvmdump
    ```

  - 追加のシステム情報を表示します。

    ```
    # lvs -v
    ```

    ```
    # pvs --all
    ```

    ```
    # dmsetup info --columns
    ```

  - **/etc/lvm/backup/** ディレクトリーの最後の LVM メタデータのバックアップと、**/etc/lvm/archive/** ディレクトリー内のアーカイブバージョンを確認します。
  - 現在の設定情報を確認します。

    ```
    # lvmconfig
    ```

  - **/run/lvm/hints** キャッシュファイルで、物理ボリュームを持つデバイスを記録します。

関連情報

- **lvmdump(8)** の man ページ

8.2. 障害の発生した LVM デバイスに関する情報の表示

ボリュームが失敗した理由を特定するのに役立つ、障害の発生した LVM ボリュームに関する情報を表示できます。

手順

- **vgs** ユーティリティーまたは **lvs** ユーティリティーを使用して、障害が発生したボリュームを表示します。

例8.1障害が発生したボリュームグループ

この例では、ボリュームグループ **myvg** を構成するデバイスのいずれかが失敗しています。ボリュームグループは使用できませんが、障害が発生したデバイスに関する情報を表示できます。

```
# vgs --options +devices
 /dev/vdb1: open failed: No such device or address
 /dev/vdb1: open failed: No such device or address
  WARNING: Couldn't find device with uuid 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s.
  WARNING: VG myvg is missing PV 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s (last written to /dev/sdb1).
  WARNING: Couldn't find all devices for LV myvg/mylv while checking used and assumed devices.

VG    #PV #LV #SN Attr   VSize  VFree  Devices
myvg   2   2   0 wz-pn- <3.64t <3.60t [unknown](0)
myvg   2   2   0 wz-pn- <3.64t <3.60t [unknown](5120),/dev/vdb1(0)
```

例8.2 障害が発生した論理ボリューム

この例では、ボリュームグループの論理ボリュームが失敗したためにデバイスのいずれかが失敗しています。コマンドの出力には、障害が発生した論理ボリュームが表示されます。

```
# lvs --all --options +devices

  /dev/vdb1: open failed: No such device or address
  /dev/vdb1: open failed: No such device or address
  WARNING: Couldn't find device with uuid 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s.
  WARNING: VG myvg is missing PV 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s (last written to /dev/sdb1).
  WARNING: Couldn't find all devices for LV myvg/mylv while checking used and assumed devices.

  LV    VG  Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert Devices
  mylv myvg -wi-a---p- 20.00g                                                     [unknown](0)
[unknown](5120),/dev/sdc1(0)
```

8.3. ボリュームグループから見つからない LVM 物理ボリュームの削除

物理ボリュームに障害が発生した場合は、ボリュームグループ内の残りの物理ボリュームをアクティブにし、その物理ボリュームを使用していたすべての論理ボリュームをボリュームグループから削除できます。

手順

1. ボリュームグループ内の残りの物理ボリュームをアクティベートします。

```
# vgchange --activate y --partial myvg
```

2. 削除する論理ボリュームを確認します。

```
# vgreduce --removemissing --test myvg
```

3. ボリュームグループから、失われた物理ボリュームを使用していた論理ボリュームをすべて削除します。

```
# vgreduce --removemissing --force myvg
```

4. オプション:保持する論理ボリュームを誤って削除した場合には、**vgreduce** 操作を元に戻すことができます。

```
# vgcfgrestore myvg
```

警告

シンプールを削除すると、LVM は操作を元に戻すことができません。

8.4. 見つからない LVM 物理ボリュームのメタデータの検索

物理ボリュームのボリュームグループメタデータ領域が誤って上書きされたり、破棄されたりする場合は、メタデータ領域が正しくないことを示すエラーメッセージか、システムが特定の UUID を持つ物理ボリュームを見つけることができないことを示すエラーメッセージが表示されます。

この手順では、物理ボリュームが見つからないか、破損している、アーカイブされた最新のメタデータを見つけます。

手順

1. 物理ボリュームを含むボリュームグループのアーカイブされたメタデータファイルを検索します。アーカイブされたメタデータファイルは、**/etc/lvm/archive/volume-group-name_backup-number.vg** パスにあります。

```
# cat /etc/lvm/archive/myvg_00000-1248998876.vg
```

00000-1248998876 を backup-number に置き換えます。ボリュームグループの番号が最も高い、既知の有効なメタデータファイルの最後のものを選択します。

2. 物理ボリュームの UUID を検索します。以下の方法のいずれかを使用します。

   - 論理ボリュームを一覧表示します。

     ```
     # lvs --all --options +devices

       Couldn't find device with uuid 'FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk'.
     ```

   - アーカイブされたメタデータファイルを確認します。ボリュームグループ設定の **physical_volumes** セクションで、**id =** のラベルが付いた値として UUID を検索します。

   - **--partial** オプションを使用してボリュームグループを非アクティブにします。

     ```
     # vgchange --activate n --partial myvg

       PARTIAL MODE. Incomplete logical volumes will be processed.
       WARNING: Couldn't find device with uuid 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s.
       WARNING: VG myvg is missing PV 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s (last written to /dev/vdb1).
       0 logical volume(s) in volume group "myvg" now active
     ```

