Oracle® Solaris ZFS 管理ガイド

Part No: 819–6260–17
2010 年 9 月

ORACLE®

100908@24661

目次

はじめに

『Oracle Solaris ZFS 管理ガイド』では、Oracle Solaris ZFS ファイルシステムの設定と管理について説明します。

本書では、SPARC および x86 の両方のプラットフォームにおけるシステム管理について解説しています。

注 – この Oracle Solaris リリースでは、SPARC および x86 系列のプロセッサアーキテクチャー (UltraSPARC、SPARC64、AMD64、Pentium、Xeon EM64T) を使用するシステムをサポートします。サポートされるシステムについては、Solaris 10 Hardware Compatibility List (http://www.sun.com/bigadmin/hcl) を参照してください。本書では、プラットフォームにより実装が異なる場合は、それを特記します。

本書の x86 に関する用語については、以下を参照してください。

- 「x86」は、64 ビットおよび 32 ビットの x86 互換製品系列を指します。
- 「x64」は、AMD64 または EM64T システムに関する 64 ビット特有の情報を指します。
- 「32 ビット x86」は、x86 をベースとするシステムに関する 32 ビット特有の情報を指します。

サポートされるシステムについては、Solaris 10 Hardware Compatibility List を参照してください。

対象読者

本書は、Oracle Solaris ZFS ファイルシステムの設定と管理に関係するすべてのユーザーを対象としています。Oracle Solaris オペレーティングシステム (OS) または別のバージョンの UNIX を使用した経験があることが推奨されます。

内容の紹介

次の表で、本書の各章について説明します。

章	説明
第 1 章「Oracle Solaris ZFS ファイルシステム (概要)」	ZFS の概要およびその機能と利点について説明します。また、基本的な概念と用語について説明します。
第 2 章「Oracle Solaris ZFS 入門」	基本的なプールとファイルシステムを使って基本的な ZFS 構成を設定する手順について説明します。この章では、ZFS ファイルシステムの作成に必要なハードウェアとソフトウェアについても説明します。
第 3 章「Oracle Solaris ZFS ファイルシステムと従来のファイルシステムの相違点」	ZFS の機能のうち、従来のファイルシステムと大きく異なる重要な機能について説明します。これらの重要な相違点を理解していれば、従来のツールを使用して ZFS を操作するときの混乱を少なくできます。
第 4 章「Oracle Solaris ZFS ストレージプールの管理」	ZFS ストレージプールの作成および管理方法について詳しく説明します。
第 5 章「Oracle Solaris ZFS ルートファイルシステムのインストールと起動」	ZFS ファイルシステムのインストールと起動の方法について説明します。Oracle Solaris Live Upgrade を使用して UFS ルートファイルシステムを ZFS ルートファイルシステムに移行する方法についても説明します。
第 6 章「Oracle Solaris ZFS ファイルシステムの管理」	ZFS ファイルシステムの管理について詳しく説明します。たとえば、ファイルシステムの階層レイアウト、プロパティーが継承されること、およびマウントポイント管理および共有が自動的に行われることなどについて、それらの概念を説明します。
第 7 章「Oracle Solaris ZFS のスナップショットとクローンの操作」	ZFS のスナップショットとクローンを作成および管理する方法について説明します。
第 8 章「ACL による Oracle Solaris ZFS ファイルの保護」	アクセス制御リスト (ACL) を使用して UNIX 標準のアクセス権よりアクセス権を詳細に設定することで、ZFS ファイルを保護する方法について説明します。
第 9 章「Oracle Solaris ZFS 委任管理」	ZFS 委任管理を使用して、特権のないユーザーが ZFS 管理タスクを実行できるようにする方法について説明します。
第 10 章「Oracle Solaris ZFS の高度なトピック」	ZFS ボリュームの使用について、ゾーンがインストールされた Oracle Solaris システムでの ZFS の使用について、および代替ルートプールの使用について説明します。
第 11 章「Oracle Solaris ZFS のトラブルシューティングとプールの回復」	ZFS の障害を識別してそこから回復する方法について説明します。また、障害を回避する方法について説明します。

章	説明
付録 A 「Oracle Solaris ZFS バージョンの説明」	利用可能な ZFS のバージョン、各バージョンの機能、および Solaris OS の各リリースで提供される ZFS のバージョンと機能について説明します。

関連情報

Oracle Solaris システム管理の一般的なトピックに関する情報については、次のマニュアルを参照してください。

- 『Solaris のシステム管理 (基本編)』
- 『Solaris のシステム管理 (上級編)』
- 『Solaris のシステム管理 (デバイスとファイルシステム)』
- 『Solaris のシステム管理 (セキュリティサービス)』

マニュアル、サポート、およびトレーニング

追加リソースについては、次の Web サイトを参照してください。

- マニュアル (`http://docs.sun.com`)
- サポート (`http://www.oracle.com/us/support/systems/index.html`)
- トレーニング (`http://education.oracle.com`) – 左のナビゲーションバーで「Sun」のリンクをクリックします。

Oracle へのご意見

Oracle はドキュメントの品質向上のために、お客様のご意見やご提案をお待ちしています。誤りを見つけたり、改善に向けた提案などがある場合は、`http://docs.sun.com` で「Feedback」をクリックしてください。可能な場合には、ドキュメントのタイトルやパート番号に加えて、章、節、およびページ番号を含めてください。返信を希望するかどうかもお知らせください。

Oracle Technology Network (`http://www.oracle.com/technetwork/index.html`) では、Oracle ソフトウェアに関する広範なリソースが提供されています。

- ディスカッションフォーラム (`http://forums.oracle.com`) で技術的な問題や解決策を話し合う。
- Oracle By Example (`http://www.oracle.com/technology/obe/start/index.html`) のチュートリアルで、手順に従って操作を体験する。
- サンプルコード (`http://www.oracle.com/technology/sample_code/index.html`) をダウンロードする。

表記上の規則

このマニュアルでは、次のような字体や記号を特別な意味を持つものとして使用します。

表 P–1 表記上の規則

字体または記号	意味	例
AaBbCc123	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、コード例を示します。	.login ファイルを編集します。 ls -a を使用してすべてのファイルを表示します。 system%
AaBbCc123	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して示します。	system% **su** password:
AaBbCc123	変数を示します。実際に使用する特定の名前または値で置き換えます。	ファイルを削除するには、rm *filename* と入力します。
『 』	参照する書名を示します。	『コードマネージャ・ユーザーズガイド』を参照してください。
「 」	参照する章、節、ボタンやメニュー名、強調する単語を示します。	第 5 章「衝突の回避」を参照してください。 この操作ができるのは、「スーパーユーザー」だけです。
\	枠で囲まれたコード例で、テキストがページ行幅を超える場合に、継続を示します。	sun% **grep '^#define ** **XV_VERSION_STRING'**

Oracle Solaris OS に含まれるシェルで使用する、UNIX のデフォルトのシステムプロンプトとスーパーユーザープロンプトを次に示します。コマンド例に示されるデフォルトのシステムプロンプトは、Oracle Solaris のリリースによって異なります。

- C シェル

  machine_name% **command y|n** [*filename*]

- C シェルのスーパーユーザー

  machine_name# **command y|n** [*filename*]

- Bash シェル、Korn シェル、および Bourne シェル

  $ **command y|n** [*filename*]

- Bash シェル、Korn シェル、および Bourne シェルのスーパーユーザー

```
# command y|n [filename]
```

[ ] は省略可能な項目を示します。上記の例は、*filename* は省略してもよいことを示しています。

| は区切り文字 (セパレータ) です。この文字で分割されている引数のうち 1 つだけを指定します。

キーボードのキー名は英文で、頭文字を大文字で示します (例: Shift キーを押します)。ただし、キーボードによっては Enter キーが Return キーの動作をします。

ダッシュ (-) は 2 つのキーを同時に押すことを示します。たとえば、Ctrl-D は Control キーを押したまま D キーを押すことを意味します。

◆ ◆ ◆ 第 1 章

Oracle Solaris ZFS ファイルシステム (概要)

この章では、Oracle Solaris ZFS ファイルシステムの概要およびその機能と利点について説明します。また、このマニュアルの残りの章で使用されるいくつかの基本的な用語について説明します。

この章は、次の節で構成されます。

ZFS の新機能

この節では、ZFS ファイルシステムの新機能について概説します。

ミラー化された ZFS ストレージプールの分割 (zpool split)

Oracle Solaris 10 9/10 リリース: この Solaris リリースでは、zpool split コマンドを使用して、ミラー化されたストレージプールを分割できます。これにより、元のミラー化プール内の 1 つまたは複数のディスクが切り離され、別の同一のプールが作成されます。

詳細は、89 ページの「ミラー化 ZFS ストレージプールを分割して新しいプールを作成する」を参照してください。

新しい ZFS システムプロセス

Oracle Solaris 10 9/10 リリース: この Solaris リリースでは、個々の ZFS ストレージプールに zpool-*poolname* という名前のプロセスが関連付けられます。このプロセス内のスレッドは、プールと関連付けられた圧縮やチェックサム計算などの入出力タスクを処理するための、プールの入出力処理スレッドです。このプロセスの目的は、各ストレージプールの CPU 使用率を確認できるようにすることです。これらのプロセスについての情報は、ps および prstat コマンドを使用して確認できます。これらのプロセスは大域ゾーンでのみ使用可能です。詳細は、SDC(7)を参照してください。

zpool list コマンドの変更点

Oracle Solaris 10 9/10 リリース: この Solaris リリースでは、より詳細な領域割り当て情報を提供するように zpool list の出力が変更されています。次に例を示します。

```
# zpool list tank
NAME    SIZE  ALLOC   FREE    CAP  HEALTH  ALTROOT
tank    136G  55.2G  80.8G    40%  ONLINE  -
```

以前の USED および AVAIL フィールドは、ALLOC および FREE に置き換えられました。

ALLOC フィールドは、すべてのデータセットおよび内部メタデータに割り当てられた物理領域の容量を示します。FREE フィールドは、プール内で割り当てられていない領域の容量を示します。

詳細は、107 ページの「ZFS ストレージプールについての情報を表示する」を参照してください。

ZFS ストレージプールの復旧

Oracle Solaris 10 9/10 リリース: 配下のデバイスが利用不能になった場合、電源障害が発生した場合、または冗長 ZFS 構成でサポートされている数よりも多くのデバイスで障害が発生した場合、ストレージプールが損傷を受ける可能性があります。このリリースでは、損傷したストレージプールを復旧するための新しいコマンド機能が用意されています。ただし、この復旧機能を使用すると、プールが機能停止する前に発生した最後の数回分のトランザクションが失われる場合があります。

zpool clear および zpool import コマンドはどちらも、損傷したプールを復旧する可能性のある -F オプションをサポートします。また、zpool status、zpool clear、または zpool import コマンドを実行すると損傷したプールが自動的に報告され、これらのコマンドはプールの復旧方法を説明します。

