SUSE® Linux Enterprise Server

11 SP1 www.novell.com

2010 年3 月15 日 SLES 11 SP1: ストレージ管理ガイド

SLES 11 SP1: ストレージ管理ガイド

保証と著作権

Novell, Inc.
404 Wyman Street, Suite 500
Waltham, MA 02451
U.S.A.
www.novell.com

オンラインマニュアル: 本製品とその他のNovell製品の最新のオンラインマニュアルにアクセスするには、Novell Documentation [http://www.novell.com/documentation]のWebページを参照してください。

Novellの商標

サードパーティ資料

サードパーティのすべての商標および著作権は、それぞれの所有者の権利に属します。

目次

このガイドについて

このガイドでは、SUSE® Linux Enterprise Server 11 Support Pack 1 (SP1)サーバでストレージデバイスを管理する方法について説明します。

対象読者

このガイドは、システム管理者を対象としています。

フィードバック

本マニュアルおよびこの製品に含まれているその他のマニュアルについて、皆様のご意見やご要望をお寄せください。 オンラインマニュアルの各ページの下部にあるユーザコメント機能を使用するか www.novell.com/documentation/feedback.htmlにアクセスしてコメントを記入してください。

マニュアルの更新

『*SUSE Linux Enterprise Server 11 SP1ストレージ管理ガイド*』の最新バージョンについては、「SUSE Linux Enterprise Server 11 SP1用Novell®マニュアルのWebサイト [http://www.novell.com/documentation/sles11]」をご覧ください。

その他のマニュアル

デバイスのパーティショニングと管理については、『*SUSE Linux Enterprise Server 11 SP1 Installation and Administration Guide* [http://www.novell.com/documentation/sles11]』を参照してください。

マニュアルの表記規則

Novellのマニュアルでは、「より大きい」記号(>)を使用して手順内の操作と相互参照パス内の項目の順序を示します。

商標記号(®、TMなど)は、Novellの商標を示します。アスタリスク(**)は、サードパーティの商標を示します。

1 Linuxファイルシステムの概要

SUSE® Linux Enterprise Serverには、いくつもの異なるファイルシステムがあり(Ext3、Ext2、ReiserFS、XFS)、その中から適切なシステムを選択できます。各ファイルシステムには、それぞれ独自の利点と欠点があります。ハイパフォーマンスのクラスタリングシナリオの要件を満たすため、SUSE Linux Enterprise Serverでは、HASI (High-Availability Storage Infrastructure)リリースにOCFS2 (Oracle Cluster File System 2)を組み込んでいます。

1.1 用語集

メタデータ(metadata)
ファイルシステムが内包するデータ構造です。これにより、すべてのオンディスクデータが正しく構成され、アクセス可能にになります。本質的には、「「データに関するデータ」」です。ほとんどすべてのファイルシステムに独自のメタデータ構造があり、それが各ファイルシステムに異なるパフォーマンス特性が存在する理由の 1 つになっています。メタデータが破損しないよう維持するのは、非常に重要なことです。もし破損した場合、

ファイルシステム内にあるすべてのデータがアクセス不能になる可能性があるからです。

inode
: サイズ、リンク数、ファイルの内容を実際に格納しているディスクブロックへのポインタ、作成日時、変更日時、クセス日時など、ファイルに関する各種の情報を含むファイルシステムのデータ構造。

ジャーナル(journal)
: ファイルシステムのジャーナルは、ファイルシステムがそのメタデータ内で行う変更を特定のログに記録するオンディスク構造です。ジャーナル機能は、システム起動時にファイルシステム全体をチェックする長い検索プロセスが不要なため、ファイルシステムの回復時間を大幅に短縮します。ただし、それはジャーナルが再現できる場合に限定されます。

1.2 Linuxの主要なファイルシステム

SUSE Linux Enterprise Serverでは、多様なファイルシステムを選択できます。このセクションでは、それらのファイルシステムの機能および利点の概要を説明します。

ただし、すべてのアプリケーションに最適なファイルシステムは存在しません。各ファイルシステムには特定の利点と欠点があり、それらを考慮する必要があります。最も高度なファイルシステムを選択する場合でも、適切なバックアップ戦略が必要です。

このセクションで使用されるデータの*完全性*およびデータの*一貫性*という用語は、ユーザスペースデータ(ユーザが使用するアプリケーションによりファイルに書き込まれるデータ)の一貫性を指す言葉ではありません。ユーザスペースのデータが一貫しているかどうかは、アプリケーション自体が管理する必要があります。

重要項目

このセクションで特に指定のない限り、パーティションおよびファイルシステムの設定または変更に必要なすべてのステップは、YaSTを使用して実行できます。

• 1.2.1項 「Ext2」 (3 ページ)

- 1.2.2項 「Ext3」 (4 ページ)
- 1.2.3項 「Oracle Cluster File System 2」 (6 ページ)
- 1.2.4項 「ReiserFS」 (7 ページ)
- 1.2.5項 「XFS」 (9 ページ)

1.2.1 Ext2

Ext2の起源は、Linuxの歴史の初期にさかのぼります。 その前身であったExtended File Systemは、1992年4月に実装され、Linux 0.96cに統合されました。Extended File Systemは多くの変更を加えられ、Ext2として数年にわたって、最も人気のあるLinuxファイルシステムになりました。 その後、ジャーナルファイルシステムが作成され、回復時間が非常に短くなったため、Ext2の重要性は低下しました。

Ext2の利点の概要を読むと、Ext2が、かつて幅広く好まれ、そして今でも一部の分野で多くのLinuxユーザから好まれるLinuxファイルシステムである理由がわかります。

- 「堅実性と速度」 (3 ページ)
- 「容易なアップグレード性」 (4 ページ)

堅実性と速度

「古くからある標準」として、Ext2は過去に多くの改良がなされ、繰り返しテストされてきました。 これが、Ext2がしばしば非常に堅実と評される理由かもしれません。ファイルシステムが正常にアンマウントできず、システムが機能停止した場合、e2fsckはファイルシステムのデータの分析を開始します。 メタデータは一貫した状態に戻り、保留されていたファイルとデータブロックは、指定のディレクトリ(`lost+found`)に書き込まれます。 ジャーナルファイルシステムとは対照的に、e2fsckは、最近変更されたわずかなメタデータだけではなく、ファイルシステム全体を分析します。 この結果、ジャーナルファイルシステムがログデータだけをチェックするのに比べて、かなり長い時間を要します。 ファイルシステムのサイズにもよりますが、この手順は30分またはそれ以上を要することがあります。 したがって、高可用性を必要とするどのようなサーバでも、Ext2を選択することは望ましくありません。

ただし、Ext2はジャーナルを維持せず、非常にわずかなメモリを使用するだけなので、時には他のファイルシステムより高速なことがあります。

容易なアップグレード性

Ext3は、Ext2のコードをベースとし、Ext2のオンディスクフォーマットとメタデータフォーマットも共用するので、Ext2からExt3へのアップグレードは非常に容易です。

1.2.2 Ext3

Ext3は、Stephen Tweedieによって設計されました。 他のすべての次世代ファイルシステムとは異なり、Ext3は完全に新しい設計理念に基づいているわけではありません。 Ext3は、Ext2をベースとしています。 これら2つのファイルシステムは、互いに非常に似通っています。 Ext3ファイルシステムを、Ext2ファイルシステムの上に構築することも容易です。 Ext2とExt3の間にある最も重要な違いは、Ext3がジャーナルをサポートしていることです。 要約すると、Ext3には、次の3つの主要な利点があります。

- 「Ext2からの容易で信頼性の高いアップグレード」 (4 ページ)
- 「信頼性とパフォーマンス」 (5 ページ)
- 「Ext2ファイルシステムからExt3への変換」 (5 ページ)

Ext2からの容易で信頼性の高いアップグレード

Ext2のコードは、Ext3が次世代ファイルシステムであることを明確に主張するための強力な土台になりました。 Ext3では、Ext2の信頼性および堅実性がExt3で採用されたジャーナルファイルシステムの利点とうまく統合されています。 ReiserFSまたはXFSのような他のファイルシステムへの移行はかなり手間がかかります(ファイルシステム全体のバックアップを作成し、移行先ファイルシステムを新規に作成する必要があります)が、それとは異なり、Ext3への移行は数分で完了します。 ファイルシステム全体を新たに作成しなおしても、それが完璧に動作するとは限らないので、Ext3への移行は非常に安全でもあります。ジャーナルファイルシステムへのアップグレードを必要とする既存のExt2システムの数を考慮に入れると、多くのシステム管理者にとってExt3が重要な選択肢となり得る理由が容易に分かります。Ext3からExt2へのダウング

レードも、アップグレードと同じほど容易です。 Ext3ファイルシステムのアンマウントを正常に行い、Ext2ファイルシステムとして再マウントするだけです。

信頼性とパフォーマンス

他のジャーナルファイルシステムは、「メタデータのみ」のジャーナルアプローチに従っています。 つまり、メタデータは常に一貫した状態に保持されますが、ファイルシステムのデータ自体については、一貫性が自動的に保証されるわけではありません。Ext3は、メタデータとデータの両方に注意するよう設計されています。 「注意」の度合いはカスタマイズできます。 Ext3の`data=journal`モードを有効にした場合、最大の保護(データの完全性)を実現しますが、メタデータとデータの両方がジャーナル化されるので、システムの動作が遅くなります。比較的新しいアプローチは、`data=ordered`モードを使用することです。これは、データとメタデータ両方の完全性を保証しますが、ジャーナルを適用するのはメタデータのみです。 ファイルシステムドライバは、1つのメタデータの更新に対応するすべてのデータブロックを収集します。 これらのブロックは、メタデータの更新前にディスクに書き込まれます。その結果、パフォーマンスを犠牲にすることなく、メタデータとデータの両方に関する一貫性を達成できます。3番目のオプションは、`data=writeback`を使用することです。これは、対応するメタデータをジャーナルにコミットした後で、データをメインファイルシステムに書き込むことを可能にします。 多くの場合、このオプションは、パフォーマンスの点で最善と考えられています。 しかし、内部のファイルシステムの完全性が維持される一方で、クラッシュと回復を実施した後では、古いデータがファイル内に再登場することを許してしまう可能性があります。 Ext3では、デフォルトとして、`data=ordered`オプションを使用します。

Ext2ファイルシステムからExt3への変換

Ext2ファイルシステムをExt3に変換するには、次の手順に従います。

1. Ext3ジャーナルの作成には、`tune2fs -j`を`root`ユーザとして実行します。

   この結果、デフォルトのパラメータを使用してExt3ジャーナルが作成されます。

ジャーナルのサイズおよびジャーナルを常駐させるデバイスを指定するには、tune2fs -Jとともに適切なジャーナルオプションsize=およびdevice=を指定して、実行します。tune2fsプログラムの詳細については、tune2fsのマニュアルページを参照してください。

2 ファイル/etc/fstabをrootユーザとして編集して、該当するパーティションに指定されているファイルシステムタイプをext2からext3に変更し、その変更内容を保存します。

これにより、Ext3ファイルシステムが認識されるようになります。この変更結果は、次回の再起動後に有効になります。

3 Ext3パーティションとしてセットアップされたルートファイルシステムをブートするには、ext3とjbdの各モジュールをinitrdに組み込みます。

3a /etc/sysconfig/kernelをrootとして編集し、ext3およびjbdをINITRD_MODULES変数に追加し、最後に変更内容を保存します。

3b mkinitrdコマンドを実行します。

これにより新規のinitrdがビルドされ、すぐに使用できます。

4 システムを再起動します。

1.2.3 Oracle Cluster File System 2

OCFS2は、クラスタリング設定用に作成されたジャーナルファイルシステムです。Ext3などの標準の単一ノードファイルシステムとは対照的に、OCFS2では複数ノードを管理することができます。OCFS2では、1つのファイルシステムを共有ストレージ全体に展開させることができます(SANやマルチパスセットアップなど）。

OCFS2セットアップの各ノードは、すべてのデータに対して同時に読み込み/書き込みアクセスを行うことができます。そのためには、OCFS2がクラスタに対応している必要があります。つまり、OCFS2は、どのノードがクラスタを構成しているか、それらのノードが実際に利用可能かどうかを判別する機能を含む必要があります。クラスタのメンバーシップを計算するため、OCFS2

にはノードマネージャが組み込まれています。クラスタ内のノードの可用性を監視するために、OCFS2には簡単なハートビート機能が実装されています。多様なノードがファイルシステムに直接アクセスすることによって発生する問題を回避するため、OCFS2には分散ロックマネージャも組み込まれています。ノード間の通信は、TCPベースのメッセージングシステムにより処理されます。

OCFS2の主要機能と利点を次に示します。

- メタデータのキャッシングとジャーナリング
- データベースのパフォーマンスを向上する非同期、直接I/Oのサポート
- 最大16TBまでのボリュームで、最高4KBまでの複数ブロックサイズをサポート(各ボリュームで異なるブロックサイズを使用可能)
- ノード間にまたがるファイルデータの整合性
- 255台までのクラスタノードをサポート

OCFS2の詳細については、『*High Availability Storage Infrastructure Administration Guide*』を参照してください。

1.2.4 ReiserFS

2.4カーネルリリースから公式に主要機能として採用されたReiserFSは、SUSE 6.4以降、2.2.x SUSEカーネルのカーネルパッチとして利用可能となりました。ReiserFSは、Hans ReiserとNamesys開発チームにより設計されました。ReiserFSは、Ext2に代わる強力な選択肢であることを実証してきました。 ReiserFSの主要な利点としては、効率的なディスクスペース使用率、より良いディスクアクセスパフォーマンス、より高速なクラッシュリカバリ、およびデータジャーナリングの使用による信頼性の向上があります。

- 「より良いディスクスペース使用効率」 (8 ページ)
- 「より良いディスクアクセスパフォーマンス」 (8 ページ)
- 「高速なクラッシュ回復機能」 (8 ページ)
- 「データジャーナリングによる信頼性」 (8 ページ)

より良いディスクスペース使用効率

ReiserFSでは、すべてのデータは、B*-Tree(バランスドツリー)と呼ばれる構造で編成されています。このツリー構造は、より良いディスクスペース使用効率に貢献しています。小さなファイルは、B*-Treeのリーフノードに直接格納されるからです。そのようなファイルをどこか他の場所に格納して、ディスク上の実際の場所を指すポインタを維持するより優れています。 それに加えて、ストレージ(記憶領域)は 1KBまたは 4KBのチャンク単位で割り当てられるのではなく、実際に必要なサイズの構成部分(エクステント)を割り当てられます。もう1つの利点は、inodeの動的割り当てに関係しています。 これは、ファイルシステムの作成時にinodeの密度を指定する必要のある、Ext2のような従来のファイルシステムに比べて、ファイルシステムの柔軟性を高めます。

より良いディスクアクセスパフォーマンス

小規模なファイルでは、多くの場合、ファイルのデータと「stat_data」 (inode)情報が互いに隣り合って保存されます。 これらは1回のディスクI/O操作で読み取れるので、ただ1回のディスクアクセスで、必要な情報すべてを取得できることを意味します。

高速なクラッシュ回復機能

ジャーナルを使用して、メタデータに加えられた最新の変更結果を記録しているので、ファイルシステムが大規模な場合を含め、ファイルシステムを数秒でチェックできます。

データジャーナリングによる信頼性

ReiserFSは、「1.2.2項 「Ext3」 (4 ページ)」に概略されているコンセプトに類似のデータジャーナリングおよびオーダードモードもサポートしています。デフォルトのモードは、`data=ordered`です。このモードでは、データとメタデータの完全性は保証されますが、メタデータのジャーナリングだけが行われます。

1.2.5 XFS

本来は、IRIX OS用のファイルシステムを意図してSGIがXFSの開発を開始したのは、1990年代初期です。 XFSの開発動機は、ハイパフォーマンスの64ビットジャーナルファイルシステムの作成により、非常に厳しいコンピューティングの課題に対応することでした。XFSは大規模なファイルを操作する点で非常に優れていて、ハイエンドのハードウェアを適切に活用します。 しかし、XFSには1つの欠点があります。 ReiserFSの場合と同様、XFSではメタデータの完全性は重視されていますが、データの完全性はそれほどではありません。

ただし、XFSの主要機能を一見すれば、XFSが、ハイエンドコンピューティングの分野で、他のジャーナリングファイルシステムの強力な競合相手となっている理由が分かります。

- 「アロケーショングループの採用による高いスケーラビリティ」 (9 ページ)
- 「ディスクスペースの効率的な管理によるハイパフォーマンス」 (10 ページ)
- 「事前割り当てによるファイルシステムの断片化の回避」 (10 ページ)

アロケーショングループの採用による高いスケーラビリティ

XFSファイルシステムの作成時に、ファイルシステムの基にあるブロックデバイスは、等しいサイズを持つ8つ以上の線形の領域に分割されます。 これらをアロケーショングループと呼びます。 各アロケーショングループは、独自のinodeと空きディスクスペースを管理します。 実用的には、アロケーショングループを、1つのファイルシステムの中にある複数のファイルシステムと見なすこともできます。 アロケーショングループは互いに独立しているのではなく、カーネルから複数を同時にアドレス指定できる、という特徴があります。 この機能は、XFSの高いスケーラビリティに大きく貢献しています。独立性の高いアロケーショングループは、性質上、マルチプロセッサシステムのニーズに適しています。

ディスクスペースの効率的な管理によるハイパフォーマンス

空きスペースとinodeは、各アロケーショングループ内のB^+-Treeによって処理されます。 B^+ツリーの採用は、XFSのパフォーマンスとスケーラビリティを大きく向上させています。XFSでは、プロセスを2分割して割り当てを処理する*遅延割り当て*を使用します。保留されているトランザクションはRAMの中に保存され、適切な量のスペースが確保されます。 XFSは、この時点では、データを正確にはどこに(ファイルシステムのどのブロックに)格納するか決定していません。決定可能な最後の瞬間まで、この決定は遅延(先送り)されます。 暫定的に使用される一時データは、ディスクに書き込まれません。XFSがデータの実際の保存場所を決定するまでに、その役割を終えているからです。このように、XFSは、書き込みのパフォーマンスを向上させ、ファイルシステムのフラグメンテーションを減少させます。遅延アロケーションは、他のファイルシステムより書き込みイベントの頻度を下げる結果をもたらすので、書き込み中にクラッシュが発生した場合、データ損失が深刻になる可能性が高くなります。

事前割り当てによるファイルシステムの断片化の回避

データをファイルシステムに書き込む前に、XFSはファイルが必要とする空きスペースを*予約*(プリアロケート、事前割り当て)します。 したがって、ファイルシステムの断片化は大幅に減少します。 ファイルの内容がファイルシステム全体に分散することがないので、パフォーマンスが向上します。

1.3 サポートされている他のファイルシステム

表1.1「Linux環境でのファイルシステムのタイプ」 (11 ページ)は、Linuxがサポートしている他のいくつかのファイルシステムを要約したものです。 これらは主に、他の種類のメディアや外部オペレーティングシステムとの互換性およびデータの相互交換を保証することを目的としてサポートされています。

表 1.1　Linux環境でのファイルシステムのタイプ

File System Type	説明
cramfs	Compressed ROM file system (圧縮ROMファイルシステム):ROM用の圧縮された読み込み専用ファイルシステムです。
hpfs	High Performance File System(ハイパフォーマンスファイルシステム) - IBM* OS/2*の標準ファイルシステム。読み取り専用モードでのみサポートされます。
iso9660	CD-ROMの標準ファイルシステム。
minix	このファイルシステムは、オペレーティングシステムに関する学術的なプロジェクトを起源とするもので、Linuxで最初に使用されたファイルシステムです。 現在では、フロッピーディスク用のファイルシステムとして使用されています。
msdos	fat、つまり当初はDOSで使用されていたファイルシステムであり、現在はさまざまなオペレーティングシステムで使用されています。
ncpfs	Novell®ボリュームをネットワーク経由でマウントするためのファイルシステム。
nfs	Network File System (ネットワークファイルシステム) - ネットワーク内の任意のコンピュータにデータを格納でき、ネットワーク経由でアクセスを付与できます。
smbfs	Server Message Block(サーバメッセージブロック)。Windows*のような製品が、ネットワーク経由でのファイルアクセスを可能にする目的で採用しています。
sysv	SCO UNIX*、Xenix、およびCoherent(PC用の商用UNIXシステム)が採用しています。

File System Type	説明
ufs	BSD*、SunOS*、およびNextStep*で使用されています。読み取り専用モードでサポートされています。
umsdos	UNIX on MS-DOS*(MS-DOS上のUNIX) - 標準fatファイルシステムに適用され、特別なファイルを作成することにより、UNIX機能(パーミッション、リンク、長いファイル名)を実現します。
vfat	Virtual FAT:fatファイルシステムを拡張したものです(長いファイル名をサポートします)。
ntfs	Windows NT file system (NTファイルシステム) - 読み取り専用です。

1.4 Linux環境での大規模ファイルサポート

当初、Linuxは最大2GBのファイルサイズをサポートしていました。マルチメディアが爆発的に普及する前、およびLinux環境で大規模データベースを運用することを誰も試みていないうちは、これで十分でした。サーバコンピューティングの重要性がますます高くなるにつれて、カーネルとCライブラリが変更され、2GBを超えるファイルサイズをサポートするようになりました。現在、ほぼすべての主要ファイルシステムでLFSがサポートされ、高度なコンピューティングを行うことができます。Linuxのファイルやファイルシステムに関する現在の制約については、「表1.2「ファイルシステムの最大サイズ(ディスクフォーマット時)」 (12 ページ)」で概要を参照できます。

表 1.2 *ファイルシステムの最大サイズ(ディスクフォーマット時)*

ファイルシステム	ファイルサイズ(バイト)	ファイルシステムのサイズ(バイト)
Ext2またはExt3 (ブロックサイズ(1KB)	2^{34} (16 GB)	2^{41} (2 TB)

ファイルシステム	ファイルサイズ(バイト)	ファイルシステムのサイズ(バイト)
Ext2またはExt3 (ブロックサイズ(2KB)	2^{38} (256 GB)	2^{43} (8 TB)
Ext2またはExt3 (ブロックサイズ(4KB)	2^{41} (2 TB)	2^{44}-4096 (-16 TB-4096バイト)
Ext2またはExt3 (ブロックサイズ8KB) (Alphaなどの、8KBページ採用のシステム)	2^{46} (64 TB)	2^{45} (32 TB)
ReiserFS v3	2^{46} (64 TB)	2^{45} (32 TB)
XFS	2^{63} (8 EB)	2^{63} (8 EB)
NFSv2(クライアント側)	2^{31} (2 GB)	2^{63} (8 EB)
NFSv3(クライアント側)	2^{63} (8 EB)	2^{63} (8 EB)

重要項目

表1.2「ファイルシステムの最大サイズ(ディスクフォーマット時)」 (12 ページ)は、ディスクフォーマット時の制限について説明しています。 2.6Linux カーネルは、操作するファイルとファイルシステムのサイズについて、独自の制限を課しています。管理の初期設定には、次のオプションがあります。

File Size
: 32ビットシステムでは、ファイルは2TB (2^{41}バイト)を超えることはできません。

ファイルシステムのサイズ
: ファイルシステムのサイズは、最大2^{73}バイトまで可能です。しかし、この制限は、現在使用可能なハードウェアが到達可能な範囲を上回っています。

1.5 追加情報

NovellのWebサイトに掲載されている*File System Primer* [http://wiki.novell.com/index.php/File_System_Primer]は、Linux用の各種ファイルシステムについて記述しています。ファイルシステムの説明、多数のファイルシステムがある理由、ワークロードやデータごとに最適なファイルシステムについて述べています。

ここまでに説明した各ファイルシステムのプロジェクトには、独自のWebページがあります。そこで詳しいドキュメントとFAQ、さらにメーリングリストを参照することができます。

- *E2fsprogs: Ext2/3/4 Filesystem Utilities* [http://e2fsprogs.sourceforge.net/]
- *Introducing Ext3* [http://www.ibm.com/developerworks/linux/library/l-fs7.html]
- *ReiserFSprogs* [http://chichkin_i.zelnet.ru/namesys/]
- *XFS: A High-Performance Journaling Filesytem* [http://oss.sgi.com/projects/xfs/]
- *OCFS2 Project* [http://oss.oracle.com/projects/ocfs2/]

Linuxファイルシステムの包括的マルチパートチュートリアルは、『*Advanced Filesystem Implementor's Guide* [http://www-106.ibm.com/developerworks/library/l-fs.html]』)のIBM developerWorksで参照できます。ファイルシステム(Linuxファイルシステムに限らない)の詳しい比較については、「ファイルシステム」のWikipediaプロジェクトを参照してください。 [http://en.wikipedia.org/wiki/Comparison_of_file_systems#Comparison]

2 新機能

このセクションで示す機能および動作の変更は、SUSE® Linux Enterprise Server 11用です。

- 2.1項 「SLES 11 SP1の新機能」 (17 ページ)
- 2.2項 「SLES 11の新機能」 (22 ページ)

2.1 SLES 11 SP1の新機能

バグフィックスに加え、このセクションに記載された機能や動作の変更は、SUSE® Linux Enterprise Server 11 SP1リリース用です。

- 2.1.1項 「iSCSIターゲット情報の保存」 (18 ページ)
- 2.1.2項 「iSCSIイニシエータの認証パラメータの変更」 (18 ページ)
- 2.1.3項 「MPIOデバイスの永続的な予約を許可」 (18 ページ)
- 2.1.4項 「MDADM 3.0.2」 (19 ページ)
- 2.1.5項 「MDRAID外部メタデータ用ブートローダサポート」 (19 ページ)
- 2.1.6項 「MDRAID外部メタデータ用のYaSTのインストールとブートサポート」 (19 ページ)

- 2.1.7項 「ルートファイルシステムを含むMDRAIDアレイのシャットダウン機能の向上」 (20 ページ)
- 2.1.8項 「iSCSIデバイス上のMD」 (20 ページ)
- 2.1.9項 「MD-SGPIO」 (21 ページ)
- 2.1.10項 「LVM 2ミラーのサイズ変更」 (21 ページ)
- 2.1.11項 「IBMサーバ上のアダプタ用ストレージドライバの更新」 (21 ページ)

2.1.1 iSCSIターゲット情報の保存

*YaST*の［*Network ServicesiSCSI Target*］機能に、iSCSIターゲット情報のエクスポートを可能にする［*保存*］オプションが追加されました。これにより、リソースの利用者に情報を提供することが容易になりました。

2.1.2 iSCSIイニシエータの認証パラメータの変更

*YaST*ネットワークサービスの*iSCSI*イニシエータ機能では、ターゲットデバイスに接続するための認証パラメータを変更することができます。それまでは、認証情報を変更するには、エントリを削除して、作成しなおす必要がありました。

2.1.3 MPIOデバイスの永続的な予約を許可

SCSIイニシエータは共有ストレージデバイス用にSCSI予約を発行することで、他のサーバのSCSIイニシエータがそのデバイスにアクセスできないようにします。この予約は、SCSI例外処理プロセスの一環として発生するSCSIリセット後も有効です。

次は、SCSI予約が役立つ可能性があるシナリオの例です。

- 単純なSAN環境では、永続的なSCSI予約により、あるサーバで使用中のLUNを別のサーバに追加してしまうといった管理者エラーを回避すること

ができます。このエラーが発生すると、データが破壊される可能性があります。この種のエラーを防ぐには、通常SANゾーニングが使用されます。

- フェールオーバーが設定された高可用性環境では、永続的なSCSI予約によって、他のサーバに予約されているSCSIデバイスに間違ったサーバが接続するのを防ぐことができます。

2.1.4 MDADM 3.0.2

バグ修正と改良を利用するには、MDADM (Multiple Devices Administration)ユーティリティ(mdadm)の最新バージョンを使用します。

2.1.5 MDRAID外部メタデータ用ブートローダサポート

MDRAIDユーティリティバージョン 3.0の外部メタデータ機能を使用して、Intel* Matrix Storage Technologyメタデータフォーマットで定義されたRAIDボリュームから、オペレーティングシステムをインストールおよび実行できるように、サポートが追加されました。これによって、DMRAID (Device Mapper RAID)インフラストラクチャから、MDRAID (Multiple Devices RAID)インフラストラクチャに機能が移動します。その結果、RAID5実装がより成熟し、MDカーネルインフラストラクチャの機能セットがより広範になります。このため、Intel、DDF(一般的なRAIDディスクデータフォーマット)、およびネイティブMDのメタデータを含むすべてのメタデータフォーマットで共通のRAIDドライバを使用することができます。

2.1.6 MDRAID外部メタデータ用のYaSTのインストールとブートサポート

YaST®インストーラツールに、RAID 0、1、10、5、および6用のMDRAID外部メタデータのサポートが追加されました。インストーラは、RAIDアレイ、およびプラットフォームのRAID機能が有効かどうかを検出することができます。Intel Matrix Storage Manager用のプラットフォームBIOSでRAIDが有効になっていると、「DMRAID」、「MDRAID」(推奨)、または「なし」のオプ

ションが提供されます。BIOSベースのRAIDアレイのアセンブルをサポートするように、initrdも変更されました。

2.1.7 ルートファイルシステムを含むMDRAIDアレイのシャットダウン機能の向上

すべてのMDRAIDアレイが削除済みとしてマーキングされるまで待機するように、シャットダウンスクリプトが変更されました。すべてのMDRAIDボリュームが書き込み操作を終えるまで、オペレーティングシステムのシャットダウンプロセスがダーティビットのクリアを待機するようになりました。

ルート(/)ファイルシステム(オペレーティングシステムとアプリケーションファイルを含むシステムボリューム)がソフトウェアRAIDアレイに常駐するかどうか検討できるように、起動スクリプト、シャットダウンスクリプト、およびinitrdに変更が加えられました。アレイのメタデータハンドラをシャットダウンプロセスの早期に開始して、シャットダウン中に最終的なルートファイルシステム環境を監視します。ハンドラは一般のkillallイベントからは除外されます。このプロセスでは、書き込みを休止し、シャットダウンの終了時にアレイのメタデータのダーティビット（アレイの再同期が必要かどうかを示す）をクリアすることもできます。

2.1.8 iSCSIデバイス上のMD

YaSTインストーラで、iSCSIデバイス上でMDの設定が行えるようになりました。

ブート時にRAIDアレイが必要な場合は、iSCSIイニシエータソフトウェアをboot.mdの前にロードし、iSCSIターゲットをRAID用の自動設定に使用できるようにします。

新規インストール用に、Libstorageは/etc/mdadm.confファイルを作成し、AUTO -allの行を追加します。更新中は、行は追加されません。/etc/mdadm.confにこの行が含まれている場合は、

```
AUTO -all
```

/etc/mdadm.confに明記されていない限り、RAIDは自動アセンブルされません。

2.1.9 MD-SGPIO

MD-SGPIOユーティリティは、sysfs(2)によってRAIDアレイを監視するスタンドアロンアプリケーションです。イベントによって、ストレージスシステムの筐体、またはドライブベイの各スロットに関連しているLEDライトの点滅を制御するLEDの変更要求がトリガされます。次の2種類のLEDシステムをサポートしています。

- 2-LEDシステム(動作LED、状態LED)
- 3 LEDシステム(動作LED、配置LED、障害LED)

2.1.10 LVM 2ミラーのサイズ変更

論理ボリュームのサイズ変更に使用されるコマンドlvresize、lvextend、およびlvreduceが、LVM2ミラーのサイズを変更できるようになりました。以前は、これらのコマンドは、論理ボリュームがミラーの場合にエラーを報告していました。

2.1.11 IBMサーバ上のアダプタ用ストレージドライバの更新

次のストレージドライバを利用可能な最新バージョンに更新することで、IBMサーバのストレージアダプタをサポートします。

- Adaptec™: aacraid、aic94xx
- Emulex™: lpfc
- LSI™: mptas、megaraid_sas

mptsasドライバがネイティブEEH (Enhanced Error Handler)の回復をサポートするようになりました。これは、電源プラットフォームをご利用のお客様の入出力デバイスにとって主要な機能です。

- qLogic™: qla2xxx、qla3xxx、qla4xxx

2.2 SLES 11の新機能

このセクションで示す機能および動作の変更は、SUSE® Linux Enterprise Server 11リリース用です。

2.2.1 EVMS2の廃止予定

EVMS2 (Enterprise Volume Management Systems)ストレージ管理ソリューションは、廃止されます。すべてのEVMS管理モジュールが、SUSE Linux Enterprise Server 11パッケージから削除されました。システムをアップグレードすると、ご利用になっているEVMS管理デバイスがLinux Volume Manager 2 (LVM2)によって自動認識され、管理されます。詳細については、「*SUSE Linux Enterprise* [http://www.novell.com/linux/volumemanagement/strategy.html] のストレージとボリューム管理の進化」を参照してください。

SUSE Linux Enterprise Server 10上でのEVMS2の管理についての詳細については、『*SUSE Linux Enterprise Server 10 SP3: ストレージ管理ガイド* [http://www.novell.com/documentation/sles10/stor_admin/data/bookinfo.html]』を参照してください。

2.2.2 デフォルトファイルシステムとしてのExt3

Ext3ファイルシステムは、インストール時およびファイルシステムの作成時にYaSTツールで推奨されるデフォルトファイルシステムとして、ReiserFSに置き換わっています。ただし、ReiserFSもまだサポートされています。詳細については、「*SUSE Linux Enterprise 10 File System Support(SUSE Linux Enterprise 10ファイルシステムのサポート)*」のWebページの「*ファイルシステムの将来的な方向性* [http://www.novell.com/linux/techspecs.html?tab=0]」を参照してください。

2.2.3 JFSファイルシステムの廃止予定

JFSファイルシステムのサポートが終了しました。JFSユーティリティが配布から削除されました。

2.2.4 OCFS2ファイルシステムを高可用性リリースに追加

SUSE Linux Enterprise High Availability Extensionの一部として、OCFS2ファイルシステムが完全にサポートされるようになりました。

2.2.5 /dev/disk/by-nameの廃止予定

SUSE Linux Enterprise Server 11パッケージでは、`/dev/disk/by-name`パスが廃止予定です。

2.2.6 /dev/disk/by-idディレクトリで継続的に有効なデバイス名

SUSE Linux Enterprise Server 11では、デフォルトのマルチパスセットアップは`udev`を使用して、マルチパス処理の開始時に`/dev/disk/by-id`ディレクトリ内の既存のシンボリックリンクを上書きします。マルチパス処理開始前のシンボリックリンクは、SCSIデバイスをその`scsi-xxx`名でポイントします。マルチパス処理実行中のシンボリックリンクは、SCSIデバイスをその`dm-uuid-xxx`名でポイントします。これにより、マルチパス処理が開始したかどうかに関わりなく、`/dev/disk/by-id`パス内のシンボリックリンクは、継続的に同じデバイスをポイントすることができます。自動的に正しいデバイスを指すので、設定ファイル(`lvm.conf`や`md.conf`)を変更する必要はありません。

このような動作変更が他の機能に及ぼす影響の詳細については、次のセクションを参照してください。

- 2.2.7項 「マルチパスデバイスのフィルタ」 (25 ページ)
- 2.2.8項 「マルチパスデバイスのユーザフレンドリな名前」 (25 ページ)

2.2.7 マルチパスデバイスのフィルタ

/dev/disk/by-nameディレクトリの廃止によって(2.2.5項 「/dev/disk/by-nameの廃止予定」 (24ページ)で説明済み)、設定ファイル内のマルチパスデバイスのフィルタ設定方法に影響を及ぼします。/etc/lvm/lvm.confファイルでマルチパスデバイスフィルタに/dev/disk/by-nameデバイス名パスを使用していた場合は、/dev/disk/by-idパスを使用するようにファイルを変更する必要があります。by-idパスを使用するフィルタの設定では、次の点に注意してください。

- /dev/disk/by-id/scsi-*デバイス名は継続的に有効で、この目的のために作成されました。
- フィルタに/dev/disk/by-id/dm-*という名前は使用しないでください。これらはデバイスマッパーデバイスへのシンボリックリンクで、pvscanコマンドに対して重複したPVを報告することになります。名前がLVM-pvuuidからdm-uuidに変更され、再度LVM-pvuuidに戻るように見えます。

フィルタの設定については、7.2.3項 「マルチパスデバイスでのLVM2の使用」 (62 ページ)を参照してください。

2.2.8 マルチパスデバイスのユーザフレンドリな名前

/dev/disk/by-idディレクトリ(2.2.6項 「/dev/disk/by-idディレクトリで継続的に有効なデバイス名」 (24ページ)で説明済み)でのマルチパスデバイス名の処理方法の変更により、デバイスの2つの名前が異なっているため、ユーザフレンドリな名前の設定に影響を及ぼします。マルチパス処理の設定後は、デバイスマッパー名だけをスキャンするように、設定ファイルを変更する必要があります。

たとえば、マルチパスデバイス名を使用してスキャンするように、lvm.confファイルを変更する必要があります。そのためには、/dev/disk/by-id/dm-uuid-.*-mpath-.*パスを、/dev/disk/by-idの代わりに指定します。

2.2.9 マルチパスのための高度な入出力負荷バランスオプション

次の高度な入出力負荷バランスオプションは、ラウンドロビンに加えて、デバイスマッパーマルチパスにも使用きます。

- Least-pending(最小保留)
- Length-load-balancing(長さによる負荷分散)
- Service-time(サービス時間)

詳細については、「優先度グループと属性の理解」 (93 ページ)を参照してください。

2.2.10 マルチパスツールコールアウトの場所の変更

デバイスマッパーマルチパスツールのmpath_* prio_calloutsが、/lib/libmultipath/lib*の共有ライブラリに移動されました。共有ライブラリを使用することで、デーモンの起動時、コールアウトがメモリにロードされます。これによって、すべてのパスがダウンするというシステムデッドロックのシナリオを回避することができます。このシナリオでは、プログラムをディスクからロードしなければなりませんが、この時点ではそのディスクが利用できない場合があります。

2.2.11 mkinitrd -fオプションのmpathからmultipathへの変更

デバイスマッパーマルチパスサービスをinitrdに追加するオプションが、-f mpathから-f multipathに変更になりました。

新しいinitrdを作成するためのコマンドは、次のようになりました。

```
mkinitrd -f multipath
```

3 ストレージソリューションのプラニング

ニーズを最適に満たすには、ストレージの要件と、ストレージスペースの効果的な管理および分割の方法を考慮する必要があります。このセクションの情報を使用して、SUSE® Linux Enterprise Server 11サーバでのファイルシステムのストレージ展開をプランニングできます。

3.1 デバイスのパーティショニング

YaST Expert Partitionerの使用については、「Using the YaST Partitioner」(『*SUSE Linux Enterprise Server 11 Installation and Administration Guide*』内)を参照してください。

