Red Hat Enterprise Linux 7

ストレージ管理ガイド

RHEL 7 での単一ノードストレージのデプロイおよび設定

Last Updated: 2024-04-25

Red Hat Enterprise Linux 7 ストレージ管理ガイド

RHEL 7 での単一ノードストレージのデプロイおよび設定

Milan Navrátil
Red Hat Customer Content Services

Jacquelynn East
Red Hat Customer Content Services

Don Domingo
Red Hat Customer Content Services

編集者

Marek Suchánek
Red Hat Customer Content Services
msuchane@redhat.com

Apurva Bhide
Red Hat Customer Content Services
abhide@redhat.com

法律上の通知

Copyright © 2018 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

本ガイドでは、Red Hat Enterprise Linux 7 でストレージデバイスおよびファイルシステムを効果的に管理する方法を説明します。これは、Red Hat Enterprise Linux または Fedora の基本的な知識を持つシステム管理者向けのものです。

目次

第1章 概要 **7**
1.1. RED HAT ENTERPRISE LINUX 7 の新機能および改良された機能 7

パート I. ファイルシステム **9**

第2章 ファイルシステム構造とメンテナンス **10**
2.1. ファイルシステム階層標準 (FHS) の概要 10
2.2. 特別な RED HAT ENTERPRISE LINUX ファイルの場所 18
2.3. /PROC 仮想ファイルシステム 18
2.4. 未使用ブロックの破棄 19

第3章 XFS ファイルシステム **20**
3.1. XFS ファイルシステムの作成 21
3.2. XFS ファイルシステムのマウント 22
3.3. XFS クォータ管理 23
3.4. XFS ファイルシステムのサイズの拡大 25
3.5. XFS ファイルシステムの修復 26
3.6. XFS ファイルシステムの一時停止 26
3.7. XFS ファイルシステムのバックアップおよび復元 27
3.8. エラー動作の設定 30
3.9. XFS ファイルシステムのその他のユーティリティー 33
3.10. EXT4 から XFS への移行 34

第4章 EXT3 ファイルシステム。 **37**
4.1. EXT3 ファイルシステムの作成 38
4.2. EXT3 ファイルシステムへの変換 38
4.3. EXT2 ファイルシステムへの復元 39

第5章 EXT4 ファイルシステム **41**
5.1. EXT4 ファイルシステムの作成 42
5.2. EXT4 ファイルシステムのマウント 44
5.3. EXT4 ファイルシステムのサイズ変更 45
5.4. EXT2、EXT3、または EXT4 のファイルシステムのバックアップ 45
5.5. EXT2、EXT3、または EXT4 のファイルシステムの復元 47
5.6. EXT4 ファイルシステムのその他のユーティリティー 49

第6章 BTRFS (テクノロジープレビュー) **51**
6.1. BTRFS ファイルシステムの作成 51
6.2. BTRFS ファイルシステムのマウント 51
6.3. BTRFS ファイルシステムのサイズの変更 52
6.4. 複数のデバイスの統合ボリューム管理 55
6.5. SSD の最適化 59
6.6. BTRFS リファレンス 59

第7章 GLOBAL FILE SYSTEM 2 ..... **61**

第8章 NETWORK FILE SYSTEM (NFS) **62**
8.1. NFS の概要 62
8.2. NFS クライアントの設定 65
8.3. AUTOFS 66
8.4. 一般的な NFS マウントオプション 73
8.5. NFS サーバーの起動と停止 74
8.6. NFS サーバーの設定 75

8.7. NFS のセキュア化 84
8.8. NFS および RPCBIND 87
8.9. PNFS 88
8.10. NFS での PNFS SCSI レイアウトの有効化 89
8.11. NFS のリファレンス 94

第9章 サーバーメッセージブロック (SMB) 96
9.1. SMB 共有の提供 96
9.2. SMB 共有のマウント 96

第10章 FS-CACHE 102
10.1. パフォーマンスに関する保証 103
10.2. キャッシュの設定 103
10.3. NFS でのキャッシュの使用 105
10.4. キャッシュカリング制限の設定 106
10.5. 統計情報 107
10.6. FS-CACHE の参考資料 107

パート II. ストレージ管理 109

第11章 ストレージをインストールする際の注意点 110
11.1. 特に注意を要する事項について 110

第12章 ファイルシステム検査 113
12.1. FSCK のベストプラクティス 113
12.2. FSCK のファイルシステム固有の情報 114

第13章 PARTITIONS 118
使用中のデバイスでのパーティションの操作 118
パーティションテーブルの変更 118
13.1. パーティションテーブルの表示 119
13.2. パーティションの作成 121
13.3. パーティションの削除 124
13.4. パーティションタイプの設定 125
13.5. FDISK でのパーティションのサイズ変更 125

第14章 SNAPPER でのスナップショットの作成と管理 128
14.1. SNAPPER の初期設定の作成 128
14.2. SNAPPER スナップショットの作成 129
14.3. スナップショット間での変更の追跡 132
14.4. スナップショット間の変更を元に戻す 136
14.5. SNAPPER スナップショットの削除 138

第15章 SWAP 領域 139
15.1. スワップ領域の追加 140
15.2. スワップ領域の削除 142
15.3. SWAP 領域の移動 144

第16章 SYSTEM STORAGE MANAGER (SSM) 145
16.1. SSM のバックエンド 145
16.2. 一般的な SSM タスク 147
16.3. SSM リソース 155

第17章 ディスククォータ 156
17.1. ディスククォータの設定 156
17.2. ディスククォータの管理 163

17.3. ディスククオータのリファレンス 167

第18章 RAID (REDUNDANT ARRAY OF INDEPENDENT DISKS) 168
18.1. RAID のタイプ 168
18.2. RAID レベルとリニアサポート 170
18.3. LINUX RAID サブシステム 173
18.4. ANACONDA インストーラーでの RAID のサポート 173
18.5. インストール後のルートディスクの RAID1 への変換 174
18.6. RAID セットの設定 175
18.7. 高度な RAID デバイスの作成 175

第19章 MOUNT コマンドの使用 178
19.1. 現在マウントされているファイルシステムのリスト表示 178
19.2. ファイルシステムのマウント 179
19.3. ファイルシステムのアンマウント 192
19.4. マウント コマンドリファレンス 192

第20章 VOLUME_KEY 機能 194
20.1. VOLUME_KEY コマンド 194
20.2. VOLUME_KEY を個別のユーザーとして使用する 196
20.3. 大規模な組織での VOLUME_KEY の使用 197
20.4. VOLUME_KEY リファレンス 201

第21章 ソリッドステートディスクのデプロイメントガイドライン 202
デプロイメントに関する考慮事項 203
パフォーマンスチューニングの考慮事項 206

第22章 書き込みバリア 207
22.1. 書き込みバリアの重要性 207
22.2. 書き込みバリアの有効化と無効化 208
22.3. 書き込みバリアに関する考慮事項 209

第23章 ストレージ I/O アライメントとサイズ 211
23.1. ストレージアクセスパラメーター 211
23.2. ユーザー空間アクセス 212
23.3. I/O 規格 214
23.4. I/O パラメーターのスタッキング 216
23.5. 論理ボリュームマネージャー 216
23.6. パーティションおよびファイルシステムツール 217

第24章 リモートディスクレスシステムの設定 219
24.1. ディスクレスクライアントの TFTP サービスの設定 220
24.2. ディスクレスクライアント用の DHCP の設定 221
24.3. ディスクレスクライアントのエクスポートしたファイルシステムの設定 222

第25章 オンラインストレージ管理 225
25.1. ターゲットの設定 225
25.2. ISCSI イニシエーターの作成 237
25.3. チャレンジハンドシェイク認証プロトコルの設定 239
25.4. ファイバーチャネル 240
25.5. ファイバーチャネルオーバーイーサネットインターフェイスの設定 244
25.6. システムの起動時に FCOE インターフェイスを自動マウントする設定 246
25.7. ISCSI 248
25.8. 永続的な命名 249
25.9. ストレージデバイスの削除 256

25.10. ストレージデバイスへのパスの削除 258
25.11. ストレージデバイスまたはパスの追加 259
25.12. ストレージの相互接続のスキャン 261
25.13. ISCSI 検出設定 263
25.14. ISCSI オフロードおよびインターフェイスバインディングの設定 264
25.15. ISCSI 相互接続のスキャン 270
25.16. ISCSI ターゲットへのログイン 273
25.17. オンライン論理ユニットのサイズの変更 274
25.18. RESCAN-SCSI-BUS.SH を使用した論理ユニットの追加/削除 279
25.19. リンクロス動作の変更 280
25.20. SCSI コマンドタイマーおよびデバイスステータスの制御 284
25.21. オンラインストレージ設定のトラブルシューティング 285
25.22. EH_DEADLINE を使用したエラー復旧の最大時間の設定 287

第26章 DEVICE MAPPER MULTIPATHING (DM MULTIPATH) と仮想マシン用のストレージ 289
26.1. 仮想マシン用のストレージ 289
26.2. DM MULTIPATH 290

第27章 外部アレイ管理 (LIBSTORAGEMGMT) 291
27.1. LIBSTORAGEMGMT の概要 291
27.2. LIBSTORAGEMGMT の用語 293
27.3. LIBSTORAGEMGMT のインストール 295
27.4. LIBSTORAGEMGMT の使用 297

第28章 永続メモリー : NVDIMMS 303
NVDIMMs のインターリーブ 303
永続メモリーアクセスモード 304
28.1. NDCTL を使用した永続メモリーの設定 305
28.2. ブロックデバイスとして使用する永続メモリーの設定 (レガシーモード) 308
28.3. ファイルシステムのダイレクトアクセス向け永続メモリーの設定 308
28.4. デバイス DAX モードで使用する永続メモリーの設定 310
28.5. NVDIMM のトラブルシューティング 311

第29章 NVME OVER FABRIC デバイスの概要 317
29.1. RDMA を使用した NVME OVER FABRICS 317
29.2. FC を使用した NVME OVER FABRICS 319

パート III. VDO を使用したデータの重複排除と圧縮 325

第30章 VDO 統合 326
30.1. VDO の理論的概要 326
30.2. システム要件 330
30.3. VDO の使用 336
30.4. VDO の管理 342
30.5. デプロイメントシナリオ 355
30.6. VDO のチューニング 357
30.7. VDO コマンド 366
30.8. /SYSの統計ファイル 385

第31章 VDO 評価 386
31.1. 導入部分 386
31.2. テスト環境の準備 387
31.3. データ効率のテスト手順 392
31.4. パフォーマンステストの手順 405
31.5. 問題の報告 411

31.6. まとめ 412

付録A ストレージ管理に関連する RED HAT カスタマーポータルラボ 413
SCSI デコーダー 413
FILE SYSTEM LAYOUT CALCULATOR 413
LVM RAID CALCULATOR 413
ISCSI HELPER 413
SAMBA CONFIGURATION HELPER 414
MULTIPATH HELPER 414
NFS HELPER 414
MULTIPATH CONFIGURATION VISUALIZER 414
RHEL BACKUP AND RESTORE ASSISTANT 415

付録B 更新履歴 417

索引 418

第1章 概要

『ストレージ管理ガイド』 には、Red Hat Enterprise Linux 7 でサポートされるファイルシステムおよびデータストレージ機能に関する広範な情報が含まれています。本書は、単一ノード (非クラスター化) ストレージソリューションを管理する管理者向けのクイックリファレンスになります。

ストレージ管理ガイドは、ファイルシステム、ストレージ管理、および VDO を使用したデータの重複排除と圧縮の 3 つに分かれています。

ファイルシステムのセクションでは、Red Hat Enterprise Linux 7 がサポートするさまざまなファイルシステムについて詳しく説明します。ファイルシステムについての説明のほか、それらを最大限に活用する方法についても説明します。

ストレージ管理の部分では、Red Hat Enterprise Linux 7 がサポートするさまざまなツールとストレージ管理タスクについて詳しく説明します。ファイルシステムについての説明のほか、それらを最大限に活用する方法についても説明します。

VDO を使用したデータの重複排除と圧縮のセクションでは、VDO (Virtual Data Optimizer) について説明しています。VDO を使用して、ストレージ要件を減らす方法を説明します。

1.1. RED HAT ENTERPRISE LINUX 7 の新機能および改良された機能

Red Hat Enterprise Linux 7 におけるファイルシステムの機能拡張の特徴は、以下のとおりです。

eCryptfs は含まれていません。

Red Hat Enterprise Linux 7 の時点では、eCryptfs は含まれていません。ファイルシステムの暗号化の詳細については、Red Hat のセキュリティーガイドを参照してください。

System Storage Manager

Red Hat Enterprise Linux 7 には、コマンドラインインターフェイスを提供してさまざまなストレージ技術を管理する System Storage Manager と呼ばれる新しいアプリケーションが含まれています。詳細は、16章*System Storage Manager (SSM)* を参照してください。

XFS はデフォルトのファイルシステムです。

Red Hat Enterprise Linux 7 以降、XFS がデフォルトのファイルシステムになります。XFS ファイルシステムの詳細は、3章*XFS ファイルシステム* を参照してください。

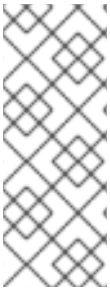
ファイルシステムの再構築

Red Hat Enterprise Linux 7 では、新しいファイルシステム構造が導入されています。ディレクトリー **/bin**、**/sbin**、/lib、および **/lib 64** は、**/usr** の下にネストされるようになりました。

Snapper

Red Hat Enterprise Linux 7 では、Snapper と呼ばれる新しいツールが導入されています。これにより、LVM および Btrfs のスナップショットの作成と管理が容易になります。詳細は、14章*Snapper でのスナップショットの作成と管理* を参照してください。

Btrfs (テクノロジープレビュー)



注記

Btrfs は、Red Hat Enterprise Linux 7 ではテクノロジープレビューとして利用できますが、Red Hat Enterprise Linux 7.4 リリース以降では非推奨になりました。Red Hat Enterprise Linux の将来のメジャーリリースで削除される予定です。

詳細は、Red Hat Enterprise Linux 7.4 リリースノートの 非推奨の機能 を参照してください。

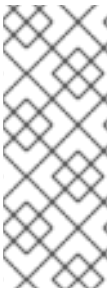
Btrfs は、統合 LVM 操作を含む、より優れたパフォーマンスとスケーラビリティーの提供を目指すローカルファイルシステムです。このファイルシステムは、Red Hat では完全にサポートされていないため、テクノロジープレビューとなります。Btrfs の詳細は、*6章Btrfs (テクノロジープレビュー)* を参照してください。

NFSv2 はサポートされなくなりました

Red Hat Enterprise Linux 7 以降、NFSv2 はサポートされなくなりました。

パート I. ファイルシステム

ファイルシステムのセクションでは、ファイルシステムの構造とメンテナンスに関する情報、Btrfs テクノロジープレビュー、および Red Hat が完全にサポートされるファイルシステム (ext3、ext4、GFS2、XFS、NFS、および FS-Cache) に関する情報を提供します。



注記

Btrfs は、Red Hat Enterprise Linux 7 ではテクノロジープレビューとして利用できますが、Red Hat Enterprise Linux 7.4 リリース以降では非推奨になりました。Red Hat Enterprise Linux の将来のメジャーリリースで削除される予定です。

詳細は、Red Hat Enterprise Linux 7.4 リリースノートの 非推奨の機能 を参照してください。

Red Hat Enterprise Linux ファイルシステムおよびストレージ制限の概要は、Red Hat ナレッジベースのRed Hat Enterprise Linux technology capabilities and limits を参照してください。

XFS は、Red Hat Enterprise Linux 7 および Red Hat におけるデフォルトのファイルシステムです。Red Hat では、別のファイルシステムを使用する強い理由がない限り、XFS を使用することを推奨します。一般的なファイルシステムとそのプロパティーに関する一般情報は、Red Hat ナレッジベースの記事How to Choose your Red Hat Enterprise Linux File System を参照してください。

第2章 ファイルシステム構造とメンテナンス

ファイルシステムの構造は、オペレーティングシステムの組織の最も基本的なレベルです。オペレーティングシステムがユーザー、アプリケーション、およびセキュリティーモデルと対話する方法は、オペレーティングシステムがストレージデバイスでファイルを編成する方法にほぼ常に依存します。一般的なファイルシステムの構造を提供することで、ユーザーとプログラムがファイルにアクセスして書き込むことができます。

ファイルシステムは、ファイルを 2 つの論理カテゴリーに分類します。

共有可能および共有不可能なファイル

共有可能 ファイルは、ローカルおよびリモートホストからアクセスできます。*Unsharable* ファイルはローカルでのみ利用可能です。

変数および静的ファイル

ドキュメントなどの *変数* ファイルはいつでも変更できます。バイナリー などの *静的* ファイルは、システム管理者のアクションなしでは変更されません。

この方法でファイルを分類すると、各ファイルの機能と、それらを保持するディレクトリーに割り当てられたパーミッションとを相互に関連付ける上で役立ちます。オペレーティングシステムとそのユーザーがファイルを操作する方法によって、ファイルが配置されるディレクトリー、そのディレクトリーが読み取り専用パーミッションまたは読み取りと書き込みのパーミッションでマウントされているかどうか、および各ユーザーがそのファイルにアクセスできるレベルが決まります。この組織のトップレベルは非常に重要です。基礎となるディレクトリーへのアクセスを制限できます。そうしないと、トップレベルから下のアクセスルールが厳密な構造に準拠していない場合に、セキュリティー問題が発生する可能性があります。

2.1. ファイルシステム階層標準 (FHS) の概要

Red Hat Enterprise Linux は、*Filesystem Hierarchy Standard* (*FHS*) のファイルシステム構造を使用します。これは、多くのファイルタイプやディレクトリーの名前、場所、およびパーミッションを定義します。

FHS ドキュメントは、すべての FHS 準拠のファイルシステムにとって信頼できるリファレンスですが、この標準では、多くの領域が未定義または拡張可能です。このセクションでは、標準の概要と、標準でカバーされていないファイルシステムの部分について説明します。

FHS コンプライアンスの 2 つの最も重要な要素は次のとおりです。

- 他の FHS 準拠システムとの互換性
- **/usr/** パーティションを読み取り専用としてマウントする機能。**/usr/** には一般的な実行ファイルが含まれており、ユーザーが変更すべきではないため、これは重要です。さらに、**/usr/** は読み取り専用としてマウントされているため、CD-ROM ドライブまたは他のマシンから、読み取り専用 NFS マウントを介してマウントできる必要があります。

2.1.1. FHS 組織

ここで記載されているディレクトリーおよびファイルは、FHS ドキュメントで指定された小さなサブセットです。最も詳しい情報については、最新の FHS ドキュメント http://refspecs.linuxfoundation.org/FHS_3.0/fhs-3.0.pdf を参照してください。file-hierarchy(7) の man ページにも概要が記載されています。

注記

利用可能なディレクトリーは、指定のシステムにインストールされた内容によって異なります。以下のリストは、見つかる可能性のあるものの一例になります。

2.1.1.1. ファイルシステム情報の収集

df コマンド

df コマンドは、システムのディスク領域の使用状況を報告します。出力は以下のようになります。

例2.1 df コマンドの出力

```
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
                      11675568   6272120   4810348  57% / /dev/sda1
                        100691      9281     86211  10% /boot
none                    322856         0    322856   0% /dev/shm
```

デフォルトでは、**df** は1キロバイトブロックでパーティションのサイズと、使用中および利用可能なディスク容量をキロバイト単位で表示します。メガバイトおよびギガバイトで情報を表示するには、**df -h** を使用します。**-h** 引数は、"human-readable" 形式を表します。**df -h** の出力は以下のようになります。

例2.2 df -h コマンドの出力

```
Filesystem           Size  Used Avail Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
                      12G  6.0G  4.6G  57% / /dev/sda1
  99M  9.1M   85M  10% /boot
none    316M     0  316M   0% /dev/shm
```

注記

この例では、マウントされたパーティション **/dev/shm** はシステムの仮想メモリーファイルシステムを表します。

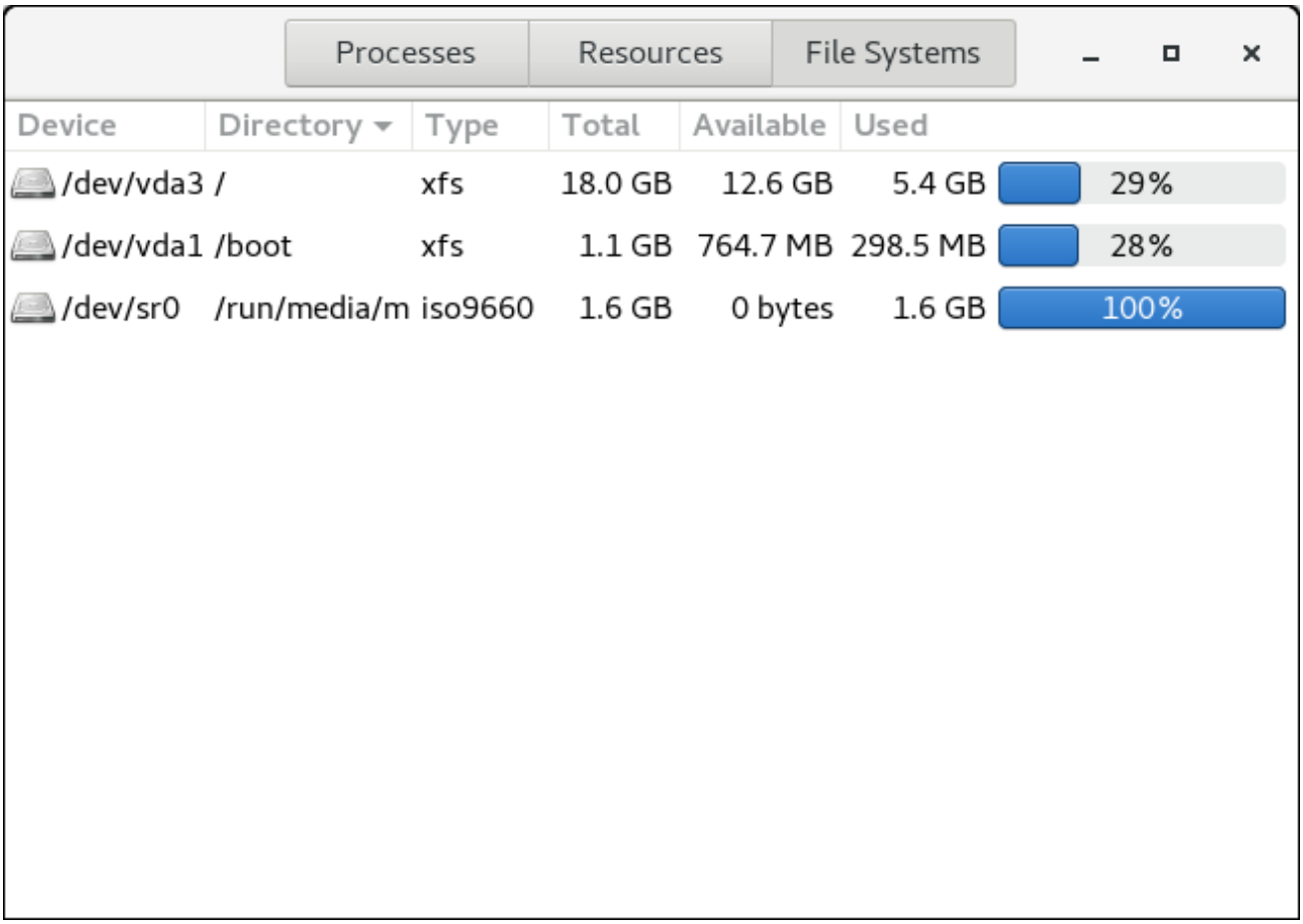
du コマンド

du コマンドは、ディレクトリー内のファイルが使用している推定容量を表示し、各サブディレクトリーのディスク使用量を表示します。**du** の出力の最後の行は、ディレクトリーの合計ディスク使用量を表示します。人間が判読できる形式でディレクトリーの合計ディスク使用量のみを表示するには、**du -hs** を使用します。その他のオプションは、**man du** を参照してください。

GNOME システムモニター

グラフィカル形式でシステムのパーティションとディスク領域の使用状況を表示するには、**Applications → Tools System Monitor** をクリックするか、**gnome-system-monitor** コマンドを使用して Gnome **System Monitor**を使用します。ファイルシステム タブ **を選択** して、システムのパーティションを表示します。以下の図は、**ファイルシステム タブを** 示しています。

図2.1 GNOME システムモニターのファイルシステムタブ



[D]

2.1.1.2. /boot/ ディレクトリー

/boot/ ディレクトリーには、Linux カーネルなど、システムを起動するために必要な静的ファイルが含まれます。これらのファイルは、システムが正しく起動するためには不可欠です。

警告

/boot/ ディレクトリーを削除しないでください。削除すると、システムが起動できなくなります。

2.1.1.3. /dev/ ディレクトリー

/dev/ ディレクトリーには、以下のデバイス種別を表すデバイスノードが含まれます。

- システムに接続されているデバイス
- カーネルが提供する仮想デバイス

これらのデバイスノードは、システムが適切に機能するために不可欠です。**udevd** デーモンは、必要に応じて **/dev/** 内のデバイスノードを作成および削除します。

/dev/ ディレクトリーおよびサブディレクトリー内のデバイスは、**文字** （マウスやキーボードなどの入出力のシリアルストリームのみを提供）または **ブロック** （ハードドライブやフロッピードライブなどのランダムにアクセス可能）のいずれかで定義されます。GNOME または KDE がインストールされている場合は、一部のストレージデバイスは (USB などで) 接続または (CD または DVD ドライブなどの) 挿入時に自動的に検出され、内容を表示するポップアップウィンドウが表示されます。

表2.1 /dev ディレクトリーの一般的なファイルの例

ファイル	説明
/dev/hda	プライマリー IDE チャネル上のマスターデバイス。
/dev/hdb	プライマリー IDE チャンネル上のスレーブデバイス。
/dev/tty0	最初の仮想コンソール。
/dev/tty1	2 番目の仮想コンソール
/dev/sda	プライマリー SCSI または SATA チャネル上の最初のデバイス。
/dev/lp0	最初の並列ポート。

有効なブロックデバイスは、以下の 2 種類のエントリーのいずれかになります。

マップされたデバイス

ボリュームグループの論理ボリューム（例： **/dev/mapper/VolGroup00-LogVol02** ）。

静的デバイス

たとえば、**/dev/*sdbX*** などの従来のストレージボリューム。*sdb* はストレージデバイス名で、*X* はパーティション番号になります。**/dev/sdb*X*** は、**/dev/disk/by-id/*WWID*** または **/dev/disk/by-uuid/*UUID*** にすることもできます。詳細は、「永続的な命名」 を参照してください。

2.1.1.4. /etc/ ディレクトリー

/etc/ ディレクトリーは、マシンのローカルとなる設定ファイル用に予約されています。バイナリーを含めることはできません。バイナリーがある場合は、それらを **/usr/bin/** または **/usr/sbin/** に移動します。

たとえば、**/etc/skel/** ディレクトリーにはスケルトンユーザーファイルが格納されます。このファイルは、ユーザーの初回作成時にホームディレクトリーを設定するために使用されます。また、アプリケーションはこのディレクトリーに設定ファイルを保存し、実行時にそれらを参照する可能性があります。**/etc/exports** ファイルは、リモートホストにどのファイルシステムをエクスポートするかを制御します。

2.1.1.5. /mnt/ ディレクトリー

/mnt/ ディレクトリーは、NFS ファイルシステムのマウントなど、一時的にマウントされたファイルシステム用に予約されています。すべてのリムーバブルストレージメディアには、**/media/** ディレクト

リーを使用します。自動的に検出されたリムーバブルメディアは、**/media** ディレクトリーにマウントされます。

重要

/mnt ディレクトリーは、インストールプログラムでは使用しないでください。

2.1.1.6. /opt/ ディレクトリー

/opt/ ディレクトリーは通常、デフォルトのインストールの一部ではないソフトウェアおよびアドオンパッケージ用に予約されています。**/opt/** にインストールするパッケージは、その名前を持つディレクトリーを作成します（例： **/opt/*packagename*/** ）。多くの場合、このようなパッケージは予測可能なサブディレクトリー構造に従います。ほとんどの場合、バイナリーを **/opt/*packagename*/bin/** に保存し、**man** ページを **/opt/*packagename*/man/** に保存します。

2.1.1.7. /proc/ ディレクトリー

/proc/ ディレクトリーには、カーネルから情報を抽出するか、カーネルに情報を送信する特別なファイルが含まれています。このような情報の例には、システムメモリー、CPU 情報、およびハードウェア設定が含まれます。**/proc/** の詳細は、「/proc 仮想ファイルシステム」 を参照してください。

2.1.1.8. /srv/ ディレクトリー

/srv/ ディレクトリーには、Red Hat Enterprise Linux システムが提供するサイト固有のデータが含まれます。このディレクトリーは、FTP、WWW、または CVS などの特定サービスのデータファイルの場所をユーザーに提供します。特定のユーザーのみに関連するデータは、**/home/** ディレクトリーに置く必要があります。

2.1.1.9. /sys/ ディレクトリー

/sys/ ディレクトリーは、カーネルに固有の新しい **sysfs** 仮想ファイルシステムを使用します。カーネルのホットプラグハードウェアデバイスのサポートが増えると、**/sys/** ディレクトリーには、**/proc/** が保持する情報と同様の情報が含まれますが、ホットプラグデバイスに固有のデバイス情報の階層ビューが表示されます。

2.1.1.10. /usr/ ディレクトリー

/usr/ ディレクトリーは、複数のマシンにまたがって共有できるファイル用です。**/usr/** ディレクトリーは、多くの場合、独自のパーティションにあり、読み取り専用でマウントされます。少なくとも、**/usr/** には以下のサブディレクトリーが含まれている必要があります。

/usr/bin

このディレクトリーはバイナリーに使用されます。

/usr/etc

このディレクトリーは、システム全体の設定ファイルに使用されます。

/usr/games

このディレクトリーにはゲームが保存されています。

/usr/include

このディレクトリーは C ヘッダーファイルに使用されます。

/usr/kerberos

このディレクトリーは、Kerberos 関連のバイナリーおよびファイルに使用されます。

/usr/lib

このディレクトリーは、シェルスクリプトまたはユーザーが直接使用するように設計されていないオブジェクトファイルやライブラリーに使用されます。

Red Hat Enterprise Linux 7.0 以降、**/lib/** ディレクトリーは **/usr/lib** に統合されました。**/usr/bin/** および **/usr/sbin/** でバイナリーを実行するために必要なライブラリーも含まれるようになりました。これらの共有ライブラリーイメージは、システムを起動したり、root ファイルシステム内でコマンドを実行したりするために使用されます。

/usr/libexec

このディレクトリーには、他のプログラムによって呼び出される小さなヘルパープログラムが含まれます。

/usr/sbin

Red Hat Enterprise Linux 7.0 以降、**/sbin** は **/usr/sbin** に移動しました。これは、システムの起動、復元、復旧、または修復に不可欠なものを含む、すべてのシステム管理バイナリーが含まれていることを意味します。**/usr/sbin/** のバイナリーを使用するには、root 権限が必要です。

/usr/share

このディレクトリーには、アーキテクチャー固有ではないファイルを保存します。

/usr/src

このディレクトリーには、ソースコードが保存されます。

/var/tmpにリンクされている /usr/tmp

このディレクトリーには、一時ファイルが保存されます。

/usr/ ディレクトリーには、**/local/** サブディレクトリーも含まれている必要があります。FHS によると、このサブディレクトリーは、ソフトウェアをローカルでインストールする際にシステム管理者によって使用されるので、システムの更新中に上書きされないようにする必要があります。 **/usr/ local** ディレクトリーには /usr/ と似ており、以下のサブディレクトリーが含まれます。

- **/usr/local/bin**
- **/usr/local/etc**
- **/usr/local/games**
- **/usr/local/include**
- **/usr/local/lib**
- **/usr/local/libexec**
- **/usr/local/sbin**
- **/usr/local/share**
- **/usr/local/src**

Red Hat Enterprise Linux の **/usr/local/** の使用は FHS とは若干異なります。FHS は、 **/usr/local/** を使用して、システムソフトウェアのアップグレードから安全に保つ必要があるソフトウェアを保存する必要があることを示しています。**RPM Package Manager** はソフトウェアのアップグレードを安全に実行できるため、ファイルを **/usr/local/** に保存してファイルを保護する必要はありません。

代わりに、Red Hat Enterprise Linux は、マシンにローカルなソフトウェアに **/usr/local/** を使用します。たとえば、**/usr/** ディレクトリーがリモートホストから読み取り専用 NFS 共有としてマウントされている場合でも、**/usr/local/** ディレクトリーにパッケージまたはプログラムをインストールすることができます。

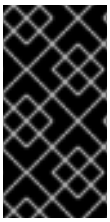
2.1.1.11. /var/ ディレクトリー

FHS では、Linux が **/usr/** を読み取り専用としてマウントする必要があるため、ログファイルを書き込むプログラムや、**spool/** または **lock/** ディレクトリーが必要なプログラムは、それらを **/var/** ディレクトリーに書き込む必要があります。FHS は、**/var/** は、スプールディレクトリーとファイル、ロギングデータ、一時的ファイルなどの変数データ用です。

以下は、**/var/** ディレクトリーにあるディレクトリーの一部です。

- **/var/account/**
- **/var/arpwatch/**
- **/var/cache/**
- **/var/crash/**
- **/var/db/**
- **/var/empty/**
- **/var/ftp/**
- **/var/gdm/**
- **/var/kerberos/**
- **/var/lib/**
- **/var/local/**
- **/var/lock/**
- **/var/log/**
- **/var/spool/mail/**にリンクされた **/var/mail**
- **/var/mailman/**
- **/var/named/**
- **/var/nis/**
- **/var/opt/**
- **/var/preserve/**

- **/var/run/**
- **/var/spool/**
- **/var/tmp/**
- **/var/tux/**
- **/var/www/**
- **/var/yp/**



重要

/var/run/media/*ユーザー* ディレクトリーには、USB ストレージメディア、DVD、CD-ROM、Zip ディスクなどのリムーバブルメディアのマウントポイントとして使用されるサブディレクトリーが含まれます。以前は、**/media/** ディレクトリーがこの目的で使用されていたことに注意してください。

メッセージ や **lastlog** などのシステムログファイルは、**/var/log/** ディレクトリーに移動します。**/var/lib/rpm/** ディレクトリーには RPM システムデータベースが含まれます。ロックファイルは、通常は ファイルを使用するプログラムのディレクトリーにある **/var/lock/** ディレクトリーに移動します。**/var/spool/** ディレクトリーには、一部のプログラムのデータファイルを保存するサブディレクトリーがあります。これらのサブディレクトリーには以下が含まれます。

- **/var/spool/at/**
- **/var/spool/clientmqueue/**
- **/var/spool/cron/**
- **/var/spool/cups/**
- **/var/spool/exim/**
- **/var/spool/lpd/**
- **/var/spool/mail/**
- **/var/spool/mailman/**
- **/var/spool/mqueue/**
- **/var/spool/news/**
- **/var/spool/postfix/**
- **/var/spool/repackage/**
- **/var/spool/rwho/**
- **/var/spool/samba/**
- **/var/spool/squid/**
- **/var/spool/squirrelmail/**

- **/var/spool/up2date/**
- **/var/spool/uucp/**
- **/var/spool/uucppublic/**
- **/var/spool/vbox/**

2.2. 特別な RED HAT ENTERPRISE LINUX ファイルの場所

Red Hat Enterprise Linux は、特別なファイルに対応するために FHS 構造を若干拡張しています。

RPM に関連するほとんどのファイルは、**/var/lib/rpm/** ディレクトリーに保持されます。RPM の詳細は、**man rpm** を参照してください。

/var/cache/yum/ ディレクトリーには、システムの RPM ヘッダー情報など **、パッケージアップデータ** が使用するファイルが含まれます。この場所は、システムの更新中にダウンロードされた RPM を一時的に保存するためにも使用できます。Red Hat Network の詳細は https://rhn.redhat.com/ を参照してください。

Red Hat Enterprise Linux に固有の別の場所は、**/etc/sysconfig/** ディレクトリーです。このディレクトリーには、さまざまな設定情報が格納されています。システムの起動時に実行されるスクリプトの多くは、このディレクトリー内のファイルを使用します。

2.3. /PROC 仮想ファイルシステム

ほとんどのファイルシステムとは異なり、**/proc** にはテキストファイルもバイナリーファイルも含まれていません。*仮想ファイル* を格納するため、**/proc** は仮想ファイルシステムと呼ばれます。これらの仮想ファイルは、大量の情報が含まれている場合でも、サイズは通常 0 バイトになります。

/proc ファイルシステムはストレージには使用されません。この主な目的は、ハードウェア、メモリー、実行中のプロセス、および他のシステムコンポーネントにファイルベースのインターフェイスを提供することです。リアルタイム情報は、対応する **/proc** ファイルを表示することで、多くのシステムコンポーネントで取得できます。**/proc** 内の一部のファイルは、カーネルを設定するために（ユーザーとアプリケーションの両方で）操作することもできます。

以下の **/proc** ファイルは、システムストレージの管理および監視に関連しています。

/proc/devices
現在設定されているさまざまな文字およびブロックデバイスを表示します。

/proc/filesystems
カーネルで現在対応しているすべてのファイルシステムタイプをリスト表示します。

/proc/mdstat
システム上の複数ディスクまたは RAID 設定に関する現在の情報が含まれます (存在する場合)。

/proc/mounts
システムで現在使用しているマウントをすべてリスト表示します。

/proc/partitions
パーティションブロック割り当て情報が含まれます。

/proc ファイルシステムの詳細は、Red Hat Enterprise Linux 7『 デプロイメントガイド』 を参照してください。

2.4. 未使用ブロックの破棄

バッチ破棄とオンライン破棄操作は、ファイルシステムで使用されていないブロックを破棄するマウントされたファイルシステムの機能です。これは、ソリッドステートドライブおよびシンプロビジョニングされたストレージの両方に役立ちます。

- **バッチ破棄操作** は、ユーザーが **fstrim** コマンドを使用して明示的に実行します。このコマンドは、ファイルシステム内にある未使用のブロックで、管理者が指定した基準に一致するものをすべて破棄します。

- **オンライン破棄操作** は、マウント時に、**mount** コマンドの一部として **-o discard** オプションを使用するか、**/etc/fstab** ファイルの **discard** オプションで指定します。ユーザーによる介入なしでリアルタイムで実行されます。オンライン破棄操作は、使用中から空きに移行しているブロックのみを破棄します。

どちらの操作タイプも、Red Hat Enterprise Linux 6.2 以降の ext4 ファイルシステムと、Red Hat Enterprise Linux 6.4 以降の XFS ファイルシステムでの使用がサポートされています。また、ファイルシステムの基礎となるブロックデバイスは、物理的な破棄操作に対応している必要があります。**/sys/block/*デバイス*/queue/discard_max_bytes ファイルに保存されている値がゼロでない場合、物理的な破棄** 操作がサポートされます。

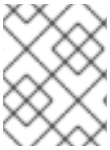
fstrim コマンドを以下のいずれかで実行している場合：

- 破棄操作をサポートしていないデバイス、または

- 複数のデバイスで設定され、そのデバイスのいずれかが破棄操作をサポートしていない論理デバイス (LVM または MD)

以下のメッセージが表示されます。

```
fstrim -v /mnt/non_discard
fstrim: /mnt/non_discard: the discard operation is not supported
```



注記

mount コマンドを使用すると、**-o discard** オプションを使用して、破棄操作に対応していないデバイスをマウントできます。

システムのワークロードがバッチ破棄を実行できない場合、またはパフォーマンスを維持するためにオンライン破棄操作が必要な場合を除いて、Red Hat はバッチ破棄操作を推奨します。

詳細は fstrim(8) および mount(8) の man ページを参照してください。

第3章 XFS ファイルシステム

XFS は、元々は Silicon Graphics, Inc で設計された非常に拡張性の高い、高性能のファイルシステムです。XFS は、Red Hat Enterprise Linux 7 のデフォルトのファイルシステムです。

XFS の主な機能

- XFS は、*メタデータジャーナリング*をサポートしているため、クラッシュリカバリーを迅速に行うことができます。
- XFS ファイルシステムは、マウントされ、アクティブな状態でデフラグし、拡張できます。
- また、Red Hat Enterprise Linux 7 は XFS 固有のバックアップおよび復元ユーティリティーをサポートしています。

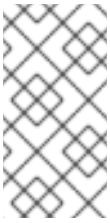
割り当て機能

XFS は、次の割り当てスキームを特長としています。

- エクステント (領域) ベースの割り当て
- ストライプを認識できる割り当てポリシー
- 遅延割り当て
- 領域の事前割り当て

遅延割り当てや他のパフォーマンスの最適化は、ext4 と同じように XFS に影響を与えます。つまり、XFS ファイルシステムへのプログラムの書き込みは、プログラムが後で **fsync （）** 呼び出しを発行しない限り、ディスク上のオンディスクである保証はありません。

ファイルシステム (ext4 および XFS) での遅延割り当ての影響の詳細は、5章*ext4 ファイルシステム* の 割り当て機能 を参照してください。



注記

ディスク容量が十分であるように見えても、ファイルの作成またはデプロイメントが予期しない ENOSPC 書き込みエラーで失敗することがあります。これは、XFS のパフォーマンス指向の設計が原因となります。実際には、残りの容量が数ブロックしかない場合にのみ発生するため、問題にはなりません。

その他の XFS 機能

XFS ファイルシステムは、以下もサポートしています。

***拡張属性* (xattr)**

これにより、システムが、ファイルごとに、名前と値の組み合わせを追加で関連付けられるようになります。これは、デフォルトで有効になっています。

クォータジャーナリング

クラッシュ後に行なわれる、時間がかかるクォータの整合性チェックが不要になります。

プロジェクト/ディレクトリークォータ

ディレクトリーツリー全体にクォータ制限を適用できます。

サブセカンド (一秒未満) のタイムスタンプ

これにより、タイムスタンプはサブセカンド (一秒未満) になることができます。

デフォルトの atime 動作は relatimeです。

XFS では、デフォルトで relatime **が** オンになっています。正しい **atime** 値を維持しながら、**noatime** と比較してオーバーヘッドはほとんどありません。

3.1. XFS ファイルシステムの作成

- XFS ファイルシステムを作成するには、以下のコマンドを使用します。

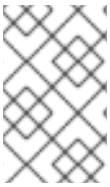
```
# mkfs.xfs block_device
```

  - *block_device* をブロックデバイスへのパスに置き換えます。たとえば、**/dev/sdb1**、**/dev/disk/by-uuid/05e99ec8-def1-4a5e-8a9d-5945339ceb2a**、または **/dev/my-volgroup/my-lv** です。
  - 通常、デフォルトのオプションは、一般的な使用に最適なものです。
  - 既存のファイルシステムを含むブロックデバイスで **mkfs.xfs** を使用する場合は、**-f** オプションを追加してそのファイルシステムを上書きします。

例3.1 mkfs.xfs コマンドの出力

以下は **mkfs.xfs** コマンドの出力例です。

```
meta-data=/dev/device            isize=256    agcount=4, agsize=3277258 blks
         =                       sectsz=512   attr=2
data     =                       bsize=4096   blocks=13109032, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0
log      =internal log           bsize=4096   blocks=6400, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```



注記

XFS ファイルシステムの作成後、そのサイズを縮小することはできません。ただし、**xfs_growfs** コマンドを使用して拡張することもできます。詳細は、「XFS ファイルシステムのサイズの拡大」 を参照してください。

ストライプ化ブロックデバイス

ストライプ化されたブロックデバイス (RAID5 アレイなど) の場合は、ファイルシステムの作成時にストライプジオメトリーを指定できます。適切なストライプジオメトリーを使用すると、XFS ファイルシステムのパフォーマンスが向上します。

LVM ボリュームまたは MD ボリュームでファイルシステムを作成する場合、**mkfs.xfs** は最適なジオメトリーを選択します。これは、ジオメトリー情報をオペレーティングシステムにエクスポートする一部のハードウェア RAID でも当てはまる場合があります。

デバイスがストライプジオメトリー情報をエクスポートすると、**mkfs** ユーティリティー(ext3、ext4、xfs の場合)は自動的にこのジオメトリーを使用します。ストレージに実際にストライプジオメトリーがある場合でも **mkfs** ユーティリティーによってストライプジオメトリーが検出されない場合は、次のオプションを使用してファイルシステムを作成するときに手動で指定できます。

su=*value*

ストライプユニットまたは RAID チャンクサイズを指定します。***値*** はバイト単位で指定する必要があります。オプションで **k**、**m**、または **g** の接尾辞を指定します。

sw=*value*

RAID デバイス内のデータディスク数、または 1 ストライプ内のストライプユニット数を指定します。

以下の例では、4 つのストライプユニットを含む RAID デバイスで 64k のチャンクサイズを指定しています。

```
# mkfs.xfs -d su=64k,sw=4 /dev/block_device
```

関連情報

XFS ファイルシステムの作成に関する詳細は、以下を参照してください。

- mkfs.xfs(8) の man ページ
- Red Hat Enterprise Linux Performance Tuning Guide のTuning XFSの章

3.2. XFS ファイルシステムのマウント

XFS ファイルシステムは、追加オプションなしでマウントできます。以下に例を示します。

```
# mount /dev/device /mount/point
```

Red Hat Enterprise Linux 7 のデフォルトは inode64 です。

注記

mke2fs とは異なり、**mkfs.xfs** は設定ファイルを使用しません。これらはすべてコマンドラインで指定されます。

書き込みバリア

デフォルトでは、XFS は書き込みバリアを使用して、書き込みキャッシュが有効なデバイスの電源が失われた場合でも、ファイルシステムの整合性を確保します。書き込みキャッシュがないデバイス、またはバッテリーバックアップされた書き込みキャッシュがあるデバイスの場合は、**nobarrier** オプションを使用してバリアを無効にします。

```
# mount -o nobarrier /dev/device /mount/point
```

書き込みバリアの詳細は、22章*書き込みバリア*を参照してください。

Direct Access テクノロジープレビュー

Red Hat Enterprise Linux 7.3 以降、ext4 および XFS のファイルシステムで **Direct Access** (DAX)がテクノロジープレビューとして利用できます。これは、アプリケーションが永続メモリーをそのアドレス空間に直接マッピングするための手段です。DAX を使用するには、システムで利用可能な永続メモリーの形式が必要です。通常は、NVDIMM (Non-Volatile Dual In-line Memory Module) の形式で、DAX に対応するファイルシステムを NVDIMM に作成する必要があります。また、ファイルシステムは **dax** マウントオプション でマウントする必要があります。これにより、dax をマウントしたファイルシステムのファイルの **mmap** が、アプリケーションのアドレス空間にストレージを直接マッピングされます。

3.3. XFS クォータ管理

XFS クォータサブシステムは、ディスク領域 (ブロック) およびファイル (inode) の使用量の制限を管理します。XFS クォータは、ユーザー、グループ、ディレクトリーレベル、またはプロジェクトレベルでこれらの項目の使用を制御または報告します。また、ユーザー、グループ、ディレクトリーまたはプロジェクトのクォータは個別に有効になりますが、グループとプロジェクトのクォータは相互に排他的であることに注意してください。

ディレクトリーまたはプロジェクトごとに管理する場合、XFS は特定のプロジェクトに関連付けられたディレクトリー階層のディスク使用量を管理します。そうすることで、XFS はプロジェクト間の組織間にまたがるグループ境界を認識します。これにより、ユーザーまたはグループのクォータを管理する際に使用できるレベルよりも、広いレベルの制御が提供されます。

XFS クォータは、特定のマウントオプションを指定して、マウント時に有効になります。各マウントオプションは **noenforce** として指定することもできます。これにより、制限を強制適用せずに使用状況のレポートが可能になります。有効なクォータマウントオプションは以下の通りです。

- **uquota**/**uqnoenforce**: ユーザークォータ
- **gquota**/**gqnoenforce**: グループクォータ
- **pquota**/**pqnoenforce**: プロジェクトクォータ

クォータを有効にすると、**xfs_quota** ツールを使用して制限を設定し、ディスク使用量を報告できます。**xfs_quota** は、デフォルトでは対話形式で、*基本モード*で実行されます。基本モードのサブコマンドは使用量を報告するだけで、すべてのユーザーが使用できます。基本的な **xfs_quota** サブコマンドには以下が含まれます。

quota *username/userID*

指定の ***username*** または numeric ***userID***の使用状況および制限を表示します

df

ブロックおよび inode の空きおよび使用済みの数を表示します。

これとは対照的に、**xfs_quota** には *エキスパートモード*もあります。このモードのサブコマンドは、制限を実際に設定することができるため、昇格した特権を持つユーザーのみが利用できます。エキスパートモードサブコマンドを対話的に使用するには、以下のコマンドを使用します。

```
# xfs_quota -x
```

エキスパートモードのサブコマンドには以下が含まれます。

report */path*

特定のファイルシステムのクォータ情報を報告します。

limit

クォータの制限を変更します。

基本モードまたはエキスパートモードのサブコマンドの完全なリストについては、サブコマンド **help** を使用します。

すべてのサブコマンドは、**-c** オプションを使用してコマンドラインから直接実行することもできます。エキスパートサブコマンドの場合は **-x** を使用します。

例3.2 サンプルクォータレポートの表示

たとえば、(**/dev/blockdevice**の) **/home** のクォータレポートのサンプルを表示するには、**xfs_quota -x -c 'report -h' /home** コマンドを使用します。これにより、以下のような出力が表示されます。

```
User quota on /home (/dev/blockdevice)
Blocks
User ID      Used   Soft   Hard Warn/Grace
---------- ---------------------------------
root            0      0      0  00 [------]
testuser   103.4G      0      0  00 [------]
...
```

ホームディレクトリーが **/home/john** のユーザー **john** に対して、inode 数のソフト制限およびハード制限をそれぞれ 500 と 700 に設定するには、以下のコマンドを使用します。

```
# xfs_quota -x -c 'limit isoft=500 ihard=700 john' /home/
```

この場合は、マウントされた xfs ファイルシステムである *mount_point* を渡します。

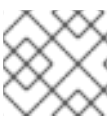
デフォルトでは、**limit** サブコマンドはターゲットをユーザーとして認識します。グループに制限を設定する場合は、（前の例のように） **-g** オプションを使用します。同様に、プロジェクトには **-p** を使用します。

ソフトブロックおよびハードブロック制限は、**isoft** または **ihard** の代わりに **bsoft** または **bhard** を使用して設定することもできます。

例3.3 ソフトブロックおよびハードブロック制限の設定

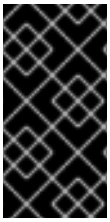
たとえば、**/target/path** ファイルシステムで **アカウンティング** をグループ化するために、ソフトブロック制限とハードブロック制限をそれぞれ 1000m と 1200m に設定するには、次のコマンドを使用します。

```
# xfs_quota -x -c 'limit -g bsoft=1000m bhard=1200m accounting' /target/path
```



注記

バイトの **bsoft** および **bhard** count コマンド。



重要

リアルタイムブロック(**rtbhard**/**rtbsoft**)は、クォータの設定時に有効な単位として **man xfs_quota** で説明されていますが、本リリースではリアルタイムサブボリュームは有効になっていません。そのため、**rtbhard** オプションおよび **rtbsoft** オプションは適用されません。

プロジェクト制限の設定

XFS ファイルシステムでは、マネージドツリーと呼ばれるファイルシステム内の個々のディレクトリー階層にクォーターを設定できます。各マネージドツリーは、**プロジェクト ID** とオプションの **プロジェクト名** によって一意に識別されます。

1. プロジェクトが制御するディレクトリーを **/etc/projects** に追加します。たとえば、以下は一意の ID が **11** の **/var/log** パスを **/etc/projects** に追加します。プロジェクト ID には、プロジェクトにマッピングされる任意の数値を指定できます。

```
# echo 11:/var/log >> /etc/projects
```

2. プロジェクト名を **/etc/projid** に追加して、プロジェクト ID をプロジェクト名にマッピングします。たとえば、以下は、前のステップで定義されたように **logfiles** というプロジェクトをプロジェクト ID **11** に関連付けます。

```
# echo logfiles:11 >> /etc/projid
```

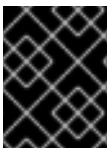
3. プロジェクトのディレクトリーを初期化します。たとえば、以下はプロジェクトディレクトリー **/var** を初期化します。

```
# xfs_quota -x -c 'project -s logfiles' /var
```

4. 初期化したディレクトリーでプロジェクトのクォータを設定します。

```
# xfs_quota -x -c 'limit -p bhard=lg logfiles' /var
```

汎用クォータ設定ツール(**クォータ**、**repquota**、および **edquota** など)を使用して XFS クォータを操作することもできます。ただし、このツールは XFS プロジェクトクォータでは使用できません。



重要

Red Hat は、他の利用可能なすべてのツールで **xfs_quota** を使用することを推奨します。

XFS クォータの設定に関する詳細は、**man xfs_quota**、man**projid (5)、および man projects (5)** を参照してください。

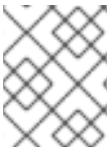
3.4. XFS ファイルシステムのサイズの拡大

xfs_growfs コマンドを使用して、マウントしたまま XFS ファイルシステムを拡張することができます。

```
# xfs_growfs /mount/point -D size
```

-D *size* オプションは指定された ***サイズ*** にファイルシステムを拡張します（ファイルシステムブロックで表現）。**xfs_growfs** は、**-D *size*** オプションを指定しないと、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。

-D *サイズ* が の XFS ファイルシステムを拡張する前に、基礎となるブロックデバイスが、後でファイルシステムを保持するのに適したサイズであることを確認してください。該当するブロックデバイスのサイズを変更する場合は、ブロックデバイスに適した方法を選択してください。



注記

XFS ファイルシステムは、マウント中にサイズを大きくすることができますが、サイズを縮小することはできません。

ファイルシステムの拡張の詳細は、**man xfs_growfs** を参照してください。

3.5. XFS ファイルシステムの修復

XFS ファイルシステムを修復するには、**xfs_repair** を使用します。

```
# xfs_repair /dev/device
```

xfs_repair ユーティリティーはスケーラビリティーが高く、inode が多数ある非常に大きなファイルシステムでも修復するように設計されています。**xfs_repair** は、他の Linux ファイルシステムとは異なり、XFS ファイルシステムが正常にアンマウントされていなくても起動時には実行されません。不完全なアンマウントが発生した場合、**xfs_repair** は単にマウント時にログを再生し、一貫したファイルシステムを確保します。

警告

xfs_repair ユーティリティーは、ダーティーログを持つ XFS ファイルシステムを修復できません。ログをクリアするには、XFS ファイルシステムをマウントおよびアンマウントします。ログが破損していて再生できない場合は、**-L** オプション（強制的にログのゼロリング）を使用してログを消去します。つまり、**xfs_repair -L /*dev*/*device*** です。これにより、さらなる破損やデータの損失が生じる可能性があることに注意してください。

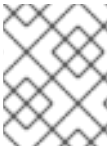
XFS ファイルシステムの修復に関する詳細は、**man xfs_repair** を参照してください。

3.6. XFS ファイルシステムの一時停止

ファイルシステムへの書き込み動作を一時停止または再開するには、以下のコマンドを使用します。

```
# xfs_freeze mount-point
```

書き込みアクティビティーを停止することで、ハードウェアベースのデバイスのスナップショットを使用して、一貫性のある状態でファイルシステムをキャプチャーできます。



注記

xfs_freeze ユーティリティーは **xfsprogs** パッケージで提供されます。これは x86_64 でのみ利用できます。

XFS ファイルシステムを一時停止 (フリーズ) するには、以下のコマンドを使用します。

```
# xfs_freeze -f /mount/point
```

XFS ファイルシステムのフリーズを解除する場合は、次のコマンドを使用します

```
# xfs_freeze -u /mount/point
```

LVM スナップショットを作成する場合、最初に **xfs_freeze** を使用してファイルシステムを一時停止する必要はありません。LVM 管理ツールが、スナップショットを取る前に XFS ファイルシステムを自動的に一時停止します。

XFS ファイルシステムのフリーズおよびアンフリーズの詳細は、**man xfs_freeze** を参照してください。

3.7. XFS ファイルシステムのバックアップおよび復元

XFS ファイルシステムのバックアップと復元には、以下のユーティリティーが含まれます。

- バックアップを作成するための **xfsdump**
- バックアップから復元するための **xfsrestore**

3.7.1. XFS バックアップおよび復元の機能

バックアップ

xfsdump ユーティリティーを使用して以下を行うことができます。

- 通常のファイルイメージへのバックアップ

  通常のファイルに書き込むことができるバックアップは 1 つだけです。

- テープドライブへのバックアップ

  xfsdump ユーティリティーを使用すると、同じテープに複数のバックアップを書き込むこともできます。バックアップは、複数のテープを分割して書き込むことができます。

  複数のファイルシステムのバックアップを 1 つのテープデバイスに作成するには、XFS バックアップがすでに含まれているテープにバックアップを書き込みます。これにより、古いバックアップに、新しいバックアップが追加されます。**xfsdump** は、デフォルトでは既存のバックアップを上書きしません。

- 増分バックアップの作成

  xfsdump ユーティリティーは *dump レベル* を使用して、他のバックアップの相対的なベースバックアップを決定します。**0** から **9** までの数字は、ダンプレベルの増加を指します。増分バックアップは、下位レベルの最後のダンプ以降に変更したファイルのみが対象となります。

- フルバックアップを実行する場合は、ファイルシステムで **レベル 0** のダンプを実行します。
- レベル 1 のダンプは、フルバックアップ後の最初の増分バックアップです。次の増分バックアップはレベル 2 になります。これは、前回のレベル 1 などのダンプ以降に変更したファイルのみが対象となります。最大でレベル 9 までが対象となります。

- ファイルを絞り込むサイズ、サブツリー、または inode のフラグを使用して、バックアップからファイルを除外

復元

xfsrestore ユーティリティーは、**xfsdump** によって生成されたバックアップからファイルシステムを復元します。**xfsrestore** ユーティリティーには 2 つのモードがあります。

- **simple** モードでは、ユーザーはレベル 0 のダンプからファイルシステム全体を復元できます。これはデフォルトのモードです。
- **cumulative** モードでは、増分バックアップ (つまりレベル 1 からレベル 9) からファイルシステムを復元できます。

各バックアップは、一意の *session ID* または *session label* で識別されます。複数のバックアップを含むテープからバックアップを復元するには、対応するセッション ID またはラベルが必要です。

バックアップから特定のファイルを抽出、追加、または削除するには、**xfsrestore** インタラクティブモードを起動します。インタラクティブモードでは、バックアップファイルを操作する一連のコマンドが提供されます。

3.7.2. XFS ファイルシステムのバックアップ

この手順では、XFS ファイルシステムのコンテンツのバックアップを、ファイルまたはテープに作成する方法を説明します。

手順3.1 XFS ファイルシステムのバックアップ

- 次のコマンドを使用して、XFS ファイルシステムのバックアップを作成します。

```
# xfsdump -l level [-L label] -f backup-destination path-to-xfs-filesystem
```

  - *level* をバックアップのダンプレベルに置き換えます。完全バックアップを実行するには **0** を、結果として生じる増分バックアップを実行するには、**1** から **9** を使用します。
  - *backup-destination* を、バックアップを保存する場所のパスに置き換えます。保存場所は、通常のファイル、テープドライブ、またはリモートテープデバイスです。たとえば、ファイルの場合は **/backup-files/Data.xfsdump**、テープドライブの場合は **/dev/st0** です。
  - *path-to-xfs-filesystem* をバックアップを作成する XFS ファイルシステムのマウントポイントに置き換えます。たとえば、**/mnt/data/** です。ファイルシステムをマウントする必要があります。
  - 複数のファイルシステムのバックアップを作成して 1 つのテープデバイスに保存する場合は、復元時にそれらを簡単に識別できるように **-L *label*** オプションを使用して、各バックアップにセッションラベルを追加します。*label* をバックアップの名前（例： **backup_data** ）に置き換えます。

例3.4 複数の XFS ファイルシステムのバックアップ

- **/boot/** ディレクトリーおよび **/data/** ディレクトリーにマウントされた XFS ファイルシステムのコンテンツのバックアップを作成し、それらをファイルとして **/backup-files/** ディレクトリーに保存するには、次のコマンドを実行します。

```
# xfsdump -l 0 -f /backup-files/boot.xfsdump /boot
# xfsdump -l 0 -f /backup-files/data.xfsdump /data
```

- 1 つのテープデバイスにある複数のファイルシステムのバックアップを作成する場合は、**-L *label*** オプションを使用して、各バックアップにセッションラベルを追加します。

```
# xfsdump -l 0 -L "backup_boot" -f /dev/st0 /boot
# xfsdump -l 0 -L "backup_data" -f /dev/st0 /data
```

関連情報

- XFS ファイルシステムのバックアップの詳細は、xfsdump(8) の man ページを参照してください。

3.7.3. バックアップからの XFS ファイルシステムの復元

この手順では、XFS ファイルシステムの内容を、ファイルまたはテープのバックアップから復元する方法を説明します。

前提条件

- 「XFS ファイルシステムのバックアップ」 の説明に従って、XFS ファイルシステムのファイルまたはテープのバックアップを作成する必要があります。

手順3.2 バックアップからの XFS ファイルシステムの復元

- バックアップを復元するコマンドは、フルバックアップから復元するか、増分バックアップから復元するか、1 つのテープデバイスから複数のバックアップを復元するかによって異なります。

```
# xfsrestore [-r] [-S session-id] [-L session-label] [-i]
           -f backup-location restoration-path
```

  - *backup-location* を、バックアップの場所に置き換えます。これは、通常のファイル、テープドライブ、またはリモートテープデバイスになります。たとえば、ファイルの場合は **/backup-files/Data.xfsdump**、テープドライブの場合は **/dev/st0** です。

  - *restoration-path* を、ファイルシステムを復元するディレクトリーへのパスに置き換えます。たとえば、**/mnt/data/** です。

  - ファイルシステムを増分 (レベル 1 からレベル 9) バックアップから復元するには、**-r** オプションを追加します。

  - 複数のバックアップを含むテープデバイスからバックアップを復元するには、**-S** オプションまたは **-L** オプションを使用してバックアップを指定します。

-S ではセッション ID でバックアップを選択できます。**-L** ではセッションラベルでバックアップを選択できます。セッション ID とセッションラベルを取得するには、**xfsrestore -I** コマンドを使用します。

session-id を、バックアップのセッション ID に置き換えます。たとえば、**b74a3586-e52e-4a4a-8775-c3334fa8ea2c** です。*session-label* を、バックアップのセッションラベルに置き換えます。たとえば、**my_backup_session_label** です。

- **xfsrestore** を対話的に使用するには、**-i** オプションを使用します。

  対話式ダイアログは、**xfsrestore** が指定のデバイスの読み取りを終了すると開始します。インタラクティブな **xfsrestore** シェルの使用可能なコマンドには、**cd**、**ls**、**add**、**delete**、および **extract** があります。コマンドの完全なリストについては、**help** コマンドを使用します。

例3.5 複数の XFS ファイルシステムの復元

XFS バックアップファイルを復元し、そのコンテンツを **/mnt/** の下のディレクトリーに保存するには、以下を実行します。

```
# xfsrestore -f /backup-files/boot.xfsdump /mnt/boot/
# xfsrestore -f /backup-files/data.xfsdump /mnt/data/
```

複数のバックアップを含むテープデバイスから復元するには、各バックアップをセッションラベルまたはセッション ID で指定します。

```
# xfsrestore -f /dev/st0 -L "backup_boot" /mnt/boot/
# xfsrestore -f /dev/st0 -S "45e9af35-efd2-4244-87bc-4762e476cbab" /mnt/data/
```

テープからバックアップを復元する際の情報メッセージ

複数のファイルシステムからのバックアップを使用してテープからバックアップを復元すると、**xfsrestore** ユーティリティーがメッセージを発行する場合があります。このメッセージは、**xfsrestore** がテープ上の各バックアップを順番に検査する際に、要求されたバックアップと一致するものが見つかったかどうかを通知します。以下に例を示します。

```
xfsrestore: preparing drive
xfsrestore: examining media file 0
xfsrestore: inventory session uuid (8590224e-3c93-469c-a311-fc8f23029b2a) does not match the media header's session uuid (7eda9f86-f1e9-4dfd-b1d4-c50467912408)
xfsrestore: examining media file 1
xfsrestore: inventory session uuid (8590224e-3c93-469c-a311-fc8f23029b2a) does not match the media header's session uuid (7eda9f86-f1e9-4dfd-b1d4-c50467912408)
[...]
```

情報メッセージは、一致するバックアップが見つかるまで継続して表示されます。

関連情報

- XFS ファイルシステムの復元の詳細は、xfsrestore(8) の man ページを参照してください。

3.8. エラー動作の設定

I/O 操作中にエラーが発生すると、XFS ドライバーは、以下の 2 つの方法のいずれかで応答します。

- 以下のいずれかになるまで再試行を続行します。
  - I/O 操作が成功するか、または
  - I/O 操作の再試行回数または時間制限を超えた場合
- エラーが永続化し、システムが停止することを考慮してください。

XFS は、現在、必要な動作を特別に設定できる以下のエラー条件を認識しています。

- **EIO**: デバイスへの書き込み試行時のエラー
- **ENOSPC**: No space left on the device
- **ENODEV**: デバイスが見つかりません

特定のハンドラーが定義されていない他のすべての考えられるエラー条件は、単一のグローバル設定を共有します。

XFS がエラーを永続的と見なす条件を、再試行の最大数と最大秒数の両方で設定できます。いずれかの条件が満たされると、XFS は再試行を停止します。

ファイルシステムのマウントを解除すると、その他の設定に関係なく、再試行をすぐにキャンセルするオプションもあります。これにより、永続的なエラーが発生した場合でも、マウント解除操作を成功できます。

3.8.1. 特定の条件および定義されていない条件の設定ファイル

エラー動作を制御する設定ファイルは、**/sys/fs/xfs/device/error/** ディレクトリーにあります。

/sys/fs/xfs/*デバイス*/error/metadata/ ディレクトリーには、特定のエラー 状態ごとにサブディレクトリーが含まれます。

- **EIO エラー状態の /sys/fs/xfs/*デバイス*/error/metadata / EIO /**
- **ENODEV エラー条件の /sys/fs/xfs/*デバイス*/error/metadata / ENODEV /**
- **ENOSPC エラー条件の /sys/fs/xfs/*デバイス*/error/metadata / ENOSPC /**

各設定ファイルには、以下の設定ファイルが含まれます。

- **/sys/fs/xfs/*device*/error/metadata/*condition*/max_retries**: XFS が操作を再試行する最大回数を制御します。
- **/sys/fs/xfs/*device*/error/metadata/*condition*/retry_timeout_seconds**: XFS が操作の再試行を停止するまでの時間制限（秒単位）

前のセクションで説明したものを除いて、他のすべての考えられるエラー状態は、これらのファイルで共通の設定を共有しています。

- **/sys/fs/xfs/*device*/error/metadata/default/max_retries**: 再試行の最大数を制御します。
- **/sys/fs/xfs/*device*/error/metadata/default/retry_timeout_seconds**: 再試行する時間制限を制御します。

3.8.2. 特定かつ未定義の条件に対するファイルシステムの動作の設定

再試行の最大数を設定するには、必要な数を **max_retries** ファイルに書き込みます。

- 特定の条件の場合:

```
# echo value > /sys/fs/xfs/device/error/metadata/condition/max_retries
```

- 未定義の条件の場合:

```
# echo value > /sys/fs/xfs/device/error/metadata/default/max_retries
```

value は、**-** 1 から可能な最大値 **int** (C 符号付き整数型)の間の数値です。これは、64 ビット Linux では **2147483647** です。

時間制限を設定するには、希望する秒数を **retry_timeout_seconds** ファイルに書き込みます。

- 特定の条件の場合:

```
# echo value > /sys/fs/xfs/device/error/metadata/condition/retry_timeout_seconds
```

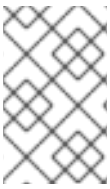
- 未定義の条件の場合:

```
# echo value > /sys/fs/xfs/device/error/metadata/default/retry_timeout_seconds
```

value は -1 から **86400** の間の数字で、これは 1 日の秒数です。

max_retries オプションと **retry_timeout_seconds** オプションの両方で、**-** 1 は永久に再試行し、**0** は即座に停止することを意味します。

device は、/**dev**/ ディレクトリーにあるデバイス名です。たとえば、**sda** などです。



注記

それぞれのエラー状態のデフォルトの動作は、エラーコンテキストによって異なります。**ENODEV** などの一部のエラーは、再試行数に関係なく致命的で回復不能であると見なされるため、デフォルト値は **0** です。

3.8.3. アンマウント動作の設定

fail_at_unmount オプションが設定されている場合、ファイルシステムはアンマウント時に他のすべてのエラー設定を上書きし、I/O 操作を再試行せずにすぐにファイルシステムを非表示にします。これにより、永続的なエラーが発生した場合でも、マウント解除操作を成功できます。

アンマウント動作を設定するには、以下を実行します。

```
# echo value > /sys/fs/xfs/device/error/fail_at_unmount
```

値 は **1** または **0** です。

- **1** は、エラーが見つかったらすぐに再試行を取り消すことを意味します。
- **0** は、**max_retries** オプションおよび **retry_timeout_seconds** オプションを尊重することを意味します。

device は、**/dev/** ディレクトリーにあるデバイス名です。たとえば、**sda** などです。

重要

fail_at_unmount オプションは、ファイルシステムのアンマウントを試行する前に必要に応じて設定する必要があります。マウント解除操作が開始されると、設定ファイルおよびディレクトリーが利用できなくなる可能性があります。

3.9. XFS ファイルシステムのその他のユーティリティー

Red Hat Enterprise Linux 7 には XFS ファイルシステムの管理用のユーティリティーが他にもあります。

xfs_fsr

マウントしている XFS ファイルシステムのデフラグを行う際に使用します。引数を指定せずに呼び出すと、**xfs_fsr** は、マウントされているすべての XFS ファイルシステム内のすべての通常ファイルをデフラグします。このユーティリティーでは、ユーザーは指定された時間にデフラグを一時停止し、後で中断したところから再開することもできます。

さらに、**xfs_fsr *</path/to/file>* のように、xfs_fsr は 1 つのファイルのみをデフラグする** こともできます。XFS はデフォルトで断片化を回避するため、Red Hat では、ファイルシステム全体を定期的にデフラグしないことを推奨しています。システム全体のデフラグは、空き領域での断片化という副作用を引き起こす可能性があります。

xfs_bmap

XFS ファイルシステム内のファイル群で使用されているディスクブロックのマップを表示します。指定したファイルによって使用されているエクステントや、該当するブロックがないファイルの領域 (ホール) をリスト表示します。

xfs_info

XFS ファイルシステムの情報を表示します。

xfs_admin

XFS ファイルシステムのパラメーターを変更します。**xfs_admin** ユーティリティーは、マウントされていないデバイスまたはファイルシステムのパラメーターのみを変更できます。

xfs_copy

XFS ファイルシステム全体のコンテンツを 1 つまたは複数のターゲットに同時にコピーします。

また、XFS ファイルシステムのデバッグや分析を行う際に以下のユーティリティーが役に立ちます。

xfs_metadump

XFS ファイルシステムのメタデータをファイルにコピーします。Red Hat は、**xfs_metadump** ユーティリティーを使用して、マウントされていないファイルシステムまたは読み取り専用でマウントされたファイルシステムをコピーすることのみをサポートします。そうしないと、生成されたダンプが破損したり、一貫性のないりする可能性があります。

xfs_mdrestore

XFS メタダンプイメージ(**xfs_metadump**を使用して生成された)をファイルシステムイメージに復元します。

xfs_db

XFS ファイルシステムをデバッグします。

これらのユーティリティーの詳細は、それぞれの **man** ページを参照してください。

3.10. EXT4 から XFS への移行

Red Hat Enterprise Linux 7.0 以降、デフォルトのファイルシステムは ext4 ではなく XFS になります。本セクションでは、XFS ファイルシステムを使用または管理する際の相違点を説明します。

ext4 ファイルシステムは、引き続き Red Hat Enterprise Linux 7 で完全にサポートされており、インストール時に選択できます。ext4 から XFS への移行は可能ですが、必須ではありません。

3.10.1. Ext3/4 と XFS の相違点

ファイルシステムの修復

Ext3/4 は、システムの起動時にユーザースペースで **e2fsck** を実行して、必要に応じてジャーナルを復元します。比較すると、XFS はマウント時にカーネル空間でジャーナルリカバリーを実行します。**fsck.xfs** シェルスクリプトが提供されますが、initscript 要件を満たすためだけに存在するため、便利なアクションは実行しません。

XFS ファイルシステムの修復またはチェックが要求されたら、**xfs_repair** コマンドを使用します。読み取り専用チェックには **-n** オプションを使用します。

xfs_repair コマンドは、ダーティーログのあるファイルシステムで動作しません。このようなファイルシステムを修復するには、最初に **アン マウント** を実行してログを再生する必要があります。ログが破損していて再生できない場合は、**-L** オプションを使用してログのゼロアウトできます。

XFS ファイルシステムのファイルシステム修復の詳細は、「XFS」 を参照してください。

メタデータエラーの動作

ext3/4 ファイルシステムでは、メタデータエラーが発生した場合の動作を設定できますが、デフォルトは単に継続します。XFS がリカバリーできないメタデータエラーに遭遇すると、ファイルシステムがシャットダウンされ、**EFSCORRUPTED** エラーが返されます。システムログには発生したエラーの詳細が含まれ、必要に応じて **xfs_repair** の実行が推奨されます。

Quotas

XFS クォータは再マウントできるオプションではありません。クォータを有効にするには、初期マウントで **-o quota** オプションを指定する必要があります。

quota パッケージの標準ツールは、基本的なクォータ管理タスク(**setquota** や **repquota**などのツール)を実行できますが、**xfs_quota** ツールは、プロジェクトクォータ管理などの XFS 固有の機能に使用できます。

quotacheck コマンドは、XFS ファイルシステムには影響しません。クォータアカウンティングが初めてオンになると、内部で自動的に **quotacheck** を実行します。XFS クォータメタデータは、ファーストクラスのジャーナル化されたメタデータオブジェクトであるため、クォータシステムは、クォータが手動でオフになるまで常に一貫しています。

ファイルシステムのサイズ変更

XFS ファイルシステムには、ファイルシステムを縮小するユーティリティーはありません。XFS ファイルシステムは、**xfs_growfs** コマンドを使用してオンラインで拡張できます。

Inode 番号

256 バイトの inode を持つ 1TB を超えるファイルシステム、または 512 バイトの inode を持つ 2TB を超えるファイルシステムでは、XFS の inode 番号が 2^32 を超える可能性があります。このような大きな inode 番号により、32 ビットの stat 呼び出しが EOVERFLOW の戻り値で失敗します。上記の問題は、デフォルトの Red Hat Enterprise Linux 7 設定 (4 つの割り当てグループでストライプ化されていない) を使用する場合に発生する可能性があります。ファイルシステムの拡張子や XFS ファイルシステムのパラメーターの変更など、カスタム設定では異なる動作が発生する場合があります。

通常、アプリケーションは、このように大きな inode 番号を正しく処理します。必要に応じて、**-o inode32** パラメーターを指定して XFS ファイルシステムをマウントし、2^32 未満の inode 番号を強制します。**inode32** を使用しても、すでに 64 ビットの数値が割り当てられている inode には影響しないことに注意してください。

重要

特定の環境に必要な場合を除き、**inode32** オプション は **使用しない** でください。**inode32** オプションは、割り当て動作を変更します。これにより、下層のディスクブロックに inode を割り当てるための領域がない場合に、ENOSPC エラーが発生する可能性があります。

投機的事前割り当て

XFS は、*投機的事前割り当て* を使用して、ファイルの書き込み時に EOF を超えてブロックを割り当てます。これにより、NFS サーバーでの同時ストリーミング書き込みワークロードによるファイルの断片化を回避します。デフォルトでは、この事前割り当てはファイルのサイズとともに増加し、"du" の出力で確認できます。投機的事前割り当てのあるファイルで 5 分間ダーティーが発生しない場合、事前割り当ては破棄されます。その時間より前に、inode がキャッシュからサイクルアウトされると、inode が回収される際に、事前割り当てが破棄されます。

投機的事前割り当てにより premature ENOSPC の問題が発生する場合は、固定事前割り当て量を **-o allocsize=*amount*** マウントオプションで指定できます。

フラグメンテーション関連のツール

フラグメント化は、割り当ての遅延や投機的な事前割り当てなどのヒューリスティックおよび動作のために、XFS ファイルシステムで重大な問題になることはめったにありません。ただし、ファイルシステムの断片化を測定したり、ファイルシステムのデフラグを行うツールは存在します。それらの使用は推奨されていません。

xfs_db frag コマンドは、すべてのファイルシステムの割り当てを 1 つの断片化番号に分割しようとします。これはパーセンテージで表されます。コマンドの出力には、その意味を理解するための十分な専門知識が必要です。たとえば、75% のフラグメント化ファクターの場合は、ファイルあたり平均 4 つのエクステントのみを意味します。このため、xfs_db のフラグメントの出力は有用とは見なされず、フラグメント化の問題を注意深く分析することが推奨されます。

警告

xfs_fsr コマンドを使用して、個々のファイルまたはファイルシステム上のすべてのファイルをデフラグできます。後者は、ファイルの局所性を破壊し、空き領域をフラグメント化する可能性があるため、特に推奨しません。

XFS と比較した ext3 および ext4 で使用されるコマンド

以下の表は、ext3 および ext4 で使用される一般的なコマンドを、XFS 固有のコマンドと比較します。

表3.1 XFS と比較した ext3 および ext4 の共通コマンド

タスク	ext3/4	XFS
ファイルシステムを作成する	**mkfs.ext4** または **mkfs.ext3**	**mkfs.xfs**
ファイルシステム検査	**e2fsck**	**xfs_repair**
ファイルシステムのサイズ変更	**resize2fs**	**xfs_growfs**
ファイルシステムのイメージを保存する	**e2image**	**xfs_metadump** および **xfs_mdrestore**
ファイルシステムのラベル付けまたはチューニングを行う	**tune2fs**	**xfs_admin**
ファイルシステムのバックアップ	**dump** および **restore**	**xfsdump** および **xfsrestore**

次の表に、XFS ファイルシステムでも機能する汎用ツールを示しますが、XFS バージョンにはより具体的な機能があるため、推奨されます。

表3.2 ext4 および XFS の一般的なツール

タスク	ext4	XFS
クォータ	**quota**	**xfs_quota**
ファイルマッピング	**filefrag**	**xfs_bmap**

リスト表示されている XFS コマンドの詳細は、3章*XFS ファイルシステム* に記載されています。詳細は、リスト化された XFS 管理ツールの man ページも参照してください。

第4章 EXT3 ファイルシステム。

ext3 ファイルシステムは、基本的に、ext2 ファイルシステムが拡張されたバージョンです。さまざまな改善点により、以下のような利点が提供されます。

可用性

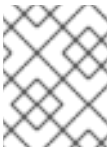
予期しない停電やシステムクラッシュ (*クリーンでないシステムシャットダウン*とも言われる) が発生すると、マシンにマウントしている各 ext2 ファイルシステムは、**e2fsck** プログラムで整合性をチェックする必要があります。これは時間を浪費するプロセスであり、大量のファイルを含む大型ボリュームでは、システムの起動時間を著しく遅らせます。このプロセスの間、そのボリュームにあるデータは使用できません。

ライブファイルシステムで **fsck -n** を実行できます。ただし、変更は行われず、部分的に書き込まれたメタデータが検出された場合、誤解を招く結果が生じる可能性があります。

スタックで LVM が使用されている場合は、ファイルシステムの LVM スナップショットを作成し、そのファイルシステムで **fsck** を実行する方法があります。

もしくは、ファイルシステムを読み込み専用で再マウントするオプションがあります。保留中のメタデータ更新 (および書き込み) は、すべて再マウントの前にディスクへ強制的に入れられます。これにより、以前に破損がない限り、ファイルシステムが一貫した状態になります。**fsck -n** を実行できるようになりました。

ext3 ファイルシステムで提供されるジャーナリングは、クリーンでないシステムシャットダウンが発生してもこの種のファイルシステムの検査が不要であることを意味します。ext3 の使用していても整合性チェックが必要になる唯一の場面は、ハードドライブの障害が発生した場合など、ごく稀なハードウェア障害のケースのみです。クリーンでないシャットダウンの発生後に ext3 ファイルシステムを復元する時間は、ファイルシステムのサイズやファイルの数量ではなく、一貫性を維持するために使用される *ジャーナル* のサイズに依存します。デフォルトのジャーナルサイズは、ハードウェアの速度に応じて、復旧するのに約1秒かかります



注記

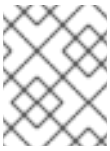
Red Hat がサポートする ext3 の唯一のジャーナリングモードは **data=ordered** （デフォルト）です。

データの整合性

ext3 ファイルシステムは、クリーンでないシステムシャットダウンが発生した際にデータの整合性が失われることを防止します。ext3 ファイルシステムにより、データが受けることのできる保護のタイプとレベルを選択できるようになります。ファイルシステムの状態に関しては、ext3 のボリュームはデフォルトで高度なレベルのデータ整合性を維持するように設定されています。

速度

一部のデータを複数回書き込みますが、ext3 のジャーナリングにより、ハードドライブのヘッドモーションが最適化されるため、ほとんどの場合、ext3 のスループットは ext2 よりも高くなります。速度を最適化するために 3 つのジャーナリングモードから選択できますが、システムに障害が発生する可能性のある状況では、モードの選択はデータの整合性がトレードオフの関係になることがあります。

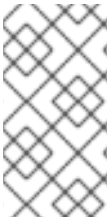


注記

Red Hat がサポートする ext3 の唯一のジャーナリングモードは **data=ordered** （デフォルト）です。

簡単なトランジション

ext2 から ext3 に簡単に移行でき、再フォーマットをせずに、堅牢なジャーナリングファイルシステムの恩恵を受けることができます。このタスクの実行方法は、「ext3 ファイルシステムへの変換」を参照してください。



注記

Red Hat Enterprise Linux 7 は、統合 extN ドライバーを提供します。これは、ext2 設定および ext3 設定を無効にして行われ、代わりにこれらのディスク上の形式に **ext4.ko** を使用します。つまり、カーネルメッセージは、使用されている ext ファイルシステムに関係なく、常に ext4 を参照します。

4.1. EXT3 ファイルシステムの作成

インストール後、ext3 ファイルシステムを新たに作成しないといけない場合があります。たとえば、システムに新しいディスクドライブを追加した時に、そのドライブにパーティション設定して ext3 ファイルシステムを使用します。

1. **mkfs.ext3** ユーティリティーを使用して、パーティションまたは LVM ボリュームを ext3 ファイルシステムでフォーマットします。

```
# mkfs.ext3 block_device
```

   - *block_device* をブロックデバイスへのパスに置き換えます。たとえば、**/dev/sdb1**、**/dev/disk/by-uuid/05e99ec8-def1-4a5e-8a9d-5945339ceb2a**、または **/dev/my-volgroup/my-lv** です。

2. **e2label** ユーティリティーを使用してファイルシステムにラベルを付けます。

```
# e2label block_device volume_label
```

UUID の設定

また、ファイルシステムに特定の UUID を設定することもできます。ファイルシステムの作成時に UUID を指定するには、**-U** オプションを使用します。

```
# mkfs.ext3 -U UUID device
```

- *UUID* は、設定する UUID に置き換えます（例： **7cd65de3-e0be-41d9-b66d-96d749c02da7** ）。
- *device* を、ext3 ファイルシステムへのパスに置き換え、UUID を追加します（例： **/dev/sda8** ）。

既存のファイルシステムの UUID を変更するには、「永続的な命名属性の変更」 を参照してください。

関連情報

- mkfs.ext3(8) の man ページ
- e2label(8) の man ページ

4.2. EXT3 ファイルシステムへの変換

tune2fs コマンドは、**ext2** ファイルシステムを **ext3** に変換します。

注記

ext2 を ext3 に変換するには、**tune2fs** を使用する前後に、常に **e2fsck** ユーティリティーを使用してファイルシステムを検査してください。ext2 から ext3 への変換を試みる前に、エラーが発生した場合に備えてすべてのファイルシステムをバックアップしてください。

さらに、Red Hat は、可能な限り ext2 から ext3 に変換するのではなく、新しい ext3 ファイルシステムを作成し、データを移行することを推奨します。

ext2 ファイルシステムを **ext3** に変換するには、root としてログインし、端末に以下のコマンドを入力します。

```
# tune2fs -j block_device
```

block_device には、変換する ext2 ファイルシステムが含まれます。

df コマンドを実行して、マウントされたファイルシステムを表示します。

4.3. EXT2 ファイルシステムへの復元

ext2 ファイルシステムに戻すには、以下の手順に従います。

分かりやすくするため、本セクションのコマンド例は、ブロックデバイスに以下の値を使用します。

/dev/mapper/VolGroup00-LogVol02

手順4.1 ext3 から ext2 に戻す

1. root としてログインし、パーティションをアンマウントして以下を入力します。

   ```
   # umount /dev/mapper/VolGroup00-LogVol02
   ```

2. 以下のコマンドを入力して、ファイルシステムの種類を ext2 に変更します。

   ```
   # tune2fs -O ^has_journal /dev/mapper/VolGroup00-LogVol02
   ```

3. 以下のコマンドを入力して、パーティションでエラーの有無を確認します。

   ```
   # e2fsck -y /dev/mapper/VolGroup00-LogVol02
   ```

4. 次に、以下を入力して ext2 ファイルシステムとしてパーティションを再度マウントします。

   ```
   # mount -t ext2 /dev/mapper/VolGroup00-LogVol02 /mount/point
   ```

   /mount/point をパーティションのマウントポイントに置き換えます。

注記

.journal ファイルがパーティションのルートレベルに存在する場合は、これを削除します。

パーティションを ext2 に永続的に変更するには、**/etc/fstab** ファイルを更新することを忘れないでください。そうしないと、起動後に元に戻されます。

第5章 EXT4 ファイルシステム

ext4 ファイルシステムは、ext3 ファイルシステムの拡張性を高めたファイルシステムです。Red Hat Enterprise Linux 7 では、最大 16 テラバイトのファイルシステムしかサポートしていなかった Red Hat Enterprise Linux 6 とは異なり、最大 16 テラバイトの個別のファイルサイズと、最大 50 テラバイトのファイルシステムをサポートすることができます。また、サブディレクトリーの数を無制限にサポートします (ext3 ファイルシステムは最大 32,000 までしかサポートしません)。ただし、リンク数が 65,000 を超えると 1 にリセットされ、増加しなくなります。bigalloc 機能は現在サポートされていません。

注記

ext3 と同様に、**fsck** を実行するには、ext4 ボリュームをアンマウントする必要があります。詳細は、*4章Ext3 ファイルシステム。* を参照してください。

主な特長

ext4 ファイルシステムはエクステントを使用します (ext2 および ext3 で使用される従来のブロックマッピングスキームとは異なります)。これにより、大きなファイルを使用する際のパフォーマンスが向上し、大きなファイルのメタデータオーバーヘッドが低減します。また、ext4 では、未使用のブロックグループと inode テーブルのセクションにそれぞれラベル付けが行なわれます。これにより、ファイルシステムの検査時にこれらを省略することができます。また、ファイルシステムの検査速度が上がるため、ファイルシステムが大きくなるほどその便宜性は顕著になります。

割り当て機能

Ext4 ファイルシステムには、以下のような割り当てスキームが備わっています。

- 永続的な事前割り当て
- 遅延割り当て
- マルチブロック割り当て
- ストライプ認識割り当て

遅延割り当てや他のパフォーマンスが最適化されるため、ext4 のディスクへのファイル書き込み動作は ext3 の場合とは異なります。ext4 では、プログラムがファイルシステムに書き込むと、プログラムが後で **fsync （）** 呼び出しを発行しない限り、ディスク上のディスク上の保証はありません。

デフォルトでは、ext3 は、**fsync （）を使用せずに、新規作成されたファイルをほぼ即時にディスクに強制し** ます。この動作により、書き込まれたデータがオンディスクにあることを確認するために **fsync （）** を使用しないプログラムのバグが妨げられました。一方、ext4 ファイルシステムは、ディスクへの変更書き込みの前に数秒間待機することが多く、書き込みを結合して再度順序付けを行うことにより、ext3 を上回るディスクパフォーマンスを実現しています。

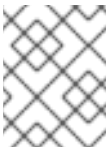
警告

ext3 とは異なり、ext4 ファイルシステムでは、トランザクションコミット時にディスクへのデータの書き込みを強制しません。このため、バッファーされた書き込みがディスクにフラッシュされるまでに時間がかかります。ファイルシステムと同様に、**fsync （）** などのデータの整合性呼び出しを使用して、データが永続ストレージに書き込まれるようにします。

Ext4 のその他の機能

ext4 ファイルシステムでは次の機能にも対応しています。

- *拡張属性* (**xattr**): これにより、システムはファイルごとに追加の名前と値のペアを関連付けることができます。
- *Quota journaling*: クラッシュ後に行なわれる時間がかかるクォータの整合性チェックが不要になります。



注記

ext4 で対応しているジャーナリングモードは **data=ordered** （デフォルト）のみです。

- *サブセカンド (一秒未満) のタイムスタンプ*- サブセカンドのタイムスタンプを指定します。

5.1. EXT4 ファイルシステムの作成

- ext4 ファイルシステムを作成するには、以下のコマンドを使用します。

```
# mkfs.ext4 block_device
```

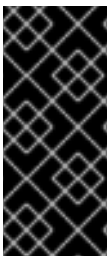
  - *block_device* をブロックデバイスへのパスに置き換えます。たとえば、**/dev/sdb1**、**/dev/disk/by-uuid/05e99ec8-def1-4a5e-8a9d-5945339ceb2a**、または **/dev/my-volgroup/my-lv** です。
  - 一般的な用途では、デフォルトのオプションが最適です。

例5.1 mkfs.ext4 コマンドの出力

以下にこのコマンドのサンプル出力を示します。出力結果には、ファイルシステムの配列や機能が表示されます。

```
~]# mkfs.ext4 /dev/sdb1
mke2fs 1.41.12 (17-May-2010)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
```

```
Stride=0 blocks, Stripe width=0 blocks
245280 inodes, 979456 blocks
48972 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=1006632960
30 block groups
32768 blocks per group, 32768 fragments per group
8176 inodes per group
Superblock backups stored on blocks:
 32768, 98304, 163840, 229376, 294912, 819200, 884736

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



重要

tune2fs を使用して、ext3 ファイルシステムで特定の ext4 機能を有効にすることができます。ただし、この方法で **tune2fs** を使用することは完全にはテストされてい **ない** ため、Red Hat Enterprise Linux 7 ではサポートされません。その結果、Red Hat は **tune2fs** を使用して変換またはマウントされた ext3 ファイルシステムの一貫したパフォーマンスと予測可能な動作を保証できません。

ストライプ化ブロックデバイス

ストライプ化されたブロックデバイス (RAID5 アレイなど) の場合は、ファイルシステムの作成時にストライプジオメトリーを指定できます。適切なストライプ配列を使用することで、ext4 ファイルシステムのパフォーマンスが大幅に改善されます。

LVM ボリュームまたは MD ボリューム上にファイルシステムを作成する場合、**mkfs.ext4** は最適なジオメトリーを選択します。オペレーティングシステムに配列情報をエクスポートするハードウェア RAID の中にも、こうした最適な配列を選択するものがあります。

ストライプジオメトリーを指定するには、以下のサブオプションを指定して **mkfs.ext4** （拡張ファイルシステムオプション）の **-E** オプションを使用します。

stride=*value*

RAID チャンクサイズを指定します。

stripe-width=*value*

RAID デバイス内のデータディスク数、または 1 ストライプ内のストライプユニット数を指定します。

両方のサブオプションについて、***値*** はファイルシステムブロック単位で指定する必要があります。たとえば、4k ブロックのファイルシステムで、64k ストライド (16 x 4096) のファイルシステムを作成する場合は、次のコマンドを使用します。

```
# mkfs.ext4 -E stride=16,stripe-width=64 /dev/block_device
```

UUID の設定

また、ファイルシステムに特定の UUID を設定することもできます。ファイルシステムの作成時に UUID を指定するには、**-U** オプションを使用します。

```
# mkfs.ext4 -U UUID device
```

- *UUID* は、設定する UUID に置き換えます（例： **7cd65de3-e0be-41d9-b66d-96d749c02da7** ）。
- *device* を、ext4 ファイルシステムへのパスに置き換え、UUID を追加します（例： **/dev/sda8** ）。

既存のファイルシステムの UUID を変更するには、「永続的な命名属性の変更」 を参照してください。

関連情報

ext4 ファイルシステムの作成に関する詳細は、以下を参照してください。

- mkfs.ext4(8) の man ページ

5.2. EXT4 ファイルシステムのマウント

ext4 ファイルシステムは、追加のオプションを使用せずにマウントできます。以下に例を示します。

```
# mount /dev/device /mount/point
```

ext4 ファイルシステムは、動作に影響を与えるマウントオプションにも対応しています。たとえば、**acl** パラメーターはアクセス制御リストを有効にし、**user_xattr** パラメーターはユーザーの拡張属性を有効にします。両方のオプションを有効にするには、以下のように **-o** を指定してそれぞれのパラメーターを使用します。

```
# mount -o acl,user_xattr /dev/device /mount/point
```

ext3 と同様に、ファイルデータでエラーが発生した場合は、**data_err=abort** オプションを使用してジャーナルを中止できます。

```
# mount -o data_err=abort /dev/device /mount/point
```

tune2fs ユーティリティーを使用すると、管理者はファイルシステムのスーパーブロックにデフォルトのマウントオプションを設定することもできます。詳細は、**man tune2fs** を参照してください。

書き込みバリア

書き込みキャッシュが有効になっているデバイスへの電力供給が停止した場合でも、ファイルシステムの整合性を確保できるようにするため、ext4 ではデフォルトで書き込みバリアを使用します。書き込みキャッシュがないデバイス、または書き込みキャッシュがバッテリーバックアップされているデバイスの場合は、以下のように **nobarrier** オプションを使用してバリアを無効にします。

```
# mount -o nobarrier /dev/device /mount/point
```

書き込みバリアの詳細は、22章 *書き込みバリア* を参照してください。

Direct Access テクノロジープレビュー

Red Hat Enterprise Linux 7.3 以降、**Direct Access** (DAX)は、ext4 および XFS ファイルシステムのテクノロジープレビューとして、アプリケーションが永続メモリーをそのアドレス空間に直接マッピングする手段を提供します。DAX を使用するには、システムで利用可能な永続メモリーの形式が必要になります。通常は、NVDIMM (Non-Volatile Dual In-line Memory Module) の形式で、DAX に対応するファイ

ルシステムを NVDIMM に作成する必要があります。また、ファイルシステムは **dax** マウントオプション でマウントする必要があります。これにより、dax をマウントしたファイルシステムのファイルの **mmap** が、アプリケーションのアドレス空間にストレージを直接マッピングされます。

5.3. EXT4 ファイルシステムのサイズ変更

ext4 ファイルシステムのサイズを大きくする前に、基礎となるブロックデバイスが将来的にファイルシステムを保持するのに十分なサイズであることを確認してください。該当するブロックデバイスのサイズを変更する場合は、ブロックデバイスに適した方法を選択してください。

ext4 ファイルシステムは、**resize2fs** コマンドを使用して、マウントしたままに拡張できます。

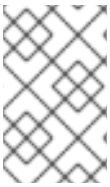
```
# resize2fs /mount/device size
```

resize2fs コマンドは、マウントされてい **ない** ext4 ファイルシステムのサイズを縮小することもできます。

```
# resize2fs /dev/device size
```

ext4 ファイルシステムのサイズを変更すると、特定のユニットを示す接尾辞が使用されていない限り、**resize2fs** ユーティリティーはファイルシステムのブロックサイズ単位でサイズを読み取ります。以下の接尾辞は、特定の単位を示しています。

- **S** - 512 バイトのセクター
- **K** - キロバイト
- **M** - メガバイト
- **G** - ギガバイト



注記

拡張時のサイズパラメーターは任意です (多くの場合は必要ありません)。**resize2fs** は、通常は論理ボリュームまたはパーティション）の利用可能な領域をすべて埋めるように、自動的に拡張します。

ext4 ファイルシステムのサイズを変更する方法は、**man resize2fs** を参照してください。

5.4. EXT2、EXT3、または EXT4 のファイルシステムのバックアップ

この手順では、ext4、ext3、または ext2 のファイルシステムのコンテンツのバックアップをファイルに作成する方法を説明します。

前提条件

- システムが長時間実行されている場合は、バックアップの前にパーティションで **e2fsck** ユーティリティーを実行します。

  ```
  # e2fsck /dev/device
  ```

手順5.1 ext2、ext3、または ext4 のファイルシステムのバックアップ

1. **/etc/fstab** ファイルの内容や **fdisk -l** コマンドの出力など、設定情報をバックアップします。これは、パーティションを復元する場合に役立ちます。

   この情報を取得するには、**sosreport** ユーティリティーまたは **sysreport** ユーティリティーを実行します。sosreport の詳細は、What is a **sosreport** and how to create one in Red Hat Enterprise Linux 4.6 and later?を参照してください。ナレッジベースの記事

2. パーティションのロールに応じて、以下を行います。

   - バックアップを作成しているパーティションがオペレーティングシステムのパーティションの場合は、システムをレスキューモードで起動します。『System Administrator's Guide』 の Booting to Rescue Mode セクションを参照してください。

   - 通常のデータパーティションのバックアップを作成する場合は、パーティションのマウントを解除します。

     マウント中にデータパーティションのバックアップを作成することは可能ですが、マウントしたデータパーティションのバックアップを作成する結果は予測できない場合があります。

     dump ユーティリティーを使用して、マウントしたファイルシステムのバックアップを作成する必要がある場合は、ファイルシステムの負荷が高くないときにバックアップを作成してください。バックアップ作成時にファイルシステムで発生しているアクティビティーが多いほど、バックアップの破損のリスクが高くなります。

3. **dump** ユーティリティーを使用して、パーティションの内容をバックアップします。

   ```
   # dump -0uf backup-file /dev/device
   ```

   backup-file を、バックアップを保存するファイルへのパスに置き換えます。*device* を、バックアップを作成する ext4 パーティションの名前に置き換えます。バックアップを、バックアップを作成しているパーティションとは別のパーティションにマウントされているディレクトリーに保存していることを確認してください。

   例5.2 複数の ext4 パーティションのバックアップの作成

   /dev/sda1、**/dev/sda2**、および **/dev/sda3** パーティションのコンテンツを **/backup-files/** ディレクトリーに保存されているバックアップファイルにバックアップするには、次のコマンドを使用します。

   ```
   # dump -0uf /backup-files/sda1.dump /dev/sda1
   # dump -0uf /backup-files/sda2.dump /dev/sda2
   # dump -0uf /backup-files/sda3.dump /dev/sda3
   ```

   リモートバックアップを作成するには、**ssh** ユーティリティーを使用するか、パスワードなしの **ssh** ログインを設定します。**ssh** およびパスワードなしのログインの詳細は、『システム管理者のガイド』 の Using the ssh Utility および Using Key-based Authentication セクションを参照してください。

   たとえば、**ssh** を使用する場合は、以下のようになります。

   例5.3 sshを使用したリモートバックアップの実行

   ```
   # dump -0u -f - /dev/device | ssh root@remoteserver.example.com dd of=backup-file
   ```

> 標準のリダイレクトを使用する場合は、**-f** オプションを個別に渡す必要があることに注意してください。

関連情報

- 詳細は、dump(8) の man ページを参照してください。

5.5. EXT2、EXT3、または EXT4 のファイルシステムの復元

この手順では、ファイルバックアップから ext4、ext3、または ext2 のファイルシステムを復元する方法を説明します。

前提条件

- 「ext2、ext3、または ext4 のファイルシステムのバックアップ」 の説明に従って、パーティションとそのメタデータのバックアップを作成する必要があります。

手順5.2 ext2、ext3、または ext4 のファイルシステムの復元

1. オペレーティングシステムパーティションを復元する場合は、システムをレスキューモードで起動します。『System Administrator's Guide』 の Booting to Rescue Mode セクションを参照してください。

   この手順は、通常のデータパーティションには不要です。

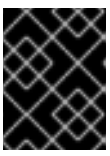
2. **fdisk** ユーティリティーまたは **parted** ユーティリティーを使用して、復元するパーティションを再構築します。

   パーティションが存在しなくなった場合は、再作成します。新しいパーティションは、復元したデータを含めるのに十分な大きさである必要があります。開始番号と終了番号を正しく取得することが重要です。これらは、バックアップ時に **fdisk** ユーティリティーから取得したパーティションの開始セクター番号と終了セクター番号です。

   パーティションの変更に関する詳細は、13章*Partitions* を参照してください。

3. **mkfs** ユーティリティーを使用して、宛先パーティションをフォーマットします。

   ```
   # mkfs.ext4 /dev/device
   ```

   重要

   バックアップファイルを保存するパーティションをフォーマット**しない**でください。

4. 新しいパーティションを作成した場合は、**/etc/fstab** ファイルのエントリーと一致するように、すべてのパーティションにラベルを付け直します。

   ```
   # e2label /dev/device label
   ```

5. 一時的なマウントポイントを作成し、そこにパーティションをマウントします。

```
# mkdir /mnt/device
# mount -t ext4 /dev/device /mnt/device
```

6. マウントしたパーティションのバックアップからデータを復元します。

```
# cd /mnt/device
# restore -rf device-backup-file
```

リモートマシンに復元するか、リモートホストに保存されているバックアップファイルから復元する場合は、**ssh** ユーティリティーを使用できます。**ssh** の詳細は、『システム管理者のガイドの Using the ssh Utility セクションを参照して』ください。

以下のコマンドには、パスワードを使用しないログインを設定する必要があることに注意してください。パスワードなしの **ssh** ログインの設定に関する詳細は、『システム管理者のガイド』の キーベース認証の使用 セクションを参照してください。

- 同じマシンに保存されているバックアップファイルから、リモートマシンのパーティションを復元する場合は、次のコマンドを実行します。

```
# ssh remote-address "cd /mnt/device && cat backup-file | /usr/sbin/restore -r -f -"
```