詳細は、321 ページの「ZFS ストレージプール全体の損傷を修復する」を参照してください。

ログデバイスに関する ZFS の拡張機能

Oracle Solaris 10 9/10 リリース: ログデバイスに関する次の拡張機能が使用できます。

- `logbias` プロパティー – このプロパティーを使用すると、特定のデータセットに対して同時に発生する要求の処理についてのヒントを ZFS に指示できます。`logbias` が `latency` に設定されている場合、ZFS はプールに別個のログデバイスが存在すればそれを使用して、短い待ち時間で要求を処理します。`logbias` が `throughput` に設定されている場合、ZFS はプールの別個のログデバイスを使用しません。その代わりに、ZFS は大域プールのスループットとリソースの使用効率を優先して同時操作を最適化します。デフォルト値は `latency` です。ほとんどの構成でデフォルト値が推奨されます。`logbias=throughput` の値を使用すると、データベースファイルの書き込みパフォーマンスが向上する場合があります。
- ログデバイスの削除 – `zpool remove` コマンドを使用して、ログデバイスを ZFS ストレージプールから削除できるようになりました。単一のログデバイスは、デバイス名を指定することによって削除できます。ミラー化されたログデバイスは、ログの最上位レベルのミラーを指定することによって削除できます。別個のログデバイスがシステムから削除されると、ZIL トランザクションレコードがメインプールに書き込まれます。

  最上位レベルの冗長仮想デバイスが数値 ID で識別されるようになりました。たとえば、2 台のディスクで構成されるミラー化ストレージプールで、最上位レベルの仮想デバイスは `mirror-0` です。

  詳細は、例 4–3 を参照してください。

トリプルパリティー RAIDZ (raidz3)

Oracle Solaris 10 9/10 リリース: この Solaris リリースでは、冗長な RAID-Z 構成でシングルパリティー、ダブルパリティー、またはトリプルパリティーを使用できるようになりました。これはそれぞれ、1 つ、2 つ、または 3 つのデバイスで障害が発生しても、データを失うことなく処理を続行できることを意味します。raidz3 キーワードを指定すれば、トリプルパリティー RAID-Z 構成にすることができます。詳細は、74 ページの「RAID-Z ストレージプールを作成する」を参照してください。

ZFS スナップショットの保持

Oracle Solaris 10 9/10 リリース: 異なる自動スナップショットポリシーを実装しており、送信側にもう存在しないという理由で古いスナップショットが `zfs receive` によって意図せず破棄されてしまう場合、この Solaris リリースのスナップショット保持機能の使用を検討することができます。

スナップショットを保持すると、そのスナップショットは破棄されなくなります。また、この機能と zfs destroy -d コマンドを使用することにより、最後のクローンの消去を保留しながら、クローンが存在するスナップショットを削除できます。

1 つのスナップショットまたはスナップショットの集合を保持できます。たとえば次の構文は、保持タグ keep を tank/home/cindys/snap@1 に付与します。

```
# zfs hold keep tank/home/cindys@snap1
```

詳細は、227 ページの「ZFS スナップショットの保持」を参照してください。

デバイスの置き換えに関する ZFS の拡張機能

Oracle Solaris 10 9/10 リリース: この Solaris リリースでは、配下のデバイスが拡張されたときのシステムイベント (*sysevent*) を使用できます。これらのイベントを認識し、autoexpand プロパティーの設定と拡張された LUN の新しいサイズに基づいてプールを調整できるように ZFS の機能が拡張されています。動的な LUN の拡張イベントを受信したときの自動プール拡張を、プールの autoexpand プロパティーを使用して有効または無効にできます。

これらの機能を利用すると、LUN を拡張したあとでプールのエクスポートとインポートまたはシステムの再起動を行わなくても、プールの拡張された領域にアクセスすることができます。

たとえば、tank プールで LUN の自動拡張を有効にします。

```
# zpool set autoexpand=on tank
```

または、autoexpand プロパティーを有効にしてプールを作成できます。

```
# zpool create -o autoexpand=on tank c1t13d0
```

autoexpand プロパティーはデフォルトで無効なため、LUN を拡張するかどうかは自由に決定できます。

zpool online - e コマンドを使って LUN を拡張することもできます。次に例を示します。

```
# zpool online -e tank c1t6d0
```

あるいは、zpool replace 機能を使って LUN を接続または使用可能にしたあとで autoexpand プロパティーを再設定できます。たとえば、8G バイトのディスク 1 台 (c0t0d0) で構成される次のプールを作成します。8G バイトのディスクを 16G バイトのディスク (c1t13d0) に置き換えても、autoexpand プロパティーを有効にするまでプールのサイズは拡張されません。

```
# zpool create pool c0t0d0
# zpool list
NAME   SIZE   ALLOC  FREE    CAP   HEALTH  ALTROOT
pool   8.44G  76.5K  8.44G     0%  ONLINE  -
# zpool replace pool c0t0d0 c1t13d0
# zpool list
NAME   SIZE   ALLOC  FREE    CAP   HEALTH  ALTROOT
pool   8.44G  91.5K  8.44G     0%  ONLINE  -
# zpool set autoexpand=on pool
# zpool list
NAME   SIZE   ALLOC  FREE    CAP   HEALTH  ALTROOT
pool   16.8G   91.5K  16.8G    0%  ONLINE  -
```

前の例で autoexpand プロパティーを有効にせずに LUN を拡張する別の方法として、デバイスがすでにオンラインであるにもかかわらず zpool online - e コマンドを使用することができます。次に例を示します。

```
# zpool create tank c0t0d0
# zpool list tank
NAME   SIZE   ALLOC  FREE    CAP   HEALTH  ALTROOT
tank   8.44G  76.5K  8.44G     0%  ONLINE  -
# zpool replace tank c0t0d0 c1t13d0
# zpool list tank
NAME   SIZE   ALLOC  FREE    CAP   HEALTH  ALTROOT
tank   8.44G  91.5K  8.44G     0%  ONLINE  -
# zpool online -e tank c1t13d0
# zpool list tank
NAME   SIZE   ALLOC  FREE    CAP   HEALTH  ALTROOT
tank   16.8G    90K  16.8G     0%  ONLINE  -
```

デバイスの置き換えに関するこのリリースのその他の拡張機能には、次のものがあります。

- 以前のリリースの ZFS では、交換用ディスクのサイズが少しでも違っていると、既存のディスクを別のディスクと交換したり、ディスクを接続したりできませんでした。このリリースでは、プールにまだ空きがある場合であれば、既存のディスクをほぼ同じサイズの別のディスクと交換したり、ほぼ同じサイズの新しいディスクを接続したりできます。
- このリリースでは、LUN を拡張するためにシステムを再起動したり、プールをエクスポートしてインポートしたりする必要はありません。すでに説明したように、 autoexpand プロパティーを有効にするか、zpool online -e コマンドを使うことによって LUN の全体サイズを拡張できます。

デバイスの置き換えについては、96 ページの「ストレージプール内のデバイスを置き換える」を参照してください。

ZFS インストールおよびフラッシュインストールのサポート

Solaris 10 10/09 リリース: この Solaris リリースでは、JumpStart プロファイルを設定して、ZFS ルートプールのフラッシュアーカイブを特定できます。詳細は、137 ページの「ZFS ルートファイルシステムのインストール (Oracle Solaris フラッシュアーカイブインストール)」を参照してください。

ZFS ユーザーおよびグループの割り当て制限

Solaris 10 10/09 リリース: 以前の Solaris リリースでは、割り当て制限と予約を ZFS ファイルシステムに適用して、ディスク領域の管理および予約を行っていました。

Solaris のこのリリースでは、特定のユーザーやグループが所有するファイルによって消費されるディスク容量に割り当て制限を設定できます。例えば、多数のユーザーまたグループが存在する環境でユーザーおよびグループの割り当て制限を設定する場合もあります。

ユーザーの割り当て制限は、`zfs userquota` プロパティーを使用して設定できます。グループの割り当て制限を設定するには、`zfs groupquota` プロパティーを使用します。次に例を示します。

```
# zfs set userquota@user1=5G tank/data
# zfs set groupquota@staff=10G tank/staff/admins
```

ユーザーまたはグループの現在の割り当て制限の設定を表示するには、次のようにします。

```
# zfs get userquota@user1 tank/data
NAME       PROPERTY         VALUE            SOURCE
tank/data  userquota@user1  5G               local
# zfs get groupquota@staff tank/staff/admins
NAME              PROPERTY          VALUE              SOURCE
tank/staff/admins  groupquota@staff  10G                local
```

割り当て制限に関する一般的な情報を表示するには、次のようにします。

```
# zfs userspace tank/data
TYPE        NAME   USED  QUOTA
POSIX User  root     3K   none
POSIX User  user1    0    5G

# zfs groupspace tank/staff/admins
TYPE         NAME   USED  QUOTA
POSIX Group  root     3K   none
POSIX Group  staff    0    10G
```

個別ユーザーのディスク容量使用状況は、userused@*user* プロパティーを使って表示できます。グループのディスク容量使用状況は、groupused@*group* プロパティーを使って表示できます。次に例を示します。

```
# zfs get userused@user1 tank/staff
NAME        PROPERTY        VALUE           SOURCE
tank/staff  userused@user1  213M            local
# zfs get groupused@staff tank/staff
NAME        PROPERTY         VALUE            SOURCE
tank/staff  groupused@staff  213M             local
```

ユーザーの割り当て制限の設定の詳細については、218 ページの「ZFS の割り当て制限と予約を設定する」を参照してください。

実行権の ZFS ACL パススルー継承

Solaris 10 10/09 リリース**:** 以前の Solaris リリースでは、すべてのファイルが 0664 アクセス権または 0666 アクセス権付きで作成されるように、ACL 継承を適用することができました。このリリースでは、ファイル作成モードの実行ビットを継承後の ACL に必要に応じて含める場合、aclinherit モードを設定して、実行権を継承後の ACL に渡すことができます。

ZFS データセットで aclinherit=passthrough-x を有効にすると、cc または gcc コンパイラツールによって生成される出力ファイルに対する実行権を含めることができます。継承された ACL に実行権が含まれていない場合、コンパイラからの実行可能な出力は、chmod コマンドを使用してファイルのアクセス権を変更するまで実行できません。

詳細は、例 8–12 を参照してください。

ZFS プロパティーの拡張機能

Solaris 10 10/09 および **Oracle Solaris 10 9/10**: これらのリリースでは、ZFS ファイルシステムに関する拡張機能として、次のものが用意されています。