8.5. LVM 物理ボリュームでのメタデータの復元

この手順では、破損したり、新しいデバイスに置き換えたりする物理ボリュームのメタデータを復元します。物理ボリュームのメタデータ領域を書き換えて、物理ボリュームからデータを復旧できる場合があります。

警告

作業用の LVM 論理ボリュームでこの手順を実行しないでください。誤った UUID を指定すると、データが失われることになります。

前提条件

- 見つからない物理ボリュームのメタデータを特定している。詳細は、「見つからない LVM 物理ボリュームのメタデータの検索」を参照してください。

手順

1. 物理ボリュームでメタデータを復元します。

   ```
   # pvcreate --uuid physical-volume-uuid \
            --restorefile /etc/lvm/archive/volume-group-name_backup-number.vg \
            block-device
   ```

注記

コマンドは、LVM メタデータ領域のみを上書きし、既存のデータ領域には影響を与えません。

例8.3 /dev/vdb1 での物理ボリュームの復元

以下の例では、以下のプロパティーで **/dev/vdb1** デバイスを物理ボリュームとしてラベル付けします。

- **FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk** の UUID
- **VG_00050.vg** に含まれるメタデータ情報 (ボリュームグループの最新のアーカイブメタデータ)

```
# pvcreate --uuid "FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk" \
        --restorefile /etc/lvm/archive/VG_00050.vg \
        /dev/vdb1


 ...
 Physical volume "/dev/vdb1" successfully created
```

2. ボリュームグループのメタデータを復元します。

```
# vgcfgrestore myvg

 Restored volume group myvg
```

3. ボリュームグループの論理ボリュームを表示します。

```
# lvs --all --options +devices myvg
```

現在、論理ボリュームは非アクティブです。以下に例を示します。

```
 LV     VG   Attr   LSize   Origin Snap%  Move Log Copy%  Devices
 mylv myvg   -wi--- 300.00G                             /dev/vdb1 (0),/dev/vdb1(0)
 mylv myvg   -wi--- 300.00G                             /dev/vdb1 (34728),/dev/vdb1(0)
```

4. 論理ボリュームのセグメントタイプが RAID の場合は、論理ボリュームを再同期します。

```
# lvchange --resync myvg/mylv
```

5. 論理ボリュームを非アクティブにします。

```
# lvchange --activate y myvg/mylv
```

6. ディスク上の LVM メタデータが、それを上書きしたものと同じかそれ以上のスペースを使用する場合は、この手順で物理ボリュームを回復できます。メタデータを上書きしたものがメタデータ領域を超えると、ボリューム上のデータが影響を受ける可能性があります。そのデータを復元するには、**fsck** コマンドを使用することができます。

検証手順

- アクティブな論理ボリュームを表示します。

```
# lvs --all --options +devices

 LV     VG   Attr   LSize   Origin Snap%  Move Log Copy%  Devices
mylv myvg   -wi--- 300.00G                                /dev/vdb1 (0),/dev/vdb1(0)
mylv myvg   -wi--- 300.00G                                /dev/vdb1 (34728),/dev/vdb1(0)
```

8.6. LVM 出力の丸めエラー

ボリュームグループの領域使用量を報告する LVM コマンドは、報告された数を **2** 進法に切り上げ、人間が判読できる出力を提供します。これには、**vgdisplay** ユーティリティーおよび **vgs** ユーティリティーが含まれます。

丸めの結果、報告された空き領域の値は、ボリュームグループが提供する物理エクステントよりも大きくなる可能性があります。報告された空き領域のサイズの論理ボリュームを作成しようとすると、以下のエラーが発生する可能性があります。

```
Insufficient free extents
```

エラーを回避するには、ボリュームグループの空き物理エクステントの数を調べる必要があります。これは、空き領域の正確な値です。次に、エクステントの数を使用して、論理ボリュームを正常に作成できます。

8.7. LVM ボリューム作成時の丸めエラーの防止

LVM 論理ボリュームを作成する場合は、丸めエラーを防ぐために論理ボリュームの論理エクステントの数を指定できます。

手順

1. ボリュームグループの空き物理エクステントの数を検索します。

```
# vgdisplay myvg
```

例8.4 ボリュームグループの空きエクステント

たとえば、以下のボリュームグループには 8780 個のの空き物理エクステントがあります。

```
--- Volume group ---
 VG Name               myvg
 System ID
 Format                lvm2
 Metadata Areas        4
 Metadata Sequence No  6
 VG Access             read/write
[...]
Free  PE / Size       8780 / 34.30 GB
```

2. 論理ボリュームを作成します。ボリュームサイズをバイトではなくエクステントに入力します。

例8.5 エクステントの数を指定して論理ボリュームを作成

```
# lvcreate --extents 8780 --name mylv myvg
```

例8.6 残りの領域をすべて使用する論理ボリュームの作成

または、論理ボリュームを拡張して、ボリュームグループ内の残りの空き領域の割合を使用できます。以下に例を示します。

```
# lvcreate --extents 100%FREE --name mylv myvg
```

検証手順

- ボリュームグループが使用するエクステントの数を確認します。

```
# vgs --options +vg_free_count,vg_extent_count

  VG     #PV #LV #SN  Attr   VSize   VFree  Free  #Ext
  myvg   2   1   0   wz--n- 34.30G    0     0     8780
```