- 別のリモートマシンに保存されているバックアップファイルから、リモートマシンのパーティションを復元するには、次のコマンドを実行します。

```
# ssh remote-machine-1 "cd /mnt/device && RSH=/usr/bin/ssh /usr/sbin/restore -rf
remote-machine-2:backup-file"
```

7. 再起動します。

```
# systemctl reboot
```

例5.4 複数の ext4 パーティションの復元

/dev/sda1 パーティション、**/dev/sda2** パーティション、および **/dev/sda3** パーティションを 例5.2「複数の ext4 パーティションのバックアップの作成」 から復元するには、以下のコマンドを実行します。

1. **fdisk** コマンドを使用して、復元するパーティションを再構築します。
2. 宛先パーティションをフォーマットします。

```
# mkfs.ext4 /dev/sda1
# mkfs.ext4 /dev/sda2
# mkfs.ext4 /dev/sda3
```

3. **/etc/fstab** ファイルと一致するように、すべてのパーティションにラベルを付け直します。

```
# e2label /dev/sda1 Boot1
# e2label /dev/sda2 Root
# e2label /dev/sda3 Data
```

4. 作業ディレクトリーを準備します。

新しいパーティションをマウントします。

```
# mkdir /mnt/sda1
# mount -t ext4 /dev/sda1 /mnt/sda1
# mkdir /mnt/sda2
# mount -t ext4 /dev/sda2 /mnt/sda2
# mkdir /mnt/sda3
# mount -t ext4 /dev/sda3 /mnt/sda3
```

バックアップファイルを含むパーティションをマウントします。

```
# mkdir /backup-files
# mount -t ext4 /dev/sda6 /backup-files
```

5. バックアップから、マウントしたパーティションにデータを復元します。

```
# cd /mnt/sda1
# restore -rf /backup-files/sda1.dump
# cd /mnt/sda2
# restore -rf /backup-files/sda2.dump
# cd /mnt/sda3
# restore -rf /backup-files/sda3.dump
```

6. 再起動します。

```
# systemctl reboot
```

関連情報

- 詳細は、restore(8) の man ページを参照してください。

5.6. EXT4 ファイルシステムのその他のユーティリティー

Red Hat Enterprise Linux 7 には、他にも ext4 ファイルシステム管理ユーティリティーがあります。

e2fsck

ext4 ファイルシステムの修復時に使用します。このツールは、ext4 ディスク構造の更新により、ext3 よりも効率的に ext4 ファイルシステムを検査および修復します。

e2label

ext4 ファイルシステムのラベル変更を行います。このツールは ext2 および ext3 のファイルシステムでも動作します。

quota

ext4 ファイルシステムで、ユーザーおよびグループごとのディスク領域 (ブロック) やファイル (inode) の使用量を制御し、それを報告します。クォータの使用についての詳細は、**man quota** および 「ディスククォータの設定」 を参照してください。

fsfreeze

ファイルシステムへのアクセスを一時停止するには、**# fsfreeze -f *mount-point*** コマンドを使用してフリーズし、**# fsfreeze -u *mount-point*** を実行してフリーズを解除します。これにより、ファイルシステムへのアクセスが停止し、ディスクに安定したイメージが作成されます。

注記

デバイスマッパードライブに **fsfreeze** を使用する必要はありません。

詳細は、**fsfreeze (8)** の man ページを参照してください。

「ext4 ファイルシステムのマウント」 で説明したように、**tune2fs** ユーティリティーは、ext2、ext3、および ext4 ファイルシステムの設定可能なファイルシステムパラメーターを調整することもできます。また、ext4 ファイルシステムのデバッグや分析を行う際には次のツールが役に立ちます。

debugfs

ext2、ext3、ext4 の各ファイルシステムのデバッグを行います。

e2image

ext2、ext3、または ext4 の重要なファイルシステムメタデータをファイルに保存します。

これらのユーティリティーの詳細は、それぞれの **man** ページを参照してください。

第6章 BTRFS (テクノロジープレビュー)

注記

Btrfs は、Red Hat Enterprise Linux 7 ではテクノロジープレビューとして利用できますが、Red Hat Enterprise Linux 7.4 リリース以降では非推奨になりました。Red Hat Enterprise Linux の将来のメジャーリリースで削除される予定です。

詳細は、Red Hat Enterprise Linux 7.4 リリースノートの 非推奨の機能 を参照してください。

Btrfs は次世代 Linux ファイルシステムで、高度な管理、信頼性、および拡張性機能を提供します。これは、スナップショット、圧縮、統合デバイス管理を提供する点でユニークです。

6.1. BTRFS ファイルシステムの作成

基本的な btrfs ファイルシステムを作成するには、次のコマンドを使用します。

```
# mkfs.btrfs /dev/device
```

デバイスが追加された btrfs ファイルシステムの作成、およびメタデータとデータのマルチデバイスプロファイルの指定に関する詳細は、「複数のデバイスの統合ボリューム管理」 を参照してください。

6.2. BTRFS ファイルシステムのマウント

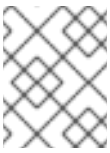
btrfs ファイルシステムにデバイスをマウントするには、次のコマンドを使用します。

```
# mount /dev/device /mount-point
```

その他の便利なマウントオプションは以下のとおりです。

device=/dev/name

このオプションを mount コマンドに追加すると、btrfs は、指定されたデバイスで btrfs ボリュームをスキャンするように指示されます。これは、btrfs 以外のデバイスをマウントしようとするとマウントが失敗するため、マウントが成功することを確認するために使用されます。



注記

これは、すべてのデバイスがファイルシステムに追加されることを意味するのではなく、それらをスキャンするだけです。

max_inline=*number*

このオプションを使用して、メタデータ B-tree リーフ内のデータのインライン化に使用できる領域の最大量 (バイト単位) を設定します。デフォルトは 8192 バイトです。4k ページの場合は、リーフに収める必要があるヘッダーが追加されているため、サイズは 3900 バイトに制限されています。

alloc_start=*number*

このオプションを使用して、ディスク割り当ての開始位置を設定します。

thread_pool=*number*

このオプションを使用して、割り当てられたワーカースレッドの数を割り当てます。

discard

このオプションは、空きブロックで discard/TRIM を有効にする場合に使用します。

noacl

このオプションを使用して、ACL の使用を無効にします。

space_cache

このオプションを使用して空き領域データをディスクに保存し、ブロックグループのキャッシュを高速化します。これは永続的な変更であり、古いカーネルで安全に起動できます。

nospace_cache

上記の **space_cache** を無効にするには、このオプションを使用します。

clear_cache

このオプションを使用して、マウント中にすべての空き領域キャッシュをクリアします。これは安全なオプションですが、スペースキャッシュの再構築がトリガーされます。そのため、再構築プロセスを終了させるために、ファイルシステムをマウントしたままにしておきます。このマウントオプションは、空き領域に問題が明らかになった後でのみ、一度使用することを目的としています。

enospc_debug

このオプションは、"no space left" 問題をデバッグするために使用します。

recovery

マウント時に自動リカバリーを有効にするには、このオプションを使用します。

6.3. BTRFS ファイルシステムのサイズの変更

btrfs ファイルシステムのサイズを変更することはできませんが、使用する各デバイスのサイズを変更することはできます。使用中のデバイスが 1 つしかない場合は、ファイルシステムのサイズを変更する場合と同じように機能します。複数のデバイスを使用中の場合は、手動でサイズを変更して、必要なサイズにする必要があります。

注記

単位サイズは大文字と小文字を区別せず、GiB の **G** または **g** の両方を受け入れます。

このコマンドは、テラバイトの場合は **t** を、ペタバイトの場合は **p** を受け入れません。**k**、**m**、および **g** のみを受け入れます。

btrfs ファイルシステムの拡張

1 つのデバイスでファイルシステムを拡大するには、次のコマンドを使用します。

```
# btrfs filesystem resize amount /mount-point
```

以下に例を示します。

```
# btrfs filesystem resize +200M /btrfssingle
Resize '/btrfssingle' of '+200M'
```

マルチデバイスファイルシステムを拡大する場合は、拡大するデバイスを指定する必要があります。最初に、指定したマウントポイントに btrfs ファイルシステムがあるすべてのデバイスを表示します。

```
# btrfs filesystem show /mount-point
```

以下に例を示します。

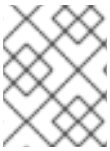
```
# btrfs filesystem show /btrfstest
Label: none  uuid: 755b41b7-7a20-4a24-abb3-45fdbed1ab39
 Total devices 4 FS bytes used 192.00KiB
 devid    1 size 1.00GiB used 224.75MiB path /dev/vdc
 devid    2 size 524.00MiB used 204.75MiB path /dev/vdd
 devid    3 size 1.00GiB used 8.00MiB path /dev/vde
 devid    4 size 1.00GiB used 8.00MiB path /dev/vdf

Btrfs v3.16.2
```

拡大するデバイスの **devid** を特定したら、次のコマンドを使用します。

```
# btrfs filesystem resize devid:amount /mount-point
```

以下に例を示します。

```
# btrfs filesystem resize 2:+200M /btrfstest
Resize '/btrfstest/' of '2:+200M'
```



注記

この *量* は、指定した数ではなく **max** にすることもできます。これにより、デバイスに残っている空き領域がすべて使用されます。

btrfs ファイルシステムの縮小

1 つのデバイスでファイルシステムを縮小するには、次のコマンドを使用します。

```
# btrfs filesystem resize amount /mount-point
```

以下に例を示します。

```
# btrfs filesystem resize -200M /btrfssingle
Resize '/btrfssingle' of '-200M'
```

マルチデバイスファイルシステムを縮小するには、縮小するデバイスを指定する必要があります。最初に、指定したマウントポイントに btrfs ファイルシステムがあるすべてのデバイスを表示します。

```
# btrfs filesystem show /mount-point
```

以下に例を示します。

```
# btrfs filesystem show /btrfstest
Label: none  uuid: 755b41b7-7a20-4a24-abb3-45fdbed1ab39
 Total devices 4 FS bytes used 192.00KiB
 devid    1 size 1.00GiB used 224.75MiB path /dev/vdc
 devid    2 size 524.00MiB used 204.75MiB path /dev/vdd
 devid    3 size 1.00GiB used 8.00MiB path /dev/vde
 devid    4 size 1.00GiB used 8.00MiB path /dev/vdf

Btrfs v3.16.2
```

縮小するデバイスの **devid** を特定したら、次のコマンドを使用します。

```
# btrfs filesystem resize devid:amount /mount-point
```

以下に例を示します。

```
# btrfs filesystem resize 2:-200M /btrfstest
Resize '/btrfstest' of '2:-200M'
```

ファイルシステムのサイズの設定

1 つのデバイスでファイルシステムのサイズを指定するには、次のコマンドを実行します。

```
# btrfs filesystem resize amount /mount-point
```

以下に例を示します。

```
# btrfs filesystem resize 700M /btrfssingle
Resize '/btrfssingle' of '700M'
```

マルチデバイスファイルシステムのファイルシステムサイズを設定するには、変更するデバイスを指定する必要があります。最初に、指定のマウントポイントに btrfs ファイルシステムを持つすべてのデバイスを表示します。

```
# btrfs filesystem show /mount-point
```

以下に例を示します。

```
# btrfs filesystem show /btrfstest
Label: none  uuid: 755b41b7-7a20-4a24-abb3-45fdbed1ab39
 Total devices 4 FS bytes used 192.00KiB
 devid    1 size 1.00GiB used 224.75MiB path /dev/vdc
 devid    2 size 724.00MiB used 204.75MiB path /dev/vdd
 devid    3 size 1.00GiB used 8.00MiB path /dev/vde
 devid    4 size 1.00GiB used 8.00MiB path /dev/vdf

Btrfs v3.16.2
```

変更するデバイスの **devid** を特定したら、次のコマンドを使用します。

```
# btrfs filesystem resize devid:amount /mount-point
```

以下に例を示します。

```
# btrfs filesystem resize 2:300M /btrfstest
Resize '/btrfstest' of '2:300M'
```

6.4. 複数のデバイスの統合ボリューム管理

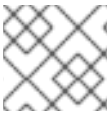
多くのデバイスでは、その上に btrfs ファイルシステムを作成できます。また、ファイルシステムの作成後にデバイスを追加することもできます。デフォルトでは、メタデータは 2 つのデバイス間でミラーリングされ、データは存在するすべてのデバイス間でストライプ化されますが、1 つのデバイスのみが存在する場合、メタデータはそのデバイス上で複製されます。

6.4.1. 複数のデバイスを使用したファイルシステムの作成

mkfs.btrfs コマンドは、「btrfs ファイルシステムの作成」 で説明され、データの場合は **-d** オプション、メタデータの場合は **-m** を受け入れます。有効な指定は以下のとおりです。

- **raid0**
- **raid1**
- **raid10**
- **dup**
- **single**

-m single オプションは、メタデータの重複を行わないように指示します。これは、ハードウェア RAID を使用する場合に望ましい場合があります。



注記

RAID 10 では、正しく実行するために 4 つ以上のデバイスが必要です。

例6.1 RAID 10 btrfs ファイルシステムの作成

4 つのデバイスにファイルシステムを作成します (メタデータミラーリング、データのストライプ化)。

```
# mkfs.btrfs /dev/device1 /dev/device2 /dev/device3 /dev/device4
```

ミラーリングせずにメタデータをストライプ化します。

```
# mkfs.btrfs -m raid0 /dev/device1 /dev/device2
```

データとメタデータの両方に raid10 を使用します。

```
# mkfs.btrfs -m raid10 -d raid10 /dev/device1 /dev/device2 /dev/device3 /dev/device4
```

1 つのドライブで、メタデータを複製しないでください。

```
# mkfs.btrfs -m single /dev/device
```

ドライブのサイズが異なる場合は、**single** オプションを使用して、各ドライブの全容量を使用します。

```
# mkfs.btrfs -d single /dev/device1 /dev/device2 /dev/device3
```

作成済みのマルチデバイスファイルシステムに、新しいデバイスを追加するには、次のコマンドを使用します。

```
# btrfs device add /dev/device1 /mount-point
```

btrfs モジュールを再起動または再読み込みした後、**btrfs device scan** コマンドを使用して、すべてのマルチデバイスファイルシステムを検出します。詳細は、「btrfs デバイスのスキャン」 を参照してください。

6.4.2. btrfs デバイスのスキャン

btrfs device scan を使用して、**/dev** の下にあるすべてのブロックデバイスをスキャンし、btrfs ボリュームをプローブします。ファイルシステムで複数のデバイスを実行している場合は、btrfs モジュールを読み込んだ後に実行する必要があります。

すべてのデバイスをスキャンするには、次のコマンドを使用します。

```
# btrfs device scan
```

1 つのデバイスをスキャンするには、以下のコマンドを使用します。

```
# btrfs device scan /dev/device
```

6.4.3. btrfs ファイルシステムへの新しいデバイスの追加

btrfs filesystem show コマンドを使用して、すべての btrfs ファイルシステムと、それらに含まれるデバイスを一覧表示します。

btrfs device add コマンドは、マウントされたファイルシステムに新しいデバイスを追加するために使用されます。

btrfs filesystem balance コマンドは、割り当てられたエクステントを既存デバイスすべてに分散（取り除く）します。

新規デバイスを追加するには、以下のコマンドを一緒に実行します。

例6.2 btrfs ファイルシステムへの新しいデバイスの追加

最初に、btrfs ファイルシステムを作成してマウントします。btrfs ファイルシステムの作成に関する詳細は 「btrfs ファイルシステムの作成」 を、btrfs ファイルシステムのマウント方法に関する詳細は 「btrfs ファイルシステムのマウント」 を参照してください。

```
# mkfs.btrfs /dev/device1
# mount /dev/device1
```

次に、マウントした btrfs ファイルシステムに、2 番目のデバイスを追加します。

```
# btrfs device add /dev/device2 /mount-point
```

これらのデバイスのメタデータとデータは、引き続き **/dev/*device1*** にのみ保存されます。すべてのデバイスに分散するには、バランスを調整する必要があります。

```
# btrfs filesystem balance /mount-point
```

ファイルシステムのバランスを調整するには、ファイルシステムのデータとメタデータをすべて読み込み、新しいデバイス全体でそれを書き換えるため、しばらく時間がかかります。

6.4.4. btrfs ファイルシステムの変換

非 RAID ファイルシステムを RAID に変換するには、デバイスを追加し、チャンク割り当てプロファイルを変更するバランスフィルターを実行します。

例6.3 btrfs ファイルシステムの変換

既存の単一デバイスシステム（この場合は **/dev/sdb1** ）を 2 つのデバイス(raid1 システム)に変換して、単一のディスク障害から保護するには、次のコマンドを使用します。

```
# mount /dev/sdb1 /mnt
# btrfs device add /dev/sdc1 /mnt
# btrfs balance start -dconvert=raid1 -mconvert=raid1 /mnt
```

重要

メタデータがシングルデバイスのデフォルトから変換されない場合は、DUP のままになります。これは、ブロックのコピーが別々のデバイスにあることを保証するものではありません。データが変換されていない場合、冗長コピーはありません。

6.4.5. btrfs デバイスの削除

btrfs device delete コマンドを使用して、オンラインデバイスを削除します。安全に削除するために、使用中のエクステントをファイルシステム内の他のデバイスに再配布します。

例6.4 btrfs ファイルシステム上のデバイスの削除

まず、いくつかの btrfs ファイルシステムを作成してマウントします。

```
# mkfs.btrfs /dev/sdb /dev/sdc /dev/sdd /dev/sde
# mount /dev/sdb /mnt
```

ファイルシステムに一部のデータを追加します。

最後に、必要なデバイスを削除します。

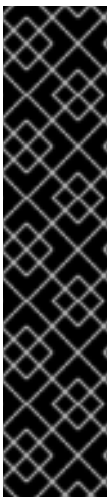
```
# btrfs device delete /dev/sdc /mnt
```

6.4.6. btrfs ファイルシステムでの障害が発生したデバイスの置き換え

スーパーブロックを引き続き読み取ることができる場合は、障害が発生したデバイスを削除するために「btrfs デバイスの削除」 を使用できます。ただし、デバイスが見つからないか、スーパーブロックが破損している場合は、ファイルシステムを劣化モードでマウントする必要があります。

```
# mkfs.btrfs -m raid1 /dev/sdb /dev/sdc /dev/sdd /dev/sde


  ssd is destroyed or removed, use -o degraded to force the mount
  to ignore missing devices


# mount -o degraded /dev/sdb /mnt


  'missing' is a special device name


# btrfs device delete missing /mnt
```

コマンド **btrfs device delete missing** は、ファイルシステムのメタデータで記述されているが、ファイルシステムがマウントされたときには存在しない最初のデバイスを削除します。



重要

見つからないデバイスを含め、特定の RAID レイアウトに必要なデバイスの最小数を下回ることは不可能です。障害が発生したデバイスを削除するために、新しいデバイスを追加する必要がある場合があります。

たとえば、2 つのデバイスを持つ raid1 レイアウトの場合、デバイスに障害が発生した場合は、以下を行います。

1. 動作が低下したモードでのマウント
2. 新しいデバイスの追加
3. 不足しているデバイスの削除

6.4.7. /etc/fstabでの btrfs ファイルシステムの登録

initrd がない場合や、btrfs デバイススキャンを実行していない場合は、ファイルシステムのすべてのデバイスを明示的に mount コマンドに渡すことで、マルチボリューム **btrfs** ファイルシステムを **マウント** できます。

例6.5 /etc/fstab エントリーの例

適切な **/etc/fstab** エントリーの例を以下に示します。

```
/dev/sdb    /mnt    btrfs    device=/dev/sdb,device=/dev/sdc,device=/dev/sdd,device=/dev/sde    0
```

UUID (Universally Unique Identifier) の使用も機能し、デバイスパスを使用するよりも安定していることに注意してください。

6.5. SSD の最適化

btrfs ファイルシステムを使用すると SSD を最適化できます。これには 2 つの方法があります。

最初の方法は、指定された単一のデバイスの **/sys/block/ デバイスの /queue/rotational がゼロである場合に、mkfs.btrfs が単一の *デバイス*** でメタデータの重複をオフにすることです。これは、コマンドラインで **-m single** を指定するのと同じです。これは、**-m dup** オプションを指定することで上書きし、メタデータを複製できます。SSD ファームウェアは両方のコピーを失う可能性があるため、複製は必要ありません。これはスペースを浪費し、パフォーマンスが犠牲になります。

2 つ目の方法は、SSD マウントオプションのグループ(**ssd**、**nossd**、および **ssd_spread**)を使用することです。

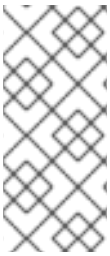
ssd オプションはいくつかのことを行います。

- より大きなメタデータのクラスターの割り当てを許可します。
- 可能な場合は、より順序立ててデータを割り当てます。
- 鍵とブロック順序に一致するように btree リーフの書き換えを無効にします。
- 複数のプロセスをバッチ処理せずにログフラグメントをコミットします。

注記

ssd マウントオプションは、ssd オプションのみを有効にします。**nossd** オプションを使用して無効にします。

一部の SSD は、ブロック番号を頻繁に再利用する場合に最高のパフォーマンスを発揮しますが、他の SSD は、クラスタリングが未使用スペースの大きなチャンクを厳密に割り当てる場合にはるかに優れたパフォーマンスを発揮します。デフォルトでは、**mount -o ssd** は、複数の空きブロックが混在している可能性のあるブロックのグループを検出します。**mount -o ssd_spread** コマンドにより、割り当てられたブロックが混在しないようにします。これにより、ローエンド SSD のパフォーマンスが向上します。



注記

ssd_spread オプションは、ssd オプションと **ssd _spread** オプションの両方を有効にします。**nossd** を使用して、両方のオプションを無効にします。

ssd オプションが指定されておらず、いずれのデバイスも回転していない場合、**ssd_spread** オプションは自動的に設定されません。

SSD ファームウェアとアプリケーションの負荷の組み合わせはそれぞれ異なるため、これらのオプションはすべて、特定のビルドでテストして、それらの使用によってパフォーマンスが向上または低下するかどうかを確認する必要があります。

6.6. BTRFS リファレンス

man ページの **btrfs (8)** は、重要なすべての管理コマンドを説明します。特に以下のものが含まれます。

- スナップショットを管理するためのすべてのサブボリュームコマンド。
- **デバイス** を管理するデバイスコマンド。

- **scrub** コマンド、**balance** コマンド、および **defragment** コマンド。

man ページの **mkfs.btrfs (8)には、btrfs** ファイルシステムの作成に関する情報と、そのファイルシステムに関するすべてのオプションが記載されています。

btrfs システムの fsck の詳細は、man ページの btrfsck (8) を参照してください。

第7章 GLOBAL FILE SYSTEM 2

Red Hat Global File System 2 (GFS2) はネイティブのファイルシステムで、直接 Linux カーネルファイルシステムのインターフェイスと相互作用します (VFS 層)。クラスターファイルシステムとして実装すると、GFS2 は分散メタデータと複数のジャーナルを採用します。

GFS2 は、理論的には 8 エクサバイトのファイルシステムに対応できる 64 ビットアーキテクチャーに基づいています。ただし、現在対応している GFS2 ファイルシステムの最大サイズは 100 TB です。システムで 100 TB を超える GFS2 ファイルシステムが必要な場合は、Red Hat サービスの担当者にお問い合わせください。

ファイルシステムのサイズを決定する際には、その復旧のニーズを考慮してください。非常に大きなファイルシステムで **fsck** コマンドを実行すると、時間がかかり、大量のメモリーを消費する可能性があります。また、ディスクまたはディスクサブシステムに障害が発生した場合、復旧時間はバックアップメディアの速度によって制限されます。

Red Hat Cluster Suite で設定する場合、Red Hat GFS2 ノードは Red Hat Cluster Suite 設定および管理ツールを使用して、設定および管理できます。続いて、Red Hat GFS2 により、Red Hat クラスター内の GFS2 ノード群でデータを共有でき、このノード群全体で整合性のあるファイルシステム名前空間の単一ビューが提供されます。これにより、異なるノードにあるプロセスで GFS2 ファイルを共有できるようになります。これは、同じノードにあるプロセスでローカルファイルシステム上のファイルを共有する場合と同様で、両者にはっきり識別できる違いはありません。Red Hat Cluster Suite の詳細は、Red Hat の 『Cluster Administration』 ガイドを参照してください。

GFS2 はリニアボリュームまたはミラーボリュームとなる論理ボリューム (LVM で作成) 上に作成してください。Red Hat Cluster Suite で LVM で作成した論理ボリュームは CLVM （クラスター全体の LVM の実装）で管理され、CLVM デーモン **clvmd** によって有効化され、Red Hat Cluster Suite クラスターで実行されます。このデーモンにより、LVM2 を使用してクラスター全体で論理ボリュームの管理が可能となり、クラスター内のすべてのノードで論理ボリュームを共有できるようになります。論理ボリュームマネージャーの詳細は、Red Hat のLogical Volume Manager Administration ガイドを参照してください。

gfs2.ko カーネルモジュールは GFS2 ファイルシステムを実装し、GFS2 クラスターノードにロードされます。

クラスター化ストレージおよび非クラスター化ストレージでの GFS2 ファイルシステムの作成および設定に関する包括的な情報は、Red Hat のGlobal File System 2 ガイドを参照してください。

第8章 NETWORK FILE SYSTEM (NFS)

ネットワークファイルシステム (*NFS*) を利用すると、リモートのホストがネットワーク経由でファイルシステムをマウントし、そのファイルシステムを、ローカルにマウントしているファイルシステムと同じように操作できるようになります。また、システム管理者は、リソースをネットワーク上の中央サーバーに統合することができるようになります。

この章では、基本的な NFS の概念と補足的な情報に焦点を絞って説明します。

8.1. NFS の概要

現在、Red Hat Enterprise Linux には、NFS のメジャーバージョンが 2 つ含まれています。

- NFS バージョン 3 (NFSv3) は、安全な非同期書き込みをサポートし、以前の NFSv2 よりもエラー処理で堅牢です。また、64 ビットのファイルサイズとオフセットにも対応しているため、クライアントは 2GB 以上のファイルデータにアクセスできます。

- NFS バージョン 4 (NFSv4)はファイアウォールやインターネットを介して動作し、**rpcbind** サービスが不要になり、ACL に対応し、ステートフルな操作を利用します。

Red Hat Enterprise Linux 7.4 リリース以降、Red Hat Enterprise Linux は NFS バージョン 4.2 (NFSv4.2) に完全に対応しています。

以下は、Red Hat Enterprise Linux における NFSv4.2 の機能です。

- スパースファイル: ファイルの領域の効率を検証し、プレースホルダーがストレージの効率を向上できるようにします。これは、1 つ以上のホールがあるファイルです。ホールは、ゼロのみで設定される、割り当てられていないデータブロックまたは初期化されていないデータブロックです。**lseek()** NFSv4.2 での 操作は、 **seek_hole()** および **seek_data()** に対応しています。これにより、アプリケーションはスパースファイルのホールの場所をマッピングできます。

- 領域の予約: ストレージサーバーが空き領域を予約することを許可します。これにより、サーバーで領域が不足することがなくなります。NFSv4.2 は、領域を予約するための **allocate()** 操作、領域の予約を解除する **deallocate()** 操作、およびファイル内の領域の事前割り当てまたは割り当て解除を行う **fallocate()** 操作に対応します。

- ラベル付き NFS: データアクセス権を強制し、NFS ファイルシステム上の個々のファイルに対して、クライアントとサーバーとの間の SELinux ラベルを有効にします。

- レイアウトの機能拡張：NFSv4.2 は、クライアントがレイアウトとの通信についてメタデータサーバーに通知するために使用できる新しい操作 **layoutstats()** を提供します。

7.4 より前のバージョンの Red Hat Enterprise Linux は、バージョン 4.1 までの NFS に対応します。

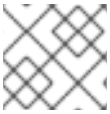
NFSv4.1 の機能は次のとおりです。

- ネットワークのパフォーマンスとセキュリティーを向上させ、Parallel NFS (pNFS) に対するクライアント側のサポートも追加されました。

- コールバックに個別の TCP 接続が必要なくなり、クライアントと通信できない場合でも、NFS サーバーは委任を許可できるようになりました。たとえば、NAT やファイアウォールが干渉する場合です。

- 応答が失われ、操作が 2 回送信された場合に特定の操作が不正確な結果を返すことがあるという以前の問題を防ぐために、1 回限りのセマンティクスを提供します (再起動操作を除く)。

NFS クライアントは、デフォルトで NFSv4.1 を使用してマウントを試行し、サーバーが NFSv4.1 に対応していない場合は NFSv4.0 にフォールバックします。サーバーが NFSv4.0 に対応していない場合、マウントは後で NFSv3 に戻ります。



注記

NFS バージョン 2 (NFSv2) は、Red Hat のサポート対象外になりました。

NFS の全バージョンで、IP ネットワーク経由で実行する *Transmission Control Protocol* (*TCP*) を使用することができ、NFSv4 の場合は TCP が必須になります。NFSv3 は、IP ネットワークで実行している *User Datagram Protocol* (UDP) を使用して、クライアントとサーバー間のステートレスなネットワーク接続を提供することができます。

UDP で NFSv3 を使用する場合は、(通常の条件下で) ステートレス UDP 接続のプロトコルオーバーヘッドが TCP より少なくなります。つまり、クリーンで適度なトラフィックのネットワーク上では、UDP の方がパフォーマンスがよくなります。ただし、UDP はステートレスのため、予期しないサーバーダウンなどが発生すると、UDP クライアントはサーバーの要求でネットワークを飽和させ続けます。また、UDP の場合にフレームがなくなると、RPC 要求全体を再転送しなければならなくなります。一方、TCP の場合、再送信が必要なのは失ったフレームのみになります。こうした理由から NFS サーバーへの接続には TCP プロトコルが推奨されます。

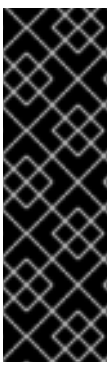
NFSv4 プロトコルには、マウントとロックのプロトコルが組み込まれています。サーバーは、既知の TCP ポート 2049 もリッスンします。そのため、NFSv4 は **rpcbind**と対話する必要はありません。[1]、、 および **rpc.statd** デーモン。**rpc.mountd** デーモンは、エクスポートを設定するために NFS サーバーで引き続き必要ですが、ネットワーク上の操作には関与しません。

注記

TCP は、Red Hat Enterprise Linux の NFS バージョン 3 のデフォルトのトランスポートプロトコルです。UDP は互換性に必要となる場合は使用できますが、その使用範囲についてはできるだけ限定することを推奨しています。NFSv4 には TCP が必要です。

すべての RPC/NFS デーモンには、ポートを設定できる **'-p'** コマンドラインオプションがあり、ファイアウォール設定が容易になります。

TCP ラッパーがクライアントへのアクセスを許可すると、NFS サーバーは **/etc/exports** 設定ファイルを参照して、クライアントがエクスポートされたファイルシステムにアクセスできるかどうかを判断します。アクセスが可能なことが確認されると、そのユーザーは、ファイルおよびディレクトリーへの全操作を行えるようになります。



重要

ファイアーウォールを有効にしている Red Hat Enterprise Linux のデフォルトインストールで NFS を正しく動作させるために、IPTables は、デフォルトの TCP ポート 2049 に設定してください。IPTables が正しく設定されていないと、NFS は正常に動作しません。

NFS 初期化スクリプトおよび **rpc.nfsd** プロセスでは、システム起動時に指定したポートへのバインドが可能になりました。ただし、このポートが使用できない場合や、別のデーモンと競合してしまう場合は、エラーが発生しやすくなる可能性があります。

8.1.1. 必要なサービス

Red Hat Enterprise Linux は、カーネルレベルのサポートとデーモンプロセスの組み合わせを使用し

て、NFS ファイル共有を提供します。すべての NFS バージョンは、クライアントとサーバー間の *Remote Procedure Calls* (*RPC*) に依存します。Red Hat Enterprise Linux 7 の RPC サービスは、**rpcbind** サービスによって制御されます。NFS ファイルシステムの共有やマウントには、実装されている NFS のバージョンに応じて、次のようなサービスが連携して動作することになります。

注記

portmap サービスは、Red Hat Enterprise Linux の以前のバージョンでは、RPC プログラム番号を IP アドレスポート番号の組み合わせにマッピングするために使用されていました。このサービスは、IPv6 サポートを有効にするために、Red Hat Enterprise Linux 7 で **rpcbind** に置き換えられました。

nfs

systemctl start nfs は、NFS サーバーと適切な RPC プロセスを開始し、共有 NFS ファイルシステムの要求を処理します。

nfslock

systemctl start nfs-lock は、適切な RPC プロセスを開始する必須サービスをアクティブにし、NFS クライアントがサーバー上のファイルをロックできるようにします。

rpcbind

rpcbind は、ローカルの RPC サービスからのポート予約を受け入れます。その後、これらのポートは、対応するリモートの RPC サービスによりアクセス可能であることが公開されます。**rpcbind** は RPC サービスの要求に応答し、要求された RPC サービスへの接続を設定します。このプロセスは NFSv4 では使用されません。

以下の RPC プロセスは、NFS サービスを容易にします。

rpc.mountd

このプロセスは、NFS サーバーが NFSv3 クライアントからの **MOUNT** 要求を処理するために使用されます。要求されている NFS 共有が現在 NFS サーバーによりエクスポートされているか、またその共有へのクライアントのアクセスが許可されているかを確認します。マウント要求が許可されると、rpc.mountd サーバーは **Success** ステータスで応答し、この NFS 共有の **File-Handle** を NFS クライアントに戻します。

rpc.nfsd

rpc.nfsd を使用すると、サーバーがアドバタイズする明示的な NFS バージョンとプロトコルを定義できます。NFS クライアントが接続するたびにサーバースレッドを提供するなど、NFS クライアントの動的な要求に対応するため、Linux カーネルと連携して動作します。このプロセスは、**nfs** サービスに対応します。

lockd

lockd は、クライアントとサーバーの両方で実行するカーネルスレッドです。*Network Lock Manager* (NLM) プロトコルを実装し、これにより、NFSv3 クライアントがサーバー上のファイルをロックできるようになります。NFS サーバーが実行中で、NFS ファイルシステムがマウントされていれば、このプロセスは常に自動的に起動します。

rpc.statd

このプロセスは、*Network Status Monitor* (NSM) RPC プロトコルを実装します。NFS サーバーが正常にシャットダウンされずに再起動すると、NFS クライアントに通知します。**rpc.statd** は、**nfslock** サービスにより自動的に開始されるため、ユーザー設定は必要ありません。このプロセ

スは NFSv4 では使用されません。

rpc.rquotad

このプロセスは、リモートユーザーのユーザークォーター情報を提供します。**rpc.rquotad** は、**nfs** サービスにより自動的に開始されるため、ユーザー設定は必要ありません。

rpc.idmapd

rpc.idmapd は、ネットワーク上の NFSv4 名(***ユーザー@ドメイン***の形式の文字列)とローカル UID と GID の間でマッピングされる NFSv4 クライアントおよびサーバーのアップコールを提供します。**idmapd** を NFSv4 で機能させるには、**/etc/idmapd.conf** ファイルを設定する必要があります。少なくとも、NFSv4 マッピングドメインを定義する Domain パラメーターを指定する必要があります。NFSv4 マッピングドメインが DNS ドメイン名と同じ場合は、このパラメーターは必要ありません。クライアントとサーバーが ID マッピングの NFSv4 マッピングドメインに合意しないと、適切に動作しません。

注記

Red Hat Enterprise Linux 7 では、NFSv4 サーバーのみが **rpc.idmapd** を使用します。NFSv4 クライアントは、キーリングベースの idmapper **nfsidmap** を使用します。**nfsidmap** は、ID マッピングを実行するためにカーネルがオンデマンドで呼び出されるスタンドアロンプログラムです。これはデーモンではありません。**nfsidmap** に問題が発生した場合、クライアントは **rpc.idmapd** の使用にフォールバックします。**nfsidmap** の詳細は、man ページの nfsidmap を参照してください。

8.2. NFS クライアントの設定

mount コマンドは、クライアント側に NFS 共有をマウントします。形式は以下のようになります。

```
# mount -t nfs -o options server:/remote/export /local/directory
```

このコマンドは、以下のような変数を使用します。

options

マウントオプションのコンマ区切りリスト。有効な NFS マウントオプションの詳細は、「一般的な NFS マウントオプション」 を参照してください。

server

マウントするファイルシステムをエクスポートするサーバーのホスト名、IP アドレス、または完全修飾ドメイン名

/remote/export

サーバーからエクスポートされるファイルシステムまたはディレクトリー、つまり、マウントするディレクトリー

/local/directory

/remote/export がマウントされているクライアントの場所

Red Hat Enterprise Linux 7 で使用される NFS プロトコルのバージョンは、**マウント** オプション **nfsvers** または **vers** で識別されます。デフォルトでは、**mount** は **mount -t nfs** で NFSv4 を使用しま

す。サーバーが NFSv4 に対応していない場合は、クライアントはサーバーがサポートしているバージョンに自動的にステップダウンします。**nfsvers/vers** オプションを使用して、サーバーでサポートされていない特定のバージョンを渡すと、マウントに失敗します。レガシーの理由により、ファイルシステムタイプ nfs4 も利用可能です。これは **mount -t nfs -o nfsvers=4 *host*:*/remote/export* */local/directory*** を実行するのと同じです。

詳細は、**man mount** を参照してください。

NFS 共有が手動でマウントされた場合は、再起動時に共有は自動的にマウントされません。Red Hat Enterprise Linux は、システムの起動時にリモートファイルシステムを自動的にマウントするための 2 つの方法(**/etc/fstab** ファイルと **autofs** サービス)を提供します。詳細は、「**/etc/fstab**を使用した NFS ファイルシステムのマウント」 および 「**autofs**」 を参照してください。

8.2.1. /etc/fstabを使用した NFS ファイルシステムのマウント

別のマシンから NFS 共有をマウントする別の方法は、**/etc/fstab** ファイルに行を追加することです。その行には、NFS サーバーのホスト名、エクスポートされるサーバーディレクトリー、および NFS 共有がマウントされるローカルマシンディレクトリーを記述する必要があります。**/etc/fstab** ファイルを変更するには、root である必要があります。

例8.1 構文の例

/etc/fstab の行の一般的な構文は以下のとおりです。

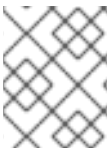
```
server:/usr/local/pub    /pub    nfs    defaults 0 0
```

このコマンドを実行する前に、マウントポイント **/pub** がクライアントマシンに存在している必要があります。この行をクライアントシステムの **/etc/fstab** に追加した後、コマンド **mount /pub** を使用すると、マウントポイント **/pub** がサーバーからマウントされます。

NFS エクスポートをマウントする有効な **/etc/fstab** エントリーには、以下の情報が含まれている必要があります。

server:*/remote/export* */local/directory* nfs *options* 0 0

変数 *server*、*/remote/export*、*/local/directory*、および *options* は、NFS 共有を手動でマウントする際に使用されるものと同じです。詳細は、「NFS クライアントの設定」 を参照してください。



注記

/etc/fstab を読み取る前に、マウントポイント */local/directory* がクライアントに存在している必要があります。それ以外の場合は、マウントに失敗します。

/etc/fstab を編集した後、システムが新しい設定を登録するようにマウントユニットを再生成します。

```
# systemctl daemon-reload
```

関連情報

- **/etc/fstab** の詳細は、**man fstab** を参照してください。

8.3. AUTOFS

/etc/fstab を使用する欠点の1つは、NFS がマウントされたファイルシステムにユーザーがアクセスする頻度に関わらず、マウントされたファイルシステムを所定の場所で維持するために、システムがリソースを割り当てる必要があることです。これは1つまたは 2 つのマウントでは問題になりませんが、システムが一度に多くのシステムへのマウントを維持している場合、システム全体のパフォーマンスに影響を与える可能性があります。**/etc/fstab** の代わりに、カーネルベースの **automount** ユーティリティーを使用します。自動マウント機能は以下の 2 つのコンポーネントで設定されます。

- ファイルシステムを実装するカーネルモジュール
- 他のすべての機能を実行するユーザー空間デーモン

automount ユーティリティーは、NFS ファイルシステムを自動的にマウントおよびアンマウントできます（オンデマンドマウント）。したがって、システムリソースを節約できます。これは、AFS、SMBFS、CIFS、およびローカルファイルシステムを含む他のファイルシステムをマウントするために使用できます。

重要

nfs-utils パッケージは、NFS ファイルサーバーとネットワークファイルシステムクライアントグループの両方に含まれるようになりました。そのため、Base グループではデフォルトでインストールされなくなりました。NFS 共有の自動マウントを試みる前に、最初に nfs-utils がシステムにインストールされていることを確認してください。

autofs は、ネットワークファイルシステムクライアントグループに含まれます。

autofs は、デフォルトのプライマリー設定ファイルとして **/etc/auto.master**（マスターマップ）を使用します。これは、Name Service Switch (NSS)メカニズムとともに **autofs** 設定(**/etc/sysconfig/autofs**内)を使用して、サポートされている別のネットワークソースと名前を使用するように変更できます。**autofs** バージョン 4 デーモンのインスタンスは、マスターマップで設定された各マウントポイントに対して実行されるため、任意のマウントポイントに対してコマンドラインから手動で実行できます。**autofs** バージョン 5 では、設定されたすべてのマウントポイントを管理するために単一のデーモンが使用されるため、これは不可能です。そのため、すべての自動マウントをマスターマップで設定する必要があります。これは、他の業界標準の自動マウント機能の通常の要件と一致しています。マウントポイント、ホスト名、エクスポートしたディレクトリー、および各種オプションは各ホストに対して手動で設定するのではなく、すべて1つのファイルセット (またはサポートされている別のネットワークソース) 内に指定することができます。

8.3.1. バージョン 4 と比較した autofs バージョン 5 の改善点

autofs バージョン 5 では、バージョン 4 と比較して、以下の機能拡張が行われています。

ダイレクトマップのサポート

autofs のダイレクトマップは、ファイルシステム階層の任意の時点でファイルシステムを自動的にマウントするメカニズムを提供します。ダイレクトマップは、マスターマップの **/-** のマウントポイントで示されます。ダイレクトマップのエントリーには、(間接マップで使用される相対パス名の代わりに) 絶対パス名がキーとして含まれています。

レイジーマウントとアンマウントのサポート

マルチマウントマップエントリーは、単一のキーの下にあるマウントポイントの階層を記述します。この良い例として、**-hosts** マップがあります。これは通常、マルチマウントマップエントリーとして **/net/*host*** の下のホストからのすべてのエクスポートを自動マウントするために使用されます。**-hosts** マップを使用する場合、**/net/*ホストの ls は、ホスト*** から各エクスポートの *autofs* トリ

ガーマウントをマウントします。次に、これらはマウントされ、アクセスされると期限切れとなります。これにより、エクスポートが多数あるサーバーにアクセスする際に必要なアクティブなマウントの数を大幅に減らすことができます。

強化された LDAP サポート

autofs 設定ファイル(**/etc/sysconfig/autofs**)は、サイトが実装する **autofs** スキーマを指定するメカニズムを提供するため、アプリケーション自体で試行とエラーによってこれを判断する必要がなくなります。さらに、共通の LDAP サーバー実装でサポートされるほとんどのメカニズムを使用して、LDAP サーバーへの認証済みバインドがサポートされるようになりました。このサポート用の新しい設定ファイル(**/etc/autofs_ldap_auth.conf**)が追加されました。デフォルトの設定ファイルは自己文書化されており、XML 形式を使用します。

Name Service Switch (nsswitch)設定の適切な使用。

Name Service Switch 設定ファイルは、特定の設定データがどこから来るのかを判別する手段を提供するために存在します。この設定の理由は、データにアクセスするための統一されたソフトウェアインターフェイスを維持しながら、管理者が最適なバックエンドデータベースを柔軟に使用できるようにするためです。バージョン 4 の自動マウント機能は、今まで以上に NSS 設定を処理できるようになっていますが、まだ完全ではありません。一方、autofs バージョン 5 は完全な実装です。

このファイルで対応している構文の詳細は、**man nsswitch.conf** を参照してください。すべての NSS データベースが有効なマップソースである訳ではなく、パーサーは無効なデータベースを拒否します。有効なソースは、ファイル、**yp**、**nis**、**nisplus**、**ldap**、および **hesiod** です。

autofs マウントポイントごとの複数のマスターマップエントリー

頻繁に使用されますが、まだ記述されていないのは、直接マウントポイント **/-** の複数のマスターマップエントリーを処理することです。各エントリーのマップキーはマージされ、1 つのマップとして機能します。

> **例8.2 autofs マウントポイントごとの複数のマスターマップエントリー**
>
> 以下は、ダイレクトマウントの connectathon テストマップの例です。
>
> ```
> /- /tmp/auto_dcthon
> /- /tmp/auto_test3_direct
> /- /tmp/auto_test4_direct
> ```

8.3.2. autofs の設定

自動マウント機能の主な設定ファイルは **/etc/auto.master** です。これは、マスターマップ とも呼ばれます。これは、「バージョン 4 と比較した autofs バージョン 5 の改善点」 で説明されているように変更できます。マスターマップには、システム上の **autofs**で制御されたマウントポイントと、それに対応する設定ファイルまたはネットワークソースが自動マウントマップとして知られています。マスターマップの形式は次のとおりです。

```
mount-point map-name options
```

この形式で使用されている変数を以下に示します。

mount-point

autofs マウントポイント（例： **/home** ）

map-name

マウントポイントのリストとマウントポイントがマウントされるファイルシステムの場所が記載されているマップソース名です。

options

指定されている場合は、それ自体にオプションが指定されていない場合に限り、指定されたマップのすべてのエントリーに適用されます。この動作は、オプションが累積されていた **autofs** バージョン 4 とは異なります。混合環境の互換性を実装させるため変更が加えられています。

例8.3 /etc/auto.master ファイル

以下は、**/etc/auto.master** ファイルのサンプル行です(**cat /etc/auto.master**で表示)。

```
/home /etc/auto.misc
```

マップの一般的な形式はそのマスターマップと同じですが、マスターマップではエントリーの末尾に表示されるオプション (options) がマウントポイント (mount-point) と場所 (location) の間に表示されます。

```
mount-point  [options]  location
```

この形式で使用されている変数を以下に示します。

mount-point

これは **autofs** マウントポイントを参照します。これは 1 つのインダイレクトマウント用の 1 つのディレクトリー名にすることも、複数のダイレクトマウント用のマウントポイントの完全パスにすることもできます。各ダイレクトマップエントリーキー(***マウントポイント***)の後に、スペースで区切られたオフセットディレクトリー(/で始まるサブディレクトリー名)のリストが続き、マルチマウントエントリーと呼ばれるものになります。

options

指定した場合は、これらは、独自のオプションを指定しないマップエントリーのマウントオプションになります。

location

これは、ローカルファイルシステムパス(Sun マップ形式のエスケープ文字 ":" が先頭に付き、マップ名が /で始まる場合)、NFS ファイルシステム、またはその他の有効なファイルシステムの場所などのファイルシステムの場所を参照します。

以下は、マップファイルの内容の例です（例： **/etc/auto.misc**）。

```
payroll -fstype=nfs personnel:/dev/hda3
sales -fstype=ext3 :/dev/hda4
```

マップファイルの最初の列は、**autofs** マウントポイント(personnel と呼ばれるサーバーの**sales** および **payroll**)を示してい **ます**。2 列目は **autofs** マウントのオプションを示し、3 番目のコラムはマウントのソースを示しています。指定された設定に従うと、autofs マウントポイントは **/home/payroll** および

/home/sales になります。**-fstype=** オプションは省略されることが多く、通常は正しい操作には必要ありません。

ディレクトリーが存在しない場合、自動マウント機能はディレクトリーを作成します。ディレクトリーが存在している状況で自動マウント機能が起動した場合は、自動マウント機能の終了時にディレクトリーが削除されることはありません。

自動マウントデーモンを起動するには、以下のコマンドを使用します。

```
# systemctl start autofs
```

自動マウントデーモンを再起動するには、以下のコマンドを使用します。

```
# systemctl restart autofs
```

指定の設定を使用して、プロセスが **/home/payroll/2006/July.sxc** などのアンマウントされた **autofs** ディレクトリーへのアクセスを必要とする場合、自動マウントデーモンはディレクトリーを自動的にマウントします。タイムアウトを指定した場合は、タイムアウト期間中ディレクトリーにアクセスしないと、ディレクトリーが自動的にアンマウントされます。

自動マウントデーモンのステータスを表示するには、以下のコマンドを使用します。

```
# systemctl status autofs
```

8.3.3. サイト設定ファイルの上書きまたは拡張

クライアントシステム上の特定マウントポイントのサイトデフォルト値を無効にする場合に便利です。たとえば、次の条件を検討します。

- 自動マウント機能マップは NIS に保存され、**/etc/nsswitch.conf** ファイルには以下のディレクティブがあります。

  ```
  automount:    files nis
  ```

- **auto.master** ファイルには以下が含まれます。

  ```
  +auto.master
  ```

- NIS の **auto.master** マップファイルには以下が含まれます。

  ```
  /home auto.home
  ```

- NIS の **auto.home** マップには以下が含まれます。

  ```
  beth        fileserver.example.com:/export/home/beth
  joe         fileserver.example.com:/export/home/joe
  *       fileserver.example.com:/export/home/&
  ```

- ファイルマップ **/etc/auto.home** は存在しません。

このような状況では、クライアントシステムが NIS マップの **auto.home** を上書きし、別のサーバーからホームディレクトリーをマウントする必要があると仮定します。この場合、クライアントは以下の **/etc/auto.master** マップを使用する必要があります。

```
/home /etc/auto.home
+auto.master
```

/etc/auto.home マップにはエントリーが含まれます。

```
*    labserver.example.com:/export/home/&
```

自動マウンタは最初に出現したマウントポイントのみを処理するため、**/home** には NIS **auto.home** マップではなく、**/etc/auto.home** の内容が含まれます。

または、サイト全体の **auto.home** マップをいくつかのエントリーで拡張するには、**/etc/auto.home** ファイルマップを作成し、新しいエントリーを追加します。最後に、NIS **auto.home** マップを含めます。次に、**/etc/auto.home** ファイルマップは以下のようになります。

```
mydir someserver:/export/mydir
+auto.home
```

これらの NIS **auto.home** マップ条件により、**ls /home** コマンドは以下を出力します。

```
beth joe mydir
```

autofs では、読み取り中のファイルマップと同じ名前のファイルマップの内容が含まれないため、この最後の例は期待どおりに機能します。したがって、**autofs** は、**nsswitch** 設定の次のマップソースに移動します。

8.3.4. LDAP を使用した自動マウント機能マップの格納

LDAP から自動マウント機能マップを取得するように設定されているすべてのシステムに、LDAP クライアントライブラリーをインストールする必要があります。Red Hat Enterprise Linux では、**openldap** パッケージは自動マウント機能 の依存関係として自動的にインストールされ **ます**。LDAP アクセスを設定するには、**/etc/openldap/ldap.conf** を変更します。BASE、URI、スキーマなどが使用するサイトに適した設定になっていることを確認してください。

自動マウントマップを LDAP に格納するために最後に確立されたスキーマは、**rfc2307bis** に記載されています。このスキーマを使用するには、スキーマ定義からコメント文字を削除して、**autofs** 設定 (**/etc/sysconfig/autofs**)に設定する必要があります。以下に例を示します。

例8.4 autofs の設定

```
DEFAULT_MAP_OBJECT_CLASS="automountMap"
DEFAULT_ENTRY_OBJECT_CLASS="automount"
DEFAULT_MAP_ATTRIBUTE="automountMapName"
DEFAULT_ENTRY_ATTRIBUTE="automountKey"
DEFAULT_VALUE_ATTRIBUTE="automountInformation"
```

設定内でコメントされていないスキーマエントリーが上記だけであることを確認します。**automountKey** は、**rfc2307bis** スキーマの **cn** 属性を置き換えます。以下は、LDAP データ交換形式(**LDIF**)の設定例です。

例8.5 LDF の設定

```
# extended LDIF
#
# LDAPv3
# base <> with scope subtree
# filter: (&(objectclass=automountMap)(automountMapName=auto.master))
# requesting: ALL
#

# auto.master, example.com
dn: automountMapName=auto.master,dc=example,dc=com
objectClass: top
objectClass: automountMap
automountMapName: auto.master

# extended LDIF
#
# LDAPv3
# base <automountMapName=auto.master,dc=example,dc=com> with scope subtree
# filter: (objectclass=automount)
# requesting: ALL
#

# /home, auto.master, example.com
dn: automountMapName=auto.master,dc=example,dc=com
objectClass: automount
cn: /home

automountKey: /home
automountInformation: auto.home

# extended LDIF
#
# LDAPv3
# base <> with scope subtree
# filter: (&(objectclass=automountMap)(automountMapName=auto.home))
# requesting: ALL
#

# auto.home, example.com
dn: automountMapName=auto.home,dc=example,dc=com
objectClass: automountMap
automountMapName: auto.home

# extended LDIF
#
# LDAPv3
# base <automountMapName=auto.home,dc=example,dc=com> with scope subtree
# filter: (objectclass=automount)
# requesting: ALL
#

# foo, auto.home, example.com
dn: automountKey=foo,automountMapName=auto.home,dc=example,dc=com
objectClass: automount
```

```
automountKey: foo
automountInformation: filer.example.com:/export/foo

# /, auto.home, example.com
dn: automountKey=/,automountMapName=auto.home,dc=example,dc=com
objectClass: automount
automountKey: /
automountInformation: filer.example.com:/export/&
```

8.4. 一般的な NFS マウントオプション

リモートホストに NFS を使用してファイルシステムをマウントする以外にも、マウントした共有を簡単に使用できるようマウント時に指定できるオプションがあります。これらのオプションは、手動の **マウント** コマンド、**/etc/fstab** 設定、および **autofs** で使用できます。

以下に NFS マウントに一般的に使用されているオプションを示します。

lookupcache=*mode*

任意のマウントポイントに対して、カーネルがディレクトリーエントリーのキャッシュを管理する方法を指定します。*mode* の有効な引数は、**すべて**、**none**、または **pos**/**positive** です。

nfsvers=*version*

使用する NFS プロトコルのバージョンを指定します。*version* は、3 または 4 になります。これは、複数の NFS サーバーを実行するホストに役立ちます。バージョンを指定しないと、NFS はカーネルおよび **mount** コマンドで対応している最新バージョンを使用します。

vers オプションは **nfsvers** と同じですが、互換性のためにこのリリースに含まれています。

noacl

ACL の処理をすべてオフにします。古いバージョンの Red Hat Enterprise Linux、Red Hat Linux、Solaris と連動させる場合に必要となることがあります。こうした古いシステムには、最新の ACL テクノロジーに対する互換性がないためです。

nolock

ファイルのロック機能を無効にします。この設定は、非常に古いバージョンの NFS サーバーに接続する場合に必要となる場合があります。

noexec

マウントしたファイルシステムでバイナリーが実行されないようにします。互換性のないバイナリーを含む、Linux 以外のファイルシステムをマウントしている場合に便利です。

nosuid

set-user-identifier ビットまたは **set-group-identifier** ビットを無効にします。これにより、リモートユーザーが **setuid** プログラムを実行してより高い権限を取得するのを防ぎます。

port=*num*

NFS サーバーポートの数値を指定します。***num*** が **0**（デフォルト値）の場合、**mount** は、使用するポート番号についてリモートホストの **rpcbind** サービスのクエリーを実行します。リモートホストの NFS デーモンがその **rpcbind** サービスに登録されていない場合は、代わりに TCP 2049 の標準 NFS ポート番号が使用されます。

rsize=*num* および wsize=*num*

このオプションは、1 回の NFS 読み取り操作または書き込み操作で転送される最大バイト数を設定します。

rsize と **wsize** には、固定のデフォルト値がありません。デフォルトでは、NFS はサーバーとクライアントの両方がサポートしている最大の値を使用します。Red Hat Enterprise Linux 7 では、クライアントとサーバーの最大値は 1,048,576 バイトです。詳細は、What are the default and maximum values for rsize and wsize with NFS mounts? 参照してください。KBase の記事。

sec=*flavors*

マウントされたエクスポート上のファイルにアクセスするために使用するセキュリティーフレーバーです。*flavors* の値は、複数のセキュリティーフレーバーのコロンで区切られたリストです。

デフォルトでは、クライアントは、クライアントとサーバーの両方をサポートするセキュリティーフレーバーの検索を試みます。サーバーが選択したフレーバーのいずれかに対応していない場合、マウント操作は失敗します。

sec=sys は、ローカルの UNIX UID および GID を使用します。 **AUTH_SYS** を使用して NFS 操作を認証します。

sec=krb5 は、ユーザー認証に、ローカルの UNIX の UID と GID ではなく、Kerberos V5 を使用します。

sec=krb5i は、ユーザー認証に Kerberos V5 を使用し、データの改ざんを防ぐ安全なチェックサムを使用して、NFS 操作の整合性チェックを行います。

sec=krb5p は、ユーザー認証に Kerberos V5 を使用し、整合性チェックを実行し、トラフィックの傍受を防ぐため NFS トラフィックの暗号化を行います。これが最も安全な設定になりますが、パフォーマンスのオーバーヘッドも最も高くなります。

tcp

NFS マウントが TCP プロトコルを使用するよう指示します。

udp

NFS マウントが UDP プロトコルを使用するよう指示します。

詳細は、**man mount** および **man nfs** を参照してください。

8.5. NFS サーバーの起動と停止

前提条件

- NFSv2 または NFSv3 の接続に対応するサーバーでは、**rpcbind**[1] サービスを実行している必要があります。**rpcbind** がアクティブであることを確認するには、次のコマンドを使用します。

```
$ systemctl status rpcbind
```

rpcbind を必要としない NFSv4 専用サーバーを設定するには、 「NFSv4 専用サーバーの設定」 を参照してください。

- Red Hat Enterprise Linux 7.0 では、NFS サーバーが NFSv3 をエクスポートし、起動時に起動できるようにするには、**nfs-lock** サービスを手動で起動して有効にする必要があります。

```
# systemctl start nfs-lock
# systemctl enable nfs-lock
```

Red Hat Enterprise Linux 7.1 以降では、必要に応じて **nfs-lock** が自動的に起動し、手動で有効にしようとすると失敗します。

手順

- NFS サーバーを起動するには、次のコマンドを使用します。

```
# systemctl start nfs
```

- システムの起動時に NFS が起動するようにするには、次のコマンドを使用します。

```
# systemctl enable nfs
```

- サーバーを停止させるには、以下を使用します。

```
# systemctl stop nfs
```

- **restart** オプションは、NFS を停止して起動する簡単な方法です。これは、NFS の設定ファイルを編集した後に設定変更を有効にする最も効率的な方法です。サーバーを再起動するには、次のコマンドを実行します。

```
# systemctl restart nfs
```

- **/etc/sysconfig/nfs** ファイルを編集したら、以下のコマンドを実行して nfs-config サービスを再起動して新しい値を有効にします。

```
# systemctl restart nfs-config
```

- **try-restart** コマンドは、現在実行されている場合にのみ **nfs** を起動します。このコマンドは、Red Hat init スクリプトの **condrestart** (*条件付き再起動*)と同等で、NFS が実行されていない場合にデーモンを起動しないため便利です。

  条件付きでサーバーを再起動するには、以下を入力します。

```
# systemctl try-restart nfs
```

- サービスを再起動せずに NFS サーバー設定ファイルの再読み込みを実行するには、以下のように入力します。

```
# systemctl reload nfs
```

8.6. NFS サーバーの設定

NFS サーバーでエクスポートを設定するには、以下の 2 つの方法があります。

- NFS 設定ファイル(**/etc/exports**)を手動で編集する。
- コマンドラインで、コマンド **exportfs**を使用する方法。

8.6.1. /etc/exports 設定ファイル

/etc/exports ファイルは、リモートホストにどのファイルシステムをエクスポートするかを制御し、オプションを指定します。以下の構文ルールに従います。

- 空白行は無視する。
- コメントを追加するには、ハッシュマーク(**#**)で行を開始します。
- 長い行はバックスラッシュ(\)で囲むことができます。
- エクスポートするファイルシステムは、それぞれ 1 行で指定する。
- 許可するホストのリストは、エクスポートするファイルシステムの後に空白文字を追加し、その後に追加する。
- 各ホストのオプションは、ホストの識別子の直後に括弧を追加し、その中に指定する。ホストと最初の括弧の間には空白を使用しない。

エクスポートするファイルシステムの各エントリーは、以下のように指定します。

> *export host*(*options*)

ここでは、以下のような変数を使用しています。

export
エクスポートするディレクトリー

host
エクスポートを共有するホストまたはネットワーク

オプション
host に使用されるオプション

各ホストにそれぞれオプションを付けて、複数のホストを 1 行で指定することができます。この場合は、以下のように、各ホスト名の後に、そのホストに対するオプションを括弧を付けて追加します。ホストは空白文字で区切ります。

> *export host1*(*options1*) *host2*(*options2*) *host3*(*options3*)

ホスト名を指定するさまざまな方法は、「ホスト名の形式」 を参照してください。

最も単純な形式では、**/etc/exports** ファイルに、エクスポートされるディレクトリーと、そのディレクトリーへのアクセスを許可するホストを指定するだけです。以下に例を示します。

> **例8.6 /etc/exports ファイル**
>
> ```
> /exported/directory bob.example.com
> ```
>
> ここで、**bob.example.com** は、NFS サーバーから **/exported/directory/** をマウントできます。この例ではオプションが指定されていないため、**デフォルト** 設定が使用されます。

デフォルトの設定は以下のようになります。

ro

エクスポートするファイルシステムは読み取り専用です。リモートホストは、このファイルシステムで共有されているデータを変更できません。このファイルシステムで変更 (読み取り/書き込み) を可能にするには、**rw** オプションを指定します。

sync

NFS サーバーは、以前の要求で発生した変更がディスクに書き込まれるまで、要求に応答しません。代わりに非同期書き込みを有効にするには、**async** オプションを指定します。

wdelay

NFS サーバーは、別の書き込み要求が差し迫っていると判断すると、ディスクへの書き込みを遅らせます。これにより、複数の書き込みコマンドが同じディスクにアクセスする回数を減らすことができるため、書き込みのオーバーヘッドが低下し、パフォーマンスが向上します。これを無効にするには、**no_wdelay** を指定します。**no_wdelay** は、デフォルトの **sync** オプションも指定されている場合にのみ使用できます。

root_squash

（ローカルからではなく） **リモート** から接続している root ユーザーが root 権限を持つことを阻止します。代わりに、NFS サーバーはユーザー ID **nfsnobody** を割り当てます。これにより、リモートの root ユーザーの権限を、最も低いローカルユーザーレベルにまで下げて (squash)、高い確率でリモートサーバーへの書き込む権限を与えないようにすることができます。root squashing を無効にするには、**no_root_squash** を指定します。

すべてのリモートユーザー (root を含む) を抑制するには、**all_squash** を使用します。特定ホストのリモートユーザーに対して、NFS サーバーが割り当てるユーザー ID とグループ ID を指定するには、**anonuid** オプションと **anongid** オプションを以下のように使用します。

```
export host(anonuid=uid,anongid=gid)
```

uid と *gid* は、それぞれユーザー ID とグループ ID の番号になります。 **anonuid** オプションと **anongid** オプションにより、共有するリモート NFS ユーザー用に、特別なユーザーアカウントおよびグループアカウントを作成できます。

デフォルトでは、*アクセス制御リスト* (*ACLs*) は、Red Hat Enterprise Linux では NFS が対応しています。この機能を無効にするには、ファイルシステムをエクスポートするときに **no_acl** オプションを指定します。

エクスポートするすべてのファイルシステムの各デフォルトは、明示的に上書きする必要があります。たとえば、**rw** オプションを指定しないと、エクスポートするファイルシステムが読み取り専用として共有されます。以下は、**/etc/exports** のサンプル行で、2 つのデフォルトオプションを上書きします。

```
/another/exported/directory 192.168.0.3(rw,async)
```

この例では、**192.168.0.3** は **/another/exported/directory/** の読み取りおよび書き込みをマウントでき、ディスクへの書き込みはすべて非同期です。エクスポートオプションの詳細は、**man exportfs** を参照してください。

さらに、デフォルト値が指定されていないオプションも利用できます。たとえば、サブツリーチェックを無効にする、安全でないポートからのアクセスの許可する、安全でないファイルロックを許可する (一部の初期 NFS クライアント実装で必要) などの機能があります。これらのあまり使用されないオプションの詳細は、**man exports** を参照してください。

重要

/etc/exports ファイルのフォーマットは、特に空白文字の使用に関して非常に正確です。ホストからエクスポートするファイルシステムの間、そしてホスト同士の間には、必ず空白文字を挿入してください。また、それ以外の場所 (コメント行を除く) には、空白文字を追加しないでください。

たとえば、以下の 2 つの行は意味が異なります。

```
/home bob.example.com(rw)
/home bob.example.com (rw)
```

最初の行は、**bob.example.com** からのユーザーのみが **/home** ディレクトリーへの読み取りおよび書き込みアクセスを許可します。2 行目では、**bob.example.com** からのユーザーがディレクトリーを読み取り専用（デフォルト）としてマウントすることを許可し、残りのユーザーは読み取り/書き込みでマウントできます。

8.6.2. exportfs コマンド

NFS を使用してリモートユーザーにエクスポートされるすべてのファイルシステムと、そのファイルシステムのアクセスレベルは、**/etc/exports** ファイルにリストされています。**nfs** サービスが起動すると、**/usr/sbin/exportfs** コマンドが起動し、このファイルを読み込み、制御を実際のマウントプロセスの **rpc.mountd** (NFSv3 の場合は)に渡してから、リモートユーザーがファイルシステムを利用できる **rpc.nfsd** に渡します。

手動で発行すると、**/usr/sbin/exportfs** コマンドを使用すると、root ユーザーが NFS サービスを再起動せずにディレクトリーを選択的にエクスポートまたはアンエクスポートできます。適切なオプションを指定すると、**/usr/sbin/exportfs** コマンドはエクスポートしたファイルシステムを **/var/lib/nfs/xtab** に書き込みます。ファイルシステムへのアクセス権限を決定する際には、**rpc.mountd** が **xtab** ファイルを参照するため、エクスポートしたファイルシステムのリストへの変更がすぐに反映されます。

以下は、**/usr/sbin/exportfs** で利用可能な一般的に使用されるオプションの一覧です。

-r

/var/lib/nfs/etab に新しいエクスポート一覧を作成して、**/etc/exports** に一覧表示されているすべてのディレクトリーをエクスポートします。このオプションにより、**/etc/exports** に加えられた変更でエクスポート一覧が効果的に更新されます。

-a

/usr/sbin/exportfs に渡される他のオプションに応じて、すべてのディレクトリーがエクスポートまたはアンエクスポートされます。その他のオプションが指定されていない場合、**/usr/sbin/exportfs** は、**/etc/exports** で指定されたすべてのファイルシステムをエクスポートします。

-o *file-systems*

/etc/exports に記載されていないエクスポートするディレクトリーを指定します。*file-systems* の部分を、エクスポートされる追加のファイルシステムに置き換えます。これらのファイルシステムは、**/etc/exports** で指定したのと同じ方法でフォーマットする必要があります。このオプションは、エクスポートするファイルシステムのリストに永続的に追加する前に、エクスポートするファイルシステムをテストするためによく使用されます。**/etc/exports** 構文の詳細は、「**/etc/exports** 設定ファイル」 を参照してください。

-i

/etc/exports を無視します。コマンドラインから指定したオプションのみが、エクスポートされるファイルシステムの定義に使用されます。

-u

すべての共有ディレクトリーをエクスポートしなくなります。コマンド **/usr/sbin/exportfs -ua** は、すべての NFS デーモンを起動したままにして NFS ファイル共有を一時停止します。NFS 共有を再度有効にするには、**exportfs -r** を使用します。

-v

exportfs コマンドの実行時にエクスポートまたはエクスポートされていないファイルシステムの詳細を表示する詳細な操作です。

exportfs コマンドにオプションが渡されていない場合は、現在エクスポートされているファイルシステムの一覧が表示されます。**exportfs** コマンドの詳細は、**man exportfs** を参照してください。

8.6.2.1. NFSv4 での exportfs の使用

Red Hat Enterprise Linux 7 では、提示されるファイルシステムは自動的に同じパスを使用して NFSv3 および NFSv4 クライアントで利用可能になるため、NFSv4 のエクスポートを設定するための特別なステップは必要ありません。これが、以前のバージョンとは異なります。

クライアントが NFSv4 を使用しないようにするには、**/etc/sysconfig/nfs** で **RPCNFSDARGS= -N 4** を設定して無効にします。

8.6.3. ファイアウォール背後での NFS の実行

NFS には **rpcbind** が必要です。これは RPC サービスにポートを動的に割り当て、ファイアウォールルールの設定に問題が発生する可能性があります。クライアントがファイアウォールの背後にある NFS 共有にアクセスできるようにするには、**/etc/sysconfig/nfs** ファイルを編集して、RPC サービスを実行するポートを設定します。クライアントがファイアウォールを介して RPC クォータにアクセスできるようにする方法は、「ファイアウォールを介した RPC クォータのアクセス」 を参照してください。

/etc/sysconfig/nfs ファイルは、デフォルトではすべてのシステムに存在しません。**/etc/sysconfig/nfs** が存在しない場合は作成し、以下を指定します。

RPCMOUNTDOPTS="-p *port*"

これにより、"-p *port*" が rpc.mount コマンドラインに追加されます(**rpc.mount -p *port***)。

nlockmgr サービスが使用するポートを指定するには、**/etc/modprobe.d/lockd.conf** ファイルの ***nlm_tcpport*** オプションおよび ***nlm_udpport*** オプションのポート番号を設定します。

NFS が起動しない場合は、**/var/log/messages** を確認してください。通常、すでに使用されているポート番号を指定すると、NFS が起動しません。**/etc/sysconfig/nfs** を編集したら、以下のコマンドを実行して、Red Hat Enterprise Linux 7.2 以前で **nfs-config** サービスを再起動する必要があります。

```
# systemctl restart nfs-config
```

次に、NFS サーバーを再起動します。

```
# systemctl restart nfs-server
```

rpcinfo -p を実行して、変更が有効になったことを確認します。

注記

NFSv4.0 コールバックがファイアウォールを通過できるようにするには、**/proc/sys/fs/nfs/nfs_callback_tcpport** を設定し、サーバーがクライアント上のそのポートに接続できるようにします。

このプロセスは、NFSv4.1 以降には必要ありません。純粋な NFSv4 環境では、**mountd**、**statd**、および **lockd** 用の他のポートは必要ありません。

8.6.3.1. NFS エクスポートの検出

NFS サーバーがエクスポートするファイルシステムを検出する方法は 2 種類あります。

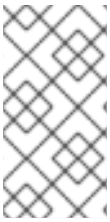
- NFSv3 に対応するサーバーでは、**showmount** コマンドを使用します。

```
$ showmount -e myserver
Export list for mysever
/exports/foo
/exports/bar
```

- NFSv4 に対応するサーバーで、root ディレクトリーを **マウント** し、確認します。

```
# mount myserver:/ /mnt/
# cd /mnt/
exports
# ls exports
foo
bar
```

NFSv4 と NFSv3 の両方に対応するサーバーでは、上記の方法はいずれも有効で、同じ結果となります。



注記

Red Hat Enterprise Linux 6 以前には、設定の仕方によって以前の NFS サーバーは別々のパス経由で NFSv4 クライアントにファイルシステムをエクスポートすることがありました。このようなサーバーでは、デフォルトで NFSv4 が有効にならないため、問題にはなりません。

8.6.4. ファイアウォールを介した RPC クォータのアクセス

ディスククォータを使用するファイルシステムをエクスポートする場合は、クォータの RPC (Remote Procedure Call) サービスを使用して、NFS クライアントにディスククォータデータを提供できます。

手順8.1 ファイアウォールの内側で RPC クォータにアクセスできるようにする

1. **rpc-rquotad** サービスを有効にするには、以下のコマンドを使用します。

```
# systemctl enable rpc-rquotad
```

2. **rpc-rquotad** サービスを起動するには、以下のコマンドを使用します。

```
# systemctl start rpc-rquotad
```

rpc-rquotad が有効になっている場合は、**nfs-server** サービスの起動後に自動的に起動することに注意してください。

3. ファイアウォールの内側でクォータ RPC サービスにアクセスできるようにするには、UDP または TCP ポート **875** を開く必要があります。デフォルトのポート番号は **/etc/services** ファイルで定義されています。

   /etc/sysconfig/rpc -rquotad ファイルの RPCRQUOTADOPTS 変数に -p *port-number* を追加して、デフォルトのポート番号をオーバーライドできます。

4. **rpc-rquotad** を再起動して、**/etc/sysconfig/rpc-rquotad** ファイルの変更を有効にします。

```
# systemctl restart rpc-rquotad
```

リモートホストからのクォータの設定

デフォルトでは、クォータはリモートホストのみが読み取ることができます。クォータの設定を許可するには、**/etc/sysconfig/rpc-rquotad** ファイルの **RPCRQUOTADOPTS** 変数に **-S** オプションを追加します。

rpc-rquotad を再起動して、**/etc/sysconfig/rpc-rquotad** ファイルの変更を有効にします。

```
# systemctl restart rpc-rquotad
```

8.6.5. ホスト名の形式

ホストは以下の形式にすることができます。

単独のマシン

完全修飾型ドメイン名 (サーバーで解決可能な形式)、ホスト名 (サーバーで解決可能な形式)、あるいは IP アドレス

ワイルドカードで指定された一連のマシン

文字列の一致を指定するには、* 文字または **?** 文字を使用します。ワイルドカードは IP アドレスでは使用しないことになっていますが、逆引き DNS ルックアップが失敗した場合には誤って動作する可能性があります。完全修飾ドメイン名でワイルドカードを指定する場合、ドット(.)はワイルドカードに含まれません。たとえば、***.example.com** には **one.example.com** が含まれていますが、**1.two.example.com は含ま** れません。

IP ネットワーク

a.b.c.d/z を使用します。ここで、*a.b.c.d* はネットワーク、*z* はネットマスクのビット数です (たとえば、192.168.0.0/24)。別の使用可能形式は *a.b.c.d/netmask* となり、ここで *a.b.c.d* がネットワークで、*netmask* がネットマスクです (たとえば、192.168.100.8/255.255.255.0)。

Netgroups

形式 *@group-name* を使用します。ここで、*group-name* は NIS netgroup の名前です。

8.6.6. NFS over RDMA の有効化 (NFSoRDMA)

Red Hat Enterprise Linux 7 では、RDMA に対応するハードウェアが存在すると、RDMA (remote direct memory access) サービスが自動的に有効になります。

RDMA で NFS を有効にするには、次のコマンドを実行します

1. rdma パッケージと rdma-core パッケージをインストールします。

   /etc/rdma/rdma.conf ファイルには、デフォルトで ***XPRTRDMA_LOAD=yes*** を設定する行が含まれています。これは、**rdma** サービスに NFSoRDMA **クライアント** モジュールを読み込むように要求します。

2. NFSoRDMA **サーバー** モジュールの自動読み込みを有効にするには、**/etc/rdma/rdma.conf** の新しい行に ***SVCRDMA_LOAD=yes*** を追加します。

   /etc/sysconfig/nfs ファイルの ***RPCNFSDARGS="--rdma=20049"*** は、NFSoRDMA サービスがクライアントをリッスンするポート番号を指定します。『RFC 5667』 は、RDMA で NFSv4 サービスを提供する際に、サーバーがポート **20049** をリッスンする必要があることを指定します。

3. **/etc/rdma/rdma.conf** ファイルの編集後に、**nfs** サービスを再起動します。

   ```
   # systemctl restart nfs
   ```

   以前のカーネルバージョンでは、変更を有効にするために **/etc/rdma/rdma.conf** の編集後にシステムを再起動する必要があることに注意してください。

8.6.7. NFSv4 専用サーバーの設定

デフォルトでは、NFS サーバーは Red Hat Enterprise Linux 7 の NFSv2、NFSv3、および NFSv4 の接続に対応します。ただし、バージョン 4.0 以降の NFS のみに対応するように NFS を設定することもできます。NFSv4 では **rpcbind** サービスがネットワークをリッスンする必要がないため、これにより、システムで開いているポートと実行中のサービスの数が最小限に抑えられます。

NFS サーバーが NFSv4 専用として設定されていると、NFSv2 または NFSv3 を使用して共有をマウントしようとするクライアントは、次のようなエラーでマウントに失敗します。

```
Requested NFS version or transport protocol is not supported.
```

手順8.2 NFSv4 専用サーバーの設定

NFS サーバーを、NFS バージョン 4.0 以降にのみ対応するように設定するには、次のコマンドを実行します。

1. **/etc/sysconfig/nfs** 設定ファイルに以下の行を追加して、NFSv2、NFSv3、および UDP を無効にします。

   ```
   RPCNFSDARGS="-N 2 -N 3 -U"
   ```

2. 必要に応じて、**RPCBIND**、**MOUNT**、および **NSM** プロトコル呼び出しのリッスンを無効にします。これは、NFSv4 専用の場合は不要です。

   このオプションを無効にすると、以下のような影響があります。

   - NFSv2 または NFSv3 を使用してサーバーから共有をマウントしようとするクライアントが応答しなくなります。
   - NFS サーバー自体が、NFSv2 および NFSv3 のファイルシステムをマウントできなくなります。

   このオプションを無効にするには、以下を実行します。

- 以下を **/etc/sysconfig/nfs** ファイルに追加します。

```
RPCMOUNTDOPTS="-N 2 -N 3"
```

- 関連するサービスを無効にします。

```
# systemctl mask --now rpc-statd.service rpcbind.service rpcbind.socket
```

3. NFS サーバーを再起動します。

```
# systemctl restart nfs
```

変更は、NFS サーバーを起動または再起動するとすぐに反映されます。

NFSv4 専用の設定の確認

netstat ユーティリティーを使用して、NFSv4 専用モードで NFS サーバーが設定されていることを確認できます。

- 以下は、NFSv4 専用サーバーでの **netstat** 出力例です。**RPCBIND**、**MOUNT**、および **NSM** をリッスンして無効になります。**nfs** は、唯一のリスニング NFS サービスです。

```
# netstat -ltu

Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:nfs             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN
tcp        0      0 localhost:smtp          0.0.0.0:*               LISTEN
tcp6       0      0 [::]:nfs                [::]:*                  LISTEN
tcp6       0      0 [::]:12432              [::]:*                  LISTEN
tcp6       0      0 [::]:12434              [::]:*                  LISTEN
tcp6       0      0 localhost:7092          [::]:*                  LISTEN
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN
udp        0      0 localhost:323           0.0.0.0:*
udp        0      0 0.0.0.0:bootpc          0.0.0.0:*
udp6       0      0 localhost:323           [::]:*
```

- 一方、NFSv4 専用サーバーを設定する前の **netstat** 出力には、**sunrpc** および **mountd** サービスが含まれます。

```
# netstat -ltu

Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:nfs             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:36069           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:52364           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:sunrpc          0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:mountd          0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN
tcp        0      0 localhost:smtp          0.0.0.0:*               LISTEN
tcp6       0      0 [::]:34941              [::]:*                  LISTEN
```

```
tcp6       0      0 [::]:nfs                [::]:*                  LISTEN
tcp6       0      0 [::]:sunrpc             [::]:*                  LISTEN
tcp6       0      0 [::]:mountd             [::]:*                  LISTEN
tcp6       0      0 [::]:12432              [::]:*                  LISTEN
tcp6       0      0 [::]:56881              [::]:*                  LISTEN
tcp6       0      0 [::]:12434              [::]:*                  LISTEN
tcp6       0      0 localhost:7092          [::]:*                  LISTEN
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN
udp        0      0 localhost:323           0.0.0.0:*
udp        0      0 0.0.0.0:37190           0.0.0.0:*
udp        0      0 0.0.0.0:876             0.0.0.0:*
udp        0      0 localhost:877           0.0.0.0:*
udp        0      0 0.0.0.0:mountd          0.0.0.0:*
udp        0      0 0.0.0.0:38588           0.0.0.0:*
udp        0      0 0.0.0.0:nfs             0.0.0.0:*
udp        0      0 0.0.0.0:bootpc          0.0.0.0:*
udp        0      0 0.0.0.0:sunrpc          0.0.0.0:*
udp6       0      0 localhost:323           [::]:*
udp6       0      0 [::]:57683              [::]:*
udp6       0      0 [::]:876                [::]:*
udp6       0      0 [::]:mountd             [::]:*
udp6       0      0 [::]:40874              [::]:*
udp6       0      0 [::]:nfs                [::]:*
udp6       0      0 [::]:sunrpc             [::]:*
```

8.7. NFS のセキュア化

NFS は、多数の既知のホストと、ファイルシステム全体を透過的に共有する場合に適しています。ただし、使いやすさがある反面、さまざまなセキュリティー問題を伴います。サーバーで NFS ファイルシステムをエクスポートする場合や、クライアントにマウントする際に、NFS セキュリティーリスクを最小限に抑え、サーバーのデータを保護するには、以下のセクションを検討してください。

8.7.1. AUTH_SYS とエクスポート制御による NFS セキュリティー保護

従来より、NFS ではエクスポートしたファイルへのアクセスを制御するために 2 種類のオプションを提供しています。

最初に、サーバーが、IP アドレスまたはホスト名で、どのファイルシステムをマウントするかを制限します。

2 つ目は、ローカルユーザーと同じ方法で、サーバーが NFS クライアント上のユーザーに対してファイルシステムの権限を強制するオプションです。従来は、**AUTH_SYS** (**AUTH_UNIX**とも呼ばれます)を使用してこれを行い、クライアントに依存し、ユーザーの UID と GID を示していました。つまり、悪意のあるクライアントや誤って設定されたクライアントがこれを誤用し、ファイルへのアクセスを許可すべきではないユーザーに対して、ファイルへのアクセスを簡単に与えてしまうことができるため注意が必要です。

こうしたリスクを抑えるため、管理者によって共通のユーザーおよびグループ ID へのユーザー権限が取り消されたり、読み取り専用のアクセスに制限されたりすることがよくあります。ただし、このソリューションにより、NFS 共有が元々想定されている方法では使用されなくなります。

また、NFS ファイルシステムをエクスポートしているシステムで使用している DNS サーバーのコントロールが攻撃者に奪われると、特定のホスト名または完全修飾ドメイン名に関連付けられているシステムが、未承認のマシンに向かう可能性があります。この時、NFS マウントには、これ以上の安全確保を

目的としたユーザー名やパスワード情報の交換が行われないため、この未承認のマシンが NFS 共有のマウントを許可されたシステムに **なってしまいます**。

NFS 経由でディレクトリーのエクスポートを行う際にワイルドカードを使用する場合は慎重に行ってください。ワイルドカードの対象が予定よりも広い範囲のシステムを対象とする可能性があります。

また、TCP ラッパーを使用して **rpcbind**[1] サービスへのアクセスを制限することもできます。**iptables** でルールを作成すると、**rpcbind**、**rpc.mountd**、および **rpc.nfsd** が使用するポートへのアクセスを制限することもできます。

NFS および **rpcbind** のセキュリティー保護に関する詳細は、**man iptables** を参照してください。

8.7.2. AUTH_GSSを使用した NFS 保護

NFSv4 は、RPCSEC_GSS と Kerberos バージョン 5 GSS-API の実装を義務付けることで、NFS のセキュリティーに革命を起こしました。ただし、RPCSEC_GSS や Kerberos のメカニズムは、NFS のいずれのバージョンでも利用できます。FIPS モードでは、FIPS が許可するアルゴリズムのみを使用できます。

AUTH_SYS とは異なり、RPCSEC_GSS Kerberos メカニズムでは、サーバーは、どのユーザーがそのファイルにアクセスしているかを正確に表すことをクライアントに依存しません。代わりに、暗号を使用してサーバーにユーザーを認証し、悪意のあるクライアントがそのユーザーの Kerberos 認証情報を持たずにユーザーになりすますことがないようにします。RPCSEC_GSS Kerberos メカニズムを使用することが、マウントを保護する最も簡単な方法になります。Kerberos の設定後は、追加の設定が不要になるためです。

Kerberos の設定

NFSv4 Kerberos 対応サーバーを設定する前に、Kerberos Key Distribution Centre (KDC) をインストールして設定する必要があります。Kerberos はネットワーク認証システムであり、対称暗号化と、信頼できるサードパーティー (KDC) を使用してクライアントとサーバーが相互に認証できるようにします。Red Hat では、Kerberos の設定に Identity Management (IdM) を使用することを推奨します。

手順8.3 RPCSEC_GSS を使用するための IdM 用の NFS サーバーとクライアントの設定

1. - NFS サーバー側で **nfs/hostname.*domain@REALM*** プリンシパルを作成します。
   - サーバーとクライアント側の両方に、**host/hostname.*domain@REALM*** を作成します。

     注記

     ホスト名 は、NFS サーバーのホスト名 と同じでなければなりません。

   - *クライアントとサーバーのキータブに、対応する鍵を追加します。*

   手順は、Red Hat Enterprise Linux 7 Linux Domain Identity, Authentication, and Policy Guide の Adding and Editing Service Entries and Keytabs and Setting up a Kerberos-aware NFS Server のセクションを参照してください。

2. *サーバーで、**sec=** オプションを使用して、希望するセキュリティーフレーバーを有効にします。すべてのセキュリティーフレーバーと非暗号化マウントを有効にするには、以下のコマンドを実行します。*

   ```
   /export *(sec=sys:krb5:krb5i:krb5p)
   ```

sec= オプションで使用する有効なセキュリティーフレーバーは次のとおりです。

- **sys**: 暗号化の保護なし（デフォルト）
- **krb5**: 認証のみ
- **krb5i**: 整合性保護
- **krb5p**: プライバシー保護

3. クライアント側で、**sec=krb5**（または設定に応じて **sec=krb5i**、または **sec=krb5p**）をマウントオプションに追加します。

```
# mount -o sec=krb5 server:/export /mnt
```

NFS クライアントの設定方法の詳細は、Red Hat Enterprise Linux 7 Linux Domain Identity, Authentication, and Policy Guide の Setting up a Kerberos-aware NFS Client セクションを参照してください。

関連情報

- Red Hat は IdM の使用を推奨しますが、Active Directory (AD) Kerberos サーバーにも対応しています。詳細は、Red Hat ナレッジベースの記事 How to set up NFS using Kerberos authentication on RHEL 7 using SSSD and Active Directory を参照してください。
- Kerberos で保護された NFS 共有に root としてファイルを書き込み、そのファイルで root 所有権を保持する必要がある場合は、https://access.redhat.com/articles/4040141 を参照してください。この設定は推奨されません。
- NFS クライアント設定の詳細は、exports(5) と nfs(5) の man ページ、および「一般的な NFS マウントオプション」を参照してください。
- **gssproxy** および **rpc.gssd** の相互運用方法など、**RPCSEC_GSS** フレームワークの詳細は GSSD フローの説明 を参照してください。

8.7.2.1. NFSv4 による NFS のセキュリティー保護

NFSv4 には、Microsoft Windows NT モデルの機能や幅広い導入の経緯があるため、POSIX モデルではなく、Microsoft Windows NT モデルをベースとした ACL サポートが含まれます。

NFSv4 のもう 1 つの重要なセキュリティー機能は、ファイルシステムのマウントに **MOUNT** プロトコルを使用することを削除することです。**MOUNT** プロトコルは、プロトコルがファイルを処理する方法が原因でセキュリティーリスクを示しました。

8.7.3. ファイル権限

リモートホストにより NFS ファイルシステムを読み取りまたは読み書きとしてマウントした場合は、パーティションが、各共有ファイルに対する唯一の保護となります。同じユーザー ID 値を共有している 2 つのユーザーが同じ NFS ファイルシステムをマウントすると、それらのユーザーはファイルを相互に変更することができます。さらに、クライアントシステムで root としてログインしているユーザーは、**su -** コマンドを使用して、NFS 共有のあるファイルにアクセスできます。

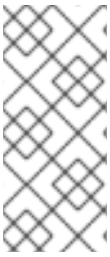
デフォルトでは、アクセス制御リスト (ACL) は、Red Hat Enterprise Linux では NFS が対応しています。Red Hat は、この機能を有効な状態にしておくことを推奨しています。

デフォルトでは、NFS がファイルシステムをエクスポートする際に、root squashing を使用します。こ

れにより、ローカルマシンで root ユーザーとして NFS 共有にアクセスするユーザーのユーザー ID が **nobody** に設定されます。root squashing は、デフォルトのオプション **root_squash** で制御されます。このオプションの詳細については、「**/etc/exports** 設定ファイル」 を参照してください。できる限りこの root squash 機能は無効にしないでください。

NFS 共有を読み取り専用としてエクスポートする場合は、**all_squash** オプションの使用を検討してください。このオプションでは、エクスポートしたファイルシステムにアクセスするすべてのユーザーが、**nfsnobody** ユーザーのユーザー ID を取得します。

8.8. NFS および RPCBIND



注記

次のセクションは、後方互換性のために **rpcbind** サービスを必要とする NFSv3 実装にのみ適用されます。

rpcbind を必要としない NFSv4 専用サーバーを設定する方法は、「NFSv4 専用サーバーの設定」 を参照してください。

rpcbind[1] ユーティリティーは、RPC サービスを、それらがリッスンするポートにマッピングします。RPC プロセスは、開始時に **rpcbind** に通知し、そのプロセスがリッスンしているポートと、そのプロセスが提供することが予想される RPC プログラム番号を登録します。クライアントシステムは、特定の RPC プログラム番号でサーバー上の **rpcbind** に問い合わせます。**rpcbind** サービスは、クライアントを適切なポート番号にリダイレクトし、要求されたサービスと通信できるようにします。

RPC ベースのサービスは、**rpcbind** を使用してクライアントの受信要求ですべての接続を行うため、これらのサービスが起動する前に **rpcbind** が使用可能である必要があります。

rpcbind サービスはアクセス制御に TCP ラッパーを使用し、**rpcbind** のアクセス制御ルールは **すべて** の RPC ベースのサービスに影響します。あるいは、NFS RPC デーモンごとにアクセス制御ルールを指定することもできます。**rpc.mountd** および **rpc.statd** の **man** ページには、これらのルールの正確な構文に関する情報が記載されています。

8.8.1. NFS と rpcbindのトラブルシューティング

rpcbind[1] は、通信に使用される RPC サービスとポート番号間の調整を提供するため、トラブルシューティングの際に **rpcbind** を使用して現在の RPC サービスの状態を表示すると便利です。**rpcinfo** コマンドは、RPC ベースの各サービスとそのポート番号、RPC プログラム番号、バージョン番号、および IP プロトコルタイプ(TCP または UDP)を表示します。

rpcbind に対して適切な RPC ベースの NFS サービスが有効になっていることを確認するには、次のコマンドを使用します。

```
# rpcinfo -p
```

例8.7 rpcinfo -p コマンドの出力

以下に、上記コマンドの出力例を示します。

```
program vers proto  port service
    100021    1   udp  32774  nlockmgr
    100021    3   udp  32774  nlockmgr
    100021    4   udp  32774  nlockmgr
```

```
100021    1   tcp  34437  nlockmgr
100021    3   tcp  34437  nlockmgr
100021    4   tcp  34437  nlockmgr
100011    1   udp    819  rquotad
100011    2   udp    819  rquotad
100011    1   tcp    822  rquotad
100011    2   tcp    822  rquotad
100003    2   udp   2049  nfs
100003    3   udp   2049  nfs
100003    2   tcp   2049  nfs
100003    3   tcp   2049  nfs
100005    1   udp    836  mountd
100005    1   tcp    839  mountd
100005    2   udp    836  mountd
100005    2   tcp    839  mountd
100005    3   udp    836  mountd
100005    3   tcp    839  mountd
```

*NFS サービスの1 つが正しく起動しないと、**rpcbind** はそのサービスに対するクライアントからの RPC 要求を正しいポートにマップできなくなります。多くの場合、NFS が **rpcinfo** の出力に存在しない場合は、NFS を再起動すると、サービスが **rpcbind** に正しく登録され、機能し始めます。*

***rpcinfo** の詳細とオプションの一覧は、**man** ページの を参照してください。*

8.9. PNFS

NFS v4.1 標準の一部としてのパラレル NFS (pNFS) のサポートは、Red Hat Enterprise Linux 6.4 以降で利用できます。pNFS アーキテクチャーは NFS の拡張性を向上させるため、パフォーマンスが強化される場合もあります。つまり、サーバーに pNFS も実装されると、クライアントは複数のサーバーで同時にデータにアクセスできるようになります。これは、ファイル、オブジェクト、ブロックの 3 つのストレージプロトコルまたはレイアウトに対応します。

注記

このプロトコルでは、pNFS のレイアウトタイプに、ファイル、オブジェクト、ブロックの 3 つを使用できます。Red Hat Enterprise Linux 6.4 クライアントではファイルレイアウトタイプのみをサポートしていましたが、Red Hat Enterprise Linux 7 ではファイルレイアウトタイプをサポートするほか、オブジェクトレイアウトタイプおよびブロックレイアウトタイプがテクノロジープレビューに含まれています。

pNFS の Flex ファイル

Flex ファイルは、スタンドアロンの NFSv3 サーバーおよび NFSv4 サーバーをスケールアウトした名前空間に集約できるようにする pNFS 用の新しいレイアウトです。Flex Flex 機能は、RFC 7862 仕様で説明されているように、NFSv4.2 標準に含まれています。

Red Hat Enterprise Linux 7.4 以降では、Flex ファイルサーバーから NFS 共有をマウントできます。

pNFS 共有のマウント

- *pNFS 機能を有効にするには、NFS バージョン 4.1 以降を使用して、pNFS が有効になっているサーバーから共有をマウントします。*

```
# mount -t nfs -o v4.1 server:/remote-export /local-directory
```

サーバーが pNFS が有効な後に、**nfs_layout_nfsv41_files** カーネルが自動的に最初のマウントに読み込まれます。出力のマウントエントリーには、**minorversion=1 が含まれている必要があり** ます。次のコマンドを使用して、モジュールが読み込まれたことを確認します。

```
$ lsmod | grep nfs_layout_nfsv41_files
```

- Flex ファイルに対応するサーバーから、Flex ファイル機能で NFS 共有をマウントするには、NFS バージョン 4.2 以降を使用します。

```
# mount -t nfs -o v4.2 server:/remote-export /local-directory
```

nfs_layout_flexfiles モジュールが読み込まれていることを確認します。

```
$ lsmod | grep nfs_layout_flexfiles
```

関連情報

pNFS の詳細は、http://www.pnfs.com を参照してください。

8.10. NFS での PNFS SCSI レイアウトの有効化

データのアクセスに pNFS SCSI レイアウトを使用するように、NFS サーバーおよびクライアントを設定できます。pNFS SCSI は、ファイルへの長期接続のシングルクライアントアクセスに伴うユースケースで利点があります。

前提条件

- クライアントとサーバーの両方で、SCSI コマンドを同じブロックデバイスに送信する必要があります。つまり、ブロックデバイスは共有 SCSI バス上になければなりません。
- ブロックデバイスに XFS ファイルシステムが含まれている必要があります。
- SCSI デバイスは、SCSI-3 Primary Commands 仕様で説明されているように、SCSI Persistent Reservation に対応している必要があります。

8.10.1. pNFS SCSI レイアウト

SCSI レイアウトは、pNFS ブロックレイアウトの作業に基づいています。このレイアウトは、SCSI デバイス全体に定義されます。これには、SCSI 永続予約に対応する必要がある論理ユニット (lUS) として、固定サイズのブロックが連続的に含まれています。LU デバイスは、SCSI デバイスの識別子で識別されます。

pNFS SCSI は、ファイルへの長期接続のシングルクライアントアクセスのユースケースで適切に実行されます。例として、メールサーバーまたはクラスターを格納している仮想マシンなどが挙げられます。

クライアントとサーバーとの間の操作

NFS クライアントがファイルから読み取るか、またはファイルに書き込むと、クライアントは **LAYOUTGET** 操作を実行します。サーバーは、SCSI デバイスのファイルの場所に反応します。クライアントは、使用する SCSI デバイスを判別するために **GETDEVICEINFO** の追加操作が必要になる場合があります。これらの操作が正しく機能すると、クライアントは、**READ** 操作および **WRITE** 操作をサーバーに送信する代わりに、SCSI デバイスに直接 I/O 要求を発行することができます。

クライアント間のエラーまたは競合により、サーバーがレイアウトを再び呼び出したり、クライアントにレイアウトを発行しなくなることがあります。この場合、クライアントは SCSI デバイスに I/O 要求を直接送信するのではなく、サーバーへの **READ** 操作および **WRITE** 操作を発行するようにフォール

バックします。

操作を監視するには、「pNFS SCSI レイアウト機能の監視」 を参照してください。

デバイスの予約
pNFS SCSI は、予約の割り当てを通じてフェンシングを処理します。サーバーがレイアウトをクライアントに発行する前に、SCSI デバイスを予約して、登録したクライアントのみがデバイスにアクセスできるようにします。クライアントが、そのSCSI デバイスに対してコマンドを実行できても、そのデバイスに登録されていない場合は、クライアントからの多くの操作が、そのデバイス上で失敗します。たとえば、サーバーが、そのデバイスのレイアウトをクライアントに送信していないと、クライアントの blkid コマンドは、XFS ファイルシステムの UUID を表示できません。

サーバーは、独自の永続予約を削除しません。これにより、クライアントやサーバーの再起動時に、デバイスのファイルシステム内のデータが保護されます。SCSI デバイスを他の目的で使用するには、NFS サーバーで、手動で永続予約を削除する必要があります。

8.10.2. pNFS と互換性がある SCSI デバイスの確認

この手順では、SCSI デバイスが pNFS SCSI レイアウトに対応しているかどうかを確認します。

前提条件

- 以下のコマンドで、sg3-utils パッケージがインストールされている。

```
# yum install sg3_utils
```

手順8.4 pNFS と互換性がある SCSI デバイスの確認

- サーバーおよびクライアントの両方で、適切なSCSI デバイスサポートを確認します。

```
# sg_persist --in --report-capabilities --verbose path-to-scsi-device
```

Persist Through Power Loss Active（**PTPL_A**）ビットが設定されていることを確認します。

例8.8 pNFS SCSI をサポートする SCSI デバイス

以下は、pNFS SCSI をサポートする SCSI デバイスに対する **sg_persist** 出力の例です。**PTPL_A** ビットは **1** を報告します。

```
    inquiry cdb: 12 00 00 00 24 00
    Persistent Reservation In cmd: 5e 02 00 00 00 00 00 20 00 00
  LIO-ORG   block11          4.0
  Peripheral device type: disk
Report capabilities response:
  Compatible Reservation Handling(CRH): 1
  Specify Initiator Ports Capable(SIP_C): 1
  All Target Ports Capable(ATP_C): 1
  Persist Through Power Loss Capable(PTPL_C): 1
  Type Mask Valid(TMV): 1
  Allow Commands: 1
  Persist Through Power Loss Active(PTPL_A): 1
    Support indicated in Type mask:
      Write Exclusive, all registrants: 1
      Exclusive Access, registrants only: 1
```

```
Write Exclusive, registrants only: 1
Exclusive Access: 1
Write Exclusive: 1
Exclusive Access, all registrants: 1
```

関連情報

- sg_persist(8) の man ページ

8.10.3. サーバーでの pNFS SCSI の設定

この手順では、NFS サーバーが pNFS SCSI レイアウトをエクスポートするように設定します。

手順8.5 サーバーでの pNFS SCSI の設定

1. サーバーで、SCSI デバイスで作成した XFS ファイルシステムをマウントします。
2. NFS バージョン 4.1 以降をエクスポートするように NFS サーバーを設定します。**/etc/nfs.conf** ファイルの **[nfsd]** セクションに以下のオプションを設定します。

```
[nfsd]

vers4.1=y
```

3. **pnfs** オプションを使用して、NFS で XFS ファイルシステムをエクスポートするように NFS サーバーを設定します。

例8.9 pNFS SCSI をエクスポートする /etc/exports のエントリー

/etc/exports 設定ファイルの以下のエントリーは、**/exported/directory/** にマウントされているファイルシステムを、pNFS SCSI レイアウトとして **allowed.example.com** クライアントにエクスポートします。

```
/exported/directory allowed.example.com(pnfs)
```

関連情報

- NFS サーバーの設定に関する詳細は、「NFS サーバーの設定」 を参照してください。

8.10.4. クライアントでの pNFS SCSI の設定

この手順では、pNFS SCSI レイアウトをマウントするように NFS クライアントを設定します。

前提条件

- NFS サーバーは、pNFS SCSI で XFS ファイルシステムをエクスポートするように設定されています。「サーバーでの pNFS SCSI の設定」 を参照してください。

手順8.6 クライアントでの pNFS SCSI の設定

- クライアントで、NFS バージョン 4.1 以降を使用して、エクスポートした XFS ファイルシステムをマウントします。

```
# mount -t nfs -o nfsvers=4.1 host:/remote/export /local/directory
```

NFS なしで XFS ファイルシステムを直接マウントしないでください。

関連情報

- NFS 共有のマウントの詳細は、「NFS クライアントの設定」 を参照してください。

8.10.5. サーバーでの pNFS SCSI 予約の解放

この手順では、NFS サーバーが SCSI デバイスを維持している永続的な予約を解放します。これにより、pNFS SCSI をエクスポートする必要がなくなったら、SCSI デバイスを別の目的で使用できるようになります。

サーバーから予約を削除する必要があります。別の IT Nexus から削除することはできません。

前提条件

- 以下のコマンドで、sg3-utils パッケージがインストールされている。

```
# yum install sg3_utils
```

手順8.7 サーバーでの pNFS SCSI 予約の解放

1. サーバーで、既存の予約をクエリーします。

```
# sg_persist --read-reservation path-to-scsi-device
```

例8.10 /dev/sda での予約のクエリー

```
# sg_persist --read-reservation /dev/sda

  LIO-ORG   block_1          4.0
  Peripheral device type: disk
  PR generation=0x8, Reservation follows:
    Key=0x100000000000000
    scope: LU_SCOPE,  type: Exclusive Access, registrants only
```

2. サーバーにある既存の登録を削除します。

```
# sg_persist --out \
            --release \
            --param-rk=reservation-key \
            --prout-type=6 \
            path-to-scsi-device
```