- スナップショットストリームプロパティーに関する **ZFS** の拡張機能 – ローカルのプロパティー設定と異なる受信プロパティーを設定できます。たとえば、圧縮プロパティーが無効に設定されたストリームを受信する一方で、受信側ファイルシステムでは圧縮を有効にする場合を考えます。このとき、受信するストリームに関して、圧縮プロパティーの値が受信側ではオフ、ローカルではオンということになります。ローカルの値は受信側の値に優先するため、送信側の設定によって受信側の値が置き換えられることを心配する必要はありません。zfs get コマンドを実行すると、圧縮プロパティーの実効値が VALUE 列の下に表示されます。

  送信側とローカルのプロパティー値に関連した、ZFS の新しいコマンドオプションとプロパティーには次のものがあります。

- ローカルのプロパティー値を受信値 (存在する場合) に戻すには、zfs inherit -S を使用します。プロパティーに受信値が存在しない場合、zfs inherit - S コマンドの動作は、-S オプションを指定しない zfs inherit コマンドと同じです。プロパティーに受信値が存在する場合、zfs inherit コマンドは、zfs inherit -S コマンドの発行によって継承値が受信値に戻されるまでの間、受信値を継承値でマスクします。
- zfs get -o を使用すると、新しい非デフォルトの RECEIVED 列を含めることができます。または、zfs get -o all コマンドを使用すると、RECEIVED を含むすべての列を含めることができます。
- zfs send -p オプションを使用すると、-R オプションを使用せずにプロパティーを送信ストリームに含めることができます。

また、zfs send -e オプションを使用すると、送信スナップショットの最後の要素を使用して新しいスナップショット名を決定できます。次の例では、poola/bee/cee@1 スナップショットを poold/eee ファイルシステムに送信し、スナップショット名の最後の要素 (cee@1) のみを使用して、受信側のファイルシステムおよびスナップショットを作成します。

```
# zfs list -rt all poola
NAME              USED  AVAIL  REFER  MOUNTPOINT
poola             134K   134G    23K  /poola
poola/bee          44K   134G    23K  /poola/bee
poola/bee/cee      21K   134G    21K  /poola/bee/cee
poola/bee/cee@1      0      -    21K  -
# zfs send -R poola/bee/cee@1 | zfs receive -e poold/eee
# zfs list -rt all poold
NAME              USED  AVAIL  REFER  MOUNTPOINT
poold             134K   134G    23K  /poold
poold/eee          44K   134G    23K  /poold/eee
poold/eee/cee      21K   134G    21K  /poold/eee/cee
poold/eee/cee@1      0      -    21K  -
```

- プール作成時の **ZFS** ファイルシステムプロパティーの設定 - ストレージプールを作成するときに ZFS ファイルシステムプロパティーを設定することができます。次の例では、プールと一緒に作成された ZFS ファイルシステムで圧縮が有効になっています。

```
# zpool create -O compression=on pool mirror c0t1d0 c0t2d0
```

- **ZFS** ファイルシステムにおけるキャッシュプロパティーの設定 - 2 つの新しい ZFS ファイルシステムプロパティーを使用して、プライマリキャッシュ (ARC) およびセカンダリキャッシュ (L2ARC) にキャッシュされる内容を制御できます。キャッシュのプロパティーは、次のように設定します。
  - primarycache – ARC にキャッシュされる内容を制御します。
  - secondarycache – L2ARC にキャッシュされる内容を制御します。

- 両方のプロパティーに指定できる値 – all、none、metadata。all に設定すると、ユーザーデータとメタデータの両方がキャッシュされます。none に設定すると、ユーザーデータも、メタデータも、キャッシュされません。metadata に設定すると、メタデータのみがキャッシュされます。デフォルトは all です。

これらのプロパティーは、既存のファイルシステムに設定できます。または、ファイルシステムの作成時に設定できます。次に例を示します。

```
# zfs set primarycache=metadata tank/datab
# zfs create -o primarycache=metadata tank/newdatab
```

既存のファイルシステムでこれらのプロパティーを設定した場合、これらのプロパティーの値に基づいて、New I/O のみがキャッシュされます。

一部のデータベース環境では、ユーザーデータをキャッシュしないほうが利点が得られることがあります。キャッシュプロパティーの設定が、使用している環境に照らし合わせて適切かどうかを判定する必要があります。

- ディスクスペースアカウンティングプロパティーの表示 – 新しい読み取り専用ファイルシステムプロパティーであり、クローン、ファイルシステム、ボリューム、およびスナップショットに関するディスク領域使用状況を確認する場合に役立ちます。属性は次のとおりです。
  - usedbychildren – このデータセットの子によって使用されるディスク領域の量を特定します。この領域は、データセットのすべての子が破棄されると、解放されます。このプロパティーの省略名は usedchild です。
  - usedbydataset – このデータセット自体によって使用されるディスク領域の量を特定します。この領域は、最初にあらゆるスナップショットが破棄されてから refreservation がすべて削除された後に、データセットが破棄されると、解放されます。このプロパティーの省略名は usedds です。
  - usedbyrefreservation – このデータセットに設定されている refreservation によって使用されるディスク領域の量を特定します。この領域は、refreservation が削除されると、解放されます。このプロパティーの省略名は usedrefreserv です。
  - usedbysnapshots – このデータセットのスナップショットによって消費されるディスク領域の量を特定します。この領域は、このデータセットのすべてのスナップショットが破棄されると、解放されます。これはスナップショットの used プロパティーの値を単純に合計した結果ではないことに注意してください。複数のスナップショットで共有されているディスク容量も存在するためです。このプロパティーの省略名は usedsnap です。

これらの新しいプロパティーは、used プロパティーの値を、ディスク容量を消費する各種の要素に分割します。具体的には、used プロパティーの値は次のように分割されます。

```
used property = usedbychildren + usedbydataset + usedbyrefreservation + usedbysnapshots
```

これらのプロパティーは、zfs list -o space コマンドを使用して表示できます。次に例を示します。

```
$ zfs list -o space
NAME                AVAIL   USED  USEDSNAP  USEDDS  USEDREFRESERV  USEDCHILD
rpool               25.4G  7.79G         0     64K              0      7.79G
rpool/ROOT          25.4G  6.29G         0     18K              0      6.29G
rpool/ROOT/snv_98   25.4G  6.29G         0   6.29G              0          0
rpool/dump          25.4G  1.00G         0   1.00G              0          0
rpool/export        25.4G    38K         0     20K              0        18K
rpool/export/home   25.4G    18K         0     18K              0          0
rpool/swap          25.8G   512M         0    111M           401M          0
```

前述のコマンドは、zfs list - o name,avail,used,usedsnap,usedds,usedrefreserv,usedchild -t filesystem,volume コマンドと同等です。

- スナップショットの表示– listsnapshots プールプロパティーは、zfs list コマンドでスナップショット情報が表示されるかどうかを制御します。デフォルト値は on です。つまり、デフォルトでは、スナップショット情報は表示されます。

  システムに存在する ZFS スナップショットの数が多い場合に、zfs list コマンドでスナップショット情報の表示を無効にするには、次のようにして listsnapshots プロパティーを無効にします。

  ```
  # zpool get listsnapshots pool
  NAME  PROPERTY       VALUE      SOURCE
  pool  listsnapshots  on         default
  # zpool set listsnaps=off pool
  ```

  listsnapshots プロパティーを無効にした場合、zfs list -t snapshots コマンドを使用してスナップショット情報を一覧表示できます。次に例を示します。

  ```
  # zfs list -t snapshot
  NAME                    USED  AVAIL  REFER  MOUNTPOINT
  pool/home@today          16K      -    22K  -
  pool/home/user1@today      0      -    18K  -
  pool/home/user2@today      0      -    18K  -
  pool/home/user3@today      0      -    18K  -
  ```

ZFS ログデバイスの回復

Solaris 10 10/09 リリース: このリリースでは、zpool status コマンド出力におけるインテントログ障害を ZFS が認識します。これらのエラーは障害管理アーキテクチャー (FMA) によっても報告されます。ZFS と FMA は両方とも、インテントログ障害から回復する方法を説明します。

たとえば、別個のログデバイスを持つプールに対する同期書き込み操作が確定される前にシステムが突然シャットダウンされた場合には、次のようなメッセージが表示されます。

```
# zpool status -x
  pool: pool
 state: FAULTED
status: One or more of the intent logs could not be read.
        Waiting for adminstrator intervention to fix the faulted pool.
action: Either restore the affected device(s) and run 'zpool online',
        or ignore the intent log records by running 'zpool clear'.
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        pool        FAULTED      0     0     0 bad intent log
          mirror    ONLINE       0     0     0
            c0t1d0  ONLINE       0     0     0
            c0t4d0  ONLINE       0     0     0
        logs        FAULTED      0     0     0 bad intent log
          c0t5d0    UNAVAIL      0     0     0 cannot open
```

そのような場合には、次の方法でログデバイスの障害を解決できます。

- ログデバイスを交換または回復します(この例の場合、ログデバイスは `c0t5d0`)。
- ログデバイスをオンラインに戻します。

  ```
  # zpool online pool c0t5d0
  ```
- 障害が発生したログデバイスのエラー状況がリセットされます。

  ```
  # zpool clear pool
  ```

障害が発生したログデバイスを交換せずにこのエラーから回復するために、`zpool clear` コマンドを使用してエラーを解決することができます。このシナリオでは、プールが縮退モードで実行され、ログレコードは、ログデバイスが交換されるまで、メインプールに書き込まれます。

ログデバイスの障害の発生を回避するため、ミラー化ログデバイスを利用することを検討してください。

ZFS ストレージプールにおけるキャッシュデバイスの使用

Solaris 10 10/09 リリース: このリリースでは、プールを作成するとき、ストレージプールデータをキャッシュするために使用されるキャッシュデバイスを指定することができます。

キャッシュデバイスにより、メインメモリーとディスクの間にキャッシュ層が追加されます。キャッシュデバイスを使用すると、ほぼ静的なコンテンツをランダムに読み込む作業負荷のパフォーマンスが大幅に向上します。

プールの作成時に 1 つ以上のキャッシュデバイスを指定できます。次に例を示します。

```
# zpool create pool mirror c0t2d0 c0t4d0 cache c0t0d0
# zpool status pool
  pool: pool
 state: ONLINE
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        pool        ONLINE       0     0     0
          mirror    ONLINE       0     0     0
            c0t2d0  ONLINE       0     0     0
            c0t4d0  ONLINE       0     0     0
        cache
          c0t0d0    ONLINE       0     0     0

errors: No known data errors
```