3.2 マルチパスのサポート

Linuxは、サーバ/ストレージデバイス間の耐障害性接続を実現するため、複数のI/Oパスの使用をサポートしています。Linuxのマルチパスサポートは、デフォルトでは無効になっています。ストレージサブシステムのベンダから提供されるマルチパスソリューションを使用する場合は、別途、Linuxマルチパスを設定する必要はありません。

3.3 ソフトウェアRAIDのサポート

Linuxは、ハードウェアRAIDデバイスおよびソフトウェアRAIDデバイスをサポートします。ハードウェアRAIDデバイスを使用する場合は、ソフトウェアRAIDデバイスは不要です。ただし、同じサーバでハードウェアRAIDデバイスとソフトウェアRAIDデバイスの両方を使用できます。

ソフトウェアRAIDデバイスが最大限のパフォーマンスを発揮するには、それぞれのRAIDパーティションを別々の物理デバイスからとる必要があります。ソフトウェアRAID1デバイスの場合、ミラーリングされたパーティションは、ディスクを共有できません。

3.4 ファイルシステムのスナップショット

Linuxは、ファイルシステムのスナップショットをサポートします。

3.5 バックアップとアーカイブのサポート

- 3.5.1項 「オープンソースバックアップ」 (29 ページ)
- 3.5.2項 「商用バックアップとアンチウィルスのサポート」 (29 ページ)

3.5.1 オープンソースバックアップ

Linuxでデータのバックアップを行うためのオープンソースツールには、tar、cpio、およびrsyncが含まれています。詳細については、これらのツールのマニュアルページを参照してください。

- PAX: POSIXファイルシステムアーカイバ。標準的なアーカイブ(バックアップ)ファイルの最も一般的なフォーマットであるcpioおよびtarの2つをサポートしています。詳細については、のマニュアルページを参照してください。
- Amanda: The Advanced Maryland Automatic Network Disk Archiver。www.amanda.org [http://www.amanda.org/]を参照してください。

3.5.2 商用バックアップとアンチウィルスのサポート

Novell® OES (Open Enterprise Server) 2 Support Pack 1 for Linuxは、SLES (SUSE Linux Enterprise Server) 10 Support Pack 2を含んでいます。OES 2 SP1をサポートするウィルス対策/バックアップのソフトウェアベンダは、SLES 10 SP2もサポートします。スケジュールされたSLES 11のサポートについては、該当するベンダのWebサイトをご覧ください。

可能なバックアップおよびウィルス対策のソフトウェアベンダの現行リストについては、『*Novell Open Enterprise Server Partner Support: Backup and Antivirus Support* [http://www.novell.com/products/openenterpriseserver/partners_communities.html]』を参照してください。このリストは、四半期ごとに更新されます。

4

LVMの設定

このセクションでは、LVM(Logical Volume Manager)の原理と多くの状況で役立つ基本機能を簡単に説明します。YaST LVMの設定は、YaST Expert Partitionerからアクセスできます。このパーティショニングツールにより、既存のパーティションを編集、および削除できます。また、LVMで使用する新規パーティションを作成することもできます。

警告

LVMを使用することでデータ損失などの危険性が増加する恐れがあります。この危険性にはアプリケーションのクラッシュ、電源障害、誤ったコマンドなども含まれます。LVMまたはボリュームの再設定を実施する前にデータを保存してください。バックアップなしでは作業を実行しないでください。

4.1 論理ボリュームマネージャ(LVM)の理解

LVMは、複数のファイルシステムにハードディスクスペースを柔軟に分散することができます。LVMが開発された理由は、インストール中に初期パーティショニングが終了した後でのみ、ハードディスクスペースのセグメンテーションを変更するニーズが発生する可能性があるためです。稼動中のシステムでパーティションを変更することは困難なので、LVMは必要に応じて論理ボリューム(LV)を作成できるメモリスペースの仮想プール(ボリュームグループ(VG))を提供します。オペレーティングシステムは物理パーティションの代わりにこれらのLVにアクセスします。ボリュームグループは2つ以上のディスクにまたがることができます。したがって、複数のディスクまたはそれらの一部で1つのVGを構成できます。この方法で、LVMは物理ディスクスペースから一種の抽象化を行います。この抽象化により、物理パーティショニングを使用する場合よりはるかに簡単で安全な方法でセグメンテーションを変更できます。

図 4.1　物理パーティショニング対LVM

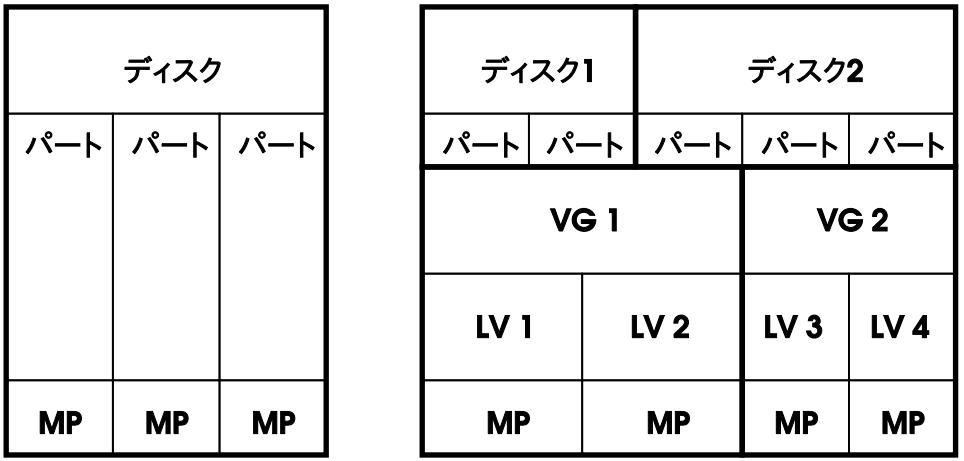


図4.1「物理パーティショニング対LVM」 (32 ページ)では物理パーティショニング(左)とLVM区分(右)を比較しています。左側は、1つのディスクが割り当てられたマウントポイント(MP)をもつ3つの物理パーティション(PART)に分かれています。これによりオペレーティングシステムはそれぞれのパーティ

ションにアクセスできます。右側では2つのディスクがそれぞれ3つの物理パーティションに分かれています。2つのLVMボリュームグループ(VG 1およびVG 2)が定義されています。VG 1にはDISK 1とDISK 2の2つのパーティションが含まれます。VG 2はDISK 2の2つのパーティションを除いた残り部分になります。LVMではボリュームグループに組み込まれた物理ディスクパーティションは物理ボリューム(PV)と呼ばれます。ボリュームグループ内に4つの論理ボリューム(LV 1からLV 4)が定義されています。これらのボリュームは、それぞれに関連づけられたマウントポイントを介してオペレーティングシステムに使用されます。別の論理ボリュームとの境界とパーティションの境界を並べることはできません。この例ではLV 1およびLV 2の間に境界があります。

LVMの機能:

- 複数のハードディスクまたはパーティションを大きな論理ボリュームにまとめることができます。
- 提供された設定が適切であれば、LV(/usrなど)は空きスペースがなくなったときに拡張することができます。
- LVMを使用することで、実行中のシステムにハードディスクまたはLVを追加できます。ただし、こうしたディスクやLVを追加するには、ホットスワップ可能なハードウェアが必要になります。
- 複数の物理ボリューム上に論理ボリュームのデータストリームを割り当てるストライピングモードを有効にすることもできます。これらの物理ボリュームが別のディスクに存在する場合、RAID 0と同様に読み込みおよび書き込みのパフォーマンスを向上できます。
- スナップショット機能は稼動中のシステムで一貫性のある(特にサーバ)バックアップを取得できます。

これらの機能とともにLVMを使用することは、頻繁に使用されるホームPCや小規模サーバではそれだけでも意義があります。データベース、音楽アーカイブ、またはユーザディレクトリのように増え続けるデータストックがある場合は、LVMが特に役に立ちます。LVMを使用すると、物理ハードディスクより大きなファイルシステムの作成が可能になります。LVMのもう1つの利点は最大256個のLVを追加できることです。ただし、LVMでの作業は従来のパーティションでの作業とは異なることに留意してください。LVMの設定に関する指示や詳細情報については、公式の『*LVM HOWTO* [http://tldp.org/HOWTO/LVM-HOWTO/]』を参照してください。

カーネルバージョン2.6から開始して、LVMバージョン2を利用することができます。これはLVMの前バージョンとの下方互換になり、これまでのボリュームグループを管理できるようにします。新しいボリュームグループを作成する場合は、新しいフォーマットまたは下方互換バージョンのどちらを使用するか決定します。LVM 2にはいずれのカーネルパッチも必要ありません。これは、カーネル2.6に統合されているデバイスマッパーを活用しています。このカーネルはLVMバージョン2のみをサポートしています。そのため、このセクションでLVMと書かれている場合、それはLVMバージョン2を指しています。

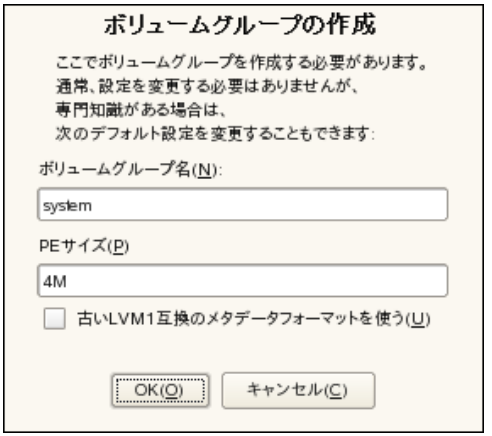
4.2 LVMパーティションの作成

LVMパーティションを作成するには、まず、［*作成*］［*Do not format(フォーマットしない)*］の順にクリックし、次に、パーティションの識別子として［*0x8E Linux LVM*］を選択します。LVMで使用するすべてのパーティションを作成した後に、［*LVM*］をクリックして、LVMの設定を開始します。

4.3 ボリュームグループの作成

システムにまだボリュームグループが存在しない場合、ボリュームグループを追加するようにプロンプトされます(図4.2「ボリュームグループの作成」(35 ページ)を参照)。［*グループの追加*］で追加グループを作成できますが、通常はボリュームグループは1つで十分です。SUSE® Linux Enterprise Serverの; システムファイルを置くボリュームグループの名前としては、`system`が推奨されています。物理エクステントサイズではボリュームグループの物理ブロックサイズを定義します。ボリュームグループにある全ディスクスペースはこの物理ブロックサイズ内で使用されます。この値は通常4MBに設定され、物理ボリュームおよび論理ボリュームには最大サイズとして256GB使用できます。物理エクステントは論理ボリュームとして256GB以上必要な場合のみ、8、16、32MBのように増やしてください。

図 4.2　ボリュームグループの作成

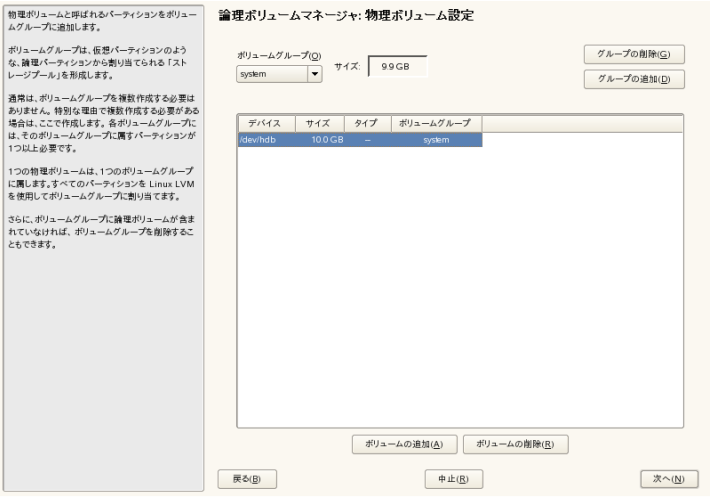


4.4 物理ボリュームの設定

ボリュームグループの作成後、次のダイアログ(「図4.3「物理ボリュームの設定」 (36ページ)」参照)に、「Linux LVM」タイプまたは「Linuxネイティブ」タイプのすべてのパーティションがリストされます。スワップパーティションまたはDOSパーティションは表示されません。パーティションがボリュームグループにすでに割り振られている場合、ボリュームグループの名前がリストに表示されます。割り当てられていないパーティションは、「--」で示されます。

複数のボリュームグループが存在する場合は、選択ボックスで現在のボリュームグループを左上に設定します。右上にあるボタンは追加ボリュームグループの作成および既存ボリュームグループの削除を実行します。ボリュームグループのパーティションが未割り当ての場合のみ、そのボリュームグループを削除できます。ボリュームグループに割り当てられたすべてのパーティションも、同様に物理ボリュームとして参照されます。

図 4.3　物理ボリュームの設定



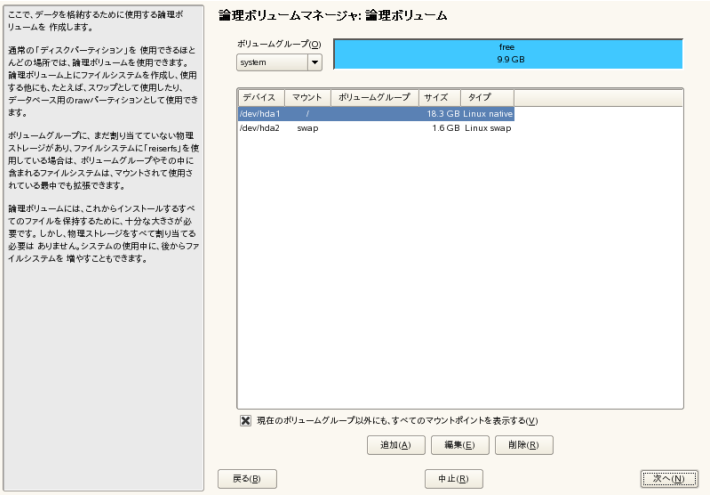
これまで未割り当てだったパーティションを、選択したボリュームグループに追加するには、まず、そのパーティションをクリックしてから［*ボリュームの追加*］をクリックします。この時点で、そのボリュームグループの名前が選択したパーティションの隣に入力されます。LVM用に予約されているパーティションをすべて1つのボリュームグループに割り当ててください。すべてのパーティションを割り当てないと、パーティションのスペースが未使用のまま残ります。ダイアログを終了する前に、すべてのボリュームグループを少なくとも1つの物理ボリュームに割り当てる必要があります。すべての物理ボリュームを割り当て終えた後、［次へ］をクリックして論理ボリュームの設定に進みます。

4.5 物理ボリュームの設定

物理ボリュームでボリュームグループが満たされたら、次のダイアログ(「図 4.4「論理ボリューム管理」 (37 ページ)」参照)を使用して、オペレーティングシステムによって使用される論理ボリュームを定義します。現在のボリュームグループを選択ボックスで左上に設定します。設定したボリュームグループの隣に現在の空き領域が表示されます。下のリストにはボリュームグループの全論理ボリュームが表示されます。マウントポイントが割り当てられている通常の全Linuxパーティション、全スワップパーティション、既存の全論理ボリュームがここにリストされています。［*追加*］、［*編集*］、［*削除*］

の各オプションを使用して、ボリュームグループ内のすべてのスペースがなくなるまで、必要に応じて論理ボリュームを管理します。各ボリュームグループに少なくとも1つの論理ボリュームを割り当ててください。

図 4.4　論理ボリューム管理



新しい論理ボリュームを作成するには(「図4.5「論理ボリュームの作成」(38 ページ)」参照)、［*追加*］をクリックし、開いたポップアップに入力します。パーティショニングの場合は、サイズ、ファイルシステム、およびマウントポイントを指定します。通常、ReiserFSまたはExt2などのファイルシステムの場合は、論理ボリュームでの作成後に、マウントポイントを指定します。この論理ボリューム上に格納されたファイルは、インストールしたシステム上の該当するマウントポイントで検出することができます。さらに、複数の物理ボリューム上(ストライピング)に存在する論理ボリュームにデータストリームを分配することも可能です。これらの物理ボリュームが別のハードディスクに存在する場合、この性質により、読み込みおよび書込みのパフォーマンスが向上します(RAID 0など)。ただし、nストライプでLVをストライピングする場合、LVが必要とするハードディスクスペースが物理ボリュームn個に等しく配分されている場合にのみ、ストライプが正しく作成されます。たとえば、使用可能な物理ボリュームが2個だけの場合、3個の論理ボリュームを持つことはできません。

警告

には、現時点でストライピングの観点からエントリの正確性を確認する機会はありません。何か間違いがあった場合、それが明らかになるのはLVMがディスクに実装された後です。

図 4.5　論理ボリュームの作成

すでにシステム上にLVMを設定した場合は、ここで既存の論理ボリュームを指定できます。続行する前に、これらの論理ボリュームを適切なマウントポイントに割り当てます。［次へ］をクリックして、Yast Expert Partitionerに戻り、作業を完了します。

4.6 LVMの直接管理

LVMをすでに設定し、一部に変更を加えるだけの場合は、他の方法があります。YaSTコントロールセンターで、［システム］［パーティショナ］の順に選択します。LVMシステムは、すでに説明した方法で管理できます。

4.7 LVMパーティションのサイズ変更

論理ボリュームのサイズ変更には、コマンドlvresize、lvextend、およびlvreduceが使用されます。構文とオプションについては、これらの各コマンドのマニュアルページを参照してください。

YaSTパーティショナでも、論理ボリュームのサイズを増やすことができます。YaSTではparted(8)を使用してパーティションサイズを増やします。

LVを拡大するには、VG上に十分な未使用スペースがなければなりません。

LVは使用中に拡大、縮小できますが、LVにあるファイルシステムの場合はそれが不可能な場合があります。LVを拡大、縮小しても、そのボリューム内のファイルシステムのサイズは自動的に変更されません。後からファイルシステムを拡大するには、別なコマンドを使用する必要があります。ファイルシステムのサイズ変更の詳細については、第5章 *ファイルシステムのサイズ変更* (41 ページ)を参照してください。

次のように適切なシーケンスを使用してください。

- LVを拡張する場合は、ファイルシステムを拡大する前にLVを拡張する必要があります。
- LVを縮小する場合は、LVを縮小する前にファイルシステムを縮小する必要があります。

論理ボリュームのサイズを拡張するには:

1. 端末コンソールを開き、rootユーザとしてログインします。
2. 仮想マシン(Xen VMなど)用に提供されているファイルシステムが論理ボリュームに含まれている場合は、VMをシャットダウンします。
3. 論理ボリューム上のファイルシステムのマウントを解除します。
4. 端末コンソールのプロンプトに対して、次のコマンドを入力し、論理ボリュームのサイズを拡大します。

```
lvextend -L +size /dev/vgname/lvname
```

*size*の場合は、10GBのように、論理ボリュームに追加したい容量を指定してください。*/dev/vgname/lvname*を、/dev/vg1/v1などの論理ボリュームへのLinuxパスに入れ替えます。次に例を示します。

```
lvextend -L +10GB /dev/vg1/v1
```

たとえば、LVをLV上の(マウント済みでアクティブな)ReiserFSで10GB拡張するには:

```
lvextend -L +10G /dev/vgname/lvname
resize_reiserfs -s +10GB -f /dev/vg-name/lv-name
```

たとえば、LVをLV上のReiserFSで5GB縮小するには:

```
umount /mountpoint-of-LV
resize_reiserfs -s -5GB /dev/vgname/lvname
lvreduce /dev/vgname/lvname
mount /dev/vgname/lvname /mountpoint-of-LV
```

5 ファイルシステムのサイズ変更

データのボリュームを増やす必要がある場合は、そのファイルシステムに割り当てられた容量を増やす必要があるかもしれません。

5.1 サイズ変更のガイドライン

パーティションまたはファイルシステムのサイズ変更には、データを失う可能性をはらむリスクが伴います。

警告

データの喪失を避けるには、データをバックアップしてから、サイズ変更タスクを開始します。

ファイルシステムのサイズを変更する場合は、次のガイドラインに従ってください。

- 5.1.1項 「サイズ変更をサポートしているファイルシステム」 (42 ページ)
- 5.1.2項 「ファイルシステムのサイズの増加」 (42 ページ)
- 5.1.3項 「ファイルシステムのサイズの削減」 (43 ページ)

5.1.1 サイズ変更をサポートしているファイルシステム

ボリュームに使用可能な容量を増やせるようにするには、ファイルシステムがサイズ変更をサポートしている必要があります。SUSE® Linux Enterprise Server 11では、Ext2、Ext3、およびReiserFSに関して、ファイルシステムのサイズ変更用ユーティリティを使用できます。このユーティリティは、次のようにサイズの増減をサポートします。

表 5.1 ファイルシステムサイズ変更のサポート

ファイルシステム	ユーティリティ	サイズを増加(拡大)	サイズの削減(縮小)
Ext2	resize2fs	はい(ただし、オフラインのみ)	はい(ただし、オフラインのみ)
Ext3	resize2fs	はい(オンラインまたはオフライン)	はい(オンラインまたはオフライン)
ReiserFS	resize_reiserfs	はい(オンラインまたはオフライン)	はい(ただし、オフラインのみ)

5.1.2 ファイルシステムのサイズの増加

デバイス上で使用可能な最大容量までファイルシステムを拡大することも、正確なサイズを指定することもできます。ファイルシステムのサイズを拡大する前に、デバイス、または論理ボリュームのサイズを拡大しておいてください。

ファイルシステムに正確なサイズを指定する場合は、その新しいサイズが次の条件を満たすかどうか確認してください。

- 新しいサイズは、既存データのサイズより大きくなければなりません。さもないと、データが失われます。
- ファイルシステムのサイズは使用可能な容量より大きくできないので、新しいサイズは、現在のデバイスサイズ以下でなければなりません。

5.1.3 ファイルシステムのサイズの削減

デバイス上のファイルシステムのサイズを削減する際には、新しいサイズが次の条件を満たすかどうか確認してください。

- 新しいサイズは、既存データのサイズより大きくなければなりません。さもないと、データが失われます。
- ファイルシステムのサイズは使用可能な容量より大きくできないので、新しいサイズは、現在のデバイスサイズ以下でなければなりません。

ファイルシステムが保存されている論理ボリュームのサイズを削減する場合は、デバイス、または論理ボリュームのサイズを削減しようとする前に、必ずファイルシステムのサイズを削減しておきます。

5.2 Ext2またはExt3ファイルシステムの拡大

Ext2とExt3のファイルシステムは、`resize2fs`コマンドでマウントまたはアンマウントする際にサイズ変更できます。

1. 端末コンソールを開いて、`root`ユーザまたは同等の権限でログインします。
2. 次の方法の1つで、ファイルシステムのサイズを増加します。
   - ファイルシステムのサイズを`/dev/sda1`と呼ばれるデバイスの、利用可能な最大サイズまで拡大するには、次のように入力します。

```
resize2fs /dev/sda1
```

sizeパラメータを指定しない場合、サイズはパーティションのサイズにデフォルト設定されます。

- ファイルシステムを特定のサイズに拡張するには、次のコマンドを入力します。

```
resize2fs /dev/sda1 size
```

*size*パラメータは、要求されたファイルシステムの新サイズを指定します。単位を指定しない場合のsizeパラメータの単位は、ファイルシステムのブロックサイズです。オプションとして、sizeパラメータの後ろに、次の単位指定子の1つを付けることができます。sは512バイトのセクタ、Kはキロバイト(1キロバイトは1024バイト)、Mはメガバイト、Gはギガバイトを表します。

サイズ変更が完了するまで待って、続行します。

3 ファイルシステムがマウントされていない場合は、この時点で、ファイルシステムをマウントします。

たとえば、Ext2ファイルシステムを、/dev/sda1という名前のデバイスに、マウントポイント/homeでマウントするには、次のように入力します。

```
mount -t ext2 /dev/sda1 /home
```

4 次のように入力して、マウントされたファイルシステムに対するサイズ変更の効果をチェックします。

```
df -h
```

ディスクフリー(df)コマンドは、ディスクの合計サイズ、使用されたブロック数、およびファイルシステム上の使用可能なブロック数を表示します。-hオプションは、読みやすい形式でサイズを出力します(1K,、234M、2Gなど)。

5.3 Reiserファイルシステムのサイズの増加

ReiserFSファイルシステムのサイズは、マウント中またはアンマウント中のいずれでも拡大することができます。

1 端末コンソールを開いて、rootユーザまたは同等の権限でログインします。

2 /dev/sda2と呼ばれるデバイスのファイルシステムのサイズを拡大するには、次のいずれかの方法を使用します。

- ファイルシステムのサイズをデバイスの使用可能な最大サイズまで拡張するには、次のように入力します。

  ```
  resize_reiserfs /dev/sda2
  ```

  サイズを指定しないと、ボリュームはパーティションのフルサイズまで拡張されます。

- ファイルシステムを特定のサイズに拡張するには、次のコマンドを入力します。

  ```
  resize_reiserfs -s size /dev/sda2
  ```

  *size*を目的のサイズ(バイト単位)で置き換えます。50000K(キロバイト)、250M(メガバイト)、2G(ギガバイト)など、値の単位を指定することもできます。代わりに、プラス(+)記号を値の前に付けることにより、現在のサイズに対する増加を指定することもできます。たとえば、次のコマンドは、/dev/sda2上のファイルシステムのサイズを500MB分増加します。

  ```
  resize_reiserfs -s +500M /dev/sda2
  ```

サイズ変更が完了するまで待って、続行します。

3 ファイルシステムがマウントされていない場合は、この時点で、ファイルシステムをマウントします。

たとえば、ReiserFSファイルシステムを、デバイス/dev/sda2に、マウントポイント/homeでマウントするには、次のように入力します。

```
mount -t reiserfs /dev/sda2 /home
```

4 次のように入力して、マウントされたファイルシステムに対するサイズ変更の効果をチェックします。

```
df -h
```

ディスクフリー(df)コマンドは、ディスクの合計サイズ、使用されたブロック数、およびファイルシステム上の使用可能なブロック数を表示します。-hオプションは、読みやすい形式でサイズを出力します(1K,、234M、2Gなど)。

5.4 Ext2またはExt3ファイルシステムのサイズの削減

Ext2とExt3のファイルシステムは、マウントまたはアンマウント時にサイズ変更できます。

1 端末コンソールを開いて、rootユーザまたは同等の権限でログインします。

2 /dev/sda1などのデバイスのファイルシステムのサイズを削減するには、次のコマンドを入力します。

```
resize2fs /dev/sda1 <size>
```

*size*を、目的のサイズを表す整数値(キロバイト単位)で置き換えます(1キロバイトは1024バイト)。

サイズ変更が完了するまで待って、続行します。

3 ファイルシステムがマウントされていない場合は、この時点で、ファイルシステムをマウントします。たとえば、Ext2ファイルシステム

を、/dev/sda1という名前のデバイスに、マウントポイント/homeでマウントするには、次のように入力します。

```
mount -t ext2 /dev/md0 /home
```

4 次のように入力して、マウントされたファイルシステムに対するサイズ変更の効果をチェックします。

```
df -h
```

ディスクフリー(df)コマンドは、ディスクの合計サイズ、使用されたブロック数、およびファイルシステム上の使用可能なブロック数を表示します。-hオプションは、読みやすい形式でサイズを出力します(1K,、234M、2Gなど)。

5.5 Reiserファイルシステムのサイズの削減

Reiserファイルシステムは、ボリュームがアンマウントされている場合のみ、サイズを削減することができます。

1 端末コンソールを開いて、rootユーザまたは同等の権限でログインします。

2 次のコマンドを入力して、デバイスをアンマウントします。

```
umount /mnt/point
```

サイズを削減するパーティションにシステムファイル(ルート/ボリュームなど)が含まれていると、ブート可能なCDまたはフロッピーからブートする場合のみ、アンマウントが可能です。

3 /dev/sda1という名前のデバイスのファイルシステムのサイズを削減するには、次のコマンドを入力します。

```
resize_reiserfs -s size /dev/sda2
```

*size*を目的のサイズ(バイト単位)で置き換えます。50000K(キロバイト)、250M(メガバイト)、2G (ギガバイト)など、値の単位を指定することもできます。代わりに、マイナス(-)記号を値の前に付けて、現在のサイズに対する削減分を指定することもできます。たとえば、次のコマンドは、/dev/md0上のファイルシステムのサイズを 500 MB分減少させます。

```
resize_reiserfs -s -500M /dev/sda2
```

サイズ変更が完了するまで待って、続行します。

4 次のコマンドで、ファイルシステムをマウントします。

```
mount -t reiserfs /dev/sda2 /mnt/point
```

5 次のように入力して、マウントされたファイルシステムに対するサイズ変更の効果をチェックします。

```
df -h
```

ディスクフリー(df)コマンドは、ディスクの合計サイズ、使用されたブロック数、およびファイルシステム上の使用可能なブロック数を表示します。-hオプションは、読みやすい形式でサイズを出力します(1K,、234M、2Gなど)。

UUIDによるデバイスのマウント

6

このセクションでは、オプションとして、デバイス名の代わりにUUIDを使用して、ブートローダファイルと/etc/fstabファイルでファイルシステムのデバイスを識別する方法について説明します。

6.1 udevによるデバイスの命名

Linux 2.6以降のカーネルでは、udevによって、永続的なデバイス名を使用した/devディレクトリのユーザスペースソリューションが提供されます。システムに対してデバイスを追加または削除する場合は、ホットプラグシステムの一部としてudevが実行されます。

ルールのリストが特定デバイス属性との比較に使用されます。udevルールのインフラストラクチャ(/etc/udev/rules.dディレクトリで定義)は、すべ

てのディスクデバイスに、それらの認識順序や当該デバイスに使用される接続に関わらず、安定した名前を提供します。udevツールは、カーネルが作成するすべての該当ブロックデバイスを調べ、一定のバス、ドライブタイプ、またはファイルシステムに基づいて、ネーミングルールを適用します。udev用の独自ルールを定義する方法については、「*Writing udev Rules* [http://reactivated.net/writing_udev_rules.html]」を参照してください。

動的なカーネル提供のデバイスノード名に加えて、udevは、/dev/diskディレクトリ内のデバイスをポイントする永続的なシンボリックリンクのクラスを保持します。このディレクトリは、さらに、by-id、by-label、by-path、およびby-uuidの各サブディレクトリに分類されます。

注記

udev以外のプログラム(LVMやmdなど)も、UUIDを生成することがありますが、それらのUUIDは/dev/diskにリストされません。

6.2 UUIDの理解

UUID (Universally Unique Identifier)は、ファイルシステムの128ビットの番号であり、ローカルシステムと他のシステム全体に渡る固有な識別子です。UUIDは、システムハードウェア情報とタイムスタンプをそのシードの一部として、ランダムに生成されます。UUIDは、通常、デバイスに固有なタグを付けるために使用されます。

- 6.2.1項 「UUIDによるファイルシステムデバイスのアセンブルまたは有効化」 (50 ページ)
- 6.2.2項 「ファイルシステムデバイスのUUIDの見つけ方」 (51 ページ)

6.2.1 UUIDによるファイルシステムデバイスのアセンブルまたは有効化

UUIDは、常に、パーティションに対して固有であり、表示の順序またはマウントの場所に依存しません。特定のSANデバイスがサーバに接続されると、システムパーティションが名前変更され、最後のデバイスに移動されます。

たとえば、ルート(/)がインストール時に/dev/sda1に割り当てられている場合は、SANの接続後に/dev/sdg1に割り当てられることがあります。この問題を回避する1つの方法は、ブートデバイスのブートローダファイルと/etc/fstabファイルでUUIDを使用することです。

製造業者がドライブに割り当てたデバイスIDは、デバイスをどこにマウントしても変更されないので、常に、ブート時に検出できます。UUIDは、ファイルシステムのプロパティであり、ドライブを再フォーマットすれば変更できます。ブートローダファイルでは、通常、デバイスの場所(/dev/sda1など)を指定して、デバイスをシステムブート時にマウントします。ブートローダは、デバイスのUUIDと管理者指定のボリュームラベルによってもデバイスをマウントできます。ただし、ラベルとファイルの場所を使用する場合は、パーティションのマウント時にラベル名を変更できません。

UUIDは、ソフトウェアRAIDデバイスのアセンブルと起動の基準として使用できます。RAIDが作成されると、mdドライバは、デバイスのUUIDを生成し、その値をmdスーパーブロックに保存します。

6.2.2 ファイルシステムデバイスのUUIDの見つけ方

どのブロックデバイスのUUIDも、/dev/disk/by-uuidディレクトリ内で見つけることができます。たとえば、次のようなUUIDがあります。

```
e014e482-1c2d-4d09-84ec-61b3aefde77a
```

6.3 ブートローダと/etc/fstabファイルでのUUIDの使用(x86の場合)

インストール後、オプションとして、次の手順に従って、x86システムのブートローダファイルと/etc/fstabファイルでシステムデバイスのUUIDを設定できます。

プロシージャを開始する前に、/boot/grub/menu.1stファイルと/etc/fstabファイルのコピーをとってください。

1 SANデバイスを接続せずに、SUSE® Linux Enterprise Server for x86をインストールします。

2 インストール後、システムをブートします。

3 rootユーザまたはそれと同等の権限で、端末コンソールを開きます。

4 /dev/disk/by-uuidディレクトリにナビゲートして、/boot、/root、およびswapをインストールしたデバイスのUUIDを見つけます。

4a 端末コンソールのプロンプトで、次のように入力します。

```
cd /dev/disk/by-uuid
```

4b 次を入力して、すべてのパーティションをリストします。

```
ll
```

4c 次のようなUUIDを見つけます。

```
e014e482-1c2d-4d09-84ec-61b3aefde77a -> /dev/sda1
```

5 YaST2でブートローダオプションを使用するか、またはテキストエディタを使用して、/boot/grub/menu.1stファイルを編集します。

たとえば、次のように変更します。

```
kernel /boot/vmlinuz root=/dev/sda1
```

変更先:

```
kernel /boot/vmlinuz
root=/dev/disk/by-uuid/e014e482-1c2d-4d09-84ec-61b3aefde77a
```

重要項目

間違えた場合は、SAN接続なしでサーバをブートし、参考として/boot/grub/menu.1stファイルのバックアップコピーを使用してエラーを修正できます。

YaSTでブートローダオプションを使用する場合は、値を変更すると、ブートローダファイルに重複行が追加されるという欠陥があります。エディタを使用して、次の重複行を削除します。

```
color white/blue black/light-gray

default 0

timeout 8

gfxmenu (sd0,1)/boot/message
```

YaSTで、ルート(/)デバイスをマウントする方法(UUIDを使用する、ラベルを使用するなど)を変更する際には、ブートローダの設定を再保存して、変更内容をブートローダに対して有効にする必要があります。

6 rootユーザまたはそれと同等の権限で、次のいずれかを実行してUUIDを/etc/fstabファイルに入力します。

- YaSTを［システム］［*ディスクの分割*］の順に開き、目的のデバイスを選択し、最後に［*fstabオプション*］を変更します。
- /etc/fstabファイルを編集して、システムデバイスのエントリを場所からUUIDに変更します。

  たとえば、ルート(/)ボリュームのデバイスパスが/dev/sda1で、そのUUIDがe014e482-1c2d-4d09-84ec-61b3aefde77aの場合、次のように行エントリを変更します。

  ```
  /dev/sda1   /            reiserfs   acl,user_xattr        1 1
  ```

  変更先:

```
UUID=e014e482-1c2d-4d09-84ec-61b3aefde77a   /  reiserfs
acl,user_xattr        1 1
```

重要項目

ファイル内に、不要な文字やスペースが残っていないようにしてください。

6.4 ブートローダと/etc/fstabファイルでのUUIDの使用(IA64の場合)

インストール後、次のプロシージャに従って、IA64システムのブートローダファイルと/etc/fstabファイルにシステムデバイスのUUIDを設定します。IA64はEFI BIOSを使用します。ファイルシステムの環境設定ファイルは、/boot/efi/SuSE/elilo.confであり、/etc/fstabではありません。

プロシージャを開始する前に、/boot/efi/SuSE/elilo.confファイルのコピーをとります。

1. SANデバイスを接続せずに、SUSE Linux Enterprise Server for IA64をインストールします。
2. インストール後、システムをブートします。
3. rootユーザまたはそれと同等の権限で、端末コンソールを開きます。
4. /dev/disk/by-uuidディレクトリにナビゲートして、/boot、/root、およびswapをインストールしたデバイスのUUIDを見つけます。

   4a 端末コンソールのプロンプトで、次のように入力します。

   ```
   cd /dev/disk/by-uuid
   ```

   4b 次を入力して、すべてのパーティションをリストします。

   ```
   ll
   ```

4c 次のようなUUIDを見つけます。

```
e014e482-1c2d-4d09-84ec-61b3aefde77a -> /dev/sda1
```

5 YaST2でブートローダオプションを使用して、ブートローダファイルを編集します。

たとえば、次のように変更します。

```
root=/dev/sda1
```

変更先:

```
root=/dev/disk/by-uuid/e014e482-1c2d-4d09-84ec-61b3aefde77a
```

6 `/boot/efi/SuSE/elilo.conf`ファイルを編集して、システムデバイスのエントリを場所からUUIDに変更します。

たとえば、次のように変更します。

```
/dev/sda1   /   reiserfs   acl,user_xattr        1 1
```

変更先:

```
UUID=e014e482-1c2d-4d09-84ec-61b3aefde77a   /  reiserfs   acl,user_xattr
        1 1
```

重要項目

ファイル内に、不要な文字やスペースが残っていないようにしてください。

6.5 追加情報

`udev(8)`によるデバイス管理の詳細については、「「Dynamic Kernel Device Management with udev」 [http://www.novell.com/documentation/sles11/book_sle_admin/data/cha_udev.html]」(『*SUSE® Linux*

Enterprise Server 11 Installation and Administration Guide』内)を参照してください。

udev(8)コマンドの詳細については、そのマニュアルページを参照してください。端末コンソールのプロンプトで、次のように入力します。

```
man 8 udev
```

デバイスのマルチパスI/Oの管理

7

このセクションでは、サーバ/ブロックストレージデバイス間のマルチパスのフェールオーバーおよびパスの負荷分散を管理する方法について説明します。

7.1 マルチパス処理の理解

7.1.1 マルチパス処理とは

マルチパス処理とは、サーバのホストバスアダプタおよびデバイスのストレージコントローラ間で、複数の物理パスをまたいで、同じ物理または論理ブロックストレージデバイスと通信するサーバの機能です。これは、通常、FC (Fibre Channel)環境またはiSCSI SAN環境で行われます。複数のチャネルが使用可能な際には、内蔵ストレージとのマルチ接続も可能です。

7.1.2 マルチパス処理のメリット

Linuxマルチ処理は、接続に耐障害性を与え、アクティブな接続全体.に負荷を分散します。マルチパス処理が設定および実行されていると、自動的に、デバイス接続の障害が特定され、I/Oが代替の接続に再経路指定されます。