例8.11 /dev/sda にある予約の削除

```
# sg_persist --out \
              --release \
              --param-rk=0x100000000000000 \
              --prout-type=6 \
```

```
        /dev/sda

LIO-ORG   block_1          4.0
Peripheral device type: disk
```

関連情報

- *sg_persist(8) の man ページ*

8.10.6. pNFS SCSI レイアウト機能の監視

pNFS クライアントとサーバーで、pNFS SCSI 操作が適切に行われることを犠牲にしているかどうか、通常の NFS 操作にフォールバックするかどうかを監視できます。

前提条件

- *pNFS SCSI クライアントとサーバーが設定されている。*

8.10.6.1. nfsstat を使用したサーバーからの pNFS SCSI 操作のチェック

*この手順では、**nfsstat** ユーティリティーを使用して、サーバーからの pNFS SCSI 操作を監視します。*

手順8.8 nfsstat を使用したサーバーからの pNFS SCSI 操作のチェック

1. *サーバーから操作サービスを監視します。*

```
# watch --differences \
      "nfsstat --server | egrep --after-context=1 read\|write\|layout"

Every 2.0s: nfsstat --server | egrep --after-context=1 read\|write\|layout

putrootfh    read         readdir      readlink     remove  rename
2         0% 0         0% 1         0% 0         0% 0         0%
--
setcltidconf verify    write        rellockowner bc_ctl  bind_conn
0         0% 0         0% 0         0% 0         0% 0         0%
--
getdevlist   layoutcommit layoutget    layoutreturn secinfononam sequence
0         0% 29        1% 49        1% 5         0% 0         0% 2435     86%
```

2. *クライアントとサーバーは、以下の場合に pNFS SCSI 操作を使用します。*

   - ***layoutget** カウンター、**layoutreturn** カウンター、および **layoutcommit** カウンターの増分値。これは、サーバーがレイアウトを提供することを意味します。*
   - *サーバーの **読み取り** および **書き込み** カウンターはインクリメントしません。これは、クライアントが SCSI デバイスに直接 I/O 要求を実行していることを意味します。*

8.10.6.2. mountstats を使用したクライアントからの pNFS SCSI 操作のチェック

*この手順では、**/proc/self/mountstats** ファイルを使用して、クライアントからの pNFS SCSI 操作を監視します。*

手順8.9 mountstats を使用したクライアントからの pNFS SCSI 操作のチェック

1. マウントごとの操作カウンターをリスト表示します。

```
# cat /proc/self/mountstats \
      | awk /scsi_lun_0/,/^$/ \
      | egrep device\|READ\|WRITE\|LAYOUT

device 192.168.122.73:/exports/scsi_lun_0 mounted on /mnt/rhel7/scsi_lun_0 with fstype
nfs4 statvers=1.1
   nfsv4:
bm0=0xfdffbfff,bm1=0x40f9be3e,bm2=0x803,acl=0x3,sessions,pnfs=LAYOUT_SCSI
           READ: 0 0 0 0 0 0 0 0
          WRITE: 0 0 0 0 0 0 0 0
      READLINK: 0 0 0 0 0 0 0 0
        READDIR: 0 0 0 0 0 0 0 0
     LAYOUTGET: 49 49 0 11172 9604 2 19448 19454
   LAYOUTCOMMIT: 28 28 0 7776 4808 0 24719 24722
   LAYOUTRETURN: 0 0 0 0 0 0 0 0
    LAYOUTSTATS: 0 0 0 0 0 0 0 0
```

2. 結果は以下のようになります。

   - **LAYOUT** 統計は、クライアントとサーバーが pNFS SCSI 操作を使用する要求を示します。
   - **READ** および **WRITE** 統計は、クライアントとサーバーが NFS 操作にフォールバックする要求を示します。

8.11. NFS のリファレンス

NFS サーバーの管理は難しい課題となる場合があります。本章では言及していませんが、NFS 共有のエクスポートやマウントに利用できるオプションは多数あります。詳細は、以下のソースを参照してください。

インストールされているドキュメント

- **man mount** - NFS サーバーとクライアント両方のマウントオプションに関する包括的な説明が含まれています。
- **man fstab**: システムの起動時にファイルシステムをマウントするために使用される **/etc/fstab** ファイルの形式の詳細を提供します。
- **man nfs** - NFS 固有のファイルシステムのエクスポートとマウントオプションの詳細を提供します。
- **man exports** - NFS ファイルシステムのエクスポート時に **/etc/exports** ファイルで使用される一般的なオプションを表示します。

便利な Web サイト

- http://linux-nfs.org – プロジェクトの更新状況を確認できる開発者向けの最新サイトです。
- http://nfs.sourceforge.net/ – 開発者向けのホームページで、少し古いですが、役に立つ情報が多数掲載されています。

- http://www.citi.umich.edu/projects/nfsv4/linux/ – Linux 2.6 カーネル用 NFSv4 のリソースです。
- http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.111.4086 – NFS バージョン 4 プロトコルの機能および拡張機能について記載しているホワイトペーパーです。

関連書籍

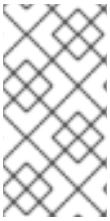
- 『Managing NFS and NIS』(Hal Stern、Mike Eisler および Ricardo Labiaga 著、O'Reilly & Associates 出版) – 利用可能な各種の NFS エクスポートやマウントオプションについて記載している優れた参考ガイドです。
- 『NFS Illustrated』(Brent Callaghan 著、Addison-Wesley Publishing Company 出版): NFS と他のネットワークファイルシステムとの比較、NFS 通信がどのように発生するかなどが詳細に紹介されています。

---

[1] **rpcbind** サービスは、**portmap** に代わるものです。これは、Red Hat Enterprise Linux の以前のバージョンで、RPC プログラム番号を IP アドレスのポート番号の組み合わせにマッピングするために使用されていました。詳細は、「必要なサービス」 を参照してください。

第9章 サーバーメッセージブロック(SMB)

Server Message Block (SMB) プロトコルは、アプリケーション層のネットワークプロトコルを実装します。これは、ファイル共有や共有プリンターなど、サーバー上のリソースにアクセスするために使用されます。SMB は、Microsoft Windows にはデフォルトで実装されています。Red Hat Enterprise Linux を実行している場合は、Samba を使用して SMB 共有を提供し、**cifs-utils** ユーティリティーを使用してリモートサーバーから SMB 共有をマウントします。



注記

SMB のコンテキストでは、SMB ダイアレクトである CIFS (Common Internet File System) プロトコルについて読むことがあります。SMB プロトコルと CIFS プロトコルの両方がサポートされており、SMB 共有と CIFS 共有のマウントに関連するカーネルモジュールとユーティリティーはどちらも **cifs** という名前を使用します。

9.1. SMB 共有の提供

『Red Hat System Administrator's Guide』 の 『Samba』 を参照してください。

9.2. SMB 共有のマウント

Red Hat Enterprise Linux では、カーネルの **cifs.ko** ファイルシステムモジュールが SMB プロトコルに対応します。ただし、SMB 共有をマウントして操作するには、cifs-utils もインストールする必要があります。

```
# yum install cifs-utils
```

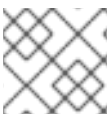
cifs-utils パッケージには、以下を行うユーティリティーがあります。

- SMB 共有と CIFS 共有をマウントする
- カーネルのキーリングで、NT Lan Manager (NTLM) の認証情報を管理する
- SMB 共有および CIFS 共有のセキュリティー記述子で、アクセス制御リスト (ACL) を設定して、表示する

9.2.1. 対応している SMB プロトコルのバージョン

cifs.ko カーネルモジュールは、以下の SMB プロトコルバージョンをサポートします。

- SMB 1
- SMB 2.0
- SMB 2.1
- SMB 3.0



注記

プロトコルのバージョンによっては、一部の SMB 機能しか実装されていません。

9.2.1.1. UNIX 拡張機能のサポート

Samba は、SMB プロトコルの **CAP_UNIX** 機能ビットを使用して UNIX 拡張機能を提供します。これらの拡張機能は、**cifs.ko** カーネルモジュールでも対応しています。ただし、Samba とカーネルモジュールはいずれも、SMB 1 プロトコルでのみ UNIX 拡張機能に対応します。

UNIX 拡張機能を使用するには、以下の手順を実行します。

1. **/etc/samba/smb.conf** ファイルの **[global]** セクションの **server min protocol** オプションを **NT1** に設定します。これは Samba サーバーのデフォルトです。

2. mount コマンドに **-o vers=1.0** オプションを指定し、SMB 1 プロトコルを使用して共有を **マウント** します。以下に例を示します。

   ```
   mount -t cifs -o vers=1.0,username=user_name //server_name/share_name /mnt/
   ```

   デフォルトで、カーネルモジュールは、SMB 2 またはサーバーでサポートされている最新のプロトコルバージョンを使用します。**-o vers=1.0** オプションを **mount** コマンドに渡すと、UNIX 拡張機能の使用に必要な SMB 1 プロトコルがカーネルモジュールで使用されることが強制されます。

UNIX 拡張機能が有効になっているかどうかを確認するには、マウントされた共有のオプションを表示します。

```
# mount
...
//server/share on /mnt type cifs (...,unix,...)
```

マウントオプションのリストに **unix** エントリーが表示されている場合は、UNIX 拡張機能が有効になっています。

9.2.2. SMB 共有の手動マウント

SMB 共有を手動でマウントするには、**mount** ユーティリティーに **-t cifs** パラメーターを指定します。

```
# mount -t cifs -o username=user_name //server_name/share_name /mnt/
Password for user_name@//server_name/share_name:  ********
```

-o options パラメーターでは、ファイル共有のマウントに使用されるオプションを指定できます。詳細は、man ページの mount.cifs(8) の 「よく使用されるマウントオプション」 セクションおよび 『OPTIONS』 セクションを参照してください。

例9.1 暗号化された SMB 3.0 接続を使用した共有のマウント

暗号化された SMB 3.0 接続を介して **DOMAIN \管理者 ユーザーとして \\サーバー\サンプル ** 共有をマウントするには、次のコマンド **を** 実行します。

```
# mount -t cifs -o username=DOMAIN\Administrator,seal,vers=3.0 //server/example /mnt/
Password for user_name@//server_name/share_name:  ********
```

9.2.3. システム起動時の SMB 共有の自動マウント

システムの起動時に SMB 共有を自動的にマウントするには、共有のエントリーを **/etc/fstab** ファイルに追加します。以下に例を示します。

```
//server_name/share_name  /mnt  cifs  credentials=/root/smb.cred  0 0
```

重要

システムが自動的に共有をマウントできるようにするには、ユーザー名、パスワード、およびドメイン名を認証情報ファイルに保存する必要があります。詳細は、「認証情報ファイルを使用した SMB 共有への認証」 を参照してください。

/etc/fstab *ファイルの 4 番目のフィールドに、認証情報ファイルへのパスなどのマウントオプションを指定します。詳細は、man ページの mount.cifs(8) の 「よく使用されるマウントオプション」 セクションおよび 『OPTIONS』 セクションを参照してください。*

共有が正常にマウントされたことを確認する場合は、次のコマンドを実行します。

```
# mount /mnt/
```

9.2.4. 認証情報ファイルを使用した SMB 共有への認証

特定の状況では、管理者はユーザー名とパスワードを入力せずに共有をマウントします。これを実装するには、認証情報ファイルを作成します。以下に例を示します。

手順9.1 認証情報ファイルの作成

1. *~/**smb.cred** などのファイルを作成し、そのファイルのユーザー名、パスワード、およびドメイン名を指定します。*

   ```
   username=user_name
   password=password
   domain=domain_name
   ```

2. *所有者だけがファイルにアクセスできるようにパーミッションを設定します。*

   ```
   # chown user_name ~/smb.cred
   # chmod 600 ~/smb.cred
   ```

*これで、**credentials=file_name** マウントオプションを **mount** ユーティリティーに渡すか、**/etc/fstab** ファイルでこれを使用して、ユーザー名とパスワードの入力を求められることなく共有をマウントできるようになりました。*

9.2.5. マルチユーザー SMB マウントの実行

*共有をマウントするために指定した認証情報により、デフォルトでマウントポイントのアクセス権が決まります。たとえば、共有をマウントするときに **DOMAIN\example** ユーザーを使用すると、どのローカルユーザーが操作を実行するかに関係なく、共有に対するすべての操作がこのユーザーとして実行されます。*

*ただし特定の状況では、システムの起動時に管理者が自動的に共有をマウントしたい場合でも、ユーザーは自分の認証情報を使用して共有のコンテンツに対して操作を実行する必要があります。このとき、**multiuser** マウントオプションを使用すると、このシナリオを設定できます。*

重要

multiuser *を使用するには、* ***sec=security_type*** *マウントオプションを、認証情報ファイルの* ***krb5*** *オプションや* ***ntlmssp*** *オプションなど、非対話式の方法で認証情報の提供に対応するセキュリティータイプに追加で設定する必要があります。**「ユーザーとして共有へのアクセス」 を参照してください。*

root *ユーザーは、* ***multiuser*** *オプションと、共有のコンテンツへの最小限のアクセス権を持つアカウントを使用して共有をマウントします。通常のユーザーは、* ***cifscreds*** *ユーティリティーを使用して、現在のセッションのカーネルキーリングにユーザー名とパスワードを指定できます。マウントされた共有のコンテンツにユーザーがアクセスすると、カーネルは、共有のマウントに最初に使用されたものではなく、カーネルキーリングからの認証情報を使用します。*

multiuser *オプションを使用した共有のマウント*

システムの起動時に、 ***multiuser*** *オプションを使用して自動的に共有をマウントするには、次の手順を実行します。*

手順9.2 multiuser オプションを使用した /etc/fstab ファイルエントリーの作成

1. ***/etc/fstab*** *ファイルに共有のエントリーを作成します。以下に例を示します。*

```
//server_name/share_name  /mnt  cifs  multiuser,sec=ntlmssp,credentials=/root/smb.cred  0 0
```

2. *共有をマウントします。*

```
# mount /mnt/
```

システムの起動時に共有を自動的にマウントしない場合は、 ***-o multiuser,sec=security_type*** *を mount コマンドに渡して手動で* ***マウント*** *します。SMB 共有の手動マウントの詳細は、**「SMB 共有の手動マウント」 を参照してください。*

SMB 共有が ***multiuser*** *オプションを使用してマウントされているかどうかの確認*

multiuser *オプションを使用して共有がマウントされているかどうかを確認するには、次のコマンドを実行します。*

```
# mount
...
//server_name/share_name on /mnt type cifs (sec=ntlmssp,multiuser,...)
```

ユーザーとして共有へのアクセス

SMB 共有が ***multiuser*** *オプションを使用してマウントされている場合、ユーザーはサーバーの認証情報をカーネルのキーリングに提供できます。*

```
# cifscreds add -u SMB_user_name server_name
Password: ********
```

これで、ユーザーがマウントされた SMB 共有を含むディレクトリーで操作を実行すると、サーバーは、共有がマウントされたときに最初に使用されたものではなく、このユーザーのファイルシステムのパーミッションを適用します。

注記

複数のユーザーが、マウントされた共有で、自身の認証情報を使用して同時に操作を実行できます。

9.2.6. よく使用されるマウントオプション

SMB 共有をマウントすると、マウントオプションにより次のことが決まります。

- *サーバーとの接続がどのように確立されるか。たとえば、サーバーに接続するときに使用される SMB プロトコルバージョンはどれか。*
- *共有が、ローカルファイルシステムにどのようにマウントされるか。たとえば、複数のローカルユーザーが、サーバーのコンテンツにアクセスできるようにするために、システムがリモートファイルとディレクトリーのパーミッションを上書きする場合など。*

***/etc/fstab** ファイルの 4 番目のフィールド、または **mount** コマンドの **-o** パラメーターで複数のオプションを設定するには、コンマで区切ります。たとえば、手順9.2「**multiuser** オプションを使用した **/etc/fstab** ファイルエントリーの作成」 を参照してください。*

以下のリストは、頻繁に使用されるマウントオプションの概要を示しています。

表9.1 よく使用されるマウントオプション

オプション	説明
credentials=file_name	*認証情報ファイルへのパスを設定します。「認証情報ファイルを使用した SMB 共有への認証」 を参照してください。*
dir_mode=mode	*サーバーが CIFS UNIX 拡張機能をサポートしていない場合は、ディレクトリーモードを設定します。*
file_mode=mode	*サーバーが CIFS UNIX 拡張機能をサポートしていない場合は、ファイルモードを設定します。*
password=password	*SMB サーバーへの認証に使用されるパスワードを設定します。あるいは、**credentials** オプションを使用して認証情報ファイルを指定します。*
seal	*SMB 3.0 以降のプロトコルバージョンを使用した接続に対する暗号化サポートを有効にします。したがって、**seal** を **3.0** 以降に設定した **vers** マウントオプションと一緒に使用します。例9.1「暗号化された SMB 3.0 接続を使用した共有のマウント」を参照してください。*
sec=security_mode	***ntlmsspi** などのセキュリティーモードを設定して、NTLMv2 パスワードハッシュとパケット署名を有効にします。対応している値のリストは、man ページの mount.cifs(8) にあるオプションの説明を参照してください。 サーバーが **ntlmv2** セキュリティーモードに対応していない場合は、**sec=ntlmssp**（デフォルト）を使用します。セキュリティー上の理由から、安全でない **ntlm** セキュリティーモードは使用しないでください。*
username=user_name	*SMB サーバーへの認証に使用されるユーザー名を設定します。あるいは、**credentials** オプションを使用して認証情報ファイルを指定します。*

オプション	説明
vers=SMB_protocol_version	サーバーとの通信に使用される SMB プロトコルバージョンを設定します。

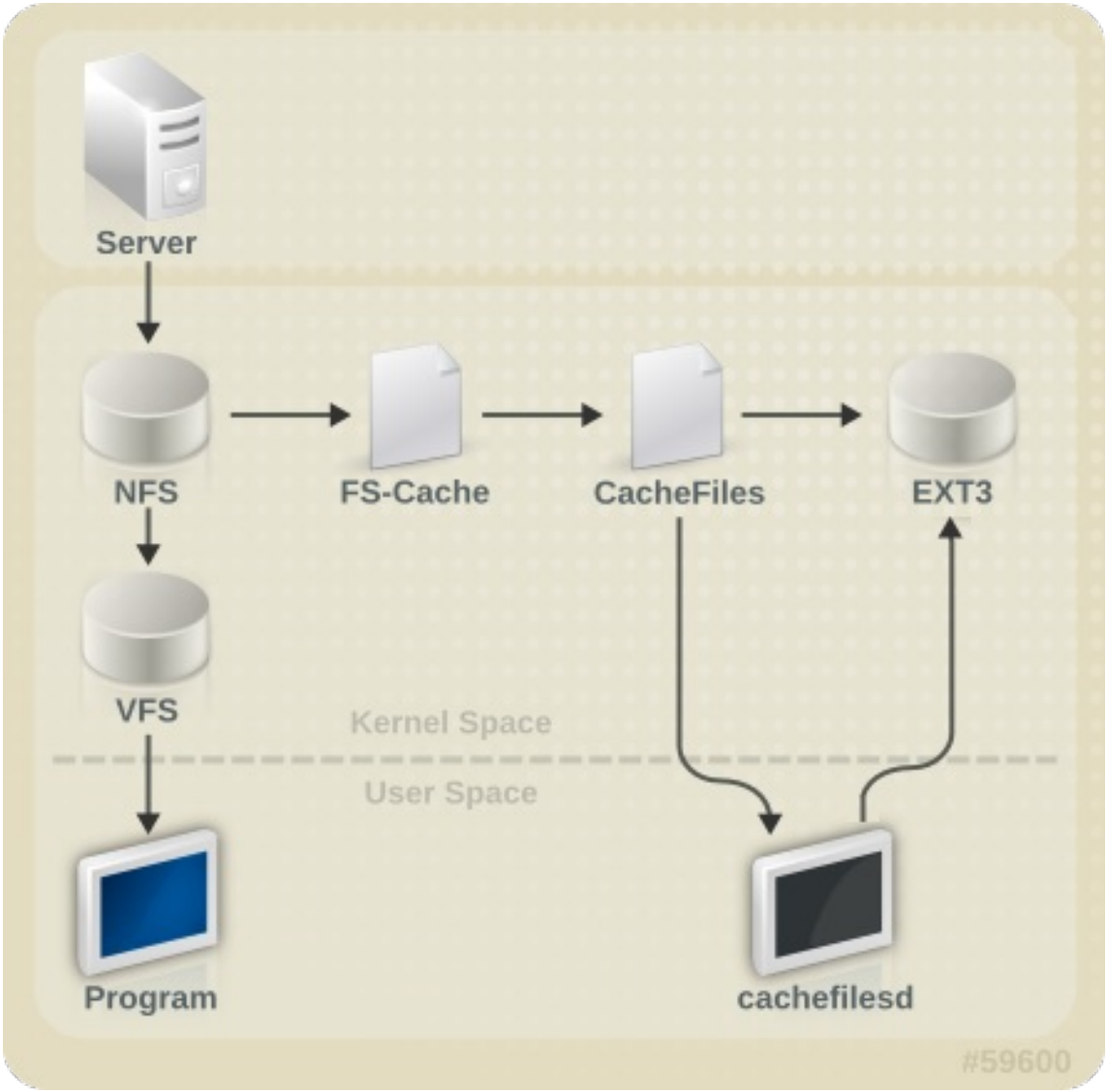
完全なリストは、man ページの mount.cifs(8) の 『OPTIONS』 セクションを参照してください。

第10章 FS-CACHE

FS-Cache は、ファイルシステムがネットワーク経由で取得したデータを取得し、ローカルディスクにキャッシュするために使用できる永続的なローカルキャッシュです。これは、ネットワーク経由でマウントされたファイルシステムからデータにアクセスするユーザーのネットワークトラフィックを最小限に抑えます (例: NFS)。

以下の図は、FS-Cache の仕組みの概要を示しています。

図10.1 FS-Cache の概要



[D]

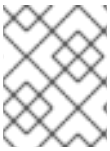
FS-Cache は、システムのユーザーおよび管理者が可能な限り透過的になるように設計されています。Solaris の **cachefs** とは異なり、サーバー上のファイルシステムは、オーバーマウントしたファイルシステムを作成せずにクライアントのローカルキャッシュと直接対話できます。NFS では、マウントオプションにより、FS-cache が有効になっている NFS 共有をマウントするようにクライアントに指示します。

FS-Cache はネットワーク上で機能するファイルシステムの基本操作を変更せず、単にデータをキャッシュできる永続的な場所でファイルシステムを提供するだけです。たとえば、クライアントは FS-Cache が有効になっているかどうかに関わらず、NFS 共有をマウントできます。さらに、キャッシュさ

れた NFS は、ファイルを部分的にキャッシュでき、事前に完全に読み取る必要がないため、(個別にまたは集合的に) キャッシュに収まらないファイルを処理できます。また、FS-Cache は、クライアントファイルシステムドライバーからキャッシュで発生するすべての I/O エラーも非表示にします。

キャッシングサービスを提供するために、FS-Cache にはキャッシュバックエンドが必要です。キャッシュバックエンドは、キャッシングサービスを提供するように設定されたストレージドライバーです(**cachefiles**)。この場合、FS-Cache には、キャッシュバックエンドとして **bmap** および拡張属性(ext3 など)をサポートするマウントされたブロックベースのファイルシステムが必要です。

FS-Cache は、ネットワークを介するかどうかに関係なく、ファイルシステムを任意にキャッシュすることはできません。共有ファイルシステムのドライバーを変更して、FS-Cache、データストレージ/取得、メタデータのセットアップと検証を操作できるようにする必要があります。FS-Cache では、永続性に対応するためにキャッシュされたファイルシステムの インデックスキー と 一貫性データ が必要になります。インデックスキーはファイルシステムオブジェクトをキャッシュオブジェクトに一致させ、一貫性データを使用してキャッシュオブジェクトが有効のままかどうかを判断します。



注記

Red Hat Enterprise Linux 7 では、cachefilesd パッケージはデフォルトでインストールされていないため、手動でインストールする必要があります。

10.1. パフォーマンスに関する保証

FS-Cache は、パフォーマンスの向上を保証 **しません** が、ネットワークの輻輳を回避することで一貫したパフォーマンスを保証します。キャッシュバックエンドを使用するとパフォーマンスが低下します。たとえば、キャッシュされた NFS 共有では、ネットワーク間のルックアップにディスクアクセスが追加されます。FS-Cache は可能な限り非同期となりますが、これができない同期パス (読み込みなど) があります。

たとえば、FS-Cache を使用して、他の方法では負荷のない GigE ネットワークを介して 2 台のコンピューター間の NFS 共有をキャッシュしても、ファイルアクセスのパフォーマンスが向上することはありません。代わりに、NFS 要求はローカルディスクからではなく、サーバーメモリーより早く満たされます。

したがって、FS-Cache の使用は、さまざまな要因における **妥協** です。たとえば、NFS トラフィックのキャッシュに FS-Cache を使用すると、クライアントは多少遅くなりますが、ネットワークの帯域幅を消費せずにローカルに読み取り要求を満たすことでネットワークおよびサーバーの読み込み負荷が大幅に削減されます。

10.2. キャッシュの設定

現在、Red Hat Enterprise Linux 7 は **cachefiles** キャッシュバックエンドのみを提供します。**cachefilesd** デーモンは **cachefiles** を開始し、管理します。**/etc/cachefilesd.conf** ファイルは、**cachefiles** がキャッシュサービスを提供する方法を制御します。

キャッシュバックエンドで最初に行うのはキャッシュとして使用するディレクトリーの設定です。これは、次のパラメーターを使用して設定します。

```
$ dir /path/to/cache
```

通常、キャッシュバックエンドディレクトリーは、以下のように **/etc/cachefilesd.conf** に **/var/cache/fscache** として設定されます。

```
$ dir /var/cache/fscache
```

キャッシュバックエンドディレクトリーを変更する場合、selinux コンテキストは **/var/cache/fscache** と同じである必要があります。

```
# semanage fcontext -a -e /var/cache/fscache /path/to/cache
# restorecon -Rv /path/to/cache
```

キャッシュを設定する際に、/path/to/cache をディレクトリー名に置き換えます。

注記

selinux コンテキストを設定するコマンドが機能しない場合は、以下のコマンドを使用します。

```
# semanage permissive -a cachefilesd_t
# semanage permissive -a cachefiles_kernel_t
```

FS-Cache は、**/path/to/cache** をホストするファイルシステムにキャッシュを保存します。ラップトップでは、root ファイルシステム(/)をホストファイルシステムとして使用することが推奨されますが、デスクトップマシンでは、キャッシュ専用のディスクパーティションをマウントする方が安全です。

FS-Cache のキャッシュバックエンドで必要とされる機能に対応するファイルシステムには、以下のファイルシステムの Red Hat Enterprise Linux 7 実装が含まれます。

- ext3 (拡張属性が有効)
- ext4
- Btrfs
- XFS

ホストファイルシステムはユーザー定義の拡張属性に対応する必要があります。FS-Cache はこの属性を使用して、整合性のメンテナンス情報を保存します。ext3 ファイルシステム（つまり **デバイス**）のユーザー定義の拡張属性を有効にするには、次のコマンドを実行します。

```
# tune2fs -o user_xattr /dev/device
```

または、ファイルシステムの拡張属性をマウント時に以下のように有効にすることもできます。

```
# mount /dev/device /path/to/cache -o user_xattr
```

キャッシュバックエンドは、キャッシュをホストしているパーティション上の一定の空き領域を維持することで動作します。空き領域を使用する他の要素に応じてキャッシュを増大および縮小し、root ファイルシステム (ラップトップなど) で安全に使用できるようにします。FS-Cache ではこの動作でデフォルト値を設定します。これは、キャッシュカリング制限 で設定できます。キャッシュカリング制限の設定に関する詳細は、「キャッシュカリング制限の設定」 を参照してください。

設定ファイルを配置したら、**cachefilesd** サービスを起動します。

```
# systemctl start cachefilesd
```

システムの起動時に **cachefilesd** が起動するように設定するには、root で以下のコマンドを実行します。

-

```
# systemctl enable cachefilesd
```

10.3. NFS でのキャッシュの使用

明示的に指示されない限り、NFS はキャッシュを使用しません。FS-Cache を使用するように NFS マウントを設定するには、**mount** コマンドに **-o fsc** オプションを追加します。

```
# mount nfs-share:/ /mount/point -o fsc
```

ファイルがダイレクト I/O または書き込みのために開いていない限り、**/mount/point** の下にあるファイルへのアクセスはすべてキャッシュを経由します。詳細は、「NFS でのキャッシュの制限」 を参照してください。NFS インデックスは NFS ファイルハンドルを使用してコンテンツをキャッシュします。**ファイル名ではなく**、ハードリンクされたファイルはキャッシュを正しく共有します。

NFS のバージョン 2、3、および 4 がキャッシュ機能に対応しています。ただし、各バージョンはキャッシュに異なるブランチを使用します。

10.3.1. キャッシュの共有

NFS キャッシュの共有には潜在的な問題がいくつかあります。キャッシュは永続的であるため、キャッシュ内のデータブロックは 4 つのキーのシーケンスでインデックス化されます。

- レベル 1: サーバーの詳細
- レベル 2: 一部のマウントオプション、セキュリティータイプ、FSID、識別子
- レベル 3: ファイルハンドル
- レベル 4: ファイル内のページ番号

スーパーブロック間での整合性の管理に関する問題を回避するには、データをキャッシュするすべての NFS のスーパーブロックに固有のレベル 2 キーを持たせます。通常、同じソースボリュームとオプションを持つ 2 つの NFS マウントはスーパーブロックを共有しているため、そのボリューム内に異なるディレクトリーをマウントする場合でもキャッシュを共有することになります。

例10.1 キャッシュの共有

次の 2 つの **マウント** コマンドを使用します。

```
mount home0:/disk0/fred /home/fred -o fsc

mount home0:/disk0/jim /home/jim -o fsc
```

/home/fred と **/home/jim** は同じオプションがあるため、スーパーブロックを共有する可能性が高くなります。特に NFS サーバー上の同じボリューム/パーティションからのものである場合(**home0**)。ここで、2 つの後続のマウントコマンドを示します。

```
mount home0:/disk0/fred /home/fred -o fsc,rsize=230

mount home0:/disk0/jim /home/jim -o fsc,rsize=231
```

この場合、**/home/fred** および **/home/jim** はレベル 2 キーの一部であるネットワークアクセスパラメーターが異なるため、スーパーブロックを共有しません。次のマウントシーケンスについても同じことが言えます。

> ***mount home0:/disk0/fred /home/fred1 -o fsc,rsize=230***
>
> ***mount home0:/disk0/fred /home/fred2 -o fsc,rsize=231***
>
> *ここでは、2 つのサブツリー(**/home/fred1** および **/home/fred2**)の内容が 2 回 キャッシュされます。*
>
> *スーパーブロックの共有を回避するもう 1 つの方法は、**nosharecache** パラメーターで明示的に禁止することです。同じ例を使用します。*
>
> ***mount home0:/disk0/fred /home/fred -o nosharecache,fsc***
>
> ***mount home0:/disk0/jim /home/jim -o nosharecache,fsc***
>
> *ただし、この場合は、レベル 2 キーに home0 **:/disk0/fred および home0:/disk 0/jim** を区別するものがないため、スーパーブロックの 1 つだけがキャッシュを使用できます。これに対処するには、少なくとも 1 つのマウントに 一意の識別子 を追加します。つまり、**fsc=unique-identifier** です。以下に例を示します。*
>
> ***mount home0:/disk0/fred /home/fred -o nosharecache,fsc***
>
> ***mount home0:/disk0/jim /home/jim -o nosharecache,fsc=jim***
>
> *ここで、一意の識別子 **jim** は **/home/jim** のキャッシュで使用されるレベル 2 キーに追加されます。*

10.3.2. NFS でのキャッシュの制限

- *ダイレクト I/O で共有ファイルシステムからファイルを開くと、自動的にキャッシュが回避されます。これは、この種のアクセスがサーバーに直接行なわれる必要があるためです。*
- *書き込みで共有ファイルシステムからファイルを開いても NFS のバージョン 2 およびバージョン 3 では動作しません。これらのバージョンのプロトコルは、クライアントが、別のクライアントからの同じファイルへの同時書き込みを検出するのに必要な整合性の管理に関する十分な情報を提供しません。*
- *ダイレクト I/O または書き込みのいずれかで共有ファイルシステムからファイルを開くと、キャッシュされたファイルのコピーがフラッシュされます。ダイレクト I/O や書き込みのためにファイルが開かれなくなるまで、FS-Cache はファイルを再キャッシュしません。*
- *さらに、FS-Cache の今回のリリースでは、通常の NFS ファイルのみをキャッシュします。FS-Cache はディレクトリー、シンボリックリンク、デバイスファイル、FIFO、ソケットを* **キャッシュしません**。

10.4. キャッシュカリング制限の設定

***cachefilesd** デーモンは、共有ファイルシステムからのリモートデータをキャッシュしてディスクの領域を解放することで機能します。これにより、利用可能な空き領域がすべて消費される可能性があり、ディスクがルートパーティションも格納している場合は問題になる可能性があります。これを制御するために、**cachefilesd** は古いオブジェクト（つまり最近アクセスされていないオブジェクト）をキャッシュから破棄して、一定量の空き領域を維持しようとします。この動作は キャッシュカリング と呼ばれます。*

*キャッシュカリングは、基盤となるファイルシステムで使用可能なブロックのパーセンテージとファイルのパーセンテージに基づいて行われます。**/etc/cachefilesd.conf** の設定で制御される制限は 6 つあります。*

brun N% (ブロックのパーセンテージ) , frun N% (ファイルのパーセンテージ)

キャッシュの空き領域と利用可能なファイルの数がこれらの制限を上回ると、カリングはオフになります。

bcull N% (ブロックのパーセンテージ), fcull N% (ファイルのパーセンテージ)

キャッシュの空き領域と利用可能なファイルの数がこれらの制限のいずれかを下回ると、カリング動作が開始します。

bstop N% (ブロックのパーセンテージ), fstop N% (ファイルのパーセンテージ)

キャッシュ内の使用可能な領域または使用可能なファイルの数がこの制限のいずれかを下回ると、カリングによってこれらの制限を超える状態になるまで、ディスク領域またはファイルのそれ以上の割り当ては許可されません。

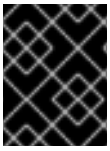
各設定の **N** のデフォルト値は以下の通りです。

- ***brun*/*frun*** - 10%
- ***bcull*/*fcull*** - 7%
- ***bstop*/*fstop*** - 3%

この設定を行う場合は、以下の条件を満たす必要があります。

- 0 DHE ***bstop*** < ***bcull*** < ***brun*** < 100
- 0 DHE ***fstop*** < ***fcull*** < ***frun*** < 100

これは、利用可能な領域と利用可能なファイルの割合であり、100 から、**df** プログラムで表示されるパーセンテージを引いたものではありません。



重要

カリングは、bxxx と fxxx のペアに同時に依存します。これらを別個に処理することはできません。

10.5. 統計情報

FS-Cache は一般的な統計情報も追跡します。この情報を表示するには、次を使用します。

```
# cat /proc/fs/fscache/stats
```

FS-Cache の統計にはディシジョンポイントとオブジェクトカウンターに関する情報が含まれます。詳細は、以下のカーネルドキュメントを参照してください。

/usr/share/doc/kernel-doc-version/Documentation/filesystems/caching/fscache.txt

10.6. FS-CACHE の参考資料

cachefilesd の詳細および設定方法は、man cachefilesd および ***man cachefilesd.conf*** を参照してください。その他にも、以下のカーネルドキュメントを参照してください。

- ***/usr/share/doc/cachefilesd-version-number/README***

- ***/usr/share/man/man5/cachefilesd.conf.5.gz***
- ***/usr/share/man/man8/cachefilesd.8.gz***

*設計の制約、利用可能な統計、機能の詳細など、FS-Cache に関する一般的な情報は、カーネルドキュメント **/usr/share/doc/kernel-doc-version/Documentation/filesystems/caching/fscache.txt** を参照してください。*

パート II. ストレージ管理

ストレージ管理のセクションは、Red Hat Enterprise Linux 7 におけるストレージに関する考慮事項から始まります。次に、パーティション、論理ボリューム管理、およびスワップパーティションに関する手順を説明します。続いて、ディスククォータと RAID システム、そして mount コマンド、volume_key、および acls の機能について説明します。その後は、SSD の調整、書き込みバリア、I/O 制限、およびディスクレスシステムが続きます。この後に、オンラインストレージに関する大きな章があり、最後にデバイスマッパーのマルチパスと仮想ストレージについて説明します。

第11章 ストレージをインストールする際の注意点

ストレージデバイスやファイルシステムの設定の多くはインストール時にしか実行することができません。ファイルシステムタイプなどの他の設定については、再フォーマットせずに変更できるのは特定の時点までになります。このようにストレージの設定については Red Hat Enterprise Linux 7 をインストールする前に慎重に計画する必要があります。

本章ではシステムのストレージ設定を計画する際の注意点について説明しています。インストール手順 (インストール時のストレージ設定を含む) は、Red Hat が提供する Installation Guide を参照してください。

サイズとストレージの制限に関して Red Hat が公式にサポートしているものについては、http://www.redhat.com/resourcelibrary/articles/articles-red-hat-enterprise-linux-6-technology-capabilities-and-limits の記事を参照してください。

11.1. 特に注意を要する事項について

本セクションでは、ストレージの設定で特に注意を要する事項について記載しています。

/home、/opt、/usr/local には別々のパーティションを用意する

今後システムをアップグレードする可能性がある場合は、**/home**、**/opt**、および **/usr/local** を別のデバイスに配置します。これにより、ユーザーデータやアプリケーションデータを保持しながら、オペレーティングシステムが含まれるデバイスやファイルシステムを再フォーマットできます。

IBM System Z における DASD デバイスと zFCP デバイス

IBM System Z のプラットフォームでは、DASD デバイスと zFCP デバイスは Channel Command Word (CCW) メカニズムで設定されます。CCW のパスをシステムに明示的に追加してからオンラインにする必要があります。DASD デバイスの場合、これは、起動コマンドラインまたは CMS 設定ファイル内で **DASD=** パラメーターとしてデバイス番号（またはデバイス番号の範囲）を一覧表示することを意味します。

zFCP デバイスの場合は、デバイス番号、論理ユニット番号 (LUN)、および ワールドワイドポート名 (WWPN) を記載する必要があります。zFCP が初期化されると CCW パスにマッピングが行われます。起動コマンドライン（または CMS 設定ファイル）の **FCP_x=** 行を使用すると、インストーラーにこの情報を指定できます。

LUKS を使用したブロックデバイスの暗号化

LUKS/**dm-crypt** を使用して暗号化用にブロックデバイスをフォーマットすると、そのデバイスに存在するフォーマットがすべて破棄されます。このため、まず暗号化するデバイスが存在する場合はそのデバイスを選択してください。次に、新しいシステムのストレージ設定をインストールプロセスの一部としてアクティブにします。

古い BIOS RAID メタデータ

ディスクから RAID メタデータを削除**せず** にファームウェア RAID 用に設定したシステムからディスクを移動すると、**Anaconda** がディスクを正しく検出できなくなる可能性があります。

警告

ディスクから RAID メタデータを削除または消去すると、保存データがすべて破棄される可能性があります。Red Hat では、これを実行する前に必ずバックアップを取っておくことを推奨します。

注記

*非推奨の **dmraid** を使用して RAID ボリュームを作成した場合は、**dmraid** ユーティリティーを使用して削除します。*

```
# dmraid -r -E /device/
```

*RAID デバイスの管理に関する詳細は、**man dmraid** および 18章RAID (Redundant Array of Independent Disks) を参照してください。*

iSCSI の検出および設定

iSCSI ドライブのプラグアンドプレイ検出の場合には、iBFT 起動が可能な ネットワークインターフェイスカード (NIC) のファームウェアで設定を行ってください。インストール時の iSCSI ターゲットの CHAP 認証がサポートされています。ただし、インストール時の iSNS 検出はサポートされていません。

FCoE の検出および設定

Fibre Channel over Ethernet (FCoE) ドライブのプラグアンドプレイ検出は、EDD で起動可能な NIC のファームウェアで設定を行ってください。

DASD

ダイレクトアクセスストレージデバイス (DASD) は、インストール時に追加または設定できません。このデバイスは CMS 設定ファイル内で指定してください。

DIF/DIX を有効にしているブロックデバイス

*DIF/DIX は、特定の SCSI ホストバスアダプターおよびブロックデバイスで提供されているハードウェアチェックサムの機能です。DIF/DIX が有効になっていると、ブロックデバイスが汎用ブロックデバイスとして使用されている場合にエラーが発生します。DIF/DIX チェックサムの計算後はバッファーされたデータが上書きされないようにするためのインターロックがバッファー書き込みパス内にないため、バッファーされた入出力または **mmap(2)** ベースの入出力が正常に動作しなくなります。*

*これにより、I/O が後でチェックサムエラーで失敗します。この問題は、すべてのブロックデバイス（またはファイルシステムベース）でバッファーされた I/O または **mmap (2**) I/O に共通するため、上書きによって生じるこれらのエラーを回避することはできません。*

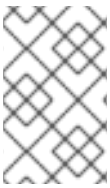
*そのため、DIF/DIX が有効になっているブロックデバイスは、**O_DIRECT** を使用するアプリケーションでのみ使用する必要があります。こうしたアプリケーションはローブロックデバイスを使用するはずです。または、ファイルシステムを介して **O_DIRECT** I/O のみを発行する限り、DIF/DIX が有効なブロッ*

クデバイスで XFS ファイルシステムを安全に使用することもできます。特定の割り当て動作を行う際にバッファーされた入出力にフォールバックを行わないファイルシステムは XFS のみです。

*DIF/DIX チェックサムの計算後に入出力データが変更しないようにするのは常にアプリケーションの役目となるため、DIF/DIX を使用できるアプリケーションを **O_DIRECT** 入出力および DIF/DIX ハードウェアでの使用を目的として設計されたアプリケーションに限ってください。*

第12章 ファイルシステム検査

ファイルシステムは、ファイルシステム固有のユーザー空間ツールを使用して、整合性をチェックし、必要に応じて修復できます。このツールは、通常 **fsck** ツールと呼ばれることが多く、**fsck** は、file system check を短くした名前になります。



注記

このようなファイルシステムチェッカーは、ファイルシステム全体でのメタデータの整合性のみを保証します。これらは、ファイルシステムに含まれる実際のデータを認識しないため、データリカバリーツールではありません。

ファイルシステムの不整合は、ハードウェアエラー、ストレージ管理エラー、ソフトウェアバグなどのさまざまな理由で発生する可能性があります。

最新のメタデータジャーナリングファイルシステムが一般的になる前は、システムがクラッシュしたり、電源が失われたりした場合は常に、ファイルシステムの検査が必要でした。これは、ファイルシステムの更新が中断し、不整合な状態が生じる可能性があったためです。その結果、ファイルシステムの検査は従来どおり、システムの起動時に **/etc/fstab** に一覧表示されている各ファイルシステムで実行されます。ジャーナリングファイルシステムの場合、ファイルシステムのメタデータジャーナリングはクラッシュ後も一貫性を保証するため、これは通常非常に短い操作になります。

ただし、ジャーナリングファイルシステムであっても、ファイルシステムの不整合や破損が生じることがあります。この場合は、ファイルシステムチェッカーを使用してファイルシステムを修復する必要があります。以下に、この手順を実行する際のベストプラクティスとその他の有用な情報について説明します。

重要

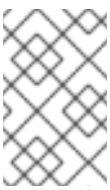
Red Hat は、マシンが起動しない場合、ファイルシステムが非常に大きい場合、またはファイルシステムがリモートストレージにある場合を除き、これを推奨しません。**/etc/fstab** の 6 番目のフィールドを **0** に設定すると、システムの起動時にファイルシステムの検査を無効にできます。

12.1. FSCK のベストプラクティス

通常、ファイルシステムの検査および修復のツールを実行すると、検出された不整合の少なくとも一部が自動的に修復されることが期待できます。場合によっては、重度にダメージを受けた inode やディレクトリーは、修復できない場合に破棄されることがあります。ファイルシステムが大きく変更する場合があります。予想外の、または好ましくない変更が永続的に行なわれないようにするには、以下の予防的な手順を実行します。

Dry run

ほとんどのファイルシステムチェッカーには、ファイルシステムを検査しますが修復しない操作モードがあります。このモードでは、チェッカーは、ファイルシステムを実際に変更することなく、検出したエラーと実行したアクションを出力します。

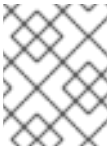


注記

整合性チェック後のフェーズでは、修復モードで実行していた場合に前のフェーズで修正されていた不整合が検出される可能性があるため、追加のエラーが出力される場合があります。

最初にファイルシステムイメージを操作する

ほとんどのファイルシステムは、メタデータのみを含むファイルシステムのスパースコピーであるメタデータイメージ の作成に対応しています。ファイルシステムチェッカーはメタデータのみを操作するため、このようなイメージを使用して、実際のファイルシステム修復のドライランを実行し、実際にどのような変更が行われるかを評価できます。変更が受け入れられる場合は、ファイルシステム自体で修復を実行できます。



注記

ファイルシステムが大幅に損傷している場合は、メタデータイメージの作成に関連して問題が発生する可能性があります。

サポート調査のためにファイルシステムイメージを保存します。

破損がソフトウェアのバグによるものである可能性がある場合、修復前のファイルシステムメタデータイメージは、サポート調査に役立つことがよくあります。修復前のイメージに見つかる破損のパターンは、根本原因の分析に役立つことがあります。

マウントされていないファイルシステムでのみ動作します。

ファイルシステムの修復は、マウントされていないファイルシステムでのみ実行する必要があります。ツールには、ファイルシステムへの単独アクセスが必要であり、それがないと追加の損傷が発生する可能性があります。ほとんどのファイルシステムツールは、修復モードでこの要件を適用しますが、一部のツールは、マウントされたファイルシステムでチェックのみモードだけをサポートします。チェックのみモードが、マウントされたファイルシステムで実行されていると、マウントされていないファイルシステムで実行しても見つからない疑わしいエラーが検出されることがあります。

ディスクエラー

*ファイルシステムの検査ツールは、ハードウェアの問題を修復できません。修復を正常に動作させるには、ファイルシステムが完全に読み取り可能かつ書き込み可能である必要があります。ハードウェアエラーが原因でファイルシステムが破損した場合は、まず **dd (8)** ユーティリティーなどを使用して、ファイルシステムを適切なディスクに移動する必要があります。*

12.2. FSCK のファイルシステム固有の情報

12.2.1. ext2、ext3、および ext4

*これらのファイルシステムはすべて、**e2fsck** バイナリーを使用して、ファイルシステムの検査と修復を実行します。ファイル名の **fsck.ext2**、**fsck.ext3**、および **fsck.ext4** はこの同じバイナリーへのハードリンクです。これらのバイナリーは、システムの起動時に自動的に実行し、その動作は確認されるファイルシステムと、そのファイルシステムの状態によって異なります。*

完全なファイルシステムの検査および修復は、メタデータジャーナリングファイルシステムではない ext2 や、ジャーナルのない ext4 ファイルシステムに対して呼び出されます。

メタデータジャーナリング機能のある ext3 ファイルシステムおよび ext4 ファイルシステムの場合、ジャーナルはユーザー空間で再生され、バイナリーは終了します。ジャーナル再生により、クラッシュ後に一貫したファイルシステムが確保されるため、これがデフォルトのアクションになります。

*このファイルシステムで、マウント中にメタデータの不整合が生じると、その事実がファイルシステムのスーパーブロックに記録されます。**e2fsck** が、このようなエラーでファイルシステムがマークされていることを検出すると、**e2fsck** はジャーナル (がある場合) の再生後にフルチェックを実行します。*

e2fsck は、**-p** オプションが指定されていない場合に、実行時にユーザー入力を要求することがあります。**-p** オプションは **e2fsck** に対して、安全に実行できるすべての修復を自動的に実行するように指示します。ユーザーの介入が必要な場合は、**e2fsck** が出力内の未修正の問題を示し、このステータスを終了コードに反映させます。

一般的に使用される **e2fsck** ランタイムオプションは次のとおりです。

-n

非変更モード。チェックのみの操作。

-b スーパーブロック

プライマリーシステムが破損している場合は、別のスーパーブロックのブロック番号を指定します。

-f

スーパーブロックに記録されたエラーがなくても、フルチェックを強制実行します。

-j *journal-dev*

外部のジャーナルデバイス (ある場合) を指定します。

-p

ユーザー入力のないファイルシステムを自動的に修復または preen (修復) する

-y

すべての質問に yes の回答を想定する

e2fsck のすべてのオプションは、 **e2fsck (8)** man ページで指定されています。

以下の 5 つの基本フェーズは、実行中の **e2fsck** によって実行されます。

1. Inode、ブロック、およびサイズのチェック。
2. ディレクトリー構造のチェック。
3. ディレクトリー接続のチェック。
4. 参照数のチェック。
5. グループサマリー情報のチェック。

e2image (8)ユーティリティーを 使用して、診断またはテストの目的で、修復前のメタデータイメージを作成できます。**-r** オプションは、ファイルシステム自体と同じサイズのスパースファイルを作成するために、テスト目的で使用する必要があります。その後、**e2fsck** は作成されたファイルで直接操作できます。イメージが診断用にアーカイブまたは提供される場合は、**-Q** オプションを指定する必要があります。これにより、送信に適したよりコンパクトなファイル形式が作成されます。

12.2.2. XFS

ブート時に修復が自動的に行なわれません。ファイルシステムの検査または修復を開始するには、**xfs_repair** ツールを使用します。

注記

*xfsprogs パッケージには **fsck.xfs** バイナリーがありますが、これは、システムの起動時に **fsck.ファイルシステム** バイナリーを検索する initscripts を満たすためにのみ存在します。**fsck.xfs** は、即座に終了コード 0 で終了します。*

*古い xfsprogs パッケージには、**xfs_check** ツールが同梱されています。このツールは非常にスピードが遅く、大きなファイルシステムに対して十分な拡張性がありません。そのため、**xfs_repair -n** が優先されるため非推奨になりました。*

***xfs_repair** を動作させるには、ファイルシステムのクリーンなログが必要です。ファイルシステムがクリーンな状態でアンマウントされなかった場合は、**xfs_repair** を使用する前にマウントおよびアンマウントする必要があります。ログが破損していて再生できない場合、**-L** オプションを使用してログをゼロにすることができます。*

重要

***-L** オプションは、ログを再生できない場合にのみ使用する必要があります。このオプションを使用すると、ログ内のすべてのメタデータの更新が破棄され、結果としてさらに不整合が発生します。*

***-n** オプションを使用して、Dry Run で、チェックのみモードの **xfs_repair** を実行することができます。このオプションを指定した場合は、ファイルシステムに変更が加えられません。*

***xfs_repair** には非常にいくつかのオプションがあります。共通に使用されるオプションには以下が含まれます。*

-n

非変更モードです。チェックのみの操作。

-L

ゼロメタデータログ。マウントによってログを再生できない場合にのみ使用します。

***-m** maxmem*

最大 MB の実行時に使用されるメモリーを制限します。必要な最小メモリーの概算を出すために 0 を指定できます。

***-l** logdev*

外部ログデバイス (ある場合) を指定します。

*xfs_repair のすべてのオプションは、**xfs_repair (8)** man ページで指定されています。*

*以下の 8 つの基本フェーズは、実行中の **xfs_repair** によって実行されます。*

1. *Inode および inode ブロックマップ (アドレス指定) のチェック。*
2. *Inode 割り当てマップのチェック。*
3. *Inode サイズのチェック。*
4. *ディレクトリーのチェック。*
5. *パス名のチェック。*

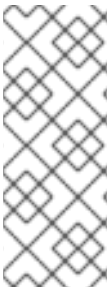
6. リンク数のチェック。
7. フリーマップのチェック。
8. スーパーブロックチェック。

詳細は、**xfs_repair (8)** man ページを参照してください。

xfs_repair は対話的ではありません。すべての操作は、ユーザーの入力なしに自動的に実行されます。

診断またはテスト目的で、修復前にメタデータイメージを作成する必要がある場合は、**xfs_metadump (8)** ユーティリティーおよび **xfs_mdrestore (8)** ユーティリティーを使用できます。

12.2.3. Btrfs



注記

Btrfs は、Red Hat Enterprise Linux 7 ではテクノロジープレビューとして利用できますが、Red Hat Enterprise Linux 7.4 リリース以降では非推奨になりました。Red Hat Enterprise Linux の将来のメジャーリリースで削除される予定です。

詳細は、Red Hat Enterprise Linux 7.4 リリースノートの 非推奨の機能 を参照してください。

btrfsck は、btrfs ファイルシステムの確認および修復に使用されます。このツールは開発の初期段階であるため、すべてのタイプのファイルシステムの破損を検出または修復できるわけではありません。

デフォルトでは、**btrfsck** はファイルシステムを変更しません。つまり、デフォルトではチェックのみモードを実行します。修復が必要な場合は、**--repair** オプションを指定する必要があります。

以下の 3 つの基本フェーズは、実行中の **btrfsck** によって実行されます。

1. エクステントのチェック。
2. ファイルシステムの root チェック。
3. ルートの参照数のチェック。

btrfs-image (8)ユーティリティーを 使用して、診断またはテストの目的で、修復前のメタデータイメージを作成できます。

第13章 PARTITIONS

注記

ブロックデバイスでパーティションを使用するメリットとデメリットの概要は、ナレッジベースの記事 https://access.redhat.com/solutions/163853 を参照してください。

parted *ユーティリティーを使用すると、以下が可能になります。*

- *既存パーティションテーブルの表示*
- *既存パーティションのサイズ変更*
- *空き領域または他のハードドライブからのパーティションの追加*

parted パッケージは、Red Hat Enterprise Linux 7 にデフォルトでインストールされます。 **parted** *を起動するには、root でログインし、次のコマンドを入力します。*

```
# parted /dev/sda
```

/dev/sda を、設定するドライブのデバイス名に置き換えます。

使用中のデバイスでのパーティションの操作

デバイスを使用しない場合は、そのデバイスのパーティションはどれもマウントできず、そのデバイスのスワップ領域も有効にできません。

パーティションの削除またはサイズ変更を行う場合は、パーティションが存在するデバイスが使用中でない必要があります。

使用中のデバイスに新しいパーティションを作成することもできますが、推奨されません。

パーティションテーブルの変更

同じディスク上の別のパーティションの使用中にパーティションテーブルを変更することは、カーネルがパーティションテーブルを再読み取りできないため、通常は推奨されません。このため、変更は実行中のシステムには適用されません。このような状況では、システムを再起動するか、次のコマンドを実行して、システムに新しいパーティションまたは変更したパーティションを登録します。

```
# partx --update --nr partition-number disk
```

現在使用しているディスクを修正する最も簡単な方法は、以下のとおりです。

1. *システムディスクの場合など、ディスクのパーティションのマウントを解除できない場合は、レスキューモードでシステムを起動します。*
2. *ファイルシステムをマウントするように求められたら、* **Skip** *を選択します。*

ドライブに使用中のパーティションが含まれていない場合は、ファイルシステムのアンマウントが解除されないようにするシステムプロセスがないか、またはアンマウントして、 **umount** *コマンドでパーティションをアンマウントし、* **swapoff** *コマンドを使用して、ハードドライブ上のすべてのスワップ領域をオフにできます。*

一般的に使用される **parted** *コマンドを確認するには、表13.1「* **parted** *コマンド」 を参照してください。*

重要

parted ユーティリティーを使用してファイルシステムを作成しないでください。代わりに mkfs ツールを使用してください。

表13.1 parted コマンド

コマンド	説明
help	利用可能なコマンドのリストを表示します。
mklabel label	パーティションテーブル用のディスクラベルを作成します。
mkpart part-type [fs-type] start-mb end-mb	新しいファイルシステムを作成せずに、パーティションを作成します。
name minor-num name	Mac と PC98 のディスクラベル用のみのパーティションに名前を付けます。
print	パーティションテーブルを表示します。
quit	partedを終了します。
rescue*start-mb end-mb*	start-mb から end-mb へ、消失したパーティションを復旧します。
rm minor-num	パーティションを削除します。
select device	設定する別のデバイスを選択します。
set minor-num flag state	パーティションにフラグを設定します。state はオンまたはオフのいずれかになります。
toggle [NUMBER [FLAG]	パーティションNUMBER 上のFLAG の状態を切り替えます。
unit UNIT	デフォルトのユニットをUNIT に設定します。

13.1. パーティションテーブルの表示

パーティションテーブルを表示するには、以下を実行します。

1. parted を起動します。
2. パーティションテーブルを表示するには、次のコマンドを使用します。

```
(parted) print
```

以下のような表が表示されます。

例13.1 パーティションテーブル

```
Model: ATA ST3160812AS (scsi)
Disk /dev/sda: 160GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start   End    Size    Type      File system  Flags
 1      32.3kB  107MB  107MB   primary   ext3         boot
 2      107MB   105GB  105GB   primary   ext3
 3      105GB   107GB  2147MB  primary   linux-swap
 4      107GB   160GB  52.9GB  extended  root
 5      107GB   133GB  26.2GB  logical   ext3
 6      133GB   133GB  107MB   logical   ext3
 7      133GB   160GB  26.6GB  logical   lvm
```

以下に、パーティションテーブルについて説明します。

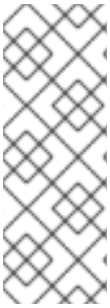
- **Model: ATA ST3160812AS (scsi)**: *ディスクの種類、製造元、モデル番号、およびインターフェイスについて説明しています。*
- **Disk /dev/sda: 160GB**: *ブロックデバイスへのファイルパスと、ストレージキャパシティーを表示しています。*
- **Partition Table: msdos**: *ディスクラベルのタイプを表示しています。*
- *パーティションテーブルでは、**Number** はパーティション番号です。たとえば、マイナー番号 1 のパーティションは **/dev/sda1** に対応します。**Start** および **End** の値はメガバイト単位です。有効な **タイプ** は、metadata、free、primary、extended、または logical です。**File system** はファイルシステムのタイプです。**Flags** 列には、パーティションに設定されたフラグが一覧表示されます。利用可能なフラグは、boot、root、swap、hidden、raid、lvm、または lba です。*

*パーティションテーブルの **File system** は、以下のいずれかになります。*

- *ext2*
- *ext3*
- *fat16*
- *fat32*
- *hfs*
- *jfs*
- *linux-swap*
- *ntfs*
- *reiserfs*
- *hp-ufs*
- *sun-ufs*

- *xfs*

デバイスのファイル ***システム*** *に値が表示されない場合は、ファイルシステムのタイプが不明であることを意味します。*



注記

parted *を再起動せずに別のデバイスを選択するには、以下のコマンドを使用して、/dev/sda を、選択するデバイスに置き換えます。*

```
(parted) select /dev/sda
```

デバイスのパーティションテーブルを表示または設定できます。

13.2. パーティションの作成

警告

使用中のデバイスに、パーティションを作成しないようにしてください。

手順13.1 パーティションの作成

1. *パーティションを作成する前に、レスキューモードで起動します (または、デバイス上のパーティションをアンマウントして、デバイス上のスワップ領域をすべてオフにします)。*
2. ***parted*** *を起動します。*

   ```
   # parted /dev/sda
   ```

   /dev/sda をパーティションを作成するデバイス名に置き換えます。
3. *現在のパーティションテーブルを表示し、十分な空き領域があるかどうかを確認します。*

   ```
   (parted) print
   ```

   十分な空き容量がない場合は、既存のパーティションのサイズを変更できます。詳細は、「fdisk でのパーティションのサイズ変更」 を参照してください。

   パーティションテーブルから、新しいパーティションの開始点と終了点、およびパーティションのタイプを決定します。プライマリーパーティションは、1 つのデバイス上に 4 つまで保有できます (この場合は拡張パーティションは含みません)。パーティションが 5 つ以上必要な場合は、プライマリーパーティションを 3 つ、拡張パーティションを 1 つにし、その拡張パーティションの中に複数の論理パーティションを追加します。ディスクパーティションの概要は、Red Hat Enterprise Linux 7 Installation Guide の付録 An Introduction to Disk Partitions を参照してください。
4. *パーティションを作成するには、以下を実行します。*

   ```
   (parted) mkpart part-type name fs-type start end
   ```

-

part-type を必要に応じて、primary、logical、または extended に置き換えます。

name を partition-name に置き換えます。GPT パーティションテーブルには name が必要です。

fs-type を、btrfs、ext2、ext3、ext4、fat16、fat32、hfs、hfs+、linux-swap、ntfs、reiserfs、または xfs のいずれかに置き換えます。fs-type はオプションです。

必要に応じて、start end をメガバイト単位で置き換えます。

たとえば、ハードドライブの 1024 メガバイトから 2048 メガバイトに ext3 ファイルシステムのプライマリーパーティションを作成するには、以下のコマンドを入力します。

```
(parted) mkpart primary 1024 2048
```

注記

代わりに **mkpartfs** コマンドを使用すると、パーティションの作成後にファイルシステムが作成されます。ただし、**parted** は ext3 ファイルシステムの作成をサポートしていません。したがって、ext3 ファイルシステムを作成する場合は、**mkpart** を使用して、後述するように **mkfs** コマンドでファイルシステムを作成します。

変更は **Enter** を押すとすぐに行われます。したがって、コマンドを確認してから確認してください。

5. パーティションテーブルを表示し、次のコマンドを使用して、作成されたパーティションのパーティションタイプ、ファイルシステムタイプ、サイズが、パーティションテーブルに正しく表示されていることを確認します。

```
(parted) print
```

新しいパーティションのマイナー番号も覚えておいてください。そのパーティション上のファイルシステムにラベルを付けることができます。

6. parted シェルを終了します。

```
(parted) quit
```

7. parted を閉じた後、以下のコマンドを実行して、カーネルが新しいパーティションを認識していることを確認します。

```
# cat /proc/partitions
```

parted が作成できるパーティションの最大数は 128 個です。GPT (GUID Partition Table) の仕様により、パーティションテーブル用に確保するエリアを拡大することでさらに多くのパーティションを作成することができますが、parted で用いられる一般的な方法で得られるエリアは、128 個のパーティションに制限されます。

13.2.1. パーティションのフォーマットとラベル付け

パーティションをフォーマットしてラベルを付けるには、以下の手順を使用します。

手順13.2 パーティションのフォーマットとラベル付け

1. パーティションにファイルシステムがない。**ext4** ファイルシステムを作成するには、次のコマンドを実行します。

```
# mkfs.ext4 /dev/sda6
```

WARNING

パーティションをフォーマットすると、そのパーティションに現存するすべてのデータが永久に抹消されます。

2. パーティションにファイルシステムのラベルを付けます。たとえば、新しいパーティションのファイルシステムが **/dev/sda6** で、**Work** のラベルを付ける場合は、以下を使用します。

```
# e2label /dev/sda6 "Work"
```

デフォルトでは、インストールプログラムはパーティションのマウントポイントをラベルとして使用して、ラベルが固有なものとなるようにします。ユーザーは使用するラベルを選択できます。

3. root でマウントポイント（例：**/work**）を作成します。

13.2.2. /etc/fstabへのパーティションの追加

1. root として、**/etc/fstab** ファイルを編集し、パーティションの UUID を使用して新しいパーティションを追加します。

   パーティションの UUID の完全なリストには **blkid -o list** コマンドを使用し、個々のデバイスの詳細には **blkid device** コマンドを使用します。

   /etc/fstab の場合：

   - 最初の列には、**UUID=** の後にファイルシステムの UUID が含まれている必要があります。
   - 2 番目の列には、新しいパーティションのマウントポイントが含まれている必要があります。
   - 3 番目の列は、ファイルシステムのタイプ(**ext4** または **swap** など) である必要があります。
   - 4 番目の列は、ファイルシステムのマウントオプションのリストになります。ここでの **defaults** という用語は、システムの起動時に、パーティションがデフォルトのオプションでマウントされることを意味します。
   - 5 番目と 6 番目のフィールドは、バックアップとチェックのオプションを指定します。root 以外のパーティションの値の例は **0 2** です。

2. システムが新しい設定を登録するように、マウントユニットを再生成します。

```
# systemctl daemon-reload
```

3. ファイルシステムをマウントして、設定が機能することを確認します。

```
# mount /work
```

追加情報

- **/etc/fstab** の形式に関する詳細は、fstab(5) の man ページを参照してください。

13.3. パーティションの削除

警告

パーティションが設定されているデバイスが使用中の場合は、削除しないでください。

手順13.3 パーティションの削除

1. パーティションを削除する前に、以下のいずれかを行います。
   - レスキューモードで起動します。
   - デバイスのパーティションをアンマウントし、デバイスのスワップスペースをオフにします。
2. **parted** ユーティリティーを起動します。

```
# parted device
```

   device を、パーティションを削除するデバイス（例： **/dev/sda** ）に置き換えます。

3. 現在のパーティションテーブルを表示して、削除するパーティションのマイナー番号を確認します。

```
(parted) print
```

4. **rm** コマンドでパーティションを削除します。例えば、マイナー番号 3 のパーティションを削除するのは以下のコマンドです。

```
(parted) rm 3
```

   変更は **Enter** を押すとすぐに行われるため、コマンドを確認してからコミットします。

5. パーティションを削除したら、**print** コマンドを使用して、パーティションテーブルから削除されていることを確認します。

```
(parted) print
```

6. **parted** シェルを終了します。

```
(parted) quit
```

7. **/proc/partitions** ファイルの内容を調べて、パーティションが削除されたことをカーネルが認識していることを確認します。

```
# cat /proc/partitions
```

8. **/etc/fstab** ファイルからパーティションを削除します。削除したパーティションを宣言している行を見つけ、ファイルから削除します。

9. システムが新しい **/etc/fstab** 設定を登録するように、マウントユニットを再生成します。

```
# systemctl daemon-reload
```

13.4. パーティションタイプの設定

パーティションタイプはファイルシステムのタイプと混同しないように、実行中のシステムではほとんど使用されません。ただし、パーティションタイプは、たとえば、デバイスを自動的に識別してマウントするためにパーティションタイプを使用する **systemd-gpt-auto-generator** などのオンザフライジェネレーターにとって重要です。

fdisk ユーティリティーを起動し、**t** コマンドを使用してパーティションタイプを設定できます。以下の例は、最初のパーティションのパーティションタイプを 0x83 (Linux のデフォルト) に変更する方法を示しています。

```
# fdisk /dev/sdc
Command (m for help): t
Selected partition 1
Partition type (type L to list all types): 83
Changed type of partition 'Linux LVM' to 'Linux'.
```

parted ユーティリティーは、パーティションタイプをflags にマッピングすることでパーティションタイプを制御します。これは、エンドユーザーにとっては役に立ちません。**parted** ユーティリティーは、LVM や RAID などの特定のパーティションタイプのみを処理できます。たとえば、**parted** を使用した最初のパーティションから lvm フラグを削除するには、次のコマンドを実行します。

```
# parted /dev/sdc 'set 1 lvm off'
```

一般的に使用されるパーティションタイプとそれらを表すために使用される 16 進数のリストについては、『Red Hat Enterprise Linux 7 Installation Guide』 の付録 Partitions: Turning One Drive Into Many のパーティションタイプの表を参照してください。

13.5. FDISK でのパーティションのサイズ変更

fdisk ユーティリティーを使用すると、GPT、MBR、Sun、SGI、および BSD パーティションテーブルを作成および操作できます。GUID パーティションテーブル(GPT)のディスクでは、**fdisk** GPT サポートは実験的なフェーズであるため、**parted** ユーティリティーを使用することが推奨されます。

fdisk を使用してパーティションサイズを変更する唯一の方法は、パーティションを削除して再作成するため、パーティションのサイズを変更する前に、ファイルシステムに保存されているデータを **バックアップ** して手順をテストします。

重要

サイズを変更するパーティションは、特定ディスクの最後のパーティションにする必要があります。

Red Hat は、LVM パーティションの拡張とサイズ変更にのみ対応しています。

手順13.4 パーティションのサイズ変更

以下の手順は、参照用にのみ提供されています。 ***fdisk*** *を使用してパーティションのサイズを変更するには、以下を実行します。*

1. *デバイスをアンマウントします。*

```
# umount /dev/vda
```

2. ***fdisk disk_name*** *を実行します。以下に例を示します。*

```
# fdisk /dev/vda
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them. Be careful before using the write command.

Command (m for help):
```

3. ***p*** *オプションを使用して、削除するパーティションの行番号を決定します。*

```
Command (m for help): p
Disk /dev/vda: 16.1 GB, 16106127360 bytes, 31457280 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x0006d09a

Device    Boot      Start         End      Blocks   Id  System
/dev/vda1   *       2048     1026047      512000   83  Linux
/dev/vda2         1026048    31457279    15215616   8e  Linux LVM
```

4. ***d*** *オプションを使用してパーティションを削除します。複数のパーティションが利用可能な場合は、* ***fdisk*** *により、削除するパーティションの数を指定するようにプロンプトが表示されます。*

```
Command (m for help): d
Partition number (1,2, default 2): 2
Partition 2 is deleted
```

5. ***n*** *オプションを使用してパーティションを作成し、プロンプトに従います。今後サイズを変更する場合は、十分な領域を確保してください。* ***fdisk*** *のデフォルト動作(* ***Enter*** *を押す)は、デバイス上のすべての領域を使用することです。パーティションの終わりをセクターで指定するか、* ***+ <size> <suffix> を使用して人間が判読できるサイズ*** *を指定できます（例：+500M、または +10G）。*

fdisk *はパーティションの終わりを物理セクターに合わせるため、すべての空き領域を使用しない場合は、人間が判読可能なサイズの指定を使用することを Red Hat は推奨しています。正確な数（セクター内）を指定してサイズを指定すると、* ***fdisk*** *はパーティションの終わりをあわせません。*

```
Command (m for help): n
Partition type:
   p   primary (1 primary, 0 extended, 3 free)
   e   extended
Select (default p): *Enter*
Using default response p
Partition number (2-4, default 2): *Enter*
First sector (1026048-31457279, default 1026048): *Enter*
Using default value 1026048
Last sector, +sectors or +size{K,M,G} (1026048-31457279, default 31457279): +500M
Partition 2 of type Linux and of size 500 MiB is set
```

6. *パーティションのタイプを LVM に設定します。*

```
Command (m for help): t
Partition number (1,2, default 2): *Enter*
Hex code (type L to list all codes): 8e
Changed type of partition 'Linux' to 'Linux LVM'
```

7. *エラーが選択したパーティションで不安定になる可能性があるため、変更が正しい場合は、* ***w*** *オプションで変更を書き込みます。*

8. *デバイスで* ***e2fsck*** *を実行して、整合性をチェックします。*

```
# e2fsck /dev/vda
e2fsck 1.41.12 (17-May-2010)
Pass 1:Checking inodes, blocks, and sizes
Pass 2:Checking directory structure
Pass 3:Checking directory connectivity
Pass 4:Checking reference counts
Pass 5:Checking group summary information
ext4-1:11/131072 files (0.0% non-contiguous),27050/524128 blocks
```

9. *デバイスをマウントします。*

```
# mount /dev/vda
```

詳細は、fdisk(8) の man ページを参照してください。

第14章 SNAPPER でのスナップショットの作成と管理

スナップショットボリュームは、ターゲットボリュームのポイントインタイムコピーで、ファイルシステムを以前の状態に戻すことができます。Snapper は、Btrfs ファイルシステムおよびシンプロビジョニングの LVM ファイルシステムのスナップショットを作成して維持するコマンドラインツールです。

14.1. SNAPPER の初期設定の作成

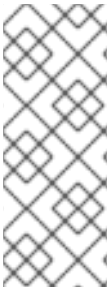
Snapper は、操作するボリュームごとに個別の設定ファイルを必要とします。設定ファイルは手動で設定する必要があります。デフォルトでは、root ユーザーのみが snapper コマンドを実行できます。

警告

シンプロビジョニングを使用している場合は、シンプールの空き領域を監視します。シンプールが完全に消費されると、データが失われる可能性があります。詳細は、『Red Hat Enterprise Linux 7 Logical Volume Manager Administration Guide』の Thinly-Provisioned Logical Volumes (Thin Volumes) を参照してください。

Btrfs ツールおよびファイルシステムはテクノロジープレビューとして提供されるため、実稼働システムには適さないことに注意してください。

root 以外のユーザーまたはグループに特定の Snapper コマンドの使用を許可することは可能ですが、Red Hat では、権限のないユーザーまたはグループに対しては、権限を **昇格させない** ことを推奨しています。このような設定は SELinux をバイパスし、セキュリティーリスクをもたらす可能性があります。Red Hat では、これらの機能をセキュリティーチームで確認し、代わりに **sudo** インフラストラクチャーの使用を検討してください。



注記

Btrfs は、Red Hat Enterprise Linux 7 ではテクノロジープレビューとして利用できますが、Red Hat Enterprise Linux 7.4 リリース以降では非推奨になりました。Red Hat Enterprise Linux の将来のメジャーリリースで削除される予定です。

詳細は、Red Hat Enterprise Linux 7.4 リリースノートの 非推奨の機能 を参照してください。

手順14.1 Snapper 設定ファイルの作成

1. 以下のいずれかを作成または選択します。
   - シンプロビジョニングされた論理ボリューム (Red Hat がサポートするファイルシステムがその上にある)、または
   - Btrfs サブボリューム。
2. ファイルシステムをマウントします。
3. このボリュームを定義する設定ファイルを作成します。

   LVM2 の場合:

```
# snapper -c config_name create-config -f "lvm(fs_type)" /mount-point
```

たとえば、/lvm_mount にマウントした ext4 ファイルシステムを持つ LVM2 サブボリュームに、lvm_config という設定ファイルを作成する場合は、次のコマンドを実行します。

```
# snapper -c lvm_config create-config -f "lvm(ext4)" /lvm_mount
```

Btrfs の場合:

```
# snapper -c config_name create-config -f btrfs /mount-point
```

- **-c config_name** オプションは、設定ファイルの名前を指定します。
- **create-config** は snapper に設定ファイルを作成するように指示します。
- **-f file_system** は snapper に、使用するファイルシステムを指示します。これを省略すると、snapper がファイルシステムの検出を試みます。
- **/mount-point** は、サブボリュームまたはシンプロビジョニングされた LVM2 ファイルシステムがマウントされる場所です。

または、**/btrfs_mount** にマウントされている Btrfs サブボリュームに **btrfs_config** という設定ファイルを作成するには、次のコマンドを実行します。

```
# snapper -c btrfs_config create-config -f btrfs /btrfs_mount
```

設定ファイルは **/etc/snapper/configs/** ディレクトリーに保存されます。

14.2. SNAPPER スナップショットの作成

Snapper では、以下のタイプのスナップショットを作成できます。

プレスナップショット

プレスナップショットは、ポストスナップショットの作成元として機能します。この 2 つは密接に関連しており、2 つのポイント間でファイルシステムの修正を追跡するように設計されています。ポストスナップショットを作成する前に、プレスナップショットを作成する必要があります。

ポストスナップショット

ポストスナップショットは、プレスナップショットのエンドポイントとして機能します。結合されたプレスナップショットとポストスナップショットは、比較の範囲を定義します。デフォルトでは、すべての新しい snapper ボリュームは、関連するポストスナップショットが正常に作成された後にバックグラウンド比較を作成するように設定されています。

シングルスナップショット

1 つのスナップショットとは、特定の時点で作成されたスタンドアロンのスナップショットのことです。これらを使用すると、修正のタイムラインを追跡し、後で戻るための一般的なポイントを持つために使用することができます

14.2.1. プレスナップショットとポストスナップショットのペアの作成

14.2.1.1. Snapper を使用したプレスナップショットの作成

プレスナップショットの作成には、以下のコマンドを実行します。

```
# snapper -c config_name create -t pre
```

-c config_name オプションは、名前付き設定ファイルの仕様に従ってスナップショットを作成します。設定ファイルが存在しない場合は、「Snapper の初期設定の作成」 を参照してください。

create -t オプションは、作成するスナップショットのタイプを指定します。許可されるエントリーは、**pre**、**post** または **single** です。

たとえば、「Snapper の初期設定の作成」 で作成した **lvm_config** 設定ファイルを使用してプレスナップショットを作成するには、以下のコマンドを実行します。

```
# snapper -c SnapperExample create -t pre -p
1
```

-p オプションは、作成されたスナップショットの番号を出力しますが、使用するかどうかは任意です。

14.2.1.2. Snapper を使用しポストスナップショットの作成

ポストスナップショットはスナップショットのエンドポイントであり、「Snapper を使用したプレスナップショットの作成」 の手順に従って、親のプレスナップショットの後に作成する必要があります。

手順14.2 ポストスナップショットの作成

1. プレスナップショットの数を指定します。

   ```
   # snapper -c config_name list
   ```

   たとえば、設定ファイル **lvm_config** を使用して作成されたスナップショットの一覧を表示するには、次のコマンドを実行します。

   ```
   # snapper -c lvm_config list
   Type   | # | Pre # | Date                | User | Cleanup  | Description | Userdata
   -------+---+-------+---------------------+------+----------+-------------+---------
   single | 0 |       |                     | root |          | current     |
   pre    | 1 |       | Mon 06<...>         | root |          |             |
   ```

   この出力は、プレスナップショットが番号1 であることを示しています。

2. 事前に作成したプレスナップショットにリンクされる、ポストスナップショットを作成します。

   ```
   # snapper -c config_file create -t post --pre-num pre_snapshot_number
   ```

   - **-t post** オプションは、ポストスナップショットの作成を指定します。
   - **--pre-num** オプションは、対応するプレスナップショットを指定します。

   たとえば、**lvm_config** 設定ファイルを使用してポストスナップショットを作成し、プレスナップショット番号1 にリンクするには、次のコマンドを実行します。

```
# snapper -c lvm_config create -t post --pre-num 1 -p
2
```

-p オプションは、作成されたスナップショットの番号を出力しますが、使用するかどうかは任意です。

3. プレスナップショット 1 とポストスナップショット 2 が作成され、ペアになりました。**list** コマンドを使用してこれを確認します。

```
# snapper -c lvm_config list
Type   | # | Pre # | Date                | User | Cleanup  | Description | Userdata
-------+---+-------+---------------------+------+----------+-------------+---------
single | 0 |       |                     | root |          | current     |
pre    | 1 |       | Mon 06<...>         | root |          |             |
post   | 2 | 1     | Mon 06<...>         | root |          |             |
```

14.2.1.3. スナップショット前およびスナップショット後のコマンドのラップ

また、コマンドをプレスナップショットおよびポストスナップショットにラップすることもできます。これはテスト時に役立ちます。手順14.3「スナップショット前およびスナップショット後のコマンドのラップ」(以下の手順のショートカット) を参照してください。

1. **snapper create pre snapshot** コマンドを実行している。
2. コマンドの実行、またはファイルシステムのコンテンツに影響を与える可能性があるアクションを実行するコマンドのリスト。
3. **snapper create post snapshot** コマンドを実行している。

手順14.3 スナップショット前およびスナップショット後のコマンドのラップ

1. プレスナップショットとポストスナップショットでコマンドをラップするには、以下のコマンドを実行します。

```
# snapper -c lvm_config create --command "command_to_be_tracked"
```

たとえば、**/lvm_mount/hello_file** ファイルの作成を追跡するには、次のコマンドを実行します。

```
# snapper -c lvm_config create --command "echo Hello > /lvm_mount/hello_file"
```

2. これを確認するには、**status** コマンドを使用します。

```
# snapper -c config_file status first_snapshot_number..second_snapshot_number
```

たとえば、最初の手順で行った変更を追跡するには、以下のコマンドを実行します。

```
# snapper -c lvm_config status 3..4
+..... /lvm_mount/hello_file
```

必要に応じて、**list** コマンドを使用してスナップショットの番号を確認します。

status コマンドの詳細は、「スナップショット間での変更の追跡」 を参照してください。

上記の例で使用されているコマンドが、スナップショットがキャプチャーする唯一のものであるという保証はないことに注意してください。Snapper は、ユーザーが変更したものだけでなく、システムが変更したものも記録します。

14.2.2.1 つのスナップショットの作成

*単一の snapper スナップショットの作成は、プレスナップショットまたはポストスナップショットの作成に似ていますが、create **-t** オプションのみが単一を指定します。1 つのスナップショットを使用すれば、他のスナップショットとは無関係に、1 つのスナップショットを作成できます。ただし、比較を自動的に生成したり、追加情報をリスト表示したりせずに、LVM2 シンボリュームのスナップショットを簡単に作成したい場合は、「スナップショット」 で説明しているように、Red Hat では、この目的で Snapper の代わりに System Storage Manager を使用することを推奨しています。*

1 つのスナップショットを作成するには、以下を実行します。

```
# snapper -c config_name create -t single
```

*たとえば、次のコマンドは、**lvm_config** 設定ファイルを使用して単一のスナップショットを作成します。*

```
# snapper -c lvm_config create -t single
```

*1 つのスナップショットは変更を追跡するように特別に設計されていませんが、**snapper diff** コマンド、**xadiff** コマンド、および **status** コマンドを使用して、2 つのスナップショットを比較することができます。これらのコマンドの詳細は、「スナップショット間での変更の追跡」 を参照してください。*

14.2.3. 自動スナップショットを作成するための Snapper の設定

自動スナップショットの作成は、Snapper の重要な機能の 1 つです。デフォルトでは、ボリュームに Snapper を設定すると、Snapper は 1 時間ごとのボリュームのスナップショットの作成を開始します。

デフォルト設定では、Snapper は以下を保持します。

- *10 時間ごとのスナップショットと、最後の 1 時間ごとのスナップショットは、1 日単位のスナップショットとして保存されます。*
- *1 日単位 10 のスナップショットと、1 カ月の最後の 1 日単位のスナップショットは、月単位のスナップショットとして保存されます。*
- *月単位 10 のスナップショット、および最後の月単位のスナップショットは年単位のスナップショットとして保存されます。*
- *年単位 10 のスナップショット*

デフォルトで、Snapper が保持するスナップショットは、合計 50 個以下であることに注意してください。ただし、Snapper は、デフォルトで 1,800 秒未満前に作成されたすべてのスナップショットを保持します。

*デフォルト設定は **/etc/snapper/config-templates/default** ファイルで指定されます。**snapper create-config** コマンドを使用して設定を作成すると、指定されていない値はデフォルト設定に基づいて設定されます。**/etc/snapper/configs/config_name** ファイルで定義されているボリュームの設定を編集できます。*

14.3. スナップショット間での変更の追跡

status、**diff**、および **xadiff** コマンドを使用して、スナップショット間でサブボリュームに加えられた変更を追跡します。

status

status コマンドは、2 つのスナップショット間で作成、変更、または削除されたファイルおよびディレクトリーの一覧を表示します。これは、2 つのスナップショット間の変更の包括的なリストです。このコマンドを使用すると、詳細を必要以上に表示することなく、変更の概要を取得できます。

詳細は、「**status** コマンドでの変更の比較」 を参照してください。

diff

diff コマンドは、少なくとも 1 つの変更が検出された場合に、**status** コマンドから受信した 2 つのスナップショット間で変更されたファイルおよびディレクトリーの差分を表示します。

詳細は、「**diff** コマンドでの変更の比較」 を参照してください。

xadiff

xadiff コマンドは、ファイルまたはディレクトリーの拡張属性が 2 つのスナップショット間でどのように変更されたかを比較します。

詳細は、「**xadiff** コマンドでの変更の比較」 を参照してください。

14.3.1. status コマンドでの変更の比較

status コマンドは、2 つのスナップショット間で作成、変更、または削除されたファイルおよびディレクトリーの一覧を表示します。

2 つのスナップショット間のファイルのステータスを表示するには、以下を使用します。

```
# snapper -c config_file status first_snapshot_number..second_snapshot_number
```

必要に応じて、**list** コマンドを使用してスナップショット番号を確認します。

たとえば、次のコマンドは、設定ファイル **lvm_config** を使用して、スナップショット 1 から 2 の間に加えられた変更を表示します。

```
#snapper -c lvm_config status 1..2
tp.... /lvm_mount/dir1
-..... /lvm_mount/dir1/file_a
c.ug.. /lvm_mount/file2
+..... /lvm_mount/file3
....x. /lvm_mount/file4
cp..xa /lvm_mount/file5
```

出力の最初の部分にある文字とドットを列として読み取ります。

```
+..... /lvm_mount/file3
||||||
123456
```

列 1 は、ファイル (ディレクトリーエントリー) タイプの変更を示しています。以下の値が使用できます。

コラム 1

出力	意味
.	何も変更されていません。
+	ファイルが作成されました。
-	ファイルが削除されました。
c	コンテンツが変更されました。
t	ディレクトリーエントリーのタイプが変更されました。(例: 以前のシンボリックリンクが同じファイル名の通常のファイルに変更されるなど。)

列 2 は、ファイルの権限に変更があったことを示しています。以下の値が使用できます。

コラム 2

出力	意味
.	パーミッションは変更されませんでした。
p	パーミッションが変更されました。

列 3 は、ユーザーの所有権が変更したことを示しています。以下の値が使用できます。

コラム 3

出力	意味
.	ユーザーの所有権は変更されていません。
u	ユーザーの所有権が変更されました。

列 4 は、グループの所有権が変更したことを示しています。以下の値が使用できます。

列 4

出力	意味
.	グループの所有権は変更されていません。

出力	意味
g	グループの所有権が変更されました。

列 5 は、拡張属性の変更を示しています。以下の値が使用できます。

列 5

出力	意味
.	拡張属性は変更されていません。
x	拡張属性が変更されました。

列 6 は、アクセス制御リスト (ACL) の変更を示しています。以下の値が使用できます。

列 6

出力	意味
.	ACL は変更されていません。
a	ACL が変更されました。

14.3.2. diff コマンドでの変更の比較

***diff** コマンドは、2 つのスナップショット間で変更されたファイルとディレクトリーの変更を表示します。*

```
# snapper -c config_name diff first_snapshot_number..second_snapshot_number
```

*必要に応じて、**list** コマンドを使用してスナップショットの番号を確認します。*

*たとえば、**lvm_config** 設定ファイルを使用して作成されたスナップショット 1 とスナップショット 2 の間にファイルに加えられた変更を比較するには、次のコマンドを実行します。*

```
# snapper -c lvm_config diff 1..2
--- /lvm_mount/.snapshots/13/snapshot/file4 19<...>
+++ /lvm_mount/.snapshots/14/snapshot/file4 20<...>
@@ -0,0 +1 @@
+words
```

*この出力は、**file4** が「単語」を追加するように変更されたことを示しています。*

14.3.3. xadiff コマンドでの変更の比較

xadiff コマンドは、ファイルまたはディレクトリーの拡張属性が2 つのスナップショット間でどのように変更されたかを比較します。

```
# snapper -c config_name xadiff first_snapshot_number..second_snapshot_number
```

必要に応じて、**list** コマンドを使用してスナップショットの番号を確認します。

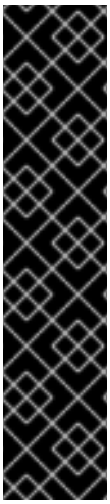
たとえば、**lvm_config** 設定ファイルを使用して作成されたスナップショット番号1 とスナップショット番号2 の間のxadiff 出力を表示するには、次のコマンドを実行します。

```
# snapper -c lvm_config xadiff 1..2
```

14.4. スナップショット間の変更を元に戻す

2 つの既存のSnapper スナップショット間で行われた変更を元に戻すには、次の形式で **undochange** コマンドを実行します。**1** は最初のスナップショットで、**2** は2 番目のスナップショットです。

```
snapper -c config_name undochange 1..2
```



重要

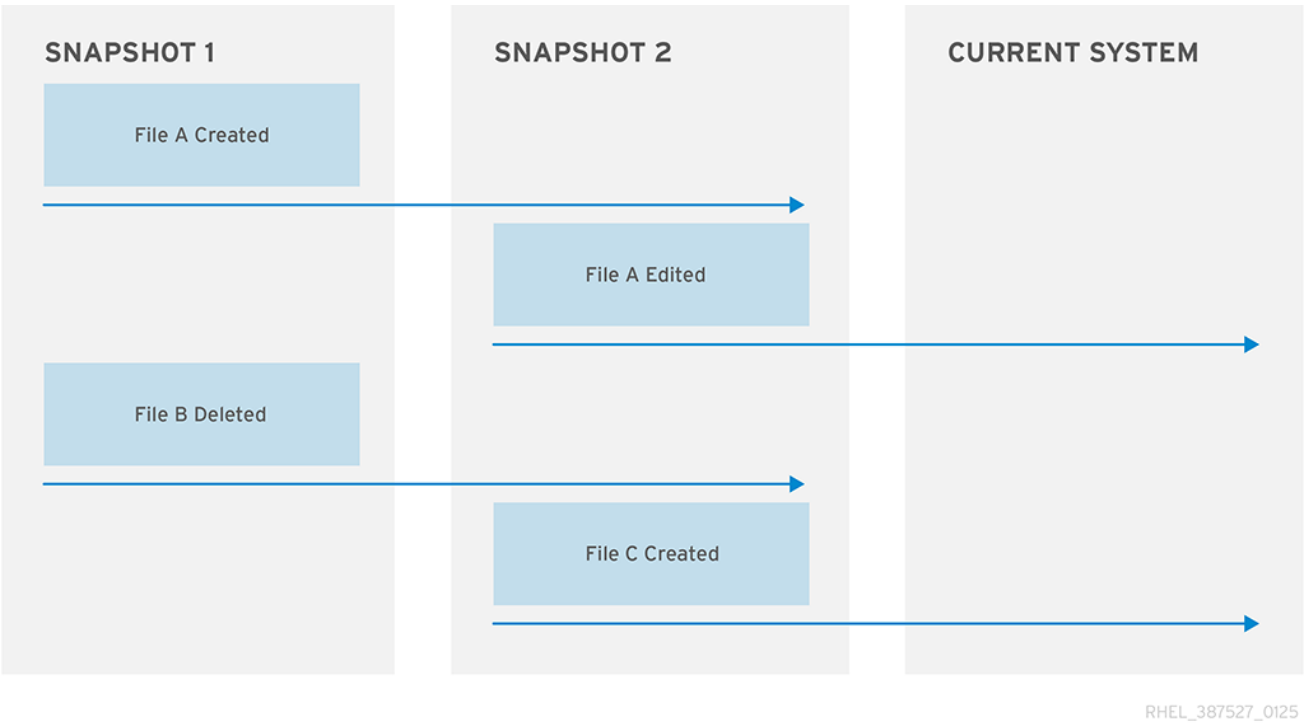
undochange コマンドを使用しても、Snapper ボリュームは元の状態に戻らず、データの一貫性は確保されません。たとえば、スナップショット2 の後で、指定された範囲外で発生したファイルの変更は、たとえばスナップショット1 の状態に戻った後も変更されないままとなります。たとえば、**undochange** を実行してユーザーの作成を元に戻すと、そのユーザーが所有するファイルはそのまま残ることができます。

また、スナップショットとしてファイルの一貫性を確保するメカニズムもないため、**undochange** コマンドの使用時に、すでに存在する不整合をスナップショットに戻すことができます。

root ファイルシステムでSnapper **undochange** コマンドを使用しないでください。失敗する可能性があるためです。

以下の図は、**undochange** コマンドがどのように機能するかを示しています。

図14.1 時間の経過に伴う Snapper のステータス



[D]

*図は、**snapshot_1** が作成された時点、**file_a** が作成され、**file_b** が削除されることを示しています。次に **Snapshot_2** が作成され、その後に **file_a** が編集され、**file_c** が作成されます。これが、システムの現在の状態になります。現在のシステムには、編集バージョンの **file_a** があり、**file_b** はなく、新しく作成された **file_c** があります。*

***undochange** コマンドが呼び出されると、Snapper は、最初にリストされたスナップショットと 2 番目のスナップショットの間に変更されたファイルのリストを生成します。図では、**snapper -c SnapperExample undochange 1..2** コマンドを使用すると、Snapper は変更されたファイルのリスト（つまり、**file_a** が作成され、**file_b** が削除されます）を作成し、それらを現在のシステムに適用します。そのため、以下に留意してください。*

- *現在のシステムには **file_a** がありません。これは、**snapshot_1** の作成時にまだ作成されていないためです。*
- ***file_b** は存在します。**snapshot_1** から現在のシステムにコピーされます。*
- ***file_c** は、作成が指定の時間外であったため、存在します。*

***file_b** と **file_c** が競合すると、システムが破損する可能性があることに注意してください。*

***snapper -c SnapperExample undochange 2..1** コマンドを使用することもできます。この場合、現在のシステムは、編集したバージョンの **file_a** を **snapshot_1** からコピーされたものに置き換えます。これにより、**snapshot_2** の作成後に行われたファイルの編集が取り消されます。*

mount および unmount コマンドを使用して変更を元に戻す

***undochange** コマンドは、変更を元に戻すのに最適な方法であるとは限りません。**status** および **diff** コマンドを使用すると、修飾された決定を作成し、Snapper の代わりに **mount** コマンドおよび **unmount** コマンドを使用できます。**mount** コマンドおよび **unmount** コマンドは、スナップショットをマウントし、Snapper ワークフローとは別にコンテンツを参照する場合にのみ役立ちます。*

*必要に応じて、**mount** コマンドは、マウントする前にそれぞれの LVM Snapper スナップショットをアクティブにします。たとえば、スナップショットをマウントしたり、古いバージョンの複数のファイル*

を手動で抽出したりする場合は、**mount** コマンドおよび **unmount** コマンドを使用します。ファイルを手動で元に戻すには、マウントされたスナップショットから現在のファイルシステムにファイルをコピーします。現在のファイルシステムであるスナップショット 0 は、手順14.1「Snapper 設定ファイルの作成」 で作成されたライブファイルシステムです。ファイルを、元の /mount-point のサブツリーにコピーします。

明示的なクライアント側の要求には、**mount** コマンドおよび **unmount** コマンドを使用します。**/etc/snapper/configs/config_name** ファイルには、ユーザーおよびグループを追加できる ALLOW_USERS= 変数および ALLOW_GROUPS= 変数が含まれています。次に、**snapperd** を使用すると、追加したユーザーおよびグループのマウント操作を実行できます。

14.5. SNAPPER スナップショットの削除

スナップショットを削除する場合は、以下を行います。

```
# snapper -c config_name delete snapshot_number
```

list コマンドを使用して、スナップショットが正常に削除されたことを確認できます。

第15章 SWAP 領域

Linux のスワップ領域は、物理メモリー (RAM) が不足すると使用されます。システムに多くのメモリーリソースが必要で、RAM が不足すると、メモリーの非アクティブなページがスワップ領域に移動します。スワップ領域は、RAM が少ないマシンで役に立ちますが、RAM の代わりに使用しないようにしてください。スワップ領域はハードドライブにあり、そのアクセス速度は物理メモリーに比べると遅くなります。スワップ領域の設定は、専用のスワップパーティション (推奨)、スワップファイル、またはスワップパーティションとスワップファイルの組み合せが考えられます。**Btrfs はスワップ領域を サポートしない** ことに注意してください。

過去数年、推奨されるスワップ領域のサイズは、システムの RAM サイズに比例して増加していました。しかし、最近のシステムには通常、数百ギガバイトの RAM が含まれます。結果として、推奨されるスワップ領域は、システムのメモリーではなく、システムメモリーのワークロードの機能とみなされます。

表15.1「システムの推奨 swap 領域」 では、システムの RAM の容量別に推奨されるスワップパーティションのサイズと、システムがハイバネートするために十分なメモリーが必要かどうかを示しています。推奨されるスワップパーティションのサイズは、インストール時に自動的に確定されます。ハイバネートを可能にするには、カスタムのパーティション分割段階でスワップ領域を編集する必要があります。

表15.1「システムの推奨 swap 領域」 の推奨では、メモリーが少ないシステム (1 GB 以下) で特に重要になります。このようなシステムで十分なスワップ領域を割り当てられないと、不安定になる問題が生じたり、インストールしたシステムが起動できなくなる可能性があります。

表15.1 システムの推奨 swap 領域

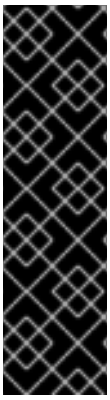
システム内の RAM の容量	推奨されるスワップ領域	ハイバネートを許可する場合に推奨されるスワップ領域
⩽ 2 GB	RAM 容量の 2 倍	RAM 容量の 3 倍
> 2 GB ~ 8 GB	RAM 容量と同じ	RAM 容量の 2 倍
> 8 GB ~ 64 GB	最低 4GB	RAM 容量の 1.5 倍
> 64 GB	最低 4GB	ハイバネートは推奨されない

注記

64 GB を超える RAM を搭載したシステムでハイバネーションが推奨されない理由は 2 つあります。第 1 に、ハイバネーションには、膨張した (そしておそらくあまり使用されない) スワップ領域用に余分なスペースが必要です。第 2 に、常駐データを RAM からディスクに移動したり、ディスクに戻したりするプロセスは、完了するまでに長い時間がかかる場合があります。

ご使用のシステムが 表15.1「システムの推奨 swap 領域」 に記載されている境界線上 (RAM が 2GB、8GB または 64GB) になる場合、swap サイズやハイバネートへの対応を決める際は適宜判断してください。システムリソースに余裕がある場合は、スワップ領域を増やすとパフォーマンスが向上することがあります。

高速のドライブ、コントローラー、およびインターフェイスを搭載したシステムでは、複数のストレージデバイスにスワップ領域を分散すると、スワップ領域のパフォーマンスも向上します。



重要

スワップ領域として割り当てたファイルシステムおよびLVM2 ボリュームは、変更時に **使用しない** *でください。システムプロセスまたはカーネルがスワップ領域を使用していると、スワップの修正に失敗します。* ***free*** *コマンドおよび* ***cat /proc/swaps*** *コマンドを使用して、スワップの量と使用中の場所を確認します。*

システムが ***レスキュー*** *モードで起動している間にスワップ領域を変更する必要があります。『*Red Hat Enterprise Linux 7 Installation Guide *』 の Booting Your Computer in Rescue Mode を参照してください。ファイルシステムをマウントするように求められたら、* ***Skip*** *を選択します。*

15.1. スワップ領域の追加

場合によっては、インストールした後にさらにswap 領域の追加が必要になることもあります。たとえば、システムのRAM 容量を1GB から2 GB にアップグレードするときに、スワップスペースが2 GB しかないとします。メモリーを大幅に消費する操作を実行している場合や、大量のメモリーを必要とするアプリケーションを実行する場合は、スワップ領域を4 GB に増やすことが有益となる可能性があります。

選択肢が3 つあります: 新規のswap パーティションの作成、新規のswap ファイルの作成、あるいは既存のLVM2 論理ボリューム上でswap の拡張。この中では、既存論理ボリュームを拡張することが推奨されます。

15.1.1. LVM2 論理ボリュームでのスワップ領域の拡張

Red Hat Enterprise Linux 7 は、デフォルトで、使用可能なすべての領域をインストール時に使用します。この設定を選択している場合は、まずswap 領域として使用するボリュームグループに新しい物理ボリュームを追加しなければなりません。

swap 領域のボリュームグループにストレージを追加した後に、それを拡張することができるようになります。これを行うには、以下の手順を実行します(***/dev/VolGroup00/LogVol01*** *が2 GB 拡張するボリュームであると想定)。*

手順15.1 LVM2 論理ボリュームでのスワップ領域の拡張

1. *関連付けられている論理ボリュームのスワップ機能を無効にします。*

```
# swapoff -v /dev/VolGroup00/LogVol01
```

2. *LVM2 論理ボリュームのサイズを2 GB 増やします。*

```
# lvresize /dev/VolGroup00/LogVol01 -L +2G
```

3. *新しいスワップ領域をフォーマットします。*

```
# mkswap /dev/VolGroup00/LogVol01
```

4. *拡張論理ボリュームを有効にします。*

```
# swapon -v /dev/VolGroup00/LogVol01
```

5. スワップの論理ボリュームの拡張に成功したかどうかをテストするには、アクティブなスワップ容量を調べます。

```
$ cat /proc/swaps
$ free -h
```

15.1.2. スワップの LVM2 論理ボリュームの作成

サイズが 2 GB のスワップボリュームグループを追加するには、**/dev/VolGroup00/LogVol02** が追加するスワップボリュームであると想定します。

1. サイズが 2 GB の LVM2 論理ボリュームを作成します。

```
# lvcreate VolGroup00 -n LogVol02 -L 2G
```

2. 新しいスワップ領域をフォーマットします。

```
# mkswap /dev/VolGroup00/LogVol02
```

3. 以下のエントリーを **/etc/fstab** ファイルに追加します。

```
/dev/VolGroup00/LogVol02   swap     swap    defaults     0 0
```

4. システムが新しい設定を登録するように、マウントユニットを再生成します。

```
# systemctl daemon-reload
```

5. 論理ボリュームでスワップをアクティブにします。

```
# swapon -v /dev/VolGroup00/LogVol02
```

6. スワップ論理ボリュームの作成に成功したかどうかをテストするには、アクティブなスワップ容量を調べます。

```
$ cat /proc/swaps
$ free -h
```

15.1.3. スワップファイルの作成

swap ファイルを追加します。

手順15.2 スワップファイルの追加

1. 新しいスワップファイルのサイズをメガバイト単位で指定してから、そのサイズに 1024 をかけてブロック数を指定します。たとえばスワップファイルのサイズが 64 MB の場合は、ブロック数が 65536 になります。

2. 空のファイルの作成:

```
# dd if=/dev/zero of=/swapfile bs=1024 count=65536
```

希望のブロックサイズと同等の値に count を置き換えます。

3. 次のコマンドでスワップファイルをセットアップします。

```
# mkswap /swapfile
```

4. スワップファイルのセキュリティーを変更して、全ユーザーで読み込みができないようにします。

```
# chmod 0600 /swapfile
```

5. システムの起動時にスワップファイルを有効にするには、root で **/etc/fstab** を編集して、以下のエントリーを追加します。

```
/swapfile          swap            swap    defaults        0 0
```

次にシステムが起動すると新しいスワップファイルが有効になります。

6. システムが新しい **/etc/fstab** 設定を登録するように、マウントユニットを再生成します。

```
# systemctl daemon-reload
```

7. スワップファイルをすぐに有効にするには、次のコマンドを実行します。

```
# swapon /swapfile
```

8. 新しいスワップファイルを作成して有効にできたことをテストするには、アクティブなスワップ容量を調べます。

```
$ cat /proc/swaps
$ free -h
```