キャッシュデバイスを追加すると、そのキャッシュデバイスにメインメモリーの内容が徐々に書き込まれていきます。キャッシュデバイスのサイズによっては、デバイスがいっぱいになるまでに 1 時間以上かかる場合もあります。`zpool iostat` コマンドを次のように使用して、容量と読み込みを監視できます。

```
# zpool iostat -v pool 5
```

プールの作成後に、プールに対してキャッシュデバイスの追加や削除を行うことができます。

詳細は、76 ページの「キャッシュデバイスを使用して ZFS ストレージプールを作成する」および例 4–4 を参照してください。

ZFS 環境でのゾーンの移行

Solaris 10 5/09 リリース: このリリースでは、ZFS 環境で Oracle Solaris Live Upgrade を使用してゾーンを移行するためのサポートが拡張されています。詳細は、156 ページの「ゾーンが含まれているシステムを Oracle Solaris Live Upgrade で移行またはアップグレードする (Solaris 10 5/09 以降)」を参照してください。

このリリースで認識されている問題のリストについては、『Solaris 10 5/09 ご使用にあたって』を参照してください。

ZFS のインストールおよび起動のサポート

Solaris 10 10/08 リリース: このリリースでは、ZFS ルートファイルシステムをインストールおよび起動できます。初期インストールまたは JumpStart 機能を使用して、ZFS ルートファイルシステムをインストールできます。あるいは、Oracle Solaris Live Upgrade を使用して、UFS ルートファイルシステムを ZFS ルートファイルシステムに移行できます。スワップデバイスとダンプデバイスに関する ZFS のサポートも追加されています。詳細は、第 5 章「Oracle Solaris ZFS ルートファイルシステムのインストールと起動」を参照してください。

このリリースで認識されている問題のリストについては、次のサイトを参照してください。

http://hub.opensolaris.org/bin/view/Community+Group+zfs/boot

『Solaris 10 10/08 ご使用にあたって』も参照してください。

マウント解除せずにデータセットをロールバックする

Solaris 10 10/08 リリース: このリリースでは、事前にマウント解除しなくてもデータセットをロールバックできます。そのため、マウント解除処理を強制的に行う `zfs rollback -f` オプションは不要になりました。`-f` オプションは今後はサポートされず、指定しても無視されます。

zfs send コマンドの拡張機能

Solaris 10 10/08 リリース**:** このリリースでは、`zfs send` コマンドに次の拡張機能が追加されています。このコマンドを使用して、次の操作を実行できるようになりました。

- 1 つのスナップショットのすべての増分ストリームを累積スナップショットに送信します。次に例を示します。

```
# zfs list
NAME                      USED  AVAIL  REFER  MOUNTPOINT
pool                      428K  16.5G    20K  /pool
pool/fs                    71K  16.5G    21K  /pool/fs
pool/fs@snapA              16K      -  18.5K  -
pool/fs@snapB              17K      -    20K  -
pool/fs@snapC              17K      -  20.5K  -
pool/fs@snapD                0      -    21K  -
# zfs send -I pool/fs@snapA pool/fs@snapD > /snaps/fs@combo
```

  この構文は、`fs@snapA` から `fs@snapD` までのすべての増分スナップショットを `fs@combo` に送信します。

- 元のスナップショットからの増分ストリームを送信してクローンを作成します。増分ストリームを受け入れるには、元のスナップショットが受信側にすでに存在している必要があります。次に例を示します。

```
# zfs send -I pool/fs@snap1 pool/clone@snapA > /snaps/fsclonesnap-I
.
.
# zfs receive -F pool/clone < /snaps/fsclonesnap-I
```

- 指定されたスナップショットまでのすべての下位ファイルシステムの複製ストリームを送信します。受信時には、すべてのプロパティー、スナップショット、下位ファイルシステム、およびクローンが維持されます。次に例を示します。

```
# zfs send -R pool/fs@snap > snaps/fs-R
```

詳細な例については、例 7–1 を参照してください。

- 増分複製ストリームを送信します。次に例を示します。

```
# zfs send -R -[iI] @snapA pool/fs@snapD
```

詳細な例については、例 7–1 を参照してください。

詳細は、238 ページの「複雑な ZFS スナップショットストリームを送信および受信する」を参照してください。

ファイルシステムデータのみに対する ZFS の割り当て制限と予約

Solaris 10 10/08 リリース: このリリースでは、ZFS の割り当て制限と予約の既存機能に加え、データセットの割り当て制限と予約が追加されました。この機能では、スナップショットやクローンなどの子孫はディスク容量の計算に含まれません。

- `refquota` プロパティーは、データセットが消費できるディスク容量に対して強い制限値を設定します。この強い制限値には、スナップショットやクローンなどの下位データで使用されるディスク容量は含まれません。
- `refreservation` プロパティーは、1 つのデータセットに対して保証される最小限のディスク容量を設定します。下位データは含まれません。

たとえば、`studentA` に 10G バイトの `refquota` 制限を設定すると、「基準」ディスク容量として 10G バイトの強い制限値を設定することができます。柔軟性を高めるために、20G バイトの割り当て制限を設定して、`studentA` のスナップショットを管理することもできます。

```
# zfs set refquota=10g tank/studentA
# zfs set quota=20g tank/studentA
```

詳細は、218 ページの「ZFS の割り当て制限と予約を設定する」を参照してください。

ZFS ストレージプールのプロパティー

Solaris 10 10/08 リリース: ZFS ストレージプールのプロパティーは以前のリリースで導入されました。このリリースでは、`cachefile` および `failmode` の 2 つのプロパティーが用意されています。

ここでは、このリリースの新しいストレージプールプロパティーについて説明します。

- cachefile プロパティー – このプロパティーは、プール構成情報がキャッシュされる場所を制御します。システムの起動時に、キャッシュ内のすべてのプールが自動的にインポートされます。ただし、インストール環境とクラスタ化環境では、プールが自動的にインポートされないようにするために、この情報を別の場所にキャッシュすることが必要になる場合もあります。

  プール構成を別の場所にキャッシュするようにこのプロパティーを設定し、あとで zpool import -c コマンドを使用してインポートすることができます。ほとんどの ZFS 構成で、このプロパティーは使用されません。

  cachefile プロパティーは持続性を持たず、ディスクには格納されません。このプロパティーは、プール情報をキャッシュしないように指定するために以前の Solaris リリースで使用されていた temporary プロパティーに代わるものです。

- failmode プロパティー – このプロパティーは、デバイスの接続が失われたことによる壊滅的なプール障害やプールの全デバイスの障害が発生した場合の動作を決定します。failmode プロパティーの値は、wait、continue、または panic に設定できます。デフォルト値は wait です。これは、デバイスを再接続するか障害の発生したデバイスを交換してから、zpool clear コマンドでエラーを解決する必要があることを意味します。

  failmode プロパティーは、ほかの設定可能な ZFS プロパティーと同様に、プールの作成前または作成後に設定することができます。次に例を示します。

```
# zpool set failmode=continue tank
# zpool get failmode tank
NAME  PROPERTY  VALUE     SOURCE
tank  failmode  continue  local

# zpool create -o failmode=continue users mirror c0t1d0 c1t1d0
```

  プールプロパティーについては、表 4–1 を参照してください。

ZFS コマンドの履歴の拡張機能 (zpool history)

Solaris 10 10/08 リリース: zpool history コマンドが拡張され、次の新機能が追加されています。

- ZFS ファイルシステムのイベント情報が表示されるようになりました。次に例を示します。

```
# zpool history
History for 'rpool':
2010-06-23.09:30:12 zpool create -f -o failmode=continue -R /a -m legacy -o
cachefile=/tmp/root/etc/zfs/zpool.cache rpool c1t0d0s0
2010-06-23.09:30:13 zfs set canmount=noauto rpool
2010-06-23.09:30:13 zfs set mountpoint=/rpool rpool
2010-06-23.09:30:13 zfs create -o mountpoint=legacy rpool/ROOT
2010-06-23.09:30:14 zfs create -b 8192 -V 2048m rpool/swap
2010-06-23.09:30:14 zfs create -b 131072 -V 1024m rpool/dump
2010-06-23.09:30:15 zfs create -o canmount=noauto rpool/ROOT/zfsBE
2010-06-23.09:30:16 zpool set bootfs=rpool/ROOT/zfsBE rpool
```

```
2010-06-23.09:30:16 zfs set mountpoint=/ rpool/ROOT/zfsBE
2010-06-23.09:30:16 zfs set canmount=on rpool
2010-06-23.09:30:16 zfs create -o mountpoint=/export rpool/export
2010-06-23.09:30:17 zfs create rpool/export/home
```

- `-l` オプションを使用して、ユーザー名、ホスト名、および操作が実行されたゾーンを含む長形式を表示できます。次に例を示します。

```
# zpool history -l rpool
History for 'tank':
2010-06-24.13:07:58 zpool create tank mirror c2t2d0 c2t5d0 [user root on neo:global]
2010-06-24.13:08:23 zpool scrub tank [user root on neo:global]
2010-06-24.13:38:42 zpool clear tank [user root on neo:global]
2010-06-29.11:44:18 zfs create tank/home [user root on neo:global]
2010-06-29.13:28:51 zpool clear tank c2t5d0 [user root on neo:global]
2010-06-30.14:07:40 zpool add tank spare c2t1d0 [user root on neo:global
```

- `-i` オプションを使用すると、診断目的のために内部イベント情報を表示できます。次に例を示します。

```
# zpool history -i tank
History for 'tank':
2010-06-24.13:07:58 zpool create tank mirror c2t2d0 c2t5d0
2010-06-24.13:08:23 [internal pool scrub txg:6] func=1 mintxg=0 maxtxg=6
2010-06-24.13:08:23 [internal pool create txg:6] pool spa 22; zfs spa 22; zpl 4; uts neo 5.10 Generic_142909-13 s
2010-06-24.13:08:23 [internal pool scrub done txg:6] complete=1
2010-06-24.13:08:23 zpool scrub tank
2010-06-24.13:38:42 zpool clear tank
2010-06-24.13:38:42 [internal pool scrub txg:69] func=1 mintxg=3 maxtxg=8
2010-06-24.13:38:42 [internal pool scrub done txg:69] complete=1
2010-06-29.11:44:18 [internal create txg:14241] dataset = 34
2010-06-29.11:44:18 zfs create tank/home
2010-06-29.13:28:51 zpool clear tank c2t5d0
2010-06-30.14:07:40 zpool add tank spare c2t1d0
```

`zpool history` コマンドの使用方法の詳細については、301 ページの「ZFS の問題を解決する」を参照してください。

ZFS ファイルシステムをアップグレードする (zfs upgrade)

Solaris 10 10/08 リリース: このリリースでは、今後の ZFS ファイルシステムの拡張機能を既存のファイルシステムに提供する `zfs upgrade` コマンドが追加されています。ZFS ストレージプールには、プール拡張を既存のストレージプールに提供する同様のアップグレード機能が備わっています。