接続に関しては、多くのトラブルが欠陥のあるアダプタ、ケーブル、またはコントローラが原因で発生します。デバイスにマルチパスI/Oを設定すると、マルチパスドライバがデバイス間のアクティブな接続を監視します。マルチパスドライバは、アクティブなパスのI/Oエラーを検出すると、トラフィックをデバイスの指定セカンダリパスにフェールオーバーします。該当するパスが正常に戻ると、そのパスに制御を戻すことができます。

7.2 マルチパス処理のプラニング

7.2.1 マルチパス処理のガイドライン

マルチパスI/Oソリューションのプラニング時には、このセクションのガイドラインに従ってください。

- 「仮想化環境」 (61 ページ)

前提条件

- マルチパス処理は、デバイスレベルで管理されます。
- マルチパス処理対象のデバイスに使用するストレージアレイで、マルチパス処理がサポートされている必要があります。詳細については、7.2.9項「マルチパス処理用にサポートされているストレージアレイ」 (67 ページ)を参照してください。
- サーバのホストバスアダプタおよびブロックストレージデバイスのバスコントローラ間に複数の物理パスが存在している場合のみ、マルチパス処理を設定する必要があります。論理デバイスのマルチパスは、サーバの見地から設定します。

ベンダ提供のマルチパスソリューション

一部のストレージアレイについては、アレイの物理および論理デバイスのマルチパス処理を管理するための独自のマルチパス処理ソフトウェアがベンダから提供されます。この場合は、ベンダの指示に従って、それらのデバイスのマルチ処理を設定してください。

ディスク管理タスク

マルチパスを持つ物理デバイスまたは論理デバイスのマルチパス処理を設定する前に、まず、次のようにディスク管理タスクを実行してください。

- サードパーティーツールで、物理ディスクを小さな論理ディスクに切り分けます。
- サードパーティーツールで、物理ディスクまたは論理ディスクをパーティションに分割します。稼動中のシステムでパーティションを変更した場合は、DM-MP(Device Mapper Multipath: デバイスマッパーマルチパス)モジュールによるそれら変更の自動的な検出や反映は行われません。DM-MPIOは再初期化する必要があり、それには、通常、再起動が必要です。
- サードパーティーのSANアレイ管理ツールを使用して、ハードウェアRAIDデバイスを作成および設定します。

- サードパーティーのSANアレイ管理ツールを使用して、LUNなどの論理デバイスを作成します。所定のアレイにサポートされている論理デバイスタイプは、アレイベンダによって異なります。

ソフトウェアRAID

LinuxのソフトウェアRAIDの管理ソフトウェアは、マルチパス処理の上で実行されます。複数のI/Oパスを持ち、ソフトウェアRAIDで使用予定の各デバイスは、まず、マルチパス処理用に設定してから、ソフトウェアRAIDデバイスとして作成する必要があります。マルチパスデバイスは自動検出できません。ソフトウェアRAIDは、その下で実行されているマルチパス処理管理を認識しません。

高可用性ソリューション

クラスタリング用の高可用性ソリューションは、通常、マルチパス処理サーバの上で実行されます。たとえば、LAN上のデバイスをミラーリングするDRBD (Distributed Replicated Block Device)高可用性ソリューションは、マルチパス処理をベースとして実行されます。複数のI/Oパスを持ち、DRDBソリューションで使用予定のデバイスごとに、マルチパス処理用デバイスを設定してから、DRBDを設定する必要があります。

ボリュームマネージャ

LVM2やEVMSなどのボリュームマネージャは、マルチパス処理をベースとして実行されます。LVM2またはEVMSを使用してセグメントマネージャおよびそのファイルシステムを作成するには、その前に、デバイスのマルチパス処理を設定する必要があります。

仮想化環境

仮想化環境でマルチパス処理を使用する場合、マルチパス処理は、ホストサーバ環境で制御されます。デバイスのマルチパス処理を設定してから、デバイスを仮想ゲストマシンに割り当ててください。

7.2.2 マルチパスデバイスのby-id名の使用

LUNディレクトリ全体を使用する場合は(たとえばSAN機能を使用してストレージのパーティションを行っている場合など)、mkfs、fstab、ご使用のアプリケーションなどに、/dev/disk/by-id/xxxという名前を使用することができます。

/etc/multipath.confファイルでユーザフレンドリ名のオプションが有効になっている場合は、/dev/disk/by-id/dm-uuid-.*-mpath-.*デバイス名を使用できます。これはこの名前がデバイスIDの別名だからです。詳細については、「/etc/multipath.confでのユーザフレンドリな名前またはエイリアス名の設定」 (86 ページ)を参照してください。

7.2.3 マルチパスデバイスでのLVM2の使用

LVM2は、デフォルトでは、マルチパスデバイスを認識しません。マルチパスデバイスを可能な物理ボリュームとしてLVM2に認識させるには、/etc/lvm/lvm.confファイルを変更する必要があります。その場合、LVM2が物理パスをスキャンせずに使用し、マルチパスI/Oレイヤを介してマルチパスI/Oストレージだけにアクセスするように変更することが重要です。ユーザフレンドリな名前を使用している場合は、マルチパス処理の設定後にデバイス(/dev/disk/by-id/dm-uuid-.*-mpath-)のデバイスマッパー名だけをスキャンするようにパスを指定します。

/etc/lvm/lvm.confを変更してマルチパスを使用するには次の手順に従います。

1. /etc/lvm/lvm.confファイルをテキストエディタで開きます。

   /etc/lvm/lvm.confが存在しない場合は、次のコマンドを端末コンソールのプロンプトで入力することによって、現在のLVM設定に基づいた設定ファイルを作成できます。

   ```
   lvm dumpconfig > /etc/lvm/lvm.conf
   ```

2. /etc/lvm/lvm.confにあるfilterエントリとtypesエントリを、次のように変更します。

```
filter = [ "a|/dev/disk/by-id/.*|", "r|.*|" ]
types = [ "device-mapper", 1 ]
```

このように設定すると、LVM2はby-idパスだけをスキャンし、それ以外のすべてのパスを拒否します。

ユーザフレンドリな名前を使用している場合は、次のようにパスを指定し、マルチパス処理の設定後に、デバイスマッパー名だけがスキャンされるようにします。

```
filter = [ "a|/dev/disk/by-id/dm-uuid-.*-mpath-.*|", "r|.*|" ]
```

3 非マルチパスデバイスでもLVM2を使用する場合は、filterエントリとtypesエントリに必要な調整を加えて、ご使用の設定に合わせます。さもないと、lvm.confファイルをマルチパス向けに変更後、pvscanで他のLVMデバイスが認識できなくなります。

LVMを使用して設定されたデバイスだけをLVMキャッシュに含めるので、filterでどの非マルチパスデバイスを含めるかについて詳細に指定できる必要があります。

たとえば、ローカルディスクが/dev/sda、すべてのSANデバイスが/dev/sdbとそれ以上の場合、次のようにfilterでローカルパスとマルチパスを指定します。

```
filter = [ "a|/dev/sda.*|", "a|/dev/disk/by-id/.*|", "r|.*|" ]
types = [ "device-mapper", 253 ]
```

4 ファイルを保存します。

5 dm-multipathを/etc/sysconfig/kernel:INITRD_MODULESに追加します。

6 新しいinitrdを作成して、デバイスマッパーマルチパスのサービスが変更後の設定でロードされるようにします。<Enter>端末コンソールのプロンプトで、次のように入力します。

```
mkinitrd –f multipath
```

7 サーバを再起動して、変更内容を適用します。

7.2.4 マルチパスデバイスでのmdadmの使用

mdadmツールでは、デバイスのノードパスではなく、IDでデバイスにアクセスする必要があります。したがって、/etc/mdadm.conf内のDEVICEエントリを次のように設定してください。

```
DEVICE /dev/disk/by-id/*
```

ユーザフレンドリな名前を使用している場合は、次のようにパスを指定し、マルチパス処理の設定後に、デバイスマッパー名だけがスキャンされるようにします。

```
DEVICE /dev/disk/by-id/dm-uuid-.*-mpath-.*
```

7.2.5 マルチパスデバイスでの--noflushの使用

マルチパスデバイス上で実行する場合は、オプション--noflushを必ず使用する必要があります。

たとえば、テーブルのリロードを行うスクリプトでは、マルチパストポロジ情報が必要なので、再開時に--noflushオプションを使用して、残っているI/Oがフラッシュされないようにします。

```
load
resume --noflush
```

7.2.6 ルートデバイスがマルチパスの場合のSANタイムアウト設定

マルチパスデバイスにルート(/)があるシステムは、すべてのパスに障害が発生し、それらのパスがシステムから削除されると、停止することがあります。

これは、ストレージサブシステム(ファイバチャネルストレージアレイなど)からdev_loss_tmoタイムアウトを受信するからです。

システムデバイスがマルチパスを使用して設定され、マルチパスのno_path_retry設定がアクティブな場合は、ストレージサブシステムのdev_loss_tmo設定を適宜変更して、すべてのパスがダウンするシナリオでデバイスが削除されないようにする必要があります。dev_loss_tmo値をマルチパスのno_path_retry設定以上にすることを強くお勧めします。

ストレージサブシステムのdev_los_tmoの推奨設定は、次のとおりです。

```
<dev_loss_tmo> = <no_path_retry> * <polling_interval>
```

マルチパス値については、次の定義が適用されます。

- no_path_retryは、パスが失われたとみなされて入出力のキューイングが停止されるまでのマルチパス入出力の再試行数です。
- polling_intervalは、パッチチェックの間隔(秒単位)です。

これらの各マルチパス値は、/etc/multipath.conf環境設定ファイルから設定する必要があります。詳細については、7.4.5項 「/etc/multipath.confファイルの作成と設定」 (83 ページ)を参照してください。

7.2.7 マルチパスデバイスのパーティショニング

アップグレード中は、マルチパスデバイスの処理方法の動作変更によって、設定に影響を及ぼすことがあります。

- 「SUSE Linux Enterprise Server 11」 (66 ページ)
- 「SUSE Linux Enterprise Server 10」 (66 ページ)
- 「SUSE Linux Enterprise Server 9」 (66 ページ)

SUSE Linux Enterprise Server 11

SUSE Linux Enterprise Server 11では、デフォルトのマルチパスセットアップはudevを使用して、マルチパス処理の開始時に/dev/disk/by-idディレクトリ内の既存のシンボリックリンクを上書きします。マルチパス処理開始前のシンボリックリンクは、SCSIデバイスをそのscsi-xxx名でポイントします。マルチパス処理実行中のシンボリックリンクは、SCSIデバイスをそのdm-uuid-xxx名でポイントします。これによって、マルチパス処理が開始したかどうかに関わりなく、/dev/disk/by-idパス内のシンボリックリンクは、継続的に同じデバイスをポイントします。自動的に正しいデバイスを指すので、設定ファイル(lvm.confやmd.conf)を変更する必要はありません。

SUSE Linux Enterprise Server 10

SUSE Linux Enterprise Server 10では、kpartxソフトウェアを/etc/init.d/boot.multipathで使用することで、再起動なしで、任意の新しく作成されたパーティションの環境設定ファイルmultipath.confの/dev/dm-*行にシンボリックリンクが追加されます。これにより、udevdがトリガされて、/dev/disk/by-*にシンボリックリンクを書き込みます。主なメリットは、サーバの再起動なしで、新しいパラメータでkpartxを呼び出せることです。

SUSE Linux Enterprise Server 9

SUSE Linux Enterprise Server 9では、マルチパスI/Oデバイス自体をパーティショニングすることはできません。ベースになる物理デバイスが既にパーティショニングされている場合、マルチパスI/Oデバイスはそれらのパーティションを反映し、レイヤが/dev/disk/by-id/<name>p1 ... pNデバイスを提供するので、マルチパスI/Oレイヤを介してパーティションにアクセスできます。結果として、マルチパスI/Oの有効化の前にデバイスをパーティション分割する必要があります。実行中のシステムのパーティション分割を変更した場合、これらの変更は、DM-MPIOでは自動的な検出および反映はされません。デバイスは、再初期化する必要があり、その際には、通常、再起動が必要です。

7.2.8 マルチパスI/O用にサポートされているアーキテクチャ

マルチパス処理のドライバおよびツールは、サポートされている7つのプロセッサアーキテクチャ(IA32、AMD64/EM64T、IPF/IA64、32ビットと64ビットのp-Series、31ビットと64ビットのz-Series)をすべてサポートします。

7.2.9 マルチパス処理用にサポートされているストレージアレイ

マルチパス処理のドライバおよびツールは、大半のストレージアレイをサポートします。マルチパスデバイスを格納するストレージアレイは、マルチパス処理用のドライバとツールを使用するために、マルチパス処理をサポートする必要があります。一部のストレージアレイベンダは、独自のマルチパス処理管理ツールを提供しています。ベンダのハードウェアマニュアルを参照して、どのような設定が必要か判別してください。

- 「マルチパス処理用に自動検出されるストレージアレイ」 (67 ページ)
- 「マルチパス処理サポートについてテスト済みのストレージアレイ」 (69 ページ)
- 「特定のハードウェアハンドラを必要とするストレージアレイ」 (69 ページ)

マルチパス処理用に自動検出されるストレージアレイ

`multipath-tools`パッケージは、次のようなストレージアレイを自動的に検出します。

3PARdata VV
Compaq* HSV110
Compaq MSA1000
DDN SAN MultiDirector
DEC* HSG80
EMC* CLARiiON* CX
EMC Symmetrix*

FSC CentricStor*
Hewlett Packard* (HP*) A6189A
HP HSV110
HP HSV210
HP Open
Hitachi* DF400
Hitachi DF500
Hitachi DF600
IBM* 3542
IBM ProFibre 4000R
NetApp*
SGI* TP9100
SGI TP9300
SGI TP9400
SGI TP9500
STK OPENstorage DS280
Sun* StorEdge 3510
Sun T4

ただし、他のほとんどのストレージアレイも有効です。ストレージアレイが自動的に検出されると、マルチパス処理のデフォルト設定が適用されます。デフォルト以外の設定を使用したい場合は、手動で`/etc/multipath.conf`ファイルを作成および設定する必要があります。詳細については、7.4.5項「/etc/multipath.confファイルの作成と設定」 (83 ページ)を参照してください。

IBM zSeries*デバイスをマルチパス処理でテストした結果、dev_loss_tmoパラメータを90秒に、fast_io_fail_tmoパラメータを5秒に設定する必要があることわかりました。zSeriesデバイスをご使用の場合は、手動で`/etc/multipath.conf`ファイルの作成と設定を行い、値を指定する必要があります。詳細については、「/etc/multipath.conf内のzSeriesのデフォルト設定」 (89 ページ)を参照してください。

自動検出されないハードウェアについては、`/etc/multipath.conf`ファイルの`DEVICES`セクションに環境設定の適切なエントリが必要です。この場合は、手動で、環境設定ファイルを作成し、設定する必要があります。詳細については、7.4.5項 「/etc/multipath.confファイルの作成と設定」 (83 ページ)を参照してください。

次の点に留意してください。

- 自動検出されたすべてのストレージアレイがSUSE Linux Enterprise Serverでテスト済みというわけではありません。詳細については、「マルチパス処理サポートについてテスト済みのストレージアレイ」 (69 ページ)を参照してください。

- 一部のストレージアレイでは、特定のハードウェアハンドラが必要なことがあります。ハードウェアハンドラは、パスグループの切り替え時とI/Oエラーの処理時に、ハードウェア固有のアクションを実行するカーネルモジュールです。詳細については、「特定のハードウェアハンドラを必要とするストレージアレイ」 (69 ページ)を参照してください。

- `/etc/multipath.conf`ファイルの変更後、`mkinitrd`を実行してシステム上にINITRDを作成し、次に、再起動で変更内容を有効にする必要があります。

マルチパス処理サポートについてテスト済みのストレージアレイ

次のストレージアレイは、SUSE Linux Enterprise Serverでテスト済みです:

EMC
Hitachi
Hewlett-Packard/Compaq
IBM
NetApp
SGI

他のベンダのストレージアレイもほとんど機能するはずです。該当するベンダのマニュアルを参照してください。`multipath-tools`パッケージによって認識されるデフォルトストレージアレイのリストについては、「「マルチパス処理用に自動検出されるストレージアレイ」 (67 ページ)」を参照してください。

特定のハードウェアハンドラを必要とするストレージアレイ

あるパスから他のパスにフェイルオーバーするには特別なコマンドが必要なストレージアレイ、または非標準の特別なエラー処理が必要なストレージア

レイには、より拡張されたサポートが必要なことがあります。したがって、デバイスマッパーマルチパスサービスには、ハードウェアハンドラ用フックがあります。たとえば、そのようなEMC CLARiiON CXファミリアレイ用ハンドラが1つ、既に提供されています。

重要項目

ハードウェアベンダのマニュアルを参照して、そのハードウェアハンドラをデバイスマッパーマルチパス用にインストールする必要があるかどうか判別してください。

`multipath -t`コマンドは、特定のハードウェアハンドラで特別な処理を必要とするストレージアレイの内部テーブルを表示します。ただし、表示されるリストは、サポートされているアレイの包括的なリストではありません。特別な処理を必要とし、`multipath-tools`の開発者がツールの開発中にアクセスしたアレイだけがリストされます。

重要項目

真のアクティブ/アクティブマルチパスサポートを持つアレイは、特別な処理を必要としないので、`multipath -t`コマンドでは表示されません。

また、`multipath -t`テーブルでリストされている場合でも、必ずしも、その特定ハードウェアでSUSE Linux Enterprise Serverがテスト済みということではありません。テスト済みのストレージアレイのリストについては、「「マルチパス処理サポートについてテスト済みのストレージアレイ」 (69ページ)」を参照してください。

7.3 マルチパス管理ツール

SUSE Linux Enterprise Server 10以降のマルチパス処理サポートは、Linux 2.6カーネルのデバイスマッパーマルチパスモジュールと`multipath-tools`ユーザスペースパッケージに基づいています。MDADM (Multiple Devices Administration)ユーティリティ(`mdadm`)を使用すると、マルチパスデバイスの状態を表示できます。

- 7.3.1項 「デバイスマッパーマルチパスモジュール」 (71 ページ)

- 7.3.2項 「マルチパスI/O管理ツール」 (74 ページ)
- 7.3.3項 「マルチパスデバイスへのMDADMの使用」 (76 ページ)
- 7.3.4項 「Linux multipath(8)コマンド」 (77 ページ)

7.3.1 デバイスマッパーマルチパスモジュール

デバイスマッパーマルチパス(DM-MP)モジュールは、Linuxにマルチパス処理機能を提供します。DM-MPIOは、SUSE Linux Enterprise Server 11でのマルチパス処理の推奨ソリューションです。DM-MPは、Enterprise Server 11に標準装備される唯一のマルチパス処理オプションであり、Novell®およびSUSEによって完全にサポートされています。

DM-MPIOは、多様なセットアップでマルチパス処理サブシステムを自動設定します。デバイスごとに最大8個のパスの設定がサポートされています。アクティブ/パッシブ(1つのパスがアクティブで、他のパスがパッシブ)またはアクティブ/アクティブ(ラウンドロビン方式の負荷分散で全パスがアクティブ)の構成がサポートされています。

DM-MPIOフレームワークは、2つの方法で拡張できます。

- 特定のハードウェアハンドラの使用詳細については、「特定のハードウェアハンドラを必要とするストレージアレイ」 (69 ページ)を参照してください。
- ラウンドロビンアルゴリズムより高度な負荷分散アルゴリズムの使用

DM-MPIOのユーザスペースコンポーネントにより、自動的なパスの検出とグループ化のほか、自動的なパスの再テストが実行されるので、障害が発生したパスは、正常に戻ると、自動的に復帰します。これにより、管理者の手間を最低限に抑えることができます。

DM-MPIOは、デバイス自体の障害ではなく、デバイスへのパスの障害からシステムを保護します。アクティブなパスの1つが失われると(たとえば、ネットワークアダプタが破損する、光ファイバケーブルが外れるなど)、残りのパスにI/Oをリダイレクトします。アクティブ/パッシブ構成の場合は、パスがパッシブパスの1つにフェールオーバーします。ラウンドロビン式負荷分散構

成を使用している場合は、トラフィックの負荷が残りの正常なパス全体に分散されます。すべてのアクティブパスに障害が起きた場合は、アクティブでないセカンダリパスが有効になり、約30秒の遅延でフェールオーバーが開始されます。

ディスクアレイに複数のストレージプロセッサがある場合は、アクセスしたいLUNを所有するストレージプロセッサにSANスイッチが接続していることを確認してください。ほとんどのディスクアレイでは、すべてのLUNが両方のストレージプロセッサに属しているので、両方の接続がアクティブです。

注記

一部のディスクアレイでは、ストレージアレイがストレージプロセッサを介してトラフィックを管理するので、一度に1つのストレージプロセッサだけが提示されます。1つのプロセッサがアクティブとなり、もう1つのプロセッサは障害が発生するまでパッシブとなります。間違ったストレージプロセッサ(パッシブなパスを持つプロセッサ)に接続している場合は、予期されたLUNが表示されなかったり、それらのLUNが表示されてもアクセスしようとするとエラーが発生することがあります。

*表 **7.1** ストレージアレイのマルチパスI/O機能*

ストレージアレイの機能	説明
アクティブ/パッシブコントローラ	1つのコントローラはアクティブで、すべてのLUNに対応します。2つ目のコントローラは、スタンバイとして機能します。2つ目のコントローラは、オペレーティングシステムが冗長なパスを認識するように、マルチパスコンポーネントに対するLUNの提示も行います。プライマリコントローラに障害が発生した場合は、セカンダリコントローラが引き継ぎ、すべてのLUNに対応します。 一部のアレイでは、LUNをさまざまなコントローラに割り当てることができます。所定のLUNは、そのアクティブコントローラとなる1つのコントローラに割り当てられます。一度に、1つのコントローラが所定のLUNのディスクI/Oを行い、2つ目のコントローラ

ストレージアレイの機能	説明
	がそのLUNのスタンバイコントローラとなります。2つ目のコントローラは、パスの提示もしますが、ディスクI/Oは行えません。そのLUNを使用するサーバは、LUNの割り当て先のコントローラに接続します。LUNのセットに対するプライマリコントローラに障害が発生すると、セカンダリコントローラが引き継ぎ、すべてのLUNに対応します。
アクティブ/パッシブコントローラ	両方のコントローラがすべてのLUNの負荷を共有し、任意の所定のLUNのディスクI/Oを処理できます。1つのコントローラに障害が発生すると、2つ目のコントローラが自動的にすべてのトラフィックを処理します。
負荷分散	デバイスマッパーマルチパスドライバは、自動的に、すべてのアクティブパス全体にトラフィックの負荷を分散します。
コントローラのフェールオーバー	アクティブなコントローラがパッシブなコントローラにフェールオーバーすると、デバイスマッパーマルチパスドライバがホスト/スタンバイ間のパスを自動的に有効にし、それらをプライマリパスにします。
ブート/ルートデバイスのサポート	マルチパス処理は、SUSE Linux Enterprise Server 10以降のルート(/)デバイスに対してサポートされます。ホストサーバは、ブートデバイス用の、現在アクティブなコントローラおよびストレージプロセッサに接続する必要があります。 マルチパス処理は、SUSE Linux Enterprise Server 11以降の/bootデバイスに対してサポートされています。

デバイスマッパーマルチパスは、マルチパスデバイスの各パスを個別のSCSIデバイスとして検出します。SCSIデバイス名は、/dev/sd*N*の形式をとります。ここで、*N*は、デバイスに対して自動生成される文字であり、aで始まり、デバイスの生成に応じてシーケンシャルに発行されます(/dev/sda、/dev/

sdbなど)。デバイス数が26を超えると、文字が2つ使用され、/dev/sdzの次のデバイスは/dev/sdaa、その次は、その次は/dev/sdabと続きます。

複数のパスが自動的に検出されない場合は、それらを/etc/multipath.confファイルで手動設定できます。multipath.confファイルは、システム管理者によって作成および設定されるまで存在しません。詳細については、7.4.5項 「/etc/multipath.confファイルの作成と設定」 (83 ページ)を参照してください。

7.3.2 マルチパスI/O管理ツール

multipath-toolsユーザスペースパッケージは、自動的にパスを検出し、グループ化します。このパッケージは、自動的にパスの定期テストを行うので、障害が発生したパスは、正常に戻ると、自動的に復帰します。これにより、管理者の手間を最低限に抑えることができます。

表 7.2　multipath-toolsパッケージに含まれるツール

ツール	説明
multipath	システムをスキャンしてマルチパスデバイスを検出し、アセンブルします。
multipathd	mapsイベントを待機し、multipathを実行します。
devmap-name	デバイスマップ(devmap)のためのudevに意味のあるデバイス名を与えます。
kpartx	マルチパスデバイス上のパーティションにリニアdevmapをマップします。これにより、デバイス上のパーティションのマルチパスモニタリングを作成することが可能になります。

パッケージのファイルリストは、サーバのアーキテクチャによって異なることがあります。multipath-toolsパッケージに含まれたファイルのリストについては、「*SUSE Linux Enterprise Server Technical Specifications*」>「*Package Descriptions*」 Webページ [http://www.novell.com/products/server/techspecs.html]にアクセスし、目的のアーキテクチャを見つけ、 [パッ

ケージを*名前*でソート］を選択し、次に、「multipath-tools」で検索して、そのアーキテクチャのパッケージリストを見つけます。

コマンドオプション`rpm -ql`または`rpm -qpl`を使用してパッケージ自体をクエリすることによって、RPMファイルのファイルリストを判別することもできます。

- インストールしたパッケージをクエリするには、次のコマンドを入力します。

  ```
  rpm -ql <package_name>
  ```

- インストールしていないパッケージをクエリするには、次のコマンドを入力します。

  ```
  rpm -qpl <URL_or_path_to_package>
  ```

`multipath-tools`パッケージがインストールされているかどうか確認するには、次の操作を行います。

- 端末コンソールのプロンプトで、次のように入力します。

  ```
  rpm -q multipath-tools
  ```

  パッケージがインストールされている場合は、パッケージ名を繰り返し、バージョン情報を示す応答メッセージが表示されます。

  ```
  multipath-tools-04.7-34.23
  ```

  パッケージがインストールされていない場合は、次の応答メッセージが表示されます。

  ```
  package multipath-tools is not installed
  ```

7.3.3 マルチパスデバイスへのMDADMの使用

デフォルトのデバイスハンドラはUdevであり、デバイスは、デバイスノード名ではなく、Worldwide IDによって、システムに自動的に認識されます。これによって、環境設定ファイル(mdadm.confとlvm.conf)がマルチパスデバイスを正しく認識しないという、MDADMおよびLVMの旧リリースにあった問題が解決します。

LVM2の場合のようにmdadmでは、デバイスノードパスではなく、IDによってデバイスをアクセスする必要があります。したがって、/etc/mdadm.conf内のDEVICEエントリを次のように設定してください。

```
DEVICE /dev/disk/by-id/*
```

ユーザフレンドリな名前を使用している場合は、次のようにパスを指定し、マルチパス処理の設定後に、デバイスマッパー名だけがスキャンされるようにします。

```
DEVICE /dev/disk/by-id/dm-uuid-.*-mpath-.*
```

MDADMのインストールを確認するには、次を行います。

- 端末コンソールのプロンプトで、次のコマンドを入力し、mdadmパッケージがインストールされているかどうか確認します。

  ```
  rpm -q mdadm
  ```

  パッケージがインストールされている場合は、パッケージ名を繰り返し、バージョン情報を示す応答メッセージが表示されます。次に例を示します。

  ```
  mdadm-2.6-0.11
  ```

  パッケージがインストールされていない場合は、次の応答メッセージが表示されます。

  ```
  package mdadm is not installed
  ```

/etc/lvm/lvm.confファイルの変更方法については、「7.2.3項 「マルチパスデバイスでのLVM2の使用」 (62 ページ)」を参照してください。

7.3.4 Linux multipath(8)コマンド

マルチパスデバイスを設定し、管理するには、multipath(8)コマンドを使用します。

multipath(8)コマンドの一般構文:

```
multipath [-v verbosity] [-d] [-h|-l|-ll|-f|-F] [-p failover | multibus |
group_by_serial | group_by_prio| group_by_node_name ]
```

一般的な例

multipath
　すべてのマルチパスデバイスを設定します。

multipath *devicename*
　特定のマルチパスデバイスの設定.

　*devicename*を、/dev/sdb (udevにより$DEVNAME変数で表示)またはmajor:minor形式などのデバイスノード名で置き換えます。

multipath -f
　マルチパスマップとそのデバイスにマップされたパーティションを選択的に抑制します。

multipath -d
　可能なマルチパスデバイスを表示しますが、デバイスの作成やデバイスマップの更新は行いません(ドライラン)。

multipath -v2 -d
　マルチパスデバイスを設定し、それらのマルチパスマップ情報を表示します。multipath -v2 -dの-v2オプションは、ローカルディスクだけを表示します。

multipath -v2 *devicename*
: 特定のマルチパスデバイスを設定し、そのマルチパスマップ情報を表示します。-v2オプションを使用すると、ローカルディスクのみが表示されます。

multipath -v3 -d
: マルチパスデバイスを設定し、それらのマルチパスマップ情報を表示します。-v3オプションを使用すると、フルパスリストが表示されます。

multipath -v3 *devicename*
: 特定のマルチパスデバイスを設定し、そのマルチパスマップ情報を表示します。-v3オプションを使用すると、フルパスリストが表示されます。

multipath -ll
: すべてのマルチパスデバイスの状態を表示します。

multipath -ll *devicename*
: 指定されたマルチパスデバイスの状態を表示します。

multipath -F
: 未使用のマルチパスデバイスのマップをすべてフラッシュします。これによって、マルチパスが解消したり、デバイスが削除されることはありません。

multipath -F *devicename*
: 特定のマルチパスデバイスの未使用マルチパスデバイスマップをフラッシュします。これによって、マルチパスが解消したり、デバイスが削除されることはありません。

multipath -p [failover | multibus | group_by_serial | group_by_prio | group_by_node_name]
: 表7.3「multipath -pコマンドのグループポリシーオプション」 (79 ページ)に示されているグループポリシーオプションの1つを指定して、グループポリシーを設定します。

*表 **7.3** multipath -pコマンドのグループポリシーオプション*

ポリシーオプション	説明
failover (フェールオーバー)	優先度グループごとに1つのパス一度に使用できるスパスは1つだけです。
multibus	1つの優先度グループ内にすべてのパス
group_by_serial	検出されたSCSIシリアル番号(コントローラノードの全世界規模の番号)ごとに1つの優先度グループ
group_by_prio	パス優先度値ごとに1つの優先度グループ同じ優先度のパスは同じ優先度グループに属します。優先度は、コールアウトプログラムで決定されます。それらのプログラムは、グローバル、コントローラごと、またはマルチパスごとのオプションとして/etc/multipath.conf環境設定ファイルで指定されます。
group_by_node_name	ターゲットノード名ごとに1つの優先度グループターゲットノード名は、/sys/class/fc_transport/target*/node_nameにフェッチされます。

7.4 マルチパス処理用システムの設定

- 7.4.1項 「マルチパス処理用SANデバイスの準備」 (80 ページ)
- 7.4.2項 「マルチパスデバイスのパーティショニング」 (81 ページ)
- 7.4.3項 「マルチパス処理のためのサーバ設定」 (82 ページ)
- 7.4.4項 「ブートシーケンスへのmultipathdの追加」 (82 ページ)
- 7.4.5項 「/etc/multipath.confファイルの作成と設定」 (83 ページ)

7.4.1 マルチパス処理用SANデバイスの準備

SANデバイスのマルチパスI/Oを設定する前に、必要に応じて、次のようにSANデバイスを準備してください。

- ベンダのツールで、SANデバイスを設定し、ゾーン化します。
- ベンダのツールで、ストレージアレイ上のホストLUNのパーミッションを設定します。
- Linux HBAドライバモジュールをインストールします。モジュールがインストールされると、ドライバがHBAを自動的にスキャンして、ホスト用のパーミッションを持つSANデバイスを検出します。それらのSANデバイスは、以降の設定のため、ホストに提示されます。

  注記

  ご使用のHBAドライバのネイティブマルチパス処理が有効化していないことを確認してください。

  詳細については、ベンダの特定マニュアルを参照してください。
- ドライバモジュールがロードされたら、特定アレイのLUNまたはパーティションに割り当てられたデバイスノードを検出します。
- SANデバイスがサーバ上でルートデバイスとして使用される場合は、7.2.6項 「ルートデバイスがマルチパスの場合のSANタイムアウト設定」(64ページ)に示されているように、デバイスのタイムアウト設定を変更します。

HBAドライバがLUNを認識しない場合は、`lsscsi`を使用して、SCSIデバイスがオペレーティングシステムによって正しく認識されているかどうかチェックできます。LUNがHBAドライバによって認識されない場合は、SANのゾーン化セットアップをチェックします。特に、LUNのマスキングがアクティブであるかどうか、LUNがサーバに正しく割り当てられているかどうかをチェックしてください。

LUNがHBAドライバによって認識されても、対応するブロックデバイスが存在しない場合は、カーネルパラメータを追加して、SCSIデバイスのスキャン動作を変更する必要があります(LUNが連続的に番号付けされていないことを

示すなど)。詳細については、Novell Support Knowledgebaseの「*Options for SCSI Device Scanning* [http://support.novell.com/techcenter/sdb/en/2005/06/drahn_scsi_scanning.html]」を参照してください。

7.4.2 マルチパスデバイスのパーティショニング

複数のパスを持つパーティショニングデバイスは、推奨できませんが、サポートされています。

- 「SUSE Linux Enterprise Server 10」 (81 ページ)
- 「SUSE Linux Enterprise Server 9」 (81 ページ)

SUSE Linux Enterprise Server 10

SUSE Linux Enterprise Server 10では、`kpartx`ツールで、再起動なしで、マルチパスデバイス上にパーティションを作成できます。マルチパス処理の設定前に、YaST2のパーティショナ機能またはサードパーティーのパーティショニングツールの使用により、デバイスをパーティショニングすることもできます。

SUSE Linux Enterprise Server 9

SUSE Linux Enterprise Server 9では、デバイスをパーティショニングしたい場合、マルチパス処理の設定前に、YaST2のパーティショナ機能またはサードパーティーのパーティショニングツールの使用により、デバイスをパーティショニングする必要があります。これが必要なのは、既存のマルチパスデバイスのパーティショニングがサポートされていないからです。マルチパスデバイス上のパーティショニングを試行すると、失敗します。

デバイスにパーティションを設定する場合は、DM-MPIOがそれらのパーティションを自動的に認識し、次のように、p1からp*n*をデバイスのIDに付加することで該当のデバイスを表示します。

`/dev/disk/by-id/26353900f02796769p1`

マルチパスデバイスをパーティション分割するには、DM-MPIOサービスを無効にし、通常のデバイスノード(/dev/sdcなど)をパーティション分割し、最後に、再起動してDM-MPIOサービスが新しいパーティションを認識できるようにします。

7.4.3 マルチパス処理のためのサーバ設定

INITRDで、マルチパスI/Oデバイスの接続先コントローラのデバイスドライバを自動的にロードするには、システムを手動で設定する必要があります。それには、必要なドライバモジュールを変数INITRD_MODULES(/etc/sysconfig/kernelファイル内)に追加する必要があります。

たとえばシステムにccissドライバからアクセスされるRAIDコントローラと、ドライバqla2xxxからアクセスされるQLogic*コントローラに接続されたマルチパスデバイスが含まれている場合は、次のような入力になります。

```
INITRD_MODULES="cciss"
```

QLogicドライバは、起動時に自動的にはロードされないので、ここで追加します。

```
INITRD_MODULES="cciss qla23xx"
```

/etc/sysconfig/kernelの変更後は、システム上でINITRDを再作成し(mkinitrdコマンド使用)、次に、再起動して変更内容を有効にする必要があります。

ブートマネージャとしてLILOを使う場合は、/sbin/liloコマンドでLILOを再インストールします。GRUBを使っている場合は、何も操作する必要はありません。

7.4.4 ブートシーケンスへのmultipathdの追加

このセクションに示すどちらかの方法を使用して、マルチパスI/Oサービス(multipathd)をブートシーケンスに追加します。

- 「YaSTを使用してmultipathdを追加する」 (83 ページ)
- 「コマンドラインを使用してmultipathdを追加する」 (83 ページ)

YaSTを使用してmultipathdを追加する

1 YaST®内で、［システム］>［システムサービス(ランレベル)］>［簡易モード］の順にクリックします。

2 ［multipathd］を選択してから、［Enable(有効)］をクリックします。

3 ［OK］をクリックして、サービスの開始メッセージを確認します。

4 ［終了］をクリックしてから、［はい］をクリックします。

変更内容は、サーバが再起動するまで有効にはなりません。

コマンドラインを使用してmultipathdを追加する

1 端末コンソールを開いて、rootユーザまたは同等の権限でログインします。

2 端末コンソールのプロンプトで、次のように入力します。

```
insserv multipathd
```

7.4.5 /etc/multipath.confファイルの作成と設定

/etc/multipath.confファイルは、作成しない限り、存在しません。/usr/share/doc/packages/multipath-tools/multipath.conf.syntheticファイルには、マルチパス設定の指針となるサンプルファイル /etc/multipath.confが含まれています。属性とそれらのオプションに関する詳しいコメントの付いたテンプレートについては、/usr/share/doc/packages/multipath-tools/multipath.conf.annotatedを参照してください。

- 「multipath.confファイルの作成」 (84 ページ)
- 「etc/multipath.confファイルのセットアップの確認」 (85 ページ)
- 「/etc/multipath.confでのユーザフレンドリな名前またはエイリアス名の設定」 (86 ページ)
- 「/etc/multipath.confでの非マルチパスデバイスのブラックリスト化」 (87 ページ)
- 「/etc/multipath.confでのデフォルトマルチパス動作の設定」 (88 ページ)
- 「/etc/multipath.conf内のzSeriesのデフォルト設定」 (89 ページ)
- 「/etc/multipath.confファイルの変更の適用」 (90 ページ)

multipath.confファイルの作成

/etc/multipath.confファイルが存在しない場合は、次のように、例をコピーしてファイルを作成していください。

1 端末コンソールで、rootユーザとしてログインします。

2 次のコマンドを入力して(すべて1行で)、テンプレートをコピーします。

```
cp /usr/share/doc/packages/multipath-tools/multipath.conf.synthetic
/etc/multipath.conf
```

3 /usr/share/doc/packages/multipath-tools/multipath.conf .annotatedファイルを、システムのマルチパス処理の設定方法を決定する参考として使用します。

4 SAN用の適切なdeviceエントリがあることを確認します。大半のベンダは、deviceセクションの正しいセットアップに関するマニュアルを提供しています。

/etc/multipath.confファイルでは、異なるSANには異なるdeviceセクションが必要です。自動的に検出されるストレージサブシステムを使用する場合(「「マルチパス処理サポートについてテスト済みのストレージアレイ」 (69 ページ)」参照)、そのデバイスのデフォルトエント

リを使用できます。/etc/multipath.confファイルをさらに設定する必要はありません。

5 ファイルを保存します。

etc/multipath.confファイルのセットアップの確認

設定後、次のコマンドを入力して、「ドライラン」を実行できます。

```
multipath -v3 -d
```

このコマンドは、デバイスをスキャンし、次に、セットアップの内容を表示します。出力の例を以下に示します。

```
26353900f02796769
[size=127 GB]
[features="0"]
[hwhandler="1    emc"]

\_ round-robin 0 [first]
  \_ 1:0:1:2 sdav 66:240  [ready ]
  \_ 0:0:1:2 sdr  65:16   [ready ]

\_ round-robin 0
  \_ 1:0:0:2 sdag 66:0    [ready ]
  \_ 0:0:0:2 sdc   8:32   [ready ]
```

パスは、優先度グループでグループ化されます。一度に1つの優先度グループだけがアクティブに使用されます。アクティブ/アクティブ構成をモデル化するには、すべてのパスを同じグループにします。アクティブ/パッシブ構成をモデル化する場合は、並行してアクティブにしないパスを複数の別の優先度グループに振り分けます。これは、通常、デバイス検出時に自動的に行われます。

出力として、順序、グループ内でのI/O負荷の分散に使用されるスケジュールポリシー、および各優先度グループのパスが表示されます。また、各パスに対して、その物理アドレス(ホスト:バス:ターゲット:LUN)、デバイスノード名、メジャー:マイナー番号、および状態が表示されます。

/etc/multipath.confでのユーザフレンドリな名前またはエイリアス名の設定

マルチパスデバイスは、そのWWID (Worldwide Identifier)か、またはそれに割り当てられたエイリアスのどちらかで識別できます。WWIDは、グローバルに一意であり、変わらないことを保証されたマルチパスデバイスの識別子です。マルチパス処理で使用されるデフォルト名は、/dev/disk/by-idディレクトリにある論理ユニットのIDです。/dev/sdn形式と/dev/dm-n形式のデバイスノード名は、再起動時に変更される可能性があるので、マルチパスデバイスは、IDで参照することをお勧めします。

/dev/mapperディレクトリにあるマルチパスデバイス名は、LUNのIDを参照し、/var/lib/multipath/bindingsファイルを使用して関連付けを追跡するので、常に、一貫性を保ちます。これらのデバイスには、ユーザフレンドリな名前が付いています(/dev/disk/by-id/dm-uuid-.*-mpath-.*など)。

/etc/multipath.confファイルでALIASディレクティブを使用すると、独自のデバイス名を指定できます。別名は、IDと/dev/disk/by-id/dm-uuid-.*-mpath-の使用を無効にします。* 名前.