15.2. スワップ領域の削除

インストールの後に swap 領域を減らすことが賢明な場合もあります。たとえば、システムの RAM 容量を 1 GB から 512 MB にダウングレードするとします。しかし、依然として 2 GB のスワップ容量が割り当てられています。ディスク領域が大きくなる (2 GB など) と無駄になる可能性があるため、スワップ領域を 1 GB に減らすことでメリットを得られることがあります。

ここでも選択肢が 3 つあります: swap 用に使用していた LVM2 論理ボリューム全体を削除、swap ファイルの削除、あるいは既存の LVM2 論理ボリューム上の swap 領域の低減。

15.2.1. LVM2 論理ボリュームでのスワップ領域の縮小

LVM2 スワップ論理ボリュームを縮小するには(**/dev/VolGroup00/LogVol01** が縮小するボリュームであると想定):

手順15.3 LVM2 の swap 論理ボリュームの削減

1. 関連付けられている論理ボリュームのスワップ機能を無効にします。

```
# swapoff -v /dev/VolGroup00/LogVol01
```

2. LVM2 論理ボリュームのサイズを変更して 512 MB 削減します。

```
# lvreduce /dev/VolGroup00/LogVol01 -L -512M
```

3. 新しいスワップ領域をフォーマットします。

```
# mkswap /dev/VolGroup00/LogVol01
```

4. 論理ボリュームでスワップをアクティブにします。

```
# swapon -v /dev/VolGroup00/LogVol01
```

5. スワップ論理ボリュームの縮小に成功したかどうかをテストするには、アクティブなスワップ容量を調べます。

```
$ cat /proc/swaps
$ free -h
```

15.2.2. スワップの LVM2 論理ボリュームの削除

swap ボリュームグループを削除するには(**/dev/VolGroup00/LogVol02** が削除するスワップボリュームであると想定)、以下を実行します。

手順15.4 swap ボリュームグループの削除

1. 関連付けられている論理ボリュームのスワップ機能を無効にします。

```
# swapoff -v /dev/VolGroup00/LogVol02
```

2. LVM2 論理ボリュームを削除します。

```
# lvremove /dev/VolGroup00/LogVol02
```

3. **/etc/fstab** ファイルから、以下の関連エントリーを削除します。

```
/dev/VolGroup00/LogVol02   swap     swap    defaults     0 0
```

4. システムが新しい設定を登録するように、マウントユニットを再生成します。

```
# systemctl daemon-reload
```

5. 削除されたスワップストレージへの参照をすべて **/etc/default/grub** ファイルから削除します。

```
# vi /etc/default/grub
```

6. grub 設定を再構築します。

    a. BIOS ベースのマシンでは、次を実行します。

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
```

b. UEFI ベースのマシンでは、次を実行します。

```
# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

7. 論理ボリュームの削除に成功したかどうかをテストするには、アクティブなスワップ容量を調べます。

```
$ cat /proc/swaps
$ free -h
```

15.2.3. スワップファイルの削除

swap ファイルを削除します。

手順15.5 swap ファイルの削除

1. シェルプロンプトで次のコマンドを実行してスワップファイルを無効にします(**/swapfile** はスワップファイルです)。

```
# swapoff -v /swapfile
```

2. **/etc/fstab** ファイルからエントリーを削除します。
3. システムが新しい設定を登録するように、マウントユニットを再生成します。

```
# systemctl daemon-reload
```

4. 実際のファイルを削除します。

```
# rm /swapfile
```

15.3. SWAP 領域の移動

スワップ領域をある場所から別の場所に移動するには、以下のコマンドを実行します。

1. スワップ領域の削除 「スワップ領域の削除」。
2. スワップ領域の追加 「スワップ領域の追加」。

第16章 SYSTEM STORAGE MANAGER (SSM)

System Storage Manager (SSM) は、さまざまなテクノロジーでストレージを管理するためのコマンドラインインターフェイスを提供します。ストレージシステムは、デバイスマッパー (DM)、論理ボリュームマネージャー (LVM)、および複数のデバイス (MD) の使用により、ますます複雑になっています。これにより、ユーザーフレンドリーではないシステムが作成され、エラーや問題が発生しやすくなっています。SSM は、統一されたユーザーインターフェイスを作成することにより、これを軽減します。このインターフェイスを使用すると、複雑なシステムを簡単に実行できます。たとえば、SSM を使用せずに新しいファイルシステムを作成してマウントするには、5 つのコマンドを使用する必要があります。SSM では、必要なのは 1 つだけです。

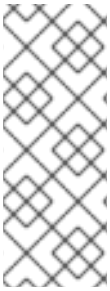
本章では、SSM がさまざまなバックエンドと相互作用する方法および一般的なユースケースの一部を説明します。

16.1. SSM のバックエンド

SSM は、**ssmlib/main.py** のコア抽象化レイヤーを使用します。これは、基礎となるテクノロジーの詳細を無視して、デバイス、プール、およびボリュームの抽象化に準拠します。バックエンドを **ssmlib/main.py** に登録して、**作成**、**スナップショット**、ボリュームやプールの **削除** など、特定のストレージテクノロジーメソッドを処理します。

SSM には複数のバックエンドが登録されています。次のセクションでは、それらの基本情報と、プール、ボリューム、スナップショット、およびデバイスの処理方法の定義について説明します。

16.1.1. Btrfs バックエンド



注記

Btrfs は、Red Hat Enterprise Linux 7 ではテクノロジープレビューとして利用できますが、Red Hat Enterprise Linux 7.4 リリース以降では非推奨になりました。Red Hat Enterprise Linux の将来のメジャーリリースで削除される予定です。

詳細は、Red Hat Enterprise Linux 7.4 リリースノートの 非推奨の機能 を参照してください。

多くの高度な機能を備えたファイルシステムである Btrfs は、SSM のボリューム管理バックエンドとして使用されます。プール、ボリューム、およびスナップショットは、Btrfs バックエンドで作成できます。

16.1.1.1. Btrfs プール

Btrfs ファイルシステム自体がプールです。デバイスを追加して拡張したり、デバイスを削除して縮小したりできます。SSM は、Btrfs プールの作成時に Btrfs ファイルシステムを作成します。つまり、すべての新しい Btrfs プールには、プールと同じ名前のボリュームが 1 つありますが、これはプール全体を削除しないと削除できません。デフォルトの Btrfs プール名は **btrfs_pool** です。

プールの名前がファイルシステムのラベルとして使用されます。ラベルのないシステムに既存の Btrfs ファイルシステムがある場合、Btrfs プールは **btrfs_device_base_nameの形式で内部使用の名前** を生成します。

16.1.1.2. Btrfs ボリューム

プール内の最初のボリュームの後に作成されるボリュームは、サブボリュームと同じです。SSM は、サブボリュームを作成するために、マウントを解除すると、Btrfs ファイルシステムを一時的にマウントします。

*ボリュームの名前は、Btrfs ファイルシステムのサブボリュームパスとして使用されます。たとえば、サブボリュームは **/dev/lvm_pool/lvol001** と表示されます。ボリュームを作成するには、このパスにあるすべてのオブジェクトが存在している必要があります。ボリュームは、マウントポイントで参照することもできます。*

16.1.1.3. Btrfs スナップショット

スナップショットは、SSM を使用するシステムの Btrfs ボリュームから作成できます。Btrfs では、サブボリュームとスナップショットが区別されないことに注意してください。これは、SSM が Btrfs スナップショットの宛先を認識できないことを意味しますが、特別な名前の形式を認識しようとします。スナップショットの作成時に指定した名前が特定のパターンに該当する場合、スナップショットは認識されず、通常の Btrfs ボリュームとしてリスト表示されます。

16.1.1.4. Btrfs デバイス

Btrfs では、特別なデバイスを作成する必要はありません。

16.1.2. LVM バックエンド

プール、ボリューム、およびスナップショットは、LVM で作成できます。以下の定義は、LVM の観点からのものです。

16.1.2.1. LVM プール

*LVM プールは、LVM ボリュームグループと同じです。つまり、デバイスのグループ化と、新しい論理ボリュームを LVM プールから作成できます。デフォルトの LVM プール名は **lvm_pool** です。*

16.1.2.2. LVM ボリューム

LVM ボリュームは、通常の論理ボリュームと同じです。

16.1.2.3. LVM スナップショット

*LVM ボリュームからスナップショットを作成すると、他の LVM ボリュームと同様に処理できる新しい **スナップショット** ボリュームが作成されます。Btrfs とは異なり、LVM ではスナップショットと通常のボリュームを区別できるため、スナップショット名を特定のパターンに一致させる必要はありません。*

16.1.2.4. LVM デバイス

SSM を使用すると、ユーザーに対して物理デバイスに LVM バックエンドを透過的に作成する必要があります。

16.1.3. 暗号化バックエンド

*SSM の crypt バックエンドは、**cryptsetup** および **dm-crypt ターゲット** を使用して暗号化されたボリュームを管理します。暗号化バックエンドは、通常のブロックデバイス (または LVM や MD ボリュームなどのその他のボリューム) で暗号化ボリュームを作成するための通常のバックエンドとして使用することも、暗号化した LVM ボリュームを 1 つの手順で作成することもできます。*

暗号化バックエンドで作成できるのはボリュームのみです。プールには対応しておらず、特別なデバイスは必要ありません。

以下のセクションでは、暗号化の観点からボリュームとスナップショットを定義します。

16.1.3.1. 暗号化ボリューム

crypt ボリュームは **dm-crypt** により作成され、暗号化されていない形式で元の暗号化デバイスのデータを表します。RAID またはデバイスの連結には対応していません。

luks と plain の 2 つのモード、つまり拡張機能に対応しています。Luks はデフォルトで使用されます。拡張機能の詳細は、**man cryptsetup** を参照してください。

16.1.3.2. 暗号化スナップショット

暗号化バックエンドはスナップショットに対応していませんが、暗号化したボリュームが LVM ボリュームの上に作成されている場合は、そのボリューム自体をスナップショットすることができます。その後、**cryptsetup** を使用してスナップショットを開くことができます。

16.1.4. 複数のデバイス (MD) のバックエンド

現在、MD バックエンドは、システム内の MD ボリュームに関する情報の収集に限られています。

16.2. 一般的な SSM タスク

次のセクションでは、一般的な SSM タスクについて説明します。

16.2.1. SSM のインストール

SSM をインストールするには、次のコマンドを使用します。

```
# yum install system-storage-manager
```

サポート対象のパッケージがインストールされている場合に限り、有効になっているバックエンドが複数あります。

- LVM バックエンドには **lvm2** パッケージが必要です。
- Btrfs バックエンドには **btrfs-progs** パッケージが必要です。
- Crypt バックエンドには、**device-mapper** パッケージおよび **cryptsetup** パッケージが必要です。

16.2.2. 検出されたすべてのデバイスに関する情報の表示

検出されたすべてのデバイス、プール、ボリューム、およびスナップショットに関する情報を表示するには、**list** コマンドを使用します。オプションのない **ssm list** コマンドは、以下の出力を表示します。

```
# ssm list
----------------------------------------------------------
Device        Free      Used      Total  Pool  Mount point
----------------------------------------------------------
/dev/sda                          2.00 GB        PARTITIONED
/dev/sda1                         47.83 MB       /test
```

```
/dev/vda                  15.00 GB       PARTITIONED
/dev/vda1                 500.00 MB      /boot
/dev/vda2  0.00 KB  14.51 GB   14.51 GB  rhel
----------------------------------------------------------
-------------------------------------------------
Pool  Type  Devices     Free      Used     Total
-------------------------------------------------
rhel  lvm   1       0.00 KB  14.51 GB  14.51 GB
-------------------------------------------------
------------------------------------------------------------------------------
Volume         Pool  Volume size  FS     FS size       Free  Type    Mount point
------------------------------------------------------------------------------
/dev/rhel/root  rhel     13.53 GB  xfs   13.52 GB    9.64 GB  linear  /
/dev/rhel/swap  rhel   1000.00 MB                             linear
/dev/sda1              47.83 MB  xfs   44.50 MB   44.41 MB  part    /test
/dev/vda1             500.00 MB  xfs  496.67 MB  403.56 MB  part    /boot
------------------------------------------------------------------------------
```

この表示は、引数を使用してさらに絞り込んで、表示するものを指定できます。利用可能なオプションの一覧は、**ssm list --help** コマンドで確認できます。

注記

指定した引数によっては、SSM がすべてを表示しない場合があります。

- **devices** 引数または **dev** 引数を実行すると、デバイスの一部が省略されます。たとえば、CDRoms デバイスや DM/MD デバイスは、ボリュームとして記載されているときに意図的に非表示になります。
- バックエンドの中にはスナップショットに対応していないものや、通常のボリュームとスナップショットを区別できないものがあります。これらのバックエンドのいずれかで **snapshot** 引数を実行すると、SSM はスナップショットを識別するためにボリューム名を認識しようとします。SSM の正規表現がスナップショットパターンと一致しない場合、スナップショットは認識されません。
- メインの Btrfs ボリューム (ファイルシステム自体) を除き、マウントされていない Btrfs ボリュームは表示されません。

16.2.3. 新しいプール、論理ボリューム、およびファイルシステムの作成

本セクションでは、デバイス **/dev/vdb** および **/dev/vdc**、1G の論理ボリューム、および XFS ファイルシステムを持つデフォルト名で新しいプールが作成されます。

このシナリオを作成するコマンドは **ssm create --fs xfs -s 1G /dev/vdb /dev/vdc** です。以下のオプションが使用されます。

- **--fs** オプションは、必要なファイルシステムのタイプを指定します。現在対応しているファイルシステムの種類は次のとおりです。
  - ext3
  - ext4
  - xfs
  - btrfs

- *-s は、論理ボリュームのサイズを指定します。単位を定義するために、次の接尾辞がサポートされています。*
  - *キロバイトの場合は k または K*
  - *M または m （メガバイト）*
  - *ギガバイトの場合は G または g*
  - *テラバイトの場合は T または t*
  - *p また は p （ペタバイト）*
  - *e または e （エクサバイト）*
- *また、-s オプションを使用すると、新しいサイズをパーセンテージで指定できます。例を参照してください。*
  - *プールサイズの合計 10 パーセントで 10 %*
  - *10 %FREE: 空きプール領域の 10 パーセントの場合*
  - *使用されているプール領域の 10 %USED*

リストされた 2 つのデバイス(/dev/vdb と /dev/vdc)は、作成する 2 つのデバイスです。

```
# ssm create --fs xfs -s 1G /dev/vdb /dev/vdc
  Physical volume "/dev/vdb" successfully created
  Physical volume "/dev/vdc" successfully created
  Volume group "lvm_pool" successfully created
  Logical volume "lvol001" created
```

ssm コマンド には、便利なオプションが 2 つあります。1 つ目は -p pool コマンドです。ボリュームを作成するプールを指定します。存在しない場合は、SSM により作成されます。これは、指定され

た例では省略されており、SSM はデフォルトの名前 lvm_pool を使用していました。ただし、特定の名前を使用して、既存の命名規則に適合させるには、-p オプションを使用する必要があります。

2 つ目の便利なオプションは、-n name コマンドです。新しく作成した論理ボリュームの名前を指定します。-p と同様に、これは特定の名前を使用して既存の命名規則に適合させるために必要です。

この 2 つのオプションの例を以下に示します。

```
# ssm create --fs xfs -p new_pool -n XFS_Volume /dev/vdd
  Volume group "new_pool" successfully created
  Logical volume "XFS_Volume" created
```

SSM は、1 つのコマンドだけで、2 つの物理ボリューム、プール、および論理ボリュームを作成しました。

16.2.4. ファイルシステムの整合性の確認

ssm check コマンドは、ボリューム上でファイルシステムの整合性をチェックします。複数のボリュームを指定して確認することもできます。ボリュームにファイルシステムがない場合は、ボリュームがスキップされます。

ボリューム lvol001 内のすべてのデバイスをチェックするには、ssm check /dev/lvm_pool/lvol001 コマンドを実行します。

```
# ssm check /dev/lvm_pool/lvol001
Checking xfs file system on '/dev/mapper/lvm_pool-lvol001'.
Phase 1 - find and verify superblock...
Phase 2 - using internal log
        - scan filesystem freespace and inode maps...
        - found root inode chunk
Phase 3 - for each AG...
        - scan (but don't clear) agi unlinked lists...
        - process known inodes and perform inode discovery...
        - agno = 0
        - agno = 1
        - agno = 2
        - agno = 3
        - agno = 4
        - agno = 5
        - agno = 6
        - process newly discovered inodes...
Phase 4 - check for duplicate blocks...
        - setting up duplicate extent list...
        - check for inodes claiming duplicate blocks...
```

```
        - agno = 0
        - agno = 1
        - agno = 2
        - agno = 3
        - agno = 4
        - agno = 5
        - agno = 6
No modify flag set, skipping phase 5
Phase 6 - check inode connectivity...
        - traversing filesystem ...
        - traversal finished ...
        - moving disconnected inodes to lost+found ...
Phase 7 - verify link counts...
No modify flag set, skipping filesystem flush and exiting.
```

16.2.5. ボリュームのサイズを大きくする

ssm resize コマンドは、指定したボリュームおよびファイルシステムのサイズを変更します。ファイルシステムがない場合は、ボリューム自体のサイズのみが変更されます。

この例では、現在 **/dev/vdb** に論理ボリュームが **1** つあり、これは **900MB** の **lvol001** です。

```
# ssm list
----------------------------------------------------------
Device        Free      Used      Total  Pool      Mount point
----------------------------------------------------------
/dev/vda                       15.00 GB           PARTITIONED
/dev/vda1                      500.00 MB          /boot
/dev/vda2    0.00 KB   14.51 GB   14.51 GB  rhel
/dev/vdb   120.00 MB  900.00 MB    1.00 GB  lvm_pool
/dev/vdc                       1.00 GB
----------------------------------------------------------
-----------------------------------------------------
Pool      Type  Devices      Free      Used      Total
-----------------------------------------------------
lvm_pool  lvm   1        120.00 MB  900.00 MB  1020.00 MB
rhel      lvm   1          0.00 KB   14.51 GB    14.51 GB
-----------------------------------------------------
-------------------------------------------------------------------------------------
Volume                Pool      Volume size  FS     FS size       Free  Type    Mount point
-------------------------------------------------------------------------------------
/dev/rhel/root        rhel        13.53 GB  xfs   13.52 GB    9.64 GB  linear  /
/dev/rhel/swap        rhel       1000.00 MB                           linear
/dev/lvm_pool/lvol001  lvm_pool    900.00 MB  xfs  896.67 MB  896.54 MB  linear
/dev/vda1                       500.00 MB  xfs  496.67 MB  403.56 MB  part    /boot
-------------------------------------------------------------------------------------
```

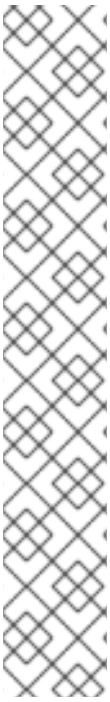
論理ボリュームをさらに **500MB** 増やす必要があります。これを行うには、プールにデバイスを追加する必要があります。

```
~]# ssm resize -s +500M /dev/lvm_pool/lvol001 /dev/vdc
  Physical volume "/dev/vdc" successfully created
  Volume group "lvm_pool" successfully extended
Phase 1 - find and verify superblock...
Phase 2 - using internal log
        - scan filesystem freespace and inode maps...
        - found root inode chunk
Phase 3 - for each AG...
        - scan (but don't clear) agi unlinked lists...
        - process known inodes and perform inode discovery...
        - agno = 0
        - agno = 1
        - agno = 2
        - agno = 3
        - process newly discovered inodes...
Phase 4 - check for duplicate blocks...
        - setting up duplicate extent list...
        - check for inodes claiming duplicate blocks...
        - agno = 0
        - agno = 1
        - agno = 2
        - agno = 3
No modify flag set, skipping phase 5
Phase 6 - check inode connectivity...
        - traversing filesystem ...
        - traversal finished ...
        - moving disconnected inodes to lost+found ...
Phase 7 - verify link counts...
No modify flag set, skipping filesystem flush and exiting.
  Extending logical volume lvol001 to 1.37 GiB
  Logical volume lvol001 successfully resized
meta-data=/dev/mapper/lvm_pool-lvol001 isize=256    agcount=4, agsize=57600 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=0
data     =                       bsize=4096   blocks=230400, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=0
log      =internal               bsize=4096   blocks=853, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
data blocks changed from 230400 to 358400
```

SSM はデバイスでチェックを実行し、指定した量だけボリュームを拡張します。これは、ssm list コマンドで検証できます。

```
# ssm list
----------------------------------------------------------------------
Device        Free       Used      Total  Pool      Mount point
----------------------------------------------------------------------
/dev/vda                          15.00 GB           PARTITIONED
/dev/vda1                         500.00 MB          /boot
/dev/vda2    0.00 KB    14.51 GB   14.51 GB  rhel
```

```
/dev/vdb     0.00 KB  1020.00 MB    1.00 GB  lvm_pool
/dev/vdc   640.00 MB   380.00 MB    1.00 GB  lvm_pool
---------------------------------------------------------------
---------------------------------------------------
Pool      Type  Devices      Free     Used     Total
---------------------------------------------------
lvm_pool  lvm   2        640.00 MB   1.37 GB   1.99 GB
rhel      lvm   1          0.00 KB  14.51 GB  14.51 GB
---------------------------------------------------
------------------------------------------------------------------------------------------
Volume               Pool      Volume size  FS      FS size       Free  Type    Mount point
------------------------------------------------------------------------------------------
/dev/rhel/root       rhel        13.53 GB  xfs    13.52 GB     9.64 GB  linear  /
/dev/rhel/swap       rhel      1000.00 MB                         linear
/dev/lvm_pool/lvol001  lvm_pool     1.37 GB  xfs     1.36 GB     1.36 GB  linear
/dev/vda1                       500.00 MB  xfs   496.67 MB   403.56 MB  part    /boot
------------------------------------------------------------------------------------------
```



注記

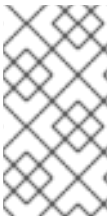
LVM ボリュームのサイズは縮小のみ可能で、その他のボリュームタイプでは対応していません。これは、+ の代わりに - を使用して行います。たとえば、LVM ボリュームのサイズを 50M 減らすには、次のコマンドを実行します。

```
# ssm resize -s-50M /dev/lvm_pool/lvol002
  Rounding size to boundary between physical extents: 972.00 MiB
  WARNING: Reducing active logical volume to 972.00 MiB
  THIS MAY DESTROY YOUR DATA (filesystem etc.)
Do you really want to reduce lvol002? [y/n]: y
  Reducing logical volume lvol002 to 972.00 MiB
  Logical volume lvol002 successfully resized
```

+ または - を指定しないと、値は絶対として取得されます。

16.2.6. スナップショット

既存のボリュームのスナップショットを作成するには、ssm snapshot コマンドを使用します。



注記

ボリュームが属するバックエンドがスナップショットをサポートしていない場合、この操作は失敗します。

lvol001 のスナップショットを作成するには、次のコマンドを使用します。

```
# ssm snapshot /dev/lvm_pool/lvol001
  Logical volume "snap20150519T130900" created
```

これを確認するには、ssm list を使用して、追加のスナップショットセクションをメモします。

```
# ssm list
----------------------------------------------------------
Device        Free       Used      Total  Pool      Mount point
----------------------------------------------------------
/dev/vda                          15.00 GB           PARTITIONED
/dev/vda1                         500.00 MB          /boot
/dev/vda2  0.00 KB    14.51 GB   14.51 GB  rhel
/dev/vdb   0.00 KB  1020.00 MB    1.00 GB  lvm_pool
/dev/vdc                          1.00 GB
----------------------------------------------------------
-----------------------------------------------------
Pool      Type  Devices     Free       Used       Total
-----------------------------------------------------
lvm_pool  lvm   1        0.00 KB  1020.00 MB  1020.00 MB
rhel      lvm   1        0.00 KB    14.51 GB    14.51 GB
-----------------------------------------------------
-------------------------------------------------------------------------------------------
Volume                Pool      Volume size  FS      FS size       Free  Type    Mount point
-------------------------------------------------------------------------------------------
/dev/rhel/root        rhel         13.53 GB  xfs    13.52 GB     9.64 GB  linear  /
/dev/rhel/swap        rhel       1000.00 MB                              linear
/dev/lvm_pool/lvol001  lvm_pool    900.00 MB  xfs   896.67 MB   896.54 MB  linear
/dev/vda1                        500.00 MB  xfs   496.67 MB   403.56 MB  part    /boot
-------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------
Snapshot                        Origin   Pool      Volume size     Size  Type
-----------------------------------------------------------------------------
/dev/lvm_pool/snap20150519T130900  lvol001  lvm_pool    120.00 MB  0.00 KB  linear
-----------------------------------------------------------------------------
```

16.2.7. 項目の削除

ssm remove は、デバイス、プール、またはボリュームのいずれかの項目を削除するために使用されます。

注記

削除したデバイスがプールで使用されている場合は、失敗します。これは、-f 引数を使用して強制できます。

ボリュームを削除したときにマウントされていた場合は、マウントに失敗します。デバイスとは異なり、-f 引数を指定して強制することはできません。

lvm_pool とその中のものをすべて削除するには、次のコマンドを使用します。

```
# ssm remove lvm_pool
Do you really want to remove volume group "lvm_pool" containing 2 logical volumes? [y/n]: y
Do you really want to remove active logical volume snap20150519T130900? [y/n]: y
  Logical volume "snap20150519T130900" successfully removed
Do you really want to remove active logical volume lvol001? [y/n]: y
  Logical volume "lvol001" successfully removed
  Volume group "lvm_pool" successfully removed
```

16.3. SSM リソース

SSM の詳細は、以下の資料を参照してください。

- man ssm ページには、適切な説明と例が記載されています。また、記載するコマンドやオプションの詳細については、こちらを参照してください。
- SSM のローカルドキュメントは、doc/ ディレクトリーに保存されます。
- SSM Wiki は、http://storagemanager.sourceforge.net/index.html からアクセスできます。
- https://lists.sourceforge.net/lists/listinfo/storagemanager-devel からメーリングリストのサブスクライブ、http://sourceforge.net/mailarchive/forum.php?forum_name=storagemanager-devel からメーリングリストのアーカイブが可能です。メーリングリストは、開発者がコミュニケーションをとる場所です。現在、ユーザーのメーリングリストはありませんので、そこにも質問を投稿してください。

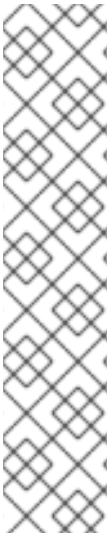
第17章 ディスククォータ

ディスク領域はディスククォータによって制限できます。ディスククォータは、ユーザーが過度のディスク領域を消費するか、パーティションが満杯になる前にシステム管理者に警告をします。

ディスククォータは、ユーザーグループにも個別のユーザーにも設定できます。これにより、ユーザーが参加しているプロジェクトに割り振られた領域 (プロジェクトごとに所有グループが存在すると想定) とは別に、ユーザー固有のファイル (電子メールなど) に割り振った領域を管理することが可能になります。

さらにクォータは、消費されるディスクブロックの数の制御だけでなく、inode (UNIX ファイルシステム内のファイルに関する情報を含むデータ構造) の数も制御するように設定できます。inode はファイル関連の情報を組み込むように使用されるため、これが作成されるファイルの数を制御することも可能にします。

ディスククォータを実装するには、quota RPM をインストールする必要があります。



注記

本章はすべてのファイルシステムを対象としていますが、一部のファイルシステムには独自のクォータ管理ツールがあります。該当するファイルシステムについては、対応する説明を参照してください。

XFS ファイルシステムの場合は、「XFS クォータ管理」 を参照してください。

Btrfs にはディスククォータがないため、使用できません。

17.1. ディスククォータの設定

ディスククォータを実装するには、以下の手順を行います。

1. /etc/fstab ファイルを変更して、ファイルシステムごとのクォータを有効にします。
2. ファイルシステムを再マウントします。

3.
クォータデータベースファイルを作成して、ディスク使用状況テーブルを生成します。

4.
クォータポリシーを割り当てます。

この各手順は、以下のセクションで詳しく説明します。

17.1.1. クォータの有効化

手順17.1 クォータの有効化

1.
root でログインします。

2.
/etc/fstab ファイルを編集します。

3.
クォータを必要とするファイルシステムに usrquota オプションまたは grpquota オプションのいずれか、または両方のオプションを追加します。

例17.1 /etc/fstabを編集します。

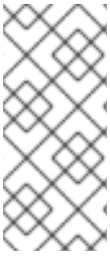
たとえば、テキストエディター vim を使用するには、以下を入力します。

```
# vim /etc/fstab
```

例17.2 クォータの追加

```
/dev/VolGroup00/LogVol00 /        ext3    defaults        1 1
LABEL=/boot              /boot    ext3    defaults        1 2
none                     /dev/pts  devpts  gid=5,mode=620  0 0
none                     /dev/shm  tmpfs   defaults        0 0
none                     /proc     proc    defaults        0 0
none                     /sys      sysfs   defaults        0 0
/dev/VolGroup00/LogVol02 /home     ext3    defaults,usrquota,grpquota  1 2
/dev/VolGroup00/LogVol01 swap      swap    defaults        0 0 . . .
```

この例では、/home ファイルシステムで、ユーザーおよびグループの両方のクォータが有効になっています。



注記

以下の例では、Red Hat Enterprise Linux のインストール時に別の /home パーティションが作成されたことを前提としています。root (/)パーティションは、/etc/fstab ファイルでクォータポリシーを設定するために使用できます。

17.1.2. ファイルシステムの再マウント

usrquota オプションまたは grpquota オプション、または両方のオプションを追加したら、fstab エントリーが変更された各ファイルシステムを再マウントします。ファイルシステムがどのプロセスでも使用されていない場合は、以下のいずれかの方法を使用します。

- *umount コマンドの後に mount コマンドを実行してファイルシステムを再 マウント します。さまざまなファイルシステムタイプのマウントとアンマウントを行う具体的な構文は、umount と mount の man ページを参照してください。*

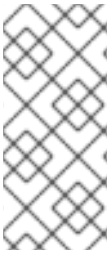
- *mount -o remount file-systemコマンド(file-system はファイルシステムの名前)を実行して、ファイルシステムを再マウントします。たとえば、/home ファイルシステムを再マウントするには、mount -o remount /home コマンドを実行します。*

ファイルシステムが現在使用中の場合、そのファイルシステムを再マウントする最も簡単な方法は、システムを再起動することです。

17.1.3. クォータデータベースファイルの作成

クォータが有効にされたそれぞれのファイルシステムを再マウントした後は、quotacheck コマンドを実行します。

quotacheck コマンドは、クォータが有効なファイルシステムを検証し、現在のディスク使用状況のテーブルをファイルシステムごとに構築します。このテーブルは、ディスク使用状況のオペレーティングシステム用コピーを更新するのに使用されます。また、ファイルシステムのディスククォータが更新されます。



注記

ディスク使用量のテーブルはマウント時に自動的に作成されるため、quotacheck コマンドは XFS には影響しません。詳細は、man ページの xfs_quota (8) を参照してください。

手順17.2 クォータデータベースファイルの作成

1. 次のコマンドを使用して、ファイルシステムにクォータファイルを作成します。

```
# quotacheck -cug /file system
```

2. 次のコマンドを使用して、ファイルシステムごとの現在のディスク使用量の表を生成します。

```
# quotacheck -avug
```

クォータファイルの作成に使用するオプションを以下に示します。

c

クォータファイルを、クォータを有効にしたファイルシステムごとに作成するように指定します。

u

ユーザークォータを確認します。

g

グループクォータを確認します。**-g** のみを指定すると、グループクォータファイルのみが作成されます。

-u オプションまたは **-g** オプションのいずれも指定しない場合は、ユーザークォータファイルのみが作成されます。

以下のオプションは、現在のディスク使用率の表を生成するために使用されます。

a

クォータが有効にされた、ローカルマウントのファイルシステムをすべてチェック

v

クォータチェックの進行状態について詳細情報を表示

u

ユーザーディスククォータの情報をチェック

g

グループディスククォータの情報をチェック

***quotacheck** の実行が終了すると、有効なクォータ (ユーザーまたはグループのいずれか、または両方) に対応するクォータファイルに、**/home** などの、クォータが有効になっているローカルにマウントされた各ファイルシステムのデータが取り込まれます。*

17.1.4. ユーザーごとのクォータ割り当て

*最後の手順は、**edquota** コマンドを使用したディスククォータの割り当てです。*

前提条件

- *ユーザーは、ユーザークォータを設定する前に存在する必要があります。*

手順17.3 ユーザーごとのクォータ割り当て

1. *ユーザーにクォータを割り当てるには、次のコマンドを使用します。*

```
# edquota username
```

__username__ を、クォータを割り当てるユーザーに置き換えます。

2.
ユーザーのクォータが設定されていることを確認するには、次のコマンドを使用します。

```
# quota username
```

例17.3 ユーザーへのクォータの割り当て

たとえば、/home パーティションの /etc/fstab でクォータが有効になっている場合（以下の例では/dev/VolGroup00/LogVol02 ）、コマンド edquota testuser を実行すると、システムのデフォルトとして設定されているエディターに次のように表示されます。

```
Disk quotas for user testuser (uid 501):
  Filesystem                   blocks     soft     hard    inodes   soft   hard
  /dev/VolGroup00/LogVol02  440436        0        0     37418      0      0
```

注記

EDITOR 環境変数で定義されるテキストエディターは、edquota により使用されます。エディターを変更するには、~/.bash_profile ファイルの EDITOR 環境変数を、選択したエディターのフルパスに設定します。

最初の列は、クォータが有効になっているファイルシステムの名前です。2 列目には、ユーザーが現在使用しているブロック数が示されます。その次の 2 列は、ファイルシステム上のユーザーのソフトブロック制限およびハードブロック制限を設定するのに使用されます。inodes 列には、ユーザーが現在使用している inode の数が表示されます。最後の 2 列は、ファイルシステムのユーザーに対するソフトおよびハードの inode 制限を設定するのに使用されます。

ハードブロック制限は、ユーザーまたはグループが使用できる最大ディスク容量 (絶対値) です。この制限に達すると、それ以上のディスク領域は使用できなくなります。

ソフトブロック制限は、使用可能な最大ディスク容量を定義します。ただし、ハード制限とは異なり、ソフト制限は一定時間超過する可能性があります。この時間は 猶予期間 として知られています。猶予期間の単位は、秒、分、時間、日、週、または月で表されます。

いずれかの値が 0 に設定されていると、その制限は設定されません。テキストエディターで必要な制限に変更します。

例17.4 必要な制限の変更

以下に例を示します。

```
Disk quotas for user testuser (uid 501):
Filesystem              blocks    soft    hard   inodes   soft   hard
/dev/VolGroup00/LogVol02  440436   500000   550000    37418     0     0
```

ユーザーのクォータが設定されていることを確認するには、以下のコマンドを使用します。

```
# quota testuser
Disk quotas for user username (uid 501):
  Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
    /dev/sdb    1000*   1000    1000            0       0       0
```

17.1.5. グループごとのクォータ割り当て

クォータは、グループごとに割り当てることもできます。

前提条件

- グループは、グループクォータを設定する前に存在している必要があります。

手順17.4 グループごとのクォータ割り当て

1. グループクォータを設定するには、次のコマンドを使用します。

```
# edquota -g groupname
```

2. グループクォータが設定されていることを確認するには、次のコマンドを使用します。

```
# quota -g groupname
```

例17.5 クォータのグループへの割り当て

たとえば、**devel** グループのグループクォータを設定するには、コマンドを使用します。

```
# edquota -g devel
```

このコマンドにより、グループの既存クォータがテキストエディターに表示されます。

```
Disk quotas for group devel (gid 505):
Filesystem                blocks    soft     hard    inodes    soft    hard
/dev/VolGroup00/LogVol02  440400       0        0     37418       0       0
```

この制限を変更して、ファイルを保存します。

グループクォータが設定されたことを確認するには、次のコマンドを使用します。

```
# quota -g devel
```

17.1.6. ソフト制限の猶予期間の設定

所定のクォータがソフトリミットを持つ場合、その猶予期間 (ソフトリミットを超過してもよい期間の値) は以下のコマンドで編集できます。

```
# edquota -t
```

このコマンドはユーザーまたはグループのいずれかの **inode** またはブロックのクォータに対して機能します。

重要

他の **edquota** コマンドは特定のユーザーまたはグループのクォータで動作しますが、**-t** オプションは、クォータが有効になっているすべてのファイルシステムで動作します。

17.2. ディスククォータの管理

クォータが実装されている場合、ほとんどは、クォータを超えているかどうかを監視し、クォータが正確であることを確認するという形で、いくつかのメンテナンスが必要になります。

ユーザーが繰り返しクォータを超過したり、常にソフト制限に達している場合、システム管理者には、ユーザーのタイプや、ユーザーの作業にディスク容量が及ぼす影響の度合に応じて 2 つの選択肢があります。管理者は、ユーザーが使用するディスク領域を節約する方法をわかるようにするか、ユーザーのディスククォータを拡大するかのいずれかを行うことができます。

17.2.1. 有効化と無効化

クォータはゼロに設定することなく、無効にすることができます。すべてのユーザーとグループのクォータをオフにするには、以下のコマンドを使用します。

```
# quotaoff -vaug
```

-u オプションまたは **-g** オプションのいずれも指定しない場合は、ユーザークォータのみが無効になります。**-g** のみを指定すると、グループクォータのみが無効になります。**-v** スイッチにより、コマンドの実行時に詳細なステータス情報が表示されます。

ユーザーおよびグループのクォータを再度有効にするには、次のコマンドを使用します。

```
# quotaon
```

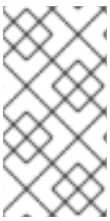
すべてのファイルシステムでユーザーおよびグループのクォータを有効にするには、次のコマンドを使用します。

```
# quotaon -vaug
```

-u オプションまたは **-g** オプションのいずれも指定しない場合は、ユーザークォータのみが有効になります。**-g** のみが指定されている場合は、グループのクォータのみが有効になります。

/home などの特定のファイルシステムのクォータを有効にするには、次のコマンドを使用します。

```
# quotaon -vug /home
```



注記

quotaon コマンドは、マウント時に自動的に実行されるため、**XFS** に常に必要であるとは限りません。詳細は、**man** ページの **quotaon (8)**を 参照してください。

17.2.2. ディスククォータに関するレポート

ディスク使用量レポートを作成するには、**repquota** ユーティリティーの実行が必要になります。

例17.6 repquota コマンドの出力

たとえば、repquota /home コマンドは以下の出力を生成します。

```
*** Report for user quotas on device /dev/mapper/VolGroup00-LogVol02
Block grace time: 7days; Inode grace time: 7days
  Block limits   File limits
User  used soft hard grace used soft hard grace
----------------------------------------------------------------------
root      --      36       0       0              4     0     0
kristin   --     540       0       0            125     0     0
testuser  --  440400  500000  550000          37418     0     0
```

クォータが有効なすべてのファイルシステム (オプション -a) のディスク使用状況レポートを表示するには、次のコマンドを使用します。

```
# repquota -a
```

レポートは読みやすいですが、いくつか説明しておくべき点があります。各ユーザーの後に表示される -- は、ブロックまたは inode の制限を超えるかを簡単に判断できます。いずれかのソフト制限を超えると、対応する - の代わりに + が表示されます。最初の - はブロックの制限を表し、2 つ目は inode の制限を表します。

通常、grace 列は空白です。ソフト制限が超過した場合、その列には猶予期間に残り時間量に相当する時間指定が含まれます。猶予期間が過ぎると、代わりに 何も 表示されません。

17.2.3. クォータの精度維持

システムクラッシュなどの理由で、ファイルシステムのマウントを正しく解除できない場合は、次のコマンドを実行する必要があります。

```
# quotacheck
```

ただし、システムがクラッシュしていないとしても、quotacheck は定期的に実行できます。quotacheck を定期的に実行する安全な方法には、以下が含まれます。

次回の再起動時に quotacheck を確実に実行する

ほとんどのシステムに最適な方法

この方法は、定期的に再起動する (ビジーな) 複数ユーザーシステムに最も適しています。

シェルスクリプトを /etc/cron.daily/ ディレクトリーまたは /etc/cron.weekly/ ディレクトリーに保存するか、次のコマンドを使用してシェルスクリプトをスケジュールします。

```
# crontab -e
```

crontab -e コマンドには、touch /forcequotacheck コマンドが含まれます。このスクリプトは root ディレクトリーに空の forcequotacheck ファイルを作成するため、起動時にシステムの init スクリプトがこれを検索します。このディレクトリーが検出されると、init スクリプトは quotacheck を実行します。その後、init スクリプトは /forcequotacheck ファイルを削除します。このように、cron でこのファイルが定期的に作成されるようにスケジュールすることにより、次回の再起動時に quotacheck を確実に実行することができます。

cron の詳細は、man cron を参照してください。

シングルユーザーモードで quotacheck を実行

quotacheck を安全に実行する別の方法として、クオータファイルのデータ破損の可能性を回避するためにシングルユーザーモードでシステムを起動して、以下のコマンドを実行する方法があります。

```
# quotaoff -vug /file_system
```

```
# quotacheck -vug /file_system
```

```
# quotaon -vug /file_system
```

実行中のシステムで quotacheck を実行

必要な場合には、いずれのユーザーもログインしておらず、チェックされているファイルシステムに開いているファイルがない状態のマシン上で quotacheck を実行することができます。quotacheck -vug file_system コマンドを実行します。このコマンドは、quotacheck が指定された file_system を読み取り専用に再マウントできない場合に失敗します。チェックの後には、ファイルシステムは読み込み/書き込みとして再マウントされることに注意してください。

警告

読み込み/書き込みでマウントされているライブのファイルシステム上での quotacheck の実行は、quota ファイルが破損する可能性があるため、推奨されません。

cron の設定に関する詳細は、man cron を参照してください。

17.3. ディスククオータのリファレンス

ディスククォータの詳細は、以下のコマンドの man ページを参照してください。

- quotacheck
- edquota
- repquota
- quota
- quotaon
- quotaoff

第18章 RAID (REDUNDANT ARRAY OF INDEPENDENT DISKS)

RAID の登場した背景には、容量が小さく手頃なディスクドライブを複数集めてアレイに結合させ、容量が大きく高価なドライブに負けないパフォーマンスと冗長性を実現しようとする動きがありました。この複数のデバイスからなるアレイは、コンピューター上では単一の論理ストレージユニットまたはドライブとして表されます。

RAID では複数のディスクに情報を拡散させることができます。ディスクのストライピング (RAID レベル 0)、ディスクのミラーリング (RAID レベル 1)、パリティーによるディスクのストライピング (RAID レベル 5) などの技術を使用して冗長性を得ながら待ち時間を抑え、帯域幅を増幅させることでハードディスクがクラッシュした場合の復元力を最大限に引き出します。

データは、一貫して同じサイズの大きさのチャンク (通常は 256K または 512K、 ただし他の値も可) に分割され、アレイ内の各ドライブに分散されます。各チャンクは、使用している RAID レベルに応じて、RAID アレイのハードドライブに書き込まれます。データが読み込まれるとこのプロセスが逆をたどります。その動作はアレイ内の複数のドライブがまるで一台の大容量ドライブであるかのように見えます。

システム管理者や大容量のデータを管理しているユーザーは、RAID 技術を使用することでメリットが得られます。RAID をデプロイする主な理由を以下に示します。

- 速度を高める
- 1 台の仮想ディスクを使用してストレージ容量を増加する
- ディスク障害によるデータ損失を最小限に抑える

18.1. RAID のタイプ

考えられる RAID アプローチには、ファームウェア RAID、ハードウェア RAID、およびソフトウェア RAID の 3 つがあります。

ファームウェア RAID

ファームウェア RAID (ATARAID とも呼ばれる) とは、ソフトウェア RAID の種類で、ファームウェアベースのメニューを使用して RAID セットを設定できます。このタイプの RAID で使用されるファームウェアは BIOS にもフックされるため、その RAID セットから起動できます。異なるベンダーは、異

なるオンディスクメタデータ形式を使用して、RAID セットのメンバーをマークします。Intel Matrix RAID は、ファームウェア RAID システムの一例を示しています。

ハードウェア RAID

ハードウェアベースのアレイは、RAID サブシステムをホストとは別に管理します。ホストに対して、1 RAID アレイごとに 1 つのディスクを表します。

ハードウェア RAID デバイスはシステムの内部にあっても外部にあってもかまいません。内部デバイスは一般的には特殊なコントローラーカードで設定され、RAID のタスクはオペレーティングシステムに対して透過的に処理されます。外部デバイスは一般的には SCSI、ファイバーチャネル、iSCSI、InfiniBand、または他の高速ネットワークなどの相互接続でシステムに接続され、システムには論理ボリュームとして表されます。

RAID コントローラーカードは、オペレーティングシステムへの SCSI コントローラーのように動作し、実際のドライブ通信をすべて処理します。ドライブはユーザーによって RAID コントローラーに接続されてから (通常の SCSI コントローラー同様に)、RAID コントローラーの設定に追加されます。オペレーティングシステムはこの違いを認識できません。

ソフトウェア RAID

ソフトウェア RAID では、カーネルディスク (ブロックデバイス) コード内に各種の RAID レベルを実装しています。高価ディスクコントローラーカードやホットスワップシャーシなど、最優先的な解決策を提供します。[2] 必須ではありません。ソフトウェア RAID は、SCSI ディスクだけでなく安価な IDE ディスクでも機能します。最近の高速な CPU でもソフトウェア RAID は一般的にハードウェア RAID より優れたパフォーマンスを見せます。

Linux カーネルには マルチディスク (MD) ドライバーが含まれ、これにより RAID ソリューションは完全にハードウェアに依存しなくてもよくなります。ソフトウェアベースのアレイのパフォーマンスは、サーバーの CPU パフォーマンスと負荷によって異なります。

Linux ソフトウェア RAID スタックの主な機能:

- マルチスレッド設計
- 再構築なしで Linux マシン間でのアレイの移植性

- アイドルシステムリソースを使用したバックグラウンドのアレイ再構築
- ホットスワップ可能なドライブのサポート
- CPU の自動検出でストリーミング SIMD サポートなどの特定 CPU の機能を活用
- アレイ内のディスク上にある不良セクターの自動修正
- RAID データの整合性を定期的にチェックしアレイの健全性を確保
- 重要なイベントで指定された電子メールアドレスに送信された電子メールアラートによるアレイのプロアクティブな監視
- 書き込みを集中としたビットマップ、アレイ全体を再同期させるのではなく再同期を必要とするディスク部分を正確にカーネルに認識させることで再同期イベントの速度を大幅に高速化
- チェックポイントを再同期して、再同期中にコンピューターを再起動すると、起動時に再同期が中断したところから再開し、最初からやり直さないようにします。
- インストール後のアレイのパラメーター変更が可能です。たとえば、新しいデバイスを追加しても、4 つのディスクの RAID5 アレイを 5 つのディスク RAID5 アレイに増大させることができます。この拡張操作はライブで行うため、新しいアレイで再インストールする必要はありません。

18.2. RAID レベルとリニアサポート

RAID は、レベル 0、1、4、5、6、10、リニアなどのさまざまな設定に対応します。これらの RAID タイプは以下のように定義されます。

レベル 0

RAID レベル 0 は、多くの場合ストライプ化と呼ばれていますが、パフォーマンス指向のストライプ化データマッピング技術です。これは、アレイに書き込まれるデータがストライプに分割さ

れ、アレイのメンバーディスク全体に書き込まれることを意味します。これにより低い固有コストで高い I/O パフォーマンスを実現できますが、冗長性は提供されません。

多くの RAID レベル 0 実装は、アレイ内の最小デバイスのサイズまで、メンバーデバイス全体にデータをストライプ化します。つまり、複数のデバイスのサイズが少し異なる場合、それぞれのデバイスは最小ドライブと同じサイズであるかのように処理されます。そのため、レベル 0 アレイの一般的なストレージ容量は、ハードウェア RAID 内の最小メンバーディスクの容量と同じか、アレイ内のディスク数またはパーティション数で乗算したソフトウェア RAID 内の最小メンバーパーティションの容量と同じになります。

レベル 1

RAID レベル 1 またはミラーリングは、他の RAID 形式よりも長く使用されています。レベル 1 は、アレイ内の各メンバーディスクに同一データを書き込むことで冗長性を提供し、各ディスクに対してミラーリングコピーをそのまま残します。ミラーリングは、データの可用性の単純化と高レベルにより、いまでも人気があります。レベル 1 は 2 つ以上のディスクと連携して、非常に優れたデータ信頼性を提供し、読み取り集中型のアプリケーションに対してパフォーマンスが向上しますが、比較的コストが高くなります。[3]

レベル 1 アレイのストレージ容量は、ハードウェア RAID 内でミラーリングされている最小サイズのハードディスクの容量と同じか、ソフトウェア RAID 内でミラーリングされている最小のパーティションと同じ容量になります。レベル 1 の冗長性は、すべての RAID タイプの中で最も高いレベルであり、アレイは 1 つのディスクのみで動作できます。

レベル 4

レベル 4 でパリティーを使用 [4] データを保護するため、1 つのディスクドライブで連結します。専用パリティーディスクは RAID アレイへのすべての書き込みトランザクションに固有のボトルネックを表すため、レベル 4 は、ライトバックキャッシュなどの付随技術なしで、またはシステム管理者がこれを使用してソフトウェア RAID デバイスを意図的に設計する特定の状況で使用されることはほとんどありません。ボトルネック (配列にデータが入力されると、書き込みトランザクションがほとんどまたはまったくない配列など) を念頭に置いてください。RAID レベル 4 にはほとんど使用されないため、Anaconda ではこのオプションとしては使用できません。ただし、実際には必要な場合は、ユーザーが手動で作成できます。

ハードウェア RAID レベル 4 のストレージ容量は、最小メンバーパーティションの容量にパーティションの数を掛けて、-1 を引いたものになります。RAID レベル 4 アレイのパフォーマンスは常に非対称になります。つまり、読み込みは書き込みを上回ります。これは、パリティーを生成するときに書き込みが余分な CPU 帯域幅とメインメモリーの帯域幅を消費し、データだけでなくパリティーも書き込むため、実際のデータをディスクに書き込むときに余分なバス帯域幅も消費するためです。読み取りが必要なのは、アレイが劣化状態にない限り、パリティーではなくデータでを読み取るだけです。その結果、通常の動作条件下で同じ量のデータ転送を行う場合は、読み取りによってドライブへのトラフィック、またはコンピューターのバスを経由するトラフィックが少なくなります。

レベル 5

これは RAID の最も一般的なタイプです。RAID レベル 5 は、アレイのすべてのメンバーディスクドライブにパリティーを分散することにより、レベル 4 に固有の書き込みボトルネックを排除します。パリティー計算プロセス自体のみがパフォーマンスのボトルネックです。最新の CPU とソフトウェア RAID では、最近の CPU がパリティーが非常に高速になるため、通常はボトルネックではありません。ただし、ソフトウェア RAID5 アレイに多数のメンバーデバイスがあり、組み合わせたすべてのデバイス間のデータ転送速度が十分であれば、このボトルネックは再生できます。

レベル 4 と同様に、レベル 5 のパフォーマンスは非対称であり、読み取りは書き込みを大幅に上回ります。RAID レベル 5 のストレージ容量は、レベル 4 と同じです。

レベル 6

パフォーマンスではなくデータの冗長性と保存が最重要事項であるが、レベル 1 の領域の非効率性が許容できない場合は、これが RAID の一般的なレベルです。レベル 6 では、複雑なパリティースキームを使用して、アレイ内の 2 つのドライブから失われたドライブから復旧できます。複雑なパリティースキームにより、ソフトウェア RAID デバイスで CPU 幅が大幅に高くなり、書き込みトランザクションの際に増大度が高まります。したがって、レベル 6 はレベル 4 や 5 よりもパフォーマンスにおいて、非常に非対称です。

RAID レベル 6 アレイの合計容量は、RAID レベル 5 および 4 と同様に計算されますが、デバイス数から追加パリティーストレージ領域用に 2 つのデバイス (1 ではなく) を引きます。

レベル 10

この RAID レベルでは、レベル 0 のパフォーマンスとレベル 1 の冗長性を組み合わせます。また、2 を超えるデバイスを持つレベル 1 アレイで領域の一部を軽減するのに役立ちます。レベル 10 では、データごとに 2 つのコピーのみを格納するように設定された 3 ドライブアレイを作成することができます。これにより、全体用のアレイサイズを最小デバイスのみと同じサイズ (3 つのデバイス、レベル 1 アレイなど) ではなく、最小サイズのデバイスが 1.5 倍になります。

レベル 10 アレイを作成する際の使用可能なオプションの数が多く、特定のユースケースに適切なオプションを選択することが簡単ではないため、インストール時に作成するのは実用的とは言えません。コマンドラインの mdadm ツールを使用して手動で作成できます。オプションと、それぞれのパフォーマンスのトレードオフに関する詳細は、man md を参照してください。

リニア RAID

リニア RAID は、より大きな仮想ドライブを作成するドライブのグループ化です。リニア RAID では、あるメンバードライブからチャンクが順次割り当てられます。最初のドライブが完全に

満杯になったときにのみ次のドライブに移動します。これにより、メンバードライブ間の I/O 操作が分割される可能性はないため、パフォーマンスの向上は見られません。リニア RAID には冗長性がなく、信頼性は低下します。メンバードライブに障害が発生した場合は、アレイ全体を使用することはできません。容量はすべてのメンバーディスクの合計になります。

18.3. LINUX RAID サブシステム

Linux の RAID は以下のサブシステムから設定されます。

Linux ハードウェア RAID のコントローラードライバー

Linux には、ハードウェア RAID コントローラーに固有の RAID サブシステムがありません。特殊な RAID チップセットを使用するため、ハードウェア RAID コントローラーには独自のドライバーが含まれます。これらのドライバーにより、システムは通常のディスクとして RAID セットを検出できるようになります。

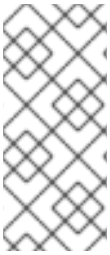
mdraid

mdraid サブシステムは、Linux 用のソフトウェア RAID ソリューションとして設計されており、Linux 下のソフトウェア RAID のソリューションでもあります。このサブシステムは独自のメタデータ形式を使用します。これは一般的にネイティブの mdraid メタデータと呼ばれます。

mdraid は、外部メタデータとして知られる他のメタデータ形式にも対応しています。Red Hat Enterprise Linux 7 は、外部メタデータで mdraid を使用して、ISW/IMSM (Intel ファームウェア RAID)セットにアクセスします。mdraid セットは、mdadm ユーティリティーを使用して設定および制御します。

dmraid

dmraid ツールは、さまざまなファームウェア RAID 実装で使用されます。dmraid は Intel ファームウェア RAID にも対応していますが、Red Hat Enterprise Linux 7 は mdraid を使用して Intel ファームウェア RAID セットにアクセスします。



注記

dmraid は、Red Hat Enterprise Linux 7.5 リリース以降非推奨になりました。Red Hat Enterprise Linux の将来のメジャーリリースで削除される予定です。詳細は、Red Hat Enterprise Linux 7.5 リリースノートの 非推奨の機能 を参照してください。

18.4. ANACONDA インストーラーでの RAID のサポート

Anaconda インストーラーは、システム上のハードウェアとファームウェア RAID セットを自動的に検出し、インストールで使用できるようにします。Anaconda は、mdraid を使用したソフトウェア RAID にも対応しており、既存の mdraid セットを認識できます。

Anaconda は、インストール時に RAID セットを作成するためのユーティリティーを提供します。ただし、このユーティリティーは、（ディスク全体ではなく）パーティションのみを新しいセットのメンバーにします。セットにディスク全体を使用するには、ディスク全体にまたがる 1 つのパーティションを作成し、そのパーティションを RAID セットのメンバーとして使用します。

root ファイルシステムが RAID セットを使用する場合、Anaconda は特殊なカーネルコマンドラインオプションをブートローダー設定に追加し、root ファイルシステムを検索する前に、どの RAID セットをアクティブにするかを initrd に指示します。

インストール時の RAID の設定方法は、Red Hat Enterprise Linux 7 Installation Guide を参照してください。

18.5. インストール後のルートディスクの RAID1 への変換

Red Hat Enterprise Linux 7 のインストール後に、RAID 以外の root ディスクを RAID1 ミラーに変換する必要がある場合は、Red Hat ナレッジベースの記事 How do I convert my root disk to RAID1 after installation of Red Hat Enterprise Linux 7? を参照してください。

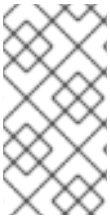
PowerPC (PPC) アーキテクチャーでは、以下の追加手順を行う必要があります。

1. PowerPC Reference Platform (PReP)ブートパーティションのコンテンツを /dev/sda1 から /dev/sdb1 にコピーします。

```
# dd if=/dev/sda1 of=/dev/sdb1
```

2. 両方のディスクの最初のパーティションで Prep フラグおよび boot フラグを更新します。

```
$ parted /dev/sda set 1 prep on
$ parted /dev/sda set 1 boot on

$ parted /dev/sdb set 1 prep on
$ parted /dev/sdb set 1 boot on
```



注記

PowerPC マシンでは grub2-install /dev/sda コマンドを実行しても動作せず、エラーが返されますが、システムは想定どおりに起動します。

18.6. RAID セットの設定

一般的には、ほとんどの RAID セットがその作成時にファームウェアメニューやインストーラーを使用して設定されます。システムのインストール後、できればマシンを再起動したりファームウェアメニューに入らずに RAID セットの作成や変更を行う必要が生じることがあります。

RAID セットを簡単に設定したり、ディスクの追加後でも新たなセットを定義したりすることができるハードウェア RAID コントローラーがあります。これらのコントローラーには標準の API がないためドライバー固有のユーティリティーを使用する必要があります。詳細は、ハードウェア RAID コントローラーのドライバーのドキュメントを参照してください。

mdadm

mdadm コマンドラインツールは、Linux でソフトウェア RAID を管理するために使用されます(mdraid)。さまざまな mdadm モードとオプションの詳細は、man mdadm を参照してください。の man ページには、ソフトウェア RAID アレイの作成、監視、アセンブルなどの一般的な操作に役立つ例も含まれています。

dmraid

名前が示すように、dmraid はデバイスマッパー RAID セットの管理に使用されます。dmraid ツールは、複数のメタデータ形式のハンドラーを使用して ATARAID デバイスを検索し、それぞれがさまざまな形式をサポートしています。対応している形式の完全なリストについては、dmraid -l を実行します。

「Linux RAID サブシステム」 で前述したように、dmraid ツールは作成後に RAID セットを設定できません。dmraid の使用に関する詳細は、 man dmraid を参照してください。

18.7. 高度な RAID デバイスの作成

インストール完了後では作成できないアレイ上にオペレーティングシステムをインストールしたい場合があります。通常、これは複雑な RAID デバイスに /boot または root ファイルシステムアレイを設定することを意味します。このような場合には、Anaconda で対応していないアレイオプションの使用が必要になる場合があります。これを回避するには、以下の手順を行います。

手順18.1 高度な RAID デバイスの作成

1. インストールディスクを挿入します。

2. 初回起動時に、Install または Upgrade の代わりに Rescue Mode を選択します。システムがレスキューモード で完全に起動すると、コマンドラインターミナルが表示されます。

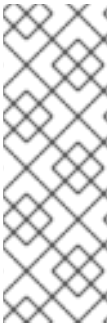
3. このターミナルから parted を使用して、ターゲットハードドライブに RAID パーティションを作成します。次に、mdadm を使用して、利用可能なすべての設定およびオプションを使用して、これらのパーティションから RAID アレイを手動で作成します。これらの方法の詳細については、13章Partitions、man parted、および man mdadm を参照してください。

4. アレイを作成したら、必要に応じてアレイにファイルシステムを作成することもできます。

5. コンピューターを再起動します。今回は Install または Upgrade を選択して通常どおりインストールします。Anaconda はシステム内のディスクを検索すると、既存の RAID デバイスを見つけます。

6. システムのディスクの使用方法を尋ねたら、カスタムレイアウト を選択して 次へ をクリックし ます。デバイスリストに、既存の MD RAID デバイスが表示されます。

7. RAID デバイスを選択し、編集 をクリックしてそのマウントポイントと（必要に応じて）使用するファイルシステムのタイプを設定し、完了 をクリックし ます。Anaconda は、この既存の RAID デバイスにインストールを実行し、レスキューモード で作成したときに選択したカスタムオプションを保持 します。



注記

インストーラーの制限されたレスキューモード には man ページは含まれません。man mdadm と man md の両方には、カスタム RAID アレイを作成するための有用な情報が含まれていますが、回避策全体で必要になる場合があります。そのため、これらの man ページが存在するマシンにアクセスできるか、レスキューモード で起動してカスタムアレイを作成する前にそれらを出力するのに役立ちます。

[2] ホットスワップシャーシを使用すると、システムの電源を切らずにハードドライブを削除できます。

[3]

RAID レベル 1 は、アレイ内のすべてのディスクに同じ情報を書き込むため、コストが高まり、データの信頼性を提供しますが、レベル 5 のようなパリティーベースの RAID レベルよりもはやくなり領域の効率が低くなります。ただし、この領域の非効率性にはパフォーマンス上の利点があります。パリティーベースの RAID レベルは、パリティーを生成するためにかなり多くの CPU 電力を消費しますが、RAID レベル 1 は単に同じデータを、CPU オーバーヘッドが非常に少ない複数の RAID メンバーに同じデータを複数回書き込むだけです。そのため、RAID レベル 1 は、ソフトウェア RAID が使用されているマシンや、マシンの CPU リソースが一貫して RAID アクティビティー以外の操作でアレイ化されます。

[4]

パリティー情報は、アレイ内の残りのメンバーディスクのコンテンツに基づいて計算されます。この情報は、アレイ内のいずれかのディスクに障害が発生した場合にデータの再構築に使用できます。その後、再構築されたデータを使用して、交換前に失敗したディスクに I/O 要求に対応でき、交換後に失敗したディスクを接続します。

第19章 MOUNT コマンドの使用

Linux、UNIX、および類似のオペレーティングシステムでは、さまざまなパーティションおよびリムーバブルデバイス (CD、DVD、USB フラッシュドライブなど) にあるファイルシステムをディレクトリーツリーの特定のポイント (マウントポイント) に接続して、再度切り離すことができます。ファイルシステムを接続またはデタッチするには、mount コマンドまたは umount コマンドを使用します。本章では、これらのコマンドの基本的な使い方や、マウントポイントの移動や共有サブツリーの作成などの高度なテクニックについてもいくつか扱います。

19.1. 現在マウントされているファイルシステムのリスト表示

現在接続されているファイルシステムをすべて表示するには、次のコマンドに引数を付けずに使用します。

```
$ mount
```

上記のコマンドで既知のマウントポイントのリストが表示されます。行ごとにデバイス名、ファイルシステムのタイプ、マウントしているディレクトリー、およびマウントオプションなどの情報が以下のような形で表示されます。

```
device on directory type type (options)
```

マウントされたファイルシステムをツリーのような形式で一覧表示できる findmnt ユーティリティーは、Red Hat Enterprise Linux 6.1 からも利用できます。現在接続されているファイルシステムをすべて表示するには、findmnt コマンドに引数を指定せずに実行します。

```
$ findmnt
```

19.1.1. ファイルシステムタイプの指定

デフォルトでは、mount コマンドの出力には、sysfs や tmpfs などのさまざまな仮想ファイルシステムが含まれます。特定のファイルシステムタイプを持つデバイスのみを表示するには、-t オプションを指定します。

```
$ mount -t type
```

同様に、findmnt コマンドを使用して、特定のファイルシステムを持つデバイスのみを表示するには、次のコマンドを実行します。

```
$ findmnt -t type
```

一般的なファイルシステムのタイプのリストは、表19.1「一般的なファイルシステムのタイプ」 を参照してください。使用例については 例19.1「現在マウントされている ext4 ファイルシステムの一覧表示」 を参照してください。

例19.1 現在マウントされている ext4 ファイルシステムの一覧表示

通常、/ パーティションと /boot パーティションの両方が ext4 を使用するようにフォーマットされます。このファイルシステムを使用するマウントポイントのみを表示するには、以下のコマンドを使用します。

```
$ mount -t ext4
/dev/sda2 on / type ext4 (rw)
/dev/sda1 on /boot type ext4 (rw)
```

findmnt コマンドを使用してマウントポイントを一覧表示するには、以下を入力します。

```
$ findmnt -t ext4
TARGET SOURCE    FSTYPE OPTIONS
/      /dev/sda2 ext4   rw,realtime,seclabel,barrier=1,data=ordered
/boot  /dev/sda1 ext4   rw,realtime,seclabel,barrier=1,data=ordered
```

19.2. ファイルシステムのマウント

特定のファイルシステムを添付するには、以下の形式で mount コマンドを使用します。

```
$ mount [option…] device directory
```

デバイス は、以下の方法で識別できます。

- ブロックデバイス へのフルパス： /dev/sda3など
- ユニバーサル一意識別子 (UUID): UUID=34795a28-ca6d-4fd8-a347-73671d0c19cb
- ボリュームラベル: たとえば、LABEL=homeです。

ファイルシステムがマウントされると、directory の元のコンテンツにアクセスできなくなります。

重要: ディレクトリーが使用されていないことを確認してください。

Linux では、すでにファイルシステムが接続されているディレクトリーに対してファイルシステムをマウントする動作が阻止されることはありません。特定のディレクトリーがマウントポイントとして動作するかどうかを確認するには、ディレクトリーを引数として findmnt ユーティリティーを実行し、終了コードを確認します。

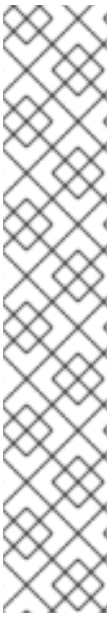
```
findmnt directory; echo $?
```

ディレクトリーにファイルシステムが接続されていない場合、指定されたコマンドは 1 を返します。

必要な情報をすべて入力せずに、つまりデバイス名、ターゲットディレクトリー、ファイルシステムタイプを入力せずに mount コマンドを実行すると、mount は /etc/fstab ファイルの内容を読み取り、指定したファイルシステムがリストにあるかどうかをチェックします。/etc/fstab ファイルには、選択したファイルシステムがマウントされるデバイス名およびディレクトリーの一覧と、ファイルシステムタイプおよびマウントオプションが含まれます。したがって、/etc/fstab で指定されたファイルシステムをマウントする場合は、以下のいずれかのオプションを選択できます。

```
mount [option...] directory
mount [option...] device
```

root でコマンドを実行しない限り、ファイルシステムのマウントにはパーミッションが必要であることに注意してください(「マウントオプションの指定」を参照してください)。



注記: 特定デバイスの UUID とラベルの確認

UUID を確認し、デバイスが特定のデバイスのラベルを使用しているかどうかを確認するには、以下の形式で blkid コマンドを使用します。

```
blkid device
```

たとえば、/dev/sda3 に関する情報を表示するには、次のコマンドを実行します。

```
# blkid /dev/sda3
/dev/sda3: LABEL="home" UUID="34795a28-ca6d-4fd8-a347-73671d0c19cb"
TYPE="ext3"
```

19.2.1. ファイルシステムタイプの指定

ほとんどの場合、mount はファイルシステムを自動的に検出します。ただし、NFS (Network File System)や CIFS (Common Internet File System)など、認識されない特定のファイルシステムがあり、手動で指定する必要があります。ファイルシステムのタイプを指定するには、以下の形式で mount コマンドを使用します。

```
$ mount -t type device directory
```

表19.1「一般的なファイルシステムのタイプ」は、mount コマンドで使用できる一般的なファイルシステムタイプの一覧を提供します。利用可能なファイルシステムタイプの完全なリストは、「man ページドキュメント」を参照してください。

表19.1 一般的なファイルシステムのタイプ

型	説明
ext2	ext2 ファイルシステム。
ext3	ext3 ファイルシステム。
ext4	ext4 ファイルシステム。
btrfs	btrfs ファイルシステム。
xfs	xfs ファイルシステム。
iso9660	ISO 9660 ファイルシステム。通常、これは光学メディア (通常は CD) で使用されます。
nfs	NFS ファイルシステム。通常、これはネットワーク経由でファイルにアクセスするために使用されます。
nfs4	NFSv4 ファイルシステム。通常、これはネットワーク経由でファイルにアクセスするために使用されます。
udf	UDF ファイルシステム。通常、これは光学メディア (通常は DVD) で使用されます。
vfat	FAT ファイルシステム。通常、これは Windows オペレーティングシステムを実行しているマシンや、USB フラッシュドライブやフロッピーディスクなどの特定のデジタルメディアで使用されます。

使用例は、例19.2「USB フラッシドライブのマウント」 を参照してください。

> 例19.2 USB フラッシドライブのマウント
>
> 多くの場合、古い USB フラッシュドライブは FAT ファイルシステムを使用します。このようなドライブが /dev/sdc1 デバイスを使用し、/media/flashdisk/ ディレクトリーが存在すると仮定して、root で次のコマンドを実行します。
>
> ```
> ~]# mount -t vfat /dev/sdc1 /media/flashdisk
> ```

19.2.2. マウントオプションの指定

追加のマウントオプションを指定するには、以下の形式でコマンドを実行します。

```
mount -o options device directory
```

複数のオプションを指定する場合は、コンマの後にスペースを挿入しないでください。または、mount は、スペースに続く値を追加のパラメーターとして誤って解釈します。

表19.2「一般的なマウントオプション」 一般的なマウントオプションのリストを提供します。利用可能なオプションのリストは、「man ページドキュメント」 に記載の関連する man ページを参照してください。

表19.2 一般的なマウントオプション

オプション	説明
async	ファイルシステム上での非同期の入/出力を許可します。
auto	mount -a コマンドを使用して、ファイルシステムを自動的にマウントできるようにします。
defaults	async,auto,dev,exec,nouser,rw,suid のエイリアスを指定します。
exec	特定のファイルシステムでのバイナリーファイルの実行を許可します。
loop	イメージをループデバイスとしてマウントします。

オプション	説明
noauto	デフォルトの動作では、mount -a コマンドを使用したファイルシステムの自動マウントが禁止されます。
noexec	特定のファイルシステムでのバイナリーファイルの実行は許可しません。
nouser	通常のユーザー(root以外のユーザー)によるファイルシステムのマウントとアンマウントを禁止します。
remount	ファイルシステムがすでにマウントされている場合は再度マウントを行います。
ro	読み取り専用でファイルシステムをマウントします。
rw	ファイルシステムを読み取りと書き込み両方でマウントします。
user	通常のユーザー（つまり root以外のユーザー）がファイルシステムをマウントおよびアンマウントすることを許可します。

使用例は、例19.3「ISO イメージのマウント」 を参照してください。

例19.3 ISO イメージのマウント

ISO イメージ (または一般的にはディスクイメージ) はループデバイスを使用することでマウントすることができます。Fedora 14 インストールディスクの ISO イメージが現在の作業ディレクトリーに存在し、/media/cdrom/ ディレクトリーが存在すると仮定して、次のコマンドを実行してイメージをこのディレクトリーにマウントします。

```
# mount -o ro,loop Fedora-14-x86_64-Live-Desktop.iso /media/cdrom
```

ISO9660 は設計上、読み取り専用のファイルシステムになっていることに注意してください。

19.2.3. マウントの共有

システム管理作業の中には、同じファイルシステムにディレクトリーツリー内の複数の場所からのアクセスしないといけない場合があります (chroot 環境を準備する場合など)。Linux では同じファイルシステムを複数のディレクトリーに必要なだけマウントすることが可能です。また、mount コマンドは、特定のマウントを複製する手段を提供する --bind オプションを実装します。以下のような使用法になります。

```
$ mount --bind old_directory new_directory
```

上記のコマンドにより、ユーザーはいずれの場所からでもファイルシステムにアクセスできるようになりますが、これは元のディレクトリー内にマウントされているファイルシステムには適用されません。これらのマウントも含めるには、以下のコマンドを使用します。

```
$ mount --rbind old_directory new_directory
```

さらに **Red Hat Enterprise Linux 7** では、可能な限り柔軟性を持たせるために、共有サブツリー と呼ばれる機能を実装しています。次の 4 種類のマウントを使用することができます。

共有マウント

共有マウントにより、任意のマウントポイントと同一の複製マウントポイントを作成することができます。マウントポイントが共有マウントとしてマークされている場合は、元のマウントポイント内のすべてのマウントが複製マウントポイントに反映されます (その逆も同様です)。マウントポイントのタイプを共有マウントに変更するには、シェルプロンプトで以下を入力します。

```
$ mount --make-shared mount_point
```

または、選択したマウントポイントとその下のすべてのマウントポイントのマウントタイプを変更する場合は、次のコマンドを実行します。

```
$ mount --make-rshared mount_point
```

使用例は、例19.4「共有マウントポイントの作成」 を参照してください。

例19.4 共有マウントポイントの作成

他のファイルシステムが一般的にマウントされる場所は 2 つあります。リムーバブルメディア用の **/media/** ディレクトリーと、一時的にマウントされたファイルシステム用の **/mnt/** ディレクトリーです。共有マウントを使用すると、この 2 つのディレクトリーで同じコンテンツを共有できます。これを行うには、**root** で、**/media/** ディレクトリーを共有としてマークします。

```
# mount --bind /media /media
# mount --make-shared /media
```

以下のコマンドを使用して、**/mnt/** に複製を作成します。

```
# mount --bind /media /mnt
```

/media/ 内のマウントが /mnt/ にも表示されることを確認できます。たとえば、CD-ROM ドライブに空でないメディアが含まれ、/media/cdrom/ ディレクトリーが存在する場合は、以下のコマンドを実行します。

```
# mount /dev/cdrom /media/cdrom
# ls /media/cdrom
EFI  GPL  isolinux  LiveOS
# ls /mnt/cdrom
EFI  GPL  isolinux  LiveOS
```

同様に、/mnt/ ディレクトリーにマウントされているファイルシステムが /media/ に反映されていることを確認できます。たとえば、/dev/sdc1 デバイスを使用する空でない USB フラッシュドライブが接続されており、/mnt/flashdisk/ ディレクトリーが存在する場合は、以下を入力します。

```
# # mount /dev/sdc1 /mnt/flashdisk
# ls /media/flashdisk
en-US  publican.cfg
# ls /mnt/flashdisk
en-US  publican.cfg
```

スレーブマウント

スレーブマウントにより、所定のマウントポイントの複製を作成する際に制限を課すことができます。マウントポイントがスレーブマウントとしてマークされている場合は、元のマウントポイント内のすべてのマウントが複製マウントポイントに反映されますが、スレーブマウント内のマウントは元のポイントに反映されません。マウントポイントのタイプをスレーブマウントに変更するには、シェルプロンプトで次を入力します。

```
mount --make-slave mount_point
```

選択したマウントポイントとその下にあるすべてのマウントポイントのマウントタイプを変更することも可能です。次のように入力します。

```
mount --make-rslave mount_point
```

使用例は、例19.5「スレーブマウントポイントの作成」 を参照してください。

例19.5 スレーブマウントポイントの作成

この例は、/media/ ディレクトリーのコンテンツが /mnt/ にも表示される方法を示していますが、/mnt/ ディレクトリーのマウントが /media/ に反映されません。root で、最初に /media/ ディレクトリーを共有としてマークします。

```
~]# mount --bind /media /media
~]# mount --make-shared /media
```

次に、その複製を /mnt/ で作成します。ただし、"slave" とマークします。

```
~]# mount --bind /media /mnt
~]# mount --make-slave /mnt
```

/media/ 内のマウントが /mnt/ にも表示されていることを確認します。たとえば、CD-ROM ドライブに空でないメディアが含まれ、/media/cdrom/ ディレクトリーが存在する場合は、以下のコマンドを実行します。

```
~]# mount /dev/cdrom /media/cdrom
~]# ls /media/cdrom
EFI  GPL  isolinux  LiveOS
~]# ls /mnt/cdrom
EFI  GPL  isolinux  LiveOS
```

また、/mnt/ ディレクトリーにマウントされているファイルシステムが /media/ に反映されていないことを確認します。たとえば、/dev/sdc1 デバイスを使用する空でない USB フラッシュドライブが接続されており、/mnt/flashdisk/ ディレクトリーが存在する場合は、以下を入力します。

```
~]# mount /dev/sdc1 /mnt/flashdisk
~]# ls /media/flashdisk
~]# ls /mnt/flashdisk
en-US  publican.cfg
```

プライベートマウント

プライベートマウントはマウントのデフォルトタイプであり、共有マウントやスレーブマウントと異なり、伝播イベントの受信や転送は一切行いません。マウントポイントを明示的にプライベートマウントにするには、シェルプロンプトで以下を入力します。

```
mount --make-private mount_point
```

または、選択したマウントポイントとその下にあるすべてのマウントポイントを変更すること

もできます。

```
mount --make-rprivate mount_point
```

使用例は、例19.6「プライベートマウントポイントの作成」 を参照してください。

例19.6 プライベートマウントポイントの作成

例19.4「共有マウントポイントの作成」 のシナリオを考慮に入れて、root で以下のコマンドを使用して、共有マウントポイントが事前に作成されていると仮定します。

```
~]# mount --bind /media /media
~]# mount --make-shared /media
~]# mount --bind /media /mnt
```

/mnt/ ディレクトリーをプライベートとしてマークするには、以下を入力します。

```
~]# mount --make-private /mnt
```

/media/ 内のマウントが /mnt/ に表示されないことを確認することができるようになりました。たとえば、CD-ROM ドライブに空でないメディアが含まれ、/media/cdrom/ ディレクトリーが存在する場合は、以下のコマンドを実行します。

```
~]# mount /dev/cdrom /media/cdrom
~]# ls /media/cdrom
EFI  GPL  isolinux  LiveOS
~]# ls /mnt/cdrom
~]#
```

/mnt/ ディレクトリーにマウントされているファイルシステムが /media/ に反映されていないことを確認することもできます。たとえば、/dev/sdc1 デバイスを使用する空でない USB フラッシュドライブが接続されており、/mnt/flashdisk/ ディレクトリーが存在する場合は、以下を入力します。

```
~]# mount /dev/sdc1 /mnt/flashdisk
~]# ls /media/flashdisk
~]# ls /mnt/flashdisk
en-US  publican.cfg
```

バインド不可能なマウント

任意のマウントポイントに対して一切複製が行われないようにするには、バインド不能のマウントを使用します。マウントポイントのタイプをバインド不能のマウントに変更するには、次のようにシェルプロンプトに入力します。

```
mount --make-unbindable mount_point
```

または、選択したマウントポイントとその下にあるすべてのマウントポイントを変更することもできます。

```
mount --make-runbindable mount_point
```

使用例は、例19.7「バインド不可能なマウントポイントの作成」 を参照してください。

例19.7 バインド不可能なマウントポイントの作成

/media/ ディレクトリーが共有されないようにするには、root として以下を実行します。

```
# mount --bind /media /media
# mount --make-unbindable /media
```

この方法だと、これ以降、このマウントの複製を作成しようとすると、以下のエラーが出て失敗します。

```
# mount --bind /media /mnt
mount: wrong fs type, bad option, bad superblock on /media,
missing codepage or helper program, or other error
In some cases useful info is found in syslog - try
dmesg | tail  or so
```

19.2.4. マウントポイントの移動

ファイルシステムがマウントされているディレクトリーを変更するには、次のコマンドを使用します。

```
# mount --move old_directory new_directory
```

使用例は、例19.8「既存の NFS マウントポイントの移動」 を参照してください。

例19.8 既存の NFS マウントポイントの移動

NFS ストレージにはユーザーディレクトリーが含まれ、すでに /mnt/userdirs/ にマウントされています。root で以下のコマンドを使用して、このマウントポイントを /home に移動します。

```
# mount --move /mnt/userdirs /home
```

マウントポイントが正しく移動したことを確認するため、両方のディレクトリーのコンテンツを表示させます。

```
# ls /mnt/userdirs
# ls /home
jill  joe
```

19.2.5. rootの読み取り専用パーミッションの設定

場合によっては、読み取り専用権限で root ファイルシステムをマウントする必要があります。ユースケースの例には、システムの予期せぬ電源切断後に行うセキュリティーの向上またはデータ整合性の保持が含まれます。

19.2.5.1. 起動時に読み取り専用パーミッションでマウントする ルート の設定

1. /etc/sysconfig/readonly-root ファイルで、READONLY を yes に変更します。

```
# Set to 'yes' to mount the file systems as read-only.
READONLY=yes
[output truncated]
```

2. /etc/fstab ファイルで、ルートエントリー(/)で デフォルト を ro に変更します。

```
/dev/mapper/luks-c376919e... / ext4 ro,x-systemd.device-timeout=0 1 1
```

3. /etc/default/grub ファイルの GRUB_CMDLINE_LINUX ディレクティブに ro を追加し、rw が含まれていないことを確認します。

```
GRUB_CMDLINE_LINUX="crashkernel=auto rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap
rhgb quiet ro"
```

4. GRUB2 設定ファイルを再作成します。

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
```

5. tmpfs ファイルシステムに書き込みパーミッションでマウントするファイルおよびディレクトリーを追加する必要がある場合は、/etc/rwtab.d/ ディレクトリーにテキストファイルを作成し、そこに設定を配置します。たとえば、/etc/example/file を書き込みパーミッションでマウントするには、以下の行を /etc/rwtab.d/ のサンプル ファイルに追加します。

```
files /etc/example/file
```

重要

tmpfs のファイルおよびディレクトリーへの変更は、再起動後は維持されません。

この手順の詳細は、「書き込みパーミッションを保持するファイルおよびディレクトリー」 を参照してください。

6. システムを再起動します。

19.2.5.2. root を即座に再マウントする

root (/)がシステム起動時に読み取り専用権限でマウントされている場合は、書き込み権限で再マウントできます。

```
# mount -o remount,rw /
```

これは、/ が読み取り専用パーミッションで誤ってマウントされている場合に特に便利です。

読み取り専用権限で / を再マウントするには、次のコマンドを実行します。

```
# mount -o remount,ro /
```

注記

このコマンドは、読み取り専用パーミッションで / 全体 をマウントします。より良い方法は、「起動時に読み取り専用パーミッションでマウントする ルート の設定」 で説明されているように、特定のファイルとディレクトリーを RAM にコピーして、それらの書き込み権限を保持することです。

19.2.5.3. 書き込みパーミッションを保持するファイルおよびディレクトリー

システムが正しく機能するためには、一部のファイルやディレクトリーで書き込みパーミッションが必要とされます。root が読み取り専用モードの場合は、tmpfs 一時ファイルシステムの RAM にマウントされます。このようなファイルおよびディレクトリーのデフォルトセットは、以下を含む /etc/rwtab ファイルから読み込まれます。

```
dirs /var/cache/man
dirs /var/gdm
[output truncated]
empty /tmp
empty /var/cache/foomatic
[output truncated]
files /etc/adjtime
files /etc/ntp.conf
[output truncated]
```

/etc/rwtab ファイルのエントリーは、以下の形式に従います。

```
how the file or directory is copied to tmpfs        path to the file or directory
```

ファイルまたはディレクトリーは、以下の 3 つの方法で tmpfs にコピーすることができます。

- *空の パス: 空のパスが tmpfs にコピーされます。例： empty /tmp*
- *dirs path: ディレクトリーツリーが空の状態で tmpfs にコピーされます。例： dirs /var/run*
- *ファイル パス: ファイルまたはディレクトリーツリーはそのまま tmpfs にコピーされます。例： files /etc/resolv.conf*

カスタムパスを /etc/rwtab.d/ に追加する場合も同じ形式が適用されます。

19.3. ファイルシステムのアンマウント

以前にマウントしたファイルシステムをデタッチするには、以下のいずれかの **umount** コマンドを使用します。

```
$ umount directory
$ umount device
```

root でログインしているときにこれを実行しない限り、ファイルシステムのマウントを解除するには、正しいパーミッションが利用可能である必要があることに注意してください。詳細は、「マウントオプションの指定」 を参照してください。使用例は、例19.9「CD のアンマウント」 を参照してください。

重要: ディレクトリーが使用されていないことを確認してください。

ファイルシステムが使用されている場合（たとえば、このファイルシステムでファイルを読み取っている場合や、カーネルで使用されている場合など）、**umount** コマンドを実行してもエラーを出して失敗します。ファイルシステムにアクセスしているプロセスを確認するには、以下の形式で **fuser** コマンドを使用します。

```
$ fuser -m directory
```

たとえば、**/media/cdrom/** ディレクトリーにマウントされているファイルシステムにアクセスしているプロセスの一覧を表示するには、次のコマンドを実行します。

```
$ fuser -m /media/cdrom
/media/cdrom:        1793  2013  2022  2435 10532c 10672c
```

例19.9 CD のアンマウント

以前に **/media/cdrom/** ディレクトリーにマウントされていた **CD** をアンマウントするには、次のコマンドを使用します。

```
$ umount /media/cdrom
```

19.4. マウント コマンドリファレンス

コマンドなどの詳細については、以下のドキュメントをご覧ください。

man ページドキュメント

- *man 8 mount: mount コマンドの man ページです。使い方などに関する詳細が記載されています。*
- *man 8 umount: umount コマンドの man ページです。使い方などに関する詳細が記載されています。*
- *man 8 findmnt: findmnt コマンドの man ページです。使い方などに関する詳細が記載されています。*
- *man 5 fstab: /etc/fstab ファイル形式の詳細な説明を示す man ページです。*

便利な Web サイト

- *『Shared subtrees』 — 共有サブツリーの概念について解説されている LWN の記事です。*

第20章 VOLUME_KEY 機能

volume_key 関数は、libvolume_key と volume_key の 2 つのツールを提供します。libvolume_key は、ストレージボリューム暗号化キーを操作し、ボリュームとは別に保存するためのライブラリーです。volume_key は、暗号化されたハードドライブへのアクセスを復元するために鍵とパスフレーズを抽出するために使用される、関連するコマンドラインツールです。

第一ユーザーがキーやパスワードを忘れてしまった、ユーザーが突然退職してしまった、ハードウェアまたはソフトウェアの障害で暗号化していたボリュームのヘッダーが破損したためデータを抽出する必要がある、などといった場合にこの機能は便利です。企業設定では、IT ヘルプデスクは、コンピューターをエンドユーザーに渡す前に volume_key を使用して暗号化キーをバックアップできます。

現在、volume_key は LUKS ボリュームの暗号化形式のみをサポートします。

注記

volume_key は、Red Hat Enterprise Linux 7 サーバーの標準インストールには含まれていません。インストール方法については は、http://fedoraproject.org/wiki/Disk_encryption_key_escrow_use_cases を参照してください。

20.1. VOLUME_KEY コマンド

volume_key の形式は次のとおりです。

```
volume_key [OPTION]... OPERAND
```

volume_key のオペランドおよび操作のモードは、以下のオプションのいずれかを指定して決定されます。

--save

このコマンドは、オペランドのボリューム [パケット] を想定しています。パケット が提供されている場合、volume_key はそこからキーとパスフレーズを抽出します。packet が指定されていない場合は、volume_key は ボリューム から鍵とパスフレーズを抽出し、必要に応じてユーザーに要求します。この鍵およびパスフレーズは、1 つ以上の出力パケットに格納されます。

--restore

このコマンドは、オペランドのボリュームパケットを想定します。次に volume を開き、packet のキーとパスフレーズを使用して volume を再度アクセス可能にし、必要に応じてユーザー (たとえば、ユーザーが新しいパスフレーズの入力を許可するなど) を求めるプロンプトを出します。

--setup-volume

このコマンドは、オペランドのボリュームパケット名 を想定しています。次に volume を開き、packet のキーとパスフレーズを使用して、復号されたデータを name として使用するための volume を設定します。

Name は、dm-crypt ボリュームの名前です。この操作により、復号化されたボリュームは /dev/mapper/名 として利用可能になります。

この動作は、たとえば、新しいパスフレーズを追加しても、ボリューム を永続的に変更することはありません。ユーザーは復号された volume にアクセスし、変更できます。

--reencrypt、--secrets、および --dump

この 3 つのコマンドは、出力方法が異なりますが、同様の機能を実行します。それらは、それぞれがオペランド パケット を必要とし、それぞれが パケット を開き、必要に応じて復号化します。--reencrypt は、情報を 1 つ以上の新しい出力パケットに保存します。--secrets は、パケットに含まれるキーとパスフレーズを出力します。--dump は パケット の内容を出力しますが、鍵とパスフレーズはデフォルトでは出力されません。これは、--with-secrets をコマンドに追加することで変更できます。--unencrypted コマンドを使用して、暗号化されていない部分（存在する場合）のみをダンプすることもできます。パスフレーズや秘密鍵のアクセスは必要ありません。

各オプションは、以下のように追加できます。

-o、--output packet

このコマンドは、デフォルトの鍵またはパスフレーズをパケットに書き込みます。デフォルトの鍵またはパスフレーズは、ボリューム形式によって異なります。有効期限が切れる可能性が低いものであることを確認し、--restore がボリュームへのアクセスを復元できるようにします。

--output-format format

このコマンドは、すべての出力パケットに指定された format 形式を使用します。現在、format は以下のいずれかになります。

- *非対称: CMS を使用してパケット全体を暗号化し、証明書を必要とします。*

- *asymmetric_wrap_secret_only: シークレットまたは鍵とパスフレーズのみをラップし、証明書を必要とします。*

- *passphrase: GPG を使用してパケット全体を暗号化し、パスフレーズを必要とします。*

--create-random-passphrase packet

このコマンドは、ランダムな英数字のパスフレーズを生成し、それを ボリューム に追加して (他のパスフレーズに影響を与えることなく)、このランダムなパスフレーズを パケット に格納します。

20.2. VOLUME_KEY を個別のユーザーとして使用する

volume_key は、以下の手順に従って暗号鍵を保存するために使用できます。

注記

このファイルのすべての例で、/path/to/volume は LUKS デバイスであり、そこに含まれるプレーンテキストデバイスではありません。blkid -s type /path/to/volume should report type="crypto_LUKS".

手順20.1 volume_key スタンドアロンの使用

1. *以下を実行します。*

```
volume_key --save /path/to/volume -o escrow-packet
```

次に、キーを保護するために escrow パケットパスフレーズを要求するプロンプトが表示されます。

2. 生成された escrow-packet ファイルを保存し、パスフレーズを忘れないようにします。

ボリュームパスフレーズを忘れた場合は、保存した escrow パケットを使用して、データへのアクセスを復元します。

手順20.2 escrow パケットを使用したデータへのアクセスの復元

1. volume_key を実行し、escrow パケットが利用できる環境でシステムを起動します（レスキューモードなど）。

2. 以下を実行します。

```
volume_key --restore /path/to/volume escrow-packet
```

escrow パケットの作成時に使用した escrow パケットパスフレーズと、ボリュームの新しいパスフレーズを尋ねるプロンプトが表示されます。

3. 選択したパスフレーズを使用してボリュームをマウントします。

暗号化されたボリュームの LUKS ヘッダーのパスフレーズスロットを解放するには、コマンド cryptsetup luksKillSlot を使用して、古いパスフレーズを削除します。

20.3. 大規模な組織での VOLUME_KEY の使用

大規模な組織では、すべてのシステム管理者が知っている単一のパスワードを使用し、システムごとに個別のパスワードを追跡することは非現実的であり、セキュリティー上のリスクがあります。これに対応するために、volume_key は非対称暗号を使用して、任意のコンピューターで暗号化されたデータへのアクセスに必要なパスワードを知っている人の数を最小限に抑えることができます。

このセクションでは、暗号化キーを保存する前の準備、暗号化キーの保存方法、ボリュームへのアクセスの復元、および緊急パスフレーズの設定に必要な手順について説明します。

20.3.1. 暗号化キーを保存するための準備

暗号化キーの保存を開始するには、いくつかの準備が必要です。

手順20.3 準備

1. X509 証明書/プライベートペアを作成します。

2. 信頼できるユーザーを指定します。このユーザーは、秘密鍵を危険にさらさないという点で信頼されています。このようなユーザーは、escrow パケットを復号化できます。

3. escrow パケットの復号に使用するシステムを選択します。このようなシステムでは、秘密鍵を含む NSS データベースを設定します。

   秘密鍵が NSS データベースで作成されていない場合は、以下の手順を行います。

   - 証明書と秘密鍵を PKCS#12 ファイルに保存します。

   - 以下を実行します。

     ```
     certutil -d /the/nss/directory -N
     ```

     この時点で、NSS データベースのパスワードを選択できます。各 NSS データベースには異なるパスワードを設定できるため、各ユーザーが個別の NSS データベースを使用する場合に、指定したユーザーは単一のパスワードを共有する必要はありません。

   - 以下を実行します。

     ```
     pk12util -d /the/nss/directory -i the-pkcs12-file
     ```

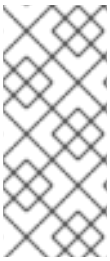
4. システムをインストールしている人、または既存システムに鍵を保存している人に証明書

を配布します。

5. 保存した秘密鍵については、マシンおよびボリュームから検索できるように、ストレージを準備します。たとえば、マシンごとに 1 つのサブディレクトリーを持つ単純なディレクトリーや、その他のシステム管理タスクに使用されるデータベースなどです。

20.3.2. 暗号化キーの保存

必要な準備 (「暗号化キーを保存するための準備」 を参照) の完了後に、以下の手順で暗号鍵を保存できるようになりました。



注記

このファイルのすべての例で、/path/to/volume は LUKS デバイスであり、含まれるプレーンテキストデバイスではありません。blkid -s type /path/to/volume は、タイプ="crypto_LUKS" を報告する必要があります。

手順20.4 暗号化キーの保存

1. 以下を実行します。

```
volume_key --save /path/to/volume -c /path/to/cert escrow-packet
```

2. 準備したストレージに、生成された escrow-packet ファイルを保存し、システムおよびボリュームと関連付けます。

この手順は、手動で実行するか、システムのインストールの一部としてスクリプトを使用して実行できます。

20.3.3. ボリュームへのアクセスの復元

暗号鍵を保存 (「暗号化キーを保存するための準備」および「暗号化キーの保存」を参照) したら、必要に応じてドライバーへのアクセスを復元できます。

手順20.5 ボリュームへのアクセスの復元

1. パケットストレージからボリュームの escrow パケットを取得し、指定したユーザーのいずれかに送信して復号化します。

2. *指定されたユーザーは、以下を実行します。*

```
volume_key --reencrypt -d /the/nss/directory escrow-packet-in -o escrow-packet-out
```

NSS データベースのパスワードを提供した後、指定されたユーザーは escrow-packet-out を暗号化するパスフレーズを選択します。このパスフレーズは毎回異なるものにでき、指定されたユーザーからターゲットシステムに暗号化キーが移動されている間のみ暗号化キーを保護します。

3. *指定されたユーザーから escrow-packet-out ファイルとパスフレーズを取得します。*

4. *レスキューモードなどで volume_key を実行し、escrow-packet-out ファイルを使用できる環境でターゲットシステムを起動します。*

5. *以下を実行します。*

```
volume_key --restore /path/to/volume escrow-packet-out
```

指定ユーザーが選択したパケットパスフレーズとボリュームの新しいパスフレーズを求めるプロンプトが表示されます。

6. *選択したボリュームパスフレーズを使用してボリュームをマウントします。*

たとえば、cryptsetup luksKillSlot を使用して忘れた古いパスフレーズを削除し、暗号化されたボリュームの LUKS ヘッダーのパスフレーズスロットを解放することができます。これは、コマンド cryptsetup luksKillSlot device key-slot コマンドで行います。詳細と例については、cryptsetup --help を参照してください。

20.3.4. 緊急パスフレーズの設定

状況によっては (出張など)、システム管理者が影響を受けるシステムを直接作業するのは実用的ではありませんが、ユーザーは引き続きデータにアクセスする必要があります。この場合、volume_key はパスフレーズと暗号鍵と連携できます。

システムのインストール中に、次のコマンドを実行します。

```
volume_key --save /path/to/volume -c /path/to/ert --create-random-passphrase passphrase-packet
```

ランダムなパスフレーズを生成し、指定したボリュームに追加して、passphrase-packet に格納します。--create-random-passphrase オプションと -o オプションを組み合わせて両方のパケットを同時に生成することもできます。

ユーザーがパスワードを忘れた場合、指定されたユーザーは以下を実行します。

```
volume_key --secrets -d /your/nss/directory passphrase-packet
```

これは、ランダムなパスフレーズを示しています。このパスフレーズをエンドユーザーに渡します。

20.4. VOLUME_KEY リファレンス

volume_key の詳細は、以下を参照してください。

- /usr/share/doc/volume_key-*/README にある readme ファイルで
- man volume_key を使用した volume_key の man ページ
- http://fedoraproject.org/wiki/Disk_encryption_key_escrow_use_cases

第21章 ソリッドステートディスクのデプロイメントガイドライン

ソリッドステートディスク (SSD) は、NAND フラッシュチップを使用してデータを永続的に保存するストレージデバイスです。これは、回転する磁気プラッターにデータを保存する従来のディスクと異なります。SSD では、論理ブロックアドレス (LBA) の全範囲にわたるデータへのアクセス時間は一定ですが、回転メディアを使用する古いディスクでは、広いアドレス範囲にまたがるアクセスパターンにシークコストがかかります。そのため、SSD デバイスのレイテンシーとスループットは改善されています。

使用中のブロック数がディスクの容量に近づくと、パフォーマンスが低下します。パフォーマンスへの影響の度合いはベンダーにより大きく異なります。ただし、すべてのデバイスである程度の低下が発生します。

劣化の問題に対処するために、ホストシステム (Linux カーネルなど) は破棄要求を使用して、指定した範囲のブロックが使用されなくなったことをストレージに通知する場合があります。SSD はこの情報を使用して、ウェアレベリング用の空きブロックを使用して、内部のスペースを解放できます。破棄は、ストレージがストレージプロトコル (ATA または SCSI) の観点からサポートをアドバタイズする場合にのみ発行されます。破棄要求は、ストレージプロトコルに固有のネゴシエートされた破棄コマンド (ATA の場合はTRIM コマンド、UNMAP が設定された WRITE SAME、SCSI の場合は UNMAP コマンド)を使用してストレージに発行されます。

discard サポートを有効にすることは、次の点が当てはまる場合に最も役立ちます。

- ファイルシステムにはまだ空き領域がある。
- 基盤となるストレージデバイスの論理ブロックのほとんどは、すでに書き込まれている。

TRIM の詳細は、Data Set Management T13 Specifications を参照してください。

UNMAP の詳細は、SCSI Block Commands 3 T10 Specification の 4.7.3.4 セクションを参照してください。

注記

市場内のすべてのソリッドステートデバイスに 破棄 サポートがあるわけではありません。ソリッドステートデバイスに discard サポートがあるかどうかを判断するには、デバイスの内部アロケーションユニットのサイズである /sys/block/sda/queue/discard_granularity を確認します。

デプロイメントに関する考慮事項

SSD の内部レイアウトと動作の関係から、内部 消去ブロック境界 でデバイスを分割することが最適になります。SSD がトポロジー情報をエクスポートする場合、Red Hat Enterprise Linux 7 のパーティションユーティリティーはデフォルトを選択します。ただし、デバイスがトポロジー情報をエクスポート しない 場合、Red Hat は、最初のパーティションを 1MB の境界に作成することを推奨します。

SSD には、ベンダーの選択に応じてさまざまなタイプの TRIM メカニズムがあります。ディスクの初期バージョンは、read コマンドの後に起こりうるデータ漏洩を妥協することで、パフォーマンスを向上させていました。

TRIM メカニズムの種類は次のとおりです。

- *非決定論的 TRIM*
- *確定的 TRIM (DRAT)*
- *TRIM 後の確定的読み取りゼロ(RZAT)*

最初の 2 つのタイプの TRIM メカニズムは、TRIM が異なるデータまたは同じデータを返した後、LBA への 読み取り コマンドとしてデータ漏えいを引き起こす可能性があります。RZAT は read コマンドの後にゼロを返し、Red Hat ではデータ漏えいを避けるためにこの TRIM メカニズムを推奨しています。これは SSD でのみ影響を受けます。RZAT メカニズムに対応するディスクを選択します。

使用される TRIM メカニズムのタイプは、ハードウェアの実装によって異なります。ATA で TRIM メカニズムのタイプを見つけるには、hdparm コマンドを使用します。TRIM メカニズムのタイプを確認するには、以下の例を参照してください。

```
# hdparm -I /dev/sda | grep TRIM
Data Set Management TRIM supported (limit 8 block)
Deterministic read data after TRIM
```

詳細は、man hdparm を参照してください。

論理ボリュームマネージャー (LVM)、デバイスマッパー (DM) ターゲット、および LVM が使用する MD (ソフトウェア RAID) ターゲットは、破棄をサポートします。破棄に対応しない DM ターゲット

は、dm-snapshot、dm-crypt、および dm-raid45 だけです。dm-mirror の破棄のサポートは、Red Hat Enterprise Linux 6.1 で追加され、7.0 以降は MD は破棄をサポートします。

SSD で RAID レベル 5 を使用すると、SSD がdiscard を正しく処理しない場合、パフォーマンスが低下します。raid456.conf ファイルまたは GRUB2 設定で破棄を設定できます。手順は、以下の手順を参照してください。

手順21.1 raid456.conf で破棄を設定する

devices_handle_discard_safely モジュールパラメーターは raid456 モジュールに設定されます。raid456.conf ファイルで破棄を有効にするには、次のコマンドを実行します。

1. *ハードウェアが破棄に対応していることを確認します。*

```
# cat /sys/block/disk-name/queue/discard_zeroes_data
```

戻り値が 1 の場合は、破棄がサポートされます。コマンドが 0 を返すと、RAID コードはディスクをゼロにする必要があり、これには時間がかかります。

2. */etc/modprobe.d/raid456.conf ファイルを作成し、以下の行を追加します。*

```
options raid456 devices_handle_discard_safely=Y
```

3. *dracut -f コマンドを使用して、初期 ramdisk (initrd)を再構築します。*

4. *システムを再起動して、変更を有効にします。*

手順21.2 GRUB2 設定での破棄の設定

devices_handle_discard_safely モジュールパラメーターは raid456 モジュールに設定されます。GRUB2 設定で破棄を有効にするには、次のコマンドを実行します。

1. *ハードウェアが破棄に対応していることを確認します。*