次に例を示します。

```
# zfs upgrade
This system is currently running ZFS filesystem version 3.

All filesystems are formatted with the current version.
```

注 – アップグレードされたファイルシステム、およびこれらのアップグレードされたファイルシステムから zfs send コマンドによって作成されたストリームには、古いソフトウェアリリースを実行しているシステムからはアクセスできません。

ZFS 委任管理

Solaris 10 10/08 リリース: このリリースでは、詳細なアクセス権を付与して、権限のないユーザーに ZFS 管理タスクの実行を許可することができます。

zfs allow と zfs unallow の各コマンドを使ってアクセス権を委任および削除できます。

プールの delegation プロパティーを使って委任管理を変更できます。次に例を示します。

```
# zpool get delegation users
NAME  PROPERTY    VALUE       SOURCE
users  delegation  on          default
# zpool set delegation=off users
# zpool get delegation users
NAME  PROPERTY    VALUE       SOURCE
users  delegation  off         local
```

デフォルトでは、delegation プロパティーは有効になっています。

詳細は、第 9 章「Oracle Solaris ZFS 委任管理」および zfs(1M) を参照してください。

別個の ZFS ログデバイスの設定

Solaris 10 10/08 リリース: 同期トランザクションの POSIX 要件を満たすために、ZFS インテントログ (ZIL) が提供されています。たとえば、多くの場合、データベースがシステムコールから戻るときは、そのトランザクションが安定したストレージデバイス上に置かれている必要があります。NFS やその他のアプリケーションでは、データの安定性を確保するために fsync() も使用できます。デフォルトでは、ZIL はメインストレージプール内のブロックから割り当てられます。この Solaris リリースでは、ZIL ブロックが引き続きメインストレージプールから割り当てられるか、それとも別個のログデバイスから割り当てられるかを指定できます。NVRAM や専用ディスクなどで、別個のインテントログデバイスを ZFS ストレージプール内で使用することにより、パフォーマンスを向上できる可能性があります。

ZFS インテントログ用のログデバイスは、データベースのログファイルとは関連がありません。

ZFS ログデバイスの設定は、ストレージプールの作成時または作成後に行えます。ログデバイスの設定の例については、75 ページの「ログデバイスを持つ ZFS ストレージプールを作成する」および82 ページの「ストレージプールにデバイスを追加する」を参照してください。

既存のログデバイスにログデバイスを接続して、ミラー化ログデバイスを作成できます。この操作は、ミラー化されていないストレージプール内にデバイスを接続する操作と同じです。

使用している環境で別個の ZFS ログデバイスを設定することが適切かどうかを判断するには、次の点を考慮してください。

- 別個のログデバイスを実装することによって得られるパフォーマンスの向上は、デバイスの種類、プールのハードウェア構成、およびアプリケーションの作業負荷によって異なります。パフォーマンスの基礎情報については、次のブログを参照してください。

  http://blogs.sun.com/perrin/entry/slog_blog_or_blogging_on
- ログデバイスは複製解除したりミラー化したりできますが、ログデバイスで RAID-Z はサポートされていません。
- 別個のログデバイスがミラー化されていない場合、ログを格納しているデバイスに障害が発生すると、ログブロックの格納はストレージプールに戻ります。
- ログデバイスは、より大規模なストレージプールの一部として、追加、置き換え、接続、切り離し、インポート、およびエクスポートすることができます。ログデバイスの削除は Solaris 10 9/10 リリースから可能になっています。
- ログデバイスの最小サイズは、プール内の各デバイスの最小サイズと同じで、64M バイトです。ログデバイスに格納される可能性のある処理中のデータは比較的少量です。ログのトランザクション (システムコール) がコミットされると、ログブロックは解放されます。
- ログデバイスの最大サイズは物理メモリーのサイズの約 1/2 になるようにしてください。これは、格納される可能性のある処理中のデータの最大量です。たとえば、16G バイトの物理メモリーを備えたシステムの場合、ログデバイスの最大サイズとして 8G バイトを検討してください。

ZFS 中間データセットを作成する

Solaris 10 10/08 リリース: `-zfs create`、`zfs clone`、および `zfs rename` コマンドで `p` オプションを使用すると、中間データセットがまだ存在しない場合にそれをすばやく作成することができます。

次の例では、ZFS データセット (`users/area51`) が `datab` ストレージプールに作成されます。

```
# zfs list
NAME                        USED  AVAIL  REFER  MOUNTPOINT
datab                       106K  16.5G    18K  /datab
# zfs create -p -o compression=on datab/users/area51
```

作成処理中に中間データセットがすでに存在していれば、この処理は正常に完了します。

指定したプロパティーは、中間データセットではなく、ターゲットデータセットに適用されます。次に例を示します。

```
# zfs get mountpoint,compression datab/users/area51
NAME                PROPERTY     VALUE                SOURCE
datab/users/area51  mountpoint   /datab/users/area51  default
datab/users/area51  compression  on                   local
```

中間データセットは、デフォルトのマウントポイントで作成されます。中間データセットに対する追加のプロパティーはすべて無効になります。次に例を示します。

```
# zfs get mountpoint,compression datab/users
NAME         PROPERTY     VALUE         SOURCE
datab/users  mountpoint   /datab/users  default
datab/users  compression  off           default
```

詳細は、zfs(1M) のマニュアルページを参照してください。

ZFS ホットプラグの拡張機能

Solaris 10 10/08 リリース: このリリースの ZFS では、削除されたデバイスへの応答がより効果的になり、挿入されたデバイスを自動的に識別できるようになっています。

- zpool replace コマンドを使用しなくても、既存のデバイスを同等のデバイスに置き換えることができます。

  autoreplace プロパティーは、自動デバイス交換を制御します。オフに設定されている場合、管理者が zpool replace コマンドを使ってデバイス交換を開始する必要があります。オンに設定されている場合、そのプールに以前属していたデバイスと物理的に同じ位置にある新しいデバイスは、いずれも自動的にフォーマットされ、置き換えられます。デフォルトの動作は「オフ」です。
- システムの実行中にデバイスまたはホットスペアが物理的に取り外されると、ストレージプールの状態は REMOVED になります。可能であれば、取り外されたデバイスはホットスペアデバイスで置き換えられます。
- デバイスをいったん取り外してから挿入し直すと、デバイスはオンラインになります。デバイスを挿入し直したときにホットスペアがアクティブになっていた場合は、オンライン処理が完了すると、そのホットスペアが取り外されます。

- デバイスの着脱時の自動検出はハードウェアに依存しているため、すべてのプラットフォームには対応していない可能性があります。たとえば、USB デバイスは挿入時に自動的に構成されます。ただし、`cfgadm -c configure` コマンドを使用して SATA ドライブを構成する必要がある場合もあります。
- ホットスペアは、オンラインおよび使用可能かどうか定期的に確認されます。

詳細は、`zpool(1M)` のマニュアルページを参照してください。

ZFS スナップショットの名前を再帰的に変更する (`zfs rename -r`)

Solaris 10 10/08 リリース: `zfs rename -r` コマンドを使用して、すべての ZFS 子孫スナップショットの名前を再帰的に変更することができます。次に例を示します。

まず、一連の ZFS ファイルシステムのスナップショットが作成されます。

```
# zfs snapshot -r users/home@today
# zfs list
NAME                      USED  AVAIL  REFER  MOUNTPOINT
users                     216K  16.5G    20K  /users
users/home                 76K  16.5G    22K  /users/home
users/home@today            0      -    22K  -
users/home/markm           18K  16.5G    18K  /users/home/markm
users/home/markm@today      0      -    18K  -
users/home/marks           18K  16.5G    18K  /users/home/marks
users/home/marks@today      0      -    18K  -
users/home/neil            18K  16.5G    18K  /users/home/neil
users/home/neil@today       0      -    18K  -
```

その翌日にスナップショットの名前が変更されます。

```
# zfs rename -r users/home@today @yesterday
# zfs list
NAME                         USED  AVAIL  REFER  MOUNTPOINT
users                        216K  16.5G    20K  /users
users/home                    76K  16.5G    22K  /users/home
users/home@yesterday           0      -    22K  -
users/home/markm              18K  16.5G    18K  /users/home/markm
users/home/markm@yesterday     0      -    18K  -
users/home/marks              18K  16.5G    18K  /users/home/marks
users/home/marks@yesterday     0      -    18K  -
users/home/neil               18K  16.5G    18K  /users/home/neil
users/home/neil@yesterday      0      -    18K  -
```

スナップショットは、再帰的な名前変更が可能な唯一の種類のデータセットです。

スナップショットの詳細については、225 ページの「ZFS スナップショットの概要」と、周期的スナップショットの作成方法を記述した次のブログエントリを参照してください。

http://blogs.sun.com/mmusante/entry/rolling_snapshots_made_easy

gzip 圧縮を ZFS に使用できる

Solaris 10 10/08 リリース: この Solaris リリースでは、ZFS ファイルシステムに対して `lzjb` 圧縮だけでなく `gzip` 圧縮も設定できます。圧縮は、`gzip` と指定することも、`gzip-`*N* (*N* は 1 ~ 9) と指定することもできます。次に例を示します。

```
# zfs create -o compression=gzip users/home/snapshots
# zfs get compression users/home/snapshots
NAME                  PROPERTY     VALUE            SOURCE
users/home/snapshots  compression  gzip             local
# zfs create -o compression=gzip-9 users/home/oldfiles
# zfs get compression users/home/oldfiles
NAME                   PROPERTY     VALUE           SOURCE
users/home/oldfiles    compression  gzip-9          local
```

ZFS のプロパティーの設定方法の詳細については、206 ページの「ZFS プロパティーを設定する」を参照してください。

ZFS ユーザーデータの複数のコピーを保存する

Solaris 10 10/08 リリース: 信頼性を高める機能として、可能であれば、ZFS ファイルシステムのメタデータが異なるディスクにまたがって何度か自動的に保存されます。この機能は、「ditto ブロック」として知られています。

この Solaris リリースでは、`zfs set copies` コマンドを使用して、ファイルシステムごとにユーザーデータの複数のコピーを保存することもできます。次に例を示します。

```
# zfs set copies=2 users/home
# zfs get copies users/home
NAME        PROPERTY  VALUE       SOURCE
users/home  copies    2           local
```

使用できる値は 1、2、または 3 です。デフォルト値は 1。これらのコピーは、ミラー化構成または RAID-Z 構成などのプールレベルの冗長性を補うものです。

ZFS ユーザーデータの複数のコピーを保存する利点は次のとおりです。

- すべての ZFS 構成について、メディア障害 (一般に「ビット腐敗」と呼ばれる) などの回復不能なブロックの読み取り障害から回復できるようにすることで、データ保持機能を向上させます。
- 使用できるディスクが 1 台だけの場合でもデータ保護が提供されます。
- ストレージプールの機能を超えて、ファイルシステムごとにデータ保護ポリシーを選択できます。