重要項目

ルートデバイスにはエイリアスを使わないことを推奨します。デバイス名が異なると、カーネルコマンドラインでマルチパス処理をシームレスにオフにする機能が失われるからです。

multipath.conf設定の例については、/usr/share/doc/packages/multipath-tools/multipath.conf.syntheticファイルを参照してください。

1. 端末コンソールで、rootユーザとしてログインします。
2. /etc/multipath.confファイルをテキストエディタで開きます。
3. Defaultsディレクティブとその閉じ括弧を非コメント化します。

4 user_friendly_namesオプションを非コメント化し、次に、その値をNoからYesに変更します。

次に例を示します。

```
## Use user friendly names, instead of using WWIDs as names.
defaults {
  user_friendly_names yes
}
```

5 オプションとして、aliasディレクティブ(multipathセクションにある)を使用して、独自にユーザフレンドリなデバイス名を指定できます。

次に例を示します。

```
multipath {
  wwid 26353900f02796769
  alias sdd4l0
}
```

6 変更内容を保存し、ファイルを閉じます。

/etc/multipath.confでの非マルチパスデバイスのブラックリスト化

/etc/multipath.confファイルには、blacklistセクションがあり、このセクションにすべての非マルチパスデバイスがリストされます。たとえば、ローカルIDEハードドライブおよびフロッピードライブは、通常、マルチパス化されません。multipathが管理対象とするシングルパスデバイスがあり、multipathにそれらを無視させたい場合は、それらのシングルパスデバイスをblacklistセクションに入力して、問題を解決します。

注記

キーワードdevnode_blacklistは廃止され、キーワードblacklistに代わりました。

たとえば、ccissドライバからローカルデバイスとすべてのアレイを、multipathによる管理から外してブラックリストに載せるには、blacklistセクションを次のように指定します。

```
blacklist {
      wwid 26353900f02796769
      devnode "^(ram|raw|loop|fd|md|dm-|sr|scd|st|sda)[0-9]*"
      devnode "^hd[a-z][0-9]*"
      devnode "^cciss!c[0-9]d[0-9].*"
}
```

アレイ全体でなく、ドライバからのパーティションだけをブラックリスト化することもできます。たとえば、次の正規表現を使用すると、アレイ全体ではなく、cciss ドライバからのパーティションだけがブラックリスト化されます。

```
^cciss!c[0-9]d[0-9]*[p[0-9]*]
```

/etc/multipath.confファイルの変更後、mkinitrdを実行してシステム上にINITRDを作成し、次に、再起動で変更内容を有効にします。

その後は、multipath -llコマンドの発行時に、ローカルデバイスがマルチパスマップに一覧されなくなります。

/etc/multipath.confでのデフォルトマルチパス動作の設定

/etc/multipath.confファイルには、デフォルト動作を指定できるdefaultsセクションが含まれています。deviceセクションで、フィールドが別途指定されていない場合は、そのSAN構成にデフォルト設定が適用されます。

次に示すdefaultsセクションでは、シンプルなフェールオーバーポリシーが指定されています。

```
defaults {
       multipath_tool  "/sbin/multipath -v0"
       udev_dir        /dev
       polling_interval 10
       default_selector        "round-robin 0"
       default_path_grouping_policy    failover
```

```
        default_getuid  "/sbin/scsi_id -g -u -s /block/%n"
        default_prio_callout    "/bin/true"
        default_features        "0"
        rr_min_io               100
        failback                immediate
```

注記

default_getuidコマンドラインでは、サンプルファイル/usr/share/doc/packages/multipath-tools/multipath.conf.synthetic内(およびデフォルトの注釈付きサンプルファイルにも含まれている)の/lib/udev/scsi_idというサンプルパスの代わりに、上記の例にあるように、パス/sbin/scsi_idを使用します。SLES 11 SP1以上の場合は、サンプルファイルに正しいパスである/sbin/scsi_idが含まれているはずです。

/etc/multipath.conf内のzSeriesのデフォルト設定

IBM zSeriesデバイスをマルチパス処理でテストした結果、dev_loss_tmoパラメータを90秒に、fast_io_fail_tmoパラメータを5秒に設定する必要があることわかりました。zSeriesデバイスをご使用の場合は、/etc/multipath.confファイルを変更して、値を次のように指定します。

```
defaults {
        dev_loss_tmo 90
        fast_io_fail_tmo 5
}
```

dev_loss_tmoパラメータは、マルチパスリンクに不良のマーキングがされるまでの秒数を設定します。パスに障害が発生したら、そのパスの現在のI/Oが失敗します。デフォルト値は使用するデバイスドライバによって異なります。0～600秒の値の範囲が有効です。ドライバの内部タイムアウトを使用するには、ゼロ(0)または600より大きい値に設定します。

fast_io_fail_tmoパラメータは、リンク障害を検出した場合に、I/Oが失敗するまでの待機時間を設定します。ドライバに到達したI/Oは失敗します。ブロックしたキューにI/Oがある場合は、I/Oはdev_loss_tmoで指定された時間が経過するまでは失敗せず、キューのブロックが解除されます。

/etc/multipath.confファイルの変更の適用

/etc/multipath.confファイルに対する変更は、multipathdの実行中は有効になりません。変更を行ったら、ファイルを保存して閉じ、次のように、変更内容を適用してください。

1. multipathdサービスを停止します。
2. 次のコマンドの入力で、古いmultipathバインディングをクリアします。

   ```
   /sbin/multipath -F
   ```

3. 次のコマンドの入力で、新しいmultipathバインディングを作成します。

   ```
   /sbin/multipath -v2 -l
   ```

4. multipathdサービスを開始します。
5. mkinitrdを実行して、システム上にINITRDを再作成し、再起動して変更内容を有効にします。

7.5 マルチパスI/Oサービスの有効化と機動

マルチパスサービスを有効にし、再起動時に起動するには:

1. 端末コンソールを開いて、rootユーザまたは同等の権限でログインします。
2. 端末コンソールのプロンプトで、次のように入力します。

   ```
   chkconfig multipathd on
   ```

   ```
   chkconfig boot.multipath on
   ```

boot.multipathサービスがシステムブートで自動的に開始しない場合は、次の手順に従って、手動でサービスを開始します。

1 端末コンソールを開いて、rootユーザまたは同等の権限でログインします。

2 <Enter>

```
/etc/init.d/boot.multipath start

/etc/init.d/multipathd start
```

7.6 パスフェールオーバーのポリシーと優先度の設定

Linuxホスト内で、ストレージコントローラへのパスが複数ある場合は、各パスが別個のブロックデバイスとして表示され、その結果、1つのLUNに複数のブロックデバイスが存在することになります。デバイスマッパーマルチパスサービスは、同じLUN IDを持つ複数のパスワードを検出し、そのIDで新しいマルチパスデバイスを作成します。たとえば、1つの非ゾーン化されたファイバチャネルのスイッチを介して2つのポートでストレージコントローラに接続した2つのHBAを持つホストは、4つのブロックデバイスを認識します(/dev/sda、/dev/sdb、/dev/sdc、/dev/sdd)。デバイスマッパーマルチパスサービスは、1つのブロックデバイス/dev/mpath/mpath1を作成します。このデバイスは、既に示した4つのブロックデバイスを介してI/Oを再経路指定します。

このセクションでは、フェールオーバーのポリシーを指定し、パスの優先順位を設定する方法について説明します。

- 7.6.1項 「パスのフェールオーバーポリシーの設定」 (92 ページ)
- 7.6.2項 「フェールオーバーポリシーの設定」 (93 ページ)
- 7.6.3項 「スクリプトの使用によるパス優先度の設定」 (102 ページ)
- 7.6.4項 「ALUAの設定 (mpath_prio_alua)」 (103 ページ)

• 7.6.5項 「ターゲットパスグループの報告」 (106 ページ)

7.6.1 パスのフェールオーバーポリシーの設定

multipathコマンドを-pオプション付きで使用して、パスフェールオーバーポリシーを設定します。

```
multipath devicename -p policy
```

次のポリシーオプションの1つで、*policy*を置き換えます。

表 7.4 multipath -pコマンドのグループポリシーオプション

ポリシーオプション	説明
failover (フェールオーバー)	優先度グループごとに1つのパス
multibus	1つの優先度グループ内にすべてのパス
group_by_serial	検出されたシリアル番号ごとに1つの優先度グループ
group_by_prio	パス優先度値ごとに1つの優先度グループ優先度は、コールアウトプログラムで決定されます。それらのプログラムは、グローバル、コントローラごと、またはマルチパスごとのオプションとして/etc/multipath.conf環境設定ファイルで指定されます。
group_by_node_name	ターゲットノード名ごとに1つの優先度グループターゲットノード 名は、/sys/class/fc_transport/target*/node_nameにフェッチされます。

7.6.2 フェールオーバーポリシーの設定

デバイスのフェールオーバーポリシーは、手動で、/etc/multipath.conf ファイルに入力する必要があります。すべての設定とオプションの例は、/usr/share/doc/packages/multipath-tools/multipath.conf .annotatedファイルにあります。

- 「優先度グループと属性の理解」 (93 ページ)
- 「ラウンドロビン式負荷分散の設定」 (101 ページ)
- 「単一パスフェールオーバーの設定」 (101 ページ)
- 「ラウンドロビン式負荷分散用I/Oパスのグループ化」 (102 ページ)

優先度グループと属性の理解

*優先度グループ*は、同じ物理LUNに属するパスのコレクションです。デフォルトでは、I/Oは、グループ内のすべてのパス全体にラウンドロビン方式で配分されます。multipathコマンドは、SANのpath_grouping_policy設定に基づいてそのSANの各LUNごとに、自動的に優先度グループを作成します。multipathコマンドは、グループ内のパス数にグループの優先度を掛け合わせて、どのグループがプライマリか決定します。計算された値が最も高いグループがプライマリグループです。プライマリグループ内のすべてのパスが失敗すると、次に値の高い優先度グループがアクティブになります。

*パス優先度*は、パスに割り当てられた整数値です。優先度は、値が高いほど高くなります。パスごとに優先度を割り当てるには、外部プログラムが使用されます。所定のデバイスに関して、同じ優先度のパスが同じ優先度グループに属します。

表 7.5 *マルチパス属性*

マルチパス属性	説明	値
user_friendly_names	IDを使用するか、または/var/lib/multipath/	**対応:** 実際のIDの代わりに、マルチパスデバイスのエイリアスとして、ユーザフレンドリな名前を自動生成します。

マルチパス属性	説明	値
	bindingsファイルを使用して永続的で固有なエイリアスを/dev/mapper/mpathN形式のマルチパスデバイスに割り当てるか指定します。	デフォルト/dev/disk/by-id/に示されたWWIDを使用します。
blacklist	非マルチパスデバイスとして無視するデバイス名のリスト（cciss、fd、hd、md、dm、sr、scd、st、ram、raw、loopなど）を指定します。	例については、「「/etc/multipath.confでの非マルチパスデバイスのブラックリスト化」(87ページ)」を参照してください。
blacklist_exceptions	ブラックリストに含まれている場合でも、マルチパスデバイスとして扱うデバイス名のリストを指定します。	例として、/usr/share/doc/packages/multipath-tools/multipath.conf.annotatedファイルを参照してください。
failback (フェールバック)	エラーになったパスの回復を監視するかどうか指定し、パスサービス回復後のグループのフェールバックのタイミングを示します。 エラーになったパスは、回復すると、この設定に基づいてマルチパス対応パスのリストに戻されま	**immediate:** パスが回復したら、ただちにパスを有効にします。 **n (> 0):** パスが回復したら、*n*秒後にパスを有効にします。0より大きい整数値を指定してください。 **manual:** デフォルト。エラーになったパスの回復は監視されません。管理者がmultipathコマンドを実行して、有効なパスと優先度グループを更新します。

マルチパス属性	説明	値
	す。multipathは、優先度グループを評価し、プライマリパスの優先度がセカンダリパスのそれを超えると、アクティブな優先度グループを変更します。	

getuid	固有のパス識別子を取得するためのコールのデフォルトプログラムと引数を指定します。絶対パスで指定してください。	`/sbin/scsi_id -g -u -s`
		
		これがデフォルトの場所および引数になります。
		
		例:
		
		`getuid "/sbin/scsi_id -g -u -d /dev/%n"`

no_path_retry	パスの障害時に使用する動作を指定します。	**n (> 0):** multipathコマンドで待ち行列が停止し、パスがエラーになるまでの再試行数を指定します。0より大きい整数値を指定してください。 **失敗:** 即時失敗(待ち行列なし)を指定します。 **キュー** 待ち行列を停止しません(パスが復帰するまで永久に待機します)。
path_grouping_policy	所定のコントローラがホストとなるマルチパスデバイスのパ	**フェールオーバー:** 優先度グループごとに1つのパスが割り当てられるので、一度に1つのパスだけが使用されます。

マルチパス属性	説明	値
	スグループ化ポリシーを指定します。	**multibus:** (デフォルト)すべての有効なパスが1つの優先度グループに属します。トラフィックが、グループ内のアクティブなパスすべてに渡って負荷分散されます。 **group_by_prio:** パス優先度値ごとに、1つの優先度グループが存在します。同じ優先度のパスは同じ優先度グループに属します。優先度は外部プログラムによって割り当てられます。 **group_by_serial:** パスがSCSIターゲットシリアル番号(コントローラノードのWWN)でグループ化されます。 **group_by_node_name:** ターゲットノード名ごとに1つの優先度グループが割り当てられます。ターゲットノード名は/sys/class/fc_transport/target*/node_nameにフェッチされます。
path_checker	パスの状態を判別します。	**directio:** (multipath-toolsバージョン0.4.8以降でのデフォルト)。直接I/Oを持つ最初のセクタを読み込みます。DASDデバイスの場合、有用です。/var/log/messagesに障害メッセージをログ記録します。 **readsector0:** (multipath-tools version 0.4.7以前でのデフォルト)。デバイスの最初のセクタを読み込みます。/var/

マルチパス属性	説明	値
		log/messagesに障害メッセージをログ記録します。 **tur:** デバイスに対するSCSIテストユニットレディコマンドを発行します。これはLUNによってサポートされている場合の推奨設定です。このコマンドは、障害時に、/var/log/messagesにメッセージを出力しません。 一部のSANベンダは、カスタムオプションとしてpath_checkerを提供しています。 • **emc_clariion:** EMC ClariionのEVPDページ0xC0をクエリしてパスの状態を判別します。 • **hp_sw:** Active/Standbyファームウェアを持つHPストレージアレイのパスの状態(アップ、ダウン、またはゴースト)をチェックします。 • **rdac:** LSI/Engenio RDACストレージコントローラのパスｍｐ状態をチェックします。
path_selector	負荷分散に使用するパスセレクタアルゴリズムを指定します。	**round-robin 0:** 優先度グループ内のすべてのアクティブパスに渡るトラフィックの分散に使用される負荷分散アルゴリズム(デフォルト)

マルチパス属性	説明	値
		SUSE Linux Enterprise Server 11以降では、次のような追加のI/O分散オプションが提供されます。 **least-pending(最小保留):** bioベースのデバイスマッパーマルチパスに対して、least-pending-I/O動的負荷分散ポリシーを提供します。この負荷分散ポリシーでは、パス上で保留になっている未処理のサービス要求数を考慮し、保留のサービス要求が最も少ないパスを選択します。 このポリシーは、SAN環境に異機種混合コンポーネントがある場合には特に有効です。たとえば、1台の8GB HBAと1台の2GB HBAが同じサーバに接続されている場合、このアルゴリズムを使用することで、8GB HBAを有効活用できます。 **length-load-balancing(長さによる負荷分散):** least-pendingオプションと同様に、パス上で実行中のI/Oの数に基づく、動的負荷分散装置。 **service-time(サービス時間):** 遅延に従って、パス上のI/Oを調整するサービス時間に基づく負荷分散装置。
pg_timeout	パスグループのタイムアウト処理を指定します。	なし(内部デフォルト)

マルチパス属性	説明	値
prio_callout マルチパスの prio_callouts は、/lib/libmultipath/lib*内の共有ライブラリ内にあります。共有ライブラリを使用することで、デーモンの起動時、コールアウトがメモリにロードされます。	マルチパスマップのレイアウトを決定するプログラムと引数を指定します。 multipathコマンドでクエリされると、指定のmpath_prio_*コールアウトプログラムがマルチパスレイアウト全体との関係で所定のパスの優先度を返します。 この属性がgroup_by_prioのpath_grouping_policyとともに使用されると,同じ優先度を持つすべてのパスが1つのマルチパスグループにグループ化されます。総計の優先度が最も高いグループがアクティブグループになります。 グループ内のすべてのパスが失敗すると、総計の優先度が次に高いグループがアクティブになります。さらに、フェールオーバーコマンド(ハードウェアハンドラにより決定)がター	prio_callout属性を使用しない場合は、すべてのパスが同等になりますこれがデフォルトの設定です。 **/bin/true:** group_by_priorityを使用しない場合に使用します。 prioritizerプログラムは、multipathコマンドによってクエリされると、パス優先度を生成します。プログラム名は、mpath_prio_で開始し、後に使用するデバイスタイプまたは負荷分散の方法が続きます。現在の優先度プログラムには、次のものがあります。 **mpath_prio_alua %n:** SCSI-3 ALUA設定に基づいてパス優先度を生成します。 **mpath_prio_balance_units:** すべてのパスに同じ優先度を生成します。 **mpath_prio_emc %n:** EMCアレイのパス優先度を生成します。 **mpath_prio_hds_modular %b:** Hitachi HDS Modularストレージアレイのパス優先度を生成します。 **mpath_prio_hp_sw %n:** アクティブ/スタンバイモードのCompaq/HPコントローラのパス優先度を生成します。

マルチパス属性	説明	値
	ゲットに送信されることもあります。 mpath_prio_*プログラムとして、指定のセットアップ用にベンダまたは管理者が作成したカスタムスクリプトを使用することも可能です。 コマンドライン内の%nは、/devディレクトリ内のデバイス名に展開します。 %bは、/devディレクトリ内の*major:minor*形式のデバイス番号に展開します。 %dは、/dev/disk/by-idディレクトリ内のデバイスIDに展開します。 デバイスがホットプラグ可能な場合は、%nの代わりに%dフラグを使用してください。これにより、デバイスが利用可能になった時点とudevがデバイスノードを作成した時点の間に短い時間間隔が	**mpath_prio_netapp %n:** NetAppアレイのパス優先度を生成します。 **mpath_prio_random %n:** パスごとにランダムな優先度を生成します。 **mpath_prio_rdac %n:** LSI/Engenio RDACコントローラのパスの優先度を生成します。 **mpath_prio_tpc %n:** オプションとして、ベンダまたは管理者が作成したスクリプトを使用できます。このスクリプトは、各パスの優先度を指定するファイルから優先度を取得します。 **mpath_prio_spec.sh %n:** ユーザ作成スクリプトのパスを提供します。このスクリプトにより、2つ目のデータファイルに含まれた情報に基づくマルチパス処理の優先度が生成されます。（このパスとファイル名は、例として提供されています。ご使用のスクリプトの場所を指定してください)。このスクリプトは、ベンダまたは管理者が作成できます。スクリプトのターゲットファイルは、すべてのマルチパスデバイスの各パスを識別し、パスごとに優先度を指定します。例については、「7.6.3項 「スクリプトの使用によるパス優先度の設定」(102ページ)」を参照してください。

マルチパス属性	説明	値
	あるという問題が解決されます。	
rr_min_io	path_selector設定の指定のアルゴリズムで決定される同じパスグループ内の次のパスに切り替わる前に、パスに経路指定されるI/Oトランザクションの数を指定します。	**n (>0):** 0より大きい整数値を指定してください。 **1000:** デフォルト.
rr_weight	パスの重み付けの方法を指定します。	**ユニフォーム:** デフォルト. すべてのパスが同じラウンドロビン方式の重み付けを持ちます。 **priorities:** 各パスの重み付けは、パスの優先度にrr_min_io設定値を掛け合わせて決定します。

ラウンドロビン式負荷分散の設定

すべてのパスがアクティブです。一定の秒数または一定数のI/Oトランザクションの後で、シーケンスの次のオープンパスに移動するように、I/Oを設定します。

単一パスフェールオーバーの設定

優先度が最も高い（最も低い値の)単一パスがトランザクションに対してアクティブになります。他のパスは、フェールオーバーに使用できますが、フェールオーバーの発生までは使用されません。

ラウンドロビン式負荷分散用I/Oパスのグループ化

同じ優先度を持つ複数のパスがアクティブグループを形成します。そのグループのすべてのパスがエラーになると、デバイスが優先度の次に高いグループにフェールオーバーします。グループのすべてのパスが、ラウンドロビン方式の負荷分散で、トラフィックロードを共有します。

7.6.3 スクリプトの使用によるパス優先度の設定

デバイスマッパーマルチパス(DM-MPIO)と対話することにより、`prio_callout`設定のリソースとしての設定時にLUNへのパスの優先度を提供するスクリプトを作成できます。

まず、各デバイスの情報と各パスに割り当てたい優先度値をリストするテキストファイルを設定します。たとえば、ファイルを`/usr/local/etc/primary-paths`と指定します。次の形式で、パスごとに1行入力します。

```
host_wwpn target_wwpn scsi_id priority_value
```

デバイス上の各パスの優先度を返します。変数FILE_PRIMARY_PATHSが各デバイスの適切なデータ(host wwpn、target wwpn、scsi_id、priority value)を含む実際のファイルに解決することを確認します。

1つのLUNに8つのパスが対応する場合、`primary-paths`ファイルの内容が、次のようになることがあります。

```
0x10000000c95ebeb4 0x200200a0b8122c6e 2:0:0:0 sdb
3600a0b8000122c6d00000000453174fc 50

0x10000000c95ebeb4 0x200200a0b8122c6e 2:0:0:1 sdc
3600a0b80000fd6320000000045317563 2

0x10000000c95ebeb4 0x200200a0b8122c6e 2:0:0:2 sdd
3600a0b8000122c6d0000000345317524 50

0x10000000c95ebeb4 0x200200a0b8122c6e 2:0:0:3 sde
3600a0b80000fd6320000000245317593 2
```

```
0x10000000c95ebeb4 0x200300a0b8122c6e 2:0:1:0 sdi
3600a0b8000122c6d00000000453174fc 5

0x10000000c95ebeb4 0x200300a0b8122c6e 2:0:1:1 sdj
3600a0b80000fd6320000000045317563 51

0x10000000c95ebeb4 0x200300a0b8122c6e 2:0:1:2 sdk
3600a0b8000122c6d0000000345317524 5

0x10000000c95ebeb4 0x200300a0b8122c6e 2:0:1:3 sdl
3600a0b80000fd6320000000245317593 51
```

「表7.5「マルチパス属性」 (93ページ)」に示された例を続行するには、/usr/local/sbin/path_prio.shという名前のスクリプトを作成します。パスとファイル名は任意に指定できます。スクリプトは、次の処理を実行します。

- multipathからのクエリ時に、/usr/local/etc/primary-pathsファイルからデバイスとそのパスをグレップ検索します。
- multipathに戻り、ファイル内のそのエントリの最後のカラムにある優先度値を使用します。

7.6.4 ALUAの設定 (mpath_prio_alua)

mpath_prio_alua(8)コマンドは、Linux multipath(8)コマンドの優先度コールアウトとして使用されます。このコマンドが返す番号をDM-MPIOが使用して、同じ優先度を持つSCSIデバイスをグループ化します。このパス優先度ツールは、ALUA (Asynchronous Logical Unit Access)をベースとしています。

- 「構文」 (104 ページ)
- 「必要条件」 (104 ページ)
- 「オプション」 (104 ページ)
- 「戻り値」 (105 ページ)

構文

```
mpath_prio_alua [-d directory] [-h] [-v] [-V] device [device...]
```

必要条件

SCSIデバイス

オプション

-d*ディレクトリ*
: デバイスノード名の一覧を見つけることのできるLinuxディレクトリパスを指定します。デフォルトディレクトリは、/devです。このオプションを使用する際には、管理するデバイスのデバイスノード名(sdaなど)だけを指定します。

-h
: このコマンドのヘルプを表示して、終了します。

-v
: 詳細出力をオンにして、読みやすい形式でステータスを表示します。指定のデバッガーが属するポートグループやその現在の状態に関する情報が表示されます。

-V
: このツールのバージョン情報を表示して、終了します。

device [*device*...]
: 管理するSCSIデバイス(または複数のデバイス)を指定します。ターゲットポートグループの報告コマンド(sg_rtpg(8))をサポートするSCSIデバイスを指定してください。デバイスノード名には、次の形式の1つを使用します。

  - 完全なLinuxディレクトリパス(/dev/sdaなど)-dオプションとは併用しないでください。
  - デバイスノード名のみ(sdaなど)。-dオプションの使用で、ディレクトリパスを指定します。

- スペース無しでコロンで区切ったデバイスのメジャーおよびマイナー番号(8:0など)この形式を使用すると、/devディレクトリに、tmpdev-<major>:<minor>-<pid>形式の名前で、一時デバイスノードが作成されます。(たとえば、/dev/tmpdev-8:0-<pid>)。

戻り値

成功すると、0値とグループの優先度値が返ります。「表7.6「デバイスマッパーマルチパスのALUA優先度」 (105 ページ)」は、mpath_prio_aluaコマンドが返す優先度値を示しています。

*表 **7.6** デバイスマッパーマルチパスのALUA優先度*

優先度値	説明
50	デバイスはアクティブで最適化されたグループに属します。
10	デバイスは、アクティブだが最適化されていないグループに属します。
1	デバイスはスタンバイグループに属します。
0	他のすべてのグループ。

multipathコマンドでの処理方法により、値の間隔は大きくとられています。multipathコマンドは、グループ内のパス数とグループの優先度の値を掛け合わせた結果の値が最も高いグループを選択します。たとえば、非最適化パスグループが6つのパスを持ち(6 x 10 = 60)、最適化パスグループが1つのパスを持つ(1 x 50 = 50)場合は、非最適化グループが一番高い値を持つので、multipathは、非最適化グループを選択します。デバイスに対するトラフィックは、グループ内の6つのパスすべてを、ラウンドロビン方式で使用します。

エラーが発生すると、コマンドの失敗原因を示す、1～5の値が返ります。詳細については、mpath_prio_aluaのマニュアルページを参照してください。

7.6.5 ターゲットパスグループの報告

SCSIターゲットポートグループの報告(sg_rtpg(8))コマンドを使用します。詳細については、sg_rtpg(8)のマニュアルページを参照してください。

7.7 特定ホストバスアダプタのフェイルオーバーの調整

マルチパスI/Oの使用時に、ホストバスアダプタ(HBA)のエラーまたはケーブルエラーが発生した場合は、マルチパス処理の未使用時より速い報告が必要です。HBAレベルでのフェールオーバーを無効にするように、HBAのタイムアウト設定を構成して、エラーが可能な限り素早くマルチパスI/Oレイヤまでプロパゲートするようにします。マルチパスI/Oレイヤでは、別の正常なパスへのI/Oのリダイレクトが可能です。

フェールオーバーのHBA処理を無効にするには、/etc/modprobe.conf .localファイルでドライバのオプションを変更します。ドライバのフェールオーバー設定を無効にする方法については、HBAベンダのマニュアルを参照してください。

たとえば、ホットバスアダプタのQLogic qla2xxxファミリの場合は、次の設定をお勧めします。

```
options qla2xxx qlport_down_retry=1
```

7.8 ルートデバイスのマルチパスI/Oの設定

重要項目

SUSE Linux Enterprise Server 10 SP1の初期リリース以前では、マルチパス上のルートパーティション(/)は、/bootパーティションが別個の非マルチパスパーティションの場合だけサポートされています。そうでない場合は、ブートローダが書き込まれません。

SUSE Linux Enterprise Server 11では、DM-MPIOが使用可能となり、/boot および/root用にサポートされています。また、YaST2インストーラ内のYaSTパーティショナは、インストール中のマルチパスの有効化をサポートします。

- 7.8.1項 「マルチパスストレージLUNにSLESをインストールするためのマルチパスI/Oの有効化」 (107 ページ)
- 7.8.2項 「アクティブ/パッシブマルチパスストレージLUNにSLESをインストールするためのマルチパスI/Oの有効化」 (108 ページ)
- 7.8.3項 「既存ルートデバイス用マルチパスI/Oの有効化」 (111 ページ)
- 7.8.4項 「ルートデバイスのマルチパスI/Oの無効化」 (111 ページ)

7.8.1 マルチパスストレージLUNにSLESをインストールするためのマルチパスI/Oの有効化

multipathdデーモンは、システムのインストール時に自動的にアクティブになりません。このデーモンは、YaSTパーティショナの［マルチパスの設定］オプションを使用することによって起動できます。

1. YaST2インストール設定ページでのインストール時に、［パーティション分割］をクリックして、YaSTパーティショナを開きます。
2. ［カスタムパーティション - エキスパート用］を選択します。
3. ［ハードディスク］メインアイコンを選択し、［設定］ボタンをクリックし、最後に、［マルチパスの設定］を選択します。
4. multipathを起動します。

   YaST2がディスクの再スキャンを開始し、利用可能なマルチパスデバイスを表示します(/dev/mapper/3600a0b80000f4593000012ae4ab0ae65など)。これが、以降の処理すべての対象デバイスになります。

5 ［次へ］をクリックして、インストールを続行します。

7.8.2 アクティブ/パッシブマルチパスストレージLUNにSLESをインストールするためのマルチパスI/Oの有効化

multipathdデーモンは、システムのインストール時に自動的にアクティブになりません。このデーモンは、YaSTパーティショナの［マルチパスの設定］オプションを使用することによって起動できます。

1 YaST2インストール設定ページでのインストール時に、［パーティション分割］をクリックして、YaSTパーティショナを開きます。

2 ［カスタムパーティション - エキスパート用］を選択します。

3 ［ハードディスク］メインアイコンを選択し、［設定］ボタンをクリックし、最後に、［マルチパスの設定］を選択します。

4 multipathを起動します。

YaST2がディスクの再スキャンを開始し、利用可能なマルチパスデバイスを表示します(`/dev/mapper/3600a0b80000f4593000012ae4ab0ae65`など)。これが、以降の処理すべての対象デバイスになります。デバイスのパスとUUIDを書き留めてください。後で必要になります。

5 ［次へ］をクリックして、インストールを続行します。

6 すべての設定が完了し、インストールが終了すると、YaST2は、ブートローダ情報の書き込みを開始し、システム再起動のカウントダウンを表示します。［中止］をクリックしてカウンタを中止し、<CTRL>+<ALT>+<F5>を押してコンソールにアクセスします。

7 コンソールを使用して、`/boot/grub/device.map`ファイルのhd0エントリにパッシブパスが入力されているかどうか判別します。

これは、インストールではアクティブパスとパッシブパスが区別されないので必要です。

7a 次のように入力して、ルートデバイスを/mntにマウントします。

```
mount /dev/mapper/UUID_part2 /mnt
```

例えば、次のように入力して、すべてのフォントについてアンチエイリアスを無効にします。

```
mount /dev/mapper/3600a0b80000f4593000012ae4ab0ae65_part2 /mnt
```

7b 次のように入力して、ブートデバイスを/mnt/bootにマウントします。

```
mount /dev/mapper/UUID_part1 /mnt/boot
```

例えば、次のように入力して、すべてのフォントについてアンチエイリアスを無効にします。

```
mount /dev/mapper/3600a0b80000f4593000012ae4ab0ae65_part1 /mnt/boot
```

7c 次のように入力して、/mnt/boot/grub/device.mapファイルを開きます。

```
less /mnt/boot/grub/device.map
```

7d /mnt/boot/grub/device.mapファイルでhd0エントリがパッシブパスをポイントしているかどうか判別し、次のいずれかを実行します。

- **アクティブパス:** アクションは不要です。ステップ8 (109ページ)をスキップして、ステップ9 (110ページ)に進みます。
- **パッシブパス:** 設定を変更し、ブートローダを再インストールする必要があります。ステップ8 (109ページ)に進みます。

8 hd0エントリがパッシブパスをポイントする場合は、設定を変更し、ブートローダを再インストールします。

8a コンソールのコンソールプロンプトで、次のコマンドを入力します。

```
mount -o bind /dev /mnt/dev

mount -o bind /sys /mnt/sys

mount -o bind /proc /mnt/proc

chroot
```

8b コンソールで、multipath -llを実行し、その出力をチェックして、アクティブパスを見つけます。

パッシブパスにはghostフラグが付いています。

8c /mnt/boot/grub/device.mapファイルでhd0エントリをアクティブパスに変更し、変更内容を保存し、ファイルを閉じます。

8d MBRからのブートが選択されていた場合、/etc/grub.confは次のようになります。

```
setup --stage2=/boot/grub/stage2 (hd0) (hd0,0)
quit
```

8e 次のコマンドを入力して、ブートローダを再インストールします。

```
grub < /etc/grub.conf
```

8f 次のコマンドを入力します。

```
exit

umount /mnt/*

umount /mnt
```

9 <CTRL>+<ALT>+<F7>を押して、YaSTグラフィック環境に戻ります。

10 ［*OK*］をクリックして、インストールを再起動します。

7.8.3 既存ルートデバイス用マルチパスI/Oの有効化

1 Linuxをインストールし、1つだけパスをアクティブにします。このパスは、パーティショナでby-idシンボリックリンクがリストされるパスがお勧めです。

2 インストール時に使用した/disk//disk/by-idパスを使用してデバイスをマウントします。

3 インストール後、dm-multipathを/etc/sysconfig/kernel:INITRD_MODULESに追加します。

4 System Zの場合、mkinitrdの実行前に、/etc/zipl.confファイルを編集してzipl.conf内のby-path情報を、/etc/fstabで使用されたby-id情報に変更します。

5 /sbin/mkinitrdを再実行して、initrdイメージを更新します。

6 System Zの場合は、mkinitrdの実行後、ziplを実行します。

7 サーバを再起動します。

7.8.4 ルートデバイスのマルチパスI/Oの無効化

- multipath=offをカーネルコマンドラインに追加します。

  これは、ルートデバイスだけに影響します。他のすべてのデバイスは影響されません。

7.9 既存ソフトウェアRAID用マルチパスI/Oの設定

理想的には、デバイスのマルチパス処理を設定してから、それらのデバイスをソフトウェアRAIDデバイスのコンポーネントとして使用してください。ソフトウェアRAIDデバイスの作成後にマルチパス処理を追加した場合は、再起動時にmultipathサービスの後でDM-MPIOサービスが開始することがあります。その場合は、マルチパス処理がRAIDに使用できないように見えます。このセクションのプロシージャを使用すると、すでに存在しているソフトウェアRAIDに対してマルチパス処理を実行できます。

たとえば、次のような場合は、ソフトウェアRAID内のデバイスにマルチパス処理を設定する必要があることがあります。

- 新規インストールまたはアップグレード時にパーティショニング設定の一部として、新しいソフトウェアRAIDを作成する場合
- マルチパス処理用に設定しなかったデバイスをメンバーデバイスまたはスペアとしてソフトウェアRAIDで使用する場合
- 新しいHBAアダプタをサーバに追加するか、またはSAN内でストレージサブシステムを拡張することで、システムを大きくする場合

注記

以降の説明では、ソフトウェアRAIDデバイスを/dev/mapper/mpath0(カーネルによって認識されるデバイス名)と想定しています。ソフトウェアRAIDのデバイス名の指定は、必ず変更してください。

---

1. 端末コンソールを開いて、rootユーザまたは同等の権限でログインします。

   特に指示のない限り、この端末を使用して、以降のステップでコマンドを入力します。

2. ソフトウェアRAIDデバイスが現在マウントされているか、または実行中の場合、デバイスごとに次のコマンドを入力して、デバイスをマウント解除し、停止します。

```
umount /dev/mapper/mpath0

mdadm --misc --stop /dev/mapper/mpath0
```

3 次のように入力して、boot.mdサービスを停止します。

```
/etc/init.d/boot.md stop
```

4 以降のコマンドで、boot.multipathサービスおよびmultipathdサービスを開始します。

```
/etc/init.d/boot.multipath start

/etc/init.s/multipathd start
```