```
# cat /sys/block/disk-name/queue/discard_zeroes_data
```

戻り値が 1 の場合は、破棄がサポートされます。コマンドが 0 を返すと、RAID コードはディスクをゼロにする必要があり、これには時間がかかります。

2. /etc/default/grub ファイルに以下の行を追加します。

```
raid456.devices_handle_discard_safely=Y
```

3. GRUB2 設定ファイルの場所は、BIOS ファームウェアがインストールされているシステムと、UEFI がインストールされているシステムで異なります。次のいずれかのコマンドを使用して、GRUB2 設定ファイルを再作成します。

- BIOS ファームウェアを使用している場合は、以下のコマンドを使用します。

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
```

- UEFI ファームウェアを使用している場合は、以下のコマンドを使用します。

```
# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

4. システムを再起動して、変更を有効にします。

注記

Red Hat Enterprise Linux 7 では、破棄は ext4 ファイルシステムおよび XFS ファイルシステムでのみサポートされています。

Red Hat Enterprise Linux 6.3 以前では、ext4 ファイルシステムのみが破棄を完全にサポートしています。Red Hat Enterprise Linux 6.4 以降、ext4 と XFS の両方のファイルシステムが破棄を完全にサポートしています。デバイスで破棄コマンドを有効にするには、mount コマンドの discard オプションを使用します。たとえば、破棄を有効にして /dev/sda2 を /mnt にマウントするには、次のコマンドを実行します。

```
# mount -t ext4 -o discard /dev/sda2 /mnt
```

デフォルトでは、ext4 は discard コマンドを発行しません。主に、破棄を適切に実装しない可能性のあるデバイスでの問題を回避します。Linux swap コードは、有効なデバイスを 破棄 するために

discard コマンドを発行します。この動作を制御するオプションはありません。

パフォーマンスチューニングの考慮事項

ソリッドステートディスクに関するパフォーマンスチューニングの考慮事項は、『Red Hat Enterprise Linux 7 Performance Tuning Guide』 の Solid-State Disks セクションを参照してください。

第22章 書き込みバリア

書き込みバリアは、揮発性の書き込みキャッシュを備えたストレージデバイスの電源が切れた場合でも、ファイルシステムのメタデータが永続ストレージに正しく書き込まれ、順序付けられるようにするために使用されるカーネルメカニズムです。書き込みバリアが有効になっているファイルシステムにより、fsync （） を介して送信されるデータは、電源の損失後も持続します。

書き込みバリアを有効にすると、アプリケーションによってはパフォーマンスが大幅に低下する場合があります。具体的には、fsync （） を頻繁に使用するか、多くの小さなファイルを作成および削除するアプリケーションは、はるかに遅くなります。

22.1. 書き込みバリアの重要性

ファイルシステムはメタデータを安全に更新し、整合性を確保します。ジャーナル化されたファイルシステムは、メタデータの更新をトランザクションにバンドルし、次の方法で永続ストレージに送信します。

1. ファイルシステムは、トランザクションの本文をストレージデバイスに送信します。

2. ファイルシステムは、コミットブロックを送信します。

3. トランザクションとそれに対応するコミットブロックがディスクに書き込まれる場合、ファイルシステムは、トランザクションが電源障害に耐えられると想定します。

ただし、電源障害時のファイルシステムの整合性は、キャッシュが余分にあるストレージデバイスでは複雑になります。ローカルの S-ATA や SAS ドライブのようなストレージターゲットデバイスには、32MB から 64MB のサイズの書き込みキャッシュがある場合があります (最新のドライブを使用する場合)。ハードウェア RAID コントローラーには、多くの場合、内部書き込みキャッシュが含まれます。さらに、Net App、IBM、Hitachi、EMC などのハイエンドアレイにも大きなキャッシュがあります。

書き込みキャッシュがあるストレージデバイスは、データがキャッシュにあると I/O を完了と報告します。キャッシュが電力を失うと、そのデータも失われます。さらに悪いことに、キャッシュが永続ストレージにデステージされると、元のメタデータの順序が変更される可能性があります。これが発生すると、完全に関連付けられたトランザクションがない状態で、コミットブロックがディスクに存在する可能性があります。その結果、ジャーナルは、電力損失からの復旧中に、初期化されていないトランザクションブロックをファイルシステムに再生する場合があります。これにより、データの不整合や破損が発生する可能性があります。

書き込みバリアのしくみ

書き込みバリアは、I/O の前後でストレージの書き込みキャッシュフラッシュを介して Linux カーネルに実装されます。これは 順序が重要 となります。トランザクションが書き込まれた後、ストレージキャッシュがフラッシュされ、コミットブロックが書き込まれ、キャッシュが再度フラッシュされます。これにより、以下が確保されます。

- *ディスクに、すべてのデータが含まれている。*
- *順序の並べ替えは発生していません。*

バリアを有効にすると、fsync （） 呼び出しにより、ストレージキャッシュフラッシュも発行されます。これにより、fsync （） が返された直後に電源の損失が発生した場合でも、ファイルデータがディスク上の永続化されるようになります。

22.2. 書き込みバリアの有効化と無効化

電力損失時のデータ破損のリスクを軽減するために、一部のストレージデバイスはバッテリーバックアップ式の書き込みキャッシュを使用します。一般に、ハイエンドアレイと一部のハードウェアコントローラーは、バッテリーでバックアップされた書き込みキャッシュを使用します。ただし、キャッシュの揮発性はカーネルには表示されないため、Red Hat Enterprise Linux 7 では、対応しているすべてのジャーナリングファイルシステムで、デフォルトで書き込みバリアが有効になっています。

注記

書き込みキャッシュは、I/O パフォーマンスを向上させるように設計されています。ただし、書き込みバリアを有効にすると、キャッシュが常にフラッシュされるため、パフォーマンスが大幅に低下する可能性があります。

揮発性ではなく、バッテリーでバックアップされた書き込みキャッシュがあるデバイスと、書き込みキャッシュが無効になっているデバイスの場合、マウントに -o nobarrier オプションを使用して、マウント時に書き込みバリアを安全に無効にできます。ただし、一部のデバイスは書き込みバリアに対応していないため、このようなデバイスは /var/log/messages にエラーメッセージを記録します。詳細は、表22.1「ファイルシステムごとにバリアエラーメッセージを書き込む」 を参照してください。

表22.1 ファイルシステムごとにバリアエラーメッセージを書き込む

ファイルシステム	*エラーメッセージ*

ファイルシステム	エラーメッセージ
ext3/ext4	JBD: barrier-based sync failed on device - disabling barriers
XFS	Filesystem device - Disabling barriers, trial barrier write failed
btrfs	btrfs: disabling barriers on dev device

22.3. 書き込みバリアに関する考慮事項

一部のシステム設定では、データを保護するための書き込みバリアは必要ありません。多くの場合、書き込みバリアを有効にするとパフォーマンスが大幅に低下するため、書き込みバリアは他の方法が推奨されます。

書き込みキャッシュの無効化

別の方法としてデータ整合性の問題を回避する 1 つの方法は、書き込みキャッシュが電源障害時にデータを失わないようにすることです。可能な場合、これを設定する最良の方法は、書き込みキャッシュを無効にすることです。1 つ以上の SATA ドライブを備えたシンプルなサーバーまたはデスクトップ (ローカルの SATA コントローラー Intel AHCI パート外) で、次のコマンドを使用して、ターゲットの SATA ドライブの書き込みキャッシュを無効にできます。

```
# hdparm -W0 /device/
```

バッテリーでバックアップされる書き込みキャッシュ

システムがバッテリークアップ式書き込みキャッシュを備えたハードウェア RAID コントローラーを使用する場合は常に、書き込みバリアも不要です。システムにそのようなコントローラーが装備されていて、そのコンポーネントドライブで書き込みキャッシュが無効になっている場合、コントローラーはライトスルーキャッシュとして機能します。これは、書き込みキャッシュデータが電力損失に耐えることをカーネルに通知します。

ほとんどのコントローラーは、ベンダー固有のツールを使用して、ターゲットドライブのクエリーおよび操作を行います。たとえば、LSI Megaraid SAS コントローラーは、バッテリーでバックアップされた書き込みキャッシュを使用します。このタイプのコントローラーでは、ターゲットドライブを管理する MegaCli64 ツールが必要です。LSI Megaraid SAS 用のすべてのバックエンドドライブの状態を表示するには、次のコマンドを実行します。

```
# MegaCli64 -LDGetProp -DskCache -LAll -aALL
```

LSI Megaraid SAS 用のすべてのバックエンドドライブの書き込みキャッシュを無効にするには、次のコマンドを実行します。

```
# MegaCli64 -LDSetProp -DisDskCache -Lall -aALL
```

注記

ハードウェア RAID カードは、システムの稼働中にバッテリーを再充電します。システムの電源を長時間切ったままにしておくと、バッテリーの充電が失われ、停電時に保存されているデータは脆弱なままになります。

ハイエンドアレイ

ハイエンドアレイは、電源の障害発生時にデータを保護するさまざまな方法を提供します。このため、外部 RAID ストレージの内部ドライブの状態を確認する必要はありません。

NFS

データの整合性は NFS サーバー側で処理されるため、NFS クライアントは書き込みバリアを有効にする必要はありません。そのため、NFS サーバーは、(書き込みバリアまたはその他の手段のどちらを介しても) 電源損失全体にわたってデータの永続性を確保するために設定する必要があります。

第23章 ストレージ I/O アライメントとサイズ

最近の SCSI 規格および ATA 規格の強化により、ストレージデバイスには推奨される (場合によっては必須の) I/O アライメント および I/O サイズ を示すことができるようになりました。この情報は、物理セクターのサイズを 512 バイトから 4k バイトに増やす新しいディスクドライブで特に役立ちます。この情報は、チャンクサイズやストライプサイズがパフォーマンスに影響を与える RAID デバイスにも役立ちます。

Linux I/O スタックは、ベンダーが提供する I/O アライメントと I/O サイズ情報を処理するように強化され、ストレージ管理ツール(parted、lvm、mkfs.* など)がデータの配置とアクセスを最適化できるようになりました。レガシーデバイスが I/O のアラインメントとサイズのデータをエクスポートしていない場合、Red Hat Enterprise Linux 7 のストレージ管理ツールは、4k (またはそれ以上の 2 の累乗) 境界で I/O を控えめに調整します。これにより、必要/推奨される I/O アライメントやサイズが記載されていなくても、4k セクターデバイスが正しく動作するようになります。

オペレーティングシステムがデバイスから取得した情報を特定する方法は、「ユーザー空間アクセス」 を参照してください。このデータは、その後にストレージ管理ツールによってデータ配置を決定するために使用されます。

Red Hat Enterprise Linux 7 では IO スケジューラーが変更されました。SATA ドライブを除き、デフォルトの IO スケジューラーが期限になりました。CFQ は、SATA ドライブのデフォルトの IO スケジューラーです。ストレージを高速化するために、Deadline は CFQ を上回り、使用時には特別なチューニングなしでパフォーマンスが向上します。

一部のディスク (SAS ローテーションディスクなど) ではデフォルトが適切でない場合は、IO スケジューラーを CFQ に変更します。このインスタンスはワークロードによって異なります。

23.1. ストレージアクセスパラメーター

オペレーティングシステムは、次の情報を使用して I/O アライメントとサイズを判断します。

physical_block_size

デバイスが動作できる最小の内部ユニット

logical_block_size

デバイス上の場所指定に外部で使用される

alignment_offset

基礎となる物理的なアライメントのオフセットとなる Linux ブロックデバイス (パーティション/MD/LVM デバイス) の先頭部分のバイト数

minimum_io_size

ランダムな I/O に対して推奨のデバイス最小ユニット

optimal_io_size

I/O ストリーミングで推奨されるデバイスのユニット

たとえば、特定の 4K セクターデバイスは内部で 4K physical_block_size を使用しますが、より詳細な 512 バイト logical_block_size を Linux に公開する場合があります。この不一致により、I/O がずれる可能性があります。これに対処するために、Red Hat Enterprise Linux 7 I/O スタックは、ブロックデバイスの開始が基礎となる物理アライメントからオフセットされている場合に、alignment_offset を考慮に入れて、自然に整列された境界(physical_block_size)ですべてのデータ領域の起動を試みます。

ストレージベンダーは、デバイスのランダム I/O (minimum_io_size)およびストリーミング I/O (optimal_io_size)の推奨最小単位に関する I/O ヒント を提供することもできます。たとえば、minimum_io_size と optimal_io_size は、それぞれ RAID デバイスのチャンクサイズとストライプサイズに対応している可能性があります。

23.2. ユーザー空間アクセス

適切にアライメントおよびサイズ設定された I/O を使用するように常に注意してください。これは、ダイレクト I/O アクセスでは特に重要です。ダイレクト I/O は、logical_block_size 境界と、logical_block_size の倍数で整列する必要があります。

ネイティブ 4K デバイス(logical_block_size が 4K)の場合は、アプリケーションがデバイスの logical_block_size の倍数でダイレクト I/O を実行することが重要です。つまり、アプリケーションは、4k 境界で整列された I/O ではなく 512 バイトで整列された I/O を実行するネイティブの 4k デバイスでは失敗します。

これを回避するには、アプリケーションでデバイスの I/O パラメーターを調べて、適切な I/O アライメントとサイズが使用されていることを確認する必要があります。前述のように、I/O パラメーター

は、sysfs インターフェイスおよびブロックデバイス ioctl インターフェイスの両方で公開されています。

詳細は、man libblkid を参照してください。この man ページは、libblkid-devel パッケージで提供されます。

sysfs インターフェイス

- /sys/block/disk/alignment_offset

  または

  /sys/block/disk/partition/alignment_offset

  注記

  ファイルの場所は、ディスクが物理ディスク (ローカルディスク、ローカル RAID、またはマルチパス LUN) か、仮想ディスクかによって異なります。最初のファイルの場所は物理ディスクに適用され、2 番目のファイルの場所は仮想ディスクに適用されます。これは、virtio-blk がパーティションのアラインメント値を常に報告するためです。物理ディスクは、アライメント値を報告する場合と報告しない場合があります。

- /sys/block/disk/queue/physical_block_size
- /sys/block/disk/queue/logical_block_size
- /sys/block/disk/queue/minimum_io_size
- /sys/block/disk/queue/optimal_io_size

カーネルは、I/O パラメーター情報を提供しないレガシーデバイスに、この sysfs 属性をエクスポートします。以下に例を示します。

> *例23.1 sysfs インターフェイス*
>
> ```
> alignment_offset:    0
> physical_block_size: 512
> logical_block_size:  512
> minimum_io_size:     512
> optimal_io_size:     0
> ```

ブロックデバイス ioctls

- *BLKALIGNOFF: alignment_offset*
- *BLKPBSZGET:physical_block_size*
- *BLKSSZGET:logical_block_size*
- *BLKIOMIN:minimum_io_size*
- *BLKIOOPT:optimal_io_size*

23.3. I/O 規格

本セクションでは、ATA および SCSI デバイスで使用される I/O 規格について説明します。

ATA

ATA デバイスは、IDENTIFY DEVICE コマンドを使用して適切な情報を報告する必要があります。ATA デバイスは、physical_block_size、logical_block_size、および alignment_offset の I/O パラ

メーターのみを報告します。追加の I/O ヒントは、ATA コマンドセットの範囲外です。

SCSI

Red Hat Enterprise Linux 7 に対応する I/O パラメーターには、少なくとも SCSI プライマリーコマンド (SPC-3) のバージョン 3 が必要です。カーネルは、SPC-3 への準拠を要求するデバイスに、拡張照会 (BLOCK LIMITS VPD ページへのアクセスを取得)と READ CAPACITY (16) コマンドのみを送信します。

READ CAPACITY (16) コマンドは、ブロックサイズとアライメントオフセットを提供します。

- LOGICAL BLOCK LENGTH IN BYTES は、/sys/block/ディスク/queue/physical_block_sizeの導出に使用されます。
- LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT は、/sys/block/ディスク/queue/logical_block_sizeの導出に使用されます。
- LOWEST ALIGNED LOGICAL BLOCK ADDRESS は、 以下を取得するために使用されます。
  - /sys/block/disk/alignment_offset
  - /sys/block/disk/partition/alignment_offset

BLOCK LIMITS VPD ページ(0xb0)は、I/O ヒントを提供します。また、OPTIMAL TRANSFER LENGTH GRANULARITY および OPTIMAL TRANSFER LENGTH を使用して以下を取得します。

- /sys/block/disk/queue/minimum_io_size
- /sys/block/disk/queue/optimal_io_size

sg3_utils パッケージは、BLOCK LIMITS VPD ページにアクセスするために使用できる sg_inq ユーティリティーを提供します。そのためには、以下のコマンドを実行します。

-

```
# sg_inq -p 0xb0 disk
```

23.4. I/O パラメーターのスタッキング

Linux I/O スタックのすべてのレイヤーは、さまざまな I/O パラメーターをスタックに伝播するように設計されています。レイヤーが属性を使用するか、多くのデバイスを集約する場合、レイヤーは適切な I/O パラメーターを公開して、上位レイヤーのデバイスまたはツールが変換時にストレージを正確に表示できるようにする必要があります。実用例を以下に示します。

- I/O スタックの 1 つのレイヤーのみがゼロ以外の alignment_offset について調整する必要があります。レイヤーがそれに応じて調整されると、alignment_offset がゼロのデバイスがエクスポートされます。

- LVM で作成されたストライプ化デバイスマッパー(DM)デバイスは、ストライプ数（ディスク数）とユーザー提供のチャンクサイズを基準にして minimum_io_size と optimal_io_size をエクスポートする必要があります。

Red Hat Enterprise Linux 7 では、デバイスマッパーおよび Software Raid (MD) デバイスドライバーを使用して、異なる I/O パラメーターを持つデバイスを任意に組み合わせることができます。カーネルのブロック層は、個々のデバイスの I/O パラメーターを合理的に組み合わせようとします。カーネルは、異種デバイスの組み合わせを妨げません。ただし、そうすることに伴うリスクに注意してください。

たとえば、512 バイトのデバイスと 4K デバイスは、logical_block_size が 4K の単一の論理 DM デバイスに統合できます。このようなハイブリッドデバイスに階層化されたファイルシステムは、4K がアトミックに書き込まれることを想定していますが、実際には、512 バイトのデバイスに発行されると 8 つの論理ブロックアドレスにまたがることになります。高レベルの DM デバイスに 4K logical_block_size を使用すると、システムクラッシュが発生した場合に 512 バイトのデバイスへの部分的な書き込みが可能になります。

複数のデバイスの I/O パラメーターを組み合わせて競合が発生すると、ブロック層が、デバイスが部分的な書き込みの影響を受けやすく、かつ/またはアラインが正しくないという警告を発することがあります。

23.5. 論理ボリュームマネージャー

LVM は、カーネルの DM デバイスを管理するのに使用されるユーザー空間ツールを提供します。LVM は、（特定の DM デバイスが使用する）データ領域の開始をシフトして、LVM が管理するデバイスに関連付けられたゼロ以外の alignment_offset を考慮します。これは、論理ボリュームが適切に配置されることを意味します(alignment_offset=0)。

デフォルトでは、LVM は任意の alignment_offset に対して調整しますが、/etc/lvm/lvm.conf で data_alignment_offset_detection を 0 に設定すると、この動作を無効にできます。これを無効にすることは推奨されません。

LVM は、デバイスの I/O ヒントも検出します。デバイスのデータ領域の開始は、sysfs で公開される minimum_io_size または optimal_io_size の倍数になります。optimal_io_size が定義されていない場合（つまり 0）、LVM は minimum_io_size を使用します。

デフォルトでは、LVM はこれらの I/O ヒントを自動的に決定しますが、/etc/lvm/lvm.conf で data_alignment_detection を 0 に設定すると、この動作を無効にできます。これを無効にすることは推奨されません。

23.6. パーティションおよびファイルシステムツール

本セクションでは、さまざまなパーティションツールおよびファイルシステム管理ツールがデバイスの I/O パラメーターと相互作用する方法を説明します。

util-linux-ng の libblkid と fdisk

util-linux-ng パッケージで提供される libblkid ライブラリーには、デバイスの I/O パラメーターにアクセスするためのプログラム API が含まれています。libblkid を使用すると、アプリケーション、特に Direct I/O を使用するアプリケーションが、I/O 要求のサイズを適切に調整できます。util-linux-ng の fdisk ユーティリティーは、libblkid を使用して、すべてのパーティションの配置を最適化するためにデバイスの I/O パラメーターを決定します。fdisk ユーティリティーは、すべてのパーティションを 1MB の境界に合わせます。

parted と libparted

parted の libparted ライブラリーは、libblkid の I/O パラメーター API も使用します。Red Hat Enterprise Linux 7 インストーラーである Anaconda は libparted を使用します。つまり、インストーラーまたは parted のいずれかで作成されたすべてのパーティションが適切に調整されます。I/O パラメーターを提供していないように見えるデバイスで作成されたすべてのパーティションの場合、デフォルトの調整は 1MB になります。

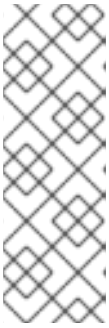
parted が使用するヒューリスティックは次のとおりです。

- 報告された alignment_offset は、常に最初のプライマリーパーティションの開始オフセットとして使用します。
- optimal_io_size が定義されている場合(0ではない)は、すべてのパーティションを

optimal_io_size 境界に合わせます。

- *optimal_io_size が定義されていない場合(0など)、alignment_offset は 0 で、minimum_io_size は 2 の累乗で、1MB のデフォルトアライメントを使用します。*

これは、I/O ヒントを提供していないように見えるレガシーデバイスのキャッチオールです。このため、デフォルトでは、すべてのパーティションは 1MB の境界にアライメントされます。



注記

Red Hat Enterprise Linux 7 は、I/O ヒントを提供しないデバイスと、alignment_offset=0 および optimal_io_ size=0 でこれを行うデバイスを区別できません。このようなデバイスは、SAS 4K デバイスが 1 つだけの場合があります。そのため、最悪の場合はディスクの起動時に 1MB の領域が失われます。

ファイルシステムツール

異なる mkfs.filesystem ユーティリティーも、デバイスの I/O パラメーターを使用するように強化されています。これらのユーティリティーでは、ファイルシステムをフォーマットして、基礎となるストレージデバイスの logical_block_size よりも小さいブロックサイズを使用することができません。

mkfs.gfs2 を除き、その他すべての mkfs.filesystem ユーティリティーは I/O ヒントを使用して、ディスク上のデータ構造と、基盤となるストレージデバイスの minimum_io_size および optimal_io_size に関連するデータ領域をレイアウトします。これにより、ファイルシステムを、さまざまな RAID (ストライプ化) レイアウトに合わせて最適にフォーマットできます。

第24章 リモートディスクレスシステムの設定

PXE で起動する基本的なリモートディスクレスシステムを設定するには、以下のパッケージが必要です。

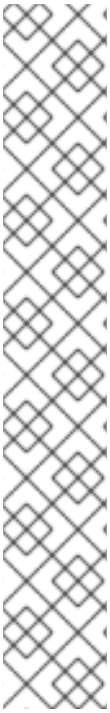
- tftp-server
- xinetd
- dhcp
- syslinux
- dracut-network

注記

dracut-network パッケージをインストールしたら、以下の行を /etc/dracut.conf に追加します。

```
add_dracutmodules+="nfs"
```

リモートディスクレスシステムの起動には、tftp サービス(tftp-serverにより提供)と DHCP サービス(dhcpにより提供)の両方が必要です。tftp サービスは、PXE ローダーを介してネットワーク経由でカーネルイメージと initrd を取得するために使用されます。



注記

SELinux は NFSv4.2 でのみサポートされます。SELinux を使用するには、次の行を追加して、/etc/sysconfig/nfs で NFS を明示的に有効にする必要があります。

RPCNFSDARGS="-V 4.2"

次に、/var/lib/tftpboot/pxelinux.cfg/default で、root=nfs:server-ip:/exported/root/directory を root=nfs:server-ip:/exported/root/directory,vers=4.2 に変更します。

最後に、NFS サーバーを再起動します。

次のセクションでは、ネットワーク環境にリモートディスクレスシステムをデプロイするのに必要な手順の概要を説明します。

重要

一部の RPM パッケージは、ファイル機能(setcap や getcapなど)を使用して開始しました。ただし、現在 NFS ではこれらの機能に対応していないため、ファイル機能を使用するパッケージのインストールまたは更新に失敗します。

24.1. ディスクレスクライアントの TFTP サービスの設定

前提条件

- *必要なパッケージをインストールしている。24章リモートディスクレスシステムの設定 を参照してください。*

手順

tftp を設定するには、以下の手順を実行します。

手順24.1 tftpを設定するには、以下を実行します。

1. *ネットワーク上での PXE ブートを有効にします。*

```
# systemctl enable --now tftp
```

2. tftp root ディレクトリー(chroot)は、/var/lib/tftpboot にあります。/usr/share/syslinux/pxelinux.0 を /var/lib/tftpboot/ にコピーします。

```
# cp /usr/share/syslinux/pxelinux.0 /var/lib/tftpboot/
```

3. tftp の root ディレクトリーに pxelinux.cfg ディレクトリーを作成します。

```
# mkdir -p /var/lib/tftpboot/pxelinux.cfg/
```

4. tftp トラフィックを許可するようにファイアウォールルールを設定します。

tftp は TCP ラッパーに対応しているため、/etc/hosts.allow 設定ファイルで tftp へのホストアクセスを設定できます。TCP ラッパーおよび /etc/hosts.allow 設定ファイルの設定に関する詳細は、Red Hat Enterprise Linux 7 セキュリティーガイド を参照してください。hosts_access(5) は、/etc/hosts.allow に関する情報も提供します。

次のステップ

ディスクレスクライアントの tftp を設定したら、DHCP、NFS、エクスポートしたファイルシステムを適宜設定します。DHCP、NFS、およびエクスポートしたファイルシステムの設定方法は、「ディスクレスクライアント用の DHCP の設定」 および 「ディスクレスクライアントのエクスポートしたファイルシステムの設定」 を参照してください。

24.2. ディスクレスクライアント用の DHCP の設定

前提条件

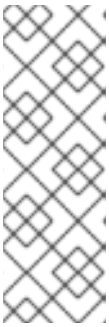
- 必要なパッケージをインストールしている。24章リモートディスクレスシステムの設定 を参照してください。
- tftp サービスを設定します。「ディスクレスクライアントの tftp サービスの設定」 を参照してください。

手順

1. tftp サーバーの設定後に、同じホストマシンに DHCP サービスを設定する必要があります。DHCP サーバーの設定手順は、Configuring a DHCP Server を参照してください。
2. /etc/dhcp/dhcpd.conf に以下の設定を追加して、DHCP サーバーで PXE ブートを有効にします。

```
allow booting;
allow bootp;
class "pxeclients" {
  match if substring(option vendor-class-identifier, 0, 9) = "PXEClient";
  next-server server-ip;
  filename "pxelinux.0";
}
```

- server-ip を、tftp サービスと DHCP サービスが置かれているホストマシンの IP アドレスに置き換えます。



注記

libvirt 仮想マシンをディスクレスクライアントとして使用する場合、libvirt は DHCP サービスを提供し、スタンドアロン DHCP サーバーは使用されません。このような場合、libvirt ネットワーク設定の bootp file='filename' オプションを使用して、ネットワークブートを有効にする必要があります(virsh net-edit)。

次のステップ

tftp と DHCP が設定されたので、NFS とエクスポートされたファイルシステムを設定します。手順は、「ディスクレスクライアントのエクスポートしたファイルシステムの設定」 を参照してください。

24.3. ディスクレスクライアントのエクスポートしたファイルシステムの設定

前提条件

- 必要なパッケージをインストールしている。24章リモートディスクレスシステムの設定 を参照してください。
- tftp サービスを設定します。「ディスクレスクライアントの tftp サービスの設定」 を参照してください。
- DHCP を設定している。「ディスクレスクライアント用の DHCP の設定」 を参照してください。

手順

1. エクスポートしたファイルシステムのルートディレクトリー (ネットワーク上のディスクレスクライアントが使用) は、NFS 経由で共有されます。root ディレクトリーを /etc/exports に

追加して、root ディレクトリーをエクスポートするように NFS サービスを設定します。手順は、「/etc/exports 設定ファイル」を参照してください。

2. ディスクレスクライアントに完全に対応できるようにするには、root ディレクトリーには Red Hat Enterprise Linux の完全なインストールが含まれている必要があります。既存のインストールのクローンを作成するか、新しいベースシステムをインストールできます。

- 実行中のシステムと同期するには、rsync ユーティリティーを使用します。

```
# rsync -a -e ssh --exclude='/proc/*' --exclude='/sys/*' \
    hostname.com:/exported-root-directory
```

  - hostname.com を、rsync を介して同期する実行中のシステムのホスト名に置き換えます。
  - exported-root-directory を、エクスポートしたファイルシステムへのパスに置き換えます。

- Red Hat Enterprise Linux をエクスポートした場所にインストールするには、--installroot オプションを指定して yum ユーティリティーを使用します。

```
# yum install @Base kernel dracut-network nfs-utils \
    --installroot=exported-root-directory --releasever=/
```

エクスポートするファイルシステムは、ディスクレスクライアントが使用できるようにする前に追加で設定する必要があります。空き領域が足りない場合は、次の手順を実行します。

手順24.2 ファイルシステムの設定

1. ディスクレスクライアントが使用するカーネル(vmlinuz-kernel-version)を選択し、tftp ブートディレクトリーにコピーします。

```
# cp /boot/vmlinuz-kernel-version /var/lib/tftpboot/
```

2. NFS サポートで initrd (initramfs-kernel-version.img)を作成します。

```
# dracut --add nfs initramfs-kernel-version.img kernel-version
```

3.
次のコマンドを使用して、initrd のファイル権限を 644 に変更します。

```
# chmod 644 initramfs-kernel-version.img
```

警告

initrd のファイルパーミッションが変更されないと、pxelinux.0 ブートローダーが file not found エラーで失敗します。

4.
作成された initramfs-kernel-version.img を tftp ブートディレクトリーにもコピーします。

5.
/var/lib/tftpboot/ ディレクトリーで initrd およびカーネルを使用するようにデフォルトのブート設定を編集します。この設定は、エクスポートしたファイルシステム (/exported/root/directory)を読み書きとしてマウントするようにディスクレスクライアントの root に指示する必要があります。/var/lib/tftpboot/pxelinux.cfg/default ファイルに以下の設定を追加します。

```
default rhel7

label rhel7
  kernel vmlinuz-kernel-version
  append initrd=initramfs-kernel-version.img root=nfs:server-ip:/exported/root/directory rw
```

server-ip を、tftp サービスと DHCP サービスが置かれているホストマシンの IP アドレスに置き換えます。

これで、NFS 共有がディスクレスクライアントにエクスポートできるようになりました。これらのクライアントは、PXE 経由でネットワーク経由で起動できます。

第25章 オンラインストレージ管理

多くの場合、オペレーティングシステムの実行中に、再起動せずにストレージデバイスを追加、削除、またはサイズ変更することが望ましいです。本章では、システムの実行中に、Red Hat Enterprise Linux 7 ホストシステムでストレージデバイスを再設定するために使用できる手順を概説します。これは、iSCSI およびファイバーチャネルストレージの相互接続を対象としています。その他の相互接続タイプは、今後追加される可能性があります。

本章では、ストレージデバイスの追加、削除、変更、および監視を中心に説明します。ファイバーチャネルまたは iSCSI プロトコルの詳細は説明されていません。これらのプロトコルに関する詳細は、他のドキュメントを参照してください。

本章では、さまざまな sysfs オブジェクトを参照します。Red Hat は、sysfs オブジェクト名とディレクトリー構造が、Red Hat Enterprise Linux のメジャーリリースで変更される可能性があることを推奨しています。これは、アップストリームの Linux カーネルでは安定した内部 API が提供されていないためです。転送可能な方法で sysfs オブジェクトを参照する方法は、カーネルソースツリーのドキュメント /usr/share/doc/kernel-doc-version/Documentation/sysfs-rules.txt を参照してください。

警告

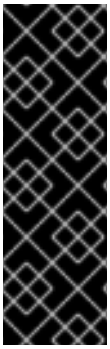
オンラインストレージの再設定は慎重に行う必要があります。プロセス中にシステムに障害が発生したり、中断したりすると、予期しない結果になることがあります。Red Hat では、変更操作時にシステムへの負荷を可能な限り低減することを推奨しています。これにより、設定変更中に I/O エラー、メモリー不足エラー、または同様のエラーが発生する可能性が低くなります。以下のセクションでは、これに関するより具体的なガイドラインを説明します。

また、Red Hat では、オンラインストレージを再設定する前にすべてのデータのバックアップを作成することを推奨しています。

25.1. ターゲットの設定

Red Hat Enterprise Linux 7 は、カーネルターゲットの設定ファイルを直接操作することなく、Linux-IO ターゲットの設定を表示、編集、および保存するためのフロントエンドとして targetcli シェルを使用します。targetcli ツールは、管理者がローカルストレージリソースをエクスポートできるコマンドラインインターフェイスです。ローカルストレージリソースは、ファイル、ボリューム、ローカル SCSI デバイス、または RAM ディスクのいずれかによってバッキングされます。targetcli ツールにはツリーベースのレイアウトがあり、組み込みのタブ補完が含まれており、完全なオートコンプリートサポートとインラインドキュメントが提供されます。

targetcli は可能な限り簡素化されているため、targetcli の階層は、常にカーネルインターフェイスと完全に一致するわけではありません。



重要

targetcli で加えられた変更を永続化するには、ターゲットサービスを起動して有効にします。

```
# systemctl start target
# systemctl enable target
```

25.1.1. targetcli のインストールおよび実行

targetcli をインストールするには、以下を使用します。

```
# yum install targetcli
```

ターゲット サービスを起動します。

```
# systemctl start target
```

システムの起動時に ターゲット が開始するように設定します。

```
# systemctl enable target
```

ファイアウォールでポート 3260 を開き、ファイアウォール設定を再読み込みします。

```
# firewall-cmd --permanent --add-port=3260/tcp
Success
# firewall-cmd --reload
Success
```

targetcli コマンドを使用してから、ツリーインターフェイスのレイアウトに ls コマンドを使用します。

```
# targetcli
:
/> ls
o- /.....[...]
```

```
o- backstores.....[...]
| o- block.....[Storage Objects: 0]
| o- fileio.....[Storage Objects: 0]
| o- pscsi.....[Storage Objects: 0]
| o- ramdisk.....[Storage Ojbects: 0]
o- iscsi.....[Targets: 0]
o- loopback.....[Targets: 0]
```

注記

Red Hat Enterprise Linux 7.0 では、Bash から targetcli コマンド(targetcli iscsi/ create など)を使用しても機能せず、エラーが返されません。Red Hat Enterprise Linux 7.1 より、シェルスクリプトで targetcli を使用することがより有用なエラーステータスコードが提供されています。

25.1.2. バックストアの作成

バックストアを使用すると、エクスポートされた LUN のデータをローカルマシンに保存するさまざまな方法をサポートできます。ストレージオブジェクトを作成すると、バックストアが使用するリソースが定義されます。

注記

Red Hat Enterprise Linux 6 では、バッキングストアという用語は、作成されたマッピングを指すために使用されます。ただし、バックストアのさまざまな使用方法の混同を避けるため、Red Hat Enterprise Linux 7 ではストレージオブジェクトは作成されたマッピングを参照し、バックストアはさまざまなタイプのバッキングデバイスを説明するために使用されます。

LIO がサポートするバックストアデバイスは次のとおりです。

FILEIO (Linux ファイルバッキングストレージ)

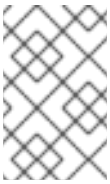
FILEIO ストレージオブジェクトは、write_back 操作または write_thru 操作のいずれかをサポートできます。write_back は、ローカルファイルシステムキャッシュを有効にします。これにより、パフォーマンスが向上しますが、データの損失のリスクが高まります。write_thru を優先して write_back を無効にするには、write_back=false を使用することが推奨されます。

fileio ストレージオブジェクトを作成するには、/backstores/fileio create file_name file_location file_size write_back=false コマンドを実行します。以下に例を示します。

```
/> /backstores/fileio create file1 /tmp/disk1.img 200M write_back=false
Created fileio file1 with size 209715200
```

BLOCK (Linux BLOCK デバイス)

ブロックドライバーを使用すると、/sys/block に表示されるブロックデバイスを LIO で使用できます。これには物理デバイス (HDD、SSD、CD、DVD など) および論理デバイス (ソフトウェアまたはハードウェアの RAID ボリューム、LVM ボリュームなど) が含まれます。



注記

BLOCK バックストアは通常、最高のパフォーマンスを提供します。

任意のブロックデバイスを使用して BLOCK バックストアを作成する場合は、以下のコマンドを使用します。

```
# fdisk /dev/vdb
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table
Building a new DOS disklabel with disk identifier 0x39dc48fb.

Command (m for help): n
Partition type:
  p   primary (0 primary, 0 extended, 4 free)
  e   extended
Select (default p): *Enter*
Using default response p
Partition number (1-4, default 1): *Enter*
First sector (2048-2097151, default 2048): *Enter*
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-2097151, default 2097151): +250M
Partition 1 of type Linux and of size 250 MiB is set

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

```
/> /backstores/block create name=block_backend dev=/dev/vdb
Generating a wwn serial.
Created block storage object block_backend using /dev/vdb.
```

注記

論理ボリューム上に BLOCK バックストアを作成することもできます。

PSCSI (Linux パススルー SCSI デバイス)

SCSI エミュレーションなしで SCSI コマンドの直接パススルーをサポートするストレージオブジェクト、および /proc/scsi/scsi (SAS ハードドライブなど)の lsscsi とともに表示される基礎となる SCSI デバイスを持つストレージオブジェクトは、バックストアとして設定できます。このサブシステムでは、SCSI-3 以降に対応しています。

警告

PSCSI は、上級ユーザーのみが使用してください。非対称論理ユニット割り当て (ALUA) や永続予約 (VMware ESX や vSphere で使用される永続予約など) は、通常はデバイスのファームウェアに実装されず、誤作動やクラッシュが発生する原因となることがあります。疑わしい場合は、代わりに BLOCK を使用して実稼働環境のセットアップを行います。

物理 SCSI デバイス（この例では /dev/sr0 を使用する TYPE_ROM デバイス）用の PSCSI バックストアを作成するには、次のコマンドを実行します。

```
/> backstores/pscsi/ create name=pscsi_backend dev=/dev/sr0
Generating a wwn serial.
Created pscsi storage object pscsi_backend using /dev/sr0
```

メモリーコピー RAM ディスク (Linux RAMDISK_MCP)

メモリーコピー RAM ディスク(ramdisk)は、完全な SCSI エミュレーションと、イニシエーターのメモリーコピーを使用した個別のメモリーマッピングを持つ RAM ディスクを提供します。これにより、マルチセッションの機能を利用できます。これは、特に実稼働環境での高速で不揮発性の大容量ストレージで有用です。

1GB の RAM ディスクバックストアを作成するには、次のコマンドを使用します。

```
/> backstores/ramdisk/ create name=rd_backend size=1GB
Generating a wwn serial.
Created rd_mcp ramdisk rd_backend with size 1GB.
```

25.1.3. iSCSI ターゲットの作成

iSCSI ターゲットを作成するには、次のコマンドを実行します。

手順25.1 iSCSI ターゲットの作成

1. targetcli を実行します。

2. iSCSI 設定パスに移動します。

```
/> iscsi/
```

注記

cd コマンドは、ディレクトリーの変更も許可され、移動先のパスを一覧表示するだけです。

3. デフォルトのターゲット名を使用して iSCSI ターゲットを作成します。

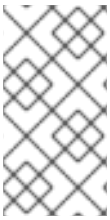
```
/iscsi> create
Created target
iqn.2003-01.org.linux-iscsi.hostname.x8664:sn.78b473f296ff
Created TPG1
```

または、指定した名前を使用して iSCSI ターゲットを作成します。

```
/iscsi > create iqn.2006-04.com.example:444
Created target iqn.2006-04.com.example:444
Created TPG1
```

4. ls を使用してターゲットが一覧表示されるときに、新しく作成されたターゲットが表示されていることを確認します。

```
/iscsi > ls
o- iscsi.....[1 Target]
  o- iqn.2006-04.com.example:444.....[1 TPG]
    o- tpg1.....[enabled, auth]
      o- acls.....[0 ACL]
      o- luns.....[0 LUN]
      o- portals.....[0 Portal]
```



注記

Red Hat Enterprise Linux 7.1 以降では、ターゲットを作成するたびに、デフォルトのポータルも作成されます。

25.1.4. iSCSI ポータルの設定

iSCSI ポータルを設定するには、iSCSI ターゲットを作成し、それを TPG に関連付ける必要があります。手順は、「iSCSI ターゲットの作成」 を参照してください。

注記

Red Hat Enterprise Linux 7.1 以降では、iSCSI ターゲットを作成すると、デフォルトのポータルも作成されます。このポータルは、デフォルトのポート番号 (0.0.0.0:3260) を持つすべての IP アドレスをリッスンするように設定されています。これを削除し、指定されたポータルのみを追加するには、/iscsi/iqn-name/tpg1/portals delete ip_address=0.0.0.0 ip_port=3260 を使用し、必要な情報で新しいポータルを作成します。

手順25.2 iSCSI ポータルの作成

1. TPG に移動します。

```
/iscsi> iqn.2006-04.example:444/tpg1/
```

2. ポータルを作成する方法は、デフォルトのポータルを作成する方法と、リッスンする IP アドレスを指定するポータルを作成する方法の 2 つがあります。

デフォルトのポータルを作成すると、デフォルトの iSCSI ポート 3260 が使用され、ターゲットがそのポートのすべての IP アドレスをリッスンできるようになります。

```
/iscsi/iqn.20...mple:444/tpg1> portals/ create
Using default IP port 3260
Binding to INADDR_Any (0.0.0.0)
```

```
Created network portal 0.0.0.0:3260
```

リッスンする IP アドレスを指定するポータルを作成するには、次のコマンドを使用します。

```
/iscsi/iqn.20...mple:444/tpg1> portals/ create 192.168.122.137
Using default IP port 3260
Created network portal 192.168.122.137:3260
```

3. 新しく作成したポータルが ls コマンドで表示されていることを確認します。

```
/iscsi/iqn.20...mple:444/tpg1> ls
o- tpg..................................... [enambled, auth]
  o- acls ........................................[0 ACL]
  o- luns ........................................[0 LUN]
  o- portals ..................................[1 Portal]
    o- 192.168.122.137:3260.....................[OK]
```

25.1.5. LUN の設定

LUN を設定するには、最初にストレージオブジェクトを作成します。詳細は、「バックストアの作成」 を参照してください。

手順25.3 LUN の設定

1. 作成したストレージオブジェクトの LUN を作成します。

```
/iscsi/iqn.20...mple:444/tpg1> luns/ create /backstores/ramdisk/rd_backend
Created LUN 0.

/iscsi/iqn.20...mple:444/tpg1> luns/ create /backstores/block/block_backend
Created LUN 1.

/iscsi/iqn.20...mple:444/tpg1> luns/ create /backstores/fileio/file1
Created LUN 2.
```

2. 変更を表示します。

```
/iscsi/iqn.20...mple:444/tpg1> ls
o- tpg..................................... [enambled, auth]
  o- acls ........................................[0 ACL]
  o- luns ........................................[3 LUNs]
  |  o- lun0.........................[ramdisk/ramdisk1]
  |  o- lun1.....................[block/block1 (/dev/vdb1)]
```

```
| o- lun2.....[fileio/file1 (/foo.img)]
o- portals .....[1 Portal]
  o- 192.168.122.137:3260.....[OK]
```

注記

デフォルトの LUN 名は 0 で始まります。これは、Red Hat Enterprise Linux 6 で tgtd を使用する場合にあった 1 とは対照的です。

3. ACL を設定します。詳細は、「ACL の設定」 を参照してください。

重要

デフォルトでは、読み書きパーミッションを持つ LUN が作成されます。ACL を作成してから新しい LUN を追加すると、その LUN は、利用可能なすべての ACL に自動的にマッピングされます。セキュリティーリスクが発生する可能性があります。以下の手順に従って、読み取り専用として LUN を作成します。

手順25.4 読み取り専用 LUN の作成

1. 読み取り専用権限で LUN を作成するには、最初に次のコマンドを使用します。

```
/> set global auto_add_mapped_luns=false
Parameter auto_add_mapped_luns is now 'false'.
```

これにより、LUN が既存の ACL へ自動的にマッピングされないようになり、LUN を手動でマッピングできるようになります。

2. 次に、iscsi/target_iqn_name/tpg1/acls/initiator_iqn_name/ create mapped_lun=next_sequential_LUN_number tpg_lun_or_backstore=backstore write_protect=1 コマンドを使用して、LUN を手動で作成します。

```
/> iscsi/iqn.2015-06.com.redhat:target/tpg1/acls/iqn.2015-06.com.redhat:initiator/ create
mapped_lun=1 tpg_lun_or_backstore=/backstores/block/block2 write_protect=1
Created LUN 1.
Created Mapped LUN 1.
/> ls
o- / ..... [...]
  o- backstores ..... [...]
  <snip>
  o- iscsi ..... [Targets: 1]
  | o- iqn.2015-06.com.redhat:target ..... [TPGs: 1]
```

```
|   o- tpg1 ..... [no-gen-acls, no-auth]
|     o- acls ..... [ACLs: 2]
|     | o- iqn.2015-06.com.redhat:initiator .. [Mapped LUNs: 2]
|     | | o- mapped_lun0 ..... [lun0 block/disk1 (rw)]
|     | | o- mapped_lun1 ..... [lun1 block/disk2 (ro)]
|     o- luns ..... [LUNs: 2]
|     | o- lun0 ..... [block/disk1 (/dev/vdb)]
|     | o- lun1 ..... [block/disk2 (/dev/vdc)]
<snip>
```

(mapped_lun0 の (rw) とは異なり) mapped_lun1 行の最後に (ro) が表示されますが、これは、読み取り専用であることを表しています。

3. ACL を設定します。詳細は、「ACL の設定」 を参照してください。

25.1.6. ACL の設定

接続する各イニシエーターに ACL を作成します。これにより、イニシエーターが接続する際に認証が強制され、各イニシエーターに公開できるのは LUN のみとなります。通常、各イニシエーターは LUN への排他的アクセス権を持っています。ターゲットとイニシエーターにはどちらも一意の識別名があります。ACL を設定するには、イニシエーターの一意の名前がわかっている必要があります。open-iscsi イニシエーターの場合は、/etc/iscsi/initiatorname.iscsi にあります。

手順25.5 ACL の設定

1. acls ディレクトリーに移動します。

```
/iscsi/iqn.20...mple:444/tpg1> acls/
```

2. ACL を作成します。イニシエーターの /etc/iscsi/initiatorname.iscsi にあるイニシエーター名を使用するか、覚えやすい名前を使用する場合は、「iSCSI イニシエーターの作成」 を参照して、ACL がイニシエーターと一致するようにします。以下に例を示します。

```
/iscsi/iqn.20...444/tpg1/acls> create iqn.2006-04.com.example.foo:888
Created Node ACL for iqn.2006-04.com.example.foo:888
Created mapped LUN 2.
Created mapped LUN 1.
Created mapped LUN 0.
```

注記

指定した例の動作は、使用される設定によって異なります。この場合、グローバル設定 auto_add_mapped_luns が使用されます。これにより、作成された ACL に LUN が自動的にマッピングされます。

ターゲットサーバーの TPG ノードに、ユーザーが作成した ACL を設定します。

```
/iscsi/iqn.20...scsi:444/tpg1> set attribute generate_node_acls=1
```

3.
変更を表示します。

```
/iscsi/iqn.20...444/tpg1/acls> ls
o- acls ..................................................[1 ACL]
  o- iqn.2006-04.com.example.foo:888 ....[3 Mapped LUNs, auth]
    o- mapped_lun0 .............[lun0 ramdisk/ramdisk1 (rw)]
    o- mapped_lun1 .................[lun1 block/block1 (rw)]
    o- mapped_lun2 .................[lun2 fileio/file1 (rw)]
```

25.1.7. ファイバーチャネルオーバーイーサネット (FCoE) ターゲットの設定

「ファイバーチャネルオーバーイーサネットインターフェイスの設定」 で説明されているように、FCoE 経由で LUN をマウントする以外にも、targetcli を使用して FCoE 経由で他のマシンに LUN をエクスポートすることもできます。

重要

続行する前に、「ファイバーチャネルオーバーイーサネットインターフェイスの設定」 を参照して、基本的な FCoE 設定が完了し、fcoeadm -i が設定した FCoE インターフェイスを表示することを確認します。

手順25.6 FCoE ターゲットの設定

1.
FCoE ターゲットを設定するには、targetcli パッケージとその依存関係をインストールする必要があります。targetcli の基本および設定に関する詳細は、「ターゲットの設定」 を参照してください。

2.
FCoE インターフェイスに FCoE ターゲットインスタンスを作成します。

```
/> tcm_fc/ create 00:11:22:33:44:55:66:77
```

FCoE インターフェイスがシステムに存在する場合は、create の後にタブ補完を実行すると、利用可能なインターフェイスが一覧表示されます。そうでない場合は、fcoeadm -i が有効なインターフェイスを表示していることを確認します。

3. **バックストアをターゲットインスタンスにマッピングします。**

例25.1 ターゲットインスタンスへのバックストアのマッピングの例

```
/> tcm_fc/00:11:22:33:44:55:66:77
/> luns/ create /backstores/fileio/example2
```

4. **FCoE イニシエーターから LUN へのアクセスを許可します。**

```
/> acls/ create 00:99:88:77:66:55:44:33
```

これにより、そのイニシエーターが LUN にアクセスできるようになります。

5. **再起動後も変更を永続化するには、saveconfig コマンドを使用し、プロンプトが表示されたら yes と入力します。これを行わないと、システムの再起動後に設定が失われます。**

6. **exit を入力するか、または ctrl+D を入力して targetcli を終了します。**

25.1.8. targetcliを使用したオブジェクトの削除

バックストアを削除するには、次のコマンドを実行します。

```
/> /backstores/backstore-type/backstore-name
```

ACL などの iSCSI ターゲットの一部を削除する場合は、次のコマンドを使用します。

```
/> /iscsi/iqn-name/tpg/acls/ delete iqn-name
```

すべての ACL、LUN、およびポータルを含むターゲット全体を削除するには、次のコマンドを使用します。

```
/> /iscsi delete iqn-name
```

25.1.9. targetcli リファレンス

targetcli の詳細は、以下の資料を参照してください。

man targetcli

man ページの targetcliこれには、サンプルウォークスルーが含まれます。

Linux SCSI ターゲット Wiki

http://linux-iscsi.org/wiki/Targetcli

Andy Grover によるスクリーンキャスト

https://www.youtube.com/watch?v=BkBGTBadOO8

注記

これは、2012 年 2 月 28 日にアップロードされました。そのため、サービス名が targetcli から ターゲット に変更されました。

25.2. iSCSI イニシエーターの作成

Red Hat Enterprise Linux 7 では、iSCSI サービスはデフォルトで遅延して起動され、iscsiadm コマンドの実行後にサービスが起動します。

手順25.7 iSCSI イニシエーターの作成

1. iscsi-initiator-utils をインストールします。

```
# yum install iscsi-initiator-utils -y
```

2.
「ACL の設定」 で ACL にカスタム名が指定されている場合は、それに応じて /etc/iscsi/initiatorname.iscsi ファイルを変更します。以下に例を示します。

```
# cat /etc/iscsi/initiatorname.iscsi
InitiatorName=iqn.2006-04.com.example.node1

# vi /etc/iscsi/initiatorname.iscsi
```

3.
ターゲットを検出します。

```
# iscsiadm -m discovery -t st -p target-ip-address
10.64.24.179:3260,1 iqn.2006-04.com.example:3260
```

4.
手順 3 で検出したターゲット IQN でターゲットにログインします。

```
# iscsiadm -m node -T iqn.2006-04.com.example:3260 -l
Logging in to [iface: default, target: iqn.2006-04.com.example:3260, portal:
10.64.24.179,3260] (multiple)
Login to [iface: default, target: iqn.2006-04.com.example:3260, portal:
10.64.24.179,3260] successful.
```

この手順は、「ACL の設定」 で説明するように。特定のイニシエーター名が ACL に追加されている限り、同じ LUN に接続されている任意の数のイニシエーターが使用することができます。

5.
iSCSI ディスク名を確認して、この iSCSI ディスクにファイルシステムを作成します。

```
# grep "Attached SCSI" /var/log/messages

# mkfs.ext4 /dev/disk_name
```

disk_name を、/var/log/messages に表示される iSCSI ディスク名に置き換えます。

6.
ファイルシステムをマウントします。

```
# mkdir /mount/point
# mount /dev/disk_name /mount/point
```

/mount/point をパーティションのマウントポイントに置き換えます。

7. システムの起動時にファイルシステムを自動的にマウントするように /etc/fstab を編集します。

```
# vim /etc/fstab
/dev/disk_name /mount/point ext4 _netdev 0 0
```

disk_name を iSCSI ディスク名に置き換えます。

8. ターゲットからログオフします。

```
# iscsiadm -m node -T iqn.2006-04.com.example:3260 -u
```

25.3. チャレンジハンドシェイク認証プロトコルの設定

ACL を設定し、iSCSI イニシエーターを作成したら、チャレンジハンドシェイク認証プロトコル (CHAP) を設定します。ACL の設定方法および iSCSI イニシエーターの作成に関する詳細は、「ACL の設定」 および 「iSCSI イニシエーターの作成」 を参照してください。

CHAP を使用すると、パスワードでターゲットを保護できます。イニシエーターは、このパスワードでターゲットに接続できることを認識している必要があります。

手順25.8 ターゲットの CHAP の設定

1. 属性認証を設定します。

```
/iscsi/iqn.20...mple:444/tpg1> set attribute authentication=1

Parameter authentication is now '1'.
```

2. ユーザー ID とパスワードを設定します。

```
/iscsi/iqn.20...mple:444/tpg1> set auth userid=redhat

Parameter userid is now 'redhat'.
```

```
/iscsi/iqn.20...mple:444/tpg1> set auth password=redhat_passwd

Parameter password is now 'redhat_passwd'.
```

手順25.9 イニシエーターの CHAP の設定

1. **iscsid.conf ファイルを編集します。**

   - **iscsid.conf ファイルで CHAP 認証を有効にします。**

     ```
     # vi /etc/iscsi/iscsid.conf

     node.session.auth.authmethod = CHAP
     ```

     デフォルトでは、node.session.auth.authmethod オプションは None に設定されています。

   - **ターゲットのユーザー名とパスワードを iscsid.conf ファイルに追加します。**

     ```
     node.session.auth.username = redhat
     node.session.auth.password = redhat_passwd
     ```

2. **iscsid サービスを再起動します。**

   ```
   # systemctl restart iscsid.service
   ```

詳細は、targetcli および iscsiadm の man ページを参照してください。

25.4. ファイバーチャネル

本セクションでは、ファイバーチャネル API、ネイティブの Red Hat Enterprise Linux 7 ファイバーチャネルドライバー、およびこれらのドライバーのファイバーチャネル機能を説明します。

25.4.1. ファイバーチャネル API

以下は、ユーザー空間 API の提供に使用されるファイルを含む /sys/class/ ディレクトリーの一覧です。各項目では、ホスト番号は H、バス番号は B、ターゲットは T、論理ユニット番号(LUN)は L、リモートポート番号は R です。

重要な影響

マルチパスソフトウェアを使用している場合は、このセクションに記載される値のいずれかを変更する前に、ハードウェアベンダーにお問い合わせになることが推奨されます。

トランスポート： /sys/class/fc_transport/targetH:B:T/

- *port_id - 24 ビットポート ID/アドレス*
- *node_name - 64 ビットノード名*
- *port_name - 64 ビットポート名*

リモートポート： /sys/class/fc_remote_ports/rport-H:B-R/

- *port_id*
- *node_name*
- *port_name*
- *dev_loss_tmo: scsi デバイスがシステムから削除されるタイミングを制御します。dev_loss_tmo がトリガーされると、scsi デバイスが削除されます。*

  multipath.conf では、dev_loss_tmo を infinity に設定できます。この値は 2,147,483,647 秒、または 68 年に設定し、dev_loss_tmo の最大値です。

  Red Hat Enterprise Linux 7 では、fast_io_fail_tmo オプションを設定しないと、dev_loss_tmo は 600 秒に制限されます。デフォルトでは、multipathd サービスが実行している場合は、Red Hat Enterprise Linux 7 で fast_io_fail_tmo が 5 秒に設定されています。それ以外の場合は、off に設定されます。

- *fast_io_fail_tmo: リンクに bad のマークを付けるまで待機する秒数を指定します。リンクに bad のマークが付けられると、対応するパス上の既存の実行中の I/O または新し*

い I/O が失敗します。

I/O がブロックされたキューにある場合、dev_loss_tmo の期限が切れ、キューがブロックが解除されるまで失敗しません。

fast_io_fail_tmo を off 以外の値に設定すると、dev_loss_tmo はカプセル化されません。fast_io_fail_tmo を off に設定すると、システムからデバイスが削除されるまで I/O は失敗します。fast_io_fail_tmo を数値に設定すると、fast_io_fail_tmo タイムアウトがトリガーされると、I/O はすぐに失敗します。

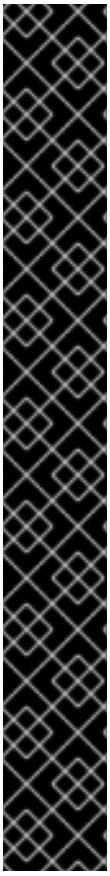
Host: /sys/class/fc_host/hostH/

- *port_id*
- *issue_lip: リモートポートを再検出するようにドライバーに指示します。*

25.4.2. ネイティブファイバーチャネルのドライバーおよび機能

Red Hat Enterprise Linux 7 には、以下のネイティブファイバーチャネルドライバーが同梱されます。

- *lpfc*
- *qla2xxx*
- *zfcp*
- *bfa*



重要

qla2xxx ドライバーは、デフォルトでイニシエーターモードで実行されます。Linux-IO で qla2xxx を使用するには、対応する qlini_mode モジュールパラメーターでファイバーチャネルターゲットモードを有効にします。

まず、ql2200-firmware などの qla デバイス用のファームウェアパッケージがインストールされていることを確認します。

ターゲットモードを有効にするには、以下のパラメーターを /usr/lib/modprobe.d/qla2xxx.conf qla2xxx モジュール設定ファイルに追加します。

```
options qla2xxx qlini_mode=disabled
```

次に、dracut -f コマンドを使用して初期 ramdisk (initrd)を再構築し、システムを再起動して変更を有効にします。

表25.1「ファイバーチャネル API の機能」 は、各ネイティブの Red Hat Enterprise Linux 7 ドライバーのさまざまなファイバーチャネル API 機能について説明します。X は、機能のサポートを示します。

表25.1 ファイバーチャネル API の機能

	lpfc	qla2xxx	zfcp	bfa
Transport port_id	X	X	X	X
Transport node_name	X	X	X	X
Transport port_name	X	X	X	X
リモートポートの dev_loss_tmo	X	X	X	X
リモートポート fast_io_fail_tmo	X	X [a]	X [b]	X

	lpfc	qla2xxx	zfcp	bfa
host port_id	X	X	X	X
Host issue_lip	X	X		X

[a] Red Hat Enterprise Linux 5.4 以降でサポート

[b] Red Hat Enterprise Linux 6.0 以降でサポート

25.5. ファイバーチャネルオーバーイーサネットインターフェイスの設定

ファイバーチャネルオーバーイーサネット (FCoE) インターフェイスの設定および導入には、以下の 2 つのパッケージが必要です。

- fcoe-utils
- lldpad

これらのパッケージをインストールしたら、以下の手順を実行して、仮想 LAN (VLAN) 上で FCoE を有効にします。

手順25.10 FCoE を使用するためにイーサネットインターフェイスの設定

1. 新しい VLAN を設定するには、既存のネットワークスクリプト（例： /etc/fcoe/cfg-eth0 ）のコピーを作成し、名前を FCoE をサポートするイーサネットデバイスに変更します。これにより、設定するデフォルトファイルが提供されます。FCoE デバイスが ethX の場合は、次のコマンドを実行します。

```
# cp /etc/fcoe/cfg-ethx  /etc/fcoe/cfg-ethX
```

必要に応じて cfg-ethX の内容を変更します。特に、ハードウェアデータセンターブリッジ交換 Exchange (DCBX)プロトコルクライアントを実装するネットワークインターフェイスの場合は、DCB_REQUIRED を no に設定します。

2.
システムの起動時にデバイスを自動的にロードする場合は、対応する /etc/sysconfig/network-scripts/ifcfg-ethX ファイルで ONBOOT=yes を設定します。たとえば、FCoE デバイスが eth2 の場合は、それに応じて /etc/sysconfig/network-scripts/ifcfg-eth2 を編集します。

3.
以下を実行して、データセンターのブリッジングデーモン(dcbd)を起動します。

```
# systemctl start lldpad
```

4.
ハードウェア DCBX クライアントを実装するネットワークインターフェイスの場合は、この手順を省略してください。

ソフトウェア DCBX クライアントを必要とするインターフェイスの場合は、以下のコマンドを実行して、イーサネットインターフェイスでデータセンターブリッジングを有効にします。

```
# dcbtool sc ethX dcb on
```

次に、以下を実行して、イーサネットインターフェイスで FCoE を有効にします。

```
# dcbtool sc ethX app:fcoe e:1
```

このコマンドは、イーサネットインターフェイスの dcbd 設定が変更されていない場合にのみ機能することに注意してください。

5.
現在使用している FCoE デバイスを読み込みます。

```
# ip link set dev ethX up
```

6.
以下を使用して FCoE を起動します。

```
# systemctl start fcoe
```

ファブリック上のその他の設定がすべて正しくなると、FCoE デバイスがすぐに表示されます。設定した FCoE デバイスを表示するには、次のコマンドを実行します。

```
# fcoeadm -i
```

FCoE を使用するようにイーサネットインターフェイスを正しく設定した後、Red Hat は、システムの起動時に FCoE および lldpad サービスを設定することを推奨します。これを行うには、systemctl ユーティリティーを使用します。

```
# systemctl enable lldpad
```

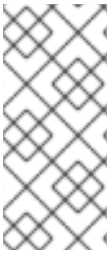
```
# systemctl enable fcoe
```

注記

systemctl stop fcoe コマンドを実行するとデーモンは停止しますが、FCoE インターフェイスの設定はリセットされません。これを行うには、# systemctl -s SIGHUP kill fcoe コマンドを実行します。

Red Hat Enterprise Linux 7 以降、Network Manager には、DCB 対応のイーサネットインターフェイスの DCB 設定をクエリーおよび設定する機能があります。

25.6. システムの起動時に FCOE インターフェイスを自動マウントする設定



注記

本セクションの手順は、Red Hat Enterprise Linux 6.1 の時点で /usr/share/doc/fcoe-utils-バージョン/README で入手できます。マイナーリリース全体で考えられる変更については、そのドキュメントを参照してください。

新しく検出されたディスクは、udev ルール、autofs、およびその他の同様の方法でマウントできます。ただし、特定のサービスの起動時に FCoE ディスクのマウントが必要になる場合があります。このような場合は、fcoe サービスが実行されるとすぐ に、FCoE ディスクを必要とするサービスが開始される 前 に、FCoE ディスクをマウントする必要があります。

システムの起動時に FCoE ディスクを自動的にマウントするように設定するには、fcoe サービスの起動スクリプトに適切な FCoE マウントコードを追加します。fcoe 起動スクリプトは /lib/systemd/system/fcoe.service です。

FCoE マウントコードは、単純なフォーマットの FCoE ディスク、LVM、またはマルチパスデバイスノードのいずれを使用しているかに関係なく、システム設定ごとに異なります。

-

例25.2 FCoE マウントコード

以下は、/etc/fstab のワイルドカードで指定されたファイルシステムをマウントするための FCoE マウントコードの例です。

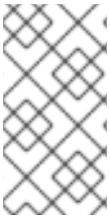
```
mount_fcoe_disks_from_fstab()
{
    local timeout=20
    local done=1
    local fcoe_disks=($(egrep 'by-path\/fc-.*_netdev' /etc/fstab | cut -d ' ' -f1))

    test -z $fcoe_disks && return 0

    echo -n "Waiting for fcoe disks . "
    while [ $timeout -gt 0 ]; do
 for disk in ${fcoe_disks[*]}; do
  if ! test -b $disk; then
   done=0
   break
  fi
 done

 test $done -eq 1 && break;
 sleep 1
 echo -n ". "
 done=1
 let timeout--
    done

    if test $timeout -eq 0; then
 echo "timeout!"
    else
 echo "done!"
    fi

    # mount any newly discovered disk
    mount -a 2>/dev/null
}
```

mount_fcoe_disks_from_fstab 関数は、fcoe サービススクリプトが fcoe mon デーモンを起動した後 に呼び出す必要があります。これにより、/etc/fstab で以下のパスで指定された FCoE ディスクがマウントされます。

```
/dev/disk/by-path/fc-0xXX:0xXX /mnt/fcoe-disk1 ext3  defaults,_netdev    0 0
/dev/disk/by-path/fc-0xYY:0xYY /mnt/fcoe-disk2 ext3  defaults,_netdev    0 0
```

fc- および _netdev サブ文字列を持つエントリーにより、mount_fcoe_disks_from_fstab 関数が FCoE ディスクマウントエントリーを識別できます。/etc/fstab エントリーの詳細は、man 5 fstab を

参照してください。



注記

fcoe サービスは、FCoE ディスク検出のタイムアウトを実装しません。このため、FCoE マウントコードには独自のタイムアウト期間を実装する必要があります。

25.7. ISCSI

本セクションでは、iSCSI API および iscsiadm ユーティリティーを説明します。iscsiadm ユーティリティーを使用する前に、yum install iscsi-initiator-utils を実行して iscsi-initiator-utils パッケージを最初にインストールします。

Red Hat Enterprise Linux 7 では、iSCSI サービスはデフォルトで遅延起動します。root が iSCSI デバイスにない場合や、node.startup = automatic でマークされたノードがない場合、iSCSI サービスは iscsiadm コマンドが実行されるまで起動しません。これには iscsid または iscsi カーネルモジュールの開始が必要になります。たとえば、検出コマンド iscsiadm -m discovery -t st -p ip:port を実行すると、iscsiadm が iSCSI サービスを開始します。

iscsid デーモンを強制的に実行し、iSCSI カーネルモジュールを読み込むには、systemctl start iscsid.service を実行します。

25.7.1. iSCSI API

セッションの実行に関する情報を取得するには、次のコマンドを実行します。

```
# iscsiadm -m session -P 3
```

このコマンドは、セッション/デバイスの状態、セッション ID (sid)、いくつかのネゴシエートしたパラメーター、およびセッション経由でアクセス可能な SCSI デバイスを表示します。

より短い出力 (たとえば sid-to-node マッピングのみの表示) には、次のコマンドを実行します。

```
# iscsiadm -m session -P 0
```

または

```
# iscsiadm -m session
```

以下のコマンドは、実行中のセッションのリストを次の形式で出力します。

```
driver [sid] target_ip:port,target_portal_group_tag proper_target_name
```

例25.3 iscsisadm -m session コマンドの出力

以下に例を示します。

```
# iscsiadm -m session

tcp [2] 10.15.84.19:3260,2 iqn.1992-08.com.netapp:sn.33615311
tcp [3] 10.15.85.19:3260,3 iqn.1992-08.com.netapp:sn.33615311
```

iSCSI API の詳細は、/usr/share/doc/iscsi-initiator-utils-version/README を参照してください。

25.8. 永続的な命名

Red Hat Enterprise Linux では、ストレージデバイスを識別する方法が複数あります。特にドライブへのインストール時やドライブの再フォーマット時に誤ったデバイスにアクセスしないようにするため、適切なオプションを使用して各デバイスを識別することが重要になります。

25.8.1. ストレージデバイスのメジャー番号およびマイナー番号

sd ドライバーによって管理されるストレージデバイスは、メジャーデバイス番号とそれに関連するマイナー番号の集合によって内部的に識別されます。この目的で使用される主要なデバイス番号は、連続した範囲ではありません。各ストレージデバイスは、メジャー番号とマイナー番号の範囲で表され、デバイス全体またはデバイス内のパーティションを識別するために使用されます。デバイスに割り当てられたメジャー番号とマイナー番号は、sd <letter (s)>[number (s)] の形で直接関連付けられます。sd ドライバーが新しいデバイスを検出するたびに、利用可能なメジャー番号とマイナー番号の範囲が割り当てられます。オペレーティングシステムからデバイスを削除すると、メジャー番号およびマイナー番号の範囲が解放され、後で再利用できます。

メジャー番号およびマイナー番号の範囲と関連する sd 名は、検出時に各デバイスに割り当てられます。つまり、デバイス検出の順序が変更されると、メジャー番号とマイナー番号の範囲と関連する sd 名間の関連付けが変更される可能性があります。これは、一部のハードウェア設定 (たとえば、内部 SCSI コントローラーとシャーシ内の物理的な場所によって割り当てられた SCSI ターゲット ID を持つディスク) ではまれですが、それでも発生する可能性があります。これが発生する可能性がある状況の例を以下に示します。

ディスクの電源がオンにならなかったり、SCSI コントローラーへの応答に失敗したりする場合があります。この場合は、通常のデバイスプローブにより検出されません。ディスクはシステムからアクセスできず、後続のデバイスには、シフトダウンした関連する sd 名など、メジャー番号とマイナー番号の範囲があります。たとえば、通常 sdb と呼ばれるディスクが検出されない場合、通常は sdc と呼ばれるディスクは sdb として表示されます。

- SCSI コントローラー (ホストバスアダプター、または HBA) の初期化に失敗し、その HBA に接続したすべてのディスクが検出されない場合があります。その後プローブされた HBA に接続されているディスクには、異なるメジャー番号とマイナー番号の範囲、および関連する sd 名が割り当てられます。

- システムに異なるタイプの HBA が存在すると、ドライバーの初期化の順序が変更される可能性があります。これにより、これらの HBA に接続されたディスクが別の順序で検出されます。これは、HBA がシステム上の別の PCI スロットに移動した場合にも発生することがあります。

- ストレージアレイや干渉するスイッチの電源が切れた場合など、ストレージデバイスがプローブされたときに、ファイバーチャネル、iSCSI、または FCoE アダプターを持つシステムに接続されたディスクがアクセスできなくなる可能性があります。これは、電源障害の後にシステムが再起動したときに、ストレージアレイのオンラインにかかる時間が、システムの起動にかかる時間よりも長くなる場合に発生します。一部のファイバーチャネルドライバーは、永続的な SCSI ターゲット ID を WWPN マッピングに指定するメカニズムをサポートしていますが、メジャー番号およびマイナー番号の範囲や関連する sd 名は予約されず、一貫性のある SCSI ターゲット ID 番号のみが提供されます。

このため、/etc/fstab ファイルなど、デバイスを参照する際にメジャー番号およびマイナー番号の範囲または関連する sd 名を使用することは望ましくありません。間違ったデバイスがマウントされ、データが破損する可能性があります。

ただし、場合によっては、別のメカニズムが使用されている場合でも sd 名を参照する必要がある場合があります（デバイスによってエラーが報告される場合など）。これは、Linux カーネルがデバイスに関するカーネルメッセージで sd 名（および SCSI ホスト/チャネル/ターゲット/LUN タプル）を使用するためです。

25.8.2. World Wide Identifier (WWID)

World Wide Identifier (WWID) は、デバイスを確実に識別するために使用できます。これは、永続的で、SCSI 規格によりすべての SCSI デバイスが必要とするシステムに依存しない識別子です。各ストレージデバイスの WWID 識別子は一意となることが保証され、デバイスのアクセスに使用されるパスに依存しません。

この識別子は、SCSI Inquiry を発行して Device Identification Vital Product Data (0x83ページ)ま

たは Unit Serial Number (0x80ページ)を取得することで取得できます。これらの WWID から現在の /dev/sd 名へのマッピングは、/dev/disk/by-id/ ディレクトリーに保持されているシンボリックリンクで確認できます。

> *例25.4 WWID*
>
> *たとえば、ページ 0x83 識別子を持つデバイスには、以下のものがあります。*
>
> ```
> scsi-3600508b400105e210000900000490000 -> ../../sda
> ```
>
> *または、ページ 0x80 識別子を持つデバイスには、以下が含まれます。*
>
> ```
> scsi-SSEAGATE_ST373453LW_3HW1RHM6 -> ../../sda
> ```

Red Hat Enterprise Linux は、WWID ベースのデバイス名から、そのシステムの現在の /dev/sd 名への適切なマッピングを自動的に維持します。デバイスへのパスが変更したり、別のシステムからデバイスにアクセスする場合でも、アプリケーションはディスク上のデータを参照するために /dev/disk/by-id/ 名を使用できます。

システムからデバイスへのパスが複数ある場合、DM Multipath は WWID を使用してこれを検出します。その後、DM Multipath は /dev/mapper/wwid ディレクトリーに単一の疑似デバイスを表示します（例： /dev/mapper/3600508b400105df70000e00000ac0000 ）。

multipath -l コマンドは、非永続識別子へのマッピングを表示します。Host:Channel:Target: Target :LUN、/dev/sd 名、および major:minor 番号。

```
3600508b400105df70000e00000ac0000 dm-2 vendor,product
[size=20G][features=1 queue_if_no_path][hwhandler=0][rw]
\_ round-robin 0 [prio=0][active]
 \_ 5:0:1:1 sdc 8:32  [active][undef]
 \_ 6:0:1:1 sdg 8:96  [active][undef]
\_ round-robin 0 [prio=0][enabled]
 \_ 5:0:0:1 sdb 8:16  [active][undef]
 \_ 6:0:0:1 sdf 8:80  [active][undef]
```

DM Multipath は、各 WWID ベースのデバイス名から、システム上の対応する /dev/sd 名への適切なマッピングを自動的に維持します。これらの名前は、パスが変更しても持続し、他のシステムからデバイスにアクセスする際に一貫性を保持します。

(DM Multipathの) user_friendly_names 機能を使用すると、WWID は /dev/mapper/mpathn 形式

の名前にマップされます。デフォルトでは、このマッピングは /etc/multipath/bindings ファイルで維持されます。これらの mpathn 名は、そのファイルが維持されている限り永続的です。

重要

user_friendly_names を使用する場合は、クラスター内で一貫した名前を取得するために追加の手順が必要です。DM Multipath ブックの Consistent Multipath Device Names in a Cluster を参照してください。

システムが提供するこれらの永続名に加えて、udev ルールを使用して独自の永続的な名前を実装し、ストレージの WWID にマップすることもできます。

25.8.3. /dev/disk/by-*の udev メカニズムによって管理されるデバイス名

udev メカニズムは、以下の 3 つの主要コンポーネントで設定されています。

カーネル

デバイスの追加、削除、または変更時にユーザー空間に送信されるイベントを生成します。

udevd サービス

イベントを受信します。

udev ルール

udev サービスがカーネルイベントを受信するときに実行するアクションを指定します。

このメカニズムは、ストレージデバイスだけでなく、Linux のすべてのデバイスに使用されます。ストレージデバイスの場合、Red Hat Enterprise Linux には、/dev/disk/ ディレクトリーにシンボリックリンクを作成する udev ルールが含まれます。これにより、ストレージデバイスは、そのコンテンツ、一意の識別子、シリアル番号、またはデバイスへのアクセスに使用されるハードウェアパスによって参照されます。

/dev/disk/by-label/

このディレクトリーのエントリーは、デバイスに格納されているコンテンツ (つまりデータ) 内のラベルにより、ストレージデバイスを参照するシンボリック名を提供します。blkid ユーティリ

ティーは、デバイスからデータを読み込み、デバイスの名前（ラベル）を決定するために使用されます。以下に例を示します。

```
/dev/disk/by-label/Boot
```

注記

情報はデバイスのコンテンツ (つまりデータ) から取得されるため、コンテンツが別のデバイスにコピーされても、ラベルは同じままになります。

このラベルは、以下の構文を使用して /etc/fstab でデバイスを参照することもできます。

```
LABEL=Boot
```

/dev/disk/by-uuid/

このディレクトリーのエントリーは、デバイスに格納されているコンテンツ (つまりデータ) 内の 一意の ID により、ストレージデバイスを参照するシンボリック名を提供します。blkid ユーティリティーは、デバイスからデータを読み取って、デバイスの一意の識別子(UUID)を取得するために使用されます。以下に例を示します。

```
UUID=3e6be9de-8139-11d1-9106-a43f08d823a6
```

/dev/disk/by-id/

このディレクトリーのエントリーは、(その他のすべてのストレージデバイスとは異なる) 固有の ID により、ストレージデバイスを参照するシンボリック名を提供します。この識別子はデバイスのプロパティーですが、デバイスのコンテンツ (つまりデータ) には保存されません。以下に例を示します。

```
/dev/disk/by-id/scsi-3600508e000000000ce506dc50ab0ad05
```

```
/dev/disk/by-id/wwn-0x600508e000000000ce506dc50ab0ad05
```

この id は、デバイスの World-Wide ID またはデバイスのシリアル番号から取得されます。/dev/disk/by-id/ エントリーには、パーティション番号を含めることもできます。以下に例を示します。

```
/dev/disk/by-id/scsi-3600508e000000000ce506dc50ab0ad05-part1
```

```
/dev/disk/by-id/wwn-0x600508e000000000ce506dc50ab0ad05-part1
```

/dev/disk/by-path/

このディレクトリーのエントリーは、シンボリック名を提供します。このシンボリック名は、PCI 階層内のストレージコントローラーへの参照から始まり、デバイスのアクセスに使用されるハードウェアパスによりストレージデバイスを参照し、SCSI ホスト、チャネル、ターゲット、LUN 番号、パーティション番号 (オプション) を含みます。これらの名前は、メジャー番号およびマイナー番号または sd 名の使用が推奨されますが、ファイバーチャネル SAN 環境ではターゲット番号が変更されないように注意する必要があります（たとえば、永続的なバインディングを使用）。また、ホストアダプターを別の PCI スロットに移動すると、名前が更新されます。また、HBA がプローブに失敗した場合、ドライバーが別の順序で読み込まれている場合、またはシステムに新しい HBA がインストールされている場合に、SCSI ホスト番号が変更する可能性があります。パス別のリスト表示の例を以下に示します。

```
/dev/disk/by-path/pci-0000:03:00.0-scsi-0:1:0:0
```

/dev/disk/by-path/ エントリーには、以下のようなパーティション番号を含めることもできます。

```
/dev/disk/by-path/pci-0000:03:00.0-scsi-0:1:0:0-part1
```

25.8.3.1. udev デバイス命名規則の制約

udev 命名規則の制約の一部は次のとおりです。

- udev メカニズムは、udev イベントに対して udev ルールが処理される際にストレージデバイスをクエリーする機能に依存する可能性があるため、クエリーの実行時にデバイスにアクセスできない可能性があります。これは、ファイバーチャネル、iSCSI、または FCoE ストレージデバイスといった、デバイスがサーバーシャーシにない場合に発生する可能性が高くなります。
- また、カーネルはいつでも udev イベントを送信する可能性があり、デバイスにアクセスできない場合に /dev/disk/by-*/ リンクが削除される可能性があります。
- udev イベントが生成され、それが処理されるまでに遅延が生じる可能性があります。たとえば、多数のデバイスが検出され、ユーザー空間の udevd サービスが各デバイスのルールを処理するのにある程度の時間がかかる場合などです。これにより、カーネルがデバイスを検出したときと、/dev/disk/by-*/ 名が利用可能になるまでの間に遅延が発生する可能性がありました。

- ルールによって呼び出される blkid などの外部プログラムはデバイスを短期間開き、他の目的でデバイスにアクセスできなくなる可能性があります。

25.8.3.2. 永続的な命名属性の変更

udev の命名属性は永続的ですが、システムの再起動後も独自に変更されないため、設定可能なものもあります。以下の永続的な命名属性には、カスタム値を設定できます。

- UUID: ファイルシステムの UUID

- LABEL: ファイルシステムラベル

UUID 属性と LABEL 属性はファイルシステムに関連するため、使用するツールはそのパーティションのファイルシステムによって異なります。

- XFS ファイルシステムの UUID 属性または LABEL 属性を変更するには、ファイルシステムをアンマウントしてから、xfs_admin ユーティリティーを使用して属性を変更します。

```
# umount /dev/device
# xfs_admin [-U new_uuid] [-L new_label] /dev/device
# udevadm settle
```

- ext4、ext3、または ext2 ファイルシステムの UUID 属性または LABEL 属性を変更するには、tune2fs ユーティリティーを使用します。

```
# tune2fs [-U new_uuid] [-L new_label] /dev/device
# udevadm settle
```

new_uuid を設定する UUID （例： 1cdfbc07-1c90-4984-b5ec-f61943f5ea50 ）に置き換えます。new_label を、ラベル（例： backup_data ）に置き換えます。

注記

udev 属性の変更はバックグラウンドで行われ、時間がかかる場合があります。udevadm settle コマンドは、変更が完全に登録されるまで待機します。これにより、次のコマンドが新しい属性を正しく利用できるようになります。

また、新しいデバイスの作成後に コマンドを使用する必要があります。たとえば、parted ツールを使用してカスタムの PARTUUID 属性または PARTLABEL 属性でパーティションを作成した後、または新規ファイルシステムを作成した後などです。

25.9. ストレージデバイスの削除

ストレージデバイス自体へのアクセスを削除する前に、デバイスからのデータのバックアップを作成することが推奨されます。その後、I/O をフラッシュして、デバイスへのオペレーティングシステムの参照をすべて削除します (以下で説明します)。デバイスがマルチパスを使用している場合は、マルチパスの "pseudo device" (「World Wide Identifier (WWID)」) と、デバイスへのパスを表す識別子のそれぞれに対してこれを実行します。マルチパスデバイスへのパスのみを削除し、それ以外のパスが残る場合は、「ストレージデバイスまたはパスの追加」 で説明されているように、手順が簡単になります。

I/O フラッシュは負荷に追加されるため、システムがメモリー圧力下にある場合は、ストレージデバイスを取り外すことは推奨されません。メモリー不足のレベルを確認するには、vmstat 1 100 コマンドを実行します。以下の場合はデバイスの削除は推奨されません。

- *空きメモリーは、100 あたり 10 を超えるサンプルで、合計メモリーの 5% 未満です（コマンド free を使用して合計メモリーを表示することもできます）。*
- *スワッピングが有効である(vmstat 出力の si 列および so 列がゼロ以外の)。*

デバイスへのアクセスをすべて削除する一般的な手順は、以下のとおりです。

手順25.11 デバイスの完全削除

1. *必要に応じて、デバイスのすべてのユーザーとデバイスデータのバックアップを閉じます。*
2. *umount を使用して、デバイスをマウントしたファイルシステムをアンマウントします。*
3.

デバイスを使用して、md ボリュームおよび LVM ボリュームからデバイスを削除します。デバイスが LVM ボリュームグループのメンバーである場合は、pvmove コマンドを使用してデバイスからデータを移動し、vgreduce コマンドを使用して物理ボリュームを削除し、（必要に応じて） pvremove を使用してディスクから LVM メタデータを削除する必要がある場合があります。

4.

multipath -l コマンドを実行して、マルチパス デバイスとして設定されているデバイスの一覧を表示します。デバイスがマルチパスデバイスとして設定されている場合は、multipath -f device コマンドを実行して未処理の I/O をフラッシュし、マルチパスデバイスを削除します。

5.

使用されているパスに、未処理の I/O をフラッシュします。これは、I/O フラッシュを引き起こす umount または vgreduce 操作がない raw デバイスにとって重要です。この手順は、次の場合にのみ行う必要があります。

- デバイスがマルチパスデバイスとして設定されていない場合。または、
- デバイスはマルチパスデバイスとして設定され、I/O は過去のある時点で個々のパスに直接発行されている場合。

未処理の I/O をフラッシュするには、次のコマンドを使用します。

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

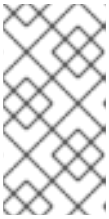
6.

システム上のアプリケーション、スクリプト、またはユーティリティーで、/dev/sd、/dev/disk/by-path、または major:minor 番号など、デバイスのパスベースの名前への参照をすべて削除します。これは、将来追加される別のデバイスが、現在のデバイスと誤認されないようにするために重要です。

7.

最後に、SCSI サブシステムからデバイスへの各パスを削除します。これを行うには、echo 1 > /sys/block/device-name/device/delete コマンドを使用します。device-name は sde などになります。

この操作の別のバリエーションは echo 1 > /sys/class/scsi_device/h:c:t:l/device/delete です。ここで、h は HBA 番号、c は HBA のチャンネル、t は SCSI ターゲット ID、l は LUN です。



注記

このコマンドの古い形式である echo "scsi remove-single-device 0 0 0 0" > /proc/scsi/scsi が非推奨になりました。

lsscsi、scsi_id、multipath -l、ls -l /dev/disk/by- などのさまざまなコマンドから、デバイスの device-name、HBA 番号、HBA チャンネル、SCSI ターゲット ID、および LUN を決定できます。*

手順25.11「デバイスの完全削除」 を実行したら、実行中のシステムからデバイスを物理的に削除できます。その際に、他のデバイスへの I/O を停止する必要はありません。

変更を反映するようにオペレーティングシステムの状態を更新する他の手順 (「ストレージの相互接続のスキャン」 に説明されているようなデバイスの物理的な取り外しとそれに続く SCSI バスの再スキャンなど) は推奨しません。これにより、I/O タイムアウトによる遅延が発生し、デバイスが予想外に削除される可能性があります。相互接続の再スキャンを実行する必要がある場合は、「ストレージの相互接続のスキャン」 の説明に従って、I/O を一時停止しているときに再スキャンを実行する必要があります。

25.10. ストレージデバイスへのパスの削除

マルチパスを使用するデバイスへのパスを (デバイスへのその他のパスには影響を及ぼさずに) 削除する場合、一般的な手順は以下のようになります。

手順25.12 ストレージデバイスへのパスの削除

1. *システム上のアプリケーション、スクリプト、またはユーティリティーで、/dev/sd、/dev/disk/by-path、または major:minor 番号など、デバイスのパスベースの名前への参照をすべて削除します。これは、将来追加される別のデバイスが、現在のデバイスと誤認されないようにするために重要です。*

2. *echo offline > /sys/block/sda/device/state を使用してパスをオフラインにします。*

   これにより、このパスのデバイスに送信される後続の I/O がすぐに失敗します。device-mapper-multipath は、デバイスへの残りのパスを引き続き使用します。

3. *SCSI サブシステムからパスを削除します。これを行うには、echo 1 > /sys/block/device-name/device/delete コマンドを使用します。device-name は、たとえば 手順25.11「デバイスの完全削除」 で説明されているように、sde にすることができます。*

手順25.12「ストレージデバイスへのパスの削除」 の実行後に、実行中のシステムからパスを安全に削除できます。この処理中に、I/O を停止する必要はありません。device-mapper-multipath は、設定されたパスのグループ化とフェイルオーバーポリシーに従って I/O を残りのパスに再ルーティングします。

ケーブルを物理的に取り外した後、SCSI バスを再スキャンして、変更を反映するようにオペレーティングシステムの状態を更新するなど、他の手順は推奨しません。これにより、I/O タイムアウトによる遅延が発生し、デバイスが予想外に削除される可能性があります。相互接続の再スキャンを実行する必要がある場合は、「ストレージの相互接続のスキャン」 の説明に従って、I/O を一時停止しているときに再スキャンを実行する必要があります。

25.11. ストレージデバイスまたはパスの追加

デバイスを追加する際には、新しいデバイスに割り当てたパスベースのデバイス名（例：/dev/sd 名、major:minor 番号、および /dev/disk/by-path 名など）が、以前は削除されているデバイスによって使用されている可能性があることに注意してください。そのため、パスベースのデバイス名への古い参照がすべて削除されていることを確認してください。新しいデバイスを古いデバイスと間違える可能性があります。

手順25.13 ストレージデバイスまたはパスの追加

1. *ストレージデバイスまたはパスを追加する最初の手順は、新規ストレージデバイス、または既存デバイスへの新規パスへのアクセスを物理的に有効にすることです。これは、ファイバーチャネルまたは iSCSI ストレージサーバーで、ベンダー固有のコマンドを使用して実行します。その際、ホストに提示される新しいストレージの LUN 値をメモしてください。ストレージサーバーがファイバーチャネルの場合は、ストレージサーバーの WWNN (World Wide Node Name) も確認し、ストレージサーバーのすべてのポートに 1 つの WWNN があるかどうかを判断します。そうでない場合は、新しい LUN へのアクセスに使用される各ポートの World Wide Port Name (WWPN) をメモします。*

2. *次に、オペレーティングシステムが新しいストレージデバイス、または既存のデバイスへのパスを認識するようにします。推奨されるコマンドは以下のとおりです。*

```
$ echo "c t l" > /sys/class/scsi_host/hosth/scan
```

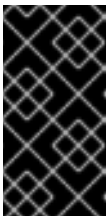
上記のコマンドでは、h は HBA 番号、c は HBA のチャンネル、t は SCSI ターゲット ID、l は LUN です。

注記

このコマンドの古い形式である echo "scsi add-single-device 0 0 0 0" > /proc/scsi/scsi が非推奨になりました。

a.

ファイバーチャネルハードウェアによっては、ループ初期化プロトコル (LIP) 操作が実行されるまで、RAID アレイで新しく作成された LUN がオペレーティングシステムに表示されない場合があります。実行手順については、「ストレージの相互接続のスキャン」を参照してください。



重要

LIP が必要な場合は、この操作の実行中に I/O を停止する必要があります。

b.

新しい LUN が RAID アレイに追加されましたが、オペレーティングシステムでまだ設定されていない場合は、sg 3_utils パッケージに含まれる sg_luns コマンドを使用して、アレイでエクスポートされている LUN の一覧を確認します。これにより、SCSI REPORT LUNS コマンドを RAID アレイに発行し、存在する LUN の一覧を返します。

すべてのポートに単一の WWNN を実装するファイバーチャネルストレージサーバーの場合は、sysfs で WWNN を検索して、正しい h、c、および t 値(HBA 番号、HBA チャネル、および SCSI ターゲット ID)を決定できます。

例25.5 正しい h、c、および t の値を決定する

たとえば、ストレージサーバーの WWNN が 0x5006016090203181 の場合は、以下を使用します。

```
$ grep 5006016090203181 /sys/class/fc_transport/*/node_name
```

これにより、以下のような出力が表示されます。

```
/sys/class/fc_transport/target5:0:2/node_name:0x5006016090203181
/sys/class/fc_transport/target5:0:3/node_name:0x5006016090203181
/sys/class/fc_transport/target6:0:2/node_name:0x5006016090203181
/sys/class/fc_transport/target6:0:3/node_name:0x5006016090203181
```

これは、このターゲットへのファイバーチャネルルートが 4 つあることを示していま

> す (それぞれ 2 つのストレージポートにつながるシングルチャネル HBA が 2 つ)。LUN の値が 56 であるとすると、次のコマンドは最初のパスを設定します。
>
> ```
> $ echo "0 2 56" >  /sys/class/scsi_host/host5/scan
> ```

これは、新規デバイスへのパスごとに行う必要があります。

すべてのポートに 1 つの WWNN を実装しないファイバーチャネルストレージサーバーの場合は、sysfs で各 WWPN を検索して、正しい HBA 番号、HBA チャネル、および SCSI ターゲット ID を決定できます。

HBA 番号、HBA チャネル、および SCSI ターゲット ID を決定する別の方法として、新しいデバイスと同じパスにすでに設定されているデバイスを参照する方法があります。これは、lsscsi、scsi_id、multipath -l、ls -l /dev/disk/by-* などのさまざまなコマンドで実行できます。この情報と新しいデバイスの LUN 番号を上記のように使用して、新しいデバイスへのパスをプローブおよび設定できます。

3.