注 – ストレージプールでの ditto ブロックの割り当てによっては、複数のコピーが単一のディスクに保存される場合もあります。そのあとでディスク全体の障害が発生すると、すべての ditto ブロックが使用不可になる可能性があります。

誤って非冗長プールを作成した場合や、データ保持ポリシーを設定する必要がある場合は、ditto ブロックの使用を検討することもできます。

単一ディスクのプールまたは複数ディスクのプールを備えたシステムで複数のコピーを保存することにより、全体的なデータ保護がどのように影響を受けるかについて詳しくは、次のブログを参照してください。

http://blogs.sun.com/relling/entry/zfs_copies_and_data_protection

ZFS のプロパティーの設定方法の詳細については、206 ページの「ZFS プロパティーを設定する」を参照してください。

改善された zpool status の出力

Solaris 10 8/07 リリース: `zpool status -v` コマンドを使用すると、永続的なエラーが発生しているファイルの一覧を表示できます。以前は、`find -inum` コマンドを使用して、表示された i ノードの一覧からファイル名を特定する必要がありました。

永続的なエラーが発生しているファイル一覧の表示に関する詳細は、320 ページの「破壊されたファイルまたはディレクトリを修復する」を参照してください。

ZFS および Solaris iSCSI の向上

Solaris 10 8/07 リリース: この Solaris リリースでは、ZFS ボリュームに `shareiscsi` プロパティーを設定することで、ZFS ボリュームを Solaris iSCSI ターゲットデバイスとして作成できます。この方法は、Solaris iSCSI ターゲットをすばやく設定するのに便利です。次に例を示します。

```
# zfs create -V 2g tank/volumes/v2
# zfs set shareiscsi=on tank/volumes/v2
# iscsitadm list target
Target: tank/volumes/v2
    iSCSI Name: iqn.1986-03.com.sun:02:984fe301-c412-ccc1-cc80-cf9a72aa062a
    Connections: 0
```

iSCSI ターゲットが作成されたら、iSCSI イニシエータを設定できます。Solaris iSCSI イニシエータの設定方法については、『Solaris のシステム管理 (デバイスとファイルシステム)』の第 14 章「Oracle Solaris iSCSI ターゲットおよびイニシエータの構成 (手順)」を参照してください。

ZFS ボリュームを iSCSI ターゲットとして管理するための詳細については、287 ページの「ZFS ボリュームを Solaris iSCSI ターゲットとして使用する」を参照してください。

ZFS コマンドの履歴 (zpool history)

Solaris 10 8/07 リリース: この Solaris リリースでは、正常に実行された `zfs` および `zpool` コマンドが ZFS によって自動的に記録され、プールの状態の情報が更新されます。次に例を示します。

```
# zpool history
History for 'newpool':
2007-04-25.11:37:31 zpool create newpool mirror c0t8d0 c0t10d0
2007-04-25.11:37:46 zpool replace newpool c0t10d0 c0t9d0
2007-04-25.11:38:04 zpool attach newpool c0t9d0 c0t11d0
2007-04-25.11:38:09 zfs create newpool/user1
2007-04-25.11:38:15 zfs destroy newpool/user1

History for 'tank':
2007-04-25.11:46:28 zpool create tank mirror c1t0d0 c2t0d0 mirror c3t0d0 c4t0d0
```

この機能によって、ユーザーや Oracle のサポート担当者は実行された「正確な」ZFS コマンドを特定し、エラーシナリオのトラブルシューティングを行うことができます。

`zpool history` コマンドでは、特定のストレージプールを指定できます。次に例を示します。

```
# zpool history newpool
History for 'newpool':
2007-04-25.11:37:31 zpool create newpool mirror c0t8d0 c0t10d0
2007-04-25.11:37:46 zpool replace newpool c0t10d0 c0t9d0
2007-04-25.11:38:04 zpool attach newpool c0t9d0 c0t11d0
2007-04-25.11:38:09 zfs create newpool/user1
2007-04-25.11:38:15 zfs destroy newpool/user1
```

この Solaris リリースでは、`zpool history` コマンドで、ユーザー ID (*user-ID*)、ホスト名 (*hostname*)、またはゾーン名 (*zone-name*) は記録されません。ただし、この情報が記録されるのは Solaris 10 10/08 リリースからです。詳細は、32 ページの「ZFS コマンドの履歴の拡張機能 (zpool history)」を参照してください。

ZFS に関する問題のトラブルシューティングの詳細については、301 ページの「ZFS の問題を解決する」を参照してください。

ZFS プロパティーの改善

ZFS xattr プロパティー

Solaris 10 8/07 リリース: xattr プロパティーを使用すると、特定の ZFS ファイルシステムの拡張属性を無効または有効にできます。デフォルト値は on です。ZFS プロパティーについては、189 ページの「ZFS のプロパティーの紹介」を参照してください。

ZFS canmount プロパティー

Solaris 10 8/07 リリース: 新しいプロパティーである canmount を使用すると、zfs mount コマンドを使ってデータセットをマウントできるかどうかを指定できます。詳細については、201 ページの「canmount プロパティー」を参照してください。

ZFS ユーザープロパティー

Solaris 10 8/07 リリース: ZFS では、内部統計情報のエクスポートや ZFS ファイルシステムの動作の制御に使用できる標準のネイティブプロパティーのほかに、ユーザープロパティーも用意されています。ユーザープロパティーは ZFS の動作には影響しませんが、これらを使用すると、使用環境内で意味のある情報をデータセットに注釈として付けることができます。

詳細については、203 ページの「ZFS ユーザープロパティー」を参照してください。

ZFS ファイルシステムの作成時にプロパティーを設定する

Solaris 10 8/07 リリース: この Solaris リリースでは、ファイルシステムの作成後だけでなく作成時にプロパティーを設定できます。

次の 2 つの例は、同等の構文を示しています。

```
# zfs create tank/home
# zfs set mountpoint=/export/zfs tank/home
# zfs set sharenfs=on tank/home
# zfs set compression=on tank/home

# zfs create -o mountpoint=/export/zfs -o sharenfs=on -o compression=on tank/home
```

すべての ZFS ファイルシステムの情報を表示する

Solaris 10 8/07 リリース: この Solaris リリースでは、データセットを指定しないかまたは all を指定する場合、さまざまな形式の zfs get コマンドを使用してすべてのデータセットに関する情報を表示できます。以前のリリースでは、zfs get コマンドですべてのデータセットに関する情報を取得することはできませんでした。

次に例を示します。

```
# zfs get -s local all
tank/home               atime          off                    local
tank/home/bonwick       atime          off                    local
tank/home/marks         quota          50G                    local
```

新しい zfs receive -F オプション

Solaris 10 8/07 リリース: この Solaris リリースでは、-zfs receive コマンドに新しい F オプションを指定することで、受信が開始する前に、最新のスナップショットへのファイルシステムのロールバックを強制することができます。このオプションの使用は、ロールバックが発生してから受信が開始するまでの間にファイルシステムが変更されるときに必要となる場合があります。

詳細は、237 ページの「ZFS スナップショットを受信する」を参照してください。

再帰的な ZFS スナップショット

Solaris 10 11/06 リリース: zfs snapshot コマンドを使ってファイルシステムのスナップショットを作成する場合、-r オプションを使用すると、すべての子孫ファイルシステムのスナップショットを再帰的に作成できます。また、-r オプションを使用すると、スナップショットが破棄されたときにすべての子孫スナップショットを再帰的に破棄できます。

再帰的な ZFS スナップショットは、1 つの原子動作としてすばやく作成されます。スナップショットは、まとめて (一度にすべて) 作成されるか、まったく作成されないかのどちらかです。そのような操作の利点は、子孫ファイルシステムにまたがる場合でも、常にある一貫した時間のスナップショットデータが取得されることです。

詳細は、226 ページの「ZFS スナップショットを作成および破棄する」を参照してください。

ダブルパリティー RAID-Z (raidz2)

Solaris 10 11/06 リリース: 冗長な RAID-Z 構成でシングルパリティー構成またはダブルパリティー構成を使用できるようになりました。これはそれぞれ、1 つまたは 2 つのデバイスで障害が発生しても、データを失うことなく処理を続行できることを意味します。raidz2 キーワードを指定すれば、ダブルパリティー RAID-Z 構成にすることができます。あるいは、raidz または raidz1 キーワードを指定すれば、シングルパリティー RAID-Z 構成にすることができます。

詳細は、74 ページの「RAID-Z ストレージプールを作成する」または zpool(1M) のマニュアルページを参照してください。

ZFS ストレージプールのデバイスのホットスペア

Solaris 10 11/06 リリース: ZFS ホットスペア機能を使用すると、1 つ以上のストレージプールで障害が発生したデバイスまたはエラー状態のデバイスの置き換えに使用可能なディスクを特定できます。デバイスを「ホットスペア」として指定しておくと、プールのアクティブデバイスで障害が発生した場合に、そのデバイスがホットスペアに自動的に置き換えられます。ストレージプールのデバイスを手動でホットスペアに置き換えることもできます。

詳細は、98 ページの「ストレージプールにホットスペアを指定する」および zpool(1M) のマニュアルページを参照してください。

ZFS ファイルシステムを ZFS クローンに置き換える (zfs promote)

Solaris 10 11/06 リリース: `zfs promote` コマンドを使用すると、ある既存の ZFS ファイルシステムをそのファイルシステムのクローンで置き換えることができます。この機能は、ファイルシステムの代替バージョンでテストを実行してから、その代替バージョンをアクティブファイルシステムに置き換えるときに利用できます。

詳細は、233 ページの「ZFS ファイルシステムを ZFS クローンで置き換える」および zfs(1M) のマニュアルページを参照してください。

ZFS ストレージプールをアップグレードする (zpool upgrade)

Solaris 10 6/06 リリース: `zpool upgrade` コマンドを使用すると、ストレージプールを新しいバージョンの ZFS にアップグレードして、その最新機能を利用できるようになります。また、古いバージョンの ZFS を実行している場合、`zpool status` コマンドによって通知されます。

詳細は、123 ページの「ZFS ストレージプールをアップグレードする」および zpool(1M) のマニュアルページを参照してください。

以前の Solaris リリースのプールを持つシステムで ZFS 管理コンソールを使用する場合は、必ずプールをアップグレードしてからコンソールを使用するようにしてください。プールのアップグレードが必要かどうかを調べるには、`zpool status` コマンドを使用します。ZFS 管理コンソールについては、45 ページの「Web ベースの ZFS 管理」を参照してください。