5 マルチパス処理サービスの開始後、ソフトウェアRAIDのコンポーネントデバイスが/dev/disk/by-idディレクトリにリストされているかどうか確認します。次のいずれかの操作を行います。

- **デバイスがリストされている:** デバイス名に、デバイスマッパーマルチパスのデバイス名(/dev/dm-1など)へのシンボリックリンクがあるはずです。
- **デバイスがリストされていない:** デバイスをフラッシュし、再検出することで、マルチパスサービスにデバイスを認識させます。

  この操作を行うには、次のコマンドを入力します。

  ```
  multipath -F

  multipath -v0
  ```

  これで、デバイスが/dev/disk/by-id内にリストされ、デバイスマッパーマルチパスのデバイス名へのシンボリックリンクを持ちます。次に例を示します。

  ```
  lrwxrwxrwx 1 root root 10 Jun 15 09:36 scsi-mpath1 -> ../../dm-1
  ```

6 次のように入力して、boot.mdサービスとRAIDデバイスを再起動します。

```
/etc/init.d/boot.md start
```

7 次のように入力して、ソフトウェアRAIDの状態をチェックします。

```
mdadm --detail /dev/mapper/mpath0
```

RAIDのコンポーネントデバイスは、そのデバイスマッパーマルチパスのデバイス名(/dev/disk/by-idディレクトリにデバイスのシンボリックリンクとしてリストされている）と一致する必要があります。

8 新しいinitrdを作成して、デバイスマッパーマルチパスのサービスが再起動時にRAIDサービスの前にロードされるようにしてください。<Enter>

```
mkinitrd -f multipath
```

9 サーバを再起動して、これらのポストインストール構成の設定を適用します。

10 RAIDステータスをチェックして、ソフトウェアRAIDアレイが、マルチパスデバイスの上に正しく示されることを確認します。<Enter>

```
mdadm --detail /dev/mapper/mpath0
```

次に例を示します。

```
Number Major Minor RaidDevice State
0 253 0 0 active sync /dev/dm-0
1 253 1 1 active sync /dev/dm-1
2 253 2 2 active sync /dev/dm-2
```

7.10 新規デバイスのスキャン(再起動なし)

ご使用のシステムがマルチパス処理用に設定されており、後からSANにストレージを追加する必要がある場合は、rescan-scsi-bus.shスクリプトを使用して新しいデバイスをスキャンすることができます。デフォルトでは、このスクリプトは典型的なLUN範囲ですべてのHBAをスキャンします。

構文

```
rescan-scsi-bus.sh [options] [host [host ...]]
```

コマンドラインでホストを指定するか(廃止予定)、--hosts=LISTオプション(推奨)を使用することができます。

オプション

ほとんどのストレージサブシステムでは、このスクリプトはオプションを指定しなくても正常に実行されます。ただし、特殊な場合はrescan-scsi-bus.shスクリプトに対して、次のいずれかのパラメータを使用する必要があります。

オプション	説明
-l	LUN 0～7のスキャンを起動。[デフォルト: 0]
-L NUM	LUN 0～NUMのスキャンを起動。[デフォルト: 0]
-w	ターゲットデバイスIDが0～15をスキャンします。［デフォルト: 0～7］
-c	チャネル0または1のスキャンを有効にします。[デフォルト: 0]

オプション	説明
-r --remove	デバイスの削除を有効にします。［デフォルト:無効］
-i --issueLip	ファイバチャネルLIPのリセットを発行します。［デフォルト:無効］
--forcerescan	既存デバイスを再度スキャンします。
--forceremove	各デバイスを削除し、再度追加します。 **警告** このオプションは危険なので、注意して使用してください。
--nooptscan	0が検出できない場合は、LUNの検索を中止しないでください。
--color	色付きのプレフィクス、OLD/NEW/DELを使用します。
--hosts=LIST	LIST内のホストだけをスキャンします。ここでLISTは、単一の値と範囲をカンマで区切ったリストです。スペースは使用できません。 --hosts=A[-B][,C[-D]]
--channels=LIST	LIST内のチャネルだけをスキャンします。ここでLISTは、単一の値と範囲をカンマで区切ったリストです。スペースは使用できません。 --channels=A[-B][,C[-D]]

オプション	説明
`--ids=LIST`	LIST内のターゲットIDだけをスキャンします。ここでLISTは、単一の値と範囲をカンマで区切ったリストです。スペースは使用できません。 `--ids=A[-B][,C[-D]]`
`--luns=LIST`	LIST内のLUNだけをスキャンします。ここでLISTは、単一の値と範囲をカンマで区切ったリストです。スペースは使用できません。 `--luns=A[-B][,C[-D]]`

手順

次のプロシージャを使用して、システムを再起動せずに、デバイスをスキャンして、マルチパス処理に使用できるようにします。

1. ストレージサブシステムで、ベンダのツールを使用してデバイスを割り当て、そのアクセス制御設定を更新して、Linuxシステムが新しいストレージをアクセスできるようにします。詳細については、ベンダのマニュアルを参照してください。

2. すべてのターゲットをスキャンしてホストの有無を調べ、LinuxカーネルのSCSIサブシステムのミドルレイヤに新しいデバイスを認識させます。端末コンソールのプロンプトで、次のように入力します。

   ```
   rescan-scsi-bus.sh [options]
   ```

3. システムログ/var/log/messagesファイル)をチェックしてスキャニングの進行状況を調べます。端末コンソールのプロンプトで、次のように入力します。

   ```
   tail -30 /var/log/messages
   ```

   このコマンドは、ログの最後の30行を表示します。次に例を示します。

```
# tail -30 /var/log/messages
. . .
Feb 14 01:03 kernel: SCSI device sde: 81920000
Feb 14 01:03 kernel: SCSI device sdf: 81920000
Feb 14 01:03 multipathd: sde: path checker registered
Feb 14 01:03 multipathd: sdf: path checker registered
Feb 14 01:03 multipathd: mpath4: event checker started
Feb 14 01:03 multipathd: mpath5: event checker started
Feb 14 01:03:multipathd: mpath4: remaining active paths: 1
Feb 14 01:03 multipathd: mpath5: remaining active paths: 1
```

4 「ステップ 2 (117 ページ)」から「ステップ 3 (117 ページ)」まで繰り返し、新しいデバイスに接続しているLinuxシステム上の他のHBAアダプタを介して、パスを追加します。

5 multipathコマンドを実行して、DM-MPIO設定用のデバイスを認識します。端末コンソールのプロンプトで、次のように入力します。

```
multipath
```

これで、新しいデバイスをマルチパス処理用に設定できます。

7.11 パーティショニングされた新規デバイスのスキャン(再起動なし)

このセクションの例を使用して、新たに追加したマルチパスLUNを再起動なしで検出します。

1 端末コンソールを開き、rootユーザとしてログインします。

2 すべてのターゲットをスキャンしてホストの有無を調べ、LinuxカーネルのSCSIサブシステムのミドルレイヤに新しいデバイスを認識させます。端末コンソールのプロンプトで、次のように入力します。

```
rescan-scsi-bus.sh [options]
```

rescan-scsi-bus-shスクリプトの構文とオプション情報の詳細については、7.10項 「新規デバイスのスキャン(再起動なし)」 (115 ページ)を参照してください。

3 次のように入力して、デバイスが認識されていること(リンクに新しいタイムスタンプが付いている)を確認します。

```
ls -lrt /dev/dm-*
```

4 次のように入力して、デバイスの新しいWWNがログに表示されることを確認します。

```
tail -33 /var/log/messages
```

5 テキストエディタで、デバイスの新しいエイリアス定義を/etc/multipath.confファイルに追加します(oradata3など)。

6 次の入力で、デバイスのパーティションテーブルを作成します。

```
fdisk /dev/dm-8
```

7 次のように入力して、udevをトリガします。

```
echo 'add' > /sys/block/dm-8/uevent
```

これによって、dm-8上のパーティションにデバイスマッパーデバイス作成されます。

8 次の入力で、新しいパーティションのファイルシステムおよびラベルを作成します。

```
mke2fs -j /dev/dm-9
```

```
tune2fs -L oradata3 /dev/dm-9
```

9 次の入力で、DM-MPIOを再起動して、エイリアスを読み込ませます。

```
/etc/init.d/multipathd restart
```

10 次の入力で、デバイスがmultipathdによって認識されていることを確認します。

```
multipath -ll
```

11 テキストエディタで、/etc/fstabファイルにマウントエントリを追加します。

この時点では、「ステップ5(119ページ)」で作成したエイリアスは、まだ、/dev/disk/by-labelディレクトリにあります。マウントエントリを/dev/dm-9パスに追加した後、次回の再起動の前に、マウントエントリを次のように変更します。

```
LABEL=oradata3
```

12 次のように入力して、マウントポイントとして使用するディレクトリを作成し、デバイスをマウントします。

```
md /oradata3
```

```
mount /oradata3
```

7.12 マルチパスI/Oステータスの表示

マルチパスI/Oのステータスをクエリすると、マルチパスマップの現在のステータスが出力されます。

multipath -lオプションを使用すると、パスチェッカが最後に実行された時点での現行パスステータスが表示されます。ただし、パスチェッカは実行されません。

multipath -llオプションを使用すると、パスチェッカが実行され、パス情報が更新され、最後に、現在のステータス情報が表示されます。このオプションは、常に、パスステータスの最新情報を表示します。

- 端末コンソールのプロンプトで、次のように入力します。

```
multipath -ll
```

各マルチパスデバイスの情報が表示されます。次に例を示します。

```
3600601607cf30e00184589a37a31d911
[size=127 GB][features="0"][hwhandler="1 emc"]


\_ round-robin 0 [active][first]
  \_ 1:0:1:2 sdav 66:240  [ready ][active]
  \_ 0:0:1:2 sdr  65:16   [ready ][active]


\_ round-robin 0 [enabled]
  \_ 1:0:0:2 sdag 66:0    [ready ][active]
  \_ 0:0:0:2 sdc  8:32    [ready ][active]
```

デバイスごとに、デバイスのID、サイズ、機能、およびハードウェアハンドラが表示されます。

デバイスへのパスは、自動的に、デバイス検出時に優先度グループとしてグループ化されます。一度に1つの優先度グループだけがアクティブになります。アクティブ/アクティブ構成の場合、すべてのパスが同じグループに属します。アクティブ/パッシブ構成の場合、パッシブパスは別個の優先度グループに属しれます。

グループごとに、次の情報が表示されます。

- ラウンドロビン方式など、グループ内でのI/O負荷の分散に使用されるスケジューリングポリシー
- グループがアクティブか、無効か、または有効か
- 最初の(優先度の最も高い)グループかどうか
- グループ内に含まれるパス

パスごとに、次の情報が表示されます。

- *host:bus:target:lun*としての物理アドレス(1:0:1:2など)
- デバイスノード 名(sdaなど)
- メジャー/マイナー番号
- デバイスのステータス

7.13 エラーになったI/Oの管理

queue_if_no_pathを有効にすることで、すべてのパスで同時に障害が発生した場合は、I/Oをキューに登録するように、マルチパス処理を設定する必要があるかもしれません。設定しておかないと、すべてのパスに障害が発生するとI/Oもすぐに失敗してしまいます。ドライバ、HBA、またはファブリックにスプリアスエラーが発生したというシナリオでは、それらのエラーですべてのパスが失われるI/Oをすべて待ち行列に入れ、エラーを上方にプロパゲートしないように、DM-MPIOを設定してください。

マルチパスデバイスをクラスタで使用する場合は、queue_if_no_pathを無効にすることができます。このようにすると、I/Oが待ち行列に入らず、自動的にパス障害となり、I/Oエラーがエスカレートしてクラスタリソースのフェールオーバーが行われます。

ただし、queue_if_no_pathを有効にすると、パスが回復しない限り、I/Oがいつまでもキューに留まることになるので、`multipathd`が実行中であり、シナリオに有効なことを確認してください。確認しておかないと、再起動するまで、またはキューの代わりに手動でフェールオーバーに戻すまで、影響を受けたマルチパスデバイスでI/Oが無限に停止する可能性があります。

シナリオをテストするには:

1. 端末コンソールで、`root`ユーザとしてログインします。
2. 次の入力で、デバイスI/Oに関して、フェールオーバーの代わりに待ち行列処理をアクティブにします。

   ```
   dmsetup message device_ID 0 queue_if_no_path
   ```

   *`device_ID`*を実際のデバイスのIDに置き換えます。たとえば、次のように入力します。

   ```
   dmsetup message 3600601607cf30e00184589a37a31d911 0 queue_if_no_path
   ```

3. 次の入力で、デバイスI/Oのフェールオーバーに戻ります。

   ```
   dmsetup message device_ID 0 fail_if_no_path
   ```

このコマンドにより、ただちに、待ち行列に入ったすべてのI/Oがエラーになります。

*device_ID*を実際のデバイスのIDに置き換えます。たとえば、次のように入力します。

```
dmsetup message 3600601607cf30e00184589a37a31d911 0 fail_if_no_path
```

待ち行列内のI/Oをすべてのパスがエラーになるシナリオ用に設定するには:

1. 端末コンソールで、rootユーザとしてログインします。
2. /etc/multipath.confファイルをテキストエディタで開きます。
3. defaultsセクションとその閉じ括弧を非コメント化した後、次のようにdefault_features設定を追加します。

```
defaults {
  default_features "1 queue_if_no_path"
}
```

4. /etc/multipath.confファイルの変更後、mkinitrdを実行してシステム上にINITRDを作成し、次に、再起動で変更内容を有効にします。
5. デバイスI/Oのフェールオーバーに戻る準備ができたら、次のように入力します。

```
dmsetup message mapname 0 fail_if_no_path
```

*mapname*を該当デバイスのマップされたエイリアス名またはデバイスIDに置き換えます。

このコマンドにより、待ち行列で待機中のすべてのI/Oがエラーとなり、エラーが呼び出し側アプリケーションにプロパゲートします。

7.14 停止したI/Oの解決

すべてパスが同時にエラーとなり、I/Oが待ち行列に入って停止している場合は、次のプロシージャを実行します。

1 端末コンソールのプロンプトで、次のコマンドを入力します。

```
dmsetup message mapname 0 fail_if_no_path
```

*mapname*をデバイスの正しいデバイスIDまたはマップされたエイリアス名で置き換えます。このコマンドにより、すべての待ち行列内のI/Oがエラーとなり、エラーが呼び出し側アプリケーションにプロパゲートします。

2 端末コンソールのプロンプトで、次のコマンドを入力して、待ち行列を再度アクティブにします。

```
dmsetup message mapname 0 queue_if_no_path
```

7.15 追加情報

SUSE Linux Enterprise Server上でマルチパスI/Oを設定し、使用する詳細については、Novellサポートナレッジベースから次の追加リソースを参照してください。

- *How to Setup/Use Multipathing on SLES* [http://support.novell.com/techcenter/sdb/en/2005/04/sles_multipathing.html]
- *Troubleshooting SLES Multipathing (MPIO) Problems (技術情報ドキュメント3231766)* [http://www.novell.com/support/search.do?cmd=displayKC&docType=kc&externalId=3231766&sliceId=SAL_Public]
- *Dynamically Adding Storage for Use with Multipath I/O (技術情報ドキュメント3000817)* [https://secure-support.novell.com/KanisaPlatform/Publishing/911/3000817_f.SAL_Public.html]

- *DM MPIO Device Blacklisting Not Honored in multipath.conf (技術情報ドキュメント3029706)* [http://www.novell.com/support/search.do?cmd=displayKC&docType=kc&externalId=3029706&sliceId=SAL_Public&dialogID=57872426&stateId=0%200%2057878058]
- *Static Load Balancing in Device-Mapper Multipathing (DM-MP) (技術情報ドキュメント3858277)* [http://www.novell.com/support/search.do?cmd=displayKC&docType=kc&externalId=3858277&sliceId=SAL_Public&dialogID=57872426&stateId=0%200%2057878058]
- *Troubleshooting SCSI (LUN) Scanning Issues (技術情報ドキュメント3955167)* [http://www.novell.com/support/search.do?cmd=displayKC&docType=kc&externalId=3955167&sliceId=SAL_Public&dialogID=57868704&stateId=0%200%2057878206]

7.16 次に行う作業

ソフトウェアRAIDを使用したい場合は、デバイス上でファイルシステムを作成してから、ソフトウェアRAIDの作成と設定をしてください。詳細については、次のリンクを参照してください。

- 第8章 *ソフトウェアRAIDの設定* (127 ページ)
- 第10章 *mdadmによるソフトウェアRAID 6および10の管理* (143 ページ)

8

ソフトウェアRAIDの設定

RAID (Redundant Array of Independent Disks)の目的は、複数のハードディスクパーティションを1つの大きい仮想ハードディスクに結合し、パフォーマンスとデータのセキュリティを最適化することです。ほとんどのRAIDコントローラはSCSIプロトコルを使用します。 これは、IDEプロトコルも効率的な方法で多数のハードディスクのアドレスを指定でき、コマンドのパラレル処理に適しているからです。一方、IDEまたはSATAハードディスクをサポートしているRAIDコントローラもあります。ソフトウェアRAIDは、ハードウェアRAIDコントローラ購入による追加コストなしで、RAIDシステムの利点を提供します。ただし、これにはいくらかのCPU時間を要し、高性能なコンピュータには適さないメモリ要件があります。

重要項目

ソフトウェアRAIDは、OCFS2などのクラスタ化されたファイルシステムの下ではサポートされません。これはRAIDが同時起動をサポートしないためです。OCFS2にRAIDを使用したい場合は、RAIDをストレージサブシステムに処理させる必要があります。

SUSE® Linux Enterpriseには、いくつかのハードディスクを1つのソフトウェアRAIDシステムに統合するオプションがあります。RAIDには、それぞれが異なる目標、利点、および属性を持ついくつかのハードディスクを1つのRAIDシステムに結合するためのいくつかの戦略が含まれています。これらは通常、*RAID レベル*と呼ばれます。

- 8.1項 「RAIDレベルの理解」 (128 ページ)
- 8.2項 「YaSTによるソフトウェアRAID設定」 (130 ページ)

- 8.3項 「トラブルシューティング」 (132 ページ)
- 8.4項 「詳細情報」 (133 ページ)

8.1 RAIDレベルの理解

このセクションでは、通常のRAIDレベル(0、1、2、3、4、5)とネストしたRAIDレベルについて説明します。

- 8.1.1項 「RAID 0」 (128 ページ)
- 8.1.2項 「RAID 1」 (128 ページ)
- 8.1.3項 「RAID 2およびRAID 3」 (129 ページ)
- 8.1.4項 「RAID 4」 (129 ページ)
- 8.1.5項 「RAID 5」 (129 ページ)
- 8.1.6項 「ネストしたRAIDレベル」 (130 ページ)

8.1.1 RAID 0

このレベルでは、各ファイルのブロックが複数のディスクドライブに分散されるので、データアクセスのパフォーマンスが向上します。このレベルはデータのバックアップを提供しないため、実際にはRAIDではありませんが、この種のシステムでは*RAID 0*という名前が一般的です。RAID 0では、2つ以上のハードディスクが互いにプールします。高いパフォーマンスが得られます。ただし、1つのハードディスクに障害が発生しただけで、RAIDシステムが破壊され、データは失われます。

8.1.2 RAID 1

このレベルでは、データが他のハードディスクに一対一でコピーされるため、データに対する適切なセキュリティが提供されます。 これは、*ハードディスクミラーリング*として知られています。ディスクが破壊された場合は、ディスクの内容のコピーをミラー先のもう1つのディスクで利用できます。した

がって、1つのディスク以外のすべてのディスクが損なわれても、データを保全できます。ただし、損傷が検出されない場合は、正しいディスクに損傷したデータがミラーリングされる可能性があり、その場合はデータが壊れます。単一ディスクアクセスの使用時と比較すると、コピープロセスで書き込みのパフォーマンスが若干低下しますが(10〜20%遅くなる)、読み取りアクセスは、通常の物理ハードディスクのどれと比べても、著しく高速です。これは、データが複製されているので、それらを並行してスキャンできるためです。RAID 1では、一般に、読み取りトランザクションの速度が単一ディスクのほぼ2倍、書き込みトランザクションの速度が単一ディスクと同じです。

8.1.3 RAID 2およびRAID 3

これらは、一般的なRAID実装ではありません。レベル2では、データは、ブロックレベルではなく、ビットレベルでストライプ化されます。レベル3は、専用パリティディスクによってバイトレベルのストライプ化を提供しますが、複数の要求を同時にサービスすることはできません。両レベルとも、まれにしか使用されません。

8.1.4 RAID 4

レベル4は、専用パリティディスクと結合されたレベル0と同様に、ブロックレベルのストライプ化を提供します。データディスクがエラーになると、パリティデータで置き換え用のディスクが作成されます。ただし、パリティディスクは、書き込みアクセスにボトルネックを生成する可能性があります。にもかかわらず、レベル4は時々使用されます。

8.1.5 RAID 5

RAID 5は、レベル0とレベル1の間をパフォーマンスおよび冗長性の面で調整して、最適化したものです。ハードディスクスペースは、使用されるディスク数から1を引いたものに等しくなります。データは、RAID 0の場合のようにハードディスク間で分散されます。パーティションの1つで作成されたパリティブロックがあるのは、セキュリティ上の理由からです。各パーティションはXORによって互いにリンクされているので、システム障害の場合に、内容が対応するパリティブロックによって再構築されます。RAID 5の場合、同時に複数のハードディスクが障害を起こすことはありません。1つのハード

ディスクに障害がある場合は、そのハードディスクをできるだけ早く交換して、データ消失の危険性をなくす必要があります。

8.1.6 ネストしたRAIDレベル

他にもいくつかのRAIDレベルが開発されています(RAIDn、RAID 10、RAID 0+1、RAID 30、RAID 50など)。それらの一部は、ハードウェアベンダによって作成された専有インプリメンテーションです。これらのレベルは、あまり広く使用されてはいないので、ここでの説明は省略します。

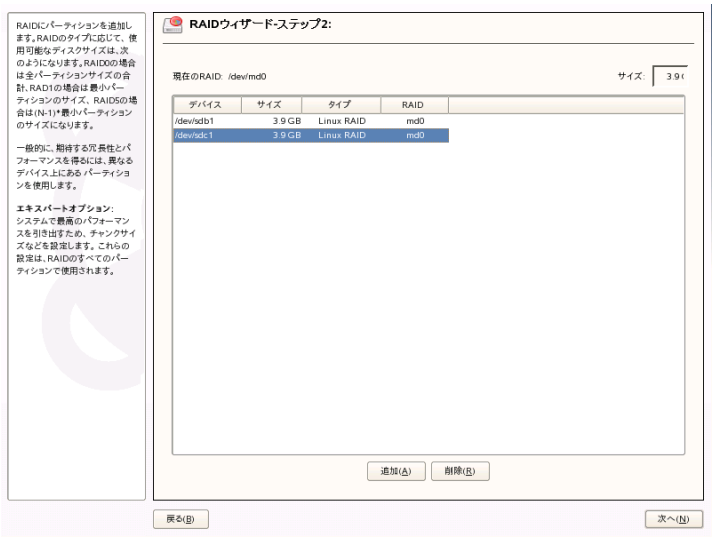
8.2 YaSTによるソフトウェアRAID設定

YaSTソフトRAID設定には、YaST Expert Partitionerからアクセスできます。このパーティション設定ツールを使用すると、既存のパーティションを編集および削除したり、ソフトウェアRAIDで使用する新規パーティションを作成できます。

RAIDパーティションを作成するには、まず、［*作成*］［*Do not format(フォーマットしない)*］の順にクリックし、次に、［*0xFD Linux RAID*］を選択します。RAID 0およびRAID 1の場合、少なくとも2つのパーティションが必要です。RAID 1の場合、パーティションは2つだけです。RAID 5を使用する場合、少なくとも3つのパーティションが必要です。各セグメントは最小サイズのパーティションと同量のスペースしか提供できないので、同じサイズのパーティションだけを使用するようお勧めします。RAIDパーティションを異なるハードディスクに保存すると、1つが損傷した場合のデータ消失のリスクが削減され(RAID 1と5)、またRAID 0のパフォーマンスを最適化できます。RAIDで使用するすべてのパーティションを作成したら、［*RAID*］［*Create RAID (RAIDの作成)*］の順にクリックして、RAID設定を開始します。

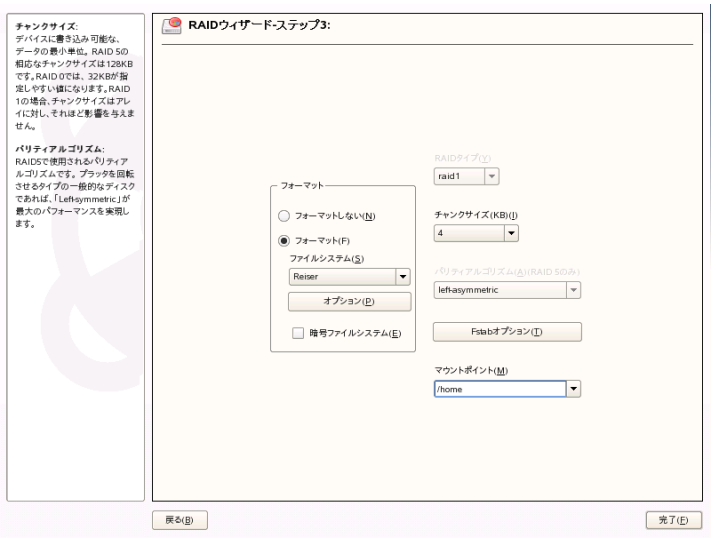
次のダイアログでは、RAIDレベル0、1、5から選択して、［次へ］をクリックします。次のダイアログ(「図8.1「RAIDパーティション」(131 ページ)」参照)では、［*Linux RAID*］または［*Linuxネイティブ*］のどちらかのタイプのすべてのパーティションがリストされます。スワップパーティションまたはDOSパーティションは表示されません。パーティションがRAIDボリュームにすでに割り当てられている場合は、RAIDデバイスの名前(たとえば/dev/md0)がリストに表示されます。割り当てられていないパーティションは、「--」で示されます。

図 8.1　RAIDパーティション



割り当て解除済みのパーティションを選択したRAIDボリュームに追加するには、まず、そのパーティションを選択して、［*追加*］をクリックします。この時点で、選択したパーティションの横に、RAIDデバイスの名前が表示されます。すべてのパーティションをRAID用の予約パーティションとして割り当てます。すべてのパーティションを割り当てないと、パーティションのスペースが未使用のまま残ります。すべてのパーティションを割り当てたら、［次へ］をクリックして、設定ダイアログに進みます。このダイアログではパフォーマンスを微調整できます(図8.2「ファイルシステム設定」 (132 ページ)を参照)。

図 8.2　ファイルシステム設定



従来のパーティションの場合と同様の設定以外だけでなく、暗号化とRAIDボリュームのマウントポイントを使用するように、ファイルシステムを設定します。［完了］で設定を完了した後、Expert Partitionerで、*RAID*と指定されている/dev/md0などのデバイスを参照してください。

8.3 トラブルシューティング

/proc/mdstatsファイルをチェックして、RAIDパーティションが破損しているかどうかを調べます。システム障害が発生した場合は、Linuxシステムをシャットダウンして、問題のあるハードディスクを、同じ方法でパーティショニングされている新しいハードディスクに置き換えます。次に、システムを再起動して、mdadm /dev/mdX --add /dev/sdXコマンドを入力します。「X」を特定のデバイス識別子に置き換えてください。これにより、ハードディスクがRAIDシステムに自動的に統合され、そのRAIDシステムが完全に再構築されます。

再構築中もすべてのデータにアクセスできますが、RAIDが完全に再構築されるまでは、パフォーマンスに問題が発生する場合があります。

8.4 詳細情報

ソフトウェアRAIDの設定方法と詳細情報が、次のHOWTOにあります。

- *The Software RAID HOWTO* [http://en.tldp.org/HOWTO/Software-RAID-HOWTO.html]
- *The Software RAID HOWTO*(/usr/share/doc/packages/mdadm/Software-RAID.HOWTO.htmlファイル)

「linux-raid [http://marc.theaimsgroup.com/?l=linux-raid]」などのLinux RAIDメーリングリストもあります。

ルートパーティション用のソフトウェアRAIDの設定

9

SUSE® Linux Enterprise Server 11では、デバイスマッパーRAIDツールがYaSTパーティショナに統合されました。インストール時にパーティショナを使用して、ルート(/)パーティションを含むシステムデバイス用にソフトウェアRAIDを作成することができます。

9.1 ソフトウェアRAIDの必須条件

設定が次の要件を満たしていることを確認してください。

- 作成予定のソフトウェアRAIDの種類によって、2つ以上のハードドライブが必要になります。
  - **RAID 0 (ストライピング):** RAID 0には2つ以上のデバイスが必要です。RAID 0は耐障害性機能を提供しないので、システムデバイスにはお勧めしません。

- **RAID 1(ミラーリング):** RAID 1には2つのデバイスが必要です。
- **RAID 5(冗長ストライピング):** RAID 5には3つ以上のデバイスが必要です。

- ハードドライブは類似のサイズで構成する必要があります。RAIDは最も小さいドライブのサイズを採用します。
- ブロックストレージデバイスには、ローカル(マシンに内蔵、または直結されたもの)、ファイバチャネルストレージサブシステム、またはiSCSIストレージサブシステムを自由に組み合わせることができます。
- ハードウェアRAIDデバイスを使用している場合は、その上でソフトウェアRAIDを実行しようとしないでください。
- iSCSIターゲットデバイスをご使用の場合は、RAIDデバイスを作成する前にiSCSIイニシエータサポートを有効にします。
- ご使用のストレージサブシステムが、ソフトウェアRAIDを使用する予定の直結されたローカルデバイス、ファイバチャネル、またはiSCSIデバイスとサーバの間で複数のI/Oパスを提供している場合は、RAIDデバイスを作成する前に、マルチパスサポートを有効にしなければなりません。

9.2 インストール時にiSCSIイニシエータサポートを有効にする

ルート(/)パーティションに使用する予定のiSCSIターゲットデバイスがある場合は、ソフトウェアRAIDデバイスを作成する前に、iSCSIイニシエータソフトウェアを有効にして、これらのデバイスを使用可能にしなければなりません。

1 ［インストールの設定］ページが表示されるまで、SUSE Linux Enterprise 11のYaSTのインストールを行います。

2 ［*パーティション分割*］をクリックして、［ハードディスクの準備］ページを開き、［*カスタムパーティション(エキスパート用)*］をクリックし、次に［*次へ*］をクリックします。

3 ［エキスパートパーティショナ］のページで、システムビューパネルの［ハードディスク］を展開し、デフォルトの提案内容を表示します。

4 ［ハードディスク］ページで、［*設定*］、［*iSCSIの設定*］を選択し、次にiSCSIイニシエータの設定を続行するかどうかのプロンプトが表示されたら［*続ける*］をクリックします。

9.3 インストール時にマルチパスI/Oのサポートを有効にする

ルート(/)パーティション用のソフトウェアRAIDの作成に使用する予定のデバイスへの複数のI/Oパスがある場合は、ソフトウェアRAIDデバイスを作成する前にマルチパスサポートを有効にする必要があります。

1 ［インストールの設定］ページが表示されるまで、SUSE Linux Enterprise 11のYaSTのインストールを行います。

2 ［*パーティション分割*］をクリックして、［ハードディスクの準備］ページを開き、［*カスタムパーティション(エキスパート用)*］をクリックし、次に［*次へ*］をクリックします。

3 ［エキスパートパーティショナ］のページで、システムビューパネルの［ハードディスク］を展開し、デフォルトの提案内容を表示します。

4 ［ハードディスク］ページで、［*設定*］、［*マルチパスの設定*］を選択し、次にマルチパスの起動を確認するプロンプトが表示されたら、［*はい*］をクリックします。

これによってデバイスが再度スキャンされ、複数のパスが展開されるので、ハードディスクのリストに各デバイスが重複して掲載されることはありません。

9.4 ルート(/)パーティション用のソフトウェアRAIDデバイスの作成

1 ［インストールの設定］ページが表示されるまで、SUSE Linux Enterprise 11のYaSTのインストールを行います。

2 ［*パーティション分割*］をクリックして、［ハードディスクの準備］ページを開き、［*カスタムパーティション(エキスパート用)*］をクリックし、次に［*次へ*］をクリックします。

3 ［エキスパートパーティショナ］ページで、システムビューパネルの［*ハードディスク*］を展開し、デフォルトの提案内容を表示し、提案されているパーティションを選択して、［*削除*］をクリックします。

4 スワップパーティションを作成します。

4a ［エキスパートパーティショナ］ページの［*ハードディスク*］で、スワップパーティションに使用するデバイスを選択し、次に［*ハードディスクパーティション*］タブの［*追加*］をクリックします。

4b ［*新しいパーティションの種類*］で、［*プライマリパーティション*］を選択し、次に［*次へ*］をクリックします。

4c ［*新しいパーティションのサイズ*］で、使用するサイズを指定し、次に［*次へ*］をクリックします。

4d ［*フォーマットオプション*］で［*パーティションをフォーマットする*］を選択し、次にドロップダウンリストから［スワップ］を選択します。

4e ［*マウントオプション*］で、［*パーティションをマウントする*］を選択し、次にドロップダウンリストから［スワップ］を選択します。

4f ［*完了*］をクリックします。

5 ソフトウェアRAIDに使用する各デバイスの*0xFD Linux RAID*フォーマットを設定します。

5a ［エキスパートパーティショナ］ページの［*ハードディスク*］で、RAIDで使用するデバイスを選択し、次に［*ハードディスクパーティション*］タブの［*追加*］をクリックします。

5b ［*新しいパーティションの種類*］で、［*プライマリパーティション*］を選択し、次に［*次へ*］をクリックします。

5c ［*新しいパーティションのズサイズ*］で、最大サイズの使用を指定し、次に［*次へ*］をクリックします。

5d ［*フォーマットオプション*］で、［*パーティションをフォーマットしない*］を選択し、次にドロップダウンリストから［*0xFD Linux RAID*］を選択します。

5e ［*マウントオプション*］で、［*パーティションをマウントしない*］を選択します。

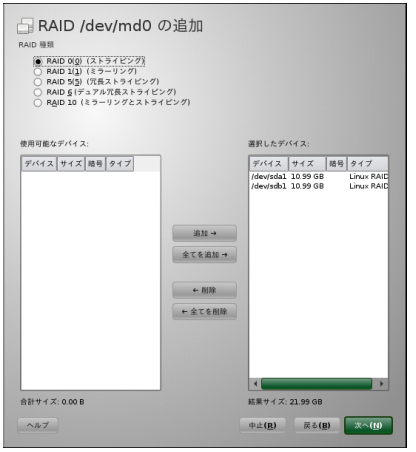
5f ［*完了*］をクリックします。

5g ソフトウェアRAIDで使用する各デバイスに、ステップ 5a (139 ページ)からステップ 5f (139 ページ)を繰り返し実行します。

6 RAIDデバイスを作成します。

6a システムビューパネルで、［*RAID*］を選択し、次に［RAID］ページで［*RAIDの追加*］をクリックします。

ステップ 5 (138 ページ)で作成したデバイスが［*使用可能なデバイス*］の一覧に表示されます。



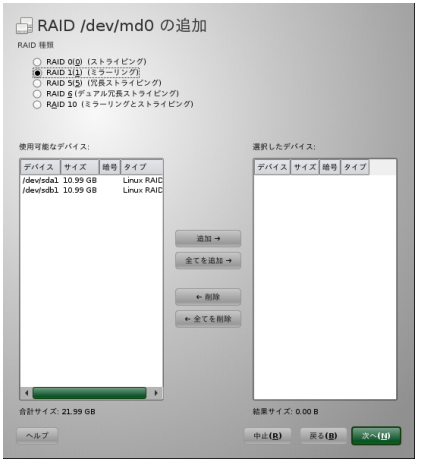
6b ［*RAID種類*］で、［*RAID 0(ストライピング)*］、［*RAID 1(ミラーリング)*］、または［*RAID 5(冗長ストライピング)*］を選択します。

たとえば、RAID 1(ミラーリング)を選択します。

6c ［*使用可能なデバイス*］パネルで、RAIDに使用するデバイスを選択し、次に［*追加*］をクリックして、［*選択したデバイス*］パネルにデバイスを移動します。

RAID 1には2つ以上のデバイスを、RAID 0には2つのデバイスを、あるいはRAID 5には3つ以上のデバイスを指定します。

ここでは一例として、RAID 1に2つのデバイスを選択します。



6d ［次へ］をクリックします。

6e ［*RAIDオプション*］で、ドロップダウンリストからチャンクサイズを選択します。

RAID 1(ミラーリング)のデフォルトチャンクサイズは4KBです。

RAID 0(ストライピング)のデフォルトチャンクサイズは32KBです。

使用可能なチャンクサイズは、4KB、8KB、16KB、32KB、64KB、128KB、256KB、512KB、1MB、2MB、または4MBです。

6f ［フォーマットのオプション］で、［パーティションをフォーマットする］を選択し、次に［ファイルシステム］ドロップダウンリストからファイルシステムの種類(Ext3など)を選択します。

6g ［マウントオプション］で、［パーティションをマウントする］を選択し、次に［マウントポイント］ドロップダウンリストから、 / を選択します。

6h ［完了］をクリックします。

ソフトウェアRAIDデバイスはデバイスマッパーによって管理され、デバイスを/dev/md0パスの下に作成します。

7 ［エキスパートパーティショナ］ページで、［*受諾*］をクリックします。

［インストールの設定］ページの［*パーティション分割*］の下に新しい提案内容が表示されます。

たとえば、次のセットアップが表示されます。

8 インストールを続行します。

サーバを再起動するたびに、デバイスマッパーが起動時に開始し、ソフトウェアRAIDが自動的に認識され、ルート(/)パーティション上のオペレーティングシステムを開始することができます。

mdadmによるソフトウェアRAID 6および10の管理

10

このセクションでは、複数デバイス管理(mdadm(8))ツールで、ソフトウェアRAID 6および10のデバイスを作成する方法を説明します。mdadmを使用すると、RAID 0、1、4、および5も作成できます。mdadmツールは、レガシープログラムmdtoolsおよびraidtoolsの機能を提供します。

10.1 RAID 6の作成

10.1.1 RAID 6の理解

RAID 6は、本来、RAID 5の拡張であり、2つ目の独立した分散パリティスキーム(デュアルパリティ)の使用により、耐障害性をさらに追加します。データ回

復プロセスで、2つのハードディスクドライブに障害が発生しても、システムは稼動し続け、データが失われることはありません。

RAID 6は、複数の同時ドライブエラーに耐えることで、非常に高いデータ耐障害性を提供します。RAID 6は、データを失うことなく、2つのデバイスの喪失を処理します。したがって、N個のドライブのデータを保存するには、N+2個のドライブが必要です。その結果、最低限4個のデバイスが必要となります。