デバイスにすべての SCSI パスを追加したら、multipath コマンドを実行して、デバイスが適切に設定されていることを確認します。この時点で、デバイスを md、LVM、mkfs、または mount に追加できます。

上記の手順に従うと、実行中のシステムにデバイスを安全に追加できます。その際に、他のデバイスへの I/O を停止する必要はありません。オペレーティングシステムの状態を更新して現在のデバイス接続を反映させる SCSI バスの再スキャン (またはリセット) を伴うその他の手順は、ストレージ I/O の進行中は推奨しません。

25.12. ストレージの相互接続のスキャン

特定のコマンドでは、1 つ以上の相互接続をリセット、スキャン、またはその両方を行うことができます。これにより、1 回の操作で複数のデバイスを追加および削除する可能性があります。このタイプのスキャンは、I/O 操作のタイムアウト時に遅延が発生したり、デバイスが予想外に削除される可能性があるため、混乱を招く可能性があります。Red Hat では、必要な場合にのみ、相互接続スキャンを使用することを推奨しています。ストレージの相互接続をスキャンする場合は、次の制限に注意してください。

- 影響を受ける相互接続のすべての I/O は、手順を実行する前に一時停止してフラッシュする必要があります。また、I/O を再開する前にスキャンの結果を確認する必要があります。
- デバイスの取り外しと同様に、システムのメモリーが逼迫している場合は、相互接続ス

キャンを行わないことが推奨されます。メモリー不足のレベルを確認するには、vmstat 1 100 コマンドを実行します。(サンプル 100 件の内 11 件以上で) 空きメモリーが合計メモリーの 5 % 未満の場合、相互接続スキャンは推奨されません。また、スワッピングがアクティブな場合(vmstat 出力のゼロ以外の si 列および so 列)は、相互接続スキャンは推奨されません。free コマンドは、合計メモリーを表示することもできます。

以下のコマンドを使用すると、ストレージの相互接続をスキャンできます。

echo "1" > /sys/class/fc_host/hostN/issue_lip

(N をホスト番号に置き換えます。)

この操作は、ループ初期化プロトコル(LIP) を実行し、相互接続をスキャンして、現在バス上にあるデバイスを反映するように SCSI レイヤーを更新します。基本的に、LIP はバスのリセットであり、デバイスの追加と削除を引き起こします。この手順は、ファイバーチャネル相互接続に新しい SCSI ターゲットを設定する場合に必要です。

issue_lip は非同期操作であることに注意してください。コマンドは、スキャン全体が完了する前に完了する可能性があります。issue_lip の終了タイミングを決定するには、/var/log/messages を監視する必要があります。

lpfc ドライバー、qla2xxx ドライバー、および bnx2fc ドライバーは、issue_lip をサポートします。Red Hat Enterprise Linux の各ドライバーがサポートする API 機能に関する詳細は、表 25.1「ファイバーチャネル API の機能」 を参照してください。

/usr/bin/rescan-scsi-bus.sh

/usr/bin/rescan-scsi-bus.sh スクリプトは、Red Hat Enterprise Linux 5.4 で導入されました。デフォルトでは、このスクリプトは、システムの SCSI バスをすべてスキャンし、SCSI レイヤーを更新して、バスの新しいデバイスを反映します。このスクリプトでは、デバイスの削除や LIP の実行を許可する追加オプションが提供されます。既知の問題など、このスクリプトの詳細は、「rescan-scsi-bus.sh を使用した論理ユニットの追加/削除」 を参照してください。

echo "- - -" > /sys/class/scsi_host/hosth/scan

これは、「ストレージデバイスまたはパスの追加」 で説明されているように、ストレージデバイスまたはパスを追加するコマンドと同じです。ただし、この場合は、チャネル番号、SCSI ターゲット ID、および LUN の値がワイルドカードに置き換わります。識別子とワイルドカードの組み合わせは自由なので、必要なだけ具体的に、あるいは幅広くコマンドを作成することができます。この手順では、LUN を追加しますが、削除することはありません。

modprobe --remove driver-name, modprobe driver-name

modprobe --remove driver-name コマンドの後に modprobe driver-nameコマンドを実行すると、ドライバー によって制御されるすべての相互接続の状態が完全に再初期化されます。かなり極端ですが、説明されているコマンドを使用することは、特定の状況で適切な場合があります。コマンドを使用して、たとえば、異なるモジュールパラメーター値でドライバーを再起動できます。

25.13. ISCSI 検出設定

デフォルトの iSCSI 設定ファイルは /etc/iscsi/iscsid.conf です。このファイルには、iscsid および iscsiadm で使用される iSCSI 設定が含まれます。

ターゲット検出中に、iscsiadm ツールは /etc/iscsi/iscsid.conf の設定を使用して、2 種類のレコードを作成します。

/var/lib/iscsi/nodes内のノードレコード

ターゲットにログインすると、iscsiadm はこのファイル内の設定を使用します。

/var/lib/iscsi/discovery_typeにおける検出レコード

同じ宛先に対して検出を実行すると、iscsiadm はこのファイル内の設定を使用します。

検出に異なる設定を使用する前に、現在の検出レコード（例： /var/lib/iscsi/discovery_type）を最初に削除します。これを行うには、以下のコマンドを使用します。[5]

```
# iscsiadm -m discovery -t discovery_type -p target_IP:port -o delete
```

ここで、discovery_type は sendtargets、isns、または fw のいずれかになります。

さまざまなタイプの検出の詳細については、iscsiadm(8) man ページの 『DISCOVERY TYPES』 セクションを参照してください。

検出レコードの設定を再設定するには、次の 2 つの方法があります。

- 検出を実行する前に、/etc/iscsi/iscsid.conf ファイルを直接編集します。検出設定では、接頭辞 discovery を使用します。表示するには、以下を実行します。

```
# iscsiadm -m discovery -t discovery_type -p target_IP:port
```

- または、iscsiadm を使用して、以下のように検出レコード設定を直接変更することもできます。

```
# iscsiadm -m discovery -t discovery_type -p target_IP:port -o update -n setting -v %value
```

利用可能なsettingオプションと、それぞれに有効なvalueオプションの詳細は、man ページの iscsiadm(8) を参照してください。

検出設定を設定した後、新しいターゲットを検出しようとすると、それ以降は新しい設定が使用されます。新しい iSCSI ターゲットをスキャンする方法は、「iSCSI 相互接続のスキャン」 を参照してください。

iSCSI ターゲット検出の設定に関する詳細は、iscsiadm および iscsid の man ページを参照してください。/etc/iscsi/iscsid.conf ファイルには、適切な設定構文の例も含まれています。

25.14. ISCSI オフロードおよびインターフェイスバインディングの設定

本章では、ソフトウェア iSCSI を使用する場合に、セッションを NIC ポートにバインドするために iSCSI インターフェイスを設定する方法を説明します。また、オフロードをサポートするネットワークデバイスで使用するインターフェイスを設定する方法についても説明します。

ネットワークサブシステムは、iSCSI インターフェイスがバインドに使用するパス/NIC を決定するように設定できます。たとえば、ポータルと NIC が別のサブネットに設定されている場合は、iSCSI インターフェイスをバインド用に手動で設定する必要はありません。

バインディング用に iSCSI インターフェイスを設定する前に、以下のコマンドを実行します。

```
$ ping -I ethX target_IP
```

ping に失敗すると、NIC にセッションをバインドできなくなります。その場合は、まずネットワーク設定を確認してください。

25.14.1. 利用可能な iface 設定の表示

iSCSI オフロードおよびインターフェイスバインディングは、以下の iSCSI イニシエーターの実装でサポートされています。

ソフトウェア iSCSI

このスタックは、セッションごとに iSCSI ホストインスタンス(scsi_host)を割り当て、セッションごとに単一の接続を使用します。その結果、/sys/class_scsi_host および /proc/scsi は、ログインしている各接続/セッションの scsi_host を報告します。

iSCSI のオフロード

このスタックは、PCI デバイスごとに scsi_host を割り当てます。そのため、ホストバスアダプター上の各ポートは異なる PCI デバイスとして表示され、HBA ポートごとに異なる scsi_host が表示されます。

両方のタイプのイニシエーター実装を管理するために、iscsiadm は iface 構造を使用します。この構造では、セッションのバインドに使用される各 HBA ポート、ソフトウェア iSCSI、またはネットワークデバイス(ethX)の /var/lib/iscsi/ iface s に iface 設定を入力する必要があります。

利用可能な iface 設定を表示するには、iscsiadm -m iface を実行します。これにより、iface 情報が次の形式で表示されます。

```
iface_name transport_name,hardware_address,ip_address,net_ifacename,initiator_name
```

各値/設定の説明は、以下の表を参照してください。

表25.2 iface 設定

設定	説明
iface_name	iface 設定名。
transport_name	ドライバーの名前

設定	説明
hardware_address	MAC アドレス
ip_address	このポートに使用する IP アドレス
net_iface_name	ソフトウェア iSCSI セッションの vlan または alias バインディングに使用される名前。iSCSI オフロードの場合、この値は再起動後も持続しないため、net_iface_name は < empty > になります。
initiator_name	この設定は、イニシエーターのデフォルト名を上書きするために使用されます。これは、/etc/iscsi/initiatorname.iscsiで定義されます。

例25.6 iscsiadm -m iface コマンドの出力例

以下は、iscsiadm -m iface コマンドの出力例です。

```
iface0 qla4xxx,00:c0:dd:08:63:e8,20.15.0.7,default,iqn.2005-06.com.redhat:madmax
iface1 qla4xxx,00:c0:dd:08:63:ea,20.15.0.9,default,iqn.2005-06.com.redhat:madmax
```

ソフトウェア iSCSI の場合、iface 設定には一意の名前(65 文字未満)が必要です。オフロードをサポートするネットワークデバイスの iface_name は、transport_name.hardware_name の形式で表示されます。

例25.7 Chelsio ネットワークカードを使用した iscsiadm -m iface 出力

たとえば、Chelsio ネットワークカードを使用したシステムでの iscsiadm -m iface の出力例は、以下のように表示されます。

```
default tcp,<empty>,<empty>,<empty>,<empty>
iser iser,<empty>,<empty>,<empty>,<empty>
cxgb3i.00:07:43:05:97:07 cxgb3i,00:07:43:05:97:07,<empty>,<empty>,<empty>
```

特定の iface 設定の設定をよりわかりやすい方法で表示することもできます。これを行うには、-I iface_name オプションを使用します。設定は以下の形式で表示されます。

-

```
iface.setting = value
```

例25.8 Chelsio コンバージドネットワークアダプターでの iface 設定の使用

前の例を使用すると、同じ Chelsio コンバージドネットワークアダプター(iscsiadm -m iface -I cxgb3i.00:07:43:05:97:07)の iface設定は以下のように表示されます。

```
# BEGIN RECORD 2.0-871
iface.iscsi_ifacename = cxgb3i.00:07:43:05:97:07
iface.net_ifacename = <empty>
iface.ipaddress = <empty>
iface.hwaddress = 00:07:43:05:97:07
iface.transport_name = cxgb3i
iface.initiatorname = <empty>
# END RECORD
```

25.14.2. ソフトウェア iSCSI 用の iface の設定

前述のように、セッションのバインドに使用される各ネットワークオブジェクトに iface 設定が必要です。

前

ソフトウェア iSCSI の iface 設定を作成するには、次のコマンドを実行します。

```
# iscsiadm -m iface -I iface_name --op=new
```

これにより、指定した iface _name を持つ新しい空 の iface 設定が作成されます。既存の iface 設定にすでに同じ iface_name がある場合は、空の新しい設定によって上書きされます。

特定の iface 設定を設定するには、以下のコマンドを使用します。

```
# iscsiadm -m iface -I iface_name --op=update -n iface.setting -v hw_address
```

例25.9 iface0の MAC アドレスの設定

たとえば、iface0 の MAC アドレス(hardware_address)を 00:0F: 1F:92:6B:BF に設定するには、次のコマンドを実行します。

```
# iscsiadm -m iface -I iface0 --op=update -n iface.hwaddress -v 00:0F:1F:92:6B:BF
```

警告

デフォルト または iser を iface 名として使用しないでください。どちらの文字列も、後方互換性のために iscsiadm で使用される特別な値です。手動で作成された default または iser という名前の iface 設定は、後方互換性を無効にします。

25.14.3. iSCSI オフロード用の iface の設定

デフォルトでは、iscsiadm は、ポートごとに iface 設定を作成します。利用可能な iface 設定を表示するには、ソフトウェア iSCSI の場合と同じコマンドである iscsiadm -m iface を使用します。

iSCSI オフロードにネットワークカードの iface を使用する前に、まずオフロードインターフェイスの iface.ipaddress 値を、インターフェイスが使用するイニシエーター IP アドレスに設定します。

- be2iscsi ドライバーを使用するデバイスの場合、IP アドレスは BIOS 設定画面で設定されます。
- 他のすべてのデバイスで、iface の IP アドレスを設定するには、次のコマンドを実行します。

```
# iscsiadm -m iface -I iface_name -o update -n iface.ipaddress -v initiator_ip_address
```

例25.10 Chelsio カードの iface IP アドレスの設定

たとえば、iface 名が cxgb3i .00:07:43:05:97:07 のカードを使用する場合は、iface IP アドレスを 20.15.0.66 に設定するには、次のコマンドを実行します。

```
# iscsiadm -m iface -I cxgb3i.00:07:43:05:97:07 -o update -n iface.ipaddress -v 20.15.0.66
```

25.14.4. iface のポータルへのバインド/バインド解除

iscsiadm を使用して相互接続をスキャンするたびに、最初に /var/lib/iscsi/ifaces の各 iface 設定

の iface.transport 設定を確認します。その後、iscsiadm ユーティリティーは、検出されたポータルを、 iface.transport が tcp である任意の iface にバインドします。

この動作は、互換性の理由で実装されました。これを上書きするには、以下のように -I iface_name を使用して iface にバインドするポータルを指定します。

```
# iscsiadm -m discovery -t st -p target_IP:port -I iface_name -P 1
[5]
```

デフォルトでは、iscsiadm ユーティリティーは、オフロードを使用する iface 設定にポータルを自動的にバインドしません。これは、このような iface 設定では iface.transport が tcp に設定されないためです。そのため、iface 設定は、検出されたポータルに手動でバインドする必要があります。

ポータルが既存の iface にバインドされないようにすることもできます。これを行うには、以下のように default を iface_name として使用します。

```
# iscsiadm -m discovery -t st -p IP:port -I default -P 1
```

ターゲットと iface 間のバインディングを削除するには、次のコマンドを実行します。

```
# iscsiadm -m node -targetname proper_target_name -I iface0 --op=delete[6]
```

特定の iface のバインディングをすべて削除するには、次のコマンドを実行します。

```
# iscsiadm -m node -I iface_name --op=delete
```

特定のポータル (Equalogic ターゲットなど) のバインディングを削除するには、以下のコマンドを実行します。

```
# iscsiadm -m node -p IP:port -I iface_name --op=delete
```

注記

/var/lib/iscsi/ iface に iface 設定が定義されておらず、-I オプションが使用されていない場合、iscsiadm はネットワークサブシステムが特定のポータルが使用するデバイスを決定できるようにします。

25.15. iSCSI 相互接続のスキャン

iSCSI の場合、ターゲットが、新しいストレージが追加されたことを示す iSCSI 非同期イベントを送信すると、スキャンが自動的に行われます。

ただし、ターゲットが iSCSI 非同期イベントを送信しない場合は、iscsiadm ユーティリティーを使用してターゲットを手動でスキャンする必要があります。ただし、その前に、まず適切な --targetname と --portal の値を取得する必要があります。デバイスモデルがターゲットごとに 1 つの論理ユニットとポータルのみに対応している場合は、以下のように iscsiadm を使用して、sendtargets コマンドをホストに発行します。

```
# iscsiadm -m discovery -t sendtargets -p target_IP:port
[5]
```

出力は以下の形式で表示されます。

```
target_IP:port,target_portal_group_tag proper_target_name
```

例25.11 iscsiadm を使用した sendtargets コマンドの発行

たとえば、proper_target_name が iqn.1992-08.com.netapp:sn.33615311 で、target_IP:port が 10.15.85.19:3260 のターゲットでは、出力は以下のようになります。

```
10.15.84.19:3260,2 iqn.1992-08.com.netapp:sn.33615311
10.15.85.19:3260,3 iqn.1992-08.com.netapp:sn.33615311
```

この例では、ターゲットには 2 つのポータルがあり、それぞれ target_ip:portは 10.15.84.19:3260 および 10.15 .85.19:3260 を使用します。

各セッションに使用される iface 設定を確認するには、-P 1 オプションを追加します。このオプションは、以下のように、セッション情報もツリー形式で出力します。

```
Target: proper_target_name
   Portal: target_IP:port,target_portal_group_tag
      Iface Name: iface_name
```

例25.12 iface 設定の表示

たとえば、iscsiadm -m discovery -t sendtargets -p 10.15.85.19:3260 -P 1 を使用すると、出

力は以下のようになります。

```
Target: iqn.1992-08.com.netapp:sn.33615311
   Portal: 10.15.84.19:3260,2
      Iface Name: iface2
   Portal: 10.15.85.19:3260,3
      Iface Name: iface2
```

つまり、ターゲット iqn.1992-08.com.netapp:sn.33615311 は、iface 設定として iface2 を使用します。

デバイスモデルによっては、1 つのターゲットに複数の論理ユニットとポータルが存在する場合があります。この場合、最初に sendtargets コマンドをホストに発行して、ターゲット上の新しいポータルを検索します。次に、以下を使用して既存のセッションを再スキャンします。

```
# iscsiadm -m session --rescan
```

以下のようにセッションの SID 値を指定して、特定のセッションを再スキャンすることもできます。

```
# iscsiadm -m session -r SID --rescan[7]
```

デバイスが複数のターゲットに対応している場合は、ホストに sendtargets コマンドを実行して、各 ターゲットの新しいポータルを見つける必要があります。--rescan オプションを使用して、既存のセッションを再スキャンし、既存のセッションで新しい論理ユニットを検出します。

重要

--targetname および --portal の値を取得するために使用される sendtargets コマンドは、/var/lib/iscsi/nodes データベースの内容を上書きします。このデータベースは、/etc/iscsi/iscsid.conf の設定を使用して再入力されます。ただし、セッションが現在ログインしており、使用中の場合は発生しません。

新しいターゲット/ポータルを安全に追加したり、古いターゲットを削除するには、それぞれ -o new オプションまたは -o delete オプションを使用します。たとえば、/var/lib/iscsi/nodes を上書きせずに新しいターゲット/ポータルを追加するには、以下のコマンドを使用します。

```
iscsiadm -m discovery -t st -p target_IP -o new
```

検出中にターゲットが表示されなかった /var/lib/iscsi/nodes エントリーを削除するには、次のコマンドを実行します。

```
iscsiadm -m discovery -t st -p target_IP -o delete
```

以下のように、両方のタスクを同時に実行することもできます。

```
iscsiadm -m discovery -t st -p target_IP -o delete -o new
```

sendtargets コマンドは、以下の出力を表示します。

```
ip:port,target_portal_group_tag proper_target_name
```

例25.13 sendtargets コマンドの出力

たとえば、ターゲットが 1 つ、論理ユニット、およびポータルのデバイスが target_name として equallogic-iscsi1 の場合、出力は次のようになります。

```
10.16.41.155:3260,0 iqn.2001-05.com.equallogic:6-8a0900-ac3fe0101-63aff113e344a4a2-dl585-03-1
```

proper_target_name および ip:port,target_portal_group_tag は、「iSCSI API」 の同じ名前の値と同じであることに注意してください。

この時点で、iSCSI デバイスを手動でスキャンするために必要な適切な --targetname と --portal の値が得られます。これには以下のコマンドを実行します。

```
# iscsiadm --mode node --targetname proper_target_name --portal
ip:port,target_portal_group_tag \ --login
[8]
```

例25.14 完全な iscsiadm コマンド

前の例(proper_target_name が equallogic-iscsi1の場合)を使用すると、完全なコマンドは次のようになります。

```
# iscsiadm --mode node --targetname  \ iqn.2001-05.com.equallogic:6-8a0900-ac3fe0101-
63aff113e344a4a2-dl585-03-1  \ --portal 10.16.41.155:3260,0 --login[8]
```

25.16. iSCSI ターゲットへのログイン

「iSCSI」 で説明されているように、ターゲットを検出またはログインするには、iSCSI サービスが実行している必要があります。iSCSI サービスを起動するには、次のコマンドを実行します。

```
# systemctl start iscsi
```

このコマンドが実行されると、iSCSI init スクリプトは、node.startup 設定が自動として設定されているターゲットに 自動 的にログインします。これは、すべてのターゲットの node.startup のデフォルト値です。

ターゲットへの自動ログインを防ぐには、node.startup を manual に設定します。これを実行するには、以下のコマンドを実行します。

```
# iscsiadm -m node --targetname proper_target_name -p target_IP:port -o update -n
node.startup -v manual
```

レコード全体を削除すると、自動ログインもできなくなります。これを実行するには、以下を実行します。

```
# iscsiadm -m node --targetname proper_target_name -p target_IP:port -o delete
```

ネットワーク上の iSCSI デバイスからファイルシステムを自動的にマウントするには、_netdev オプションを使用して、/etc/fstab にマウント用のパーティションエントリーを追加します。たとえば、

起動時に iSCSI デバイス sdb を /mount/iscsi に自動的にマウントするには、以下の行を /etc/fstab に追加します。

```
/dev/sdb /mnt/iscsi ext3 _netdev 0 0
```

iSCSI ターゲットに手動でログインするには、次のコマンドを使用します。

```
# iscsiadm -m node --targetname proper_target_name -p target_IP:port -l
```

注記

proper_target_name と target_IP:port は、ターゲットのフルネームと IP アドレス/ポートの組み合わせを参照します。詳細は、「iSCSI API」 および 「iSCSI 相互接続のスキャン」 を参照してください。

25.17. オンライン論理ユニットのサイズの変更

ほとんどの場合、オンライン論理ユニットの完全なサイズ変更には、論理ユニット自体のサイズ変更と、対応するマルチパスデバイスのサイズ変更の反映 (システムでマルチパスが有効になっている場合) の 2 つの作業が含まれます。

オンラインの論理ユニットのサイズを変更するには、まずストレージデバイスのアレイ管理インターフェイスを使用して、論理ユニットのサイズを変更します。この手順はアレイごとに異なります。そのため、詳細については、ストレージアレイベンダーのドキュメントを参照してください。

注記

オンラインのファイルシステムのサイズを変更するには、ファイルシステムがパーティションを設定したデバイスに存在してはいけません。

25.17.1. ファイバーチャネル論理ユニットのサイズ変更

オンラインの論理ユニットサイズを変更したら、論理ユニットを再スキャンして、システムが更新したサイズを検出したことを確認します。ファイバーチャネル論理ユニットにこれを行うには、以下のコマンドを使用します。

```
$ echo 1 > /sys/block/sdX/device/rescan
```

重要

マルチパスを使用するシステムでファイバーチャネル論理ユニットを再スキャンするには、マルチパスを設定した論理ユニットのパスを表す sd デバイス(sd1、sd2 など) ごとに、上記のコマンドを実行します。マルチパス論理ユニットのパスであるデバイスを特定するには、multipath -ll を使用します。次に、サイズ変更している論理ユニットに一致するエントリーを見つけます。サイズ変更される論理ユニットに一致するものを見つけやすくするために、各エントリーの WWID を参照することを推奨します。

25.17.2. iSCSI 論理ユニットのサイズを変更

オンラインの論理ユニットサイズを変更したら、論理ユニットを再スキャンして、システムが更新したサイズを検出したことを確認します。iSCSI デバイスに対してこれを行うには、以下のコマンドを使用します。

```
# iscsiadm -m node --targetname target_name -R
```
[5]

target_name を、デバイスが置かれているターゲットの名前に置き換えます。

注記

次のコマンドを使用して、iSCSI 論理ユニットを再スキャンすることもできます。

```
# iscsiadm -m node -R -I interface
```

interface を、サイズ変更された論理ユニットの対応するインターフェイス名に置き換えます（例： iface0）。このコマンドは、2 つの操作を実行します。

- echo "- - -" > /sys/class/scsi_host/ host / scan コマンドと同じ方法で新しいデバイスをスキャンします(「iSCSI 相互接続のスキャン」 を参照)。
- echo 1 > /sys/block/sdX/device/rescan コマンドと同じ方法で、新規/ 変更された論理ユニットを再スキャンします。このコマンドは、ファイバーチャネルの論理ユニットを再スキャンする際に使用するコマンドと同じであることに注意してください。

25.17.3. マルチパスデバイスのサイズの更新

システムでマルチパスが有効になっている場合は、論理ユニットのサイズ変更を、(論理ユニットの

サイズを変更した後 に) 対応する論理ユニットのマルチパスデバイスに反映させる必要があります。これは、multipathd を介して実行できます。これを行うには、まず service multipathd status を使用して multipathd が実行していることを確認します。multipathd が機能していることを確認したら、次のコマンドを実行します。

```
# multipathd -k"resize map multipath_device"
```

multipath_device 変数は、/dev/mapper 内のデバイスの対応するマルチパスエントリーです。システムでマルチパスの設定方法に応じて、multipath_device は 2 つの形式のいずれかになります。

- X: X はデバイスの対応するエントリーです（例： mpath0）。
- a WWID; for example, 3600508b400105e210000900000490000

サイズを変更した論理ユニットに対応するマルチパスエントリーを特定するには、multipath -ll を実行します。これにより、システムに存在するすべてのマルチパスエントリーと、対応するデバイスのメジャー番号およびマイナー番号のリストが表示されます。

重要な影響

multipath_device にキューに入れられたコマンドがある場合は、multipathd -k"resize map multipath_device " を使用しないでください。つまり、(/etc/multipath.conf で) no_path_retry パラメーターが "queue" に設定され、デバイスへのアクティブなパスがない場合は、このコマンドを使用しないでください。

マルチパスの詳細は、Red Hat Enterprise Linux 7 DM Multipathガイドを参照してください。

25.17.4. オンライン論理ユニットの読み取り/書き込み状態の変更

特定のストレージデバイスでは、デバイスの状態を読み取り/書き込み (R/W) から読み取り専用 (RO) に変更したり、RO から R/W に変更したりする機能をユーザーに提供します。通常、これはストレージデバイスの管理インターフェイスを介して行われます。変更が加えられた場合、オペレーティングシステムはデバイスの状態に関するビューを自動的に更新しません。本章で説明されている手順に従って、オペレーティングシステムに変更を認識させます。

次のコマンドを実行し、XYZ を目的のデバイス指定子に置き換えて、オペレーティングシステムが現在表示しているデバイスの R/W 状態を確認します。

```
# blockdev --getro /dev/sdXYZ
```

以下のコマンドは、Red Hat Enterprise Linux 7 でも使用できます。

```
# cat /sys/block/sdXYZ/ro 1 = read-only 0 = read-write
```

マルチパスを使用する場合は、multipath -ll コマンドの出力の 2 行目の ro または rw フィールドを参照してください。以下に例を示します。

```
36001438005deb4710000500000640000 dm-8 GZ,GZ500
[size=20G][features=0][hwhandler=0][ro]
\_ round-robin 0 [prio=200][active]
 \_ 6:0:4:1  sdax 67:16  [active][ready]
 \_ 6:0:5:1  sday 67:32  [active][ready]
\_ round-robin 0 [prio=40][enabled]
 \_ 6:0:6:1  sdaz 67:48  [active][ready]
 \_ 6:0:7:1  sdba 67:64  [active][ready]
```

R/W の状態を変更するには、以下の手順に従います。

手順25.14 R/W の状態の変更

1. デバイスを RO から R/W に移動するには、手順 2 を参照してください。

   デバイスを R/W から RO に移動するには、それ以上の書き込みが発行されないようにします。これを行うには、アプリケーションを停止するか、適切なアプリケーション固有のアクションを実行します。

   次のコマンドを使用して、すべての未処理の書き込み I/O が完了していることを確認します。

   ```
   # blockdev --flushbufs /dev/device
   ```

   device を目的のデザイン者に置き換えます。デバイスマッパーマルチパスの場合、これは dev/mapper のデバイスのエントリーになります。たとえば、/dev/mapper/mpath3 です。

2. ストレージデバイスの管理インターフェイスを使用して、論理ユニットの状態を R/W から RO に、または RO から R/W に変更します。この手順は、アレイごとに異なります。詳細は、

該当するストレージアレイベンダーのドキュメントを参照してください。

3.
デバイスの R/W 状態に関するオペレーティングシステムのビューを更新するには、デバイスの再スキャンを実行します。デバイスマッパーマルチパスを使用している場合は、デバイスへのパスごとにこの再スキャンを実行してから、マルチパスにデバイスマップをリロードするように指示するコマンドを発行します。

この手順は、「論理ユニットの再スキャン」 で詳細に説明されています。

25.17.4.1. 論理ユニットの再スキャン

「オンライン論理ユニットの読み取り/書き込み状態の変更」 の説明に従って、オンライン論理ユニットの読み取り/書き込み状態を変更したら、論理ユニットを再スキャンして、システムが以下のコマンドで更新した状態を検出します。

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

マルチパスを使用するシステムで論理ユニットを再スキャンするには、マルチパスを設定した論理ユニットのパスを表す sd デバイスごとに、上記のコマンドを実行します。たとえば、sd1、sd2、およびその他のすべての sd デバイスでコマンドを実行します。マルチパスユニットのパスとなるデバイスを特定するには、multipath -11 を使用して、変更する論理ユニットに一致するエントリーを見つけます。

例25.15 multipath -11 コマンドの使用

たとえば、上記の multipath -11 は、WWID 36001438005deb4710000500000640000 の LUN のパスを示しています。この場合は、以下のコマンドを実行します。

```
# echo 1 > /sys/block/sdax/device/rescan
# echo 1 > /sys/block/sday/device/rescan
# echo 1 > /sys/block/sdaz/device/rescan
# echo 1 > /sys/block/sdba/device/rescan
```

25.17.4.2. マルチパスデバイスの R/W 状態の更新

マルチパスが有効になっている場合、論理ユニットを再スキャンした後、その状態の変更を論理ユニットの対応するマルチパスドライブに反映する必要があります。これを行うには、以下のコマンドでマルチパスデバイスマップを再読み込みします。

```
# multipath -r
```

その後、multipath -11 コマンドを使用して変更を確認できます。

25.17.4.3. Documentation

詳細は、Red Hat ナレッジベースを参照してください。これにアクセスするには、https://www.redhat.com/wapps/sso/login.html?redirect=https://access.redhat.com/knowledge/ に移動してログインします。その後、https://access.redhat.com/kb/docs/DOC-32850 の記事を参照してください。

25.18. RESCAN-SCSI-BUS.SH を使用した論理ユニットの追加/削除

sg3_utils パッケージは、rescan-scsi-bus.sh スクリプトを提供します。このスクリプトは、（デバイスをシステムに追加した後に）必要に応じてホストの論理ユニット設定を自動的に更新できます。rescan-scsi-bus.sh スクリプトは、サポートされているデバイスで issue_lip を実行することもできます。このスクリプトの使用方法は、rescan-scsi-bus.sh --help を参照してください。

sg3_utils パッケージをインストールするには、yum install sg3_utils を実行します。

rescan-scsi-bus.sh の既知の問題

rescan-scsi-bus.sh スクリプトを使用する場合は、以下の既知の問題を書き留めておきます。

- rescan-scsi-bus.sh が正しく機能するには、LUN0 が最初にマッピングされた論理ユニットである必要があります。rescan-scsi-bus.sh は、最初にマップされた論理ユニットが LUN0 の場合にのみ検出できます。--nooptscan オプションを使用していても、最初にマップされた論理ユニットを検出しない限り、rescan-scsi-bus.sh は、他の論理ユニットをスキャンできません。
- 競合状態では、論理ユニットが最初にマッピングされている場合に rescan-scsi-bus.sh を 2 回実行する必要があります。最初のスキャン中に、rescan-scsi-bus.sh は LUN0 のみを追加します。その他の論理ユニットはすべて 2 番目のスキャンに追加されます。
- rescan-scsi-bus.sh スクリプトのバグにより、--remove オプションが使用されると、論理ユニットサイズの変更を認識するための機能が誤って実行されます。
- rescan-scsi-bus.sh スクリプトは、ISCSI 論理ユニットの削除を認識しません。

25.19. リンクロス動作の変更

本セクションでは、ファイバーチャネルまたは iSCSI プロトコルのいずれかを使用するデバイスで、リンクロス動作を変更する方法を説明します。

25.19.1. ファイバーチャネル

ドライバーがトランスポートの dev_loss_tmo コールバックを実装している場合、トランスポートの問題が検出されるとリンクを介したデバイスへのアクセス試行がブロックされます。デバイスがブロックされているかどうかを確認するには、次のコマンドを実行します。

```
$ cat /sys/block/device/device/state
```

このコマンドは、デバイスが ブロック されている場合に blocked を返します。デバイスが正常に動作している場合、このコマンドは の 実行 を返します。

手順25.15 リモートポートの状態の判断

1. リモートポートの状態を判断するには、次のコマンドを実行します。

```
$ cat
/sys/class/fc_remote_port/rport-H:B:R/port_state
```

2. このコマンドは、リモートポート（およびそのポートからアクセスされるデバイス）がブロックされると、Blocked を返します。リモートポートが正常に動作している場合、コマンドは Online を返します。

3. dev_loss_tmo 秒以内に問題が解決されない場合、rport およびデバイスはブロックされず、そのデバイスで実行しているすべての I/O （およびそのデバイスに送信される新しい I/O）は失敗します。

手順25.16 dev_loss_tmoの変更

- dev_loss_tmo 値を変更するには、目的の値を ファイルに echo します。たとえば、dev_loss_tmo を 30 秒に設定するには、以下のコマンドを実行します。

```
$ echo 30 >
/sys/class/fc_remote_port/rport-H:B:R/dev_loss_tmo
```

dev_loss_tmo の詳細は、「ファイバーチャネル API」 を参照してください。

リンクの損失が dev_loss_tmo を超えると、scsi_device デバイスおよび sdN デバイスが削除されます。通常、ファイバーチャネルクラスはデバイスをそのまま残します。つまり、/dev/sd x は /dev/sdx のままになります。これは、ターゲットのバインディングがファイバーチャネルドライバーにより保存されるため、ターゲットポートが戻ると、SCSI アドレスが忠実に再作成されます。ただし、これは保証されません。LUN のストレージ内ボックス設定に追加の変更がない場合に限り、sdx が復元されます。

25.19.2. dm-multipath での iSCSI 設定

dm-multipath が実装されている場合は、iSCSI タイマーを設定して、すぐにマルチパスレイヤーにコマンドを取得することが推奨されます。これを設定するには、/etc/multipath.conf の device { の下に以下の行をネストします。

```
features "1 queue_if_no_path"
```

これにより、dm-multipath レイヤーですべてのパスが失敗した場合に、I/O エラーが再試行され、キューに入れられます。

SAN で問題をより的確に監視するには、iSCSI タイマーの調整が必要になる場合があります。設定可能な iSCSI タイマーは、 NOP-Out 間隔/タイムアウト と replacement_timeout です。これについては、以下のセクションで説明します。

25.19.2.1. NOP-Out 間隔/タイムアウト

SAN の問題を監視するため、iSCSI レイヤーは各ターゲットに NOP-Out 要求を送信します。NOP-Out 要求がタイムアウトすると、iSCSI レイヤーは、実行中のコマンドに失敗し、可能な場合はそれらのコマンドを再キューイングするように SCSI レイヤーに指示することで応答します。

dm-multipath を使用すると、SCSI レイヤーは実行中のコマンドに失敗し、マルチパスレイヤーに延期します。次に、マルチパスレイヤーは、これらのコマンドを別のパスで再試行します。dm-multipath が使用されていない 場合、これらのコマンドは、すべて失敗する前に 5 回再試行されます。

NOP-Out 要求の間隔は、デフォルトで 5 秒です。これを調整するには、/etc/iscsi/iscsid.conf を開き、以下の行を編集します。

```
node.conn[0].timeo.noop_out_interval = [interval value]
```

設定されると、iSCSI レイヤーは、[interval value] 秒ごとに、各ターゲットに NOP-Out 要求を送信します。

デフォルトでは、NOP-Out 要求は 5 秒でタイムアウトします。[9].これを調整するには、/etc/iscsi/iscsid.conf を開き、以下の行を編集します。

```
node.conn[0].timeo.noop_out_timeout = [timeout value]
```

これにより、iSCSI レイヤーは [timeout value] 秒後に NOP-Out 要求をタイムアウトするように設定されます。

SCSI エラーハンドラー

SCSI エラーハンドラーが実行されている場合、パスで NOP-Out 要求がタイムアウトしても、パスでのコマンドの実行がすぐに失敗することはありません。代わりに、これらのコマンドは replacement_timeout 秒後 に失敗します。replacement_timeout の詳細は、「replacement_timeout」 を参照してください。

SCSI エラーハンドラーが実行されているかどうかを確認するには、次のコマンドを実行します。

```
# iscsiadm -m session -P 3
```

25.19.2.2. replacement_timeout

replacement_timeout は、コマンドが失敗する前に、タイムアウトしたパス/セッションが再確立されるまで iSCSI レイヤーが待機する時間を制御します。replacement_timeout のデフォルト値は 120 秒です。

replacement_timeout を調整するには、/etc/iscsi/iscsid.conf を開き、以下の行を編集します。

```
node.session.timeo.replacement_timeout = [replacement_timeout]
```

/etc/multipath.conf の 1 queue_if_no_path オプションは、iSCSI タイマーを設定して、マルチパス層に直ちにコマンドを推測します(「dm-multipath での iSCSI 設定」を参照)。この設定により、I/O エラーがアプリケーションに伝搬されなくなります。このため、replacement_timeout を 15-20 秒に設定することができます。

下限に replacement_timeout を設定すると、iSCSI レイヤーが失敗したパス/セッションの再確立を試みる間(NOP-Out タイムアウトの場合)、I/O は新しいパスにすばやく送信され、実行されます。すべてのパスがタイムアウトすると、マルチパスおよびデバイスマッパー層は、/etc/iscsi/iscsid.conf ではなく、/etc/multipath.conf の設定に基づいて I/O を内部でキューに入れます。

重要な影響

フェイルオーバーの速度とセキュリティーのどちらが検討されているかにかかわらず、replacement_timeout に推奨される値は他の要因によって異なります。これらの要因には、ネットワーク、ターゲット、およびシステムのワークロードが含まれます。そのため、ミッションクリティカルなシステムに適用する前に、新しい設定を replacements_timeout に完全にテストすることが推奨されます。

iSCSI および DM Multipath のオーバーライド

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

- replacement_timeout 設定オプションは、すべての iSCSI デバイスの recovery_tmo 値をグローバルに上書きします。

- DM Multipath が管理するすべての iSCSI デバイスについては、DM Multipath の fast_io_fail_tmo オプションがグローバルに recovery_tmo 値を上書きします。DM Multipath の fast_io_fail_tmo オプションは、ファイバーチャネルデバイスの fast_io_fail_tmo オプションを上書きします。

  DM Multipath fast_io_fail_tmo オプションは replacement_timeout よりも優先されます。Red Hat では、replacement_timeout を使用して、DM Multipath が管理するデバイスで recovery_tmo を上書きすることは推奨していません。これは、multipathd サービスが再読み込みされると、DM Multipath が常に recovery_tmo をリセットするためです。

25.19.3. iSCSI ルート

iSCSI ディスクから直接 root パーティションにアクセスする場合は、iSCSI レイヤーがパス/セッションの再確立を試行する機会が数回あるように、iSCSI タイマーを設定する必要があります。さらに、コマンドを SCSI レイヤーにすばやく再キューイングしないでください。これは、dm-multipath の実装時に行うべき動作とは逆になります。

まず、NOP-Outs を無効にする必要があります。これを行うには、NOP-Out の間隔とタイムアウトの両方をゼロに設定します。これを設定するには、/etc/iscsi/iscsid.conf を開き、以下のように編集します。

-

```
node.conn[0].timeo.noop_out_interval = 0
node.conn[0].timeo.noop_out_timeout = 0
```

これに合わせて、replacement_timeout を高い数値に設定する必要があります。これにより、パスやセッションが再確立されるのをシステムが長い間待機するようになります。replacement_timeout を調整するには、/etc/iscsi/iscsid.conf を開き、以下の行を編集します。

```
node.session.timeo.replacement_timeout = replacement_timeout
```

/etc/iscsi/iscsid.conf を設定したら、影響を受けるストレージのリ検出を実行する必要があります。これにより、システムが /etc/iscsi/iscsid.conf の新しい値を読み込み、使用できるようになります。iSCSI デバイスの検出方法に関する詳細は、「iSCSI 相互接続のスキャン」 を参照してください。

特定セッションのタイムアウトの設定

特定のセッションのタイムアウトを設定し、(/etc/iscsi/iscsid.confを使用する代わりに)永続化しないようにすることもできます。これを行うには、次のコマンドを実行します (必要に応じて変数を置き換えます)。

```
# iscsiadm -m node -T target_name -p target_IP:port -o update -n
node.session.timeo.replacement_timeout -v $timeout_value
```

重要な影響

ここで説明する設定は、root パーティションアクセスを含む iSCSI セッション用として推奨されます。他のタイプのストレージ(dm-multipathを使用するシステム)へのアクセスを含む iSCSI セッションについては、「dm-multipath での iSCSI 設定」 を参照してください。

25.20. SCSI コマンドタイマーおよびデバイスステータスの制御

Linux SCSI レイヤーは、各コマンドにタイマーを設定します。このタイマーが時間切れになると、SCSI レイヤーはホストバスアダプター (HBA) を静止し、すべての未処理のコマンドがタイムアウトまたは完了するのを待ちます。その後、SCSI レイヤーはドライバーのエラーハンドラーをアクティブにします。

エラーハンドラーがトリガーされると、(1 つが正常に実行されるまで) 以下の操作が順番に試行されます。

1. コマンドの中止

2. デバイスのリセット

3. バスのリセット

4. ホストをリセットします。

これらの操作がすべて失敗すると、デバイスは **offline** 状態に設定されます。これが発生すると、問題が修正され、デバイスが を 実行 するまで、そのデバイスへのすべての I/O が失敗します。

ただし、デバイスがファイバーチャネルプロトコルを使用し、**rport** がブロックされている場合は、プロセスが異なります。このような場合、ドライバーは、**rport** が再びオンラインになるまで数秒待機してからエラーハンドラーを有効にします。これにより、一時的なトランスポートの問題によりデバイスがオフラインにならないようにします。

デバイスの状態

デバイスの状態を表示するには、次のコマンドを実行します。

```
$ cat /sys/block/device-name/device/state
```

デバイスを **running** 状態に設定するには、以下を使用します。

```
# echo running > /sys/block/device-name/device/state
```

コマンドタイマー

コマンドタイマーを制御するには、**/sys/block/device-name/device/timeout** ファイルを変更します。

```
# echo value > /sys/block/device-name/device/timeout
```

コマンドの **value** を、実装するタイムアウト値（秒単位）に置き換えます。

25.21. オンラインストレージ設定のトラブルシューティング

本セクションでは、オンラインストレージの再設定中にユーザーに発生する一般的な問題へのソリューションを提供します。

論理ユニットの削除ステータスは、ホストには反映されません。

設定されているファイラーで論理ユニットを削除しても、その変更はホストには反映されません。このような場合、論理ユニットが 古くなっ たため、dm-multipath を使用すると、lvm コマンドは無期限にハングします。

これを回避するには、以下の手順を行います。

手順25.17 古い論理ユニットの操作

1. /etc/lvm/cache/.cache 内のどの mpath リンクエントリーが古い論理ユニットに固有であるかを判別します。これを実行するには、以下のコマンドを実行します。

```
$ ls -l /dev/mpath | grep stale-logical-unit
```

例25.16 特定の mpath リンクエントリーの決定

たとえば、stale-logical-unit が 3600d0230003414f30000203a7bc41a00 の場合、以下の結果が表示される場合があります。

```
lrwxrwxrwx 1 root root 7 Aug  2 10:33 /3600d0230003414f30000203a7bc41a00 ->
../dm-4
lrwxrwxrwx 1 root root 7 Aug  2 10:33 /3600d0230003414f30000203a7bc41a00p1 -
> ../dm-5
```

つまり、3600d0230003414f30000203a7bc41a00 は、dm- 4 と dm -5 の 2 つの mpath リンクにマップされ ます。

2. 次に、/etc/lvm/cache/.cache を開きます。stale-logical-unit と、stale-logical-unit がマップする mpath リンクを含むすべての行を削除します。

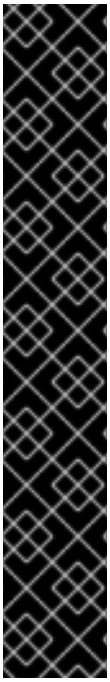
例25.17 関連行の削除

前の手順と同じ例を使用する場合、削除する必要のある行は次のとおりです。

```
/dev/dm-4
```

```
/dev/dm-5
/dev/mapper/3600d0230003414f30000203a7bc41a00
/dev/mapper/3600d0230003414f30000203a7bc41a00p1
/dev/mpath/3600d0230003414f30000203a7bc41a00
/dev/mpath/3600d0230003414f30000203a7bc41a00p1
```

25.22. EH_DEADLINE を使用したエラー復旧の最大時間の設定



重要

ほとんどの場合、eh_deadline パラメーターを有効にする必要はありません。eh_deadline パラメーターは、ファイバーチャネルスイッチとターゲットポート間でリンクの損失が発生した場合や、Host Bus Adapter (HBA)が Registered State Change Notifications (RSCN)を受信しない場合など、特定のシナリオで役立ちます。このような場合、I/O 要求やエラーからの復旧コマンドは、エラーに遭遇することなく、すべてタイムアウトになります。この環境で eh_deadline を設定すると、復旧時間に上限が設定されます。これにより、マルチパスによって、失敗した I/O を別のパスで再試行できます。

ただし、RSCN が有効になっている場合、HBA はリンクが利用不可になるか、あるいはその両方に登録されません。I/O およびエラー回復コマンドがすぐに失敗するため、eh_deadline 機能は追加の利点を提供しません。これにより、マルチパスを再試行できます。

SCSI ホストオブジェクトの eh_deadline パラメーターを使用すると、HBA 全体を停止およびリセットする前に、SCSI エラー処理メカニズムがエラーリカバリーを実行しようとする最大時間を設定できます。

eh_deadline の値は秒単位で指定されます。デフォルト設定は off で、時間制限を無効にし、すべてのエラー回復を可能にします。sysfs の使用に加えて、scsi_mod.eh_deadline カーネルパラメーターを使用すると、すべての SCSI HBA にデフォルト値を設定できます。

eh_deadline の期限が切れると HBA がリセットされることに注意してください。これは、障害が発生したものだけでなく、HBA 上のすべてのターゲットパスに影響します。結果として、他の理由で冗長パスの一部が使用できない場合、I/O エラーが発生する可能性があります。eh_deadline は、すべてのターゲットに完全な冗長マルチパス設定がある場合にのみ有効にします。

---

[5]
target_IP 変数および port 変数は、それぞれターゲット/ポータルの IP アドレスとポートの組み合わせを参照します。詳細は、「iSCSI API」 および 「iSCSI 相互接続のスキャン」 を参照してくださ

い。

[6]
Refer to 「iSCSI 相互接続のスキャン」 for information on proper_target_name.

[7]
For information on how to retrieve a session's SID value, refer to 「iSCSI API」.

[8]
This is a single command split into multiple lines, to accommodate printed and PDF versions of this document. All concatenated lines — preceded by the backslash (\) — should be treated as one command, sans backslashes.

[9]
Red Hat Enterprise Linux 5.4 以前では、デフォルトの NOP-Out 要求のタイムアウトは 15 秒でした。

第26章 DEVICE MAPPER MULTIPATHING (DM MULTIPATH) と仮想マシン用のストレージ

Red Hat Enterprise Linux 7 は、 DM Multipath および 仮想マシン用のストレージ に対応しています。

26.1. 仮想マシン用のストレージ

Red Hat Enterprise Linux 7 は、仮想ストレージ用に以下のファイルシステムまたはオンラインストレージ方式をサポートしています。

- ファイバーチャンネル
- iSCSI
- NFS
- GFS2

Red Hat Enterprise Linux 7 の仮想化では、 libvirt を使用して仮想インスタンスを管理します。libvirt ユーティリティーは、ストレージプール の概念を使用して、仮想化ゲストのストレージを管理します。ストレージプールは、小さなボリュームに分割したり、ゲストに直接割り当てたりできるストレージです。ストレージプールのボリュームは、仮想ゲストに割り当てることができます。利用可能なストレージプールのカテゴリーは以下のとおりです。

ローカルストレージのプール

ローカルストレージには、ホスト、ローカルディレクトリー、直接接続のディスク、および LVM ボリュームグループに直接接続されているストレージデバイス、ファイル、またはディレクトリーが含まれます。

ネットワーク (共有) ストレージプール

ネットワークストレージは、標準プロトコルを使用して、ネットワーク上で共有されるストレージデバイスをカバーします。これには、ファイバーチャネル、iSCSI、NFS、GFS2、および SCSI RDMA プロトコルを使用する共有ストレージデバイスが含まれ、ホスト間で仮想化ゲストを移行するための要件になります。

Red Hat Enterprise Linux がサポートするストレージプールのタイプに関する詳細は、Red Hat Virtualization Deployment and Administration Guideを参照してください。

26.2. DM MULTIPATH

DM Multipath は、サーバーノードとストレージアレイ間で、複数の I/O パスを 1 つのデバイスに設定できる機能です。これらの I/O パスは、個別のケーブル、スイッチ、コントローラーを含むことができる物理的な SAN 接続です。複数の I/O パスを集め、集めたパスで設定される新しいデバイスを 1 つ作成するのがマルチパス機能です。

DM Multipath は、主に以下の理由で使用されます。

冗長性

DM Multipath は、アクティブ/パッシブ設定でフェイルオーバーを提供できます。アクティブ/パッシブ設定では、パスは、I/O には常に半分しか使用されません。I/O パスのいずれかの要素に障害が発生した場合、DM マルチパスは代替パスに切り替わります。

パフォーマンスの向上

DM Multipath を active/active モードに設定すると、I/O はラウンドロビン式でパス全体に分散されます。一部の設定では、DM Multipath は I/O パスの負荷を検出し、負荷を動的に再調整できます。

詳細は、Red Hat DM Multipathガイドを参照してください。

第27章 外部アレイ管理 (LIBSTORAGEMGMT)

Red Hat Enterprise Linux 7 には、libStorageMgmt と呼ばれる新しい外部アレイ管理ライブラリーが同梱されます。

27.1. LIBSTORAGEMGMT の概要

libStorageMgmt ライブラリーは、ストレージアレイに依存しないアプリケーションプログラミングインターフェイス(API)です。開発者は、この API を使用してさまざまなストレージアレイを管理し、ハードウェアアクセラレーション機能を活用できます。

このライブラリーは、他の高レベルの管理ツールおよびアプリケーションのビルディングブロックとして使用されます。エンドシステム管理者は、これをツールとして使用してストレージを手動で管理し、スクリプトを使用してストレージ管理タスクを自動化することもできます。

libStorageMgmt ライブラリーを使用すると、以下の操作を実行できます。

- ストレージプール、ボリューム、アクセスグループ、またはファイルシステムのリストを表示します。
- ボリューム、アクセスグループ、ファイルシステム、または NFS のエクスポートを作成して削除します。
- ボリューム、アクセスグループ、またはイニシエーターへのアクセスを許可および削除します。
- スナップショット、クローン、およびコピーを使用してボリュームを複製します。
- アクセスグループの作成と削除、およびグループのメンバーの編集

CPU、相互接続帯域幅などのサーバーリソースは、操作がすべてアレイで行われるため、使用されません。

libstoragemgmt パッケージは、以下を提供します。

- *クライアントアプリケーションおよびプラグイン開発者向けの安定した C および Python の API。*
- *ライブラリーを使用するコマンドラインインターフェイス(lsmcli)*
- *プラグインを実行するデーモン(lsmd)。*
- *クライアントアプリケーション(sim)のテストを可能にするシミュレータープラグイン。*
- *アレイとインターフェイスするためのプラグインアーキテクチャー。*

警告

このライブラリーとそれに関連するツールには、管理するアレイにあるすべてのデータを破棄する機能があります。実稼働システムで作業する前に、ストレージシミュレータープラグインに対してアプリケーションとスクリプトを開発およびテストして、ロジックエラーを削除することを強く推奨します。可能であれば、実稼働環境にデプロイする前に、実際の非実稼働ハードウェアでアプリケーションとスクリプトをテストすることも強く推奨されます。

Red Hat Enterprise Linux 7 の libStorageMgmt ライブラリーは、REPORTED LUNS DATA HAS CHANGED ユニット注意を処理するデフォルトの udev ルールを追加します。

ストレージ設定の変更が行われると、複数の Unit Attention ASC/ASCQ コードのいずれかが変更を報告します。その後、uevent が生成され、sysfs で自動的に再スキャンされます。

ファイル /lib/udev/rules.d/90-scsi-ua.rules には、カーネルが生成できる他のイベントを列挙できる例が含まれています。

libStorageMgmt ライブラリーは、プラグインアーキテクチャーを使用して、ストレージアレイの相違点に対応します。libStorageMgmt プラグインとその作成方法の詳細は、Red Hat Developer Guide を参照してください。

27.2. LIBSTORAGEMGMT の用語

アレイベンダーやストレージ標準によっては、似たような機能を参照する場合に異なる用語が使用されています。このライブラリーでは、以下の用語を使用します。

ストレージアレイ

ブロックアクセス (FC、FCoE、iSCSI)、またはネットワークアタッチストレージ (NAS) を介したファイルアクセスを提供するストレージシステム。

ボリューム

ストレージエリアネットワーク (SAN) ストレージアレイは、FC、iSCSI、または FCoE などのさまざまなトランスポートを介して、Host Bus Adapter (HBA) にボリュームを公開できます。ホスト OS は、これをブロックデバイスとして扱います。multipath[2] が有効になっている場合、1 つのボリュームを多くのディスクに公開できます。

これは、論理ユニット番号 (LUN)、SNIA 用語を使用した StorageVolume、または仮想ディスクとしても知られています。

プール

ストレージ領域のグループ。ファイルシステムまたはボリュームは、プールから作成できます。プールは、ディスク、ボリューム、およびその他のプールから作成できます。プールには RAID 設定やシンプロビジョニング設定が含まれる場合もあります。

これは、SNIA 用語を使用した StoragePool としても知られています。

スナップショット

特定の時点で、読み取り専用で、領域を効率的に使用するデータのコピー。

これは、読み取り専用スナップショットとも呼ばれます。

Clone

特定の時点で、読み取りおよび書き込み可能で、領域を効率的に使用するデータのコピー。

これは、読み取りおよび書き込み可能なスナップショットとも呼ばれます。

コピー

データの完全なビット単位のコピー。領域全体を占めます。

ミラーリング

継続的に更新されるコピー (同期および非同期)。

アクセスグループ

1 つ以上のストレージボリュームへのアクセスを許可されている iSCSI、FC、および FCoE のイニシエーターのコレクション。これにより、指定のイニシエーターがアクセスできるのはストレージボリュームのみとなります。

これはイニシエーターグループとしても知られています。

アクセス許可

指定したアクセスグループまたはイニシエーターへのボリュームの公開。libStorageMgmt ライブラリーは、現在、特定の論理ユニット番号を選択する機能を備えた LUN マッピングをサポートしていません。libStorageMgmt ライブラリーを使用すると、ストレージアレイは次に割り当てに使用できる LUN を選択できます。SAN からの起動や、256 を超えるボリュームのマスクを設定する場合は、OS、ストレージアレイ、または HBA のドキュメントを必ずお読みください。

アクセス許可は LUN マスキングとも呼ばれます。

システム

ストレージアレイまたは直接接続ストレージ RAID を表します。

ファイルシステム

ネットワークアタッチストレージ (NAS) ストレージアレイは、NFS プロトコルまたは CIFS プロトコルのいずれかを使用して、IP ネットワークを介して、OS をホストするファイルシステムを

公開できます。ホスト OS は、クライアントのオペレーティングシステムに応じて、マウントポイントまたはファイルを含むフォルダーとして扱います。

ディスク

データを保持する物理ディスク。通常、これは RAID 設定を含むプールを作成する場合に使用されます。

これは、SNIA 用語を使用した DiskDrive としても知られています。

イニシエーター

ファイバーチャネル (FC) またはファイバーチャネルオーバーイーサネット (FCoE) では、イニシエーターは World Wide Port Name (WWPN) または World Wide Node Name (WWNN) です。iSCSI では、イニシエーターは iSCSI 修飾名 (IQN) です。NFS または CIFS では、イニシエーターはホストのホスト名または IP アドレスです。

子の依存関係

一部のアレイには、作成元 (親ボリュームまたはファイルシステム) と、子 (スナップショットやクローンなど) の間に暗黙の関係があります。たとえば、依存している子が 1 つ以上ある場合は、親を削除できません。API は、このような関係が存在するかどうかを判断するメソッドと、必要なブロックを複製して依存関係を削除するメソッドを提供します。

27.3. LIBSTORAGEMGMT のインストール

コマンドライン、必要なランタイムライブラリー、およびシミュレータープラグインを使用するために libStorageMgmt をインストールするには、以下のコマンドを使用します。

```
# yum install libstoragemgmt libstoragemgmt-python
```

ライブラリーを使用する C アプリケーションを開発するには、以下のコマンドを使用して libstoragemgmt-devel パッケージをインストールします。

```
# yum install libstoragemgmt-devel
```

ハードウェアアレイで使用するために libStorageMgmt をインストールするには、以下のコマンドで適切なプラグインパッケージを 1 つ以上選択します。

```
# yum install libstoragemgmt-name-plugin
```

利用可能なプラグインは以下のとおりです。

libstoragemgmt-smis-plugin

一般的な SMI-S アレイのサポート。

libstoragemgmt-netapp-plugin

NetApp ファイルの特定のサポート。

libstoragemgmt-nstor-plugin

NexentaStor の特定のサポート。

libstoragemgmt-targetd-plugin

targetd の特定のサポート。

次に、デーモンがインストールされ、次の再起動後の開始時に実行されるように設定されます。再起動せずにすぐに使用するには、デーモンを手動で起動します。

アレイの管理には、プラグインを介したサポートが必要です。ベースインストールパッケージには、さまざまなベンダー向けのオープンソースプラグインが同梱されています。アレイのサポートが改善されると、追加のプラグインパッケージが個別に利用できるようになります。現在、対応しているハードウェアは常に変更され、改善されています。

libStorageMgmt デーモン(lsmd)は、システムの標準サービスのように動作します。

libStorageMgmt サービスのステータスを確認するには、以下を実行します。

```
# systemctl status libstoragemgmt
```

サービスを停止するには、以下を実行します。

```
# systemctl stop libstoragemgmt
```

サービスを起動するには、以下を実行します。

```
# systemctl start libstoragemgmt
```

27.4. LIBSTORAGEMGMT の使用

libStorageMgmt を対話的に使用するには、lsmcli ツールを使用します。

lsmcli ツールを実行するには、以下の 2 つが必要です。

- アレイに接続するプラグインと、アレイに必要な設定可能なオプションを識別するために使用される URI (Uniform Resource Identifier) です。
- アレイの有効なユーザー名およびパスワード。

URI の形式は、以下のとおりです。

```
plugin+optional-transport://user-name@host:port/?query-string-parameters
```

プラグインごとに、必要なものについて異なる要件があります。

例27.1 さまざまなプラグイン要件の例

ユーザー名やパスワードを必要としないシミュレータープラグイン

```
sim://
```

ユーザー名に root を使用した SSL 経由の NetApp プラグイン

```
ONTAP+ssl://root@filer.company.com/
```

EMC アレイ用 SSL 経由の SMI-S プラグイン

```
smis+ssl://admin@provider.com:5989/?namespace=root/emc
```

URI を使用するには、以下の 3 つのオプションがあります。

1. コマンドの一部として URI を渡します。

```
$ lsmcli -u sim://
```

2. URI を環境変数に保存します。

```
$ export LSMCLI_URI=sim://
```

3. URI をファイル ~/.lsmcli に配置します。このファイルには、"=" で区切られた名前と値のペアが含まれます。現在対応している設定は 'uri' のみです。

使用する URI をこの順序で決定する必要があります。この 3 つすべてが指定されている場合は、コマンドラインの最初の 1 つだけが使用されます。

コマンドラインで -P オプションを指定するか、環境変数 LSMCLI_PASSWORD に配置してパスワードを指定します。

例27.2 lsmcli の例

コマンドラインを使用して新しいボリュームを作成し、イニシエーターに表示する例。

この接続で対応しているアレイのリストを表示します。

```
$ lsmcli list --type SYSTEMS
ID     | Name                          | Status
-------+-------------------------------+--------
sim-01 | LSM simulated storage plug-in | OK
```

ストレージプールのリストを表示します。

```
$ lsmcli list --type POOLS -H
ID   | Name         | Total space          | Free space           | System ID
-----+--------------+----------------------+----------------------+----------
POO2 | Pool 2       | 18446744073709551616 | 18446744073709551616 | sim-01
POO3 | Pool 3       | 18446744073709551616 | 18446744073709551616 | sim-01
POO1 | Pool 1       | 18446744073709551616 | 18446744073709551616 | sim-01
POO4 | lsm_test_aggr | 18446744073709551616 | 18446744073709551616 | sim-01
```

ボリュームを作成します。

```
$ lsmcli volume-create --name volume_name --size 20G --pool POO1 -H
ID   | Name        | vpd83                            | bs  | #blocks  | status | ...
-----+-------------+----------------------------------+-----+----------+--------+----
Vol1 | volume_name | F7DDF7CA945C66238F593BC38137BD2F | 512 | 41943040 | OK     | ...
```

iSCSI イニシエーターを含むアクセスグループを作成します。

```
$ lsmcli --create-access-group example_ag --id iqn.1994-05.com.domain:01.89bd01 --type ISCSI
--system sim-01
ID                               | Name       | Initiator ID                     |SystemID
---------------------------------+------------+----------------------------------+--------
782d00c8ac63819d6cca7069282e03a0 | example_ag | iqn.1994-05.com.domain:01.89bd01 |sim-
01
```

iSCSI イニシエーターが含まれるアクセスグループを作成します。

```
$ lsmcli access-group-create --name example_ag --init iqn.1994-05.com.domain:01.89bd01 --init-
type ISCSI --sys sim-01
ID                               | Name       | Initiator IDs                    | System ID
---------------------------------+------------+----------------------------------+-----------
782d00c8ac63819d6cca7069282e03a0 | example_ag | iqn.1994-05.com.domain:01.89bd01 | sim-
01
```

新しく作成したボリュームでアクセスグループを表示できるようにします。

```
$ lsmcli access-group-grant --ag 782d00c8ac63819d6cca7069282e03a0 --vol Vol1 --access RW
```

ライブラリーのデザインでは、プロセス間通信 (IPC) を使用して、クライアントとプラグインをプロセスごとに分けています。これにより、プラグインのバグによりクライアントアプリケーションがクラッシュすることを防ぎます。また、プラグインの作成者が独自のライセンスでプラグインを作成する手段も提供します。クライアントが、URI を渡してライブラリーを開くと、クライアントライブラリーが URI を調べて、使用するプラグインを判断します。

このプラグインは、技術的にはスタンドアロンのアプリケーションですが、コマンドラインでファイル記述子を渡すように設計されています。クライアントライブラリーは、適切な Unix ドメインソケットを開いてデーモンをフォークさせ、プラグインを実行します。これにより、クライアントライブラリーで、プラグインとの通信チャンネルを指定できます。デーモンは、既存のクライアントに影響を及ぼさずに再起動できます。クライアントが、そのプラグインでライブラリーを開いている間、プラグインプロセスが実行しています。1 つ以上のコマンドを送信してプラグインを閉じると、プラグインプロセスがクリーンアップしてから終了します。

lsmcli のデフォルトの動作は、操作が完了するまで待機することです。要求されている操作によっては、これに数時間かかることがあります。通常の使用法に戻すために、コマンドラインで-bオプションを使用することができます。終了コードが 0 の場合、コマンドは完了します。終了コードが 7 の場合、コマンドは進行中で、ジョブ識別子が標準出力に書き込まれます。次に、ユーザーまたはスクリプトはジョブ ID を取得し、lsmcli --jobstatus JobID を使用して必要に応じてコマンドのステータスをクエリーできます。ジョブが完了すると、終了値が 0 になり、結果が標準出力に出力されます。コマンドがまだ進行中の場合、戻り値は 7 になり、完了率が標準出力に出力されます。

例27.3 非同期の例

コマンドがすぐに戻るように、-bオプションを渡してボリュームを作成します。

```
$ lsmcli volume-create --name async_created --size 20G --pool POO1 -b JOB_3
```

終了値を確認します。

```
$ echo $?
7
```

7 は、ジョブが進行中であることを示します。

ジョブが完了したかどうかを確認します。

```
$ lsmcli job-status --job JOB_3
33
```

終了値を確認します。7 は、ジョブが進行中であることを示しており、標準出力には完了率が表示されます。ここでは 33% と表示されています。

```
$ echo $?
7
```

しばらく待って、終了値を再度確認します。

```
$ lsmcli job-status --job JOB_3
ID   | Name          | vpd83                            | Block Size  | ...
-----+---------------+----------------------------------+-------------+-----
Vol2 | async_created | 855C9BA51991B0CC122A3791996F6B15 | 512         | ...
```

0 は成功を意味し、標準出力には新しいボリュームが表示されます。

*スクリプトを記述する場合は、**-t SeparatorCharacters** オプションを渡します。これにより、出力の解析が容易になります。*

例27.4 スクリプトの例

```
$ lsmcli list --type volumes -t#
Vol1#volume_name#049167B5D09EC0A173E92A63F6C3EA2A#512#41943040#21474836480#OK#sim-01#POO1
Vol2#async_created#3E771A2E807F68A32FA5E15C235B60CC#512#41943040#21474836480#OK#sim-01#POO1
```

```
$ lsmcli list --type volumes -t " | "
Vol1 | volume_name | 049167B5D09EC0A173E92A63F6C3EA2A | 512 | 41943040 |
21474836480 | OK | 21474836480 | sim-01 | POO1
Vol2 | async_created | 3E771A2E807F68A32FA5E15C235B60CC | 512 | 41943040 |
21474836480 | OK | sim-01 | POO1
```

```
$ lsmcli list --type volumes -s
----------------------------------------------
ID         | Vol1
Name       | volume_name
VPD83      | 049167B5D09EC0A173E92A63F6C3EA2A
Block Size | 512
#blocks    | 41943040
Size       | 21474836480
Status     | OK
System ID  | sim-01
Pool ID    | POO1
----------------------------------------------
ID         | Vol2
Name       | async_created
VPD83      | 3E771A2E807F68A32FA5E15C235B60CC
Block Size | 512
#blocks    | 41943040
Size       | 21474836480
Status     | OK
```

```
System ID  | sim-01
Pool ID    | POO1
----------------------------------------------
```

簡単ではないスクリプトには Python ライブラリーを使用することが推奨されます。

lsmcli の詳細は、lsmcli の man ページまたは lsmcli --help を参照してください。

第28章 永続メモリー : NVDIMMS

永続メモリー(pmem)は、ストレージクラスメモリーとしても呼び出され、メモリーとストレージの組み合わせです。PMEM は、ストレージの耐久性と低アクセスレイテンシー、動的 RAM (DRAM)の高帯域幅を組み合わせたものです。

- 永続メモリーはバイト単位でアドレスを指定できるため、CPU 読み込みおよびストア命令でアクセスできます。従来のブロックベースのストレージへのアクセスに必要な read() または write() システムコールに加えて、pmem はダイレクトロードおよびストアプログラミングモデルもサポートします。

- 永続メモリーのパフォーマンス特性は、アクセスレイテンシーが非常に低い (通常、数十から数百ナノ秒) DRAM と似ています。

- 永続メモリーの内容は、ストレージと同様に、電源が切れても保持されます。

永続メモリーの使用は、以下のようなユースケースで利点があります。

高速な起動: データセットはすでにメモリーにあります。

高速な起動は、ウォームキャッシュ効果とも呼ばれます。起動後、ファイルサーバーのメモリーにファイルの内容はありません。クライアントがデータを接続して読み書きすると、そのデータはページキャッシュにキャッシュされます。最終的に、キャッシュには、ほとんどのホットデータが含まれます。システムを再起動したら、プロセスを再起動する必要があります。

永続メモリーを使用すると、アプリケーションが適切に設計されていれば、システムの再起動後もアプリケーションのウォームキャッシュを維持できます。この例には、ページキャッシュは含まれません。アプリケーションは、永続メモリーに直接データをキャッシュします。

高速書き込みキャッシュ

ファイルサーバーは多くの場合、耐久性のあるメディアにデータが存在するまで、クライアントの書き込み要求を承認しません。永続メモリーを高速書き込みキャッシュとして使用すると、pmem のレイテンシーが低くなるため、ファイルサーバーは書き込み要求をすばやく確認できます。

NVDIMMs のインターリーブ

Non-Volatile Dual In-line Memory Modules (NVDIMMs) は、通常の DRAM と同じ方法でインターリーブセットにグループ化できます。インターリーブセットは、複数の DIMM にわたる RAID 0 (スト

ライプ) のようなものです。

NVDIMMS のインターリーブには、以下の利点があります。

- DRAM と同様に、NVDIMMs もインターリーブセットに設定するとパフォーマンスが向上します。
- これを使用すると、複数の小規模な NVDIMMs を、1 つのより大きな論理デバイスに統合できます。

システム BIOS または UEFI ファームウェアを使用して、インターリーブセットを設定します。

Linux では、1 つのインターリーブセットに 1 つのリージョンデバイスが作成されます。

リージョンデバイスとラベルの関係は次のとおりです。

- NVDIMMs がラベルに対応している場合は、リージョンデバイスを名前空間にさらに細かく分けることができます。
- NVDIMMs がラベルに対応していない場合、リージョンデバイスは、単一の名前空間しか含むことができません。この場合、カーネルは、リージョン全体に対応するデフォルトの namespace を作成します。

永続メモリーアクセスモード

永続メモリーデバイスは、セクター、fsdax、devdax （デバイスダイレクトアクセス）または raw モードで使用できます。

sector モード

ストレージは高速ブロックデバイスとして提示されます。セクターモードの使用は、永続メモ

リーを使用するように変更されていない従来のアプリケーションや、デバイスマッパーを含む完全な I/O スタックを利用するアプリケーションに役立ちます。

fsdax モード

これにより、Storage Networking Industry Association (SNIA) 非揮発性メモリー (NVM) プログラミングモデル仕様 で説明されているように、永続メモリーデバイスは、ダイレクトアクセスプログラミングをサポートできます。このモードでは、I/O はカーネルのストレージスタックを回避するため、多くのデバイスマッパードライバーが使用できなくなります。

devdax モード

devdax （デバイス DAX）モードは、DAX キャラクターデバイスノードを使用して永続メモリーへの raw アクセスを提供します。devdax デバイスのデータは、CPU キャッシュのフラッシュとフェンシング命令を使用して永続化できます。特定のデータベースおよび仮想マシンのハイパーバイザーは、devdax モードの利点を活用できます。ファイルシステムは、デバイス devdax インスタンスで作成することはできません。

raw モード

raw モードの名前空間にいくつかの制限があるため、使用すべきではありません。

28.1. NDCTL を使用した永続メモリーの設定

ndctl ユーティリティーを使用して永続メモリーデバイスを設定します。ndctl ユーティリティーをインストールするには、次のコマンドを使用します。

```
# yum install ndctl
```

手順28.1 ラベルをサポートしないデバイスの永続メモリーの設定

1. システムで利用可能な pmem リージョンを一覧表示します。以下の例では、コマンドは、ラベルをサポートしない NVDIMM-N デバイスをリスト表示します。

```
# ndctl list --regions
[
  {
    "dev":"region1",
    "size":34359738368,
    "available_size":0,
    "type":"pmem"
  },
  {
```

```
    "dev":"region0",
    "size":34359738368,
    "available_size":0,
    "type":"pmem"
  }
]
```

Red Hat Enterprise Linux では、各リージョンにデフォルトの名前空間が作成されます。これは、ここにある NVDIMM-N デバイスがラベルに対応していないためです。そのため、使用可能なサイズは 0 バイトです。

2. システムでアクティブでない名前空間のリストを表示します。

```
# ndctl list --namespaces --idle
[
  {
    "dev":"namespace1.0",
    "mode":"raw",
    "size":34359738368,
    "state":"disabled",
    "numa_node":1
  },
  {
    "dev":"namespace0.0",
    "mode":"raw",
    "size":34359738368,
    "state":"disabled",
    "numa_node":0
  }
]
```

3. **この領域を使用できるように、非アクティブの名前空間を再設定します。たとえば、DAX に対応するファイルシステムに namespace0.0 を使用する場合は、次のコマンドを使用します。**

```
# ndctl create-namespace --force --reconfig=namespace0.0 --mode=fsdax --map=mem
{
  "dev":"namespace0.0",
  "mode":"fsdax",
  "size":"32.00 GiB (34.36 GB)",
  "uuid":"ab91cc8f-4c3e-482e-a86f-78d177ac655d",
  "blockdev":"pmem0",
  "numa_node":0
}
```

手順28.2 ラベルをサポートするデバイスの永続メモリーの設定

1. システムで利用可能な **pmem** リージョンを一覧表示します。以下の例では、コマンドは、

ラベルをサポートする NVDIMM-N デバイスをリスト表示します。

```
# ndctl list --regions
[
  {
    "dev":"region5",
    "size":270582939648,
    "available_size":270582939648,
    "type":"pmem",
    "iset_id":-7337419320239190016
  },
  {
    "dev":"region4",
    "size":270582939648,
    "available_size":270582939648,
    "type":"pmem",
    "iset_id":-137289417188962304
  }
]
```

2.

NVDIMM デバイスがラベルをサポートしている場合、デフォルトの名前空間は作成されず、--force フラグまたは --reconfigure フラグを使用せずに、リージョンから 1 つ以上の名前空間を割り当てることができます。

```
# ndctl create-namespace --region=region4 --mode=fsdax --map=dev --size=36G
{
  "dev":"namespace4.0",
  "mode":"fsdax",
  "size":"35.44 GiB (38.05 GB)",
  "uuid":"9c5330b5-dc90-4f7a-bccd-5b558fa881fe",
  "blockdev":"pmem4",
  "numa_node":0
}
```

これで、同じリージョンから別の名前空間を作成できるようになります。

```
# ndctl create-namespace --region=region4 --mode=fsdax --map=dev --size=36G
{
  "dev":"namespace4.1",
  "mode":"fsdax",
  "size":"35.44 GiB (38.05 GB)",
  "uuid":"91868e21-830c-4b8f-a472-353bf482a26d",
  "blockdev":"pmem4.1",
  "numa_node":0
}
```

以下のコマンドを使用して、同じリージョンから異なるタイプの名前空間を作成することもできます。

```
# ndctl create-namespace --region=region4 --mode=devdax --align=2M --size=36G
{
  "dev":"namespace4.2",
  "mode":"devdax",
  "size":"35.44 GiB (38.05 GB)",
  "uuid":"a188c847-4153-4477-81bb-7143e32ffc5c",
  "daxregion":
  {
    "id":4,
    "size":"35.44 GiB (38.05 GB)",
    "align":2097152,
    "devices":[
     {
       "chardev":"dax4.2",
       "size":"35.44 GiB (38.05 GB)"
     }]
  },
    "numa_node":0
}
```

ndctl ユーティリティーの詳細は、man ndctl を参照してください。

28.2. ブロックデバイスとして使用する永続メモリーの設定 (レガシーモード)

永続メモリーを高速ブロックデバイスとして使用するには、名前空間をセクターモードに設定します。

```
# ndctl create-namespace --force --reconfig=namespace1.0 --mode=sector
{
  "dev":"namespace1.0",
  "mode":"sector",
  "size":17162027008,
  "uuid":"029caa76-7be3-4439-8890-9c2e374bcc76",
  "sector_size":4096,
  "blockdev":"pmem1s"
}
```

この例では、**namespace1.0** はセクターモードに再設定されます。ブロックデバイス名が **pmem1** から **pmem1 s** に変更されました。このデバイスは、システム上の他のブロックデバイスと同じように使用できます。たとえば、デバイスをパーティション化して、デバイスにファイルシステムを作成したり、デバイスをソフトウェア **RAID** セットの一部として設定したり、デバイスを **dm-cache** のキャッシュデバイスにしたりできます。

28.3. ファイルシステムのダイレクトアクセス向け永続メモリーの設定

ファイルシステムの直接アクセスでは、名前空間を **fsdax** モードに設定する必要があります。このモードでは、ダイレクトアクセスプログラミングモデルが可能になります。デバイスが **fsdax** モードで

設定されている場合、その上にファイルシステムを作成し、**-o fsdax** マウントオプションでマウントできます。次に、このファイルシステムのファイルで **mmap()** 操作を実行するアプリケーションは、そのストレージに直接アクセスします。以下の例を参照してください。

```
# ndctl create-namespace --force --reconfig=namespace0.0 --mode=fsdax --map=mem
{
  "dev":"namespace0.0",
  "mode":"fsdax",
  "size":17177772032,
  "uuid":"e6944638-46aa-4e06-a722-0b3f16a5acbf",
  "blockdev":"pmem0"
}
```

この例では、**namespace0.0** は名前空間 **fsdax** モードに変換されます。**--map=mem** 引数を使用すると、**ndctl** はダイレクトメモリーアクセス(DMA)に使用されるオペレーティングシステムのデータ構造をシステム **DRAM** に配置します。

DMA を実行するには、カーネルに、メモリー領域の各ページのデータ構造が必要です。このデータ構造のオーバーヘッドは、**4 KiB** ページごとに **64** バイトです。小さいデバイスでは、問題なく **DRAM** に収まるのに十分なオーバーヘッド量があります。たとえば、**16 GiB** の名前区間のページ構造に必要なのは **256 MiB** だけです。**NVDIMM** デバイスは通常小さくて高価であるため、**--map=mem** パラメーターが示すように、カーネルのページトラッキングデータ構造を **DRAM** に保存することが推奨されます。

将来的には、**NVDIMM** デバイスのサイズはテラバイトになる可能性があります。このようなデバイスでは、ページトラッキングデータ構造の保存に必要なメモリーの量が、システム内の **DRAM** の量を超える可能性があります。永続メモリーの **1 TiB** には、ページ構造だけで **16 GiB** が必要です。そのため、**--map=dev** パラメーターを指定して、データ構造を永続メモリー自体に保存することが推奨されます。

fsdax モードで名前空間を設定すると、名前空間がファイルシステムの準備が整います。**Red Hat Enterprise Linux 7.3** 以降、**Ext4** と **XFS** の両方のファイルシステムで、テクノロジープレビューとして永続メモリーを使用できるようになりました。ファイルシステムの作成に特別な引数は必要ありません。**DAX** 機能を取得するには、**dax** マウントオプションを使用してファイルシステムをマウントします。以下に例を示します。

```
# mkfs -t xfs /dev/pmem0
# mount -o dax /dev/pmem0 /mnt/pmem/
```

次に、アプリケーションは永続メモリーを使用して **/mnt/pmem/** ディレクトリーにファイルを作成し、ファイルを開き、**mmap** 操作を使用して直接アクセス用のファイルをマップできます。

ダイレクトアクセスに使用する **pmem** デバイスにパーティションを作成する場合は、パーティションはページ境界に合わせる必要があります。**Intel 64** および **AMD64** アーキテクチャーでは、パーティ

ションの開始および終了に 4KiB 以上のアライメントを使用しますが、推奨されるアライメントは 2MiB です。parted ツールは、デフォルトでは 1MiB の境界にパーティションを合わせます。最初のパーティションには、パーティションの開始部分として 2MiB を指定します。パーティションのサイズが 2MiB の倍数の場合は、その他のパーティションも合わせてアライメントされます。

28.4. デバイス DAX モードで使用する永続メモリーの設定

デバイス DAX (devdax)は、ファイルシステムの関与なしに、アプリケーションがストレージに直接アクセスできる手段を提供します。デバイス DAX の利点は、ndctl ユーティリティーの --align オプションを使用して設定できる、保証されたフォールトの粒度を提供することです。

```
# ndctl create-namespace --force --reconfig=namespace0.0 --mode=devdax --align=2M
```

指定されたコマンドにより、オペレーティングシステムが一度に 2MiB ページで障害が発生することが保証されます。Intel 64 アーキテクチャーおよび AMD64 アーキテクチャーでは、以下のフォールト粒度に対応しています。

- *4KiB*
- *2MiB*
- *1GiB*

デバイス DAX ノード(/dev/daxN.M)は、以下のシステムコールのみをサポートします。

- *open()*
- *close()*
- *mmap()*
- *fallocate()*

read() ユースケースは永続メモリープログラミングに関連付けられるため、write() バリアントおよ

び バリアントはサポートされません。

28.5. NVDIMM のトラブルシューティング

28.5.1. S.M.A.R.T を使用した NVDIMM の正常性の監視

一部の NVDIMMs は、正常性情報を取得する Self-Monitoring, Analysis and Reporting Technology (S.M.A.R.T.) インターフェイスに対応しています。

NVDIMM 正常性を定期的に監視して、データの損失を防ぎます。S.M.A.R.T. が NVDIMM の正常性ステータスで問題を報告する場合は、「破損した NVDIMM の検出と置き換え」 の説明に従って置き換えます。

前提条件

- 一部のシステムでは、以下のコマンドを使用して正常性情報を取得するために、acpi_ipmi ドライバーを読み込む必要があります。

```
# modprobe acpi_ipmi
```

手順

- 正常性情報にアクセスするには、次のコマンドを使用します。

```
# ndctl list --dimms --health
...
  {
    "dev":"nmem0",
    "id":"802c-01-1513-b3009166",
    "handle":1,
    "phys_id":22,
    "health":
    {
      "health_state":"ok",
      "temperature_celsius":25.000000,
      "spares_percentage":99,
      "alarm_temperature":false,
      "alarm_spares":false,
      "temperature_threshold":50.000000,
      "spares_threshold":20,
      "life_used_percentage":1,
      "shutdown_state":"clean"
    }
  }
...
```

28.5.2. 破損した NVDIMM の検出と置き換え

システムログまたは S.M.A.R.T. に NVDIMM 関連のエラーメッセージが記録される場合は、NVDIMM デバイスがエラーを起こしていることが考えられます。この場合は、以下を行う必要があります。

1. どの NVDIMM デバイスに障害が発生しているかを検出する
2. 保存されているデータのバックアップを作成する
3. デバイスを物理的に交換する

手順28.3 破損した NVDIMM の検出と置き換え

1. 破損した DIMM を検出するには、次のコマンドを使用します。

```
# ndctl list --dimms --regions --health --media-errors --human
```

badblocks フィールドは、NVDIMM が破損していることを示しています。dev フィールドに名前を書き留めます。以下の例では、nmem0 という名前の NVDIMM が破損しています。

例28.1 NVDIMM デバイスの正常性ステータス

```
# ndctl list --dimms --regions --health --media-errors --human

...
  "regions":[
    {
      "dev":"region0",
      "size":"250.00 GiB (268.44 GB)",
      "available_size":0,
      "type":"pmem",
      "numa_node":0,
      "iset_id":"0xXXXXXXXXXXXXXXXX",
      "mappings":[
       {
         "dimm":"nmem1",
         "offset":"0x10000000",
         "length":"0x1f40000000",
         "position":1
       },
       {
         "dimm":"nmem0",
         "offset":"0x10000000",
```

```
        "length":"0x1f40000000",
        "position":0
      }
     ],
     "badblock_count":1,
     "badblocks":[
      {
       "offset":65536,
       "length":1,
       "dimms":[
        "nmem0"
       ]
      }
     ],
     "persistence_domain":"memory_controller"
    }
   ]
  }
```

2. 次のコマンドを使用して、破損した NVDIMM の phys_id 属性を検索します。

```
# ndctl list --dimms --human
```

前の例では、nmem0 が破損した NVDIMM であることがわかります。したがって、nmem0 の phys_id 属性を確認します。以下の例では、phys_id は 0x10 です。

例28.2 NVDIMMs の phys_id 属性

```
# ndctl list --dimms --human

[
  {
    "dev":"nmem1",
    "id":"XXXX-XX-XXXX-XXXXXXXX",
    "handle":"0x120",
    "phys_id":"0x1c"
  },
  {
    "dev":"nmem0",
    "id":"XXXX-XX-XXXX-XXXXXXXX",
    "handle":"0x20",
    "phys_id":"0x10",
    "flag_failed_flush":true,
    "flag_smart_event":true
  }
]
```

3.

次のコマンドを使用して、破損した NVDIMM のメモリースロットを確認します。

```
# dmidecode
```

出力で、Handle 識別子が破損した NVDIMM の phys_id 属性と一致するエントリーを見つけます。Locator フィールドには、破損した NVDIMM が使用するメモリースロットが一覧表示されます。以下の例では、nmem0 デバイスは 0x0010 識別子と一致し、DIMM-XXX-YYYY メモリースロットを使用します。

例28.3 NVDIMM メモリースロットリスティング

```
# dmidecode

...
Handle 0x0010, DMI type 17, 40 bytes
Memory Device
        Array Handle: 0x0004
        Error Information Handle: Not Provided
        Total Width: 72 bits
        Data Width: 64 bits
        Size: 125 GB
        Form Factor: DIMM
        Set: 1
        Locator: DIMM-XXX-YYYY
        Bank Locator: Bank0
        Type: Other
        Type Detail: Non-Volatile Registered (Buffered)
...
```

4.

NVDIMM 上の名前空間にある全データのバックアップを作成します。NVDIMM を交換する前にデータのバックアップを作成しないと、システムから NVDIMM を削除したときにデータが失われます。

警告

時折、NVDIMM が完全に破損すると、バックアップが失敗することがあります。

これを防ぐため、「S.M.A.R.T を使用した NVDIMM の正常性の監視」 の説明に従って、S.M.A.R.T. を使用して NVDIMMs デバイスを定期的に監視し、破損する前にエラーを起こしている NVDIMMs を交換してください。

次のコマンドを使用して、NVDIMM の名前空間をリスト表示します。

```
# ndctl list --namespaces --dimm=DIMM-ID-number
```

以下の例では、nmem0 デバイスには、バックアップする必要がある namespace 0.0 および namespace 0.2 の名前空間が含まれます。

例28.4 NVDIMM 名前空間のリスト表示

```
# ndctl list --namespaces --dimm=0

[
  {
    "dev":"namespace0.2",
    "mode":"sector",
    "size":67042312192,
    "uuid":"XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX",
    "raw_uuid":"XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX",
    "sector_size":4096,
    "blockdev":"pmem0.2s",
    "numa_node":0
  },
  {
    "dev":"namespace0.0",
    "mode":"sector",
    "size":67042312192,
    "uuid":"XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX",
    "raw_uuid":"XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX",
    "sector_size":4096,
    "blockdev":"pmem0s",
    "numa_node":0
  }
]
```

5.

破損した NVDIMM を物理的に交換します。

第29章 NVME OVER FABRIC デバイスの概要

Non-volatile Memory Express (NVMe) は、ホストソフトウェアユーティリティーがソリッドステートドライブと通信できるようにするインターフェイスです。次のタイプのファブリックトランスポートを使用して、NVMe over fabric デバイスを設定します。

- RDMA (Remote Direct Memory Access) を使用する NVMe over fabricsNVMe/RDMA の設定方法の詳細は、「RDMA を使用した NVMe over fabrics」 を参照してください。

- ファイバーチャネル (FC) を使用する NVMe over fabricsFC-NVMe の設定方法の詳細は、「FC を使用した NVMe over fabrics」 を参照してください。

FC および RDMA を使用する場合は、ソリッドステートドライブをシステムにローカルにする必要はありません。また、FC または RDMA コントローラーを介してリモートで設定できます。

29.1. RDMA を使用した NVME OVER FABRICS

次のセクションでは、RDMA (NVMe/RDMA) イニシエーター設定で NVMe over RDMA をデプロイする方法を説明します。

29.1.1. RDMA クライアントでの NVMe の設定

この手順を使用して、NVMe 管理コマンドラインインターフェイス(nvme-cli)を使用して NVMe/RDMA クライアントを設定します。

1. nvme-cli パッケージをインストールします。

```
# yum install nvme-cli
```

2. nvme-rdma モジュールが読み込まれていない場合は読み込みます。

```
# modprobe nvme-rdma
```

3. NVMe ターゲットで利用可能なサブシステムを検出します。

```
# nvme discover -t rdma -a 172.31.0.202 -s 4420

Discovery Log Number of Records 1, Generation counter 2
=====Discovery Log Entry 0======
trtype:  rdma
adrfam:  ipv4
subtype: nvme subsystem
treq:    not specified, sq flow control disable supported
portid:  1
trsvcid: 4420
subnqn:  testnqn
traddr:  172.31.0.202
rdma_prtype: not specified
rdma_qptype: connected
rdma_cms:    rdma-cm
rdma_pkey: 0x0000
```

4. **検出されたサブシステムに接続します。**

```
# nvme connect -t rdma -n testnqn -a 172.31.0.202 -s 4420

# lsblk

NAME                      MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda                         8:0    0 465.8G  0 disk
├─sda1                      8:1    0     1G  0 part /boot
└─sda2                      8:2    0 464.8G  0 part
  ├─rhel_rdma--virt--03-root 253:0    0    50G  0 lvm  /
  ├─rhel_rdma--virt--03-swap 253:1    0     4G  0 lvm  [SWAP]
  └─rhel_rdma--virt--03-home 253:2    0 410.8G  0 lvm  /home
nvme0n1

# cat /sys/class/nvme/nvme0/transport

rdma
```

testnqn を、NVMe サブシステムの名前に置き換えます。

172.31.0.202 を、目的の IP アドレスに置き換えます。

4420 を、ポート番号に置き換えます。

5. **現在接続されている NVMe デバイスのリストを表示します。**

```
# nvme list
```

6. *(必要に応じて) ターゲットから切断します。*

```
# nvme disconnect -n testnqn

NQN:testnqn disconnected 1 controller(s)

# lsblk

NAME                      MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda                         8:0    0 465.8G  0 disk
├─sda1                      8:1    0     1G  0 part /boot
└─sda2                      8:2    0 464.8G  0 part
├─rhel_rdma--virt--03-root 253:0    0    50G  0 lvm  /
├─rhel_rdma--virt--03-swap 253:1    0     4G  0 lvm  [SWAP]
└─rhel_rdma--virt--03-home 253:2    0 410.8G  0 lvm  /home
```

関連情報

- 詳細は、**nvme** の **man** ページおよび **NVMe-cli Github リポジトリー** を参照してください。

29.2. FC を使用した NVME OVER FABRICS

NVMe over Fiber Channel (FC-NVMe) は、特定の Broadcom Emulex および Marvell Qlogic Fibre Channel アダプターで使用すると、イニシエーターモードで完全にサポートされます。システム管理者として、以下のセクションのタスクを完了し、FC-NVMe をデプロイします。

- 「Broadcom アダプターの NVMe イニシエーターの設定」
- 「QLogic アダプターの NVMe イニシエーターの設定」

29.2.1. Broadcom アダプターの NVMe イニシエーターの設定

この手順を使用して、NVMe 管理コマンドラインインターフェイス(nvme-cli)ツールを使用して、Broadcom アダプタークライアントの NVMe イニシエーターを設定します。

1. **nvme-cli ツールをインストールします。**

```
# yum install nvme-cli
```

これにより、/etc/nvme/ ディレクトリーに hostnqn ファイルが作成されます。hostnqn ファイルは NVMe ホストを識別します。新しい hostnqn を生成するには、以下を実行します。

```
# nvme gen-hostnqn
```

2. 以下の内容で /etc/modprobe.d/lpfc.conf ファイルを作成します。

```
options lpfc lpfc_enable_fc4_type=3
```

3. initramfs イメージを再構築します。

```
# dracut --force
```

4. ホストシステムを再起動して、lpfc ドライバーを再設定します。

```
# systemctl reboot
```

5. ローカルポートおよびリモートポートの WWNN および WWPN を見つけ、この出力を使用してサブシステム NQN を見つけます。

```
# cat /sys/class/scsi_host/host*/nvme_info

NVME Initiator Enabled
XRI Dist lpfc0 Total 6144 IO 5894 ELS 250
NVME LPORT lpfc0 WWPN x10000090fae0b5f5 WWNN x20000090fae0b5f5 DID x010f00
ONLINE
NVME RPORT       WWPN x204700a098cbcac6 WWNN x204600a098cbcac6 DID x01050e
TARGET DISCSRVC ONLINE

NVME Statistics
LS: Xmt 000000000e Cmpl 000000000e Abort 00000000
LS XMIT: Err 00000000  CMPL: xb 00000000 Err 00000000
Total FCP Cmpl 00000000000008ea Issue 00000000000008ec OutIO 0000000000000002
    abort 00000000 noxri 00000000 nondlp 00000000 qdepth 00000000 wqerr 00000000 err
00000000
FCP CMPL: xb 00000000 Err 00000000
```

```
# nvme discover --transport fc \ --traddr nn-0x204600a098cbcac6:pn-0x204700a098cbcac6 \
--host-traddr nn-0x20000090fae0b5f5:pn-0x10000090fae0b5f5

Discovery Log Number of Records 2, Generation counter 49530
=====Discovery Log Entry 0======
```

```
trtype:  fc
adrfam:  fibre-channel
subtype: nvme subsystem
treq:    not specified
portid:  0
trsvcid: none
subnqn: nqn.1992-
08.com.netapp:sn.e18bfca87d5e11e98c0800a098cbcac6:subsystem.st14_nvme_ss_1_1
traddr:  nn-0x204600a098cbcac6:pn-0x204700a098cbcac6
```

nn-0x204600a098cbcac6:pn-0x204700a098cbcac6 を、traddr に置き換えます。

nn-0x20000090fae0b5f5:pn-0x10000090fae0b5f5 を、host_traddr に置き換えます。

6. **nvme-cli を使用して NVMe ターゲットに接続します。**

```
# nvme connect --transport fc --traddr nn-0x204600a098cbcac6:pn-0x204700a098cbcac6 --
host-traddr nn-0x20000090fae0b5f5:pn-0x10000090fae0b5f5 -n nqn.1992-
08.com.netapp:sn.e18bfca87d5e11e98c0800a098cbcac6:subsystem.st14_nvme_ss_1_1
```

nn-0x204600a098cbcac6:pn-0x204700a098cbcac6 を traddr に置き換えます。

nn-0x20000090fae0b5f5:pn-0x10000090fae0b5f5 を host_traddr に置き換えます。

**nqn.1992-
08.com.netapp:sn.e18bfca87d5e11e98c0800a098cbcac6:subsystem.st14_nvme_ss_1_ 1
を、subnqn に置き換えます。**

7. **NVMe デバイスが現在接続されていることを確認します。**

```
# nvme list
Node             SN                   Model                                    Namespace Usage
Format           FW Rev
---------------- -------------------- ---------------------------------------- --------- ------------------------
- ---------------- --------
/dev/nvme0n1     80BgLFM7xMJbAAAAAAAC NetApp ONTAP Controller                   1
107.37  GB / 107.37  GB      4 KiB +  0 B   FFFFFFFF
# lsblk |grep nvme
  nvme0n1                 259:0    0   100G  0 disk
```

関連情報

- 詳細は、nvme の man ページおよび NVMe-cli Github リポジトリー を参照してください。

29.2.2. QLogic アダプターの NVMe イニシエーターの設定

この手順を使用して、NVMe 管理コマンドラインインターフェイス (nvme-cli) ツールを使用して、Qlogic アダプタークライアントの NVMe イニシエーターを設定します。

1. nvme-cli ツールをインストールします。

```
# yum install nvme-cli
```

これにより、/etc/nvme/ ディレクトリーに hostnqn ファイルが作成されます。hostnqn ファイルは NVMe ホストを識別します。新しい hostnqn を生成するには、以下を実行します。

```
# nvme gen-hostnqn
```

2. qla2xxx モジュールを削除して再読み込みします。

```
# rmmod qla2xxx
# modprobe qla2xxx
```

3. ローカルポートおよびリモートポートの WWNN および WWPN を検索します。

```
# dmesg |grep traddr
    [    6.139862] qla2xxx [0000:04:00.0]-ffff:0: register_localport: host-traddr=nn-0x20000024ff19bb62:pn-0x21000024ff19bb62 on portID:10700
[    6.241762] qla2xxx [0000:04:00.0]-2102:0: qla_nvme_register_remote: traddr=nn-0x203b00a098cbcac6:pn-0x203d00a098cbcac6 PortID:01050d
```

この host-traddr および traddr を使用して、サブシステム NQN を見つけます。

```
# nvme discover --transport fc --traddr nn-0x203b00a098cbcac6:pn-0x203d00a098cbcac6 --host-traddr nn-0x20000024ff19bb62:pn-0x21000024ff19bb62

Discovery Log Number of Records 2, Generation counter 49530
=====Discovery Log Entry 0======
trtype:  fc
```

```
adrfam:  fibre-channel
subtype: nvme subsystem
treq:    not specified
portid:  0
trsvcid: none
subnqn:  nqn.1992-
08.com.netapp:sn.c9ecc9187b1111e98c0800a098cbcac6:subsystem.vs_nvme_multipath_1_su
bsystem_468
traddr:  nn-0x203b00a098cbcac6:pn-0x203d00a098cbcac6
```

nn-0x203b00a098cbcac6:pn-0x203d00a098cbcac6 を traddr に置き換えます。

nn-0x20000024ff19bb62:pn-0x21000024ff19bb62 を、host_traddr に置き換えます。

4. **nvme-cli ツールを使用して NVMe ターゲットに接続します。**

```
# nvme connect --transport fc --traddr nn-0x203b00a098cbcac6:pn-0x203d00a098cbcac6 --
host_traddr nn-0x20000024ff19bb62:pn-0x21000024ff19bb62 -n nqn.1992-
08.com.netapp:sn.c9ecc9187b1111e98c0800a098cbcac6:subsystem.vs_nvme_multipath_1_su
bsystem_468
```

nn-0x203b00a098cbcac6:pn-0x203d00a098cbcac6 を traddr に置き換えます。

nn-0x20000024ff19bb62:pn-0x21000024ff19bb62 を、host_traddr に置き換えます。

nqn.1992-08.com.netapp:sn.c9ecc9187b1111e98c0800a098cbcac6:subsystem.vs_nvme_multipath_1_subsystem_468 を subnqn に置き換えます。

5. **NVMe デバイスが現在接続されていることを確認します。**

```
# nvme list
Node             SN                   Model                                    Namespace Usage
Format           FW Rev
---------------- -------------------- ---------------------------------------- --------- -------------------------
- ---------------- --------
/dev/nvme0n1     80BgLFM7xMJbAAAAAAAC NetApp ONTAP Controller                   1
107.37  GB / 107.37  GB      4 KiB +  0 B   FFFFFFFF
# lsblk |grep nvme
nvme0n1                 259:0    0   100G  0 disk
```

関連情報

- *詳細は、nvme の man ページおよび NVMe-cli Github リポジトリー を参照してください。*

パート III. VDO を使用したデータの重複排除と圧縮

本パートでは、VDO (Virtual Data Optimizer) を使用できるようにすることで、既存のストレージ管理アプリケーションに重複排除ブロックストレージ機能を提供する方法を説明します。

第30章 VDO 統合

30.1. VDO の理論的概要

VDO (Virtual Data Optimizer) は、ブロックストレージの圧縮および重複排除のプールを簡単に作成できるブロック仮想化技術です。

- 重複排除 とは、重複ブロックの複数のコピーを削除することで、ストレージリソースの消費を低減させるための技術です。

  同じデータを複数回書き込むのではなく、VDO は各重複ブロックを検出し、元のブロックへの参照として記録します。VDO は、VDO の上にあるストレージ層により使用されている論理ブロックアドレスから、VDO の下にあるストレージ層で使用される物理ブロックアドレスへのマッピングを維持します。

  重複排除後、複数の論理ブロックアドレスは、同じ物理ブロックアドレスにマッピングできます。これらは、共有ブロック と呼ばれます。ブロックストレージの共有は、VDO が存在しない場合に、読み込みブロックと書き込みブロックが行われるストレージのユーザーには表示されません。共有ブロックを上書きすると、新しい物理ブロックが割り当てられて新しいブロックデータが保存され、共有物理ブロックにマッピングされているその他の論理ブロックアドレスが変更されないようにします。

- 圧縮 は、ログファイルやデータベースなど、必ずしもブロックレベルの冗長性を示さないファイル形式で適切に機能するデータ削減技術です。詳細は、「圧縮の使用」 を参照してください。

VDO ソリューションは、以下のコンポーネントで設定されます。

kvdo

Linux Device Mapper 層に読み込まれるカーネルモジュールは、重複排除され、圧縮され、シンプロビジョニングされたブロックストレージボリュームを提供します。

uds

ボリューム上の Universal Deduplication Service (UDS) インデックスと通信し、データの重複を分析するカーネルモジュール。

コマンドラインツール

最適化されたストレージの設定および管理

30.1.1. UDS カーネルモジュール(uds)

UDS インデックスは、VDO 製品の基盤を提供します。新しい各データについて、その部分が保存してあるデータ内容と同一であるかどうかを素早く判断します。インデックスが一致すると、ストレージシステムは、同じ情報を複数格納しないように、既存の項目を内部的に参照できます。

UDS インデックスは、カーネル内で uds カーネルモジュールとして実行されます。

30.1.2. VDO カーネルモジュール(kvdo)

kvdo Linux カーネルモジュールは、Linux Device Mapper レイヤー内でブロック層の重複排除サービスを提供します。Linux カーネルでは、Device Mapper はブロックストレージのプールを管理する一般的なフレームワークとして機能し、カーネルのブロックインターフェイスと実際のストレージデバイスドライバーとの間でストレージスタックにブロック処理モジュールを挿入できるようにします。

kvdo モジュールは、ブロックストレージ用に直接アクセスできるブロックデバイスとして公開され、XFS や ext4 など、利用可能な多くの Linux ファイルシステムのいずれかを介して提示されます。kvdo が VDO ボリュームからデータの（論理）ブロックを読み取る要求を受信すると、要求された論理ブロックを基礎となる物理ブロックにマッピングし、要求されたデータを読み取り、返します。

kvdo は、データのブロックを VDO ボリュームに書き込む要求を受信すると、最初にそれが DISCARD または TRIM 要求であるかどうか、データが均一にゼロであるかどうかを確認します。これらの条件のいずれかが保持されている場合、kvdo はブロックマップを更新し、要求を承認します。それ以外の場合は、リクエストで使用する物理ブロックが割り当てられます。

VDO 書き込みポリシーの概要

kvdo モジュールが同期モードで動作している場合は、以下を行います。

1. リクエストのデータを一時的に、割り当てられたブロックに書き込み、リクエストを承認します。

2. 承認が完了すると、ブロックデータの MurmurHash-3 署名を計算してブロックの重複排除が試行されます。これは、VDO インデックスに送信されます。

3. *VDO インデックスに同じ署名を持つブロックのエントリーが含まれている場合、kvdo は示されたブロックを読み取り、2 つのブロックのバイトバイバイト比較を行い、同一であることを確認します。*

4. *同一であると、kvdo はブロックマップを更新し、論理ブロックが対応する物理ブロックをポイントするようにし、割り当てられた物理ブロックを解放します。*

5. *VDO インデックスに、書き込まれているブロックの署名のエントリーが含まれていないか、示されたブロックに実際に同じデータが含まれていない場合、kvdo はブロックマップを更新して、一時的な物理ブロックを永続的にします。*

kvdo が非同期モードで動作している場合は、以下を行います。

1. *データを書き込む代わりに、リクエストをすぐに承認します。*

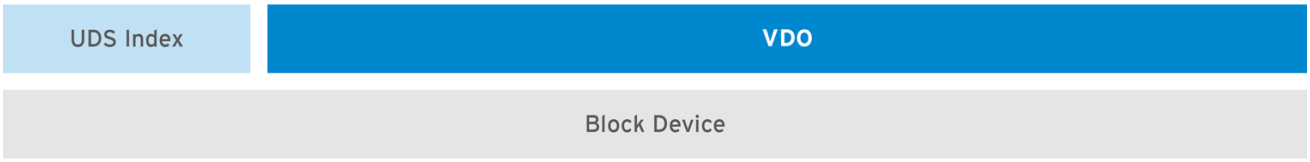
2. *上記の説明と同じように、ブロックの重複排除試行が行われます。*

3. *ブロックが重複していると、kvdo はブロックマップを更新し、割り当てられたブロックを解放します。それ以外の場合は、要求のデータが割り当てられたブロックに書き込まれ、ブロックマップが更新されて物理ブロックが永続的になります。*

30.1.3. VDO ボリューム

VDO は、バッキングストアとしてブロックデバイスを使用します。これは、複数のディスク、パーティション、またはフラットファイルで設定される物理ストレージの集約を含めることができます。ストレージ管理ツールで VDO ボリュームを作成すると、VDO は、UDS インデックスと VDO ボリュームの両方にボリュームから領域を確保します。これは相互作用し、重複排除されたブロックストレージをユーザーおよびアプリケーションに提供します。図30.1「VDO ディスクの設定」 を参照してください。

図30.1 VDO ディスクの設定



スラブ

VDO ボリュームの物理ストレージは複数のスラブに分割され、それぞれが物理領域内の連続した領域になります。指定のボリュームのスラブはすべて同じサイズになります。この値は、128 MB の 2 乗で、最大 32 GB にすることができます。

小規模なテストシステムで VDO の評価を容易にするために、デフォルトのスラブサイズは 2 GB です。1 つの VDO ボリュームに、最大 8192 個のスラブが含まれる場合があります。したがって、デフォルト設定の 2 GB のスラブを使用する場合、許可される物理ストレージは最大 16 TB です。32 GB のスラブを使用する場合、許可される物理ストレージは最大 256 TB です。スラブ全体が、メタデータ用に VDO により予約されているため、ユーザーデータの格納には使用できません。

スラブサイズは、VDO ボリュームのパフォーマンスには影響しません。

表30.1 物理ボリュームのサイズ別に推奨される VDO スラブサイズ

物理ボリュームのサイズ	推奨されるスラブサイズ
10 - 99 GB	1 GB
100 GB - 1 TB	2 GB
2 - 256 TB	32 GB

スラブのサイズは、vdo create コマンドに
--vdoSlabSize=megabytes
オプションを指定することで制御できます。

物理サイズと利用可能な物理サイズ

物理サイズと利用可能な物理サイズは、どちらも、VDO が利用できるブロックデバイスのディスク領域の量を表します。

- *物理サイズ は、基盤となるブロックデバイスと同じサイズになります。VDO は、以下の目的でこのストレージを使用します。*
  - *重複排除および圧縮される可能性があるユーザーデータ*

- UDS インデックスなどの VDO メタデータ

- 利用可能な物理サイズ は、VDO がユーザーデータに使用できる物理サイズの一部です。

  これは、メタデータのサイズを引いた物理サイズと同等で、指定のスラブサイズでボリュームをスラブに分割した後の残りを引いたものと同じです。

さまざまなサイズのブロックデバイスで、ストレージ VDO メタデータに必要な量の例は、「物理ボリュームのサイズ別の VDO システム要件の例」 を参照してください。

論理サイズ

--vdoLogicalSize オプションを指定しない場合は、使用可能な物理ボリュームのサイズが、論理ボリュームのデフォルトサイズとなります。図30.1「VDO ディスクの設定」 では、VDO で重複排除したストレージターゲットがブロックデバイス上に完全に配置されています。つまり、VDO ボリュームの物理サイズは、基礎となるブロックデバイスと同じサイズになります。

VDO は現在、絶対最大論理サイズ 4PB の物理ボリュームの最大 254 倍の論理サイズに対応します。

30.1.4. コマンドラインツール

VDO には、設定および管理用の次のコマンドラインツールが同梱されています。

vdo

VDO ボリュームの作成、設定、および制御

vdostats

使用率とパフォーマンスの統計を提供します。

30.2. システム要件

プロセッサーアーキテクチャー

Intel 64 命令セットを実装するプロセッサー (AMD64 または Intel 64 アーキテクチャーのプロセッ

サー) が 1 つ以上必要です。

RAM

各 VDO ボリュームには、2 つの異なるメモリー要件があります。

- VDO モジュールには、マネージドの物理ストレージ 1TB ごとに 370MB と、追加の 268MB が必要です。
- Universal Deduplication Service (UDS) インデックスには、最低 250MB の DRAM が必要です。これは、重複排除が使用するデフォルトの容量でもあります。UDS のメモリー使用量は、「UDS インデックスのメモリー要件」 を参照してください。

ストレージ

VDO ボリュームは、シンプロビジョニングしたブロックデバイスです。物理ボリュームで実行しないようにするには、ボリュームを、後で拡張できるストレージの上に配置します。このような拡張可能なストレージの例は、LVM ボリュームまたは MD RAID アレイです。

1 つの VDO ボリュームで、最大 256TB の物理ストレージを使用するように設定できます。VDO が管理するボリュームの使用可能サイズを、VDO が提供するストレージプールの物理サイズから判断するための計算は、「VDO ストレージ領域の要件」 を参照してください。

追加のシステムソフトウェア

VDO は、次のソフトウェアに依存します。

- LVM
- Python 2.7

yum パッケージマネージャーは、必要なすべてのソフトウェア依存関係を自動的にインストールします。

ストレージスタックでの VDO の配置

原則として、VDO の下に配置する必要があるストレージ層と、VDO の上に配置する必要があるストレージ層があります。

- *VDO の下 : DM-Multipath、DM-Crypt、およびソフトウェア RAID (LVM または mdraid)。*

- *VDO の上: LVM キャッシュ、LVM スナップショット、および LVM シンプロビジョニング。*

以下の設定はサポートされていません。

- *VDO ボリュームの上に VDO - ストレージ → LVM → VDO*

- *LVM スナップショット上の VDO*

- *LVM キャッシュ上の VDO*

- *ループバックデバイス上の VDO*

- *LVM シンプロビジョニング上の VDO*

- *VDO の上に 暗号化されたボリューム - ストレージ → VDO → DM-Crypt*

- *VDO ボリューム上のパーティション： fdisk、parted、および同様のパーティション*

- *VDO ボリューム上の RAID (LVM、MD、またはその他のタイプ)*

重要

VDO は、sync と async の 2 つの書き込みモードに対応します。VDO が sync モードの場合、基礎となるストレージがデータを永続的に書き込みすると、VDO デバイスへの書き込みが確認されます。VDO が async モードの場合、永続ストレージに書き込まれる前に書き込みが確認されます。

VDO 書き込みポリシーを設定して、基盤となるストレージの動作を一致させることが重要です。デフォルトでは、VDO 書き込みポリシーは auto オプションに設定されており、適切なポリシーが自動的に選択されます。

詳細は、「VDO 書き込みモードの選択」 を参照してください。

30.2.1. UDS インデックスのメモリー要件

UDS インデックスは 2 つの部分から成ります。

- 一意のブロックごとに最大 1 つのエントリーを含むメモリーでは、コンパクトな表が用いられています。
- 発生する際に、インデックスに対して示される関連のブロック名を順に記録するオンディスクコンポーネント。

UDS は、メモリーの各エントリーに対して平均 4 バイトを使用します (キャッシュを含む)。

オンディスクコンポーネントは、UDS に渡されるデータの境界履歴を維持します。UDS は、直近で確認されたブロックの名前を含む、この重複排除ウィンドウ内のデータの重複排除アドバイスを提供します。重複排除ウィンドウでは、大規模なデータリポジトリーに対して必要なメモリーの量を制限する際に、UDS はできるだけ効率的にデータのインデックスを作成します。重複排除ウィンドウのバインドされた特徴にもかかわらず、重複排除レベルが高い多くのデータセットでは、一時的な局所性 も多く確認されます。つまり、多くの重複排除は、ほぼ同時に書き込まれたブロックセット間で発生します。さらに、通常、書き込まれたデータは、以前に書き込まれたデータよりも、最近書き込まれたデータを複製する可能性が高くなります。したがって、特定の時間間隔での特定のワークロードでは、UDS が最新のデータのみをインデックス付けしても、すべてのデータをインデックス付けしても、重複排除率は同じになることがよくあります。

重複データの場合では、一時的な局所性が示される傾向もあるため、ストレージシステム内のすべてのブロックにインデックスを作成する必要はほとんどありません。そうでない場合、インデックスメモリーのコストは、重複排除によるストレージコストの削減を上回ります。インデックスサイズの要件

は、データの摂取率に密接に関連します。たとえば、合計容量が 100 TB のストレージシステムには、毎週 1 TB の摂取率を指定します。UDS は、4 TB の重複排除ウィンドウにより、前の月に書き込まれたデータで、最も冗長性が大きいものを検出できます。