ZFS のバックアップコマンドと復元コマンドの名前が変更されている

Solaris 10 6/06 リリース: この Solaris リリースでは、`zfs backup` および `zfs restore` コマンドの名前が、それらの機能をより正確に表すように、`zfs send` および `zfs receive` にそれぞれ変更されています。これらのコマンドは ZFS データストリーム表現を送受信します。

これらのコマンドの詳細については、234 ページの「ZFS データを送信および受信する」を参照してください。

破棄されたストレージプールの回復

Solaris 10 6/06 リリース: このリリースには、`zpool import -D` コマンドが含まれています。このコマンドを使用すると、以前に `zpool destroy` コマンドで破棄されたプールを回復できます。

詳細は、121 ページの「破棄された ZFS ストレージプールを回復する」を参照してください。

ZFS が Fault Manager と統合されている

Solaris 10 6/06 リリース**:** このリリースには、プールの障害やデバイスの障害を診断および報告できる ZFS 診断エンジンが含まれています。プールまたはデバイスの障害に関連するチェックサム、入出力、デバイス、およびプールのエラーも報告されます。

この診断エンジンでは、チェックサムと入出力のエラーを予測分析する機能や、障害分析に基づいて予防処理を行う機能はありません。

ZFS で障害が発生した場合、次のようなメッセージが表示されることがあります。

```
SUNW-MSG-ID: ZFS-8000-D3, TYPE: Fault, VER: 1, SEVERITY: Major
EVENT-TIME: Wed Jun 30 14:53:39 MDT 2010
PLATFORM: SUNW,Sun-Fire-880, CSN: -, HOSTNAME: neo
SOURCE: zfs-diagnosis, REV: 1.0
EVENT-ID: 504a1188-b270-4ab0-af4e-8a77680576b8
DESC: A ZFS device failed.  Refer to http://sun.com/msg/ZFS-8000-D3 for more information.
AUTO-RESPONSE: No automated response will occur.
IMPACT: Fault tolerance of the pool may be compromised.
REC-ACTION: Run 'zpool status -x' and replace the bad device.
```

推奨される操作として、`zpool status` コマンドの指示に従って再調査することにより、障害をすばやく特定して解決することができます。

報告された ZFS の問題から回復する例については、307 ページの「見つからないデバイスに関する問題を解決する」を参照してください。

zpool clear コマンド

Solaris 10 6/06 リリース: このリリースには、デバイスやプールに関連するエラーカウントをクリアーするための `zpool clear` コマンドが含まれています。以前は、`zpool online` コマンドを使ってプール内のデバイスがオンラインになったときに、エラー数がクリアーされていました。詳細は、95 ページの「ストレージプールデバイスのエラーをクリアーする」および `zpool(1M)` のマニュアルページを参照してください。

NFSv4 ACL コンパクト形式

Solaris 10 6/06 リリース: このリリースでは、冗長およびコンパクトの 2 とおりの形式で NFSv4 ACL を設定および表示できます。どちらの ACL 形式も `chmod` コマンドを使って設定できます。コンパクト形式の ACL は、`ls -V` コマンドを使って表示できます。冗長形式の ACL は、`ls -v` コマンドを使って表示できます。

詳細は、265 ページの「ZFS ファイルの ACL をコンパクト形式で設定および表示する」、および `chmod(1)` と `ls(1)` のマニュアルページを参照してください。

ファイルシステム監視ツール (fsstat)

Solaris 10 6/06 リリース: 新しいファイルシステム監視ツール `fsstat` はファイルシステムの動作を報告します。アクティビティーは、マウントポイント単位またはファイルシステムタイプ単位で報告できます。一般的な ZFS ファイルシステムアクティビティーの例を示します。

```
$ fsstat zfs
 new  name   name  attr  attr lookup rddir  read read  write write
 file remov  chng   get   set    ops   ops   ops bytes   ops bytes
7.82M 5.92M 2.76M 1.02G 3.32M  5.60G 87.0M  363M 1.86T 20.9M  251G zfs
```

詳細は、`fsstat(1M)` のマニュアルページを参照してください。

Web ベースの ZFS 管理

Solaris 10 6/06 リリース: Web ベースの ZFS 管理ツールである ZFS 管理コンソールを使用して、次の管理タスクを実行できます。

- 新規ストレージプールを作成する。
- 既存のプールに容量を追加する。
- 既存のプールを別のシステムに移動 (エクスポート) する。
- 以前にエクスポートしたストレージプールを別のシステムで使用できるようにインポートする。

- ストレージプールに関する情報を表示する。
- ファイルシステムを作成する。
- ボリュームを作成する。
- ファイルシステムまたはボリュームのスナップショットを作成する。
- ファイルシステムを以前のスナップショットにロールバックする。

セキュリティー保護された Web ブラウザから次の URL の ZFS 管理コンソールにアクセスできます。

```
https://system-name:6789/zfs
```

正しい URL を入力しても ZFS 管理コンソールにアクセスできない場合は、サーバーが起動していない可能性があります。サーバーを起動するには、次のコマンドを実行します。

```
# /usr/sbin/smcwebserver start
```

システムをブートするときにサーバーを自動的に実行するには、次のコマンドを実行します。

```
# /usr/sbin/smcwebserver enable
```

注 – Solaris 管理コンソール (smc) を使って ZFS ストレージプールまたは ZFS ファイルシステムを管理することはできません。

ZFS の概要

ZFS ファイルシステムは、ファイルシステムの管理方法を根底から変える革命的な新しいファイルシステムであり、現在利用できるほかのファイルシステムにはない機能と特長を備えています。ZFS は堅牢で、スケーラブルで、管理が容易です。

プールされた ZFS ストレージ

ZFS では、物理ストレージを管理するために、「ストレージプール」という概念を使用します。従来のファイルシステムは、1 つの物理デバイス上に構築されていました。複数のデバイスへの対応とデータの冗長性を実現するために、「ボリュームマネージャー」の概念が導入されました。ファイルシステムを変更しなくても複数のデバイスが利用できるように、1 つのデバイスの表現を使用しています。このファイルシステムの設計は、仮想化したボリューム上の物理的なデータの配置を制御する手段を用意していないため、ファイルシステムをより複雑にし、ある面でのファイルシステムの進化を阻んできました。

ZFS では、ボリューム管理は一切不要です。ZFS では、仮想化されたボリュームを作成する代わりに、デバイスをストレージプールに集約します。ストレージプールは、ストレージの物理特性 (デバイスのレイアウト、データの冗長性など) を記述したもので、ファイルシステムを作成できる任意のデータストアとして機能します。ファイルシステムが個々のデバイスに制約されなくなり、デバイスのディスク領域をプール内のすべてのファイルシステムで共有することができます。ファイルシステムのサイズを事前に決定する必要はなくなりました。ファイルシステムのサイズは、ストレージプールに割り当てられたディスク領域内で自動的に拡張します。新しいストレージを追加すると、何も操作しなくても、プール内のすべてのファイルシステムで追加したディスク領域をすぐに使用できます。ストレージプールは多くの点で、仮想メモリーシステムと同様に機能します。DIMM メモリーをシステムに追加するとき、メモリーを構成するコマンドを実行したり、個別のプロセスにメモリーを割り当てたりすることをオペレーティングシステムによって強制されるわけではありません。追加したメモリーは、システムのすべてのプロセスによって自動的に使用されます。

トランザクションのセマンティクス

ZFS はトランザクションファイルシステムです。つまり、ファイルシステムの状態がディスク上で常に一定であることを意味します。従来のファイルシステムは、データをその場所で上書きします。このため、たとえば、データブロックが割り当てられてからディレクトリにリンクされるまでの間にシステムの電源が切断された場合、ファイルシステムは不整合な状態のままになります。従来、この問題は `fsck` コマンドを使用して解決されていました。このコマンドの機能は、ファイルシステムの状態を確認および検証し、処理中に不整合が見つかった場合はその修復を試みることでした。このようなファイルシステムの不整合の問題は管理者を大いに苦労させ、`fsck` コマンドを使ってもすべての問題が修正されるとは限りませんでした。最近では、ファイルシステムに「ジャーナリング」の概念が導入されました。ジャーナリングプロセスでは各処理がそれぞれのジャーナルに記録されるため、システムのクラッシュが発生したときに処理を安全に「再現」できます。このプロセスでは、不必要な負荷が発生します。これはデータを 2 回書き込む必要があるためで、多くの場合、ジャーナルを正しく再現できないなどの新しい問題が発生します。

トランザクションファイルシステムでは、データは「コピーオンライト」セマンティクスを使用して管理されます。データが上書きされることはなく、一覧の処理が完全に確定されるか、完全に無視されます。そのため、電源が突然切断されたりシステムがクラッシュしても、ファイルシステムが破壊されることはありません。直近に書き込まれたデータが失われることがあっても、ファイルシステム自体の整合性は常に保持されます。また、`O_DSYNC` フラグを使用して書き込まれる同期データは、書き込まれてから戻ることが常に保証されているため、失われることがありません。

チェックサムと自己修復データ

ZFS では、すべてのデータおよびメタデータが、ユーザーが選択したチェックサムアルゴリズムを使って検証されます。チェックサム検証の機能を持つ従来のファイルシステムでは、ボリューム管理層および従来のファイルシステムの設計のために、強制的にブロック単位でチェックサムが計算されていました。従来の設計では、完全なブロックを誤った位置に書き込むといった特定の障害の結果として、データは不正だがチェックサムエラーにならないという状況が生じる可能性があります。ZFS のチェックサムはそのような障害を検出し、障害モードから正常に回復できるような方法で格納されます。すべてのチェックサム検証とデータの回復は、ファイルシステム層でアプリケーションに透過的に実行されます。

また、ZFS は自己修復データも備えています。ZFS は、さまざまなレベルのデータ冗長性を備えたストレージプールをサポートします。不正なデータブロックが検出されると、ZFS は別の冗長コピーから正しいデータを取得し、不正なデータを正しいデータで置き換えて修復します。

優れたスケーラビリティー

ZFS ファイルシステムの重要な設計要素はスケーラビリティーです。ファイルシステム自体は 128 ビットで、25 京 6000 兆 (256 クアデリリオン) ゼタバイトの記憶域に対応しています。すべてのメタデータは動的に割り当てられるため、i ノードを事前に割り当てたり、初回作成時にファイルシステムのスケーラビリティーを制限する必要はありません。すべてのアルゴリズムは、スケーラビリティーを考慮して記述されています。ディレクトリには、2^{48} (256 兆) のエントリを格納することができ、ファイルシステムの数およびファイルシステムに格納できるファイル数の制限はありません。