通常モードおよび単一ディスク障害モードでは、RAID 5と比べ、RAID 6のパフォーマンスは若干低いですが、同程度です。デュアルディスク障害モードでは、RAID 6は非常に低速です。

表 10.1　RAID 5とRAID 6の比較

機能	RAID 5	RAID 6
デバイスの数	N+1(最小限3個)	N+2(最小限4個)
パリティ	分散型、シングル	分散型、デュアル
パフォーマンス	書き込みおよび再構築に中程度の影響	シーケンシャルな書き込みでは、RAID 5より影響大
耐障害性	1つのコンポーネントデバイスの障害	2つのコンポーネントデバイスの障害

10.1.2 RAID 6の作成

このセクションのプロシージャでは、RAID 6デバイス`/dev/md0`を4つのデバイスで作成します(`/dev/sda1`、`/dev/sdb1`、`/dev/sdc1`、`/dev/sdd1`)。必ず、プロシージャを変更して、実際のデバイスノードを使用するようにしてください。

1 端末コンソールを開いて、`root`ユーザまたは同等の権限でログインします。

2 RAID6デバイスを作成します。コマンドプロンプトで、次のように入力します

```
mdadm --create /dev/md0 --run --level=raid6 --chunk=128 --raid-devices=4
 /dev/sdb1 /dev/sdc1 /dev/sdc1 /dev/sdd1
```

デフォルトのチャンクサイズは 64KBです。

3 このセクションのプロシージャではRAID 6デバイス/dev/md0上でファイルシステム(Reiserファイルシステムなど)を作成します。たとえば、コマンドプロンプトで、次のように入力します。

```
mkfs.reiserfs /dev/md0
```

これとは別のファイルシステムを使用したい場合は、コマンドを変更します。

4 /etc/mdadm.confファイルを編集して、コンポーネントデバイスとRAIDデバイス /dev/md0のエントリを追加します。

5 /etc/fstabファイルを編集して、RAID 6デバイス /dev/md0のエントリを追加します。

6 サーバを再起動します。

RAID 6デバイスが/localにマウントされます。

7 RAIDアレイにホットスペアを追加します(オプション)。たとえば、コマンドプロンプトで、次のように入力します。

```
mdadm /dev/md0 –a /dev/sde1
```

10.2 mdadmによるネストしたRAID 10デバイスの作成

• 10.2.1項 「ネストしたRAIDデバイスの理解」 (146 ページ)

- 10.2.2項 「mdadmによるネストしたRAID 10 (1+0)デバイスの作成」 (148 ページ)
- 10.2.3項 「mdadmによるネストしたRAID 10 (0+1)デバイスの作成」 (149 ページ)

10.2.1 ネストしたRAIDデバイスの理解

ネストしたRAIDデバイスは、物理ディスクを使用する代わりに、その基本エレメントとして別のRAIDアレイを使用するRAIDアレイで構成されます。この構成の目的は、RAIDのパフォーマンスと耐障害性を向上することです。

Linuxは、RAID 1 (ミラーリング)アレイとRAID 0 (ストライピング)アレイのネストをサポートします。一般に、この組み合わせは、RAID 10と呼ばれます。ネストの順序を区別するため、このマニュアルでは、次の用語を使用します。

- **RAID 1+0:** まず、RAID 1（ミラー）アレイが構築され、次に、それらのアレイが組み合わされてRAID 0 (ストライプ)アレイを構成します。
- **RAID 0+1:** まず、RAID 0（ストライプ）アレイが構築され、次に、それらのアレイが組み合わされてRAID 1(ミラー)アレイを構成します。

次のテーブルでは、RAID 10ネスティングの欠点と利点を、1+0対0+1という形式で説明します。使用するストレージオブジェクトは、それぞれが専用のI/Oを持つ別々のディスクに常駐すると想定しています。

表 10.2 ネストしたRAIDレベル

RAIDレベル	説明	パフォーマンスと耐障害性
10 (1+0)	RAID 1(ミラー)アレイで構築されたRAID(ストライプ)	RAID 1+0は、高レベルのI/Oパフォーマンス、データ冗長性、およびディスク耐障害性を提供します。RAIDの各メンバーデバイスは個々にミラーリングされるので、エラーになったディスクのミラー先が異なる限り、複数ディスクの障害を許容し、データを使用することができます。

RAID レベル	説明	パフォーマンスと耐障害性
		オプションとして、ベースをなすミラーリングされたアレイごとにスペアを設定したり、すべてのミラーに対するスペアグループに対応するスペアを設定できます。
10 (0+1)	RAID 0(ストライプ)アレイで構築されたRAID 1(ミラー)	RAID 0+1は、高レベルのI/Oパフォーマンスとデータ冗長性を提供しますが、耐障害性が1+0より若干低くなります。ミラーの一方のサイドで複数のディスクがエラーになると、もう一方のミラーが使用可能になります。ただし、ミラーの両サイドで同時にディスクが失われると、すべてのデータが喪失します。 このソリューションは1+0ソリューションより耐障害性が低いですが、別のサイトで保守を実行したり、ミラーを保持する必要がある場合、ミラーのサイド全体をオフラインにしても、完全に機能するストレージデバイスを保持することができます。また、2つのサイト間の接続が失われた場合は、どちらかのサイトがもう一方のサイトから独立して稼動します。ミラーリングされたセグメントをストライプする場合はこうなりません。ミラーが低レベルで管理されているからです。 デバイスがエラーになると、RAID 0には耐障害性がないので、そのサイドのミラーがエラーになります。新しいRAID 0を作成して、エラーになったサイドに置き換え、次に、ミラーを再同期してください。

10.2.2 mdadmによるネストしたRAID 10 (1+0)デバイスの作成

ネストしたRAID 1+0は、2つ以上のRAID 1(ミラー)デバイスを作成し、それらのRAID 1デバイスをRAID 0のコンポーネントデバイスとして使用することで構築します。

重要項目

デバイスに対する複数の接続を管理する必要がある場合は、マルチパスI/Oを設定してから、RAIDデバイスを設定する必要があります。詳細については、第7章 *デバイスのマルチパスI/Oの管理* (57 ページ)を参照してください。

このセクションのプロシージャでは、次のテーブルに示すデバイス名を使用します。それらのデバイス名は、必ず、ご使用のデバイスの名前で変更してください。

表 10.3 *ネスティングでRAID 10(1+0)を作成するシナリオ*

rawデバイス	RAID 1(ミラー)	RAID 1+0(ストライピングミラー)
/dev/sdb1 /dev/sdc1	/dev/md0	/dev/md2
/dev/sdd1 /dev/sde1	/dev/md1	

1. 端末コンソールを開いて、rootユーザまたは同等の権限でログインします。

2. RAID 1デバイスごとに2つの異なるデバイスを使用して、2つのソフトウェアRAID 1デバイスを作成します。コマンドプロンプトで、次の2つのコマンドを入力します。

```
mdadm --create /dev/md0 --run --level=1 --raid-devices=2 /dev/sdb1
/dev/sdc1
```

```
mdadm --create /dev/md1 --run --level=1 --raid-devices=2 /dev/sdd1
/dev/sde1
```

3 ネストしたRAID 1+0デバイスを作成します。コマンドプロンプトで、「ステップ 2 (148 ページ)」で作成したソフトウェアRAID 1デバイスを使用して、次のコマンドを入力します。

```
mdadm --create /dev/md2 --run --level=0 --chunk=64 --raid-devices=2
/dev/md0 /dev/md1
```

デフォルトのチャンクサイズは 64KBです。

4 RAID 1+0デバイス/dev/md2上で、Reiser(reiserfs)などのファイルシステムを作成します。たとえば、コマンドプロンプトで、次のように入力します。

```
mkfs.reiserfs /dev/md2
```

これとは別のファイルシステムを使用したい場合は、コマンドを変更します。

5 /etc/mdadm.confファイルを編集して、コンポーネントデバイスとRAIDデバイス/dev/md2のエントリを追加します。

6 /etc/fstabファイルを編集して、RAID 1+0デバイス /dev/md2のエントリを追加します。

7 サーバを再起動します。

RAID 1+0デバイスが/localにマウントされます。

10.2.3 mdadmによるネストしたRAID 10 (0+1)デバイスの作成

ネストしたRAID 0+1は、2個から4個のRAID 0(ストライプ)デバイスで構築され、それらのRAID 0デバイスをミラーリングしてRAID 1のコンポーネントデバイスとします。

重要項目

デバイスに対する複数の接続を管理する必要がある場合は、マルチパスI/Oを設定してから、RAIDデバイスを設定する必要があります。詳細については、第7章 *デバイスのマルチパスI/Oの管理* (57 ページ)を参照してください。

この構成では、RAID 0がデバイスの喪失に耐えられないので、ベースのRAID 0デバイスにスペアデバイスを指定できません。デバイスがミラーの1つのサイドでエラーになった場合は、置き換え用のRAID 0デバイスを作成して、ミラーに追加します。

このセクションのプロシージャでは、次のテーブルに示すデバイス名を使用します。それらのデバイス名は、必ず、ご使用のデバイスの名前で変更してください。

表 10.4 *ネストによるRAID 10 (0+1)作成のシナリオ*

rawデバイス	RAID 0 (ストライプ)	RAID 0+1(ミラー化ストライピング)
/dev/sdb1 /dev/sdc1	/dev/md0	/dev/md2
/dev/sdd1 /dev/sde1	/dev/md1	

1. 端末コンソールを開いて、rootユーザまたは同等の権限で、ログインします。
2. RAID 0デバイスごとに2つの異なるデバイスを使用して、2つのソフトウェアRAID 0デバイスを作成します。コマンドプロンプトで、次の2つのコマンドを入力します。

```
mdadm --create /dev/md0 --run --level=0 --chunk=64 --raid-devices=2
/dev/sdb1 /dev/sdc1

mdadm --create /dev/md1 --run --level=0 --chunk=64 --raid-devices=2
/dev/sdd1 /dev/sde1
```

デフォルトのチャンクサイズは 64KBです。

3 ネストしたRAID 0+1デバイスの作成コマンドプロンプトで、「ステップ2(150ページ)」で作成したソフトウェアRAID 0デバイスを使用して、次のコマンドを入力します。

```
mdadm --create /dev/md2 --run --level=1 --raid-devices=2 /dev/md0
/dev/md1
```

4 RAID 0+1デバイス/dev/md2上で、Reiser(reiserfs)などのファイルシステムを作成します。たとえば、コマンドプロンプトで、次のように入力します。

```
mkfs.reiserfs /dev/md2
```

これとは別のファイルシステムを使用したい場合は、コマンドを変更します。

5 /etc/mdadm.confファイルを編集して、コンポーネントデバイスとRAIDデバイス/dev/md2のエントリを追加します。

6 /etc/fstabファイルを編集して、RAID 0+1デバイス/dev/md2のエントリを追加します。

7 サーバを再起動します。

RAID 0+1デバイスが/localにマウントされます。

10.3 mdadmによるコンプレックス RAID 10の作成

- 10.3.1項 「mdadm RAID10の理解」 (152 ページ)
- 10.3.2項 「mdadmによるRAID 10の作成」 (155 ページ)

10.3.1 mdadm RAID10の理解

mdadmでは、RAID10レベルで、RAID 0(ストライピング)とRAID 1(ミラーリング)の両方の機能を組み合わせた単一の複雑なソフトウェアRAIDが作成されます。すべてのデータブロックの複数のコピーが、ストライピングの規則に従って、複数のドライブ上に配置されます。コンポーネントデバイスは、すべて同じサイズにする必要があります。

- 「コンプレックスRAID10とネストしたRAID 10 (1+0)の比較」 (152 ページ)
- 「mdadm RAID10のレプリカ数」 (153 ページ)
- 「mdadm RAID10のデバイス数」 (153 ページ)
- 「nearレイアウト」 (154 ページ)
- 「farレイアウト」 (154 ページ)

コンプレックスRAID10とネストしたRAID 10 (1+0)の比較

コンプレックスRAID は、ネストしたRAID 10 (1+0)と目的は同じですが、次の点で異なります。

表 10.5 コンプレックスRAID 10対ネストしたRAID 10

機能	mdadm RAID10オプション	ネストしたRAID 10(1+0)
デバイスの数	偶数個または奇数個のコンポーネントデバイス	偶数個のコンポーネントデバイス
コンポーネントデバイス	単一のRAIDデバイスとして管理されます。	ネストしたRAIDデバイスとして管理されます。
ストライピング	ストライピングは、コンポーネントデバイス	ストライピングは、連続的に、すべてのコン

機能	mdadm RAID10オプション	ネストしたRAID 10(1+0)
	上にnearレイアウトまたはfarレイアウトを生じます。 farレイアウトでは、RAID 1ペアの数でなく、ドライブ数で増減するシーケンシャルな読み込みスループットを提供します。	ポーネントデバイスをまたぎます。
データの複数コピー	2からアレイ内のデバイス数まで	ミラーリングされたセグメントごとにコピー
ホットスペアデバイス	単一スペアですべてのコンポーネントデバイスに対応できます。	ベースをなすミラーリングされたアレイごとにスペアを設定したり、すべてのミラーに対応するスペアグループに対するスペアを設定できます。

mdadm RAID10のレプリカ数

mdadm RAID10アレイの設定時に、データブロックごとに必要なレプリカ数を指定する必要があります。デフォルトのレプリカ数は2ですが、2からアレイ内のデバイス数まで可能です。

mdadm RAID10のデバイス数

少なくとも、指定のレプリカ数と同数のコンポーネントデバイスを使用する必要があります。ただし、RAID10アレイのコンポーネントデバイス数は各データブロックのレプリカ数の倍数である必要はありません。有効なストレージサイズは、デバイス数をレプリカ数で割った数です。

たとえば、5個のコンポーネントデバイスで作成したアレイに2つのレプリカを指定した場合は、各ブロックのコピーが2つの異なるデバイスに保存されます。したがって、すべてのデータの1コピーの有効なストレージサイズは、5/2(つまり、コンポーネントデバイスのサイズの2.5倍)となります。

nearレイアウト

nearレイアウトでは、異なるコンポーネントデバイス上で、データブロックのコピーが互いに接近してストライプされます。つまり、あるデータブロックの複数のコピーが異なるデバイス内で同様にオフセットされます。nearは、RAID10のデフォルトレイアウトです。たとえば、奇数個のコンポーネントデバイスとデータの2コピーを使用する場合は、一部のコピーが、1チャンク分、デバイス内を前進します。

`mdadm` RAID 10のnearレイアウトは、半数のドライブ上のRAID 0と同様の読み書きパフォーマンスを提供します。

偶数個のディスクと2つのレプリカを使用したnearレイアウト

```
sda1 sdb1 sdc1 sde1
  0    0    1    1
  2    2    3    3
  4    4    5    5
  6    6    7    7
  8    8    9    9
```

奇数個のディスクと2つのレプリカを使用したnearレイアウト

```
sda1 sdb1 sdc1 sde1 sdf1
  0    0    1    1    2
  2    3    3    4    4
  5    5    6    6    7
  7    8    8    9    9
  10   10   11   11   12
```

farレイアウト

farレイアウトは、すべてのドライブの前半部分にデータをストライプし、次に、2つ目のデータコピーをすべてのドライブの後半部分にストライプして、ブロックのすべてのコピーが異なるドライブに配置されるようにします。値の2つ目のセットは、コンポーネントドライブの中ほどから開始します。

farレイアウトでは、mdadm RAID10の読み込みパフォーマンスは、すべてのドライブを使用したRAID 0と同様ですが、書き込みパフォーマンスは、ドライブヘッドのシーク回数が増えるので、RAID 0よりかなり遅くなります。このレイアウトは、 読み込み専用ファイルサーバなどの、読み込み集約型操作に最適です。

raid10の書き込み速度は、付近のレイアウトを使用しているraid1やraid0などの他のミラーリングRAIDの種類と同等です。これは、ファイルシステムのエレベータが生の書き込みよりも効率のよい書き込みのスケジュールを行うためです。Raid10をfarレイアウトで使用する方法は、ミラーリングによる書き込みアプリケーションに適しています。

偶数個のディスクと2つのレプリカを使用したfarレイアウト

```
sda1 sdb1 sdc1 sde1
  0    1    2    3
  3    5    6    7
  . . .
  3    1    2    3
  7    4    5    6
```

奇数個のディスクと2つのレプリカを使用したfarレイアウト

```
sda1 sdb1 sdc1 sde1 sdf1
  0    1    2    3    4
  5    6    7    8    9
  . . .
  4    0    1    2    3
  9    5    6    7    8
```

10.3.2 mdadmによるRAID 10の作成

mdadmのRAID10オプションでは、ネストなしのRAID 10デバイスが作成されます。RAID10-については、「10.3項 「mdadmによるコンプレックスRAID 10の作成」 (151 ページ)」を参照してください。

このセクションのプロシージャでは、次のテーブルに示すデバイス名を使用します。それらのデバイス名は、必ず、ご使用のデバイスの名前で変更してください。

表 10.6　mdadm RAID10オプションでRAID 10を作成するシナリオ

rawデバイス	RAID10(near/farストライピングスキーム)
/dev/sdf1	/dev/md3
/dev/sdg1	
/dev/sdh1	
/dev/sdi1	

1. YaSTで、RAIDで使用したいデバイス(/dev/sdf1、/dev/sdg1、/dev/sdh1、/dev/sdi1など)に0xFD Linux RAIDパーティションを作成します。
2. 端末コンソールを開いて、rootユーザまたは同等の権限で、ログインします。
3. RAID 10 コマンドを作成します。コマンドプロンプトで、次のように入力します(すべて同じ行)。

   ```
   mdadm --create /dev/md3 --run --level=10 --chunk=4 --raid-devices=4
   /dev/sdf1 /dev/sdg1 /dev/sdh1 /dev/sdi1
   ```

4. RAID 10デバイス /dev/md3 上にReiserファイルシステムを作成します。コマンドプロンプトで、次のように入力します

   ```
   mkfs.reiserfs /dev/md3
   ```

5. /etc/mdadm.confファイルを編集して、コンポーネントデバイスとRAIDデバイス/dev/md3のエントリを追加します。次に例を示します。

   ```
   DEVICE /dev/md3
   ```

6. /etc/fstabファイルを編集して、RAID 10デバイス /dev/md3のエントリを追加します。

7 サーバを再起動します。

RAID 10デバイスが/raid10にマウントされます。

10.4 ディグレードアレイの作成

ディグレードアレイは、一部のデバイスが欠けたアレイです。ディグレードアレイは、RAID 1、RAID 4、RAID 5、およびRAID 6に対してのみサポートされています。これらのRAIDタイプは、その耐障害性機能として、一部のデバイスの欠落に耐えるように設計されています。通常、デバイスに障害が発生すると、ディグレードアレイが生成されます。ディグレードアレイは、意図的に作成するすることもできます。

RAIDの種類	許容可能な欠落スロット数
RAID 1	1つ以外の全スロット
RAID 4	1スロット
RAID 5	1スロット
RAID 6	1個または2個のスロット

一部のデバイスが欠落したディグレードアレイを作成するには、単に、デバイス名の代わりにmissingというワードを指定します。この指定により、mdadmは、アレイ内の対応するスロットを空のまま残します。

RAID5アレイの作成時に、mdadmによって、余分なスペアドライブを持つディグレードアレイが自動的に作成されます。これは、一般に、ディグレードアレイ内にスペアを構築した方が、ディグレードアレイではないが正常でないアレイ上でパリティを再同期するより高速なためです。この機能は、--forceオプションで無効にできます。

RAIDを作成したいが、使用するデバイスの1つに既にデータが入っている場合は、ディグレードアレイを作成すると便利なことがあります。その場合は、他のデバイスでディグレードアレイを作成し、その使用中のデバイスからのデータをディグレードモードで実行中のRAIDにコピーし、デバイスをRAID

に追加して、RAIDの再構築まで待機すると、データがすべてのデバイスに行き渡ります。このプロセスの例を、次のプロシージャで示します。

1 ディグレードRAID 1デバイス/dev/md0を単一ドライブ/dev/sd1で作成します。コマンドプロンプトで、次のように入力してください。

```
mdadm --create /dev/md0 -l 1 -n 2 /dev/sda1 missing
```

追加先のデバイスは、追加するデバイスと同じか、またはそれ以上のサイズを持つ必要があります。

2 ミラーに追加したいデバイスに、RAIDアレイに移動したいデータが含まれている場合は、この時点で、そのデータを、ディグレードモードで実行中のRAIDアレイにコピーします。

3 デバイスをミラーに追加します。たとえば、add /dev/sdb1をRAIDに追加するには、コマンドプロンプトで、次のように入力します。

```
mdadm /dev/md0 -a /dev/sdb1
```

一度に1つのデバイスしか追加できます。カーネルがミラーを構築し、完全にオンラインにした後、別のミラーを追加できます。

4 構築の進捗状況を監視するには、コマンドプロンプトで、次のように入力します。

```
cat /proc/mdstat
```

毎秒更新されている間に再構築の進捗を確認するには、次のように入力します。

```
watch -n 1 cat /proc/mdstat
```

mdadmによるソフトウェアRAIDアレイのサイズ変更

11

このセクションでは、ソフトウェアRAID 1、4、5、または6のデバイスのサイズを複数デバイス管理(mdadm(8))ツールで増減する方法について説明します。

警告

このセクションに示されたタスクを開始する前に、すべてのデータに有効なバックアップがあることを確認してください。

11.1 サイズ変更プロセスの理解

既存のソフトウェアRAIDデバイスのサイズ変更には、各コンポーネントパーティションが提供するスペースの増減が必要です。

11.1.1 ソフトウェアRAIDサイズ変更のガイドライン

mdadm(8)ツールは、ソフトウェアRAIDレベル1、4、5、および6に対してだけサイズ変更をサポートします。これらのRAIDレベルには耐障害性があるので、一度に1つづつ、サイズ変更するコンポーネントパーティションを削除できます。原則として、RAIDパーティションのホットリサイズが可能ですが、その場合は、データの保全に特に注意する必要があります。

デバイス上の使用可能スペースで変更内容を利用するために、RAIDに常駐するファイルシステムもサイズ変更できる必要があります。SUSE® Linux Enterprise Server 11では、Ext2、Ext3、およびReiserFSに関して、ファイルシステムのサイズ変更用ユーティリティを使用できます。このユーティリティは、次のようにサイズの増減をサポートします。

表 11.1　ファイルシステムサイズ変更のサポート

ファイルシステム	ユーティリティ	サイズを増加	サイズを減少
Ext2またはExt3	resize2fs	はい(ただし、オフラインのみ)	はい(ただし、オフラインのみ)
ReiserFS	resize_reiserfs	はい(オンラインまたはオフライン)	はい(ただし、オフラインのみ)

パーティションまたはファイルシステムのサイズ変更には、データを失う可能性をはらむリスクが伴います。

警告

データの喪失を避けるには、データをバックアップしてから、サイズ変更タスクを開始します。

11.1.2 タスクの概要

RAIDのサイズ変更には、次のようなタスクがあります。タスクの実行順序は、サイズを増加するか、減少するかによって異なります。

*表 **11.2** RAIDのサイズ変更に必要なタスク*

仕事	説明	サイズを増大させる場合の順序	サイズを減少させる場合の順序
各コンポーネントパーティションのサイズを変更します。	各コンポーネントパーティションのアクティブなサイズを増加または減少します。一度に1つのコンポーネントパーティションだけを削除し、そのサイズを変更してから、パーティションをRAIDに戻します。	1	2
ソフトウェアRAID自体をサイズ変更します。	RAIDは、ベースのコンポーネントパーティションの増減を自動的には認識しません。したがって、RAIDに新しいサイズを知らせる必要があります。	2	3
ファイルシステムのサイズを変更します。	RAIDに常駐するファイルシステムをサイズ変更する必要があります。これは、サイズ変更のツールを提供するファイルシステムの場合のみ可能です（Ext2、Ext3、ReiserFSなど)。	3	1

11.2 ソフトウェアRAIDのサイズの増加

始める前に、「11.1項「サイズ変更プロセスの理解」 (159 ページ)」のガイドラインをレビューしてください。

- 11.2.1項 「コンポーネントパーティションのサイズの増加」 (162 ページ)
- 11.2.2項 「RAIDアレイのサイズの増加」 (164 ページ)
- 11.2.3項 「ファイルシステムのサイズの増加」 (166 ページ)

11.2.1 コンポーネントパーティションのサイズの増加

RAID1、4、5、または6のサイズを増加するには、このセクションのプロシージャを適用します。RAID内のコンポーネントパーティションごとに、RAIDからパーティションを削除し、そのサイズを変更し、パーティションをRAIDに戻し、RAIDが安定するまで待機してから続行します。パーティションが削除されている間、RAIDはディグレードモードで動作し、ディスクの耐障害性がまったくないか、または低下しています。複数の同時ディスク障害を許容できるRAIDの場合でも、一度に2つ以上のパーティションを削除しないでください。

警告

RAIDに、ディスクの耐障害性がないか、単に一貫性がない場合、パーティションのどれかを削除すると、データが失われます。パーティションの削除は注意深く行い、必ず、データのバックアップをとってください。

このセクションのプロシージャでは、次のテーブルに示すデバイス名を使用します。それらの名前は、必ず、ご使用のデバイスの名前に変更してください。

表 11.3　コンポーネントパーティションのサイズを増加するシナリオ

RAIDデバイス	コンポーネントパーティション
/dev/md0	/dev/sda1
	/dev/sdb1
	/dev/sdc1

RAID用コンポーネントパーティションのサイズを増加するには:

1. 端末コンソールを開いて、rootユーザまたは同等の権限でログインします。
2. 次のように入力して、RAIDアレイが一貫性を保っており、同期されていることを確認します。

   ```
   cat /proc/mdstat
   ```

   このコマンドの出力によって、RAIDアレイがまだ同期中と分かる場合は、同期化の完了まで待って、続行してください。
3. コンポーネントパーティションの1つをRAIDアレイから削除します。たとえば、次のように入力して、/dev/sda1を削除します。

   ```
   mdadm /dev/md0 --fail /dev/sda1 --remove /dev/sda1
   ```

   成功するためには、failとremoveの両方のアクションが行われる必要があります。
4. 次のオプションの1つを実行して、「ステップ 3 (163 ページ)」で削除したパーティションのサイズを増加します。

   - fdisk(8)、cfdisk(8)、parted(8)などのディスクパーティショナを使用して、パーティションのサイズを増加します。通常は、このオプションが選択されます。
   - パーティションの常駐ディスクを、容量のより大きいデバイスに置き換えます。

このオプションは、元ディスクの他のファイルシステムがシステムによりアクセスされない場合だけ選択できます。置き換え用デバイスをRAIDに追加すると、元のデバイスにあったデータをすべて再構築しなければならないので、データの同期にはるかに長い時間がかかります。

5 パーティションをRAIDアレイに再追加します。たとえば、次のように入力して、/dev/sda1を追加します。

```
mdadm –a /dev/md0 /dev/sda1
```

RAIDが同期され、一貫性を持つまで待機してから、次のパーティションの処理に進みます。

6 アレイ内の残りのコンポーネントデバイスごとに、「ステップ2(163ページ)」から「ステップ5 (164 ページ)」まで繰り返します。必ず、コマンドを変更して、正しいコンポーネントパーティションを使用してください。

7 カーネルがRAIDのパーティションテーブルを再読み込みできないというメッセージが表示されたら、すべてのパーティションのサイズ変更後にコンピュータを再起動して、パーティションテーブルの更新を強制する必要があります。

8 11.2.2項 「RAIDアレイのサイズの増加」 (164 ページ)に進みます。

11.2.2 RAIDアレイのサイズの増加

RAID内の各コンポーネントパーティションのサイズ変更後(「11.2.1項 「コンポーネントパーティションのサイズの増加」 (162 ページ)」参照)も、新しい使用可能スペースの認識を強制するまで、RAIDアレイの設定では、元のアレイサイズが使用され続けます。RAIDアレイのサイズを指定したり、使用可能な最大スペースを使用できます。

このセクションのプロシージャでは、RAIDデバイスのデバイス名として /dev/md0を使用しています。必ず、この名前は変更して、ご使用のデバイスの名前を使用してください。

1 端末コンソールを開いて、rootユーザまたは同等の権限でログインします。

2 次のように入力して、アレイのサイズとアレイに認識されるデバイスサイズをチェックします。

```
mdadm -D /dev/md0 | grep -e "Array Size" -e "Device Size"
```

3 次のいずれかの操作を行います。

- 次のように入力して、アレイサイズを使用可能な最大サイズまで増加します。

  ```
  mdadm --grow /dev/md0 -z max
  ```

- 次のように入力して、アレイサイズを指定の値まで増加します。

  ```
  mdadm --grow /dev/md0 -z size
  ```

  *size*を、キロバイト(1キロバイトは1024バイト)単位で目的のサイズを表す整数値で置き換えます。

4 次のように入力して、アレイのサイズとアレイに認識されるデバイスサイズを再チェックします。

```
mdadm -D /dev/md0 | grep -e "Array Size" -e "Device Size"
```

5 次のいずれかの操作を行います。

- アレイのサイズ変更が成功していたら、「11.2.3項 「ファイルシステムのサイズの増加」 (166 ページ)」を続行します。
- アレイが予期どおりにサイズ変更されていない場合は、いったん再起動してから、このプロシージャを再試行する必要があります。

11.2.3 ファイルシステムのサイズの増加

アレイサイズの増加後は(「11.2.2項 「RAIDアレイのサイズの増加」 (164 ページ)」参照)、ファイルシステムのサイズ変更ができます。

ファイルシステムのサイズを使用可能な最大スペースまで増加したり、正確なサイズを指定できます。ファイルシステムに正確なサイズを指定する場合は、その新しいサイズが次の条件を満たすかどうか確認してください。

- 新しいサイズは、既存データのサイズより大きくなければなりません。さもないと、データが失われます。
- ファイルシステムのサイズは使用可能なスペースより大きくできないので、新しいサイズは、現在のRAIDサイズ以下でなければなりません。

Ext2またはExt3

Ext2とExt3のファイルシステムは、`resize2fs`コマンドでマウントまたはアンマウントする際にサイズ変更できます。

1. 端末コンソールを開いて、`root`ユーザまたは同等の権限でログインします。
2. 次の方法の1つで、ファイルシステムのサイズを増加します。
   - ファイルシステムのサイズを、`/dev/md0`と呼ばれるソフトウェアRAIDデバイスの最大使用可能サイズまで拡張するには、次のコマンドを入力します。

     ```
     resize2fs /dev/md0
     ```

     sizeパラメータを指定しない場合、サイズはパーティションのサイズにデフォルト設定されます。
   - ファイルシステムを特定のサイズに拡張するには、次のコマンドを入力します。

     ```
     resize2fs /dev/md0 size
     ```

*size*パラメータは、要求されたファイルシステムの新サイズを指定します。単位を指定しない場合のsizeパラメータの単位は、ファイルシステムのブロックサイズです。オプションとして、sizeパラメータの後ろに、次の単位指定子の1つを付けることができます。sは512バイトのセクタ、Kはキロバイト(1キロバイトは1024バイト)、Mはメガバイト、Gはギガバイトを表します。

サイズ変更が完了するまで待って、続行します。

3 ファイルシステムがマウントされていない場合は、この時点で、ファイルシステムをマウントします。

たとえば、Ext2ファイルシステムを、/dev/md0という名前のRAIDに、マウントポイント/raidでマウントするには、次のように入力します。

```
mount -t ext2 /dev/md0 /raid
```

4 次のように入力して、マウントされたファイルシステムに対するサイズ変更の効果をチェックします。

```
df -h
```

ディスクフリー(df)コマンドは、ディスクの合計サイズ、使用されたブロック数、およびファイルシステム上の使用可能なブロック数を表示します。-hオプションは、読みやすい形式でサイズを出力します(1K,、234M、2Gなど)。

ReiserFS

Ext2およびExt3と同様に、ReiserFSファイルシステムは、マウントまたはアンマウント時にサイズを増加できます。サイズ変更は、RAIDアレイのブロックデバイス上で行われます。

1 端末コンソールを開いて、rootユーザまたは同等の権限でログインします。

2 /dev/md0と呼ばれるソフトウェアRAIDデバイスのファイルシステムのサイズを、次の方法の1つを使用して変更します。

- ファイルシステムのサイズをデバイスの使用可能な最大サイズまで拡張するには、次のように入力します。

  ```
  resize_reiserfs /dev/md0
  ```

  サイズを指定しないと、ボリュームはパーティションのフルサイズまで拡張されます。

- ファイルシステムを特定のサイズに拡張するには、次のコマンドを入力します。

  ```
  resize_reiserfs -s size /dev/md0
  ```

  *size*を目的のサイズ(バイト単位)で置き換えます。50000K(キロバイト)、250M(メガバイト)、2G(ギガバイト)など、値の単位を指定することもできます。代わりに、プラス(+)記号を値の前に付けることにより、現在のサイズに対する増加を指定することもできます。たとえば、次のコマンドは、/dev/md0上のファイルシステムのサイズを500 MB分増加します。

  ```
  resize_reiserfs -s +500M /dev/md0
  ```

サイズ変更が完了するまで待って、続行します。

3 ファイルシステムがマウントされていない場合は、この時点で、ファイルシステムをマウントします。

たとえば、ReiserFSファイルシステムを、/dev/md0というRAIDに、マウントポイント/raidでマウントするには、次のように入力します。

```
mount -t reiserfs /dev/md0 /raid
```

4 次のように入力して、マウントされたファイルシステムに対するサイズ変更の効果をチェックします。

```
df -h
```

ディスクフリー(df)コマンドは、ディスクの合計サイズ、使用されたブロック数、およびファイルシステム上の使用可能なブロック数を表示し

ます。-hオプションは、読みやすい形式でサイズを出力します(1K,、234M、2Gなど)。

11.3 ソフトウェアRAIDのサイズの削減

始める前に、「11.1項「サイズ変更プロセスの理解」 (159 ページ)」のガイドラインをレビューしてください。

- 11.3.1項 「ファイルシステムのサイズの削減」 (169 ページ)
- 11.3.2項 「コンポーネントパーティションのサイズの削減」 (172 ページ)
- 11.3.3項 「RAIDアレイサイズの削減」 (174 ページ)

11.3.1 ファイルシステムのサイズの削減

RAIDデバイス上のファイルシステムのサイズを削減する際には、新しいサイズが次の条件を満たすかどうか確認してください。

- 新しいサイズは、既存データのサイズより大きくなければなりません。さもないと、データが失われます。
- ファイルシステムのサイズは使用可能なスペースより大きくできないので、新しいサイズは、現在のRAIDサイズ以下でなければなりません。

SUSE Linux Enterprise Server SP1では、Ext2、Ext3、およびReiserFSだけが、ファイルシステムのサイズを減少させるユーティリティを提供します。以降の該当するプロシージャを使用して、ファイルシステムのサイズを減少させてください。

このセクションのプロシージャでは、ＲＡＩＤデバイスのデバイス名として`//dev/md0`を使用しています。必ず、コマンドを変更して、ご使用のデバイスの名前を使用してください。

Ext2またはExt3

Ext2とExt3のファイルシステムは、マウントまたはアンマウント時にサイズ変更できます。

1. 端末コンソールを開いて、`root`ユーザまたは同等の権限でログインします。
2. RAID上のファイルシステムのサイズを減少させるには、次のコマンドを入力します。

   ```
   resize2fs /dev/md0 <size>
   ```

   *size*を、目的のサイズを表す整数値(キロバイト単位)で置き換えます(1キロバイトは1024バイト)。

   サイズ変更が完了するまで待って、続行します。
3. ファイルシステムがマウントされていない場合は、この時点で、ファイルシステムをマウントします。たとえば、Ext2ファイルシステムを、`/dev/md0`という名前のRAIDに、マウントポイント`/raid`でマウントするには、次のように入力します。

   ```
   mount -t ext2 /dev/md0 /raid
   ```
4. 次のように入力して、マウントされたファイルシステムに対するサイズ変更の効果をチェックします。

   ```
   df -h
   ```

   ディスクフリー(`df`)コマンドは、ディスクの合計サイズ、使用されたブロック数、およびファイルシステム上の使用可能なブロック数を表示します。-hオプションは、読みやすい形式でサイズを出力します(1K,、234M、2Gなど)。

ReiserFS

ReiserFSファイルシステムのサイズは、ボリュームがアンマウントされる場合のみ、減少させることができます。

1 端末コンソールを開いて、rootユーザまたは同等の権限でログインします。

2 次のコマンドを入力して、デバイスをアンマウントします。

```
umount /mnt/point
```

サイズを削減するパーティションにシステムファイル(ルート/ボリュームなど)が含まれていると、ブート可能なCDまたはフロッピーからブートする場合のみ、アンマウントが可能です。

3 /dev/md0と呼ばれるソフトウェアRAIDデバイス上のファイルシステムのサイズを減らすには、次のように入力します。

```
resize_reiserfs -s size /dev/md0
```

*size*を目的のサイズ(バイト単位)で置き換えます。50000K(キロバイト)、250M(メガバイト)、2G (ギガバイト)など、値の単位を指定することもできます。代わりに、マイナス(-)記号を値の前に付けて、現在のサイズに対する削減分を指定することもできます。たとえば、次のコマンドは、/dev/md0上のファイルシステムのサイズを 500 MB分減少させます。

```
resize_reiserfs -s -500M /dev/md0
```

サイズ変更が完了するまで待って、続行します。

4 次のコマンドで、ファイルシステムをマウントします。

```
mount -t reiserfs /dev/md0 /mnt/point
```

5 次のように入力して、マウントされたファイルシステムに対するサイズ変更の効果をチェックします。

```
df -h
```

ディスクフリー(df)コマンドは、ディスクの合計サイズ、使用されたブロック数、およびファイルシステム上の使用可能なブロック数を表示します。-hオプションは、読みやすい形式でサイズを出力します(1K,、234M、2Gなど)。

11.3.2 コンポーネントパーティションのサイズの削減

一度に1つのRAIDのコンポーネントパーティションをサイズ変更します。コンポーネントパーティションごとに、そのパーティションをRAIDから削除し、パーティションサイズを変更し、パーティションをRAIDに戻したら、RAIDが安定するまで待機します。パーティションが削除されている間、RAIDはディグレードモードで動作し、ディスクの 耐障害性がまったくないか、または低下しています。複数の同時ディスクエラーに耐えるRAIDの場合でも、一度に2つ以上のコンポーネントパーティションを削除しないでください。

警告

RAIDに、ディスクの耐障害性がないか、単に一貫性がない場合、パーティションのどれかを削除すると、データが失われます。パーティションの削除は注意深く行い、必ず、データのバックアップをとってください。