UDS の スパースインデックス 機能 (VDO に推奨されるモード) は、メモリー内で最も関連するインデックスエントリーのみを保持しようとすることで、時間的な局所性をさらに悪用します。UDS は、同じ量のメモリーを使用しながら、10 倍以上長い重複排除ウィンドウを維持できます。スパースインデックスが最大のカバレッジを提供しますが、デンスインデックスはより多くのアドバイスを提供します。ほとんどのワークロードでは、メモリー量が同じであれば、dense インデックスと sparse インデックスの重複排除率の差はごくわずかです。

インデックスに必要なメモリーは、重複排除ウィンドウに必要なサイズで決定されます。

- *デンスインデックスの場合、UDS は、RAM 1GB あたり 1TB の重複排除ウィンドウを提供します。通常、最大 4TB のストレージシステムには 1GB のインデックスで十分です。*
- *スパースインデックスの場合、UDS は、RAM 1GB あたり 10TB の重複排除ウィンドウを提供します。通常、最大 40 TB の物理ストレージには、1 GB の sparse インデックスで十分です。*

UDS インデックスのメモリー要件の具体例は、「物理ボリュームのサイズ別の VDO システム要件の例」 を参照してください。

30.2.2. VDO ストレージ領域の要件

VDO には、VDO メタデータ用と、実際の UDS 重複排除インデックス用の両方のストレージ領域が必要です。

- *VDO は、2 種類のメタデータを基盤となる物理ストレージに書き込みます。*
  - *最初のタイプは、VDO ボリュームの物理サイズでスケーリングし、物理ストレージの 4GB ごとに約 1MB を使用し、スラブごとにさらに 1MB を使用します。*
  - *2 番目のタイプは、VDO ボリュームの論理サイズでスケーリングされ、論理ストレージが 1GB ごとに約 1.25MB を消費し、最も近いスラブに切り上げられます。*

スラブの詳細は、「VDO ボリューム」 を参照してください。

- UDS インデックスは VDO ボリュームグループ内に保存され、関連付けられた VDO インスタンスによって管理されます。必要なストレージの量は、インデックスのタイプと、インデックスに割り当てられている RAM の量により異なります。RAM 1GB ごとに、デンス UDS インデックスはストレージを 17GB 使用し、スパース UDS インデックスはストレージを 170GB 使用します。

VDO ストレージ要件の具体例は、「物理ボリュームのサイズ別の VDO システム要件の例」 を参照してください。

30.2.3. 物理ボリュームのサイズ別の VDO システム要件の例

以下の表は、基となる物理ボリュームのサイズに基づいた、VDO のシステム要件の概算を示しています。それぞれの表には、プライマリーストレージ、バックアップストレージなどの、目的のデプロイメントに適した要件が記載されています。

正確な数値は、VDO ボリュームの設定により異なります。

プライマリーストレージのデプロイメント

プライマリーストレージの場合、UDS インデックスのサイズは、物理ボリュームの 0.01% から 25% になります。

表30.2 プライマリーストレージの VDO ストレージおよびメモリーの要件

物理ボリュームのサイズ	10 GB - 1-TB	2–10 TB	11–50 TB	51–100 TB	101–256 TB
RAM の使用	250 MB	デンス: 1 GB スパース: 250 MB	2 GB	3 GB	12 GB
ディスクの使用率	2.5 GB	デンス: 10 GB スパース: 22 GB	170 GB	255 GB	1020 GB
インデックスタイプ	デンス	デンスまたはスパース	スパース	スパース	スパース

バックアップストレージのデプロイメント

バックアップストレージの場合、UDS インデックスは、バックアップセットのサイズよりは大きくなりますが、物理ボリュームと同じか、より小さくなります。バックアップセットや物理サイズが今後大きくなる可能性がある場合は、これをインデックスサイズに組み込んでください。

表30.3 バックアップストレージの VDO ストレージおよびメモリーの要件

物理ボリュームのサイズ	10 GB – 1 TB	2–10 TB	11–50 TB	51–100 TB	101–256 TB
RAM の使用	250 MB	2 GB	10 GB	20 GB	26 GB
ディスクの使用率	2.5 GB	170 GB	850 GB	1700 GB	3400 GB
インデックスタイプ	デンス	スパース	スパース	スパース	スパース

30.3. VDO の使用

30.3.1. 導入部分

VDO (Virtual Data Optimizer) は、重複排除、圧縮、およびシンプロビジョニングの形で、Linux でインラインのデータ削減を行います。VDO ボリュームを設定する場合は、VDO ボリュームを構築するブロックデバイスと、作成する論理ストレージのサイズを指定します。

- *アクティブな仮想マシンまたはコンテナーをホストする場合、Red Hat は、論理と物理の割合が 10 対 1 のストレージをプロビジョニングすることを推奨します。つまり、物理ストレージを 1 TB 使用している場合は、論理ストレージを 10 TB にします。*
- *Ceph が提供するタイプなどのオブジェクトストレージの場合、Red Hat では、論理対物理の比率を 3:1 にすることを推奨しています。つまり、物理ストレージが 1TB の場合は、論理ストレージを 3TB にします。*

いずれの場合も、VDO が作成する論理デバイスにファイルシステムを置くだけで、直接使用することも、分散クラウドストレージアーキテクチャーの一部として使用することもできます。

本章では、VDO デプロイメントの以下のユースケースについて説明します。

-

Red Hat Virtualization を使用して構築したサーバーなどの仮想サーバー用の直接接続のユースケース

- *Ceph Storage を使用して構築したものなど、オブジェクトベースの分散ストレージクラスターのクラウドストレージのユースケース。*

注記

Ceph を使用した VDO デプロイメントは現在サポートされていません。

本章では、いずれのユースケースにも簡単にデプロイできる標準の Linux ファイルシステムで使用するように VDO を設定する例を説明します。「デプロイメントの例」 の図を参照してください。

30.3.2. VDO のインストール

VDO は、以下の RPM パッケージを使用してデプロイされます。

- *vdo*
- *kmod-kvdo*

VDO をインストールするには、yum パッケージマネージャーを使用して RPM パッケージをインストールします。

```
# yum install vdo kmod-kvdo
```

30.3.3. VDO ボリュームの作成

ブロックデバイス用の VDO ボリュームを作成します。複数の VDO ボリュームを、同じマシン上のデバイスごとに作成できることに注意してください。このアプローチを選択する場合は、システムの VDO のインスタンスごとに異なる名前とデバイスを指定する必要があります。

重要

拡張可能なストレージをバッキングブロックデバイスとして使用している。詳細は、「システム要件」 を参照してください。

以下の手順では、vdo_name を、VDO ボリュームに使用する識別子(vdo1 など)に置き換えます。

1. VDO Manager を使用して VDO ボリュームを作成します。

```
# vdo create \
      --name=vdo_name \
      --device=block_device \
      --vdoLogicalSize=logical_size \
      [--vdoSlabSize=slab_size]
```

- block_device を、VDO ボリュームを作成するブロックデバイスの永続名に置き換えます。たとえば、/dev/disk/by-id/scsi-3600508b1001c264ad2af21e903ad031f などです。

  重要

  永続的なデバイス名を使用します。永続的なデバイス名を使用しないと、今後デバイスの名前が変わった場合に、VDO が正しく起動しなくなることがあります。

  永続的な名前の詳細は、「永続的な命名」 を参照してください。

- logical_size を、VDO ボリュームが存在する論理ストレージの容量に置き換えます。

  - アクティブな仮想マシンまたはコンテナーストレージには、ブロックデバイスの物理サイズの 10 倍の論理サイズを使用します。たとえば、ブロックデバイスのサイズが 1 TB の場合は、10T を使用します。

  - オブジェクトのストレージには、ブロックデバイスの物理サイズの 3 倍の論理サイズを使用します。たとえば、ブロックデバイスのサイズが 1 TB の場合は、3T を使用します。

- ブロックデバイスが 16 TiB を超える場合は、--vdoSlabSize=32G を追加して、ボリューム上のスラブサイズを 32 GiB に増やします。

  16 TiB を超えるブロックデバイスでデフォルトのスラブサイズ 2 GiB を使用すると、vdo create コマンドが失敗し、以下のエラーが発生します。

  ```
  vdo: ERROR - vdoformat: formatVDO failed on '/dev/device': VDO Status: Exceeds maximum number of slabs supported
  ```

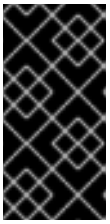
  詳細は、「VDO ボリューム」 を参照してください。

例30.1 コンテナーストレージ用の VDO の作成

たとえば、1TB のブロックデバイスにコンテナーストレージ用の VDO ボリュームを作成する場合は、次のコマンドを使用します。

```
# vdo create \
      --name=vdo1 \
      --device=/dev/disk/by-id/scsi-3600508b1001c264ad2af21e903ad031f \
      --vdoLogicalSize=10T
```

VDO ボリュームが作成されると、VDO はエントリーを /etc/vdoconf.yml 設定ファイルに追加します。次に、vdo.service systemd ユニットは、エントリーを使用して、デフォルトでボリュームを起動します。



重要

VDO ボリュームの作成中に問題が発生した場合は、ボリュームを削除してください。詳細は 「作成に失敗したボリュームの削除」 を参照してください。

2. ファイルシステムを作成します。

- XFS ファイルシステムの場合:

  ```
  # mkfs.xfs -K /dev/mapper/vdo_name
  ```

- ***ext4 ファイルシステムの場合:***

```
# mkfs.ext4 -E nodiscard /dev/mapper/vdo_name
```

3. ***ファイルシステムをマウントします。***

```
# mkdir -m 1777 /mnt/vdo_name
# mount /dev/mapper/vdo_name /mnt/vdo_name
```

4. ***ファイルシステムが自動的にマウントされるように設定するには、/etc/fstab ファイルまたは systemd マウントユニットを使用します。***

- ***/etc/fstab 設定ファイルを使用する場合は、以下のいずれかの行をファイルに追加します。***
  - ***XFS ファイルシステムの場合:***

```
/dev/mapper/vdo_name /mnt/vdo_name xfs defaults,_netdev,x-systemd.device-timeout=0,x-systemd.requires=vdo.service 0 0
```

  - ***ext4 ファイルシステムの場合:***

```
/dev/mapper/vdo_name /mnt/vdo_name ext4 defaults,_netdev,x-systemd.device-timeout=0,x-systemd.requires=vdo.service 0 0
```

- ***systemd ユニットを使用する場合は、適切なファイル名で systemd マウントユニットファイルを作成します。VDO ボリュームのマウントポイントには、以下の内容で /etc/systemd/system/mnt-vdo_name.mount ファイルを作成します。***

```
[Unit]
Description = VDO unit file to mount file system
name = vdo_name.mount
Requires = vdo.service
After = multi-user.target
Conflicts = umount.target

[Mount]
What = /dev/mapper/vdo_name
Where = /mnt/vdo_name
Type = xfs
```

```
[Install]
WantedBy = multi-user.target
```

systemd ユニットファイルの例は、/usr/share/doc/vdo/examples/systemd/VDO.mount.example にもインストールされます。

5. **VDO デバイスのファイルシステムで discard 機能を有効にします。バッチ操作とオンライン操作の両方が VDO で機能します。**

discard 機能の設定方法の詳細は、「未使用ブロックの破棄」 を参照してください。

30.3.4. VDO の監視

VDO はシンプロビジョニングされているため、ファイルシステムとアプリケーションは、使用中の論理領域のみを認識し、実際に利用可能な物理領域は認識しません。

VDO 領域の使用量と効率は、vdostats ユーティリティーを使用して監視できます。

```
# vdostats --human-readable

Device                 1K-blocks    Used     Available    Use%    Space saving%
/dev/mapper/node1osd1    926.5G      21.0G    905.5G      2%      73%
/dev/mapper/node1osd2    926.5G      28.2G    898.3G      3%      64%
```

VDO ボリュームの物理ストレージ領域が不足しそうになると、VDO は、システムログに、以下のような警告を出力します。

```
Oct  2 17:13:39 system lvm[13863]: Monitoring VDO pool vdo_name.
Oct  2 17:27:39 system lvm[13863]: WARNING: VDO pool vdo_name is now 80.69% full.
Oct  2 17:28:19 system lvm[13863]: WARNING: VDO pool vdo_name is now 85.25% full.
Oct  2 17:29:39 system lvm[13863]: WARNING: VDO pool vdo_name is now 90.64% full.
Oct  2 17:30:29 system lvm[13863]: WARNING: VDO pool vdo_name is now 96.07% full.
```

重要

VDO ボリュームで物理領域を監視し、領域不足を回避します。物理ブロックが不足すると、VDO ボリュームに最近書き込まれたデータや、未承認のデータが失われることがあります。

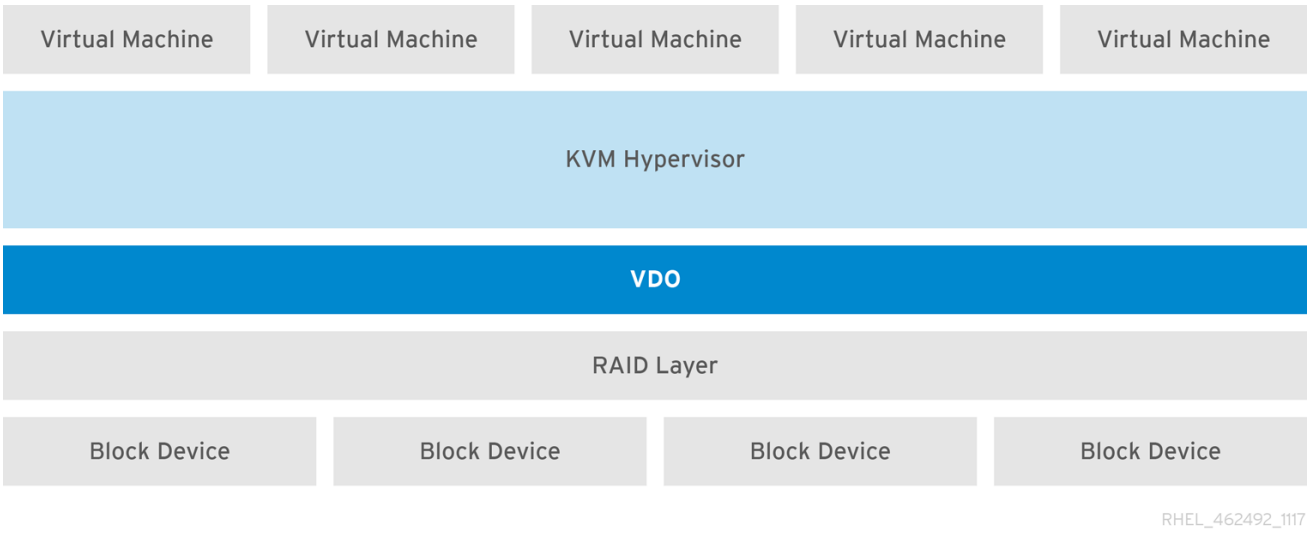
30.3.5. デプロイメントの例

以下の例は、KVM およびその他のデプロイメントで VDO を使用する方法を示しています。

KVM を使用した VDO デプロイメント

Direct Attached Storage で設定された KVM サーバーに VDO をデプロイする方法は、図 30.2「KVM を使用した VDO デプロイメント」 を参照してください。

図30.2 KVM を使用した VDO デプロイメント



その他のデプロイメントシナリオ

VDO デプロイメントの詳細は、「デプロイメントシナリオ」 を参照してください。

30.4. VDO の管理

30.4.1. VDO の起動または停止

特定の VDO ボリューム、またはすべての VDO ボリューム、および関連する UDS インデックスを開始するには、ストレージ管理ユーティリティーは以下のいずれかのコマンドを呼び出す必要があります。

```
# vdo start --name=my_vdo
# vdo start --all
```

VDO systemd ユニットは、vdo パッケージのインストール時にインストールされ、デフォルトで有効になっています。本ユニットは、システムの起動時に vdo start --all コマンドを自動的に実行し

て、アクティブな VDO ボリューム を すべて起動します。詳細は、「システム起動時に VDO ボリュームを自動的に起動する」 を参照してください。

特定の VDO ボリューム、またはすべての VDO ボリューム、および関連する UDS インデックスを停止するには、以下のいずれかのコマンドを使用します。

```
# vdo stop --name=my_vdo
# vdo stop --all
```

VDO ボリュームの停止にかかる時間は、ストレージデバイスの速度と、ボリュームへの書き込みが必要なデータ量により異なります。

- ボリュームは、UDS インデックスの 1GiB ごとに、約 1 GiB のデータを常に書き込みます。
- スパースの UDS インデックスを使用すると、ボリュームはブロックマップキャッシュのサイズと、スラブごとに最大 8MiB のデータ量を書き込みます。

シャットダウンが正常に行われずに再起動した場合、VDO は再構築を実行してメタデータの整合性を検証し、必要に応じて修復します。再構築は自動で行われ、ユーザーの介入は必要ありません。再ビルドプロセスの詳細は、「シャットダウンが適切に行われない場合の VDO ボリュームの復旧」 を参照してください。

VDO は、書き込みモードによって異なる書き込みを再構築する可能性があります。

- 同期モードでは、シャットダウンの前に VDO により確認されたすべての書き込みが再構築されます。
- 非同期モードでは、最後に確認されたフラッシュ要求の前に確認されたすべての書き込みが再構築されます。

いずれのモードでも、承認されていない書き込み、またはフラッシュが続いていない書き込みのいずれかが行われた場合は、再構築されることがあります。

VDO 書き込みモードの詳細は、「VDO 書き込みモードの選択」 を参照してください。

30.4.2. VDO 書き込みモードの選択

VDO は、sync、async、および auto の 3 つの書き込みモードに対応します。

- VDO が sync モードの場合、上記のレイヤーは write コマンドが永続ストレージにデータを書き込むことを想定します。したがって、このモードは、ファイルシステムやアプリケーションには必要ありません。FLUSH リクエストまたは FUA (強制ユニットアクセス) リクエストを発行すると、データは、重要な点で持続します。

  VDO は、書き込みコマンドの完了時にデータが永続ストレージに書き込まれることを保証する場合にのみ sync mode に設定する必要があります。つまり、ストレージには揮発性の書き込みキャッシュがないか、ライトスルーキャッシュが存在する必要があります。

- VDO が async モードの場合、書き込みコマンドが確認されると、データが永続ストレージに書き込まれる保証はありません。ファイルシステムまたはアプリケーションは、各トランザクションの重要な点でデータの永続性を保証するために、FLUSH リクエストまたは FUA リクエストを発行する必要があります。

  write コマンドの完了時に、基礎となるストレージが永続ストレージにデータを書き込むことを保証しない場合は、VDO を async モードに設定する必要があります。つまり、ストレージに揮発性のライトバックキャッシュがある場合です。

  デバイスが揮発性キャッシュを使用するかどうかを確認する方法は、「揮発性キャッシュの確認」 を参照してください。

警告

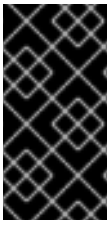
VDO が 非同期 モードで実行されている場合は、ACID (Atomicity, Consistency, Isolation, Durability)に準拠していません。VDO ボリュームの上に ACID コンプライアンスを想定するアプリケーションまたはファイルシステムがある場合、async モードにより予期しないデータ損失が発生する可能性があります。

- auto モードは、各デバイスの特性に基づいて sync または async を自動的に選択します。以下はデフォルトのオプションになります。

書き込みポリシーの動作方法のより詳細な理論的概要については、「VDO 書き込みポリシーの概要」 を参照してください。

書き込みポリシーを設定するには、**--writePolicy** を使用します。これは、「VDO ボリュームの作成」 のように **VDO** ボリュームを作成する場合、または **changeWritePolicy** サブコマンドを使用して既存の **VDO** ボリュームを変更する場合に指定できます。

```
# vdo changeWritePolicy --writePolicy=sync|async|auto --name=vdo_name
```



重要

誤った書き込みポリシーを使用すると、電源障害時にデータが失われる可能性があります。

揮発性キャッシュの確認

デバイスにライトバックキャッシュがあるかどうかを確認するには、**/sys/block/block_device/device/scsi_disk/identifier/cache_type sysfs** ファイルを読み込みます。以下に例を示します。

- デバイス **sda** には、ライトバックキャッシュ が あることを示します。

```
$ cat '/sys/block/sda/device/scsi_disk/7:0:0:0/cache_type'

write back
```

- デバイス **sdb** には、ライトバックキャッシュがない ことを示します。

```
$ cat '/sys/block/sdb/device/scsi_disk/1:2:0:0/cache_type'

None
```

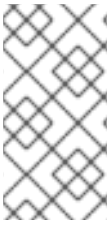
また、カーネルブートログでは、上記のデバイスに書き込みキャッシュがあるかどうかを確認できます。

```
sd 7:0:0:0: [sda] Write cache: enabled, read cache: enabled, doesn't support DPO or FUA
sd 1:2:0:0: [sdb] Write cache: disabled, read cache: disabled, supports DPO and FUA
```

システムログの読み取りに関する詳細は、『システム管理者のガイド』のViewing and Managing Log Filesの章を参照してください。

この例では、VDO に次の書き込みポリシーを使用します。

- sda デバイスの async モード
- sdb デバイスの 同期 モード



注記

cache_type の値が none または で書き込みを 行う 場合は、sync 書き込みポリシーを使用するように VDO を設定する必要があります。

30.4.3. VDO ボリュームの削除

VDO ボリュームは、次のコマンドを実行するとシステムから削除できます。

```
# vdo remove --name=my_vdo
```

VDO ボリュームを削除する前に、ファイルシステムのマウントを解除し、ストレージを使用しているアプリケーションを停止します。vdo remove コマンドは、VDO ボリュームとそれに関連する UDS インデックス、およびそれらが存在する論理ボリュームを削除します。

30.4.3.1. 作成に失敗したボリュームの削除

vdo ユーティリティーが VDO ボリュームを作成しているときに障害が発生した場合、ボリュームは中間状態のままになります。これは、システムがクラッシュしたり、電源に障害が発生したり、管理者が実行中の vdo create コマンドを中断したりした場合に発生する可能性があります。

この状況からクリーンアップするには、--force オプションを使用して、作成に失敗したボリュームを削除します。

```
# vdo remove --force --name=my_vdo
```

ボリュームの作成に失敗して、管理者がシステム設定を変更して競合を発生させたため、--force

オプションが必要となります。--force オプションを指定しないと、vdo remove コマンドが失敗し、以下のメッセージが表示されます。

```
[...]
A previous operation failed.
Recovery from the failure either failed or was interrupted.
Add '--force' to 'remove' to perform the following cleanup.
Steps to clean up VDO my_vdo:
umount -f /dev/mapper/my_vdo
udevadm settle
dmsetup remove my_vdo
vdo: ERROR - VDO volume my_vdo previous operation (create) is incomplete
```

30.4.4. UDS インデックスの設定

VDO は、UDS と呼ばれる高パフォーマンスの重複排除インデックスを使用して、格納されているときにデータの重複ブロックを検出します。重複排除ウィンドウ は、インデックスが記憶している以前に書き込まれたブロックの数です。重複排除ウィンドウのサイズは設定可能です。特定のウィンドウサイズに対して、インデックスには特定の量の RAM と特定の量のディスク容量が必要です。ウィンドウのサイズは、通常、--indexMem=size オプションを使用してインデックスメモリーのサイズを指定して決定されます。使用するディスク領域は、自動的に決定されます。

一般的には、Red Hat は、すべての実稼働環境に sparse の UDS インデックスを使用することを推奨します。これは、非常に効率的なインデックスデータ構造であり、重複排除ウィンドウでブロックごとに DRAM の約 10 分の 1 バイトが必要です。ディスクの場合は、ブロックごとに約 72 バイトのディスク領域が必要です。このインデックスの最小設定は、ディスクで 256 MB の DRAM と約 25 GB の領域を使用します。この設定を使用するには、--sparseIndex=enabled --indexMem=0.25 オプションを vdo create コマンドに指定します。この設定により、2.5 TB の重複排除ウィンドウが作成されます (つまり、2.5 TB の履歴が記録されます)。ほとんどのユースケースでは、2.5 TB の重複排除ウィンドウは、サイズが 10 TB までのストレージプールを重複排除するのに適しています。

ただし、インデックスのデフォルト設定は、dense のインデックスを使用します。このインデックスは DRAM では非常に効率が悪い (10 倍) ですが、必要なディスク領域もはるかに少ない (10 倍) ため、制約された環境での評価に便利です。

一般に、VDO ボリュームの物理サイズの 4 分の 1 である重複排除ウィンドウが推奨される設定です。ただし、これは実際の要件ではありません。小さな重複排除ウィンドウ (物理ストレージの量と比較) であっても、多くのユースケースで大量の重複データを確認できます。大概のウィンドウも使用できますが、ほとんどの場合、それを行う利点はほとんどありません。

この重要なシステムパラメーターの調整に関する追加のガイドラインについては、Red Hat テクニカルアカウントマネージャーの担当者にご相談ください。

30.4.5. シャットダウンが適切に行われない場合の VDO ボリュームの復旧

ボリュームを正常にシャットダウンせずに再起動した場合、VDO はメタデータの一部を再構築して操作を継続する必要があります。これは、ボリュームの起動時に自動的に行われます。(正常にシャットダウンしたボリュームでこのプロセスを呼び出す場合は、「再構築の強制」 も併せて参照してください。)

データの復元は、デバイスの書き込みポリシーによって異なります。

- VDO が同期ストレージで実行され、書き込みポリシーが を 同期 すると、ボリュームに書き込まれたすべてのデータが完全に復元されます。

- 書き込みポリシーが async の場合、一部の書き込みは、VDO に FLUSH コマンドを送信するか、FUA フラグでタグ付けされた書き込み I/O （ユニットアクセスの実施）を行っても、書き込みが永続化されない場合に復元されないことがあります。これは、fsync、fdatasync、sync、または umount などのデータ整合性操作を呼び出すことで、ユーザーモードから実行されます。

30.4.5.1. オンライン復旧

ほとんどの場合、クリーンでない VDO ボリュームの再構築作業のほとんどは、VDO ボリュームがオンラインに戻ってから、読み取りおよび書き込みの要求の処理中に行うことができます。最初は、書き込み要求に使用できるスペースの量が制限されている場合があります。ボリュームのメタデータがより多く回復されると、より多くの空き領域を利用できる可能性があります。さらに、VDO の回復中に書き込まれたデータは、そのデータがまだ回復されていないボリュームの一部にある場合、クラッシュ前に書き込まれたデータに対して重複排除に失敗する可能性があります。ボリュームの回復中は、データが圧縮されている可能性があります。以前に圧縮したブロックは、読み取りまたは上書きできます。

オンラインリカバリー中は、使用中のブロックやブロックの 空きブロック など、いくつかの統計が利用できなくなります。この統計は、再構築が完了すると利用可能になります。

30.4.5.2. 再構築の強制

VDO は、ほとんどのハードウェアエラーおよびソフトウェアエラーから回復できます。VDO ボリュームを正常に回復できない場合は、ボリュームの再起動後も持続する読み取り専用モードになります。ボリュームが読み取り専用モードの場合、データが損失または破損していないという保証はありません。このような場合、Red Hat は、読み取り専用のボリュームからデータをコピーして、バックアップからボリュームを復旧することを推奨します。(vdostats の operating mode 属性は、VDO ボリュームが読み取り専用モードかどうかを示します。)

データ破損のリスクを許容できる場合は、VDO ボリュームのメタデータのオフライン再構築を強制することで、ボリュームをオンラインに戻して利用できるようにできます。なお、再構築したデータ

の整合性は保証できません。

読み取り専用 VDO ボリュームの再構築を強制するには、ボリュームが実行されている場合は、最初にボリュームを停止します。

```
# vdo stop --name=my_vdo
```

次に、**--forceRebuild** オプションを使用してボリュームを再起動します。

```
# vdo start --name=my_vdo --forceRebuild
```

30.4.6. システム起動時に VDO ボリュームを自動的に起動する

システムの起動時に、**vdo systemd** ユニットは、アクティブ化 された として設定されているすべての **VDO** デバイスを自動的に起動します。

特定のボリュームが自動的に起動しないようにするには、以下のいずれかのコマンドを実行してそのボリュームを非アクティブ化します。

- 特定のボリュームを無効にするには、以下のコマンドを実行します。

```
# vdo deactivate --name=my_vdo
```

- すべてのボリュームを無効にするには、以下のコマンドを実行します。

```
# vdo deactivate --all
```

ボリュームをアクティブにするには、以下のいずれかのコマンドを実行します。

- 特定のボリュームを有効にするには、以下のコマンドを実行します。

```
# vdo activate --name=my_vdo
```

- すべてのボリュームを有効にするには、以下のコマンドを実行します。

```
# vdo activate --all
```

vdo create コマンドに --activate=disabled オプションを追加して、自動的に起動しない VDO ボリュームを作成することもできます。

VDO ボリュームの上や下に LVM ボリュームを置くシステム (図30.5「重複排除された統合ストレージ」 など) では、サービスを適切な順序で起動することが重要です。

1. LVM の下位層を最初に起動する必要があります (ほとんどのシステムでは、LVM2 パッケージのインストール時に、この層の起動が自動的に設定されます)。

2. その後、vdo systemd ユニットを起動する必要があります。

3. 最後に、現在実行している VDO ボリュームに、LVM ボリュームやその他のサービスを起動するために、追加のスクリプトを実行する必要があります。

30.4.7. 重複排除の無効化と再有効化

一部の例では、ボリュームへの読み書きを行う機能を維持しながら、VDO ボリュームに書き込まれているデータの重複排除を一時的に無効にすることが望ましい場合があります。重複排除を無効にすると、後続の書き込みが重複排除されなくなりますが、すでに重複排除されているデータはそのまま残ります。

- VDO ボリュームでの重複排除を停止するには、以下のコマンドを使用します。

```
# vdo disableDeduplication --name=my_vdo
```

これにより、関連付けられている UDS インデックスが停止し、重複排除がアクティブでないことを VDO ボリュームに通知します。

- VDO ボリュームでの重複排除を再起動するには、以下のコマンドを使用します。

```
# vdo enableDeduplication --name=my_vdo
```

これにより、関連する UDS インデックスが再起動し、重複排除が再度アクティブであることを VDO ボリュームに通知します。

--deduplication=disabled オプションを vdo create コマンドに追加することで、新しい VDO ボリュームを作成するときに重複排除を無効にすることもできます。

30.4.8. 圧縮の使用

30.4.8.1. 導入部分

ブロックレベルの重複排除のほかに、VDO では、HIOPS Compression™ テクノロジーを使用したインラインブロックレベルの圧縮も提供されます。重複排除は、仮想マシン環境とバックアップアプリケーションに最適なソリューションですが、圧縮は、通常、ログファイルやデータベースなどのブロックレベルの冗長性を見せない構造化および非構造化のファイル形式に非常に適しています。

圧縮は、重複として認識されていないブロックで動作します。一意のデータが初めて検出されると、それが圧縮されます。その後の、保存しているデータのコピーは、追加の圧縮手順なしに重複排除されます。圧縮機能は、同時に多くの圧縮操作に対応できるようにする並列化されたパッケージアルゴリズムに基づいています。最初にブロックを保存し、リクエストに応答したら、圧縮時に複数のブロックを検出する最適なパックアルゴリズムが、1 つの物理ブロックに適合します。特定の物理ブロックが追加の圧縮ブロックを保持していないと判断すると、そのブロックはストレージに書き込まれます。圧縮されていないブロックは解放され、再利用されます。要求したものに応答し、圧縮およびパッケージ化の操作を実行することにより、圧縮を使用することで課せられるレイテンシーが最小限に抑えられます。

30.4.8.2. 圧縮の有効化と無効化

VDO ボリューム圧縮はデフォルトでオンになっています。

ボリュームの作成時に、--compression=disabled オプションを vdo create コマンドに追加することで圧縮を無効にできます。

パフォーマンスを最大化するため、または圧縮される可能性が低いデータの処理を高速化するために、必要に応じて既存の VDO ボリュームで圧縮を停止できます。

- VDO ボリュームで圧縮を停止するには、次のコマンドを使用します。

```
# vdo disableCompression --name=my_vdo
```

- 再び起動するには、以下のコマンドを使用します。

```
# vdo enableCompression --name=my_vdo
```

30.4.9. 空き領域の管理

VDO はシンプロビジョニングのブロックストレージターゲットであるため、VDO が使用する物理領域は、ストレージのユーザーに提示されるボリュームのサイズと異なる場合があります。インテグレーターとシステム管理者は、この格差を利用してストレージコストを節約できますが、書き込まれたデータが期待される重複排除率を達成しない場合は、ストレージスペースが予期せず不足しないように注意する必要があります。

論理ブロック (仮想ストレージ) の数が物理ブロック (実際のストレージ) の数を超えると、ファイルシステムおよびアプリケーションで領域が予想外に不足する可能性があります。このため、VDO を使用するストレージシステムは、ストレージ管理者に、VDO の空きプールのサイズを監視する方法で提供する必要があります。この空きプールのサイズは、vdostats ユーティリティーを使用して決定できます。詳細は、「vdostats」 を参照してください。このユーティリティーのデフォルト出力には、Linux df ユーティリティーと同様の形式で、実行中のすべての VDO ボリュームの情報が一覧表示されます。以下に例を示します。

```
Device              1K-blocks   Used       Available   Use%
/dev/mapper/my_vdo  211812352   105906176  105906176     50%
```

VDO ボリュームの物理ストレージ領域が不足しそうになると、VDO は、システムログに、以下のような警告を出力します。

```
Oct  2 17:13:39 system lvm[13863]: Monitoring VDO pool my_vdo.
Oct  2 17:27:39 system lvm[13863]: WARNING: VDO pool my_vdo is now 80.69% full.
Oct  2 17:28:19 system lvm[13863]: WARNING: VDO pool my_vdo is now 85.25% full.
Oct  2 17:29:39 system lvm[13863]: WARNING: VDO pool my_vdo is now 90.64% full.
Oct  2 17:30:29 system lvm[13863]: WARNING: VDO pool my_vdo is now 96.07% full.
```

VDO の空きプールのサイズが特定のレベルを下回った場合、ストレージ管理者はデータの削除 (削除されたデータが重複しない限り領域を確保)、物理ストレージの追加、または LUN の削除を行うことで対応できます。

重要

VDO ボリュームで物理領域を監視し、領域不足を回避します。物理ブロックが不足すると、VDO ボリュームに最近書き込まれたデータや、未承認のデータが失われることがあります。

ファイルシステムの領域の回収

ファイルシステムが、DISCARD、TRIM、または UNMAP コマンドを使用してブロックが解放されていることを伝えない限り、VDO は領域を回収できません。DISCARD、TRIM、または UNMAP を使用しないファイルシステムの場合、バイナリーゼロで設定されるファイルを保存し、そのファイルを削除することで、空き領域を手動で再利用できます。

ファイルシステムは、通常、次の 2 つの方法のいずれかで DISCARD 要求を発行するように設定できます。

リアルタイム廃棄 (オンライン廃棄またはインライン廃棄)

リアルタイム破棄を有効にすると、ユーザーがファイルを削除して領域を解放するたびに、ファイルシステムはブロック層に REQ_DISCARD 要求を送信します。VDO はこの要求を受け取り、ブロックが共有されていないことを前提として、空きプールに領域を戻します。

オンライン破棄に対応するファイルシステムの場合、マウント時に discard オプションを設定することで有効にできます。

バッチ破棄

バッチ破棄は、ユーザーが開始する操作であり、ファイルシステムが未使用のブロックをブロックレイヤー (VDO) に通知します。これは、ファイルシステムに FITRIM と呼ばれる ioctl 要求を送信することで実現されます。

fstrim ユーティリティー(cronなど)を使用して、この ioctl をファイルシステムに送信できます。

discard 機能の詳細は、「未使用ブロックの破棄」 を参照してください。

ファイルシステムがない場合の領域の確保

ストレージがファイルシステムなしでブロックストレージのターゲットとして使用されている場合は、空き領域を管理することもできます。たとえば、単一の VDO ボリュームは、その上に論理ボリュームマネージャー (LVM) をインストールすることにより、複数のサブボリュームに分割できます。ボリュームのプロビジョニングを解除する前に、blkdiscard コマンドを使用して、その論理ボリュームで以前に使用された領域を解放できます。LVM は REQ_DISCARD コマンドに対応し、領域を解放するために適切な論理ブロックアドレスで VDO に要求を転送します。他のボリュームマネージャーを使用している場合は、REQ_DISCARD もサポートする必要があります。もしくは、SCSI デバイスの場合は UNMAP、または ATA デバイスの場合は TRIM をサポートする必要もあります。

ファイバーチャネルまたはイーサネットネットワーク上の領域の確保

VDO ボリューム (またはボリュームの一部) は、LIO または SCST などの SCSI ターゲットフレームワークを使用して、ファイバーチャネルストレージファブリックまたはイーサネットネットワークのホストにプロビジョニングすることもできます。SCSI イニシエーターは UNMAP コマンドを使用して、シンプロビジョニングされたストレージターゲット上の領域を解放できますが、SCSI ターゲットフレームワークは、このコマンドのサポートをアドバタイズするように設定する必要があります。通常、これはこのボリュームで シンプロビジョニング を有効にすることで行われます。UNMAP への対応は、Linux ベースの SCSI イニシエーターで確認できます。

```
# sg_vpd --page=0xb0 /dev/device
```

出力で、"Maximum unmap LBA count" の値がゼロより大きいことを確認します。

30.4.10. 論理ボリュームのサイズの拡大

管理アプリケーションは、vdo growLogical サブコマンドを使用して、VDO ボリュームの論理サイズを増やすことができます。ボリュームが大きくなったら、管理者は VDO ボリュームの上にあるすべてのデバイスまたはファイルシステムに新しいサイズを通知する必要があります。ボリュームは次のように増やすことができます。

```
# vdo growLogical --name=my_vdo --vdoLogicalSize=new_logical_size
```

このコマンドを使用すると、ストレージ管理者は最初に、領域が不足しないように、論理サイズが十分に小さい VDO ボリュームを作成できます。しばらくすると、実際のデータ削減率を評価でき、これが十分な場合には、VDO ボリュームの論理サイズを拡張して、削減して得られた領域を活用することができます。

30.4.11. 物理ボリュームのサイズの拡大

VDO ボリュームで利用可能な物理ストレージの量を増やすには、以下のコマンドを実行します。

1. *基になるデバイスのサイズを大きくします。*

   正確な手順は、デバイスの種類によって異なります。たとえば、MBR パーティションのサイズを変更するには、「fdisk でのパーティションのサイズ変更」 の説明に従って fdisk ユーティリティーを使用します。

2.

growPhysical
を使用して、新しい物理ストレージ領域を VDO ボリュームに追加します。

```
# vdo growPhysical --name=my_vdo
```

このコマンドでは、VDO ボリュームを縮小できません。

30.4.12. Ansible を使用した VDO の自動化

Ansible ツールを使用することで、VDO デプロイメントと管理を自動化できます。詳細は、以下を参照してください。

- *Ansible ドキュメント: https://docs.ansible.com/*
- *VDO Ansible モジュールのドキュメント*
  https://docs.ansible.com/ansible/latest/modules/vdo_module.html

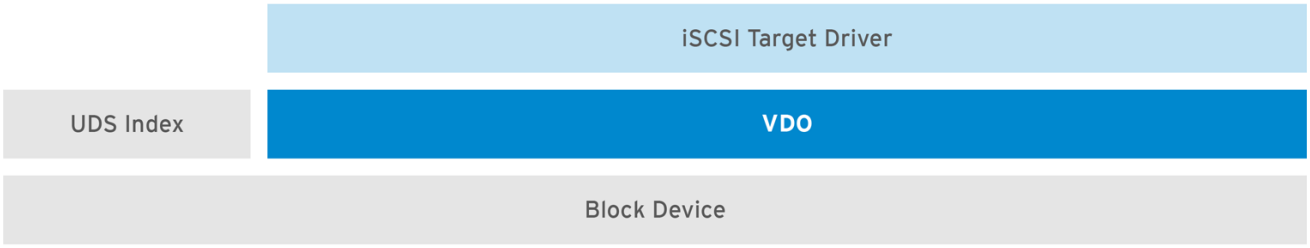
30.5. デプロイメントシナリオ

VDO はさまざまな方法でデプロイして、ブロックアクセスとファイルアクセスの両方、およびローカルストレージとリモートストレージの両方に重複排除されたストレージを提供できます。VDO は、重複排除されたストレージを標準の Linux ブロックデバイスとして公開するため、標準のファイルシステム、iSCSI ターゲットドライバー、および FC ターゲットドライバー、または統合ストレージとして使用できます。

30.5.1. iSCSI ターゲット

簡単な例として、VDO ストレージターゲット全体を、iSCSI ターゲットとしてリモートの iSCSI イニシエーターにエクスポートできます。

図30.3 重複排除したブロックストレージターゲット

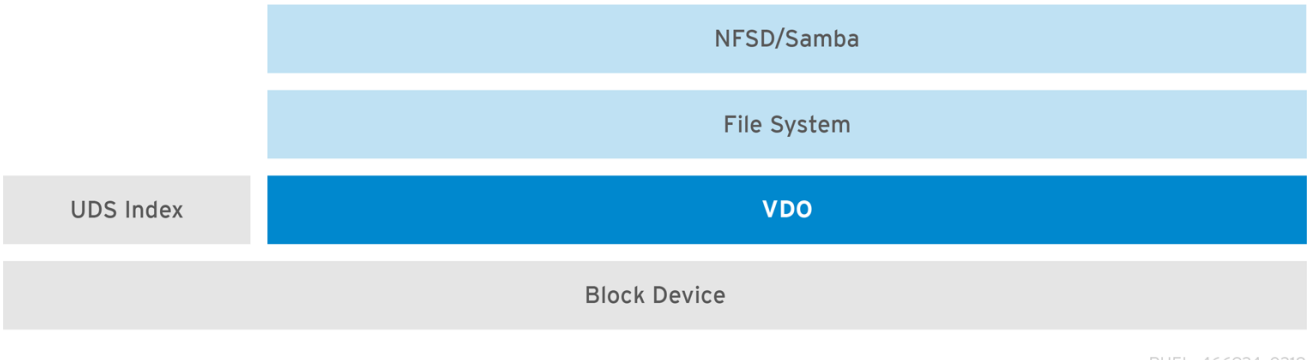


iSCSI ターゲットの詳細は http://linux-iscsi.org/ を参照してください。

30.5.2. ファイルシステム

ファイルアクセスが代わりとして必要な場合は、VDO 上にファイルシステムを作成し、Linux NFS サーバーまたは Samba を介して、NFS または CIFS のユーザーに公開できます。

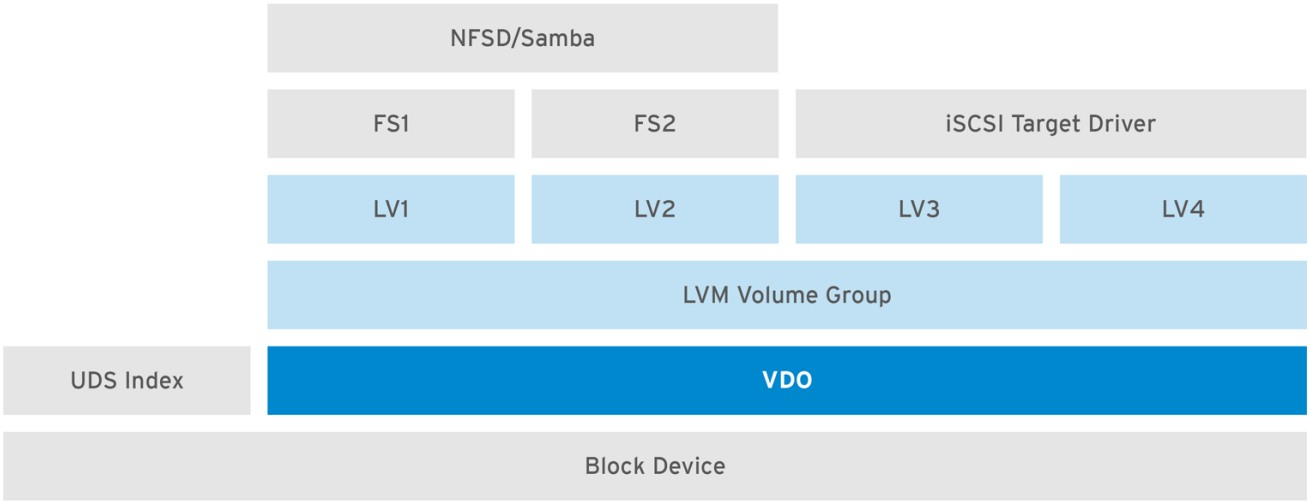
図30.4 重複排除した NAS



30.5.3. LVM

より機能豊富なシステムは、LVM をさらに利用して、同じ重複排除されたストレージプールによってすべてバックアップされる複数の LUN を提供する場合があります。図30.5「重複排除された統合ストレージ」 では、VDO ターゲットは物理ボリュームとして登録されるため、LVM が管理できます。複数の論理ボリューム(LV1 から LV4)が、重複排除されたストレージプールから作成されます。この方法で、VDO は、基となる重複排除ストレージプールへのマルチプロトコルの統合ブロック/ファイルアクセスに対応できます。

図30.5 重複排除された統合ストレージ

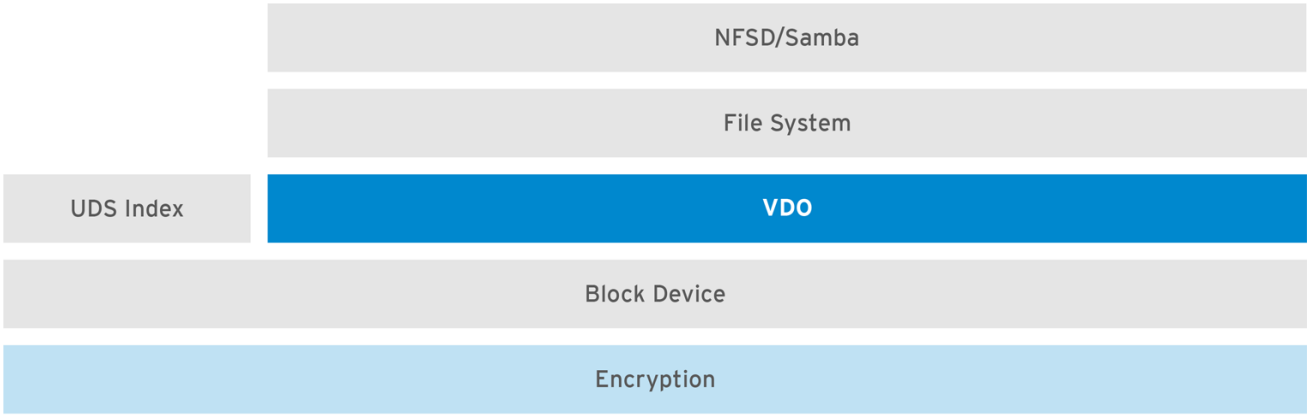


重複排除した統合ストレージ設計により、複数のファイルシステムが、LVM ツールを介して同じ重複排除ドメインを共同で使用できます。また、ファイルシステムは、LVM スナップショット、コピーオンライト、縮小機能、拡大機能、および VDO にある全機能を利用できます。

30.5.4. 暗号化

現在、データのセキュリティーは極めて重要です。ますます多くの企業がデータ暗号化に関する内部ポリシーを持っています。DM-Crypt などの Linux Device Mapper メカニズムは VDO と互換性があります。VDO ボリュームを暗号化するとデータのセキュリティーが確保され、VDO を超えるファイルシステムではディスクの最適化のために重複排除機能を引き続き利用できます。上記の VDO で暗号化を適用しても、データの重複排除はほとんど発生しないことに注意してください。暗号化は、VDO が重複排除する前に、重複するブロックを異なるものにします。

図30.6 暗号化での VDO の使用



30.6. VDO のチューニング

30.6.1. VDO チューニングの概要

データベースやその他の複雑なソフトウェアのチューニングと同様に、VDO のチューニングには、多数のシステム制約間のトレードオフが含まれ、ある程度の実験が必要です。VDO の調整に使用できる主な制御は、さまざまな種類の作業に割り当てられたスレッドの数、それらのスレッドの CPU アフィニティー設定、およびキャッシュ設定です。

30.6.2. VDO アーキテクチャーの背景

VDO カーネルドライバーはマルチスレッド化されており、複数の同時 I/O 要求間で処理コストを償却することでパフォーマンスを向上させます。1 つのスレッドで I/O 要求を最初から最後まで処理するのではなく、作業のさまざまな段階を 1 つ以上のスレッドまたはスレッドのグループに委任し、I/O 要求がパイプラインを通過するときにメッセージがそれらの間で渡されます。このようにして、1 つのスレッドで、I/O 操作が処理されるたびにグローバルデータ構造をロックおよびアンロックすることなく、グローバルデータ構造へのすべてのアクセスをシリアル化できます。VDO ドライバーのチューニングが適切に行われていないと、スレッドが要求された処理段階を完了するたびに、通常は別の要求が同じ処理のためにキューに追加されます。これらのスレッドをビジー状態に保つと、コンテキストの切り替えとスケジューリングのオーバーヘッドが削減され、パフォーマンスが向上します。基盤となるストレージシステムに I/O 操作をキューに入れることや UDS へのメッセージなど、ブロックできるオペレーティングシステムの部分にも個別のスレッドが使用されます。

VDO で使用されるさまざまなワーカースレッドタイプは次のとおりです。

論理ゾーンスレッド

文字列 kvdo:logQ を含むプロセス名を持つ 論理 スレッドは、VDO デバイスのユーザーに提示される論理ブロック番号(LBN)と、基盤となるストレージシステムの物理ブロック番号(PBN)との間のマッピングを維持します。また、同じブロックに書き込もうとする 2 つの I/O 操作が同時に処理されないように、ロックを実装します。論理ゾーンスレッドは、読み取り操作と書き込み操作の両方でアクティブになります。

LBN はチャンクに分割され (ブロックマップページ は 3MB 以上の LBN を含む)、このチャンクはスレッド間で分割されるゾーンにグループ化されます。

処理はスレッド間でかなり均等に分散させる必要がありますが、一部の不運なアクセスパターンでは、作業が 1 つのスレッドまたは別のスレッドに集中する場合があります。たとえば、特定のブロックマップページ内の LBN に頻繁にアクセスすると、論理スレッドの 1 つがそれらのすべての操作を処理します。

論理ゾーンスレッドの数は、vdo コマンドの
--vdoLogicalThreads=thread count
オプションを使用して制御できます。

物理ゾーンスレッド

物理、または kvdo:physQ のスレッドは、データブロックの割り当てを管理し、参照カウントを維持します。これらは書き込み操作時にアクティブになります。

LBN と同様、PBN は スラブ と呼ばれるチャンクに分割され、さらにゾーンに分割され、処理負荷を分散するワーカースレッドに割り当てられます。

物理ゾーンスレッドの数は、vdo コマンドの
--vdoPhysicalThreads=thread count
オプションを使用して制御できます。

I/O 送信スレッド

kvdo:bioQ スレッドは、VDO からストレージシステムにブロック I/O (bio)操作を送信します。これらは、他の VDO スレッドによってキューに入れられた I/O 要求を受け取り、それらを基盤となるデバイスドライバーに渡します。このようなスレッドは、デバイスに関連付けられたデータ構造と通信し、データ構造を更新したり、デバイスドライバーのカーネルスレッドを処理するように要求を設定したりできます。基礎となるデバイスのリクエストキューが満杯になると、I/O リクエストの送信がブロックされる可能性があるため、この作業は専用のスレッドで行われ、処理の遅延を回避します。

ps ユーティリティーまたは トップ ユーティリティーによってこれらのスレッドが頻繁に D 状態に表示される場合、VDO は頻繁にストレージシステムを I/O 要求でビジー状態に保ちます。これは、一部の SSD のように、ストレージシステムが複数のリクエストを並行して処理できる場合、またはリクエスト処理がパイプライン化されている場合に一般的に適しています。この期間にスレッドの CPU 使用率が非常に低い場合は、I/O 送信スレッドの数を減らすことができます。

CPU 使用率およびメモリー競合は、VDO の下にあるデバイスドライバーにより異なります。スレッドが追加されるにつれて I/O 要求あたりの CPU 使用率が増加する場合は、それらのデバイスドライバーの CPU、メモリー、またはロックの競合を確認してください。

I/O 送信スレッドの数は、vdo コマンドの
--vdoBioThreads=thread count
オプションを使用して制御できます。

CPU 処理スレッド

kvdo:cpuQ スレッドは、ハッシュ値の計算や、他のスレッドタイプに関連付けられたデータ構造への排他的アクセスをブロックしないデータブロックの圧縮など、CPU 集約型作業を実行するために存在します。

CPU 処理スレッドの数は、vdo コマンドの
--vdoCpuThreads=thread count
オプションを使用して制御できます。

I/O 確認スレッド

kvdo:ackQ スレッドは、VDO（カーネルページキャッシュ、ダイレクト I/O を実行するアプリケーションプログラムスレッドなど）にコールバックを発行して、I/O 要求の完了を報告します。CPU 時間の要件およびメモリーの競合は、この他のカーネルレベルのコードに依存します。

確認応答スレッドの数は、vdo コマンドの
--vdoAckThreads=thread count
オプションを使用して制御できます。

スケーラビリティーのない VDO カーネルスレッド:

重複排除スレッド

kvdo:dedupeQ スレッドは、キューに入れられた I/O 要求を受け取り、UDS に接続します。サーバーが要求を十分に迅速に処理できない場合、またはカーネルメモリーが他のシステムアクティビティーによって制約されている場合、ソケットバッファーがいっぱいになる可能性があることから、この作業は別のスレッドによって実行され、スレッドがブロックされた場合でも、他の VDO 処理を続行できます。また、長い遅延 (数秒) 後に I/O 要求をスキップするタイムアウトメカニズムもあります。

ジャーナルスレッド

kvdo:journalQ スレッドはリカバリージャーナルを更新し、書き込み用にジャーナルブロックをスケジュールします。VDO デバイスは 1 つのジャーナルのみを使用するため、この作業はスレッド間では分割できません。

パッカースレッド

圧縮が有効な場合に書き込みパスでアクティブな kvdo:packerQ スレッドは、無駄なスペースを最小限に抑えるために kvdo:cpuQ スレッドによって圧縮されたデータブロックを収集します。VDO デバイスごとに 1 つのパッカーデータ構造があり、これにより 1 つのパッカースレッドがあります。

30.6.3. 調整する値

30.6.3.1. CPU/メモリー

30.6.3.1.1. 論理、物理、cpu、ack スレッド数

論理、物理、CPU、および I/O の確認作業は、複数のスレッドに分散できます。その数は、初期の設定中、または VDO デバイスが再起動された場合は後で指定できます。

1 つのコア、つまり 1 つのスレッドは、指定された時間内に限られた量の作業を実行できます。たとえば、1 つのスレッドですべてのデータブロックのハッシュ値を計算すると、1 秒あたりに処理できるデータブロックの数にハード制限が課されます。複数のスレッド (およびコア) にわたって作業を分割すると、ボトルネックが緩和されます。

スレッドまたはコアの使用率が 100% に近づくにつれて、より多くのワークアイテムが処理のキューに追加される傾向があります。これにより、CPU のアイドルサイクルが減る可能性がありますが、個々の I/O 要求のキューイング遅延および待ち時間は通常増加します。一部のキュー理論モデルでは、使用率が 70% または 80% を超えると過剰な遅延が発生し、通常の処理時間よりも数倍長くなる可能性があります。したがって、それらのスレッドまたはコアが常にビジーであるとは限らない場合でも、使用率が 50% 以上のスレッドまたはコアの作業をさらに分散すると役立つ場合があります。

逆の場合、スレッドまたは CPU の負荷が非常に軽い (したがって、非常に頻繁にスリープ状態になっている) 場合、そのための作業を提供すると、追加のコストが発生する可能性が高くなります。(別のスレッドをウェイクアップしようとするスレッドは、スケジューラーのデータ構造に対するグローバルロックを取得する必要があり、プロセッサー間割り込みを送信して別のコアに作業を転送する可能性があります)。VDO スレッドを実行するように設定されているコアが増えると、作業がスレッド間で移動したり、スレッドがコア間で移動したりするときに、特定のデータがキャッシュされる可能性が低くなります。そのため、作業の分散が多すぎるとパフォーマンスが低下する可能性があります。

I/O 要求ごとに、論理スレッド、物理スレッド、および CPU スレッドが実行する作業はワークロードのタイプにより異なるため、システムでは、サービスを受ける予定のさまざまなタイプのワークロードでテストする必要があります。

重複排除に成功した同期モードでの書き込み操作では、新しいデータの書き込みと比較して、余分な I/O 操作 (前に保存されたデータブロックの読み込み)、いくつかの CPU サイクル (新しいデータブロックと比較して一致するか確認)、ジャーナル更新 (LBN を前に保存されたデータブロックの PBN に再マップ) が発生します。非同期モードで重複が検出された場合、上記の読み取りおよび比較操作を犠牲にして、データ書き込み操作が回避されます。重複が検出されたかどうかに関係なく、書き込みごとに発生できるジャーナル更新は 1 つだけです。

圧縮が有効になっていると、圧縮可能なデータの読み取りと書き込みに、CPU スレッドによる処理がさらに必要になります。

すべてゼロバイトを含むブロック (ゼロブロック) は、一般的に発生するため、特別に扱われます。ブロックマップでは、このようなデータを表すのに特殊なエントリーが使用され、ゼロのブロックはストレージデバイスに書き込まれたり、読み取られたりしません。したがって、すべてゼロのブロックを書き込みまたは読み取るテストでは、誤解を招く結果が生じる可能性があります。物理スレッドによる参照カウントの更新はゼロブロックまたは初期化されていないブロックには必要ないため、ゼロブロックまたは初期化されていないブロック (VDO デバイスの作成以降に書き込まれなかったブロック) を上書きするテストについても、程度は低くなりますが、同じことが当てはまります。

I/O 操作の承認は、I/O 操作ごとにコールバックが 1 つ発行されるため、実行中の作業のタイプや操作されるデータに大きく影響されない唯一のタスクです。

30.6.3.1.2. CPU アフィニティーおよび NUMA

NUMA ノードの境界を越えたメモリーのアクセスは、ローカルノードのメモリーのアクセスよりも時間がかかります。Intel プロセッサーがノードのコア間で最終レベルのキャッシュを共有すると、ノード間のキャッシュ競合は、ノード内のキャッシュ競合よりもはるかに大きな問題になります。

top などのツールは、作業を行う CPU サイクルと、停止したサイクルを区別できません。このようなツールは、キャッシュの競合と、低速なメモリーアクセスを実際の動作として解釈します。その結果、ノード間でスレッドを移動すると、スレッドの見かけの CPU 使用率が低下し、1 秒あたりに実行される操作の数が増えるように見える場合があります。

VDO のカーネルスレッドの多くは、1 つのスレッドのみがアクセスするデータ構造を維持しますが、I/O 要求自体に関するメッセージを頻繁に交換します。VDO スレッドが複数のノードで実行されている場合、またはスケジューラーがあるノードから別のノードにスレッドを再割り当てしている場合は、競合が高くなることがあります。VDO スレッドと同じノードで、他の VDO 関連の作業 (VDO への I/O 送信、ストレージデバイスの割り込み処理など) を実行できる場合は、競合をさらに減らすことができます。1 つのノードにすべての VDO 関連の作業を実行するための十分なサイクルがない場合は、他のノードに移動するスレッドを選択するときにメモリーの競合を考慮する必要があります。

実用的な場合は、taskset ユーティリティーを使用して 1 つのノードで VDO スレッドを収集します。他の VDO 関連の作業も同じノードで実行できる場合は、競合がさらに減少する可能性があります。この場合、処理に対する要求に対応するために、あるノードで CPU パワーが足りないと、別のノードに移動するスレッドを選択する際に、メモリーの競合を考慮する必要があります。たとえば、ストレージデバイスのドライバーが保持するデータ構造が多数ある場合は、デバイスの割り込み処理と VDO の I/O 送信 (デバイスのドライバーコードを呼び出す bio スレッド) の両方を別のノードに移動させると便利です。I/O 確認スレッド(ack スレッド)と上位レベルの I/O 送信スレッド（ダイレクト I/O を実行するユーザーモードスレッド、またはカーネルのページキャッシュ フラッシュ スレッド）をペアにしておくこともお勧めします。

30.6.3.1.3. 周波数のスロットル調整

消費電力が問題でない場合は、/sys/devices/system/cpu/cpu*/cpufreq/scaling_governor ファイルに文字列の performance を書き込むと、より良い結果が生じる可能性があります。これらの sysfs

ノードが存在しない場合は、Linux またはシステムの BIOS により、CPU 周波数管理を設定するための他のオプションが提供される場合があります。

特定の作業を実行するために必要な時間は、タスクの切り替えやキャッシュの競合がなくても、CPU が実行している他の作業によって異なる可能性があるため、パフォーマンスの測定は、ワークロードに基づいて頻度を動的に変化させる CPU によってさらに複雑になります。

30.6.3.2. キャッシュ

30.6.3.2.1. ブロックマップキャッシュ

VDO は、効率化のために多数のブロックマップページをキャッシュします。キャッシュサイズのデフォルトは 128 MB ですが、vdo コマンドの
--blockMapCacheSize=megabytes
オプションを使用して増やすことができます。より大きなキャッシュを使用すると、ランダムアクセスワークロードに大きなメリットがもたらされる可能性があります。

30.6.3.2.2. 読み取りキャッシュ

2 番目のキャッシュは、VDO の重複排除のアドバイスを検証するために、ストレージシステムから読み取られたデータブロックをキャッシュするために使用できます。短期間で同様のデータブロックが検出されると、必要な I/O 操作の数が減る可能性があります。

読み取りキャッシュには、圧縮されたユーザーデータを含むストレージブロックも含まれます。短期間に複数の圧縮可能ブロックが書き込まれた場合、その圧縮バージョンは、同じストレージシステムブロック内に存在する可能性があります。同様に、短時間で読み取りを行うと、キャッシュにより、ストレージシステムからの読み取りを追加で行う必要がなくなります。

vdo コマンドの
--readCache={enabled | disabled}
オプションは、読み取りキャッシュを使用するかどうかを制御します。これを有効にした場合、キャッシュの最小サイズは 8 MB ですが、
--readCacheSize=megabytes
オプションで増やすことができます。読み取りキャッシュの管理には若干のオーバーヘッドが発生するため、ストレージシステムが十分に高速であれば、パフォーマンスが向上しない可能性があります。読み取りキャッシュはデフォルトで無効になっています。

30.6.3.3. ストレージシステム I/O

30.6.3.3.1. Bio スレッド

RAID 設定の汎用ハードドライブでは、I/O 操作を送信するには 1 つまたは 2 つの bio スレッドで十分な場合があります。ストレージデバイスドライバーが、I/O 送信スレッドに著しく多くの作業 (ドラ

イバーのデータ構造の更新やデバイスとの通信) を要求し、1 つか 2 つのスレッドが非常に忙しく、ストレージデバイスがしばしばアイドルである場合、bio スレッド数を増やして補うことができます。ただし、ドライバーの実装によっては、スレッド数を増やしすぎると、キャッシュまたはスピンロックの競合が発生する可能性があります。デバイスのアクセスタイミングが NUMA ノード全体で均一でない場合は、ストレージデバイスコントローラーに最も近いノードで bio スレッドを実行すると便利な場合があります。

30.6.3.3.2. IRQ の処理

デバイスドライバーが割り込みハンドラーで重要な作業を行い、スレッド化された IRQ ハンドラーを使用しない場合、スケジューラーが最高のパフォーマンスを提供できなくなる可能性があります。ハードウェアの割り込みの処理に費やされる CPU 時間は、通常の VDO (または他の) カーネルスレッドの実行と似ている場合があります。たとえば、ハードウェア IRQ 処理にコアのサイクルの 30% が必要な場合、同じコア上のビジーなカーネルスレッドは残りの 70% しか使用できませんでした。ただし、そのスレッド用にキューに入れられた作業が、コアのサイクルの 80% を要求していた場合、スレッドが追いつくことはなく、スケジューラーは、スレッドをビジーでないコアに切り替える代わりに、そのスレッドをそのコアで実行するのを妨げたままにする可能性があります。

VDO ワークロードでそのようなデバイスドライバーを使用する場合は、ハードウェア割り込みに多くのサイクルが必要になる場合があります(トップ ディスプレイのヘッダーの %hi インジケーター)。その場合は、IRQ 処理を特定のコアに割り当て、VDO カーネルスレッドの CPU アフィニティーを調整して、そのコアで実行しないようにすると役立つ場合があります。

30.6.3.4. 最大破棄セクター

VDO デバイスに許可される DISCARD (TRIM) 操作の最大サイズは、システムの使用方法に基づいて、/sys/kvdo/max_discard_sectors で調整できます。デフォルトは 8 セクター (つまり 4KB ブロック 1 つ) です。より大きなサイズを指定することもできますが、VDO はループで一度に 1 ブロックずつ処理し、破棄された 1 つのブロックのメタデータの更新がジャーナルに書き込まれ、次のブロックで開始する前にディスクにフラッシュされるようにします。

VDO ボリュームをローカルファイルシステムとして使用すると、Linux カーネルの一般的なブロックデバイスコードが大量の破棄要求を複数の小規模な破棄要求に分割し、並行に送信するため、Red Hat テストでは、小規模な破棄サイズが最適であることがわかりました。デバイスの I/O アクティビティーが少ない場合、VDO は、多数の小規模な要求を同時に処理し、1 つの大規模な要求よりもはるかに迅速に処理できます。

VDO デバイスを SCSI ターゲットとして使用する場合は、イニシエーターとターゲットソフトウェアは考慮すべき追加要素を導入します。ターゲットの SCSI ソフトウェアが SCST の場合は、最大破棄サイズを読み込み、イニシエーターに中継します。(Red Hat では、LIO SCSI ターゲットコードと併用して VDO 設定のチューニングを行っていません。)

Linux SCSI イニシエーターコードでは一度に 1 つの廃棄操作しか許可されないため、最大サイズを超える破棄要求は、複数の小さな破棄に分割され、一度に 1 つずつターゲットシステム (および

VDO) に送信されます。したがって、VDO が多数の小さな破棄操作をシリアルで処理することに加えて、2 つのシステム間のラウンドトリップ通信時間により、さらなる遅延が追加されます。

最大廃棄サイズを大きく設定することで、この通信オーバーヘッドを削減することができますが、この大きなリクエストは全体が VDO に渡され、一度に 4KB のブロックが処理されます。ブロックごとの通信遅延はありませんが、大きなブロックに処理時間を追加すると、SCSI イニシエーターソフトウェアがタイムアウトになる場合があります。

SCSI ターゲットの使用方法として、Red Hat では、イニシエーターのタイムアウト設定内で通常の廃棄時間を適切に保持しながら、最大廃棄サイズを適度に大きく設定することを推奨しています。たとえば、数秒ごとの追加のラウンドトリップコストはパフォーマンスに大きな影響を与えないはずであり、タイムアウトが 30 秒または 60 秒の SCSI イニシエータはタイムアウトにならないはずです。

30.6.4. ボトルネックの特定

VDO のパフォーマンスに影響を与えるいくつかの重要な要素があり、最も影響力のあるものを特定するために利用できる多くのツールがあります。

top や ps などのユーティリティーで見られるように、スレッドまたは CPU の使用率が 70% を超える場合は、通常、1 つのスレッドまたは 1 つの CPU に多くの作業が集中していることを意味します。ただし、場合によっては、VDO スレッドが CPU で実行されるようにスケジュールされていても、実際には作業が行われなかったことを意味する場合があります。このシナリオは、過度のハードウェア割り込みハンドラー処理、コアまたは NUMA ノード間のメモリー競合、またはスピンロックの競合で発生する可能性があります。

top ユーティリティーを使用してシステムパフォーマンスを調べる場合、Red Hat は top -H を実行してすべてのプロセススレッドを個別に表示し、1 f j キーを入力し、その後に Enter/Return キーを入力することをお勧めします。top コマンドは、個々の CPU コアの負荷を表示し、各プロセスまたはスレッドが最後に実行された CPU を識別します。この情報により、以下の洞察が得られます。

- コアに %id (idle)と %wa (waiting-for-I/O)の値が少ない場合、ある種の作業でビジー状態が保たれています。
- コアの %hi 値が非常に低い場合、そのコアは通常の処理作業を行い、カーネルスケジューラーによって負荷分散されています。そのセットにコアを追加すると、NUMA の競合が発生しない限り、負荷が軽減される可能性があります。
- コアの %hi が数パーセントを超え、そのコアに割り当てられているスレッドは 1 つだけで、%id と %wa がゼロの場合、コアはオーバーコミットされ、スケジューラーは状況に対処

していません。この場合は、カーネルスレッドやデバイスの割り込み処理を別のコアに保持するために再割り当てする必要があります。

perf ユーティリティーは、多くの CPU のパフォーマンスカウンターを調べることができます。Red Hat は、perf top サブコマンドを開始点として使用して、スレッドまたはプロセッサーが実行する作業を調べることをお勧めします。たとえば、bioQ スレッドがスピンロックの取得を試行する多くのサイクルを費やしている場合、VDO 未満のデバイスドライバーで競合が多すぎる可能性があり、bioQ スレッドの数を減らすと状況が軽減される可能性があります。CPU 使用率が高い場合（スピンロックの取得など）は、たとえば bioQ スレッドとデバイス割り込みハンドラーが異なるノードで実行されている場合など、NUMA ノード間の競合を示している可能性があります。プロセッサーがそれらをサポートしている場合、stalled-cycles-backend、cache-misses、および node-load-misses などのカウンターが重要になる可能性があります。

sar ユーティリティーは、複数のシステム統計に関する定期的なレポートを提供できます。sar -d 1 コマンドは、ブロックデバイス使用率レベル（少なくとも 1 つの I/O 操作が進行中である割合）とキューの長さ(I/O 要求の待機数)を 1 秒に 1 回報告します。ただし、すべてのブロックデバイスドライバーがそのような情報を報告できるわけではないため、sar の有用性は、使用中のデバイスドライバーに依存する可能性があります。

30.7. VDO コマンド

このセクションでは、次の VDO ユーティリティーを説明します。

vdo

vdo ユーティリティーは、VDO の kvdo コンポーネントと UDS コンポーネントの両方を管理します。

また、圧縮の有効/無効を切り替えるためにも使用されます。

vdostats

vdostats ユーティリティーは、Linux df ユーティリティーと同様の形式で、設定（または指定

した）デバイスごとの統計情報を表示します。

30.7.1. vdo

vdo ユーティリティーは、VDO の kvdo コンポーネントと UDS コンポーネントの両方を管理します。

概要

```
vdo { activate | changeWritePolicy | create | deactivate | disableCompression | disableDeduplication |
enableCompression | enableDeduplication | growLogical | growPhysical | list | modify | printConfigFile
| remove | start | status | stop }
[ options... ]
```

サブコマンド

表30.4 VDO サブコマンド

サブコマンド	説明

サブコマンド	説明
作成	VDO ボリュームと、その関連インデックスを作成し、使用できるようにします。**--activate=disabled** を指定すると、VDO ボリュームが作成されますが、使用できません。**--force** が指定されていない場合は、既存のファイルシステムまたはフォーマットされた VDO ボリュームを上書きしません。このコマンドは、root 権限で実行する必要があります。適用可能なオプションは以下のとおりです。 • **--name=*volume*** (必須) • **--device=*device*** (必須) • **--activate={enabled | disabled}** • **--indexMem=*gigabytes*** • **--blockMapCacheSize=*megabytes*** • **--blockMapPeriod=*period*** • **--compression={enabled | disabled}** • **--confFile=*file*** • **--deduplication={enabled | disabled}** • **--emulate512={enabled | disabled}** • **--sparseIndex={enabled | disabled}** • **--vdoAckThreads=*thread count*** • **--vdoBioRotationInterval=*I/O count*** • **--vdoBioThreads=*thread count*** • **--vdoCpuThreads=*thread count*** • **--vdoHashZoneThreads=*thread count*** • **--vdoLogicalThreads=*thread count*** • **--vdoLogLevel=*level*** • **--vdoLogicalSize=*megabytes*** • **--vdoPhysicalThreads=*thread count*** • **--readCache={enabled | disabled}** • **--readCacheSize=*megabytes*** • **--vdoSlabSize=*megabytes*** • **--verbose** • **--writePolicy={ auto | sync | async }** • **--logfile=pathname**

サブコマンド	説明
remove	停止している 1 つまたは複数の VDO ボリュームと関連するインデックスを削除します。このコマンドは、root 権限で実行する必要があります。適用可能なオプションは以下のとおりです。 ● { **--name=*volume*** | **--all** } (必須) ● **--confFile=*file*** ● **--force** ● **--verbose** ● **--logfile=pathname**
start	停止、アクティブ化されている 1 つ以上の VDO ボリュームおよび関連サービスを開始します。このコマンドは、root 権限で実行する必要があります。適用可能なオプションは以下のとおりです。 ● { **--name=*volume*** | **--all** } (必須) ● **--confFile=*file*** ● **--forceRebuild** ● **--verbose** ● **--logfile=pathname**
stop	1 つ以上の VDO ボリュームおよび関連サービスの実行を停止します。このコマンドは、root 権限で実行する必要があります。適用可能なオプションは以下のとおりです。 ● { **--name=*volume*** | **--all** } (必須) ● **--confFile=*file*** ● **--force** ● **--verbose** ● **--logfile=pathname**
activate	1 つ以上の VDO ボリュームをアクティブにします。アクティブなボリュームは、 **start** コマンドを使用して起動できます。このコマンドは、root 権限で実行する必要があります。適用可能なオプションは以下のとおりです。 ● { **--name=*volume*** | **--all** } (必須) ● **--confFile=*file*** ● **--logfile=pathname** ● **--verbose**

サブコマンド	説明
deactivate	1 つ以上の VDO ボリュームを非アクティブにします。アクティブでないボリュームは、 **start** コマンドでは起動できません。現在実行中のボリュームを非アクティブにしても、ボリュームは停止しません。停止したら、再起動する前に、無効にした VDO ボリュームを有効にする必要があります。このコマンドは、root 権限で実行する必要があります。適用可能なオプションは以下のとおりです。 ● { **--name=*volume*** | **--all** } (必須) ● **--confFile=*file*** ● **--verbose** ● **--logfile=pathname**
status	VDO システムとボリュームのステータスを YAML 形式で報告します。このコマンドは root 権限を必要としませんが、root 権限なしで実行すると、情報は不完全になります。適用可能なオプションは以下のとおりです。 ● { **--name=*volume*** | **--all** } (必須) ● **--confFile=*file*** ● **--verbose** ● **--logfile=pathname** 提供される出力は、表30.6「VDO ステータス出力」 を参照してください。
リスト	起動している VDO ボリュームのリストを表示します。**--all** を指定すると、起動したボリュームと、起動していないボリュームの両方が表示されます。このコマンドは、root 権限で実行する必要があります。適用可能なオプションは以下のとおりです。 ● **--all** ● **--confFile=*file*** ● **--logfile=pathname** ● **--verbose**

サブコマンド	説明
modify	1つまたはすべての VDO ボリュームの設定パラメーターを変更します。変更は、次に VDO デバイスを起動したときに有効になります。すでに実行中のデバイスには影響しません。適用可能なオプションは以下のとおりです。 ● { **--name=*volume*** | **--all** } (必須) ● **--blockMapCacheSize=*megabytes*** ● **--blockMapPeriod=*period*** ● **--confFile=*file*** ● **--vdoAckThreads=*thread count*** ● **--vdoBioThreads=*thread count*** ● **--vdoCpuThreads=*thread count*** ● **--vdoHashZoneThreads=*thread count*** ● **--vdoLogicalThreads=*thread count*** ● **--vdoPhysicalThreads=*thread count*** ● **--readCache={enabled | disabled}** ● **--readCacheSize=*megabytes*** ● **--verbose** ● **--logfile=pathname**
changeWritePolicy	実行中の1つまたはすべての VDO ボリュームの書き込みポリシーを変更します。このコマンドは、root 権限で実行する必要があります。 ● { **--name=*volume*** | **--all** } (必須) ● **--writePolicy={ auto | sync | async }** (必須) ● **--confFile=*file*** ● **--logfile=pathname** ● **--verbose**

サブコマンド	説明
enableDeduplication	1 つ以上の VDO ボリュームでの重複排除を有効にします。このコマンドは、root 権限で実行する必要があります。適用可能なオプションは以下のとおりです。 ● { **--name=*volume*** | **--all** } (必須) ● **--confFile=*file*** ● **--verbose** ● **--logfile=pathname**
disableDeduplication	1 つ以上の VDO ボリュームでの重複排除を無効にします。このコマンドは、root 権限で実行する必要があります。適用可能なオプションは以下のとおりです。 ● { **--name=*volume*** | **--all** } (必須) ● **--confFile=*file*** ● **--verbose** ● **--logfile=pathname**
enableCompression	1 つ以上の VDO ボリュームで圧縮を有効にします。VDO ボリュームが実行中の場合は、すぐに有効になります。VDO ボリュームで圧縮を実行していない場合は、次に VDO ボリュームを起動したときに圧縮が有効になります。このコマンドは、root 権限で実行する必要があります。適用可能なオプションは以下のとおりです。 ● { **--name=*volume*** | **--all** } (必須) ● **--confFile=*file*** ● **--verbose** ● **--logfile=pathname**
disableCompression	1 つ以上の VDO ボリュームでの圧縮を無効にします。VDO ボリュームが実行中の場合は、すぐに有効になります。VDO ボリュームで圧縮を実行していない場合は、次に VDO ボリュームを起動したときに圧縮が無効になります。このコマンドは、root 権限で実行する必要があります。適用可能なオプションは以下のとおりです。 ● { **--name=*volume*** | **--all** } (必須) ● **--confFile=*file*** ● **--verbose** ● **--logfile=pathname**

サブコマンド	説明
growLogical	VDO ボリュームに論理領域を追加します。ボリュームが存在し、実行されている必要があります。このコマンドは、root 権限で実行する必要があります。適用可能なオプションは以下のとおりです。 ● **--name=*volume*** (必須) ● **--vdoLogicalSize=*megabytes*** (必須) ● **--confFile=*file*** ● **--verbose** ● **--logfile=pathname**
growPhysical	VDO ボリュームに物理領域を追加します。ボリュームが存在し、実行されている必要があります。このコマンドは、root 権限で実行する必要があります。適用可能なオプションは以下のとおりです。 ● **--name=*volume*** (必須) ● **--confFile=*file*** ● **--verbose** ● **--logfile=pathname**
printConfigFile	設定ファイルを **stdout** に出力します。このコマンドには、root 権限が必要です。適用可能なオプションは以下のとおりです。 ● **--confFile=*file*** ● **--logfile=pathname** ● **--verbose**

オプション

表30.5 VDO オプション

オプション	説明
--indexMem=*gigabytes*	UDS サーバーメモリーの量をギガバイト単位で指定します。デフォルトサイズは 1GB です。特殊な 10 進数の値である 0.25、0.5、0.75 は、正の整数として使用できます。
--sparseIndex={enabled | disabled}	スパースインデックス作成を有効または無効にします。デフォルトでは が **無効に** なります。

オプション	説明
--all	このコマンドを、設定した VDO ボリュームすべてに適用する必要があることを示しています。**--name** と併用しないでください。
--blockMapCacheSize=*megabytes*	ブロックマップページのキャッシュに割り当てるメモリーの量を指定します。値は、4096 の倍数にする必要があります。**B**(ytes)、**K**(ilobytes)、**M**(egabytes)、**G**(igabytes)、**T**（テラバイト）、**P**（トリアバイト）、または **E**(xabytes)の接尾辞が付いた値の使用はオプションです。接尾辞が指定されていない場合、値はメガバイトとして解釈されます。デフォルトは 128M です。値は 128M 以上、16T 未満にする必要があります。メモリーオーバーヘッドは 15% であることに注意してください。
--blockMapPeriod=*period*	キャッシュされたページがディスクにフラッシュされる前に蓄積される可能性のあるブロックマップ更新の数を決定する 1 から 16380 までの値。値を大きくすると、通常の操作中のパフォーマンスが低下しますが、クラッシュ後の回復時間が短くなります。デフォルト値は 16380 です。このパラメーターを調整する前に、RedHat の担当者に相談してください。
--compression={enabled | disabled}	VDO デバイス内での圧縮を有効または無効にします。デフォルトでは有効になっています。パフォーマンスを最大化するため、または圧縮される可能性が低いデータの処理を高速化するために、必要に応じて圧縮を無効にすることができます。
--confFile=*file*	別の設定ファイルを指定します。デフォルトは **/etc/vdoconf.yml** です。
--deduplication={enabled | disabled}	VDO デバイス内での重複排除を有効または無効にします。デフォルトでは が **有効** です。データの重複排除率が高くないと予想されるが、圧縮が必要な場合は、重複排除を無効にすることができます。
--emulate512={enabled | disabled}	512 バイトのブロックデバイスエミュレーションモードを有効にします。デフォルトでは が **無効に** なります。
--force	VDO ボリュームを停止する前に、マウントされたファイルシステムをアンマウントします。
--forceRebuild	読み取り専用の VDO ボリュームを開始する前に、オフラインでの再構築を強制して、オンラインに戻して使用できるようにします。**このオプションを使用すると、データの損失や破損が生じる可能性があります。**
--help	**vdo** ユーティリティーのドキュメントを表示します。
--logfile=pathname	このスクリプトのログメッセージの出力先となるファイルを指定します。警告メッセージおよびエラーメッセージは、常に syslog にも記録されます。
--name=*volume*	指定した VDO ボリュームで動作します。**--all** と併用しないでください。

オプション	説明
--device=*device*	VDO ストレージに使用するデバイスの絶対パスを指定します。
--activate={enabled | disabled}	引数 **disabled** は、VDO ボリュームのみを作成することを示します。ボリュームは起動されたり、有効化されたりしません。デフォルトでは が **有効** です。
--vdoAckThreads=*thread count*	要求された VDOI/O 操作の完了を確認するために使用するスレッドの数を指定します。デフォルトは 1 です。値は 0 以上、100 以下である必要があります。
--vdoBioRotationInterval=*I/O count*	作業を次のスレッドに転送する前に、bio 送信スレッドごとにキューに入れる I/O 操作の数を指定します。デフォルトは 64 です。値は 1 以上、1024 以下である必要があります。
--vdoBioThreads=*thread count*	ストレージデバイスへの I/O 操作の送信に使用するスレッド数を指定します。最小は 1 で、最大は 100 です。デフォルトは 4 です。値は 1 以上、100 以下である必要があります。
--vdoCpuThreads=*thread count*	ハッシュや圧縮など、CPU を集中的に使用する作業に使用するスレッドの数を指定します。デフォルトは 2 です。値は 1 以上、100 以下である必要があります。
--vdoHashZoneThreads=*thread count*	ブロックデータから計算したハッシュ値に基づいて、VDO 処理の一部を細分化するスレッド数を指定します。デフォルトは **1** です。値は 0 以上、100 以下である必要があります。**vdoHashZoneThreads**、**vdoLogicalThreads**、および **vdoPhysicalThreads** は、すべてゼロであるか、すべてゼロ以外の値にする必要があります。
--vdoLogicalThreads=*thread count*	ブロックデータから計算したハッシュ値に基づいて、VDO 処理の一部を細分化するスレッド数を指定します。この値は、0 以上、100 以下である必要があります。論理スレッド数が 9 以上の場合は、ブロックマップキャッシュのサイズも明示的に指定する必要があります。**vdoHashZoneThreads**、**vdoLogicalThreads**、および **vdoPhysicalThreads** は、すべてゼロにするか、すべてゼロ以外にする必要があります。デフォルトでは 1 回です。
--vdoLogLevel=*level*	VDO ドライバーログレベル (**critical**、**error**、**warning**、**notice**、**info**、または **debug**) を指定します。レベルでは大文字と小文字が区別されます。デフォルトは **info** です。
--vdoLogicalSize=*megabytes*	論理 VDO ボリュームのサイズをメガバイト単位で指定します。**S**(ectors)、**B**(ytes)、**K**(ilobytes)、**M**(egabytes)、**G**(igabytes)、**T**（テラバイト）、**P**(etabytes)、または **E**(xabytes)の接尾辞が付いた値の使用はオプションです。ボリュームのオーバープロビジョニングに使用されます。これはデフォルトでストレージデバイスのサイズになります。

オプション	説明
--vdoPhysicalThreads=*thread count*	物理ブロックアドレスに基づいて、VDO 処理の一部を細分化するスレッド数を指定します。この値は、0 以上、16 以下である必要があります。最初のスレッド以降にスレッドを追加するたびに、さらに 10 MB の RAM を使用します。**vdoPhysicalThreads**、**vdoHashZoneThreads**、および **vdoLogicalThreads** は、すべてゼロにするか、すべてゼロ以外にする必要があります。デフォルトでは 1 回です。
--readCache={enabled | disabled}	VDO デバイス内の読み取りキャッシュを有効または無効にします。デフォルトは **disabled** です。書き込みワークロードで重複排除のレベルが高いことが予想される場合、または高圧縮性のデータの読み取り集中型ワークロードの場合は、キャッシュを有効にする必要があります。
--readCacheSize=*megabytes*	追加の VDO デバイス読み取りキャッシュサイズをメガバイト単位で指定します。この領域は、システム定義の最小領域に追加されます。**B**(ytes)、**K**(ilobytes)、**M**(egabytes)、**G**(igabytes)、**T**（テラバイト）、**P**（トリアバイト）、または **E**(xabytes)の接尾辞が付いた値の使用はオプションです。デフォルトは 0M です。bio スレッドごとに、指定された読み取りキャッシュの MB ごとに 1.12 MB のメモリーが使用されます。
--vdoSlabSize=*megabytes*	VDO が拡張される増分のサイズを指定します。サイズを小さくすると、収納可能な物理サイズの合計が制約されます。128M から 32G の間の 2 の累乗である必要があります。デフォルトは 2G です。**S**(ectors)、**B**(ytes)、**K**(ilobytes)、**M**(egabytes)、**G**(igabytes)、**T**（テラバイト）、**P**(etabytes)、または **E**(xabytes)の接尾辞が付いた値の使用はオプションです。接尾辞を使用しない場合、この値はメガバイトとして解釈されます。
--verbose	コマンドを実行する前に出力します。
--writePolicy={ auto | sync | async }	書き込みポリシーを指定します。 • **auto**: VDO の下にあるストレージ層に基づいて **sync** または **async** を選択します。ライトバックキャッシュが存在する場合は、**async** が選択されます。それ以外の場合は **sync** が選択されます。 • **同期**: 書き込みは、データが大量に書き込まれた後にのみ承認されます。これはデフォルトポリシーになります。**基本となるストレージも同期していない場合は、このポリシーはサポートされません。** • **async**: 安定したストレージに書き込むためにデータが キャッシュ された後に書き込みが確認されます。フラッシュされていないデータがこのモードで持続する保証はありません。

status

サブコマンドは、YAML 形式で、次の情報を返します。キーは以下のように分割されます。

表30.6 VDO ステータス出力

キー	説明	
VDO のステータス	このキーの情報には、ホストの名前とステータス照会が行われている日時が含まれます。この領域で報告されるパラメーターは以下のとおりです。	
	ノード	VDO が実行しているシステムのホスト名。
	日付	vdo status コマンドを実行した日時。
カーネルモジュール	このキーの情報は、設定されたカーネルをカバーします。	
	Loaded	カーネルモジュールが読み込まれているかどうか (True または False)。
	バージョン情報	設定されている kvdo のバージョンに関する情報。
設定	このキーの情報は、VDO 設定ファイルの場所とステータスをカバーします。	
	ファイル	VDO 設定ファイルの場所。
	最終更新日	VDO 設定ファイルの最終更新日。
VDO	すべての VDO ボリュームの設定情報を提供します。各 VDO ボリュームで報告されるパラメーターは以下のとおりです。	
	ブロックサイズ	VDO ボリュームのブロックサイズ (バイト単位)。
	512 バイトエミュレーション	ボリュームが 512 バイトのエミュレーションモードで実行しているかどうかを示します。
	重複排除の有効化	ボリュームで重複排除が有効かどうか。
	論理サイズ	VDO ボリュームの論理サイズ。
	物理サイズ	VDO ボリュームの基本となる物理ストレージのサイズ。
	書き込みポリシー	書き込みポリシーの設定値 (sync または async)。
	VDO 統計	vdostats ユーティリティーの出力。

30.7.2. vdostats

vdostats ユーティリティーは、Linux df ユーティリティーと同様の形式で、設定（または指定した）デバイスごとの統計情報を表示します。

root 権限で実行されていない場合、vdostats ユーティリティーの出力が不完全になる可能性があります。

概要

```
vdostats [ --verbose | --human-readable | --si | --all ] [ --version ] [ device ...]
```

オプション

表30.7 vdostats オプション

オプション	説明
--verbose	1 つ (または複数) の VDO デバイスの使用率とブロック I/O (bios) 統計を表示します。詳しくは 表30.9「vdostats --詳細な出力」 をご覧ください。
--human-readable	ブロック値を読み取り可能な形式で表示します (ベース 2: 1 KB = 2^{10} bytes = 1024 バイト)。
--si	**--si** オプションは、**--human-readable** オプションの出力を SI ユニットを使用するように変更します (ベース 10: 1KB = 10^3 バイト = 1000 バイト)。**--human-readable** オプションが指定されていない場合は、**--si** オプションが無効になります。
--all	このオプションは、後方互換性のためにのみ使用されます。これは、**--verbose** オプションと同等になりました。
--version	**vdostats** のバージョンを表示します。
device ...	報告する 1 つ以上の特定のボリュームを指定します。この引数を省略すると、**vdostats** はすべてのデバイスでレポートします。

出力

以下の例では、オプションが指定されていない場合の出力例を示しています。これは、表30.8「vdostats のデフォルトの出力」 で説明されています。

```
Device                  1K-blocks    Used       Available    Use%   Space Saving%
/dev/mapper/my_vdo   1932562432   427698104   1504864328   22%    21%
```

表30.8 vdostats のデフォルトの出力

項目	説明
Device	VDO ボリュームへのパス。
1K-blocks	VDO ボリュームに割り当てられている 1K ブロックの合計数 (= 物理ボリュームサイズ * ブロックサイズ / 1024)
Used	VDO ボリュームで使用されている 1K ブロックの合計数 (= 使用されている物理ブロック * ブロックサイズ / 1024)
Available	VDO ボリュームで利用可能な 1K ブロックの合計数 (= 空き物理ブロック * ブロックサイズ / 1024)
Use%	VDO ボリュームで使用されている物理ブロックの割合 (= 使用済みブロック / 割り当て済みブロック * 100)
Space Saving%	VDO ボリュームに保存されている物理ブロックの割合 (= [使用されている論理ブロック - 使用されている物理ブロック] / 使用されている論理ブロック)

--human-readable オプションでは、ブロックカウントを従来の単位 (1 KB = 1024 バイト) に変換します。

```
Device              Size   Used     Available     Use%   Space Saving%
/dev/mapper/my_vdo   1.8T   407.9G    1.4T          22%    21%
```

--human-readable オプションおよび --si オプションは、ブロックカウントを SI ユニット (1 KB = 1000 バイト) に変換します。

```
Device              Size   Used     Available     Use%    Space Saving%
/dev/mapper/my_vdo   2.0T   438G      1.5T          22%     21%
```

--verbose (表30.9「vdostats --詳細な出力」) オプションは、1 つ (またはすべて) の VDO デバイスの YAML 形式で VDO デバイス統計情報を表示します。

表30.9「vdostats --詳細な出力」 で太字 に出力される統計は、今後のリリースで引き続き報告される予定です。その他のフィールドは、主にソフトウェアサポートを目的としており、今後のリリースで変更される可能性があります。管理ツールで使用することはできません。管理ツールでは、統計情報の報告順序に依存しないようにしてください。

表30.9 vdostats --詳細な出力

項目	説明
Version	この統計のバージョン。
Release version	VDO のリリースバージョン。
Data blocks used	VDO ボリュームがデータを保存するために現在使用している物理ブロックの数。
Overhead blocks used	VDO メタデータを保存するために VDO ボリュームが現在使用している物理ブロックの数。
Logical blocks used	現在マッピングされている論理ブロックの数。
Physical blocks	VDO ボリュームに割り当てられている物理ブロックの合計数。
Logical blocks	VDO ボリュームでマッピングできる論理ブロックの最大数。
1K-blocks	VDO ボリュームに割り当てられている 1K ブロックの合計数 (= 物理ボリュームサイズ * ブロックサイズ / 1024)
1K-blocks used	VDO ボリュームで使用されている 1K ブロックの合計数 (= 使用されている物理ブロック * ブロックサイズ / 1024)
1K-blocks available	VDO ボリュームで利用可能な 1K ブロックの合計数 (= 空き物理ブロック * ブロックサイズ / 1024)
Used percent	VDO ボリュームで使用されている物理ブロックの割合 (= 使用済みブロック / 割り当て済みブロック * 100)
Saving percent	VDO ボリュームに保存されている物理ブロックの割合 (= [使用されている論理ブロック - 使用されている物理ブロック] / 使用されている論理ブロック)
Block map cache size	ブロックマップキャッシュのサイズ (バイト単位)。
Write policy	アクティブな書き込みポリシー (同期または非同期)。これは、**vdo changeWritePolicy --writePolicy=auto|sync|async** で設定されます。
Block size	VDO ボリュームのブロックサイズ (バイト単位)。
Completed recovery count	VDO ボリュームが不完全なシャットダウンから復旧した回数。
Read-only recovery count	VDO ボリュームが、(**vdo start --forceRebuild** を介して) 読み取り専用モードから復元された回数。
Operating mode	VDO ボリュームが通常どおり動作しているか、復旧モードであるか、読み取り専用モードであるかを示します。

項目	説明
Recovery progress (%)	オンライン復旧の進捗状況を示します。またはボリュームが復旧モードでない場合は**N/A** を示します。
Compressed fragments written	VDO ボリュームが最後に再起動してから書き込まれた圧縮フラグメントの数。
Compressed blocks written	VDO ボリュームを最後に再起動してから書き込まれた圧縮データの物理ブロックの数。
Compressed fragments in packer	まだ書き込まれていない、処理中の圧縮フラグメントの数。
Slab count	スラブの合計数。
Slabs opened	ブロックがこれまでに割り当てられたスラブの総数。
Slabs reopened	VDO が開始されてからスラブが再開された回数。
Journal disk full count	リカバリージャーナルがいっぱいであったために、要求がリカバリージャーナル項目を作成できなかった回数。
Journal commits requested count	リカバリージャーナルがスラブジャーナルをコミットするように要求した回数。
Journal entries batching	開始されたジャーナルエントリーの書き込み数から、書き込まれたジャーナルエントリーの数を引いた数。
Journal entries started	メモリー内で作成されたジャーナルエントリーの数。
Journal entries writing	送信された書き込みのジャーナルエントリーの数から、ストレージにコミットされたジャーナルエントリーの数を引いたもの。
Journal entries written	書き込みが発行されたジャーナルエントリーの合計数。
Journal entries committed	ストレージに書き込まれたジャーナルエントリーの数。
Journal blocks batching	開始したジャーナルブロックの書き込み数から、書き込まれたジャーナルブロックの数を引いた数。
Journal blocks started	メモリー内でタッチされたジャーナルブロックの数。
Journal blocks writing	書き込まれたジャーナルブロックの数 (アクティブメモリー内のメタデータを使用) からコミットされたジャーナルブロックの数を引いた数。
Journal entries written	書き込みが発行されたジャーナルブロックの合計数。

項目	説明
Journal blocks committed	ストレージに書き込まれたジャーナルブロックの数。
Slab journal disk full count	オンディスクスラブジャーナルがいっぱいになった回数。
Slab journal flush count	フラッシュしきい値を超えるスラブジャーナルにエントリーが追加された回数。
Slab journal blocked count	ブロックしきい値を超えたスラブジャーナルにエントリーが追加された回数。
Slab journal blocks written	発行されたスラブジャーナルブロック書き込みの数。
Slab journal tail busy count	書き込み要求がスラブジャーナル書き込みの待機をブロックした回数。
Slab summary blocks written	発行されたスラブサマリーブロック書き込みの数。
Reference blocks written	発行された参照ブロック書き込みの数。
Block map dirty pages	ブロックマップキャッシュ内のダーティーページ数。
Block map clean pages	ブロックマップキャッシュ内のクリーンなページ数。
ブロックマップの空きページ	ブロックマップキャッシュの空きページ数。
Block map failed pages	書き込みエラーがあるブロックマップキャッシュページの数。
Block map incoming pages	キャッシュに読み込まれているブロックマップキャッシュページの数。
Block map outgoing pages	書き込まれているブロックマップキャッシュページの数。
Block map cache pressure	必要に時に空きページが利用できなかった回数。
Block map read count	ブロックマップページ読み取りの合計数。
Block map write count	ブロックマップページへの書き込みの合計数。
Block map failed reads	ブロックマップの読み取りエラーの合計数。

項目	説明
Block map failed writes	ブロックマップの書き込みエラーの合計数。
Block map reclaimed	回収されたブロックマップページの合計数。
Block map read outgoing	書き込み中のページのブロックマップ読み取りの合計数。
Block map found in cache	ブロックマップキャッシュヒットの合計数。
Block map discard required	ページを破棄する必要があったブロックマップ要求の合計数。
Block map wait for page	ページを待つ必要があった要求の合計数。
Block map fetch required	ページフェッチを必要としたリクエストの合計数。
Block map pages loaded	ページフェッチの合計数。
Block map pages saved	保存したページの合計数。
Block map flush count	ブロックマップが発行したフラッシュの合計数。
Invalid advice PBN count	インデックスが無効なアドバイスを返した回数
No space error count.	VDO ボリュームの領域が不足しているために失敗した書き込み要求の数。
Read only error count	VDO ボリュームが読み取り専用モードであるために失敗した書き込み要求の数。
Instance	VDO インスタンス。
512 byte emulation	512 バイトのエミュレーションがボリュームでオンまたはオフになるかを示します。
Current VDO IO requests in progress.	VDO が現在処理している I/O 要求の数。
Maximum VDO IO requests in progress	VDO が処理した同時 I/O 要求の最大数。
Current dedupe queries	現在実行中の重複排除クエリーの数。
Maximum dedupe queries	起動時の重複排除クエリーの最大数。

項目	説明
Dedupe advice valid	重複排除のアドバイスが正しかった回数。
Dedupe advice stale	重複排除のアドバイスが間違っていた回数。
Dedupe advice timeouts	重複排除クエリーがタイムアウトした回数。
Flush out	VDO が基盤となるストレージに送信したフラッシュ要求の数。
Bios in...Bios in partial...Bios out...Bios meta...Bios journal...Bios page cache...Bios out completed...Bio meta completed...Bios journal completed...Bios page cache completed...Bios acknowledged...Bios acknowledged partial...Bios in progress...	これらの統計は、特定のフラグを持つ各カテゴリーの BIOS の数をカウントします。カテゴリーは以下のとおりです。 • **bios in:** VDO が受信したブロック I/O 要求の数。 • **bios in partial:** VDO が受信した部分的なブロック I/O 要求の数。512 バイトのエミュレーションモードにのみ適用されます。 • **bios out:** VDO がストレージデバイスに送信したメタデータブロック以外の I/O 要求の数。 • **bios meta:** VDO がストレージデバイスに送信したメタデータブロックの I/O 要求の数。 • **bios journal:** VDO がストレージデバイスに送信したリカバリージャーナルブロックの I/O 要求の数。 • **bios page cache:** VDO がストレージデバイスに送信したブロックマップ I/O 要求の数。 • **bios out completed:** ストレージデバイスによって完了したメタデータブロック以外の I/O 要求の数。 • **bios meta completed:** ストレージデバイスによって完了したメタデータブロック I/O 要求の数。 • **bios journal completed:** ストレージデバイスによって完了したリカバリージャーナルブロックの I/O 要求の数。 • **bios page cache completed:** ストレージデバイスによって完了したブロックマップ I/O 要求の数。 • **bios acknowledged:** VDO によって確認されたブロック I/O 要求の数。 • **bios acknowledged partial:** VDO によって確認された部分的なブロック I/O 要求の数。512 バイトのエミュレーションモードにのみ適用されます。 • **bios in progress:** まだ確認されていない VDO に送信された bios の数。 フラグには、以下の 3 つのタイプがあります。 • **read:** 書き込み以外の bios (REQ_WRITE フラグが設定されていない bios) の数 • **write:** 書き込み bios (REQ_WRITE フラグが設定されている bios) の数 • **discard:** REQ_DISCARD フラグが設定されている bios の数

項目	説明
読み取りキャッシュアクセス	VDO が読み取りキャッシュを検索した回数。
読み取りキャッシュのヒット	読み取りキャッシュのヒット数。

30.8. /SYSの統計ファイル

実行中の VDO ボリュームの統計は、/sys/kvdo/volume_name/statistics ディレクトリーのファイルから読み込むことができます。ここで、volume_name は、VDO ボリュームの名前になります。これにより、vdostats ユーティリティーによって生成されたデータへの別のインターフェイスが提供されます。これは、シェルスクリプトおよび管理ソフトウェアによるアクセスに適しています。

以下の表に記載されているファイルに加えて、statistics ディレクトリーにファイルがあります。これらの追加の統計ファイルは、将来のリリースでサポートされることは保証されていません。

表30.10 統計ファイル

ファイル	説明
dataBlocksUsed	VDO ボリュームがデータを保存するために現在使用している物理ブロックの数。
logicalBlocksUsed	現在マッピングされている論理ブロックの数。
physicalBlocks	VDO ボリュームに割り当てられている物理ブロックの合計数。
logicalBlocks	VDO ボリュームでマッピングできる論理ブロックの最大数。
モード	VDO ボリュームが通常どおり動作しているか、復旧モードであるか、読み取り専用モードであるかを示します。

第31章 VDO 評価

31.1. 導入部分

VDO は、プライマリーストレージに対してインラインブロックレベルの重複排除、圧縮、およびシンプロビジョニング機能を提供するソフトウェアです。VDO は Linux デバイスマッパーフレームワーク内にインストールします。ここでは、既存の物理ブロックデバイスの所有権を取得し、これらをデータ効率性プロパティーを使用して、新しい高レベルのブロックデバイスに再度マッピングします。具体的には、VDO はこれらのデバイスの実効容量を 10 倍以上にすることができます。これらの利点には追加のシステムリソースが必要であるため、システムパフォーマンスに対する VDO の影響を測定する必要があります。