このセクションのプロシージャでは、次のテーブルに示すデバイス名を使用します。必ず、コマンドを変更して、ご使用のデバイスの名前を使用してください。

表 11.4 *コンポーネントパーティションのサイズを増加するシナリオ*

RAIDデバイス	コンポーネントパーティション
/dev/md0	/dev/sda1
	/dev/sdb1
	/dev/sdc1

RAIDのコンポーネントパーティションをサイズ変更するには:

1. 端末コンソールを開いて、rootユーザまたは同等の権限でログインします。
2. 次のように入力して、RAIDアレイが一貫性を保っており、同期されていることを確認します。

```
cat /proc/mdstat
```

このコマンドの出力によって、RAIDアレイがまだ同期中と分かる場合は、同期化の完了まで待って、続行してください。

3 コンポーネントパーティションの1つをRAIDアレイから削除します。たとえば、次のように入力して、/dev/sda1を削除します。

```
mdadm /dev/md0 --fail /dev/sda1 --remove /dev/sda1
```

成功するためには、failとremoveの両方のアクションが行われる必要があります。

4 次のオプションの1つを実行して、「ステップ 3 (163 ページ)」で削除したパーティションのサイズを増加します。

- fdisk、cfdisk、partedなどのディスクパーティショナを使用して、パーティションのサイズを増加します。
- パーティションの常駐ディスクを別のデバイスで置き換えます。

  このオプションは、元ディスクの他のファイルシステムがシステムによりアクセスされない場合だけ選択できます。置き換え用デバイスをRAIDに追加すると、データの同期にはるかに長い時間がかかります。

5 パーティションをRAIDアレイに再追加します。たとえば、次のように入力して、/dev/sda1を追加します。

```
mdadm -a /dev/md0 /dev/sda1
```

RAIDが同期され、一貫性を持つまで待機してから、次のパーティションの処理に進みます。

6 アレイ内の残りのコンポーネントデバイスごとに、「ステップ2(163ページ)」から「ステップ 5 (164 ページ)」まで繰り返します。必ず、コマンドを変更して、正しいコンポーネントパーティションを使用してください。

7 カーネルがRAIDのパーティションテーブルを再読み込みできないというメッセージが表示されたら、すべてのパーティションのサイズ変更後にコンピュータを再起動する必要があります。

8 11.3.3項 「RAIDアレイサイズの削減」 (174 ページ)に進みます。

11.3.3 RAIDアレイサイズの削減

RAID内の各コンポーネントパーティションのサイズ変更後も、新しい使用可能スペースの認識を強制するまで、RAIDアレイの設定では、元のアレイサイズが使用され続けます。--growオプションを使用して、使用可能なディスクスサイズの変更を読み込むように強制してください。RAIDアレイのサイズを指定したり、使用可能な最大スペースを使用できます。

このセクションのプロシージャでは、RAIDデバイスのデバイス名として `/dev/md0`を使用しています。必ず、コマンドを変更して、ご使用のデバイスの名前を使用してください。

1 端末コンソールを開いて、`root`ユーザまたは同等の権限でログインします。

2 次のように入力して、アレイのサイズとアレイに認識されるデバイスサイズをチェックします。

```
mdadm -D /dev/md0 | grep -e "Array Size" -e "Device Size"
```

3 次のいずれかの操作を行います。

- 次のコマンドで、アレイのサイズを使用可能な最大サイズまで減少させます。

  ```
  mdadm --grow /dev/md0 -z max
  ```

- 次のコマンドで、アレイサイズを指定の値まで減少させます。

  ```
  mdadm --grow /dev/md0 -z size
  ```

*size*を、目的のサイズを表す整数値(キロバイト単位)で置き換えます(1キロバイトは1024バイト)。

4 次のように入力して、アレイのサイズとアレイに認識されるデバイスサイズを再チェックします。

```
mdadm –D /dev/md0 | grep –e "Array Size" –e "Device Size"
```

5 次のいずれかの操作を行います。

- アレイのサイズ変更が成功していたら、作業完了です。
- アレイが予期どおりにサイズ変更されていない場合は、いったん再起動してから、このプロシージャを再試行する必要があります。

12

Linux用iSNS

ストレージエリアネットワーク(SAN)には、複数のネットワークにまたがる多数のディスクドライブを使用できます。これによって、デバイス検出とデバイスの所有権の判定が難しくなります。iSCSIイニシエータはSANのストレージリソースを識別し、どれにアクセスできるか判定できる必要があります。

iSNS(Internet Storage Name Service)は、標準に基づくサービスであり、SUSE Linux Enterprise Server (SLES) 10 Support Pack 2から利用可能になりました。SNSでは、TCP/IPネットワーク上のiSCSIデバイスの自動的な検出、管理、および設定を容易にします。iSNSでは、ファイバチャネルネットワークと同等の知的なストレージディスカバリと管理サービスを提供します。

重要項目

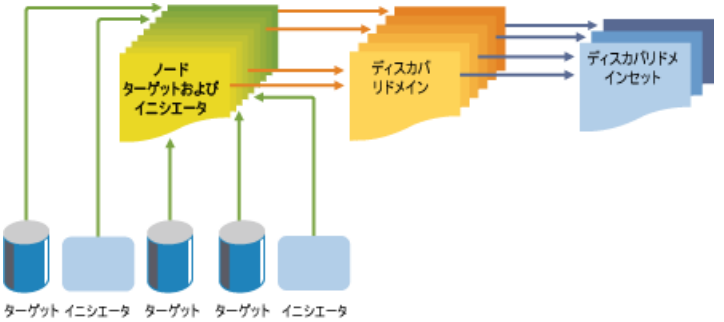
iSNSは安全な社内ネットワークだけで使用してください。

12.1 iSNSのしくみ

iSCSIイニシエータがiSCSIターゲットを検出するには、ネットワークのどのデバイスがストレージリソースで、アクセスするにはどのIPアドレスが必要かを特定する必要があります。iSNSサーバへクエリすると、iSCSIターゲットとイニシエータがアクセス許可を持つIPアドレスのリストが返されます。

iSNSを使用してiSNS検出ドメインと検出ドメインセットを作成します。次に、iSCSIターゲットとイニシエータを検出ドメインにグループ化またはまとめて、検出ドメインを検出ドメインセットにグループ化します。多くのストレージノードを複数のドメインに振り分けることで、各ホストの検出プロセスをiSNSで登録された最適なターゲットのサブセットに限定でき、これによって、不要な検出を削減し、各ホストが検出関係の確立に費やす時間を制限することで、ストレージネットワークの規模を調整できるようになります。このようにして、ディスカバリ対象のターゲットとイニシエータの数を制御し、簡略化できます。

図 12.1　iSNS検出ドメインと検出ドメインセット



iSCSIターゲットとiSCSIイニシエータは両方とも、iSNSクライアントを使用して、iSNSプロトコルによるiSNSサーバとのトランザクションを開始します。iSCSIターゲットとiSCSIイニシエータは、次にデバイス属性情報を共通検出ドメインに登録し、その他の登録されたクライアント情報をダウンロードし、検出ドメインで発生したイベントの非同期通知を受け取ります。

iSNSサーバは、iSNSプロトコルクエリとiSNSクライアントがiSNSプロトコルを使用して作成した要求に応答します。iSNSサーバはiSNSプロトコル状態変更通知を開始し、登録要求から送られてきた適切に認証された情報をiSNSデータベースに保存します。

Linux向けiSNSには、次のようなメリットがあります。

- ネットワーク接続させたストレージ資産の登録、検出、管理に役立つ情報を提供する。
- DNSインフラストラクチャと統合する。
- iSCSIストレージの登録、検出、管理を統合する。
- ストレージ管理の実装が簡素化される。
- その他のディスカバリ方法よりもスケーラビリティが向上する。

次のシナリオは、iSNSのメリットについて具体的に説明したものです。

100個のiSCSIイニシエータと100個のiSCSIターゲットが会社にあるとします。設定によっては、すべてのiSCSIイニシエータが100個のiSCSIターゲットを検出して接続しようとする可能性があります。このため、検出や接続が困難になる場合があります。イニシエータとターゲットをいくつかの検出ドメインにグループ化することで、ある部門のiSCSIイニシエータが別の部門のiSCSIターゲットを検出しないようにできます。その結果、特定部門のiSCSIイニシエータは、その部門の検出ドメインに属するiSCSIターゲットしか検出しません。

12.2 Linux用iSNSサーバのインストール

SLES 10 SP2以降のバージョンには、Linux用iSCSIサーバが含まれていますが、このサーバは、デフォルトでは、インストールも設定もされません。したがって、iSNSパッケージモジュール(`isns`モジュールおよび`yast2-isns`モジュール)をインストールし、iSNSサービスを設定する必要があります。

注記

iSNSは、iSCSIターゲットまたはiSCSIイニシエータのソフトウェアがインストールされる同じサーバにインストールできます。ただし、iSCSIターゲットソフトウェアとiSCSIイニシエータソフトウェアの両方を同じサーバにインストールすることはできません。

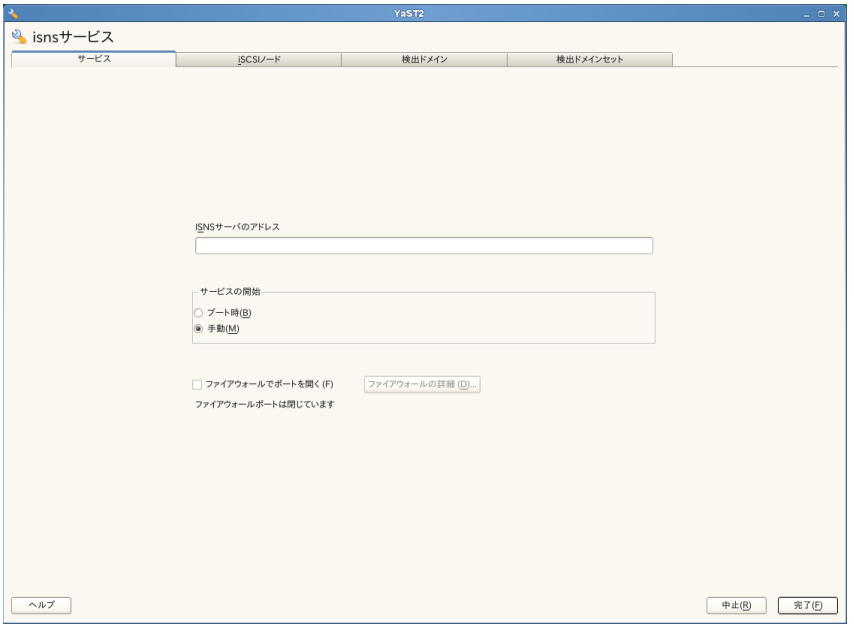
Linux向けiSNSをインストールするには、次の手順に従います。

1. YaSTを起動して、［ネットワークサービス］［*iSNS*サーバ］を選択します。

2. `isns`パッケージのインストールを促されたら、［*Install*（インストール）］をクリックします。

3. インストールダイアログの指示に従って、SUSE Linux Enterprise Server 11のインストールディスクを挿入します。

   インストールが完了すると、iSNSサービスの設定ダイアログが、自動的に、［サービス］タブを開いた状態で表示されます。



4. ［*iSNS*サーバのアドレス］で、iSNSサーバのDNS名またはIPアドレスを指定します。

5. ［*Service Start(サービスの開始)*］で、次のオプションの1つを選択します。

- **ブート時:** iSNSサービスは、サーバの起動時に自動的に開始します。
- **手動(デフォルト):** サービスを開始するには、iSNSのインストール先サーバのサーバコンソールで、`rcisns start`または`/etc/init.d/isns start`を入力する必要があります。

6 次のファイアウォール設定を指定します。

- **Open Port in Firewall(ファイアウォールのポートを開く):** このチェックボックスを選択して、ファイアウォールを開き、リモートコンピュータからサービスにアクセスできるようにします。ファイアウォールのポートは、デフォルトでは閉じています。
- **ファイアウォールの詳細:** ファイアウォールのポートを開いた場合、デフォルトでは、ポートがすべてのネットワークインタフェースで開きます。ポートを開くインタフェースを選択するには、［*Firewall Details(ファイアウォールの詳細)*］をクリックし、使用するネットワークインタフェースを選択し、次に、［*OK*］をクリックします。

7 Click［完了］をクリックして、設定を適用し、インストールを完了します。

8 12.3項 「iSNS検出ドメインの設定」 (181 ページ)に進みます。

12.3 iSNS検出ドメインの設定

iSCSIイニシエータおよびターゲットでiSNSサービスを使用するには、これらが検出ドメインに属している必要があります。

重要項目

iSNS検出ドメインを設定するには、iSNSサービスがインストール済みで、実行されている必要があります。詳細については、12.4項 「iSNSの起動」 (187 ページ)を参照してください。

- 12.3.1項 「iSNS検出ドメインの作成」 (182 ページ)
- 12.3.2項 「iSNS検出ドメインセットの作成」 (183 ページ)
- 12.3.3項 「iSCSIノードの検出ドメインへの追加」 (185 ページ)
- 12.3.4項 「検出ドメインの検出ドメインセットへの追加」 (187 ページ)

12.3.1 iSNS検出ドメインの作成

iSNSサービスをインストールすると、*デフォルトDD*というデフォルトの検出ドメインが自動的に作成されます。iSNSを使用するように設定されている既存のiSCSIターゲットとイニシエータは、デフォルト検出ドメインに自動的に追加されます。

新しい検出ドメインを作成するには、次の手順に従います。

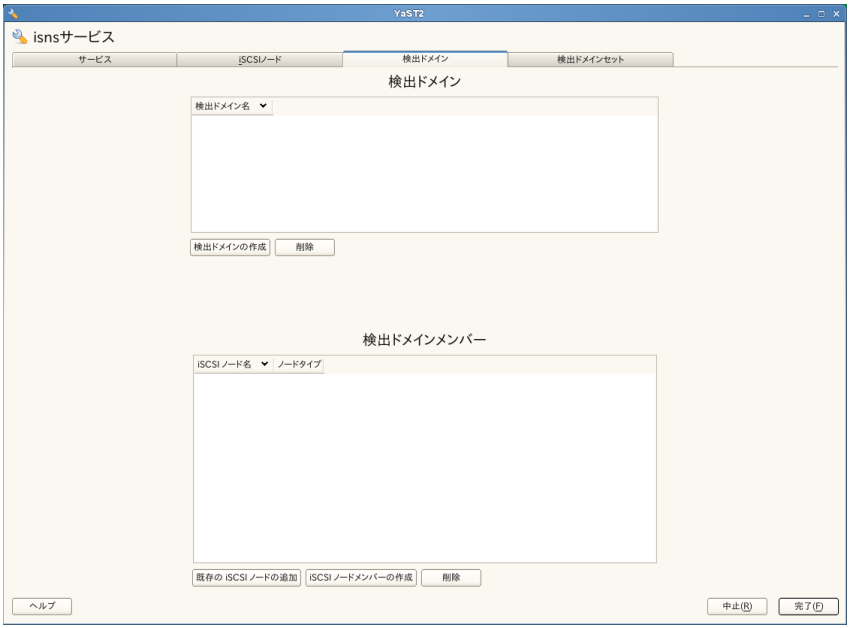
1. YaSTを起動して、［*ネットワークサービス*］の下で［*iSNSサーバ*］を選択します。
2. ［*検出ドメイン*］タブをクリックします。

   ［*検出ドメイン*］の領域にすべての検出ドメインがリストされます。新しい検出ドメインを作成したり、既存のドメインを削除できます。ドメインを削除すると、そのドメインからメンバーが削除されますが、iSCSIノードメンバーは削除されません。

   ［*検出ドメインメンバー*］の領域に、選択した検出ドメインに割り当てられているすべてのiSCSIノードがリストされます。別の検出ドメインを選択すると、その検出ドメインからのメンバーで、リストが更新されます。選択した検出ドメインからiSCSIノード を追加したり、削除できます。iSCSIノード を削除すると、そのノードは、ドメインから削除されますが、iSCSIノード自体は削除されません。

   iSCSIノード を作成すると、未登録のノード を検出ドメインのメンバーとして追加できます。iSCSIイニシエータまたはiSCSIターゲットがこのノードを登録すると、このノードは、このドメインの一部となります。

iSCSIイニシエータが検出要求を発行すると、iSNSサービスは同じ検出ドメイン内のメンバーであるすべてのiSCSIノードターゲットを返します。



3 ［*検出ドメインの作成*］ボタンをクリックします。

既存の検出ドメインを選択して［*削除*］ボタンをクリックして、その検出ドメインを削除できます。

4 作成している検出ドメインの名前を指定して、［*OK*］をクリックします。

5 12.3.2項 「iSNS検出ドメインセットの作成」 (183 ページ)に進みます。

12.3.2 iSNS検出ドメインセットの作成

検出ドメインは検出ドメインセットに属している必要があります。検出ドメインを作成してこの検出ドメインにノードを追加できますが、これはアクティブではないため、iSNSサービスは検出ドメインを検出ドメインセットに追加

するまで機能しません。iSNSをインストールすると、*デフォルトDDS*というデフォルトの検出ドメインセットが自動的に作成され、デフォルトの検出ドメインは自動的にこのドメインセットに追加されます。

検出ドメインセットを作成するには、次の手順に従います。

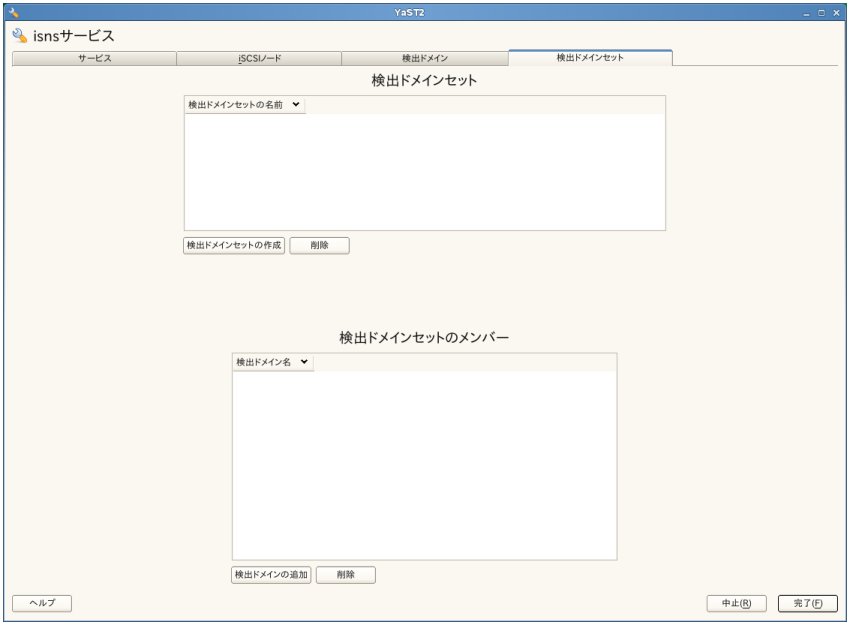
1. YaSTを起動して、［*ネットワークサービス*］の下で［*iSNSサーバ*］を選択します。
2. ［*検出ドメインセット*］タブをクリックします。

   ［*検出ドメインセット*］領域に、すべての検出ドメインセットがリストされます。検出ドメインをアクティブでするには、その検出ドメインを検出ドメインセットのメンバーでする必要があります。

   iSNSデータベースでは、検出ドメインセットは検出ドメインを包含し、検出ドメインはiSCSIノードメンバーを包含します。

   ［*検出ドメインセットのメンバー*］領域に、選択した検出ドメインセットに割り当てられているすべての検出ドメインがリストされます。別の検出ドメインセットを選択すると、その検出ドメインセットからのメンバーで、リストが更新されます。選択した検出ドメインセットから検出ドメインを追加したり、削除できます。検出ドメインを削除すると、その検出ドメインは、検出ドメインセットから削除されますが、検出ドメイン自体は削除されません。

   検出ドメインをセットに追加すると、まだ登録されていないiSNS検出ドメインを、検出ドメインセットのメンバーとして追加できます。



3 ［*検出ドメインセットの作成*］ボタンをクリックします。

既存の検出ドメインセットを選択して［*削除*］ボタンをクリックして、その検出ドメインセットを削除できます。

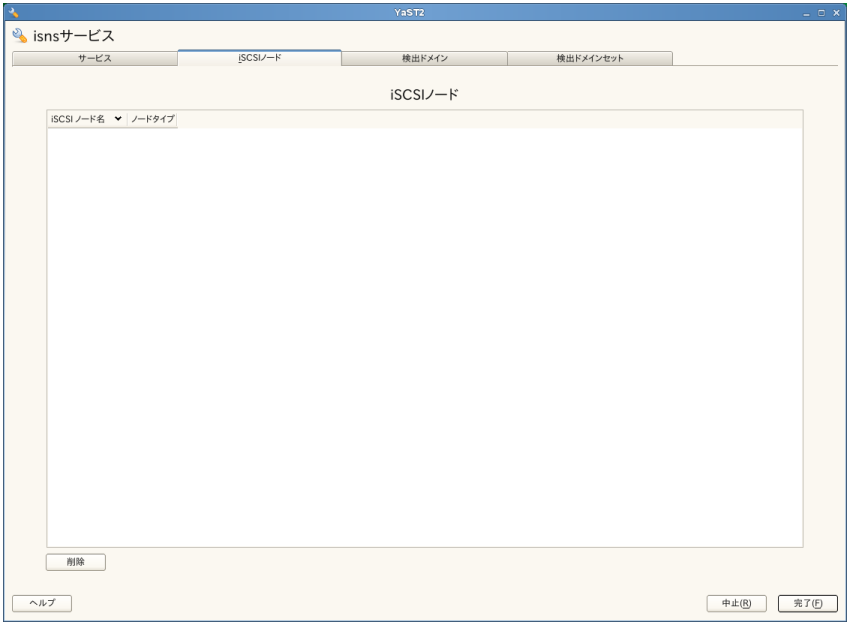
4 作成している検出ドメインセットの名前を指定して、［*OK*］をクリックします。

5 12.3.3項 「iSCSIノードの検出ドメインへの追加」 (185 ページ)に進みます。

12.3.3 iSCSIノードの検出ドメインへの追加

1 YaSTを起動して、［ネットワークサービス］の下で［*iSNS*サーバ］を選択します。

2 ［*iSCSI*ノード］タブをクリックします。



3 ノードのリストをレビューして、iSNSサービスを使用させたいiSCSIターゲットおよびイニシエータがリストされていることを確認します。

iSCSIターゲットまたはイニシエータが一覧にない場合、ノード上のiSCSIサービスを再起動する必要があります。これは、`rcopen-iscsi restart`コマンドを実行してイニシエータを再起動するか、または`rciscsitarget restart`コマンドでターゲットを再起動して実行します。

iSCSIノードを選択して［*削除*］ボタンをクリックして、そのノードをiSNSデータベースから削除できます。iSCSIノードをもう使用しない場合や名前を変更した場合に有効です。

iSCSI環境設定ファイルのiSNSの部分を削除したりコメント化していない限り、iSCSIノードは、iSCSIサービスの再開始時またはサーバの再起動時に、リスト(iSNSデータベース)に自動的に追加されます。

4 ［*検出ドメイン*］タブをクリックして該当する検出ドメインを選択し、［*メンバーの表示*］ボタンをクリックします。

5 ［*Add existing iSCSI Node*］をクリックしてドメインに追加するノードを選択し、［*ノードの追加*］をクリックします。

6 検出ドメインに追加するノードの数だけ「ステップ5(187ページ)」を繰り返し、ノードの追加が終了したら［完了］をクリックします。

iSCSIノードは複数の検出ドメインに属すことができます。

7 12.3.4項「検出ドメインの検出ドメインセットへの追加」(187ページ)に進みます。

12.3.4 検出ドメインの検出ドメインセットへの追加

1 YaSTを起動して、［ネットワークサービス］の下で［*iSNS*サーバ］を選択します。

2 ［*検出ドメインセット*］タブをクリックします。

3 ［*Create Discovery Domain Set*］を選択して、新しいセットを検出ドメインセットのリストに追加します。

4 変更する検出ドメインを選択します。

5 ［*検出ドメインの追加*］をクリックして検出ドメインセットに追加する検出ドメインを選択し、［*検出ドメインの追加*］をクリックします。

6 検出ドメインセットに追加する検出ドメインについて最後のステップを繰り返して、［完了］をクリックします。

検出ドメインは複数の検出ドメインセットに属すことができます。

12.4 iSNSの起動

iSNSは、インストール先のサーバで起動する必要があります。端末コンソールで、`root`ユーザとして、次のコマンドの1つを入力します。

```
rcisns start
```

```
/etc/init.d/isns start
```

iSNSでは、stop、status、restartの各オプションも使用できます。

iSNSはサーバの再起動時に自動的に起動するように設定することもできます。

1. YaSTを起動して、［ネットワークサービス］の下で［*iSNS*サーバ］を選択します。
2. ［サービス］タブを選択して、iSNSサーバのIPアドレスを指定して、［*SaveAddress*］を選択します。
3. 画面の［サービスの*開始*］セクションで、［ブート*時*］を選択します。

   iSNSサーバを手動で起動することもできます。この場合、rcisns startコマンドを使用して、サーバを再起動するときにサービスを毎回起動する必要があります。

12.5 iSNSの停止

iSNSは、それが実行中のサーバで停止させる必要があります。端末コンソールで、rootユーザとして、次のコマンドの1つを入力します。

```
rcisns stop
```

```
/etc/init.d/isns stop
```

12.6 詳細情報

詳細については、「*Linux iSNS for iSCSI project* [http://sourceforge.net/projects/linuxisns/]」を参照してください。プロジェクトの電子メールリストは、「*Linux iSNS - Discussion* [http://sourceforge.net/mailarchive/forum.php?forum_name=linuxisns-discussion]」に掲載されています。

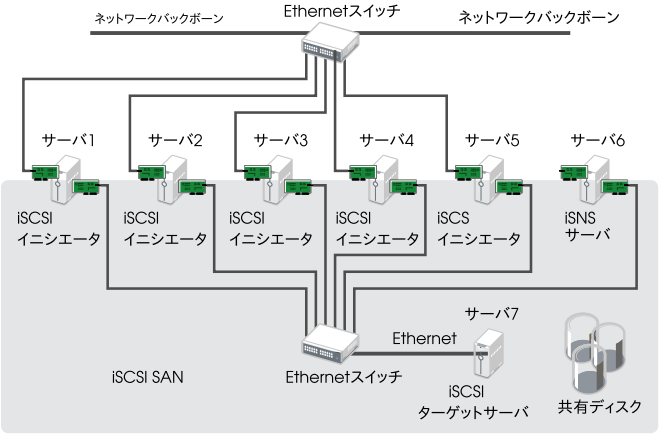
iSNSの一般情報は、「*RFC 4171: Internet Storage Name Service* [http://www.ietf.org/rfc/rfc4171]」に記載されています。

IPネットワークの大容量記憶域 - iSCSI

13

サーバの運用、またはコンピュータセンターの重要なタスクの1つには、サーバシステムに対してハードディスクスペースを提供することがあります。この用途には、多くの場合、ファイバチャネルが使用されます。iSCSI(Internet SCSI)ソリューションは、ファイバチャネルに対する低コストの代替であり、コモディティサーバおよびEthernetネットワーキング装置を活用することができます。Linux iSCSIは、iSCSIイニシエータおよびiSCSIターゲットのソフトウェアの提供により、Linuxサーバを中央ストレージシステムに接続します。

図 13.1　iSNSサーバによるiSCSI SAN



iSCSIは、ストレージネットワーキングプロトコルであり、ブロックストレージデバイスとサーバ間のTCP/IPネットワーク上でSCSIパケットのデータ転送を容易にします。iSCSIターゲットソフトウェアは、ターゲットサーバ上で実

行され、論理ユニットをiSCSIターゲットデバイスとして定義します。iSCSIイニシエータソフトウェアは異なるサーバ上で実行され、ターゲットデバイスに接続して、そのサーバ上でストレージデバイスを使用できるようにします。

重要項目

iSCSIターゲットソフトウェアとiSCSIイニシエータソフトウェアを、運用環境内の同じサーバ上で実行することはできません。

iSCSIターゲットサーバおよびイニシエータサーバは、LAN内のIPレベルでSCSIパケットを送信して通信します。イニシエータサーバ上のアプリケーションがiSCSIターゲットデバイスに対する照会を開始すると、オペレーティングシステムが必要なSCSIコマンドを発行します。するとSCSIコマンドが、*iSCSIイニシエータ*と呼ばれるソフトウェアによってIPパケットに組み込まれ、必要に応じて暗号化されます。パケットは社内IPネットワークで、*iSCSIターゲット*と呼ばれるiSCSIリモートステーションに転送されます。

多くのストレージソリューションが、iSCSIによるアクセス手段を提供しています。また、LinuxサーバにiSCSIターゲットの役割をさせることもできます。この場合、Linuxサーバをファイルシステムサービス用に最適化しておくことが重要です。iSCSIターゲットはLinux内のブロックデバイスにアクセスします。したがってRAIDソリューションを使用することでディスク容量を増やし、メモリも増量してデータキャッシングを向上させることができます。RAIDの詳細については、第8章 *ソフトウェアRAIDの設定*(127ページ)も参照してください。

- 13.1項 「iSCSIのインストール」 (192 ページ)
- 13.2項 「iSCSIターゲットのセットアップ」 (194 ページ)
- 13.3項 「iSCSIイニシエータの設定」 (204 ページ)

13.1 iSCSIのインストール

YaSTにはiSCSIターゲットとiSCSIイニシエータソフトウェアのエントリが含まれていますが、パッケージはデフォルトではインストールされません。

重要項目

iSCSIターゲットソフトウェアとiSCSIイニシエータソフトウェアを、運用環境内の同じサーバ上で実行することはできません。

- 13.1.1項 「iSCSIターゲットソフトウェアのインストール」 (193 ページ)
- 13.1.2項 「iSCSIイニシエータソフトウェアのインストール」 (193 ページ)

13.1.1 iSCSIターゲットソフトウェアのインストール

iSCSIターゲットデバイスを作成したいサーバに、iSCSIターゲットソフトウェアをインストールします。

1. YaSTを開き、`root`ユーザとしてログインします。
2. ［ネットワークサービス*iSCSI* ターゲットを選択します。
3. `iscsitarget`パッケージのインストールを促されたら、［インストール］をクリックします。
4. 画面上のインストール指示に従って、必要に応じてインストールメディアを提供します。

   インストールが完了したら、［iSCSIターゲットの概要］ページで［サービス］タブを選択した状態で、YaSTが開きます。
5. 13.2項 「iSCSIターゲットのセットアップ」 (194 ページ)に進みます。

13.1.2 iSCSIイニシエータソフトウェアのインストール

iSCSIサーバ上に設定したターゲットデバイスにアクセスしたい各サーバに、iSCSIイニシエータソフトウェアをインストールします。

1. YaSTを開き、`root`ユーザとしてログインします。

2 ［ネットワークサービス*iSCSI*イニシエータ］を選択します。

3 `open-iscsi`パッケージのインストールを促されたら、［*インストール*］をクリックします。

4 画面上のインストール指示に従って、必要に応じてインストールメディアを提供します。

インストールが完了したら、［iSCSIイニシエータの概要］ページで［サービス］タブを選択した状態で、YaSTが開きます。

5 13.3項 「iSCSIイニシエータの設定」 (204 ページ)に進みます。

13.2 iSCSIターゲットのセットアップ

SUSE® Linux Enterprise Serverには、Ardis iSCSIターゲットから進化したオープンソースのiSCSIターゲットソリューションが付属しています。基本的な設定はYaSTを使って行えますが、iSCSIの機能をフル活用するには、手動で設定を行う必要があります。

- 13.2.1項 「ストレージスペースの準備」 (194 ページ)
- 13.2.2項 「YaSTを使ったiSCSIターゲットの作成」 (197 ページ)
- 13.2.3項 「iSCSIターゲットの手動設定」 (200 ページ)
- 13.2.4項 「ietadmを使ったオンラインターゲットの設定」 (202 ページ)

13.2.1 ストレージスペースの準備

iSCSIターゲットの設定により、既存のブロックデバイスをiSCSIイニシエータにエクスポートします。YaSTのパーティショナを使用して、またはコマンドラインからパーティション分割を行うことで、未フォーマットのパーティションやデバイスを設定し、ターゲットデバイスで使用するストレージスペースを準備する必要があります。

重要項目

iSCSIターゲットとして使用するデバイスやパーティションを設定したら、ローカルパス経由で直接アクセスしないでください。作成時にはマウントポイントは指定しないでください。

- 「デバイスのパーティショニング」 (195 ページ)
- 「仮想環境でのデバイスのパーティション分割」 (196 ページ)

デバイスのパーティショニング

1 rootユーザとしてログインし、YaSTを開きます。

2 ［システムパーティショナ］を選択します。

3 パーティショナの使用に関する警告に対して［はい］を選択して続行します。

4 ［追加］をクリックしてパーティションを作成しますが、フォーマットやマウントは行いません。

iSCSIターゲットは、Linux、Linux LVM、またはLinux RAIDファイルシステムIDで未フォーマットのパーティションを使用できます。

4a ［プライマリパーティション］を選択し、［次へ］をクリックします。

4b 使用する容量を指定して、［次へ］をクリックします。

4c ［フォーマットしない］を選択し、ファイルシステムIDの種類を指定します。

4d ［マウントしない］を選択します。

4e ［完了］をクリックします。

5 後でiSCSI LUNとして使用したい各エリアに対して、ステップ4 (195ページ)を繰り返し実行します。

6 ［*受諾*］をクリックして変更を保存し、YaSTを閉じます。

仮想環境でのデバイスのパーティション分割

XenゲストサーバをiSCSIターゲットサーバとして使用することができます。iSCSIストレージデバイスに使用するストレージスペースをゲストの仮想マシンに割り当て、ゲスト環境内の仮想ディスクとしてそのスペースにアクセスします。各仮想ディスクは、ディスク全体、パーティション、ボリュームなどの物理ブロックデバイスでも、Xenホストサーバ上の大規模な物理ディスク上の単一イメージファイルが仮想ディスクになっている、ファイルバックディスクイメージのいずれでも可能です。最適なパフォーマンスを得るためには、物理ディスクまたはパーティションから各仮想ディスクを作成してください。ゲストの仮想マシンに仮想ディスクを設定したら、ゲストサーバを起動し、物理サーバの場合と同じ方法で、新しいブランクの仮想ディスクをiSCSIターゲットデバイスとして設定します。

ファイルバックディスクイメージがXenホストサーバ上に作成され、Xenゲストサーバに割り当てられます。デフォルトでは、Xenはファイルバックディスクイメージを`/var/lib/xen/images/`*`vm_name`*ディレクトリに保存します。ここで*`vm_name`*は仮想マシンの名前です。

たとえば、サイズが4GBの`/var/lib/xen/images/vm_one/xen-0`ディスクイメージを作成する場合は、まず、ディレクトリが存在することを確認し、次に、イメージ自体を作成します。

1 ホストサーバに`root`ユーザとしてログインします。

2 端末コンソールプロンプトで次のコマンドを入力します。

```
mkdir -p /var/lib/xen/images/vm_one
dd if=/dev/zero of=/var/lib/xen/images/vm_one/xen-0 seek=1M bs=4096
count=1
```

3 Xen設定ファイルで、ファイルシステムイメージをゲスト仮想マシンに割り当てます。

4 ゲストサーバに`root`ユーザとしてログインし、YaSTを使用して「デバイスのパーティショニング」 (195 ページ)の手順で仮想ブロックデバイスを設定します。

13.2.2 YaSTを使ったiSCSIターゲットの作成

1 YaSTを開き、rootユーザとしてログインします。

2 ［ネットワークサービス*iSCSI* ターゲットを選択します。

［iSCSIターゲットの概要］ページで［サービス］タブを選択した状態で、YaSTが開きます。

3 ［*サービスの開始*］領域で、次のいずれかを選択します。

- **起動時:** その後の再起動時には、イニシエータサービスが自動的に開始します。
- **手動(デフォルト):** サービスを手動で開始します。

4 ターゲットアドバタイジングにiSNSを使用する場合は、［*iSNSアクセス制御*］チェックボックスを選択し、IPアドレスを入力します。

5 必要に応じて、ファイアウォールポートを開き、リモートコンピュータからサーバへのアクセスを許可します。

5a ［*ファイアウォールでポートを開く*］チェックボックスを選択します。

5b ［*ファイアウォールの詳細*］をクリックして、ポートを開くネットワークインタフェースを指定し、ネットワークインタフェースの横のチェックボックスをオンにして有効にし、［*OK*］をクリックして設定を受け入れます。

6 このサーバで設定したターゲットデバイスへの接続に認証が必要な場合は、［*グローバル*］タブを選択し、［*認証なし*］の選択を解除して、送受信の認証に必要な資格情報を指定します。

デフォルトでは［*認証なし*］のオプションが有効になっています。認証は受信、送信、または送受信の両方に対して指定できます。ユーザ名とパスワードのペアを［*受信認証*］の一覧に追加することで、受信認証の資格情報を複数セット指定することもできます。

7 iSCSIターゲットデバイスを指定します。

7a ［ターゲット］タブを選択します。

7b まだそうしていない場合は、一覧からサンプルのiSCSIターゲットを選択して削除し、［*続ける*］をクリックして削除を確認します。

7c ［*追加*］をクリックして、新しいiSCSIターゲットを追加します。

iSCSIターゲットによって自動的に未フォーマットのパーティションやブロックデバイスが提示され、［ターゲット］および［イニシエータ］フィールドに情報が入力されます。

7d これを受諾することも、別なスペースを選択することもできます。

［*追加*］をクリックして、そのLUNに割り当てるセクタを指定することで、スペースをさらに分割して、デバイス上にLUNを作成することもできます。これらのLUNに追加のオプションが必要な場合は、［エキスパート設定］を選択します。

7e Click *Next*

7f 作成する各iSCSIターゲットデバイスに対して、ステップ7c (198ページ)からステップ 7e (198 ページ)を繰り返し実行します。

7g (オプション)［サービス］タブで［*保存*］をクリックして、設定したiSCSIターゲットの情報をファイルにエクスポートします。

こうしておくと、後でこの情報をリソースの利用者に簡単に提供することができます。

7h ［*完了*］をクリックしてデバイスを作成し、［*はい*］をクリックしてiSCSIソフトウェアスタックをリスタートします。

iSCSIターゲットを設定するには、YaSTの［*iSCSIターゲット*］モジュールを起動します。設定項目は、3つのタブに分かれています。［*Service*］タブでは、実行モードとファイアウォールの設定を行います。リモートコンピュータからiSCSIターゲットにアクセスする場合は、［*ファイアウォールでポートを開く*］を選択します。iSNSサーバが検出およびアクセス制御を管理する場合は、［*iSNSアクセス管理*］を有効にして、iSNSサーバのIPアドレスを入力