ストレージベンダーは間違いなく、新しいストレージ製品を評価するために使用する既存の社内テスト計画と専門知識を持っています。VDO レイヤーは、重複排除と圧縮を識別する際に役立つので、さまざまなテストが必要になる場合があります。効果的なテスト計画を立てるには、VDO アーキテクチャーを調べて、以下の項目を確認する必要があります。

- VDO 固有の設定可能なプロパティー (エンドユーザーアプリケーションのパフォーマンスチューニング)
- ネイティブ 4KB ブロックデバイスである場合の影響
- アクセスパターンと、重複排除および圧縮のディストリビューションへの応答
- 高負荷環境でのパフォーマンス (非常に重要)
- アプリケーションに基づいて、コスト、容量、パフォーマンスを分析します

このような要因を事前に考慮しなかった場合は、特定のテストが無効になり、お客様はテストとデータ収集の作業を繰り返す必要がありました。

31.1.1. 期待と成果

この評価ガイドは、ベンダーの内部評価の取り組みを強化するものであり、それを置き換えるものではありません。適度な時間を投資することで、エバリュエーターは、既存のストレージデバイスへの VDO の統合に関する正確な評価を作成することができます。このガイドは、以下を目的としています。

- エンジニアが、テストデバイスから最適な応答を導く設定を特定する際に役立ちます。
- 製品の設定ミスを防ぐため、基本的なチューニングパラメーターの概要を説明します。
- 実際のアプリケーションの結果と比較するための参照として、パフォーマンス結果ポートフォリオを作成します。
- さまざまなワークロードがパフォーマンスやデータ効率にどのように影響するかを特定します。
- VDO 実装により、市場投入までの時間を短縮します。

テスト結果は、Red Hat のエンジニアが、特定のストレージ環境に統合する際の VDO の動作を理解する上で役立ちます。OEM は、重複排除および圧縮機能のあるデバイスを設計する方法と、そのデバイスを最適に使用するためにお客様がアプリケーションを調整する方法を理解します。

本書の手順は、VDO を最も現実的に評価できる条件を提供するように設計されていることに留意してください。テスト手順やパラメーターを変更すると、結果が無効になる場合があります。Red Hat のセールスエンジニアは、テスト計画を変更する際のガイダンスを提供できます。

31.2. テスト環境の準備

VDO を評価する前に、ホストシステムの設定、VDO の設定、およびテスト時に使用されるワークロードを考慮することが重要です。このような選択は、データの最適化 (領域の効率化) およびパフォーマンス (帯域幅と遅延) の両方の観点でベンチマークに影響を及ぼします。テスト計画を作成する際に考慮すべき事項を以下のセクションに記載します。

31.2.1. システム設定

- 利用可能な CPU コアの数およびタイプ。これは、taskset ユーティリティーを使用して制御できます。
- 利用可能なメモリーと、取り付けられているメモリーの合計。

- *ストレージデバイスの設定*
- *Linux カーネルバージョン。Red Hat Enterprise Linux 7 では、提供される Linux カーネルバージョンは 1 つのみであることに注意してください。*
- *インストールされているパッケージ*

31.2.2. VDO 設定

- *パーティション設定スキーム*
- *VDO ボリュームで使用されるファイルシステム*
- *VDO ボリュームに割り当てられた物理ストレージのサイズ*
- *作成した論理 VDO ボリュームのサイズ*
- *スパースまたはデンスインデックス*
- *メモリーサイズの UDS インデックス*
- *VDO のスレッド設定*

31.2.3. ワークロード

- *テストデータを生成するのに使用するツールの種類*
- *同時クライアントの数*
- *書き込まれたデータ内の重複する 4KB ブロックの量*

- 読み書きのパターン
- ワーキングセットのサイズ

各テストが同じディスク環境で実行されるようにするには、特定のテストの間に VDO ボリュームを再作成する必要がある場合があります。詳細は、テストセクションを参照してください。

31.2.4. 対応しているシステム設定

Red Hat は、Intel 64 アーキテクチャー上の Red Hat Enterprise Linux 7 で VDO をテストしました。

VDO のシステム要件については、「システム要件」 を参照してください。

VDO を評価する場合は、以下のユーティリティーを使用することが推奨されます。

- 柔軟な I/O テスター バージョン 2.08 以降。fio パッケージから入手できます。
- sysstat バージョン 8.1.2-2 以降。sysstat パッケージから入手できます。

31.2.5. テスト前のシステム準備

このセクションでは、評価中に最適なパフォーマンスを実現するためにシステム設定を設定する方法について説明します。特定のテストに確立されている暗黙の範囲を超えたテストを行うと、異常な結果により、テスト時間が失われる可能性があります。たとえば、このガイドでは、100 GB のアドレス範囲でランダム読み取りを実行するテストを説明します。500 GB のワーキングセットをテストするには、VDO ブロックマップキャッシュに割り当てられる DRAM の容量を適宜増やす必要があります。

- システム設定
  - CPU が最も高いパフォーマンス設定で実行していることを確認します。
  - BIOS 設定または Linux の cpupower ユーティリティーを使用して、可能であれば

頻度のスケーリングを無効にします。

- 最大スループットを実現できる場合は、Turbo モードを有効にします。Turbo モードでは、テスト結果に多少のばらつきが生じますが、Turbo なしのテストと同等以上のパフォーマンスを発揮します。

- Linux 設定
  - ディスクベースのソリューションの場合、Linux は、キューに入っている複数の読み取り/書き込み要求を処理するための I/O スケジューラーアルゴリズムを複数提供します。デフォルトでは、Red Hat Enterprise Linux は CFQ (完全に公平なキューイング) スケジューラーを使用します。これは、多くの状況でローテーションディスク (ハードディスク) のアクセスを改善する方法で要求を調整します。代わりに、ローテーションディスクに Deadline スケジューラーを使用することを推奨します。Red Hat ラボテストでスループットとレイテンシーが改善されることが分かりました。デバイス設定を次のように変更します。

    ```
    # echo "deadline" > /sys/block/device/queue/scheduler
    ```

  - フラッシュベースのソリューションの場合、noop スケジューラーは、Red Hat ラボテストで優れたランダムアクセススループットとレイテンシーを示します。デバイス設定を次のように変更します。

    ```
    # echo "noop" > /sys/block/device/queue/scheduler
    ```

- ストレージデバイスの設定

ファイルシステム (ext4、XFS など) は、パフォーマンスに固有の影響を与える可能性があります。それらはしばしばパフォーマンス測定を歪め、結果に対する VDO の影響を分離することを困難にします。妥当な場合は、raw ブロックデバイスでパフォーマンスを測定することを推奨します。使用できない場合は、ターゲットの実装で使用されるファイルシステムを使用してデバイスをフォーマットします。

31.2.6. VDO の内部構造

評価を完全かつ成功させるには、VDO メカニズムの一般的な理解が不可欠であると考えています。このような理解は、テスターがテスト計画から逸脱したり、特定のアプリケーションやユースケースをエミュレートするための新しい要因を考案したりする場合に特に重要になります。詳細は、30章VDO 統合 を参照してください。

Red Hat テスト計画は、デフォルトの VDO 設定で動作するように書かれています。新規のテストを開発する場合は、次のセクションに記載されている VDO パラメーターの一部を調整する必要があります。

31.2.7. VDO の最適化

高負荷

最適なパフォーマンスを実現するための最も重要なストラテジーは、ストレージシステムの負荷を表す特性である最適な I/O キューの深さを決定することでしょう。最新のストレージシステムのほとんどは、I/O 深度が高い場合に最適に動作します。VDO のパフォーマンスは、同時要求が多数ある場合に最適に実証されます。

同期と非同期書き込みポリシーの比較

VDO は、2 つの書き込みポリシー (同期または非同期) のいずれかで動作します。デフォルトでは、VDO は基となるストレージデバイスに適切な書き込みポリシーを自動的に選択します。

パフォーマンスをテストする際には、選択した書き込みポリシー VDO を把握しておく必要があります。以下のコマンドは、VDO ボリュームの書き込みポリシーを示しています。

```
# vdo status --name=my_vdo
```

書き込みポリシーの詳細は、「VDO 書き込みポリシーの概要」 および 「VDO 書き込みモードの選択」 を参照してください。

メタデータのキャッシュ

VDO は、論理ブロックアドレスから物理ブロックアドレスへのマッピングの表を保持しており、特定のブロックにアクセスする場合、VDO が関連するマッピングを検索する必要があります。デフォルトでは、VDO は、DRAM で 128MB のメタデータキャッシュを割り当て、一度に 100GB の論理領域に効率的にアクセスできるようにします。テスト計画では、この設定オプションに適したワークロードが生成されます。

設定されたキャッシュサイズよりも大きなワーキングセットの場合は、サービス要求に追加の I/O が必要になります。この場合はパフォーマンスが低下します。追加のメモリーが利用可能な場合は、ブロックマップキャッシュを大きくする必要があります。ブロックマップキャッシュがメモリーに保持できる値よりもワーキングセットが大きい場合は、関連するブロックマップページをルックアップするための追加の I/O ホバーヘッドが発生する可能性があります。

VDO マルチスレッディングの設定

最適なパフォーマンスを実現するには、VDO のスレッド設定を調整する必要があります。VDO ボリュームの作成時にこの設定を変更する方法は、VDO 統合ガイドを参照してください。最適な設定を見つけるためのテストの設計方法については、Red Hat セールスエンジニアにお問い合わせください。

データコンテンツ

VDO は重複排除と圧縮を実行するため、テストデータセットを選択して、このような機能を効果的に使用する必要があります。

31.2.8. 読み取りパフォーマンスのテストに関する特別な考慮事項

読み取りパフォーマンスをテストする際には、以下の要因を考慮する必要があります。

1. *4KB のブロックに 書き込まれたことがない 場合、VDO はストレージへの I/O を実行せず、即座にゼロブロックで応答します。*

2. *4KB のブロックに 書き込まれているが、すべてゼロが含まれている 場合、VDO はストレージへの I/O を実行せず、すぐにゼロブロックで応答します。*

この動作により、読み込むデータがない場合に、読み込みのパフォーマンスが非常に高速になります。これにより、読み取りテストに実際のデータを事前に入力することが不可欠になります。

31.2.9. クロストーク

あるテストが別のテストの結果に影響を及ぼさないようにするため、各テストの繰り返しごとに、新しい VDO ボリュームを作成することが推奨されます。

31.3. データ効率のテスト手順

VDO の検証が成功するかどうかは、適切に設定されたテスト手順に従うかどうかにかかっています。このセクションでは、評価に参加する際に考慮すべきテストの例として、期待される結果とともに従うべき一連の手順を示します。

テスト環境

次のセクションのテストケースでは、テスト環境を以下のように想定します。

- 1 つ以上の Linux 物理ブロックデバイスが使用できる。
- ターゲットブロックデバイス（例： /dev/sdb）が 512 GB を超える。
- 柔軟な I/O テスター(fio)バージョン 2.1.1 以降がインストールされている。
- VDO がインストールされている。

テスト環境を完全に把握するために、各テストの開始時に以下の情報を記録する必要があります。

- 使用される Linux ビルド (カーネルビルド番号を含む)。
- rpm -qa コマンドから取得した、インストールされているパッケージの完全なリスト。
- 完全なシステム仕様:
  - CPU タイプと数量(/proc/cpuinfo で入手可能)
  - インストールされたメモリーと、ベース OS の実行後に利用可能な量(/proc/meminfo で使用可能)。
  - 使用されているドライブコントローラーのタイプ
  - 使用されているディスクのタイプと数量
- 実行中のプロセスの完全なリスト(ps aux など)。

- *VDO で使用するために作成された物理ボリュームおよびボリュームグループの名前(pvs および vgs の一覧)。*
- *VDO ボリューム (存在する場合) のフォーマットに使用されるファイルシステム。*
- *マウントされたディレクトリーの権限。*
- */etc/vdoconf.yaml の内容。*
- *VDO ファイルの場所。*

sosreport を実行すると、必要な多くの情報を取得できます。

ワークロード

VDO を効果的にテストするには、実際のワークロードをシミュレートするデータセットを使用する必要があります。データセットは、さまざまな条件下でのパフォーマンスを実証するために、重複排除や圧縮が可能なデータと可能でないデータの間のバランスを提供する必要があります。

繰り返し可能な特性を持つデータを合成して生成できるツールがいくつかあります。テスト中の使用には、特に VDbench および fio の 2 つのユーティリティーの使用が推奨されます。

本ガイドでは、fio を使用します。評価を成功させるためには、引数を理解することが重要です。

表31.1 fio オプション

引数	説明	値
--size	fio がジョブごとにターゲットに送信するデータの量 (以下の **numjobs** を参照)。	100 GB
--bs	fio が生成する各読み取り/書き込み要求のブロックサイズ。Red Hat は、デフォルトの 4KB の VDO と一致する 4KB のブロックサイズを推奨します。	4k

引数	説明	値
--numjobs	fio がベンチマークを実行するために作成するジョブの数。 各ジョブは、**--size** パラメーターで指定した量のデータを送信します。 1 番目のジョブは、**--offset** パラメーターで指定したオフセットでデバイスにデータを送信します。後続のジョブでは、**--offset_increment** パラメーターが指定されていない限り、ディスクの同じ領域に書き込みます (上書き)。これは、前のジョブがその値で開始した場所からそれぞれのジョブをオフセットします。フラッシュメモリーでピークパフォーマンスを実現するには、最低 2 つのジョブが推奨されます。通常、1 つのジョブで、ローテーションを行うディスク (HDD) のスループットをいっぱいにできます。	1 (HDD) 2 (SSD)
--thread	fio ジョブをフォークするのではなく、スレッドで実行するように指示します。これにより、コンテキストスイッチを制限することでパフォーマンスが向上する可能性があります。	<N/A>
--ioengine	Linux では、fio を使用してテストできる I/O エンジンが複数あります。Red Hat テストでは、非同期のバッファーなしエンジン (**libaio**) が使用されます。別のエンジンに興味がある場合は、Red Hat セールスエンジニアに相談してください。 Linux **libaio**エンジンは、1 つ以上のプロセスが同時にランダムな要求を行っているワークロードを評価するために使用されます。**libaio** は、データが取得される前に、単一のスレッドから複数のリクエストを非同期で作成できるようにします。これにより、リクエストが同期エンジンを介して多くのスレッドによって提供された場合に必要となるコンテキストスイッチの数が制限されます。	libaio
--direct	設定すると、direct は、Linux カーネルのページキャッシュをバイパスしてデバイスにリクエストを送信できるようにします。 Libaio Engine: **libaio** は、**direct** を有効 (=1) にして使用する必要があります。そうしないと、カーネルが、すべての I/O 要求で **sync** API に頼る可能性があります。	1 (libaio)
--iodepth	任意の時点で実行している I/O バッファーの数。 通常、高い **iodepth** の場合、無作為な読み取りまたは書き込みのパフォーマンスが向上します。深度が高い場合は、コントローラーは常にバッチ処理を要求できます。ただし、**iodepth** を高く設定しすぎる (通常は 1K を超える) と、望ましくないレイテンシーが発生する場合があります。Red Hat は、128 から 512 までの **iodepth** を推奨していますが、最後の値はトレードオフであり、アプリケーションがレイテンシーを許容する方法によって異なります。	128 (最小値)
--iodepth_batch_submit	**iodepth** バッファープールが空になり始めるときに作成される I/O の数。このパラメーターは、テスト中の I/O からバッファー作成へのタスクの切り替えを制限します。	16

引数	説明	値
--iodepth_batch_complete	一括処理 (**iodepth_batch_complete**) を送信するまでに完了する I/O の数。このパラメーターは、テスト中の I/O からバッファー作成へのタスクの切り替えを制限します。	16
--gtod_reduce	レイテンシーを計算する時刻の呼び出しを無効にします。この設定は、無効 (=0) にするとスループットが低下するため、レイテンシー測定が必要でない限り有効 (=1) にしてください。	1

31.3.1. VDO テストボリュームの設定

1.512GB の物理ボリュームに、論理サイズが 1TB の VDO ボリュームを作成する

1. ***VDO ボリュームを作成します。***

   - ***同期ストレージで VDO の async モードをテストするには、--writePolicy=async オプションを使用して非同期ボリュームを作成します。***

     ```
     # vdo create --name=vdo0 --device=/dev/sdb \
             --vdoLogicalSize=1T --writePolicy=async --verbose
     ```

   - ***同期ストレージで VDO 同期 モードをテストするには、--writePolicy=sync オプションを使用して同期ボリュームを作成します。***

     ```
     # vdo create --name=vdo0 --device=/dev/sdb \
             --vdoLogicalSize=1T --writePolicy=sync --verbose
     ```

2. ***新しいデバイスを XFS または ext4 ファイルシステムでフォーマットします。***

   - ***XFS の場合:***

     ```
     # mkfs.xfs -K /dev/mapper/vdo0
     ```

   - ***ext4 の場合:***

     ```
     # mkfs.ext4 -E nodiscard /dev/mapper/vdo0
     ```

3. フォーマットしたデバイスをマウントします。

```
# mkdir /mnt/VDOVolume
# mount /dev/mapper/vdo0 /mnt/VDOVolume && \
  chmod a+rwx /mnt/VDOVolume
```

31.3.2. VDO の効率のテスト

2.VDO ボリュームの読み取りと書き込みのテスト

1. VDO ボリュームに 32GB のランダムデータを書き込みます。

```
$ dd if=/dev/urandom of=/mnt/VDOVolume/testfile bs=4096 count=8388608
```

2. VDO ボリュームからデータを読み込み、VDO ボリュームではない別の場所に書き込みます。

```
$ dd if=/mnt/VDOVolume/testfile of=/home/user/testfile bs=4096
```

3. diff を使用して、2 つのファイルを比較します。この場合、ファイルが同じであることが報告されます。

```
$ diff -s /mnt/VDOVolume/testfile /home/user/testfile
```

4. ファイルを VDO ボリュームの 2 番目の場所にコピーします。

```
$ dd if=/home/user/testfile of=/mnt/VDOVolume/testfile2 bs=4096
```

5. 3 番目のファイルと 2 番目のファイルを比較します。これは、ファイルが同じであることを報告するはずです。

```
$ diff -s /mnt/VDOVolume/testfile2 /home/user/testfile
```

3.VDO ボリュームの削除

1. VDO ボリュームに作成されたファイルシステムをアンマウントします。

```
# umount /mnt/VDOVolume
```

2. コマンドを実行して、システムから VDO ボリューム vdo0 を削除します。

```
# vdo remove --name=vdo0
```

3. ボリュームが削除されたことを確認します。VDO パーティションの vdo リストにはリスト がないはずです。

```
# vdo list --all | grep vdo
```

4.重複排除の計測

1. 「VDO テストボリュームの設定」に従って、VDO ボリュームを作成してマウントします。

2. vdo1 から vdo10 までの VDO ボリュームに 10 個のディレクトリーを作成し、テストデータセットのコピーを 10 個保持します。

```
$ mkdir /mnt/VDOVolume/vdo{01..10}
```

3. ファイルシステムに応じて使用されているディスク領域を調べます。

```
$ df -h /mnt/VDOVolume

Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/vdo0      1.5T  198M  1.4T   1% /mnt/VDOVolume
```

結果をリストまとめることを検討してください。

統計	ベアファイルシステム	シード後	10 回コピーした後
使用されるファイルシステムのサイズ	198 MB		
使用されている VDO データ			
使用されている VDO 論理			

4. **次のコマンドを実行して、値を記録します。"Data blocks used" は、VDO で実行している物理デバイスのユーザーデータが使用しているブロック数です。"Logical blocks used" は、最適化前に使用したブロック数になります。これは、計測の開始点として使用されます。**

```
# vdostats --verbose | grep "blocks used"

data blocks used               : 1090
overhead blocks used           : 538846
logical blocks used            : 6059434
```

5. **VDO ボリュームのトップレベルに、データソースファイルを作成します。**

```
$ dd if=/dev/urandom of=/mnt/VDOVolume/sourcefile bs=4096 count=1048576

4294967296 bytes (4.3 GB) copied, 540.538 s, 7.9 MB/s
```

6. **使用中の物理ディスク領域の量を再度調べます。これは、ちょうど書き込まれたファイルのサイズに応じて、使用されているブロック数の増加を示しているはずです。**

```
$ df -h /mnt/VDOVolume

Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/vdo0      1.5T  4.2G  1.4T   1% /mnt/VDOVolume
```

```
# vdostats --verbose | grep "blocks used"

data blocks used               : 1050093 (increased by 4GB)
overhead blocks used           : 538846 (Did not change)
logical blocks used            : 7108036 (increased by 4GB)
```

7. **10 個のサブディレクトリーのそれぞれにファイルをコピーします。**

```
$ for i in {01..10}; do
  cp /mnt/VDOVolume/sourcefile /mnt/VDOVolume/vdo$i
  done
```

8. **再度、使用されている物理ディスク領域 (使用されているデータブロック) の量を確認します。この数は、ファイルシステムのジャーナリングとメタデータによるわずかな増加を除いて、上記手順 6 の結果と類似している必要があります。**

```
$ df -h /mnt/VDOVolume

Filesystem            Size  Used Avail Use% Mounted on
```

```
/dev/mapper/vdo0      1.5T   45G  1.3T   4% /mnt/VDOVolume
```

```
# vdostats --verbose | grep "blocks used"

data blocks used               : 1050836 (increased by 3M)
overhead blocks used           : 538846
logical blocks used            : 17594127 (increased by 41G)
```

9. ***テストデータを書き込む前に見つかった値から、ファイルシステムが使用する領域の値を減算します。これは、ファイルシステムの観点から、このテストで使用される領域の量です。***

10. ***記録された統計で節約する容量を確認します。***

注記:次の表では、値は MB/GB に変換されています。vdostats の "blocks" は 4,096 B です。

統計	ベアファイルシステム	シード後	10 回コピーした後
使用されるファイルシステムのサイズ	198 MB	4.2 GB	45 GB
使用されている VDO データ	4 MB	4.1 GB	4.1 GB
使用されている VDO 論理	23.6 GB*	27.8 GB	68.7 GB

1.6TB フォーマットドライブ用のファイルシステムのオーバーヘッド

5.圧縮の測定

1. ***物理サイズおよび論理サイズが 10GB 以上の VDO ボリュームを作成します。重複排除を無効にし、圧縮を有効にするオプションを追加します。***

```
# vdo create --name=vdo0 --device=/dev/sdb \
        --vdoLogicalSize=10G --verbose \
        --deduplication=disabled --compression=enabled
```

2. ***転送前に VDO 統計を確認します。使用されているデータブロックと論理ブロックを書き留めておきます (両方ともゼロにする必要があります)。***

```
# vdostats --verbose | grep "blocks used"
```

3. 新しいデバイスを XFS または ext4 ファイルシステムでフォーマットします。

   - XFS の場合:

     ```
     # mkfs.xfs -K /dev/mapper/vdo0
     ```

   - ext4 の場合:

     ```
     # mkfs.ext4 -E nodiscard /dev/mapper/vdo0
     ```

4. フォーマットしたデバイスをマウントします。

   ```
   # mkdir /mnt/VDOVolume
   # mount /dev/mapper/vdo0 /mnt/VDOVolume && \
     chmod a+rwx /mnt/VDOVolume
   ```

5. VDO ボリュームを同期して、未完了の圧縮を完了します。

   ```
   # sync && dmsetup message vdo0 0 sync-dedupe
   ```

6. VDO 統計を再検証します。使用されている論理ブロック - 使用されているデータブロックは、ファイルシステム単体の圧縮により保存されている 4KB ブロックの数になります。VDO は、ファイルシステムのオーバーヘッドと、実際のユーザーデータを最適化します。

   ```
   # vdostats --verbose | grep "blocks used"
   ```

7. /lib のコンテンツを VDO ボリュームにコピーします。合計サイズを記録します。

   ```
   # cp -vR /lib /mnt/VDOVolume

   ...
   sent 152508960 bytes  received 60448 bytes  61027763.20 bytes/sec
   total size is 152293104  speedup is 1.00
   ```

8. Linux キャッシュと VDO ボリュームを同期します。

```
# sync && dmsetup message vdo0 0 sync-dedupe
```

9. VDO 統計を再検証します。使用されている論理ブロックおよびデータブロックを確認します。

```
# vdostats --verbose | grep "blocks used"
```

- 使用されている論理ブロック - 使用されるデータブロックは、/lib ファイルのコピーに使用されるスペースの量(4 KB ブロック単位)を表します。
- 合計サイズ (「4.重複排除の計測」 の表から) - (使用される論理-使用されるデータブロック * 4096) = 圧縮により節約されたバイト数

10. VDO ボリュームを削除します。

```
# umount /mnt/VDOVolume && vdo remove --name=vdo0
```

6.VDO 圧縮の効率のテスト

1. 「VDO テストボリュームの設定」に従って、VDO ボリュームを作成してマウントします。

2. ボリュームを削除せずに、「4.重複排除の計測」 と 「5.圧縮の測定」 で実験を繰り返します。vdostats の領域節約への変更を監視します。

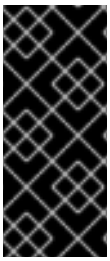
3. 用意したデータセットを試します。

7.TRIM および DISCARD について

シンプロビジョニングにより、論理ストレージまたは仮想ストレージ領域を基盤の物理ストレージよりも大きくすることができます。ファイルシステムなどのアプリケーションは、ストレージのより大きな仮想層で実行すると利点があります。データの重複排除などのデータ効率化技術は、すべてのデータを保存するために必要な物理データブロックの数を減らします。このストレージ削減効果を活用する

には、物理ストレージ層が、アプリケーションデータが削除されたタイミングを把握する必要があります。

従来のファイルシステムでは、データが削除されたときに基礎となるストレージに通知する必要はありませんでした。シンプロビジョニングされたストレージと連携するファイルシステムは、論理ブロックが不要になったときにストレージシステムに通知するために、TRIM コマンドまたは DISCARD コマンドを送信します。これらのコマンドは、discard マウントオプションを使用してブロックが削除されるたびに送信できます。または、fstrim などのユーティリティーを実行して、どの論理ブロックが使用されていないかを検出し、TRIM または DISCARD コマンドの形式でストレージシステムに情報を送信するようにファイルシステムに指示することで、制御された方法でこれらのコマンドを送信できます。



重要

シンプロビジョニングの仕組みに関する詳細は、『Red Hat Enterprise Linux 7 Logical Volume Manager Administration Guide』の*Thinly-Provisioned Logical Volumes (Thin Volumes)*を参照してください。

これがどのように機能するかを確認するには、以下を実行します。

1. 「VDO テストボリュームの設定」 に従って、新しい VDO 論理ボリュームを作成してマウントします。

2. ファイルシステムをトリミングして、不要なブロックを削除します (これには長い時間がかかる場合があります)。

```
# fstrim /mnt/VDOVolume
```

3. 以下を入力して、以下の表に初期状態を記録します。

```
$ df -m /mnt/VDOVolume
```

ファイルシステムで使用されている容量を確認し、vdostats を実行して、使用されている物理データブロックと論理データブロックの数を確認します。

4. VDO 上で実行中のファイルシステムに、重複しないデータを含む 1 GB ファイルを作成します。

-

```
$ dd if=/dev/urandom of=/mnt/VDOVolume/file bs=1M count=1K
```

次に、同じデータを収集します。ファイルシステムはさらに 1GB を使用する必要があり、使用されるデータブロックと使用される論理ブロックも同様に増加しました。

5. fstrim /mnt/VDOVolume を実行し、新しいファイルの作成後に、これが影響を与えないことを確認します。

6. 1 GB ファイルを削除します。

```
$ rm /mnt/VDOVolume/file
```

パラメーターを確認し、記録します。ファイルシステムは、ファイルが削除されたことを認識していますが、ファイルの削除が基礎となるストレージに通知されていないため、物理ブロックまたは論理ブロックの数に変更がありませんでした。

7. fstrim /mnt/VDOVolume を実行し、同じパラメーターを記録します。fstrim は、ファイルシステムの空きブロックを検索し、未使用のアドレスの VDO ボリュームに TRIM コマンドを送信します。これにより、関連する論理ブロックが解放され、VDO が TRIM を処理して基礎となる物理ブロックを解放します。

Step	使用されているファイル領域 (MB)	使用されているデータブロック	使用されている論理ブロック
初期			
1GB ファイルを追加			
fstrim を実行			
1GB ファイルを削除			
fstrim を実行			

この演習では、TRIM プロセスが必要で、基礎となるストレージが容量の使用率について正確な知識を持つことができます。fstrim は、多くのブロックを一度に分析して効率を向上させるコマンドラインツールです。別の方法として、マウント時に file system discard オプションを使用する方法があります。discard オプションは、各ファイルシステムブロックの削除後に基礎となるストレージを更新します。これによりスループットは遅くなりますが、使用率認識度は高くなります。未使用のブロックを

TRIM または DISCARD する必要性は、VDO に固有のものではないことを理解することも重要です。シンプロビジョニングされたストレージシステムにも同じ課題があります。

31.4. パフォーマンステストの手順

本セクションの目的は、VDO をインストールしたデバイスのパフォーマンスプロファイルを構築することです。各テストは、VDO のインストール有無にかかわらず実行する必要があります。これにより、VDO のパフォーマンスをベースシステムのパフォーマンスに関連付けて評価できるようになります。

31.4.1. フェーズ 1 - I/O 深さの影響、固定 4KB ブロック

これらのテストの目的は、アプリケーションの最適なスループットと最小のレイテンシーを生成する I/O 深度を決定することです。VDO は、従来のストレージデバイスで使用されていた従来の 512 B ではなく、4KB のセクターサイズを使用します。セクターサイズが大きいため、大容量のストレージをサポートし、パフォーマンスを向上させ、ほとんどのオペレーティングシステムで使用されるキャッシュバッファーサイズと一致させることができます。

1. 4KB I/O および I/O 深度が 1、8、16、32、64、128、256、512、1024 で、フォーコーナーテストを実行します。

   - 100% シーケンシャル読み取り (固定 4KB *)
   - 100% シーケンシャル書き込み (固定 4KB)
   - 100% ランダム読み取り (固定 4KB *)
   - 100% ランダム書き込み (固定 4KB **)

   * 最初に write fio ジョブを実行して、読み取りテスト中に読み取り可能な領域を事前に入力します。

** 4 KB のランダム書き込み I/O の実行後、VDO ボリュームを再作成します。

シェルテスト入力要因の例 (書き込み):

```
# for depth in 1 2 4 8 16 32 64 128 256 512 1024 2048; do
  fio --rw=write --bs=4096 --name=vdo --filename=/dev/mapper/vdo0 \
      --ioengine=libaio --numjobs=1 --thread --norandommap --runtime=300\
      --direct=1 --iodepth=$depth --scramble_buffers=1  --offset=0 \
      --size=100g
  done
```

2. 各データポイントでスループットとレイテンシーを記録してから、グラフ化します。

3. テストを繰り返してフォーコーナーテストを完了します: **--rw=randwrite**、**--rw=read**、および **--rw=randread.**

結果は以下のようなグラフになります。重要な点は、範囲全体の動作と、I/O 深度が増えるとスループットの向上が低下することが証明される変曲点です。おそらく、シーケンシャルアクセスとランダムアクセスは異なる値でピークに達しますが、すべてのタイプのストレージ設定で異なる可能性があります。図31.1「I/O 深度分析」 で、各パフォーマンス曲線の急な折れ曲がりに注意してください。マーカー 1 はポイント X でのピークシーケンシャルスループットを識別し、マーカー 2 はポイント Z でのピークランダム 4KB スループットを識別します。

- この特定のアプライアンスは、シーケンシャル 4 KB I/O 深度 > X の恩恵を受けません。この深度を超えると、帯域幅の利得が減少し、I/O 要求が増えるごとに平均要求レイテンシーが 1:1 に増加します。

- この特定のアプライアンスは、ランダム 4KB の I/O 深度 > Z の恩恵を受けません。この深度を超えると、帯域幅の利得が減少し、追加の I/O 要求ごとに平均要求レイテンシーが 1:1 増加します。

図31.1 I/O 深度分析

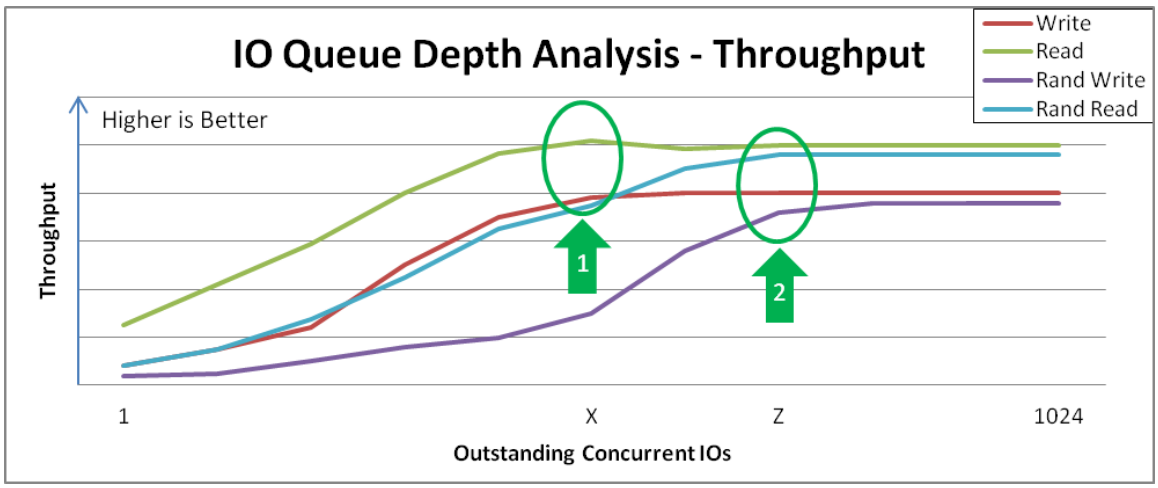
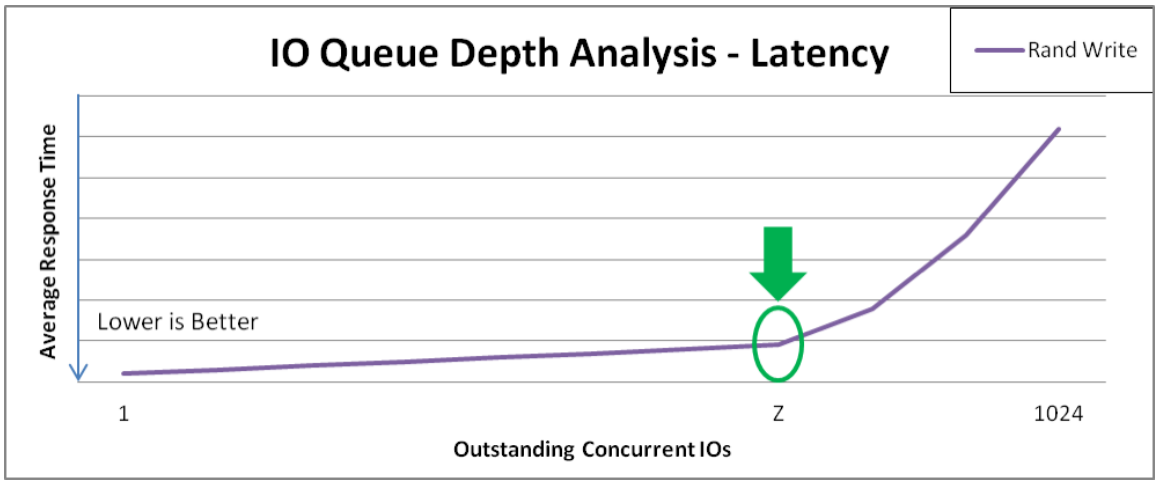


図31.2「ランダム書き込みの I/O の増加による遅延応答」 は、*図31.1「I/O 深度分析」* の曲線が急に折れ曲がった後のランダムな書き込みレイテンシーの例を示しています。ベンチマークの実践では、この点で応答時間のペナルティーが最小となる最大スループットを検証する必要があります。このアプライアンスの例のテスト計画を進めるにつれ、I/O 深度 = Z で追加データを収集します。

図31.2 ランダム書き込みの I/O の増加による遅延応答



31.4.2. フェーズ 2 - I/O リクエストサイズの影響

このテストの目的は、前のステップで決定された最適な I/O 深度でテスト対象システムの最高のパフォーマンスを生み出すブロックサイズを理解することです。

1. 8KB から 1MB の範囲でさまざまなブロックサイズ (2 の累乗) を使用して、固定 I/O 深度で フォーコーナーテストを実行します。読み取る領域を事前に入力し、テストの合間にボリュームを再作成することを忘れないでください。

2. I/O 深度を *「フェーズ 1 - I/O 深さの影響、固定 4KB ブロック」* で決定された値に設定します。

テスト入力要因の例 (読み取り):

```
# z=[see previous step]
# for iosize in 4 8 16 32 64 128 256 512 1024; do
  fio --rw=write --bs=$iosize\k --name=vdo --filename=/dev/mapper/vdo0
      --ioengine=libaio --numjobs=1 --thread --norandommap --runtime=300
      --direct=1 --iodepth=$z --scramble_buffers=1 --offset=0 --size=100g
  done
```

3. 各データポイントでスループットとレイテンシーを記録してから、グラフ化します。

4. テストを繰り返してフォーコーナーテストを完了します: **--rw=randwrite**、**--rw=read**、および **--rw=randread.**

この結果には、いくつかの重要な点があります。この例では、以下のように設定されています。

- シーケンシャル書き込みは、要求サイズ **Y** でピークスループットに達します。この曲線は、設定可能であるか、特定の要求サイズによって自然に支配されるアプリケーションが、パフォーマンスをどのように認識するかを示しています。**4KB** の **I/O** はマージの恩恵を受ける可能性があるため、要求サイズが大きいほどスループットが向上することがよくあります。

- シーケンシャル読み取りのスループットは、**Z** 点で似たようなピークスループットに到達します。これらのピークの後、**I/O** が完了するまでの全体的な遅延は、追加のスループットなしで増加することに注意してください。このサイズよりも大きな **I/O** を受け付けないようにデバイスを調整することが推奨されます。

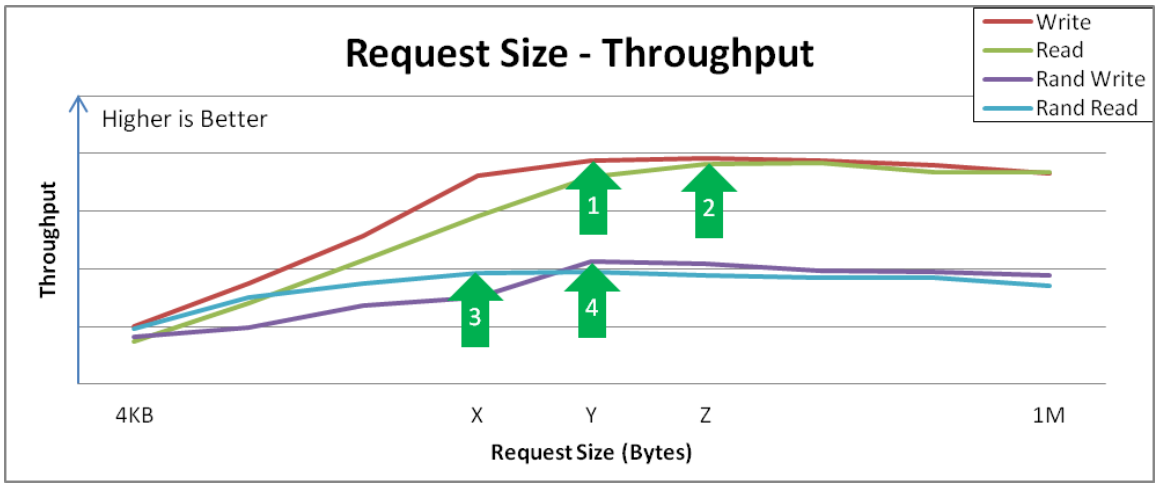
- ランダム読み取りは、ポイント **X** でピークスループットを達成します。デバイスによっては、大規模なリクエストサイズのランダムアクセスで、ほぼシーケンシャルスループット率を実現する可能性がありますが、純粋なシーケンシャルアクセスと異なる場合は、より多くのペナルティーを受けることになります。

- ランダム書き込みは、ポイント **Y** でピークスループットを達成します。ランダム書き込みは、重複排除デバイスの相互作用が最も多く、**VDO** は、特に要求サイズや **I/O** 深度が大きい場合に高いパフォーマンスを実現します。

このテスト図31.3「要求サイズとスループットの分析、および重要な変曲点」の結果は、ストレージデバイスの特性や、特定のアプリケーションのユーザーエクスペリエンスを把握する上で役立ちま

す。異なるリクエストサイズでパフォーマンスを向上させるためにさらに調整が必要な場合は、Red Hat セールスエンジニアに相談してください。

図31.3 要求サイズとスループットの分析、および重要な変曲点



31.4.3. フェーズ 3 - 読み取り/書き込み I/O のミキシングの影響

このテストの目的は、混合 I/O 負荷 (読み取り/書き込み) で提示されたときに VDO を持つアプライアンスがどのように動作するかを理解し、最適なランダムキューの深度と、4KB から 1 MB の要求サイズでの読み取り/書き込み混合の影響を分析することです。ケースに適したものを使用してください。

1. *固定の I/O 深度、8 KB から 256 KB の範囲の可変ブロックサイズ (2 の累乗)、0% から初めて 10 % 刻みで読み取り率を設定し、フォーコーナーテストを実行します。読み取る領域を事前に入力し、テストの合間にボリュームを再作成することを忘れないでください。*

2. *I/O 深度を 「フェーズ 1 - I/O 深さの影響、固定 4KB ブロック」 で決定された値に設定します。*

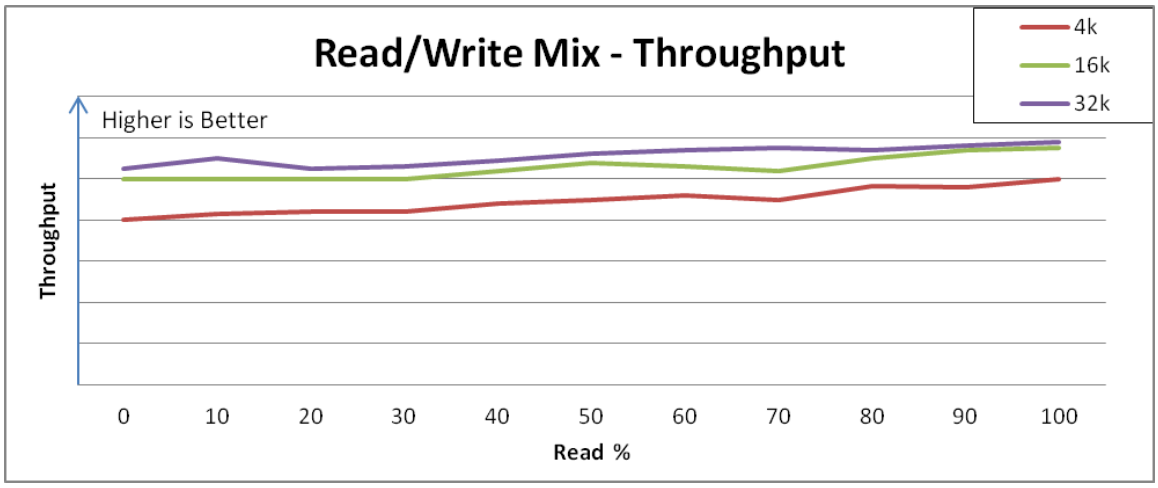
テスト入力要因の例 (読み取り/書き込みの組み合わせ):

```
# z=[see previous step]
# for readmix in 0 10 20 30 40 50 60 70 80 90 100; do
    for iosize in 4 8 16 32 64 128 256 512 1024; do
      fio --rw=rw --rwmixread=$readmix --bs=$iosize\k --name=vdo \
          --filename=/dev/mapper/vdo0 --ioengine=libaio --numjobs=1 --thread \
          --norandommap --runtime=300 --direct=0 --iodepth=$z --scramble_buffers=1 \
          --offset=0 --size=100g
    done
  done
```

3. 各データポイントでスループットとレイテンシーを記録してから、グラフ化します。

図31.4「パフォーマンスは、さまざまな読み取りと書き込みの組み合わせ混合で一貫している」は、VDO が I/O 負荷に応答する方法の例を示しています。

図31.4 パフォーマンスは、さまざまな読み取りと書き込みの組み合わせ混合で一貫している



パフォーマンス (集約) とレイテンシー (集約) は、読み取りと書き込みが混在する範囲全体で比較的一貫性があり、より低い最大書き込みスループットからより高い最大書き込みスループットへと移ります。

この動作はストレージによって異なる場合がありますが、負荷が変化してもパフォーマンスが一貫していることや、特定の読み取りと書き込みの組み合わせを示すアプリケーションのパフォーマンス期待値を理解できることが重要となります。予期しない結果が見つかった場合、Red Hat セールスエンジニアは、変更が必要なのが VDO なのか、ストレージデバイス自体なのかを理解するためのサポートを提供することができます。

注記: 同様の応答の一貫性を示さないシステムは、多くの場合、設定が最適ではないことを意味します。この場合は、Red Hat セールスエンジニアにお問い合わせください。

31.4.4. フェーズ 4: アプリケーション環境

この最終テストの目的は、実際のアプリケーション環境にデプロイする際に、VDO を使用したシステムがどのように動作するかを理解することです。可能であれば、実際のアプリケーションを使用し、これまでに学んだ知識を活用します。アプライアンスで許容されるキューの深さを制限することを検討し、可能であれば、VDO のパフォーマンスに最も有益なブロックサイズでリクエストを発行するようにアプリケーションを調整します。

リクエストサイズ、I/O 負荷、読み取り/書き込みパターンなどは、アプリケーションのユースケー

ス (ファイラー、仮想デスクトップ、データベース) により異なるため、一般に予測は困難であり、アプリケーションも特定の操作またはマルチテナントのアクセスにより、I/O のタイプが異なることが多いです。

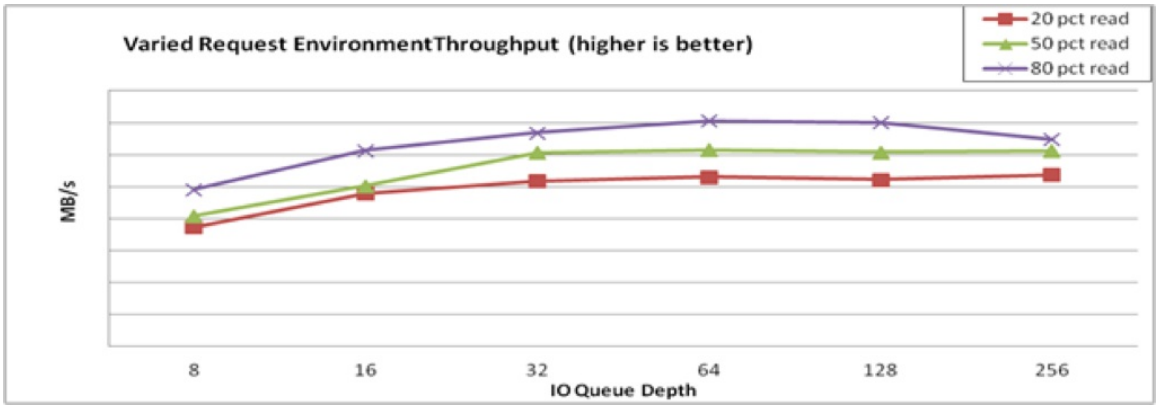
最終テストでは、混合環境における一般的な VDO パフォーマンスが示されます。想定される環境について、より具体的な詳細が分かっている場合は、その設定もテストします。

テスト入力要因の例 (読み取り/書き込みの組み合わせ):

```
# for readmix in 20 50 80; do
    for iosize in 4 8 16 32 64 128 256 512 1024; do
      fio --rw=rw --rwmixread=$readmix --bsrange=4k-256k --name=vdo \
          --filename=/dev/mapper/vdo0 --ioengine=libaio --numjobs=1 --thread \
          --norandommap --runtime=300 --direct=0 --iodepth=$iosize \
          --scramble_buffers=1 --offset=0 --size=100g
    done
  done
```

各データポイントでスループットとレイテンシーを記録してから、グラフ化します (図31.5「混合環境パフォーマンス」)。

図31.5 混合環境パフォーマンス



31.5. 問題の報告

VDO の操作中に問題が発生した場合は、Red Hat セールスエンジニアが問題を再現できるように、可能な限り多くの情報を収集することが重要です。

問題レポートには、以下の内容が含まれている必要があります。

- テスト環境の詳細な説明。詳細は 「テスト環境」 を参照してください。
- VDO 設定
- 問題を生成したユースケース
- エラー発生時に実行していたアクション
- コンソールまたは端末におけるエラーメッセージのテキスト
- カーネルログファイル
- カーネルクラッシュダンプ (利用可能な場合)
- sosreport の結果で、これは、Linux 全体を記述するデータを取得します。

31.6. まとめ

VDO をストレージシステムに統合する場合は、この計画やその他の適切に構造化された評価計画を行うことが重要です。評価プロセスは、パフォーマンスを理解し、潜在的な互換性の問題を把握する際に重要です。この評価結果のコレクションは、重複排除と圧縮を示すだけでなく、VDO を実装するシステムのパフォーマンスプロファイルを提供します。結果は、実際のアプリケーションで達成された結果が期待どおりで妥当なものか、期待を下回るものかを判断する上で役立ちます。最後に、この結果をもとに、VDO が有利に動作するアプリケーションを予測することもできます。

付録A ストレージ管理に関連する RED HAT カスタマーポータルラボ

Red Hat Customer Portal Labs のツールは、パフォーマンスの向上、問題のトラブルシューティング、セキュリティー問題の特定、設定の最適化に役立てていただくために開発しました。この付録では、ストレージ管理に関連する Red Hat カスタマーポータルラボの概要を説明します。すべての Red Hat Customer Portal Labs は https://access.redhat.com/labs/ からご利用いただくことができます。

SCSI デコーダー

SCSI decoder は、SCSI エラーメッセージの理解が難しいため、SCSI エラーメッセージを /log/* ファイルまたはログファイルスニペットにデコードするように設計されています。

SCSI decoder を使用して、各 SCSI エラーメッセージを個別に診断し、問題を効率的に解決するための解決策を入手します。

FILE SYSTEM LAYOUT CALCULATOR

File System Layout Calculator は、ユーザーが、現在のストレージまたは予定されるストレージを記述したストレージオプションを提供した後に、ext3、ext4、xfs のファイルシステムの作成に最適なパラメーターを決定します。カーソルを疑問符 ("?") に移動すると、特定のオプションの簡単な説明が表示されます。スクロールすると、すべてのオプションの概要が表示されます。

File System Layout Calculator を使用して、指定した RAID ストレージに指定したパラメーターを持つファイルシステムを作成するコマンドを生成します。生成されたコマンドをコピーして root として実行し、必要なファイルシステムを作成します。

LVM RAID CALCULATOR

LVM RAID Calculator は、ストレージオプションを指定した後に、所定の RAID ストレージの論理ボリューム (LVM) を作成するために最適なパラメーターを決定します。カーソルを疑問符 ("?") に移動すると、特定のオプションの簡単な説明が表示されます。スクロールすると、すべてのオプションの概要が表示されます。

LVM RAID Calculator は、特定の RAID ストレージに LVM を作成する一連のコマンドを生成します。生成されたコマンドを root として 1 つずつコピーして実行し、必要な LVM を作成します。

ISCSI HELPER

iSCSI Helper は、インターネットプロトコル (IP) ネットワーク越しにブロックレベルのストレージを提供し、サーバーの仮想環境でストレージプールを使用できるようにします。

iSCSI ヘルパー を使用し て、指定した設定に従って設定された iSCSI ターゲット（サーバー）または iSCSI イニシエーター（クライアント）のロール用にシステムを準備するスクリプトを生成します。

SAMBA CONFIGURATION HELPER

Samba 設定ヘルパー は、Samba 経由で基本的なファイルおよびプリンターの共有を提供する設定を作成します。

- *Server をクリックして、基本的なサーバー設定を指定します。*
- *Shares をクリックして、共有するディレクトリーを追加します。*
- *Server をクリックして、接続されたプリンターを個別に追加します。*

MULTIPATH HELPER

マルチパスヘルパー は、Red Hat Enterprise Linux 5、6、および 7 でマルチパスデバイスに最適な設定を作成します。以下の手順に従って、カスタムエイリアスやデバイスのブラックリストなど、高度なマルチパス設定を作成できます。

Multipath Helper は、確認用に multipath.conf ファイルも提供します。必要な設定が終了したら、サーバーにインストールスクリプトをダウンロードして実行します。

NFS HELPER

NFS Helper を使用すれば、新しい NFS サーバーやクライアントの設定が簡単に行えます。手順に従って、エクスポートとマウントのオプションを指定します。次に、ダウンロード可能な NFS 設定スクリプトを生成します。

MULTIPATH CONFIGURATION VISUALIZER

Multipath Configuration Visualizer は、sosreport 内のファイルを分析し、マルチパス設定を視覚化する図を提供します。Multipath Configuration Visualizer を使用して、以下を表示します。

- *サーバー側の Host Bus Adapter (HBA)、ローカルデバイス、および iSCSI デバイスを含むホストコンポーネント*

- ストレージのストレージコンポーネント
- サーバーとストレージとの間のファブリック、またはイーサネットのコンポーネント
- 上記の全コンポーネントのパス

.xz、.gz、または .bz2 のフォーマットに圧縮された sosreport をアップロードするか、クライアント側の分析のためにソースとして選択したディレクトリーに sosreport を抽出することができます。

RHEL BACKUP AND RESTORE ASSISTANT

RHEL Backup and Restore Assistant は、バックアップツールおよび復元ツールの情報と、Linux の使用に関する一般的なシナリオを提供します。

対象ツール:

- dump および restore: ext2、ext3、および ext4 のファイルシステムのバックアップ
- tar および cpio: ファイルやフォルダーをアーカイブまたは復元するため（特にテープドライブをバックアップする場合）。
- rsync: バックアップ操作の実行および場所間でのファイルとディレクトリーの同期
- dd: 関連するファイルシステムやオペレーティングシステムとは独立してブロックごとにファイルをコピーする場合。

対象シナリオ:

- 障害回復

- *ハードウェアの移行*
- *パーティションテーブルのバックアップ*
- *重要なフォルダーのバックアップ*
- *増分バックアップ*
- *差分バックアップ*

付録B 更新履歴

改訂 4-10 非同期更新	Mon Aug 10 2020	Marek Suchanek
改訂 4-09 非同期更新	Mon Jan 7 2019	Marek Suchanek
改訂 4-08 7.6 GA 公開用ドキュメントの準備	Mon Oct 23 2018	Marek Suchanek
改訂 4-07 非同期更新	Thu Sep 13 2018	Marek Suchanek
改訂 4-00 7.5 GA 公開用ドキュメントバージョン	Fri Apr 6 2018	Marek Suchanek
改訂 3-95 非同期更新	Thu Apr 5 2018	Marek Suchanek
改訂 3-93 新しい章: VDO の統合	Mon Mar 5 2018	Marek Suchanek
改訂 3-92 非同期更新	Fri Feb 9 2018	Marek Suchanek
改訂 3-90 7.5 Alpha 公開用バージョン	Wed Dec 6 2017	Marek Suchanek
改訂 3-86 非同期更新	Mon Nov 6 2017	Marek Suchanek
改訂 3-80 7.4 GA 公開用ドキュメントバージョン	Thu Jul 27 2017	Milan Navratil
改訂 3-77 非同期更新	Wed May 24 2017	Milan Navratil
改訂 3-68 7.3 GA リリースのバージョン	Fri Oct 21 2016	Milan Navratil
改訂 3-67 非同期更新	Fri Jun 17 2016	Milan Navratil
改訂 3-64 7.2 GA リリース向けのバージョン。	Wed Nov 11 2015	Jana Heves
改訂 3-33 7.1 GA 向けバージョン	Wed Feb 18 2015	Jacquelynn East
改訂 3-26 Ceph の概要を追加	Wed Jan 21 2015	Jacquelynn East
改訂 3-22	Thu Dec 4 2014	Jacquelynn East

7.1 ベータ版

改訂 3-4 targetcli に新しい章を追加	Thu Jul 17 2014	Jacquelynn East
改訂 3-1 7.0 GA リリース向けバージョン	Tue Jun 3 2014	Jacquelynn East

索引

シンボル

/boot/ ディレクトリー, /boot/ ディレクトリー

/dev/shm, df コマンド

/etc/fstab, ext3 ファイルシステムへの変換, /etc/fstabを使用した NFS ファイルシステムのマウント, ファイルシステムのマウント

/etc/fstab ファイル

ディスククォータの有効化, クォータの有効化

/home、/opt、/usr/local には別々のパーティションを用意する

ストレージをインストールする際の注意点, /home、/opt、/usr/local には別々のパーティションを用意する

/local/directory (クライアント設定、マウント)

NFS, NFS クライアントの設定

/proc

/proc/devices, /proc 仮想ファイルシステム

/proc/filesystems, /proc 仮想ファイルシステム

/proc/mdstat, /proc 仮想ファイルシステム

/proc/mounts, /proc 仮想ファイルシステム

/proc/mounts/, /proc 仮想ファイルシステム

/proc/partitions, /proc 仮想ファイルシステム

/proc/devices

仮想ファイルシステム (/proc), /proc 仮想ファイルシステム

/proc/filesystems

仮想ファイルシステム (/proc), /proc 仮想ファイルシステム

/proc/mdstat
仮想ファイルシステム (/proc), /proc 仮想ファイルシステム

/proc/mounts
仮想ファイルシステム (/proc), /proc 仮想ファイルシステム

/proc/mounts/
仮想ファイルシステム (/proc), /proc 仮想ファイルシステム

/proc/partitions
仮想ファイルシステム (/proc), /proc 仮想ファイルシステム

/remote/export (クライアント設定、 マウント)
NFS, NFS クライアントの設定

アンマウント, ファイルシステムのアンマウント

イニシエーターの実装
オフロードとインターフェイスのバインディング
iSCSI, 利用可能な iface 設定の表示

インストーラーのサポート
RAID, Anaconda インストーラーでの RAID のサポート

インストールストレージの設定
/home、/opt、/usr/local には別々のパーティションを用意する, /home、/opt、/usr/local には別々のパーティションを用意する
DIF/DIX を有効にしているブブロックデバイス, DIF/DIX を有効にしているブロックデバイス
IBM System Z における DASD デバイスと zFCP デバイス, IBM System Z における DASD デバイスと zFCP デバイス
iSCSI の検出および設定, iSCSI の検出および設定
LUKS/dm-crypt の使用、 ブロックデバイスの暗号化, LUKS を使用したブロックデバイスの暗号化
updates, ストレージをインストールする際の注意点
チャネルコマンドワード (CCW), IBM System Z における DASD デバイスと zFCP デバイス
古い BIOS RAID メタデータ, 古い BIOS RAID メタデータ
新着情報, ストレージをインストールする際の注意点

インタラクティブな操作 (xfsrestore)
XFS, 復元

インデックスキー

FS-Cache, FS-Cache

エキスパートモード (xfs_quota)
XFS, XFS クォータ管理

エクスポートしたファイルシステム
ディスクレスシステム, ディスクレスクライアントのエクスポートしたファイルシステムの設定

エラーメッセージ
書き込みバリア, 書き込みバリアの有効化と無効化

オフラインステータス
Linux SCSI レイヤー, SCSI コマンドタイマーおよびデバイスステータスの制御

オフロードとインターフェイスのバインディング
iSCSI, iSCSI オフロードおよびインターフェイスバインディングの設定

オプション (クライアント設定、 マウント)
NFS, NFS クライアントの設定

オンラインストレージ
トラブルシューティング, オンラインストレージ設定のトラブルシューティング
ファイバーチャネル, ファイバーチャネル
概要, オンラインストレージ管理
sysfs, オンラインストレージ管理

オンライン論理ユニット
読み取り/書き込み状態の変更, オンライン論理ユニットの読み取り/書き込み状態の変更

キャッシュの共有
FS-Cache, キャッシュの共有

キャッシュの設定
FS-Cache, キャッシュの設定

キャッシュカリング制限
FS-Cache, キャッシュカリング制限の設定

キャッシュバックエンド
FS-Cache, FS-Cache

クォータの管理

XFS, XFS クォータ管理

コヒーレンシーデータ
FS-Cache, FS-Cache

コマンドタイマー (SCSI)
Linux SCSI レイヤー, コマンドタイマー

サイズ変更
ext4, ext4 ファイルシステムのサイズ変更

サイズ変更された論理ユニット、サイズ変更, オンライン論理ユニットのサイズの変更

サイズ変更した論理ユニット, オンライン論理ユニットのサイズの変更

サイト設定ファイルの上書き/拡張 (autofs)
NFS, autofs の設定

サーバー (クライアント設定、マウント)
NFS, NFS クライアントの設定

システム情報
ファイルシステム, ファイルシステム情報の収集
/dev/shm, df コマンド

ストライド (ストライプジオメトリーの指定)
ext4, Ext4 ファイルシステムの作成

ストライピング
RAID, RAID レベルとリニアサポート
RAID の基礎, RAID (Redundant Array of Independent Disks)

ストライプジオメトリー
ext4, Ext4 ファイルシステムの作成

ストレージの相互接続、スキャン, ストレージの相互接続のスキャン

ストレージの相互接続のスキャン, ストレージの相互接続のスキャン

ストレージをインストールする際の注意点
/home、/opt、/usr/local には別々のパーティションを用意する, /home、/opt、/usr/local には別々のパーティションを用意する
DIF/DIX を有効にしているブブロックデバイス, DIF/DIX を有効にしているブロックデバイス
IBM System Z における DASD デバイスと zFCP デバイス, IBM System Z における DASD デバイスと zFCP デバイス

iSCSI の検出および設定, *iSCSI の検出および設定*

LUKS/dm-crypt の使用、ブロックデバイスの暗号化, *LUKS を使用したブロックデバイスの暗号化*

updates, *ストレージをインストールする際の注意点*

チャネルコマンドワード (CCW), *IBM System Z における DASD デバイスと zFCP デバイス*

古い BIOS RAID メタデータ, *古い BIOS RAID メタデータ*

新着情報, *ストレージをインストールする際の注意点*

ストレージアクセスパラメーター
I/O のアライメントとサイズ, *ストレージアクセスパラメーター*

ストレージデバイスへのパス、削除, *ストレージデバイスへのパスの削除*

ストレージデバイスへのパス、追加, *ストレージデバイスまたはパスの追加*

ストレージデバイスへのパスの削除, *ストレージデバイスへのパスの削除*

ストレージデバイスへのパスの追加, *ストレージデバイスまたはパスの追加*

スループットクラス
ソリッドステートディスク, *ソリッドステートディスクのデプロイメントガイドライン*

スレーブマウント, *マウントの共有*

スワップ領域, *swap 領域*
expanding, *スワップ領域の追加*

file
作成, *スワップファイルの作成*, *スワップファイルの削除*

LVM2
作成, *スワップの LVM2 論理ボリュームの作成*

削除中, *スワップの LVM2 論理ボリュームの削除*

拡張, *LVM2 論理ボリュームでのスワップ領域の拡張*

縮小, *LVM2 論理ボリュームでのスワップ領域の縮小*

作成, *スワップ領域の追加*

削除中, *スワップ領域の削除*

推奨サイズ, *swap 領域*

移動, *Swap 領域の移動*

セッションの実行、以下に関する情報の取得
iSCSI API, *iSCSI API*

ソフトウェア iSCSI

iSCSI, *ソフトウェア iSCSI 用の iface の設定*

オフロードとインターフェイスのバインディング
iSCSI, *ソフトウェア iSCSI 用の iface の設定*

ソフトウェア iSCSI の iface
オフロードとインターフェイスのバインディング
iSCSI, *ソフトウェア iSCSI 用の iface の設定*

ソフトウェア RAID (参照 RAID)

ソリッドステートディスク
deployment, *ソリッドステートディスクのデプロイメントガイドライン*

SSD, *ソリッドステートディスクのデプロイメントガイドライン*

TRIM コマンド, *ソリッドステートディスクのデプロイメントガイドライン*

スループットクラス, *ソリッドステートディスクのデプロイメントガイドライン*

デプロイメントのガイドライン, *ソリッドステートディスクのデプロイメントガイドライン*

ターゲット
iSCSI, *iSCSI ターゲットへのログイン*

ダイレクトマップサポート (autofs バージョン 5)
NFS, *バージョン 4 と比較した autofs バージョン 5 の改善点*

ダンプレベル
XFS, *バックアップ*

ダーティーログ (XFS ファイルシステムを修復)
XFS, *XFS ファイルシステムの修復*

ダーティーログを使用した XFS ファイルシステムの修復
XFS, *XFS ファイルシステムの修復*

チャネルコマンドワード (CCW)
ストレージをインストールする際の注意点, *IBM System Z における DASD デバイスと zFCP デバイス*

ツール (パーティションおよびその他のファイルシステム機能用)
I/O のアライメントとサイズ, *パーティションおよびファイルシステムツール*

ディスククォータ, *ディスククォータ*
グループごとに割り当て, *グループごとのクォータ割り当て*

ソフト制限, ユーザーごとのクォータ割り当て

ハード制限, ユーザーごとのクォータ割り当て

ファイルシステムごとの割り当て, ソフト制限の猶予期間の設定

ユーザーごとに割り当て, ユーザーごとのクォータ割り当て

有効化, ディスククォータの設定, 有効化と無効化
/etc/fstab、変更, クォータの有効化

quotacheck、実行中, クォータデータベースファイルの作成

クォータファイルの作成, クォータデータベースファイルの作成

無効化, 有効化と無効化

猶予期間, ユーザーごとのクォータ割り当て

管理, ディスククォータの管理
quotacheck コマンド、以下のチェックに使用, クォータの精度維持

レポート, ディスククォータに関するレポート

関連情報, ディスククオータのリファレンス

ディスクストレージ (参照 ディスククォータ)
parted (参照 parted)

ディスクレスクライアントの tftp サービスの設定
ディスクレスシステム, ディスクレスクライアントの tftp サービスの設定

ディスクレスクライアント用の DHCP の設定
ディスクレスシステム, ディスクレスクライアント用の DHCP の設定

ディスクレスシステム
DHCP、設定, ディスクレスクライアント用の DHCP の設定

tftp サービス、設定, ディスクレスクライアントの tftp サービスの設定

エクスポートしたファイルシステム, ディスクレスクライアントのエクスポートしたファイルシステムの設定

ネットワークブートサービス, リモートディスクレスシステムの設定

リモートのディスクレスシステム, リモートディスクレスシステムの設定

必要なパッケージ, リモートディスクレスシステムの設定

ディレクトリー
/boot/, /boot/ ディレクトリー

/dev/, /dev/ ディレクトリー

/etc/, /etc/ ディレクトリー

/mnt/, /mnt/ ディレクトリー

/opt/, /opt/ ディレクトリー

/proc/, /proc/ ディレクトリー

/srv/, /srv/ ディレクトリー

/sys/, /sys/ ディレクトリー

/usr/, /usr/ ディレクトリー

/var/, /var/ ディレクトリー

デバイス、削除, ストレージデバイスの削除

デバイスがブロックされているかどうかの確認

ファイバーチャネル

リンクロス動作の変更, ファイバーチャネル

デバイスのステータス

Linux SCSI レイヤー, デバイスの状態

デバイスの削除, ストレージデバイスの削除

デプロイメントのガイドライン

ソリッドステートディスク, ソリッドステートディスクのデプロイメントガイドライン

トラブルシューティング

オンラインストレージ, オンラインストレージ設定のトラブルシューティング

ドライバー (ネイティブ)、ファイバーチャネル, ネイティブファイバーチャネルのドライバーおよび機能

ネイティブのファイバーチャネルドライバー, ネイティブファイバーチャネルのドライバーおよび機能

ネットワークファイルシステム (参照 NFS)

ネットワークブートサービス

ディスクレスシステム, リモートディスクレスシステムの設定

ハイエンドアレイ

書き込みバリア, ハイエンドアレイ

ハードウェア RAID (参照 RAID)

ハードウェア RAID コントローラードライバー

RAID, Linux ハードウェア RAID のコントローラードライバー

バインド不可能なマウント, マウントの共有

バックアップ/復元

XFS, XFS ファイルシステムのバックアップおよび復元

バックアップの復元
XFS, 復元

バッテリーでバックアップされる書き込みキャッシュ
書き込みバリア, バッテリーでバックアップされる書き込みキャッシュ

パフォーマンスに関する保証
FS-Cache, パフォーマンスに関する保証

パラレル NFS
pNFS, pNFS

パリティー
RAID, RAID レベルとリニアサポート

パーティション
サイズ変更, fdisk でのパーティションのサイズ変更
フォーマット
mkfs , パーティションのフォーマットとラベル付け
リスト表示, パーティションテーブルの表示
作成, パーティションの作成
削除中, パーティションの削除

パーティションテーブル
表示する, パーティションテーブルの表示

ファイバーチャネル
オンラインストレージ, ファイバーチャネル

ファイバーチャネル API, ファイバーチャネル API

ファイバーチャネルドライバー (ネイティブ), ネイティブファイバーチャネルのドライバーおよび機能

ファイルシステム, ファイルシステム情報の収集
Btrfs, Btrfs (テクノロジープレビュー)
ext2 (参照 ext2)
ext3 (参照 ext3)
FHS 標準, FHS 組織
構造, ファイルシステム構造とメンテナンス

組織, FHS 組織

階層, ファイルシステム階層標準 (FHS) の概要

ファイルシステムのタイプ

ext4, ext4 ファイルシステム

GFS2, Global File System 2

XFS, XFS ファイルシステム

ファイルシステムの修復

XFS, XFS ファイルシステムの修復

ファイルシステムサイズの拡大

XFS, XFS ファイルシステムのサイズの拡大

ブロックされたデバイスの検証

ファイバーチャネル

リンクロス動作の変更, ファイバーチャネル

ブロックデバイス ioctls (ユーザー空間アクセス)

I/O のアライメントとサイズ, ブロックデバイス ioctls

プライベートマウント, マウントの共有

プロジェクト制限 (設定)

XFS, プロジェクト制限の設定

ホスト

ファイバーチャネル API, ファイバーチャネル API

ポートの状態 (リモート)、判断

ファイバーチャネル

リンクロス動作の変更, ファイバーチャネル

マウント (クライアント設定)

NFS, NFS クライアントの設定

マウントする, ファイルシステムのマウント

ext4, ext4 ファイルシステムのマウント

XFS, XFS ファイルシステムのマウント

マウントポイントの移動, マウントポイントの移動

ミラーリング
RAID, RAID レベルとリニアサポート

ユーザー空間の API ファイル
ファイバーチャネル API, ファイバーチャネル API

ユーザー空間アクセス
I/O のアライメントとサイズ, ユーザー空間アクセス

リニア RAID
RAID, RAID レベルとリニアサポート

リモートのディスクレスシステム
ディスクレスシステム, リモートディスクレスシステムの設定

リモートポート
ファイバーチャネル API, ファイバーチャネル API

リモートポートの状態、判断
ファイバーチャネル
リンクロス動作の変更, ファイバーチャネル

リモートポートの状態の判断
ファイバーチャネル
リンクロス動作の変更, ファイバーチャネル

リンクロス動作の変更, リンクロス動作の変更
ファイバーチャネル, ファイバーチャネル

レイジーマウント/アンマウントのサポート (autofs バージョン 5)
NFS, バージョン 4 と比較した autofs バージョン 5 の改善点

レコードタイプ
discovery
iSCSI, iSCSI 検出設定

レポート (xfs_quota エキスパートモード)
XFS, XFS クォータ管理

ログイン
iSCSI ターゲット, iSCSI ターゲットへのログイン

一時停止中
XFS, XFS ファイルシステムの一時停止

主な特長
ext4, ext4 ファイルシステム
XFS, XFS ファイルシステム

他のファイルシステムユーティリティー
ext4, ext4 ファイルシステムのその他のユーティリティー

仮想ファイルシステム (/proc)
/proc/devices, /proc 仮想ファイルシステム
/proc/filesystems, /proc 仮想ファイルシステム
/proc/mdstat, /proc 仮想ファイルシステム
/proc/mounts, /proc 仮想ファイルシステム
/proc/mounts/, /proc 仮想ファイルシステム
/proc/partitions, /proc 仮想ファイルシステム

仮想マシン用のストレージ, 仮想マシン用のストレージ

作成
ext4, Ext4 ファイルシステムの作成
XFS, XFS ファイルシステムの作成

保存用の LDAP を使用した自動マウント機能マップの保存 (autofs)
NFS, サイト設定ファイルの上書きまたは拡張

個々のユーザー
volume_key, volume_key を個別のユーザーとして使用する

共有サブツリー, マウントの共有
スレーブマウント, マウントの共有
バインド不可能なマウント, マウントの共有
プライベートマウント, マウントの共有
共有マウント, マウントの共有

共有マウント, マウントの共有

利用可能な iface 設定の表示
オフロードとインターフェイスのバインディング
iSCSI, 利用可能な iface 設定の表示

制限 (xfs_quota エキスパートモード)
XFS, XFS クォータ管理

割り当て機能
ext4, ext4 ファイルシステム
XFS, XFS ファイルシステム

古い BIOS RAID メタデータ
ストレージをインストールする際の注意点, 古い BIOS RAID メタデータ

実行中のステータス
Linux SCSI レイヤー, SCSI コマンドタイマーおよびデバイスステータスの制御

強化された LDAP サポート (autofs バージョン 5)
NFS, バージョン 4 と比較した autofs バージョン 5 の改善点

必要なパッケージ
FCoE, ファイバーチャネルオーバーイーサネットインターフェイスの設定
ディスクレスシステム, リモートディスクレスシステムの設定
追加/削除
LUN (論理ユニット番号), rescan-scsi-bus.sh を使用した論理ユニットの追加/削除

新着情報
ストレージをインストールする際の注意点, ストレージをインストールする際の注意点

既知の問題
追加/削除
LUN (論理ユニット番号), rescan-scsi-bus.sh の既知の問題

書き込みキャッシュ、無効化
書き込みバリア, 書き込みキャッシュの無効化

書き込みキャッシュの無効化
書き込みバリア, 書き込みキャッシュの無効化

書き込みバリア
ext4, ext4 ファイルシステムのマウント
NFS, NFS
XFS, 書き込みバリア
エラーメッセージ, 書き込みバリアの有効化と無効化

ハイエンドアレイ, ハイエンドアレイ

バッテリーでバックアップされる書き込みキャッシュ, バッテリーでバックアップされる書き込みキャッシュ

定義, 書き込みバリア

書き込みキャッシュの無効化, 書き込みキャッシュの無効化

書き込みバリアのしくみ, 書き込みバリアのしくみ

書き込みバリアの重要性, 書き込みバリアの重要性

有効化/無効化, 書き込みバリアの有効化と無効化

書き込みバリアのしくみ
書き込みバリア, 書き込みバリアのしくみ

書き込みバリアの重要性
書き込みバリア, 書き込みバリアの重要性

最大サイズ
GFS2, Global File System 2

最大サイズ, GFS2 ファイルシステム, Global File System 2

有効化/無効化
書き込みバリア, 書き込みバリアの有効化と無効化

概要, 概要
オンラインストレージ, オンラインストレージ管理

永続的な命名, 永続的な命名

特定のセッションのタイムアウト、設定
iSCSI の設定, 特定セッションのタイムアウトの設定

特定セッションのタイムアウト、設定
iSCSI の設定, 特定セッションのタイムアウトの設定

相互接続 (スキャン)
iSCSI, iSCSI 相互接続のスキャン

相互接続のスキャン
iSCSI, iSCSI 相互接続のスキャン

統計情報 (追跡)
FS-Cache, 統計情報

統計情報の追跡
FS-Cache, 統計情報

読み取り/書き込み状態の変更
オンライン論理ユニット, オンライン論理ユニットの読み取り/書き込み状態の変更

追加/削除
LUN (論理ユニット番号), rescan-scsi-bus.sh を使用した論理ユニットの追加/削除

適切な nsswitch 設定 (autofs バージョン 5) の使用
NFS, バージョン 4 と比較した autofs バージョン 5 の改善点

階層、 ファイルシステム, ファイルシステム階層標準 (FHS) の概要

高度な RAID デバイスの作成
RAID, 高度な RAID デバイスの作成

A

Anaconda のサポート
RAID, Anaconda インストーラーでの RAID のサポート

API、 iSCSI, iSCSI API

API、 ファイバーチャネル, ファイバーチャネル API

ATA 規格
I/O のアライメントとサイズ, ATA

autofs , autofs, autofs の設定
(参照 NFS)

autofs バージョン 5
NFS, バージョン 4 と比較した autofs バージョン 5 の改善点

autofs マウントポイントごとの複数のマスターマップエントリー (autofs バージョン 5)
NFS, バージョン 4 と比較した autofs バージョン 5 の改善点

B

bcull (キャッシュカリング制限の設定)
FS-Cache, キャッシュカリング制限の設定

brun (キャッシュカリング制限の設定)
FS-Cache, キャッシュカリング制限の設定

bstop (キャッシュカリング制限の設定)
FS-Cache, キャッシュカリング制限の設定

Btrfs
ファイルシステム, Btrfs (テクノロジープレビュー)

C

cachefiles
FS-Cache, FS-Cache

cachefilesd
FS-Cache, キャッシュの設定

CCW、チャネルコマンドワード
ストレージをインストールする際の注意点, IBM System Z における DASD デバイスと zFCP デバイス

changing dev_loss_tmo
ファイバーチャネル
リンクロス動作の変更, ファイバーチャネル

commands
volume_key, volume_key コマンド

configuration
discovery
iSCSI, iSCSI 検出設定

cumulative モード (xfsrestore)
XFS, 復元

D

debugfs (ext4 ファイルシステムのその他のユーティリティー)
ext4, ext4 ファイルシステムのその他のユーティリティー

deployment
ソリッドステートディスク, ソリッドステートディスクのデプロイメントガイドライン

dev ディレクトリー, /dev/ ディレクトリー

Device Mapper を使用したマルチパス設定, DM Multipath

dev_loss_tmo
ファイバーチャネル
リンクロス動作の変更, ファイバーチャネル

dev_loss_tmo, changing
ファイバーチャネル
リンクロス動作の変更, ファイバーチャネル

df, df コマンド

DHCP、設定
ディスクレスシステム, ディスクレスクライアント用の DHCP の設定

DIF/DIX を有効にしているブブロックデバイス
ストレージをインストールする際の注意点, DIF/DIX を有効にしているブロックデバイス

discovery
iSCSI, iSCSI 検出設定

dm-multipath
iSCSI の設定, dm-multipathでの iSCSI 設定

dmraid
RAID, dmraid

dmraid (RAID セットを設定する)
RAID, dmraid

du, du コマンド

E

e2fsck, Ext2 ファイルシステムへの復元

e2image (ext4 ファイルシステムのその他のユーティリティー)
ext4, ext4 ファイルシステムのその他のユーティリティー

e2label
ext4, ext4 ファイルシステムのその他のユーティリティー

e2label (ext4 ファイルシステムのその他のユーティリティー)
ext4, ext4 ファイルシステムのその他のユーティリティー

etc ディレクトリー, /etc/ ディレクトリー

ext2

ext3 からの復元, Ext2 ファイルシステムへの復元

ext3

ext2 からの変換, ext3 ファイルシステムへの変換

features, Ext3 ファイルシステム。

作成, Ext3 ファイルシステムの作成

ext4

debugfs (ext4 ファイルシステムのその他のユーティリティー), ext4 ファイルシステムのその他のユーティリティー

e2image (ext4 ファイルシステムのその他のユーティリティー), ext4 ファイルシステムのその他のユーティリティー

e2label, ext4 ファイルシステムのその他のユーティリティー

e2label (ext4 ファイルシステムのその他のユーティリティー), ext4 ファイルシステムのその他のユーティリティー

fsync(), ext4 ファイルシステム

mkfs.ext4, Ext4 ファイルシステムの作成

nobarrier マウントオプション, ext4 ファイルシステムのマウント

quota (ext4 ファイルシステムのその他のユーティリティー), ext4 ファイルシステムのその他のユーティリティー

resize2fs (resizing ext4), ext4 ファイルシステムのサイズ変更

stripe-width (ストライプ配列を指定), Ext4 ファイルシステムの作成

tune2fs (mounting), ext4 ファイルシステムのマウント

サイズ変更, ext4 ファイルシステムのサイズ変更

ストライド (ストライプジオメトリーの指定), Ext4 ファイルシステムの作成

ストライプジオメトリー, Ext4 ファイルシステムの作成

ファイルシステムのタイプ, ext4 ファイルシステム

マウントする, ext4 ファイルシステムのマウント

主な特長, ext4 ファイルシステム

他のファイルシステムユーティリティー, ext4 ファイルシステムのその他のユーティリティー

作成, Ext4 ファイルシステムの作成

割り当て機能, ext4 ファイルシステム

書き込みバリア, ext4 ファイルシステムのマウント

F

FCoE

FCoE を使用するためにイーサネットインターフェイスの設定, ファイバーチャネルオーバーイーサネットインターフェイスの設定

Fibre Channel over Ethernet, ファイバーチャネルオーバーイーサネットインターフェイスの設定

必要なパッケージ, ファイバーチャネルオーバーイーサネットインターフェイスの設定

FCoE を使用するためにイーサネットインターフェイスの設定

FCoE, ファイバーチャネルオーバーイーサネットインターフェイスの設定

FHS, ファイルシステム階層標準 (FHS) の概要, FHS 組織

(参照 ファイルシステム)

Fibre Channel over Ethernet

FCoE, ファイバーチャネルオーバーイーサネットインターフェイスの設定

findmnt （コマンド）

マウントのリスト表示, 現在マウントされているファイルシステムのリスト表示

FS-Cache

bcull (キャッシュカリング制限の設定), キャッシュカリング制限の設定

brun (キャッシュカリング制限の設定), キャッシュカリング制限の設定

bstop (キャッシュカリング制限の設定), キャッシュカリング制限の設定

cachefiles, FS-Cache

cachefilesd, キャッシュの設定

NFS (with の使用), NFS でのキャッシュの使用

NFS (キャッシュの制限), NFS でのキャッシュの制限

tune2fs (キャッシュの設定), キャッシュの設定

インデックスキー, FS-Cache

キャッシュの共有, キャッシュの共有

キャッシュの設定, キャッシュの設定

キャッシュカリング制限, キャッシュカリング制限の設定

キャッシュバックエンド, FS-Cache

コヒーレンシーデータ, FS-Cache

パフォーマンスに関する保証, パフォーマンスに関する保証

統計情報 (追跡), 統計情報

fsync()

ext4, ext4 ファイルシステム

XFS, XFS ファイルシステム

G

GFS2

gfs2.ko, Global File System 2

ファイルシステムのタイプ, Global File System 2

最大サイズ, Global File System 2

GFS2 ファイルシステムの最大サイズ, Global File System 2

gfs2.ko

GFS2, Global File System 2

Global File System 2

gfs2.ko, Global File System 2

ファイルシステムのタイプ, Global File System 2

最大サイズ, Global File System 2

gquota/gqnoenforce

XFS, XFS クォータ管理

I

I/O のアライメントとサイズ, ストレージ I/O アライメントとサイズ

ATA 規格, ATA

I/O パラメーターのスタッキング, I/O パラメーターのスタッキング

Linux I/O スタック, ストレージ I/O アライメントとサイズ

logical_block_size, ユーザー空間アクセス

LVM, 論理ボリュームマネージャー

READ CAPACITY(16), SCSI

SCSI 規格, SCSI

sysfs インターフェイス (ユーザー空間アクセス), sysfs インターフェイス

ストレージアクセスパラメーター, ストレージアクセスパラメーター

ツール (パーティションおよびその他のファイルシステム機能用), パーティションおよびファイルシステムツール

ブロックデバイス ioctls (ユーザー空間アクセス), ブロックデバイス ioctls

ユーザー空間アクセス, ユーザー空間アクセス

I/O パラメーターのスタッキング

I/O のアライメントとサイズ, I/O パラメーターのスタッキング

IBM System Z における DASD デバイスと zFCP デバイス
ストレージをインストールする際の注意点, IBM System Z における DASD デバイスと zFCP デバイス

iface (iSCSI オフロードの設定)
オフロードとインターフェイスのバインディング
iSCSI, iSCSI オフロード用の iface の設定

iface のバインディング/バインド解除
オフロードとインターフェイスのバインディング
iSCSI, iface のポータルへのバインド/バインド解除

iface のポータルへのバインド/バインド解除
オフロードとインターフェイスのバインディング
iSCSI, iface のポータルへのバインド/バインド解除

iface 設定
オフロードとインターフェイスのバインディング
iSCSI, 利用可能な iface 設定の表示

iface 設定、表示
オフロードとインターフェイスのバインディング
iSCSI, 利用可能な iface 設定の表示

iSCSI
discovery, iSCSI 検出設定
configuration, iSCSI 検出設定
レコードタイプ, iSCSI 検出設定

オフロードとインターフェイスのバインディング, iSCSI オフロードおよびインターフェイスバインディングの設定
iface (iSCSI オフロードの設定), iSCSI オフロード用の iface の設定
iface のポータルへのバインド/バインド解除, iface のポータルへのバインド/バインド解除
iface 設定, 利用可能な iface 設定の表示
iface 設定、表示, 利用可能な iface 設定の表示
イニシエーターの実装, 利用可能な iface 設定の表示
ソフトウェア iSCSI, ソフトウェア iSCSI 用の iface の設定

ソフトウェア iSCSI の iface, ソフトウェア iSCSI 用の iface の設定
利用可能な iface 設定の表示, 利用可能な iface 設定の表示

ソフトウェア iSCSI, ソフトウェア iSCSI 用の iface の設定
ターゲット, iSCSI ターゲットへのログイン
ログイン, iSCSI ターゲットへのログイン

相互接続のスキャン, iSCSI 相互接続のスキャン

iSCSI API, iSCSI API
iSCSI の検出および設定
ストレージをインストールする際の注意点, iSCSI の検出および設定

iSCSI ルート
iSCSI の設定, iSCSI ルート

iSCSI 論理ユニット、サイズ変更, iSCSI 論理ユニットのサイズを変更
iSCSI 論理ユニットのサイズ変更, iSCSI 論理ユニットのサイズを変更

L

levels
RAID, RAID レベルとリニアサポート

Linux I/O スタック
I/O のアライメントとサイズ, ストレージ I/O アライメントとサイズ

logical_block_size
I/O のアライメントとサイズ, ユーザー空間アクセス

LUKS/dm-crypt の使用、ブロックデバイスの暗号化
ストレージをインストールする際の注意点, LUKS を使用したブロックデバイスの暗号化

LUN (論理ユニット番号)
追加/削除, rescan-scsi-bus.sh を使用した論理ユニットの追加/削除
rescan-scsi-bus.sh, rescan-scsi-bus.sh を使用した論理ユニットの追加/削除
必要なパッケージ, rescan-scsi-bus.sh を使用した論理ユニットの追加/削除
既知の問題, rescan-scsi-bus.sh の既知の問題

LVM
I/O のアライメントとサイズ, 論理ボリュームマネージャー

M

mdadm (RAID セットの設定)
RAID, mdadm

mdraid
RAID, mdraid

mkfs , パーティションのフォーマットとラベル付け

mkfs.ext4
ext4, Ext4 ファイルシステムの作成

mkfs.xfs
XFS, XFS ファイルシステムの作成

mnt ディレクトリー, /mnt/ ディレクトリー

mount (command), mount コマンドの使用
オプション, マウントオプションの指定
ファイルシステムのマウント, ファイルシステムのマウント
マウントのリスト表示, 現在マウントされているファイルシステムのリスト表示
マウントポイントの移動, マウントポイントの移動
共有サブツリー, マウントの共有
スレーブマウント, マウントの共有
バインド不可能なマウント, マウントの共有
プライベートマウント, マウントの共有
共有マウント, マウントの共有

N

NFS
/etc/fstab , /etc/fstabを使用した NFS ファイルシステムのマウント
/local/directory (クライアント設定、 マウント), NFS クライアントの設定
/remote/export (クライアント設定、 マウント), NFS クライアントの設定
autofs
LDAP, LDAP を使用した自動マウント機能マップの格納
拡張, サイト設定ファイルの上書きまたは拡張
設定, autofs の設定
autofs バージョン 5, バージョン 4 と比較した autofs バージョン 5 の改善点

autofs マウントポイントごとの複数のマスターマップエントリー (autofs バージョン 5), バージョン 4 と比較した autofs バージョン 5 の改善点

client

autofs , autofs

マウントオプション, 一般的な NFS マウントオプション

設定, NFS クライアントの設定

condrestart, NFS サーバーの起動と停止

FS-Cache, NFS でのキャッシュの使用

NFS と rpcbind のトラブルシューティング, NFS と rpcbindのトラブルシューティング

RDMA, NFS over RDMA の有効化 (NFSoRDMA)

rfc2307bis (autofs), LDAP を使用した自動マウント機能マップの格納

rpcbind , NFS および rpcbind

security, NFS のセキュア化

NFSv3 ホストアクセス, AUTH_SYS とエクスポート制御による NFS セキュリティー保護

status, NFS サーバーの起動と停止

TCP, NFS の概要

UDP, NFS の概要

オプション (クライアント設定、 マウント), NFS クライアントの設定

サイト設定ファイルの上書き/拡張 (autofs), autofs の設定

サーバー (クライアント設定、 マウント), NFS クライアントの設定

サーバー設定, NFS サーバーの設定

/etc/exports , /etc/exports 設定ファイル

exportfs コマンド, exportfs コマンド

NFSv4 を使用した exportfs コマンド, NFSv4 での exportfs の使用

セキュリティー

NFSv4 ホストアクセス, AUTH_GSSを使用した NFS 保護

ファイル権限, ファイル権限

ダイレクトマップサポート (autofs バージョン 5), バージョン 4 と比較した autofs バージョン 5 の改善点

ファイアウォールでの設定, ファイアウォール背後での NFS の実行

ホスト名の形式, ホスト名の形式

マウント (クライアント設定), NFS クライアントの設定

レイジーマウント/アンマウントのサポート (autofs バージョン 5), バージョン 4 と比較した autofs バージョン 5 の改善点

仕組み, NFS の概要

保存用の LDAP を使用した自動マウント機能マップの保存 (autofs), サイト設定ファイルの上書きまたは拡張

停止, NFS サーバーの起動と停止

再読み込み, NFS サーバーの起動と停止

再起動, NFS サーバーの起動と停止

強化された LDAP サポート (autofs バージョン 5), バージョン 4 と比較した autofs バージョン 5 の改善点

必要なサービス, 必要なサービス

書き込みバリア, NFS

概要, Network File System (NFS)

起動, NFS サーバーの起動と停止

適切な nsswitch 設定 (autofs バージョン 5) の使用, バージョン 4 と比較した autofs バージョン 5 の改善点

関連情報, NFS のリファレンス

インストールされているドキュメント, インストールされているドキュメント

便利な Web サイト, 便利な Web サイト

関連書籍, 関連書籍

NFS (with の使用)

FS-Cache, NFS でのキャッシュの使用

NFS (キャッシュの制限)

FS-Cache, NFS でのキャッシュの制限

NFS でのキャッシュの制限

FS-Cache, NFS でのキャッシュの制限

NFS と rpcbind のトラブルシューティング

NFS, NFS と rpcbindのトラブルシューティング

nobarrier マウントオプション

ext4, ext4 ファイルシステムのマウント

XFS, 書き込みバリア

NOP-Out 要求

リンクロスの変更

iSCSI の設定, NOP-Out 間隔/タイムアウト

NOP-Outs (無効化)
iSCSI の設定, iSCSI ルート

NOP-Outs の無効化
iSCSI の設定, iSCSI ルート

O

opt ディレクトリー, /opt/ ディレクトリー

P

parted , Partitions
コマンドの表, Partitions
デバイスの選択, パーティションテーブルの表示
パーティションのサイズ変更, fdisk でのパーティションのサイズ変更
パーティションの削除, パーティションの削除
パーティションを作成する, パーティションの作成
パーティションテーブルの表示, パーティションテーブルの表示
概要, Partitions

pNFS
パラレル NFS, pNFS

pquota/pqnoenforce
XFS, XFS クォータ管理

proc ディレクトリー, /proc/ ディレクトリー

Q

queue_if_no_path
iSCSI の設定, dm-multipathでの iSCSI 設定
リンクロスの変更
iSCSI の設定, replacement_timeout

quota (ext4 ファイルシステムのその他のユーティリティー)
ext4, ext4 ファイルシステムのその他のユーティリティー

quotacheck , クォータデータベースファイルの作成

quotacheck コマンド
によるクォータ正確性のチェック, クォータの精度維持

quotaoff , 有効化と無効化

quotaon , 有効化と無効化

R

RAID
Anaconda のサポート, Anaconda インストーラーでの RAID のサポート
dmraid, dmraid
dmraid (RAID セットを設定する), dmraid
levels, RAID レベルとリニアサポート
mdadm (RAID セットの設定), mdadm
mdraid, mdraid
RAID のサブシステム, Linux RAID サブシステム
RAID セットの設定, RAID セットの設定
インストーラーのサポート, Anaconda インストーラーでの RAID のサポート
ストライピング, RAID レベルとリニアサポート
ソフトウェア RAID, RAID のタイプ
ハードウェア RAID, RAID のタイプ
ハードウェア RAID コントローラードライバー, Linux ハードウェア RAID のコントローラードライバー
パリティー, RAID レベルとリニアサポート
ミラーリング, RAID レベルとリニアサポート
リニア RAID, RAID レベルとリニアサポート
レベル 0, RAID レベルとリニアサポート
レベル 1, RAID レベルとリニアサポート
レベル 4, RAID レベルとリニアサポート
レベル 5, RAID レベルとリニアサポート
使用理由, RAID (Redundant Array of Independent Disks)
高度な RAID デバイスの作成, 高度な RAID デバイスの作成

RAID のサブシステム
RAID, Linux RAID サブシステム

RAID セットの設定
RAID, RAID セットの設定

RDMA
NFS, NFS over RDMA の有効化 (NFSoRDMA)

READ CAPACITY(16)
I/O のアライメントとサイズ, SCSI

Red Hat Enterprise Linux 固有のファイルの場所
/etc/sysconfig/, 特別な Red Hat Enterprise Linux ファイルの場所
(参照 sysconfig ディレクトリー)

/var/cache/yum, 特別な Red Hat Enterprise Linux ファイルの場所

/var/lib/rpm/, 特別な Red Hat Enterprise Linux ファイルの場所

replacement_timeout
リンクロスの変更
iSCSI の設定, SCSI エラーハンドラー, replacement_timeout

replacement_timeoutM
iSCSI の設定, iSCSI ルート

rescan-scsi-bus.sh
追加/削除
LUN (論理ユニット番号), rescan-scsi-bus.sh を使用した論理ユニットの追加/削除

resize2fs, Ext2 ファイルシステムへの復元

resize2fs (resizing ext4)
ext4, ext4 ファイルシステムのサイズ変更

rfc2307bis (autofs)
NFS, LDAP を使用した自動マウント機能マップの格納

rpcbind , NFS および rpcbind
(参照 NFS)

NFS, NFS と rpcbindのトラブルシューティング

rpcinfo , NFS と rpcbindのトラブルシューティング

status, NFS サーバーの起動と停止

rpcinfo , NFS と rpcbindのトラブルシューティング

S

SCSI エラーハンドラー
リンクロスの変更
iSCSI の設定, SCSI エラーハンドラー

SCSI コマンドタイマー
Linux SCSI レイヤー, コマンドタイマー

SCSI コマンドタイマーとデバイスステータスの制御
Linux SCSI レイヤー, SCSI コマンドタイマーおよびデバイスステータスの制御

SCSI 規格
I/O のアライメントとサイズ, SCSI

simple モード (xfsrestore)
XFS, 復元

SMB (参照 SMB)

srv ディレクトリー, /srv/ ディレクトリー

SSD
ソリッドステートディスク, ソリッドステートディスクのデプロイメントガイドライン

SSM
System Storage Manager, System Storage Manager (SSM)
list コマンド, 検出されたすべてのデバイスに関する情報の表示
resize コマンド, ボリュームのサイズを大きくする
snapshot コマンド, スナップショット
インストール, SSM のインストール
バックエンド, SSM のバックエンド

stripe-width (ストライプ配列を指定)
ext4, Ext4 ファイルシステムの作成

su (mkfs.xfs サブオプション)
XFS, XFS ファイルシステムの作成

sw (mkfs.xfs サブオプション)
XFS, XFS ファイルシステムの作成

sys ディレクトリー, /sys/ ディレクトリー

sysconfig ディレクトリー, 特別な Red Hat Enterprise Linux ファイルの場所

sysfs
概要
オンラインストレージ, オンラインストレージ管理

sysfs インターフェイス (ユーザー空間アクセス)
I/O のアライメントとサイズ, sysfs インターフェイス

System Storage Manager
SSM, System Storage Manager (SSM)
list コマンド, 検出されたすべてのデバイスに関する情報の表示
resize コマンド, ボリュームのサイズを大きくする
snapshot コマンド, スナップショット
インストール, SSM のインストール
バックエンド, SSM のバックエンド

T

tftp サービス、設定
ディスクレスシステム, ディスクレスクライアントの tftp サービスの設定

transport
ファイバーチャネル API, ファイバーチャネル API

TRIM コマンド
ソリッドステートディスク, ソリッドステートディスクのデプロイメントガイドライン

tune2fs
ext2 への復元, Ext2 ファイルシステムへの復元
ext3 への変換, ext3 ファイルシステムへの変換

tune2fs (mounting)
ext4, ext4 ファイルシステムのマウント

tune2fs (キャッシュの設定)
FS-Cache, キャッシュの設定

U

udev ルール（タイムアウト）

コマンドタイマー (SCSI), コマンドタイマー

umount, ファイルシステムのアンマウント

updates
ストレージをインストールする際の注意点, ストレージをインストールする際の注意点

uquota/uqnoenforce
XFS, XFS クォータ管理

usr ディレクトリー, /usr/ ディレクトリー

V

var ディレクトリー, /var/ ディレクトリー

var/lib/rpm/ ディレクトリー, 特別な Red Hat Enterprise Linux ファイルの場所

var/spool/up2date/ ディレクトリー, 特別な Red Hat Enterprise Linux ファイルの場所

version
新機能
autofs, バージョン 4 と比較した autofs バージョン 5 の改善点

volume_key
commands, volume_key コマンド
個々のユーザー, volume_key を個別のユーザーとして使用する

W

World Wide Identifier (WWID)
永続的な命名, World Wide Identifier (WWID)

WWID
永続的な命名, World Wide Identifier (WWID)

X

XFS
cumulative モード (xfsrestore), 復元
fsync(), XFS ファイルシステム
gquota/gqnoenforce, XFS クォータ管理
mkfs.xfs, XFS ファイルシステムの作成
nobarrier マウントオプション, 書き込みバリア

pquota/pqnoenforce, *XFS クォータ管理*

simple モード (xfsrestore), *復元*

su (mkfs.xfs サブオプション), *XFS ファイルシステムの作成*

sw (mkfs.xfs サブオプション), *XFS ファイルシステムの作成*

uquota/uqnoenforce, *XFS クォータ管理*

xfsdump, *バックアップ*

xfsprogs, *XFS ファイルシステムの一時停止*

xfsrestore, *復元*

xfs_admin, *XFS ファイルシステムのその他のユーティリティー*

xfs_bmap, *XFS ファイルシステムのその他のユーティリティー*

xfs_copy, *XFS ファイルシステムのその他のユーティリティー*

xfs_db, *XFS ファイルシステムのその他のユーティリティー*

xfs_freeze, *XFS ファイルシステムの一時停止*

xfs_fsr, *XFS ファイルシステムのその他のユーティリティー*

xfs_growfs, *XFS ファイルシステムのサイズの拡大*

xfs_info, *XFS ファイルシステムのその他のユーティリティー*

xfs_mdrestore, *XFS ファイルシステムのその他のユーティリティー*

xfs_metadump, *XFS ファイルシステムのその他のユーティリティー*

xfs_quota, *XFS クォータ管理*

xfs_repair, *XFS ファイルシステムの修復*

インタラクティブな操作 (xfsrestore), *復元*

エキスパートモード (xfs_quota), *XFS クォータ管理*

クォータの管理, *XFS クォータ管理*

ダンプレベル, *バックアップ*

ダーティーログを使用した XFS ファイルシステムの修復, *XFS ファイルシステムの修復*

バックアップ/復元, *XFS ファイルシステムのバックアップおよび復元*

ファイルシステムのタイプ, *XFS ファイルシステム*

ファイルシステムの修復, *XFS ファイルシステムの修復*

ファイルシステムサイズの拡大, *XFS ファイルシステムのサイズの拡大*

プロジェクト制限 (設定), *プロジェクト制限の設定*

マウントする, *XFS ファイルシステムのマウント*

レポート (xfs_quota エキスパートモード), *XFS クォータ管理*

一時停止中, *XFS ファイルシステムの一時停止*

主な特長, XFS ファイルシステム
作成, XFS ファイルシステムの作成
制限 (xfs_quota エキスパートモード), XFS クォータ管理
割り当て機能, XFS ファイルシステム
書き込みバリア, 書き込みバリア

xfsdump
XFS, バックアップ

xfsprogs
XFS, XFS ファイルシステムの一時停止

xfsrestore
XFS, 復元

xfs_admin
XFS, XFS ファイルシステムのその他のユーティリティー

xfs_bmap
XFS, XFS ファイルシステムのその他のユーティリティー

xfs_copy
XFS, XFS ファイルシステムのその他のユーティリティー

xfs_db
XFS, XFS ファイルシステムのその他のユーティリティー

xfs_freeze
XFS, XFS ファイルシステムの一時停止

xfs_fsr
XFS, XFS ファイルシステムのその他のユーティリティー

xfs_growfs
XFS, XFS ファイルシステムのサイズの拡大

xfs_info
XFS, XFS ファイルシステムのその他のユーティリティー

xfs_mdrestore
XFS, XFS ファイルシステムのその他のユーティリティー

xfs_metadump

XFS, XFS ファイルシステムのその他のユーティリティー

xfs_quota

XFS, XFS クォータ管理

xfs_repair

XFS, XFS ファイルシステムの修復