します。ホスト名は使用できません。IPアドレスを使用してください。iSNSの詳細は、第12章 *Linux用iSNS* (177 ページ)を参照してください。

［*Global*］タブでは、iSCSIサーバの設定を行います。ここで設定する認証方法は、サービスの検出に使用します。ターゲットにアクセスする場合のものではありません。ディスカバリへのアクセスを制限しない場合は、［*No Authentication*］を選択します。

認証が必要な場合、2つの検討事項があります。まず、イニシエータは、iSCSIターゲットでディスカバリを実行するためのパーミッションがあることを証明できなければなりません。この設定は、［*Incoming Authentication*］で行います。もう1つは、iSCSIターゲットはイニシエータに、自分が正しいターゲットであることを証明しなければなりません。そのため、iSCSIターゲットもユーザ名とパスワードを使用できます。この設定は、［*Outgoing Authentication*］で行います。認証の詳細は、*RFC 3720* [http://www.ietf.org/rfc/rfc3720.txt]を参照してください。

ターゲットは、［*Targets*］タブで定義します。新しいiSCSIターゲットを作成するには、［*追加*］をクリックします。最初のダイアログでは、エクスポートするデバイスに関する情報を指定します。

ターゲット
: ［Target］*行には、以下のような固定形式の構文を指定します。*

```
iqn.yyyy-mm.<reversed domain name>
```

この行は常に「iqn」から始まります。「yyyy-mm」の部分には、このターゲットをアクティブにする日付を指定します。命名規則の詳細については、「*RFC 3722* [http://www.ietf.org/rfc/rfc3722.txt]」を参照してください。

Identifier
: ［Identifier］*は、自由に指定することができます。*ただし、システムを体系的に管理するためにも、一定のスキーマを使用するようにしてください。

LUN
: ターゲットに複数のLUNを割り当てることができます。そのためには、［*ターゲット*］タブでターゲットを選択し、［*編集*］をクリックします。次に、既存ターゲットに新しいLUNを追加します。

パス
エクスポートするブロックデバイス、またはファイルシステムイメージのパスを追加します。

次のメニューでは、ターゲットへのアクセス制限を設定します。ここの設定は、ディスカバリの認証設定とほとんど変わりありません。この場合、少なくとも着信認証を設定する必要があります。

新しいターゲットの設定を完了するには、［次へ］をクリックします。［*Target*］タブの概要ページが表示されます。変更内容を有効にするには、［完了］をクリックします。

13.2.3 iSCSIターゲットの手動設定

iSCSIターゲットを設定するには、`/etc/ietd.conf`を編集します。このファイル中の、最初の*Target*宣言より前にあるすべてのパラメータは、ファイルのグローバルパラメータになります。この部分にある認証情報は、グローバルパラメータではありません。iSCSIターゲットの検出に用いられます。

iSNSサーバにアクセスできる場合は、まず、ファイルを、このサーバについてターゲットに通知するように設定する必要があります。iSNSサーバのアドレスは、必ず、IPアドレスで指定する必要があります。iSNSサーバのDNS名を指定することはできません。この機能の設定は、次のようになります。

```
iSNSServer 192.168.1.111
iSNSAccessControl no
```

この設定では、iSCSIターゲットが`iSNS`サーバでそれ自体を登録し、ディスカバリのイニシエータとなります。iSNSの詳細は、第12章 *Linux用iSNS* (177 ページ)を参照してください。iSNSディスカバリのためのアクセス制御はサポートされていません。［`iSNS`アクセス管理］を［いいえ］のままにします。

すべての直接iSCSI認証は、双方向で行うことができます。iSCSIターゲットがiSCSIイニシエータに認証を要求するには、`IncomingUser`を使用します。このオプションは、複数回追加できます。iSCSIイニシエータも、iSCSIターゲットに認証を要求することができます。この場合は、`OutgoingUser`を使用します。どちらの場合も、構文は同じです。

```
IncomingUser <username> <password>
OutgoingUser <username> <password>
```

認証後は、1つまたは複数のターゲット定義を指定します。定義する各ターゲットについて、Targetセクションを追加します。このセクションは、常にTarget識別子で始まり、その後に論理ユニット番号の定義が続きます。

```
Target iqn.yyyy-mm.<reversed domain name>[:identifier]
          Lun 0 Path=/dev/mapper/system-v3
          Lun 1 Path=/dev/hda4
          Lun 2 Path=/var/lib/xen/images/xen-1,Type=fileio
```

Target行では、yyyy-mmにターゲットを有効にする日付を指定し、identifierに任意の識別子を指定できます。命名規則の詳細については、「*RFC 3722* [http://www.ietf.org/rfc/rfc3722.txt]」を参照してください。この例では、3つの異なるブロックデバイスをエクスポートしています。最初のブロックデバイスは論理ボリューム(「第4章 *LVM*の設定 (31 ページ)」も参照)、2番目はIDEパーティション、3番目はローカルファイルシステムで使用可能なイメージです。これらはすべてiSCSIイニシエータへのブロックデバイスのようになります。

iSCSIターゲットを有効にする前に、Lun定義の後に、最低1つのIncomingUserを追加してください。このパラメータは、このターゲットの使用に対する認証を指定します。

変更内容を有効にするには、rcopen-iscsi restartコマンドを実行して、iscsitargetデーモンを再起動します。/procファイルシステムで、設定内容を確認してください。

```
cat /proc/net/iet/volume
tid:1 name:iqn.2006-02.com.example.iserv:systems
        lun:0 state:0 iotype:fileio path:/dev/mapper/system-v3
        lun:1 state:0 iotype:fileio path:/dev/hda4
        lun:2 state:0 iotype:fileio path:/var/lib/xen/images/xen-1
```

ここで説明しているほかにも、iSCSIターゲットの動作を制御するさまざまなオプションがあります。詳細については、ietd.confのマニュアルページを参照してください。

/procファイルシステムには、アクティブなセッションも表示されます。接続されている各イニシエータに対応するエントリが、/proc/net/iet/sessionに追加されます。

```
cat /proc/net/iet/session
tid:1 name:iqn.2006-02.com.example.iserv:system-v3
   sid:562949957419520
initiator:iqn.2005-11.de.suse:cn=rome.example.com,01.9ff842f5645
      cid:0 ip:192.168.178.42 state:active hd:none dd:none
   sid:281474980708864 initiator:iqn.2006-02.de.suse:01.6f7259c88b70
      cid:0 ip:192.168.178.72 state:active hd:none dd:none
```

13.2.4 ietadmを使ったオンラインターゲットの設定

iSCSIターゲットの設定の変更が必要な場合は、必ず、ターゲットを再起動して、設定ファイルで行った変更を有効にする必要があります。ただし、この作業を行うと、アクティブなセッションがすべて中断されます。この問題を回避するには、環境設定ファイルの/etc/ietd.confを変更すると同時に、ietadm管理ユーティリティを使って現在の設定も変更してください。

LUNを指定した新しいiSCSIターゲットを作成するには、まず設定ファイルを更新します。追加するエントリの例を以下に示します。

```
Target iqn.2006-02.com.example.iserv:system2
         Lun 0 Path=/dev/mapper/system-swap2
         IncomingUser joe secret
```

この設定を手動で行うには、次の手順に従ってください。

1. ietadm --op new --tid=2 --params Name=iqn.2006-02.com.example.iserv:system2コマンドを実行して、新しいターゲットを作成します。
2. ietadm --op new --tid=2 --lun=0 --params Path=/dev/mapper/system-swap2コマンドを実行して、LUNを追加します。

3 ietadm --op new --tid=2 --user --params=IncomingUser=joe,Password=secretコマンドを実行して、このターゲットにユーザ名とパスワードを設定します。

4 cat /proc/net/iet/volumeコマンドを実行して、設定内容を確認します。

アクティブな接続を削除することもできます。まず、cat /proc/net/iet/sessionコマンドを実行して、アクティブな接続を表示します。たとえば、次のように表示されます。

```
cat /proc/net/iet/session
tid:1 name:iqn.2006-03.com.example.iserv:system
        sid:281474980708864 initiator:iqn.1996-04.com.example:01.82725735af5
                cid:0 ip:192.168.178.72 state:active hd:none dd:none
```

セッションIDが281474980708864のセッションを削除する場合は、ietadm --op delete --tid=1 --sid=281474980708864 --cid=0コマンドを実行します。このコマンドを実行すると、クライアントシステムからデバイスにアクセスできなくなるため、このデバイスにアクセスしているデバイスがハングアップする可能性があることに注意してください。

ietadmを使って、さまざまな環境設定パラメータを変更することもできます。グローバル変数を一覧表示する場合は、ietadm --op show --tid=1 --sid=0コマンドを実行します。次のような実行結果が表示されます。

```
InitialR2T=Yes
ImmediateData=Yes
MaxConnections=1
MaxRecvDataSegmentLength=8192
MaxXmitDataSegmentLength=8192
MaxBurstLength=262144
FirstBurstLength=65536
DefaultTime2Wait=2
DefaultTime2Retain=20
MaxOutstandingR2T=1
DataPDUInOrder=Yes
DataSequenceInOrder=Yes
ErrorRecoveryLevel=0
HeaderDigest=None
DataDigest=None
OFMarker=No
IFMarker=No
OFMarkInt=Reject
IFMarkInt=Reject
```

これらのパラメータは、すべて容易に変更できます。たとえば、最大接続数を2に変更する場合は、=2を実行します。

```
ietadm --op update --tid=1 --params=MaxConnections=2.
```

/etc/ietd.confファイルでは、このパラメータに対応する行がMaxConnections 2のように指定されています。

警告

ietadmユーティリティによる変更は、システムに対して永久的なものではありません。これらの変更は、/etc/ietd.conf環境設定ファイルに追加しない限り、次の再起動で失われます。ネットワークでのiSCSIの利用方法によっては、これによって重大な問題が発生する可能性があります。

---

ietadmユーティリティには、他にも使用可能なオプションがあります。ietadm –hを使用して、概要をつかんでください。また、省略形も利用できます。省略形には、ターゲットID (tid)、セッションID (sid)、および接続ID (cid)などがあります。これらの情報は、/proc/net/iet/sessionにもあります。

13.3 iSCSIイニシエータの設定

任意のiSCSIターゲットに接続するには、iSCSIイニシエータ(iSCSIクライアントとも呼ぶ)を使用できます。これは、「13.2項 「iSCSIターゲットのセットアップ」 (194ページ)」で説明されているターゲットソリューションだけに限りません。iSCSIイニシエータの設定には、利用可能なiSCSIターゲットの検出と、iSCSIセッションの設定という2つの主要ステップがあります。どちらの設定も、YaSTを使って行うことができます。

- 13.3.1項 「YaSTを使ったiSCSIイニシエータの設定」 (205 ページ)
- 13.3.2項 「手動によるiSCSIイニシエータの設定」 (209 ページ)
- 13.3.3項 「iSCSIクライアントデータベース」 (210 ページ)
- 13.3.4項 「詳細情報」 (211 ページ)

13.3.1 YaSTを使ったiSCSIイニシエータの設定

YaSTの［iSCSIイニシエータの概要］が3つのタブに分割されます。

- **サービス:** ［*サービス*］タブでは、ブート時にiSCSIイニシエータを有効にできます。固有の*イニシエータ名*とディスカバリに使用するiSNSサーバも設定できます。iSNSのデフォルトポートは3205です。
- **接続したターゲット:** ［Connected Targets］*タブには、現在接続しているiSCSIターゲットの概要が表示されます。*このタブにも、［*検出されたターゲット*］タブのように、システムに新しいターゲットを追加するオプションが用意されています。

  このページでは、ターゲットデバイスを選択して、各iSCSIターゲットデバイスの起動時の設定を切り替えることができます。

  - **自動:** このオプションは、iSCSIサービス自体の起動時に接続するiSCSIターゲットに使用されます。これが通常の設定です。
  - **Onboot(起動時):** このオプションは、起動時、つまりルート(/)がiSCSI上にある場合に接続するiSCSIターゲットに使用します。したがって、iSCSIターゲットデバイスはサーバの起動時にinitrdによって評価されます。
- **検出されたターゲット:** ［*検出されたターゲット*］では、ネットワーク内のiSCSIターゲットを手動で検出することができます。
- 「iSCSIイニシエータの設定」 (205 ページ)
- 「iSNSによるiSCSIターゲットの検出」 (207 ページ)
- 「iSCSIターゲットの手動検出」 (207 ページ)
- 「iSCSIターゲットデバイスの起動設定」 (208 ページ)

iSCSIイニシエータの設定

1 YaSTを開き、`root`ユーザとしてログインします。

2 ［*ネットワークサービスiSCSIイニシエータ*］を選択します。

［iSCSIイニシエータの概要］ページで［サービス］タブが選択された状態で、YaSTが開きます。

3 ［*サービスの開始*］領域で、次のいずれかを選択します。

- **起動時:** その後の再起動時には、イニシエータサービスが自動的に開始します。
- **手動(デフォルト):** サービスを手動で開始します。

4 ［*イニシエータ名*］を指定、または確認します。

このサーバ上のiSCSIイニシエータに、正しい形式のイニシエータ修飾名(IQN)を指定します。イニシエータ名はネットワーク全体で固有のものでなければなりません。IQNは次の一般的なフォーマットを使用します。

```
iqn.yyyy-mm.com.mycompany:n1:n2
```

ここでn1とn2はアルファベットか数字です。次に例を示します。

```
iqn.1996-04.de.suse:01:9c83a3e15f64
```

［*イニシエータ名*］には、サーバ上の`/etc/iscsi/initiatorname.iscsi`ファイルから対応する値が自動的に入力されます。

サーバがiBFT(iSCSI Boot Firmware Table)をサポートしている場合は、［*イニシエータ名*］にはIBFT内の対応する値が入力され、このインタフェースではイニシエータ名を変更できません。代わりにBIOSセットアップを使用して変更してください。iBFTは、サーバのiSCSIターゲットとイニシエータの説明を含む、iSCSIの起動プロセスに便利な各種パラメータを含んだ情報ブロックです。

5 次のいずれかの方法を使用して、ネットワーク上のiSCSIターゲットを検出します。

- **iSNS:** iSNS (Internet Storage Name Service)を使用してiSCSIターゲットを検出するには、続いて「iSNSによるiSCSIターゲットの検出」(207 ページ)を実行します。
- **検出されたターゲット:** iSCSIターゲットデバイスを手動で検出するには、続いて「iSCSIターゲットの手動検出」 (207 ページ)を実行します。

iSNSによるiSCSIターゲットの検出

このオプションを使用する前に、ご使用の環境内でiSNSサーバをインストールし、設定しておく必要があります。詳細については、第12章 *Linux用iSNS* (177 ページ)を参照してください。

1. YaSTで［*iSCSIイニシエータ*］を選択し、次に［*サービス*］タブを選択します。
2. iSNSサーバのIPアドレスとポートを指定します。

   デフォルトのポートは3205です。
3. ［iSCSIイニシエータの概要］ページで、［*完了*］をクリックして変更内容を保存、および適用します。

iSCSIターゲットの手動検出

iSCSIイニシエータを設定しているサーバからアクセスする各iSCSIターゲットサーバについて、次の手順を繰り返し実行します。

1. YaSTで［*iSCSIイニシエータ*］を選択し、次に［*検出されたターゲット*］タブを選択します。
2. ［*検出*］をクリックして［*iSCSIイニシエータの検出*］ダイアログを開きます。
3. IPアドレスを入力し、必要に応じてポートを変更します。

   デフォルトポートは3260です。

4 認証が必要な場合は、［*認証なし*］の選択を解除して、［*受信*］または［*送信*］認証用の資格情報を指定します。

5 ［*次へ*］をクリックして、検出を開始し、iSCSIターゲットサーバに接続します。

6 資格情報が必要な場合は、検出成功後、［*ログイン*］を使用してターゲットを有効化します。

指定したiSCSIターゲットを使用するための、認証資格情報の提供を促されます。

7 ［*次へ*］をクリックして、設定を完了します

作業が正常に完了すると、［*Connected Targets*］にターゲットが表示されます。

これで、仮想iSCSIデバイスを利用できるようになりました。

8 ［iSCSIイニシエータの概要］ページで、［*完了*］をクリックして変更内容を保存、および適用します。

9 `lsscsi`コマンドを使用すると、iSCSIターゲットデバイスのローカルデバイスパスを検出することができます。

```
lsscsi
[1:0:0:0]   disk    IET      VIRTUAL-DISK     0     /dev/sda
```

iSCSIターゲットデバイスの起動設定

1 YaSTで、［*iSCSIイニシエータ*］を選択し、次に［*接続したターゲット*］タブを選択して、現在サーバに接続されているiSCSIターゲットデバイスの一覧を表示することができます。

2 管理するiSCSIターゲットデバイスを選択します。

3 ［*起動の切り替え*］をクリックして設定を変更します。

- **自動:** このオプションは、iSCSIサービス自体の起動時に接続するiSCSIターゲットに使用されます。これが通常の設定です。

- **Onboot(起動時):** このオプションは、起動時、つまりルート(/)がiSCSI上にある場合に接続するiSCSIターゲットに使用します。したがって、iSCSIターゲットデバイスはサーバの起動時にinitrdによって評価されます。

4 ［完了］をクリックして、変更を保存、および適用します。

13.3.2 手動によるiSCSIイニシエータの設定

iSCSI接続の検出や設定を行うには、iscsidが稼働していなければなりません。初めて検出(ディスカバリ)を実行する場合、iSCSIイニシエータの内部データベースが、/var/lib/open-iscsiディレクトリに作成されます。

ディスカバリがパスワードにより保護されている場合は、iscsidに認証情報を渡します。最初にディスカバリを実行する時には内部データベースが存在していないため、現時点でこれは使用できません。かわりに、/etc/iscsid.conf設定ファイルを編集して、情報を指定する必要があります。パスワード情報をiscsidに渡すには、/etc/iscsid.confファイルの最後に、次の行を追加します。

```
discovery.sendtargets.auth.authmethod = CHAP
discovery.sendtargets.auth.username = <username>
discovery.sendtargets.auth.password = <password>
```

ディスカバリは、受け取ったすべての値を内部データベースに保存します。また、検出したターゲットをすべて表示します。ディスカバリを実行するには、iscsiadm –m discovery --type=st --portal=<targetip>コマンドを使用します。次のような実行結果が表示されます。

```
149.44.171.99:3260,1 iqn.2006-02.com.example.iserv:systems
```

iSNSサーバで使用できるターゲットを検出するには、コマンドiscsiadm --mode discovery --type isns --portal <targetip>を使用します。

iSCSIターゲットに定義されている各ターゲットが、それぞれ1行に表示されます。保存されたデータの詳細については、「13.3.3項 「iSCSIクライアントデータベース」 (210 ページ)」を参照してください。

iscsiadmコマンドの--loginオプションを使用すると、必要なすべてのデバイスが作成されます。

```
iscsiadm -m node -n iqn.2006-02.com.example.iserv:systems --login
```

lsscsiコマンドを実行すると、新しく生成されたデバイスが表示されます。これらのデバイスをマウントして、アクセスできるようになりました。

13.3.3 iSCSIクライアントデータベース

iSCSIイニシエータが検出した情報は、/var/lib/open-iscsiに格納されている2つのデータベースファイルに保管されます。1つは、ディスカバリが検出したターゲット用のデータベースで、もう1つは検出したノード用のデータベースです。データベースにアクセスする場合、まずデータをディスカバリ用データベースから取得するのか、またはノードデータベースから取得するのかを指定する必要があります。指定するには、iscsiadmコマンドの-m discoveryまたは-m nodeパラメータを使用します。iscsiadmコマンドに、どちらかのパラメータを指定して実行すると、そのデータベースに保管されているレコードの概要が表示されます。

```
iscsiadm -m discovery
149.44.171.99:3260,1 iqn.2006-02.com.example.iserv:systems
```

この例のターゲット名はiqn.2006-02.com.example.iserv:systemsです。このデータセットに関連する操作を行う場合に、この名前が必要になります。ID iqn.2006-02.com.example.iserv:systemsのデータレコードのコンテンツを調べるには、次のコマンドを使用します。

```
iscsiadm -m node --targetname iqn.2006-02.com.example.iserv:systems
node.name = iqn.2006-02.com.example.iserv:systems
node.transport_name = tcp
node.tpgt = 1
node.active_conn = 1
node.startup = manual
node.session.initial_cmdsn = 0
node.session.reopen_max = 32
node.session.auth.authmethod = CHAP
node.session.auth.username = joe
node.session.auth.password = ********
node.session.auth.username_in = <empty>
node.session.auth.password_in = <empty>
```

```
node.session.timeo.replacement_timeout = 0
node.session.err_timeo.abort_timeout = 10
node.session.err_timeo.reset_timeout = 30
node.session.iscsi.InitialR2T = No
node.session.iscsi.ImmediateData = Yes
....
```

これらの変数の値を変更する場合は、iscsiadmコマンドでupdateオプションを使用します。たとえば、初期化時にiscidをiSCSIターゲットにログインさせる場合は、値にautomaticとnode.startupを設定します。

```
iscsiadm -m node -n iqn.2006-02.com.example.iserv:systems --op=update
--name=node.startup --value=automatic
```

deleteで、古くなったデータセットを削除します。ターゲットiqn.2006-02.com.example.iserv:systemsが有効なレコードでなくなった場合は、次のコマンドでこのレコードを削除してください。

```
iscsiadm -m node -n iqn.2006-02.com.example.iserv:systems --op=delete
```

重要項目

このオプションでは、確認のメッセージを表示せずかにレコードを削除するため、使用する際には細心の注意を払うようにしてください。

検出したすべてのターゲットのリストを取得するには、iscsiadm -m nodeコマンドを実行します。

13.3.4 詳細情報

iSCSIプロトコルは、数年に渡って利用されています。そのため、iSCSIとSANソリューションの比較、ベンチマークテスト、ハードウェアソリューションの解説など、さまざまなレビューや参考資料が発表または公開されています。open-iscsiの詳細に関する代表的なサイトを以下に示します。

- *Open-iSCSI Project* [http://www.open-iscsi.org/]
- *AppNote: iFolder on Open Enterprise Server Linux Cluster using iSCSI* [http://www.novell.com/coolsolutions/appnote/15394.html]

このほか、オンラインマニュアルも利用できます。iscsiadm、iscsid、ietd.confおよびietdの各マニュアルページのほか、環境設定ファイルのサンプル/etc/iscsid.confも参照してください。

14 ボリュームのスナップショット

ファイルシステムのスナップショットはコピーオンライント技術の1つで、既存のボリュームのデータブロックに対する変更を監視し、いずれかのブロックに書き込みが行われると、スナップショット時のブロックの値がスナップショットボリュームにコピーされます。こうすることで、スナップショットボリュームが削除されるまで、データのその時点のコピーが保存されます。

14.1 ボリュームスナップショットの理解

ファイルシステムのスナップショットには、元のボリュームに関するメタデータと、スナップショットの作成後に変更された元のボリュームからのデータブロックが含まれています。スナップショットを介してデータにアクセスすると、元のボリュームの瞬間的なコピーが表示されます。バックアップ媒体からデータを復元したり、変更されたデータを上書きする必要はありません。

Xenホスト環境では、仮想マシンは、仮想ディスクファイルの使用ではなくに、ストレージバックエンドとして、LVM論理ボリュームを使用する必要があります。

Linuxのスナップショットでは、ファイルシステムのその時点のビューからバックアップを作成できます。スナップショットは瞬時に作成され、削除するまで保存されます。ボリューム自体はユーザが引き続き利用できるようにしながら、スナップショットからファイルシステムのバックアップを作成できます。当初のスナップショットには、スナップショットに関するメタデータが含まれていますが、オリジナルボリュームの実際のデータは含まれていません。スナップショットはコピーオンライト技術を使用して、オリジナルデータブロックのデータ変更を検出します。スナップショットボリューム内のブロックにスナップショットをとった際に保存されている値をコピーし、オリジナルブロックに新しいデータを保存することができます。オリジナルの値からブロックが変化すると、スナップショットのサイズが拡大します。

スナップショットのサイズを決定する際には、オリジナルボリュームに対して予想されるデータ変更量、およびスナップショットの保存期間を考慮する必要があります。スナップショットボリュームに割り当てるスペースの量は、元のボリュームのサイズ、スナップショットの保持予定期間、およびスナップショットのライフタイム中に変更が予期されるデータブロックの数によって異なります。スナップショットボリュームは、作成後のサイズ変更はできません。目安として、元の論理ボリュームの約10%のサイズで、スナップショットボリュームを作成してください。スナップショットの削除前に、元のボリューム内のすべてのブロックが1回以上変更されると予期される場合は、スナップボリュームのサイズを、少なくとも元のボリュームサイズにそのボリュームに関するメタデータ用スペースを加えたサイズにする必要があります。データ変更が頻繁でないか、またはライフタイムが十分短いと予期される場合、必要なスペースは少なくなります。

重要項目

スナップショットのライフタイム中は、スナップショットを先にマウントしないと、元のボリュームをマウントできません。

スナップショットが不要になったら、必ず、システムからスナップショットを削除してください。スナップショットは、元のボリュームでデータブロックが変更されるにつれ、最終的に、満杯になります。スナップショットはいっぱいになると、使用不可になるので、元のボリュームの再マウントができなくなります。

スナップショットは、最後に作成されたものから順に削除してください。

14.2 LVMによるLinuxスナップショットの作成

LVM (Logical Volume Manager)は、ファイルシステムのスナップショットの作成に使用できます。

- 端末コンソールを開いて、rootユーザとしてログインし、次のように入力します。

  ```
  lvcreate -s -L 1G -n snap_volume source_volume_path
  ```

  次に例を示します。

  ```
  lvcreate -s -L 1G -n linux01-snap /dev/lvm/linux01
  ```

  スナップショットが/dev/lvm/linux01-snapボリュームとして作成されます。

14.3 スナップショットの監視

- 端末コンソールを開いて、rootユーザとしてログインし、次のように入力します。

  ```
  lvdisplay snap_volume
  ```

  次に例を示します。

  ```
  lvdisplay /dev/vg01/linux01-snap

  --- Logical volume ---
    LV Name                /dev/lvm/linux01
    VG Name                vg01
    LV UUID                QHVJYh-PR3s-A4SG-s4Aa-MyWN-Ra7a-HL47KL
    LV Write Access        read/write
    LV snapshot status     active destination for /dev/lvm/linux01
    LV Status              available
  ```

```
# open                 0
LV Size                80.00 GB
Current LE             1024
COW-table size         8.00 GB
COW-table LE           512
Allocated to snapshot  30%
Snapshot chunk size    8.00 KB
Segments               1
Allocation             inherit
Read ahead sectors     0
Block device           254:5
```

14.4 Linuxスナップショットの削除

- 端末コンソールを開いて、rootユーザとしてログインし、次のように入力します。

  ```
  lvremove snap_volume_path
  ```

  次に例を示します。

  ```
  lvremove /dev/lvm/linux01-snap
  ```

15 ストレージに関する問題のトラブルシュート

このセクションでは、デバイス、ソフトウェア、RAID、マルチパス I/O、およびボリュームの既知の問題に対処する方法について説明します。

15.1 DM-MPIOはブートパーティションに使用できますか?

DM-MPIO (Device Mapper Multipath I/O)は、SUSE® Linux Enterprise Server 10 Support Pack 1から、ブートパーティション用にサポートされています。

マニュアルの更新

A

このセクションでは、SUSE® Linux Enterprise Server 11の最初のリリース以降に、『*SUSE Linux Enterprise Serverストレージ管理ガイド*』に対して行われた文書内容の変更に関する情報を提供します。既存のユーザの場合は、変更箇所を見ると、変更になった内容を簡単に確認できます。新しいユーザの場合は、このままこのガイドを読んでください。

このドキュメントは次の日付に更新されました。

- A.1項 「2010年5月(SLES 11 SP1)」 (220 ページ)
- A.2項 「2010年2月23日」 (222 ページ)
- A.3項 「2010年1月20日」 (223 ページ)
- A.4項 「2009年12月1日」 (224 ページ)
- A.5項 「2009年10月20日」 (225 ページ)
- A.6項 「2009年8月3日」 (227 ページ)
- A.7項 「2009年6月22日」 (227 ページ)
- A.8項 「2009年5月21日」 (229 ページ)

A.1 2010年5月(SLES 11 SP1)

次の項が更新されています。変更内容は次のとおりです。

- A.1.1項 「デバイスのマルチパスI/Oの管理」 (220 ページ)
- A.1.2項 「IPネットワークの大容量記憶域 - iSCSI」 (221 ページ)
- A.1.3項 「新機能」 (222 ページ)

A.1.1 デバイスのマルチパスI/Oの管理

場所	変更
7.2.3項 「マルチパスデバイスでのLVM2の使用」 (62 ページ)	ステップ 3 (63 ページ)の例は修正されました。
7.2.6項 「ルートデバイスがマルチパスの場合のSANタイムアウト設定」 (64 ページ)	このセクションは新たに追加されました。
7.3.2項 「マルチパスI/O管理ツール」 (74 ページ)	パッケージのファイルリストは、サーバのアーキテクチャによって異なることがあります。multipath-toolsパッケージに含まれているファイルのリストについては、「*SUSE Linux Enterprise Server Technical Specifications*」の「*Package Descriptions*」Web ページ [http://www.novell.com/products/server/techspecs.html]にアクセスして、目的のアーキテクチャを見つけ、［パッケージを*名前*でソート］を選択し、次に、「multipath-tools」で検索して、そのアーキテクチャのパッケージリストを見つけます。

場所	変更
7.4.1項 「マルチパス処理用SANデバイスの準備」 (80 ページ)	SANデバイスがサーバ上でルートデバイスとして使用される場合は、7.2.6項 「ルートデバイスがマルチパスの場合のSANタイムアウト設定」 (64 ページ)に示されているように、デバイスのタイムアウト設定を変更します。
7.8.1項 「マルチパスストレージLUNにSLESをインストールするためのマルチパスI/Oの有効化」 (107 ページ)	このセクションは新たに追加されました。
7.8.2項 「アクティブ/パッシブマルチパスストレージLUNにSLESをインストールするためのマルチパスI/Oの有効化」 (108 ページ)	このセクションは新たに追加されました。

A.1.2 IPネットワークの大容量記憶域 - iSCSI

場所	変更
ステップ 7g (198 ページ) での 13.2.2項 「YaSTを使ったiSCSIターゲットの作成」 (197 ページ)	*YaST*の［*Network ServicesiSCSI Target*］機能の［*保存*］オプションを使用すると、iSCSIターゲット情報をエクスポートできます。これにより、この情報をリソースの利用者に簡単に提供することができます。

A.1.3 新機能

場所	変更
2.1項 「SLES 11 SP1の新機能」 (17 ページ)	このセクションは新たに追加されました。

A.2 2010年2月23日

次の項が更新されています。変更内容は次のとおりです。

- A.2.1項 「ルートパーティション用のソフトウェアRAIDの設定」 (222 ページ)
- A.2.2項 「マルチパスI/Oの管理」 (222 ページ)

A.2.1 ルートパーティション用のソフトウェアRAIDの設定

場所	変更
9.1項 「ソフトウェアRAIDの必須条件」 (135 ページ)	RAID 0の定義のエラーを修正しました。

A.2.2 マルチパスI/Oの管理

場所	変更
7.10項 「新規デバイスのスキャン(再起動なし)」 (115 ページ)	再起動せずにデバイスをスキャンできる `rescan-scsi-bus.sh`スクリプトの使用に関する情報を追加しました。

場所	変更
7.11項 「パーティショニングされた新規デバイスのスキャン(再起動なし)」 (118ページ)	再起動せずにデバイスをスキャンできる`rescan-scsi-bus.sh`スクリプトの使用に関する情報を追加しました。

A.3 2010年1月20日

次の節が更新されました。変更内容は次のとおりです。

- A.3.1項 「マルチパスI/Oの管理」 (223 ページ)

A.3.1 マルチパスI/Oの管理

場所	変更
「/etc/multipath.confでのデフォルトマルチパス動作の設定」 (88 ページ)	`default_getuid`コマンドラインでは、サンプルファイル`/usr/share/doc/packages/multipath-tools/multipath.conf.synthetic`内(およびデフォルトの注釈付きサンプルファイルにも含まれている)の`/lib/udev/scsi_id`というサンプルパスの代わりに、上記の例にあるように、パス`/sbin/scsi_id`を使用します。
表7.5「マルチパス属性」 (93 ページ)のgetuid	サンプルファイル(およびデフォルトと注釈付きのサンプルファイルの)`/usr/share/doc/packages/multipath-tools/multipath.conf.synthetic`のパス`/lib/udev/sbin_id`の代わりにパス`/sbin/scsi_id`を使用します。

A.4 2009年12月1日

次の項が更新されています。変更内容は次のとおりです。

- A.4.1項 「デバイスのマルチパスI/Oの管理」 (224 ページ)
- A.4.2項 「ファイルシステムのサイズ変更」 (225 ページ)
- A.4.3項 「新機能」 (225 ページ)

A.4.1 デバイスのマルチパスI/Oの管理

場所	変更
7.2.3項 「マルチパスデバイスでのLVM2の使用」 (62 ページ) 7.9項 「既存ソフトウェアRAID用マルチパスI/Oの設定」 (112 ページ)	-f mpathオプションが-f multipathに変更されました。 mkinitrd -f multipath
表7.5「マルチパス属性」 (93 ページ)のprio_callout	マルチパスのprio_callutsは、`/lib/libmultipath/lib*`内の共有ライブラリ内にあります。共有ライブラリを使用することで、デーモンの起動時、コールアウトがメモリにロードされます。

A.4.2 ファイルシステムのサイズ変更

場所	変更
5.1.1項 「サイズ変更をサポートしているファイルシステム」 (42 ページ)	resize2fsユーティリティが、ext3ファイルシステムのオンライン、またはオフラインのサイズ変更をサポートするようになりました。

A.4.3 新機能

場所	変更
2.2.10項 「マルチパスツールコールアウトの場所の変更」 (26 ページ)	このセクションは新たに追加されました。
2.2.11項 「mkinitrd -fオプションのmpathからmultipathへの変更」 (26 ページ)	このセクションは新たに追加されました。

A.5 2009年10月20日

次の項が更新されています。変更内容は次のとおりです。

- A.5.1項 「LVMの設定」 (226 ページ)
- A.5.2項 「デバイスのマルチパスI/Oの管理」 (226 ページ)
- A.5.3項 「新機能」 (227 ページ)

A.5.1 LVMの設定

場所	変更
4.6項 「LVMの直接管理」 (38 ページ)	YaSTコントロールセンターで、［システム］［パーティショナ］の順に選択します。

A.5.2 デバイスのマルチパスI/Oの管理

場所	変更
「/etc/multipath.confでの非マルチパスデバイスのブラックリスト化」 (87 ページ)	キーワードdevnode_blacklistは廃止され、キーワードblacklistに代わりました。
「/etc/multipath.confでのデフォルトマルチパス動作の設定」 (88 ページ)	getuid_calloutがgetuidに変更されました。
「優先度グループと属性の理解」 (93 ページ)	getuid_calloutがgetuidに変更されました。
「優先度グループと属性の理解」 (93 ページ)	least-pending(最小保留)、length-load-balancing(長さによる負荷分散)、およびservice-time(サービス時間)のオプションの説明を追加しました。

A.5.3 新機能

場所	変更
2.2.9項 「マルチパスのための高度な入出力負荷バランスオプション」 (26 ページ)	このセクションは新たに追加されました。

A.6 2009年8月3日

次の節が更新されました。この変更については次に説明しています。

- A.6.1項 「マルチパスI/Oの管理」 (227 ページ)

A.6.1 マルチパスI/Oの管理

場所	変更
7.2.5項 「マルチパスデバイスでの--noflushの使用」 (64 ページ)	このセクションは新たに追加されました。

A.7 2009年6月22日

次の項が更新されています。変更内容は次のとおりです。

- A.7.1項 「マルチパスI/Oの管理」 (228 ページ)
- A.7.2項 「mdadmによるソフトウェアRAID 6および10の管理」 (228 ページ)
- A.7.3項 「IPネットワークの大容量記憶域 - iSCSI」 (229 ページ)

A.7.1 マルチパスI/Oの管理

場所	変更
7.8項 「ルートデバイスのマルチパスI/Oの設定」 (106 ページ)	System Zにステップ 4 (111 ページ)およびステップ 6 (111 ページ)を追加しました。
7.11項 「パーティショニングされた新規デバイスのスキャン(再起動なし)」 (118 ページ)	ステップ2のコマンドラインの構文を修正しました。
7.11項 「パーティショニングされた新規デバイスのスキャン(再起動なし)」 (118 ページ)	ステップ 7 (119 ページ)ではステップ7とステップ8を変更しました。

A.7.2 mdadmによるソフトウェアRAID 6および10の管理

場所	変更
10.4項 「ディグレードアレイの作成」 (157 ページ)	毎秒更新されている間に再構築の進捗を確認するには、次のように入力します。 `watch -n 1 cat /proc/mdstat`

A.7.3 IPネットワークの大容量記憶域 - iSCSI

場所	変更
13.3.1項 「YaSTを使ったiSCSIイニシエータの設定」 (205 ページ)	わかりやすくするため、内容を整理しました。 iSCSIターゲットデバイスの起動オプションの設定の使用法に関する情報を追加しました。 • **自動:** このオプションは、iSCSIサービス自体の起動時に接続するiSCSIターゲットに使用されます。これが通常の設定です。 • **Onboot(起動時):** このオプションは、起動時、つまりルート(/)がiSCSI上にある場合に接続するiSCSIターゲットに使用します。したがって、iSCSIターゲットデバイスはサーバの起動時にinitrdによって評価されます。

A.8 2009年5月21日

次の節が更新されました。変更内容は次のとおりです。

• A.8.1項 「マルチパスI/Oの管理」 (230 ページ)

A.8.1 マルチパスI/Oの管理

場所	変更
「マルチパス処理用に自動検出されるストレージアレイ」 (67 ページ)	IBM zSeriesデバイスをマルチパス処理でテストした結果、dev_loss_tmoパラメータを90秒に、fast_io_fail_tmoパラメータを5秒に設定する必要があることわかりました。zSeriesデバイスをご使用の場合は、手動で/etc/multipath.confファイルの作成と設定を行い、値を指定する必要があります。詳細については、「/etc/multipath.conf内のzSeriesのデフォルト設定」 (89 ページ)を参照してください。
7.3.1項 「デバイスマッパーマルチパスモジュール」 (71 ページ)	マルチパス処理は、SUSE Linux Enterprise Server 11以降の/bootデバイスに対してサポートされています。
「/etc/multipath.conf内のzSeriesのデフォルト設定」 (89 ページ)	このセクションは新たに追加されました。
7.8項 「ルートデバイスのマルチパスI/Oの設定」 (106 ページ)	SUSE Linux Enterprise Server 11では、/bootおよび/root用のDM-MPがサポート、および提供されるようになりました。