ZFS スナップショット

「スナップショット」とは、ファイルシステムまたはボリュームの読み取り専用コピーのことです。スナップショットは、短時間で簡単に作成できます。最初のスナップショットのために、プール内のディスク領域が余分に消費されることはありません。

有効なデータセット内のデータが変更されると、スナップショットは古いデータを参照し続けるためのディスク領域を消費します。その場合、スナップショットのため、古いデータの領域は解放されずプールに戻されません。

簡素化された管理

ZFS の重要な特長として、管理モデルが大幅に簡素化されていることが挙げられます。ZFS では、階層構造のファイルシステムレイアウト、プロパティーの継承、およびマウントポイントと NFS 共有セマンティクスの自動管理により、複数のコマンドを使用したり、構成ファイルを編集したりしなくても、ファイルシステムを簡単に作成できます。割り当て制限や予約を設定したり、圧縮の有効/無効を切り替えたり、大量のファイルシステムのマウントポイントを管理したりする操作を、1 つのコマンドだけで簡単に実行することができます。デバイスの検査や交換のために、一連のボリュームマネージャーコマンドを別途学習する必要はありません。ファイルシステムスナップショットのストリームを送受信できます。

ZFS では、ファイルシステムを階層構造で管理するので、割り当て制限、予約、圧縮、マウントポイントなどのプロパティーを簡単に管理できます。この管理モデルでは、ファイルシステムが重要な役割を果たします。ファイルシステム自体は新しいディレクトリの作成と同じようにとても簡単に操作できるので、ユーザー、プロジェクト、ワークスペースなどのために個別のファイルシステムを作成することをお勧めします。この設計を利用して、きめの細かい管理ポイントを定義できます。

ZFS の用語

ここでは、このマニュアルで使用される基本的な用語について説明します。

代替ブート環境
: `lucreate` コマンドによって作成され、場合によっては `luupgrade` コマンドで更新されているが、アクティブなブート環境または主ブート環境ではないブート環境。`luactivate` コマンドを実行することにより、代替ブート環境を主ブート環境にすることができます。

チェックサム
: ファイルシステムブロック内の 256 ビットのハッシュデータ。チェックサム機能には、単純で高速な fletcher4 (デフォルト) から SHA256 などの暗号強度の高いハッシュまで、さまざまなものがあります。

クローン
: 初期コンテンツがスナップショットの内容と同じであるファイルシステム。

  クローンの詳細については、232 ページの「ZFS クローンの概要」を参照してください。

dataset
: 次の ZFS コンポーネントの総称名。クローン、ファイルシステム、スナップショット、およびボリューム。

  各データセットは、ZFS 名前空間内で一意の名前で識別されます。データセットは、次の形式を使用して識別されます。

pool/*path*[@*snapshot*]

pool データセットを格納するストレージプールの名前

path データセットコンポーネントのスラッシュ区切りのパス名

snapshot データセットのスナップショットを識別するオプションコンポーネント

データセットの詳細については、第 6 章「Oracle Solaris ZFS ファイルシステムの管理」を参照してください。

ファイルシステム　標準のシステム名前空間内にマウントされ、別のファイルシステムのように動作する、`filesystem` タイプの ZFS データセット。

ファイルシステムの詳細については、第 6 章「Oracle Solaris ZFS ファイルシステムの管理」を参照してください。

ミラー　複数のディスク上にデータの同一コピーを格納する仮想デバイス。ミラー上のいずれかのディスクで障害が発生した場合には、ミラー上の別のディスクにある同じデータを利用できます。

pool　デバイスの論理グループ。使用可能なストレージのレイアウトおよび物理特性を記述します。データセットのディスク領域は、プールから割り当てられます。

ストレージプールの詳細については、第 4 章「Oracle Solaris ZFS ストレージプールの管理」を参照してください。

主ブート環境　`lucreate` コマンドによって代替ブート環境の構築に使用されるブート環境。デフォルトでは、主ブート環境は現在のブート環境です。このデフォルトは、`lucreate -s` オプションを使用して無効にすることができます。

RAID-Z　データとパリティーを複数のディスクに格納する仮想デバイス。RAID-Z の詳細については、69 ページの「RAID-Z ストレージプール構成」を参照してください。

再同期化　あるデバイスのデータを別のデバイスにコピーする処理のことを「再同期化」と言います。たとえば、ミラーデバイスが置き換えられてオフラインになっている場合には、最新のミラーデバイスのデータが新しく復元されたミラーデバイスにコピーされます。この処理は、従来のボリューム管理製品では「ミラー再同期化」と呼ばれています。

	ZFS の再同期化の詳細については、317 ページの「再同期化の状態を表示する」を参照してください。
snapshot	特定の時点における ファイルシステムまたはボリュームの読み取り専用コピー。 スナップショットの詳細については、225 ページの「ZFS スナップショットの概要」を参照してください。
仮想デバイス	プール内の論理デバイス。物理デバイス、ファイル、または一連のデバイスを仮想デバイスに設定できます。 仮想デバイスの詳細については、77 ページの「ストレージプールの仮想デバイスの情報を表示する」を参照してください。
ボリューム	ブロックデバイスを表すデータセット。たとえば、スワップデバイスとして ZFS ボリュームを作成できます。 ZFS ボリュームの詳細については、285 ページの「ZFS ボリューム」を参照してください。

ZFS コンポーネントに名前を付けるときの規則

データセットやプールなどの各 ZFS コンポーネントには、次の規則に従って名前を付ける必要があります。

- 各コンポーネントに使用できる文字は、英数字および次の 4 つの特殊文字だけです。
  - 下線 (_)
  - ハイフン (-)
  - コロン (:)
  - ピリオド (.)
- プール名の先頭は英字にする必要があります。ただし、次の制限事項があります。
  - `c[0-9]` の順序で始まる名前は許可されません。
  - `log` という名前は予約されています。
  - `mirror`、`raidz`、`raidz1`、`raidz2`、`raidz3`、または `spare` で始まる名前は許可されていません。これらの名前は予約されています。
  - プール名にはパーセント記号 (%) を含めないでください。
- データセット名の先頭は英数字にする必要があります。
- データセット名にはパーセント記号 (%) を含めないでください。

また、コンポーネント名を空にすることはできません。

◆ ◆ ◆ 第 2 章

Oracle Solaris ZFS 入門

この章では、基本的な Oracle Solaris ZFS 構成の設定に関する詳しい手順を提供します。この章を読み終わると、ZFS コマンドの機能の基本を理解し、基本的なプールとファイルシステムを作成できるようになります。この章では、ZFS の包括的な概要を提供しません。また、ZFS の詳細については、以降の章を参照してください。

この章は、次の節で構成されます。

ZFS のハードウェアとソフトウェアに関する要件および推奨要件

ZFS ソフトウェアを使用する前に、次に示すハードウェアとソフトウェアの要件および推奨要件を確認してください。

- Solaris 10 6/06 以降のリリースが稼働している SPARC または x86 システムを使用してください。
- ストレージプールに必要な最小ディスク容量は、64M バイトです。最小ディスクサイズは 128M バイトです。
- Solaris システムのインストールに必要な最小メモリー容量は、768M バイトです。ただし、良好な ZFS パフォーマンスを得るには、1G バイト以上のメモリーを使用してください。
- ミラー化ディスク構成を作成する場合は複数のコントローラを使用してください。

基本的な ZFS ファイルシステムを作成する

ZFS は、簡単に管理できるように設計されています。より少ないコマンドで有効なファイルシステムを作成できるようにすることが、設計目標の 1 つになっています。たとえば、新しいプールを作成すると、新しい ZFS ファイルシステムが自動的に作成されてマウントされます。

次の例は、tank という名前の基本的なミラー化ストレージプールと tank という名前の ZFS ファイルシステムを、1 つのコマンドで作成する方法を示しています。/dev/dsk/c1t0d0 ディスク全体と /dev/dsk/c2t0d0 ディスク全体を使用することを前提としています。

```
# zpool create tank mirror c1t0d0 c2t0d0
```

冗長な ZFS プール構成の詳細については、69 ページの「ZFS ストレージプールの複製機能」を参照してください。

この新規 ZFS ファイルシステム tank は、使用可能なディスク領域を必要に応じて使用でき、/tank に自動的にマウントされます。

```
# mkfile 100m /tank/foo
# df -h /tank
Filesystem             size   used  avail capacity  Mounted on
tank                   80G   100M    80G     1%    /tank
```

プールの中に、さらに別のファイルシステムを作成することをお勧めします。ファイルシステムに基づいてプールを管理すれば、プールに含まれるさまざまなデータを管理しやすくなります。

次の例は、ストレージプール tank に fs という名前のファイルシステムを作成する方法を示しています。

```
# zfs create tank/fs
```

この新規 ZFS ファイルシステム tank/fs は、使用可能なディスク領域を必要に応じて使用でき、/tank/fs に自動的にマウントされます。

```
# mkfile 100m /tank/fs/foo
# df -h /tank/fs
Filesystem             size   used  avail capacity  Mounted on
tank/fs                 80G   100M    80G     1%    /tank/fs
```

通常、ファイルシステムの階層を作成して編成するときには、組織の要件に一致させることをお勧めします。ZFS ファイルシステム階層の作成方法については、56 ページの「ZFS ファイルシステム階層を作成する」を参照してください。

ZFS ストレージプールを作成する

前述の例では、ZFS の操作が簡単であることを示しました。この章の残りの部分では、実際の環境に近い、より詳細な例を提供します。最初に、ストレージ要件を確認して、ストレージプールを作成します。プールによって、ストレージの物理的な特性が決まります。どのようなファイルシステムを作成する場合にも、最初にプールを作成する必要があります。

▼ ZFS ストレージプールのストレージ要件を確認する方法

1 ストレージプールに使用可能なデバイスを決定します。

ストレージプールを作成する前に、データを格納するデバイスを決定する必要があります。デバイスのサイズは、128M バイト以上にしてください。オペレーティングシステムのほかの部分で使われていてはいけません。事前にフォーマットされているディスク上のスライスを個別に選択するか、1 つの大きなスライスとしてフォーマットされたディスク全体を選択することができます。

55 ページの「ZFS ストレージプールを作成する方法」のストレージ例では、ディスク `/dev/dsk/c1t0d0` および `/dev/dsk/c2t0d0` の全体が使用されることを前提としています。

ディスクの詳細、および使用方法と名前の付け方については、65 ページの「ZFS ストレージプール内でディスクを使用する」を参照してください。

2 データの複製を選択します。

ZFS では、複数の種類のデータ複製がサポートされます。プールが対応できるハードウェア障害の種類は、データ複製の種類によって決まります。ZFS では、非冗長 (ストライプ) 構成、ミラー構成、および RAID-Z (RAID-5 の一種) がサポートされます。

55 ページの「ZFS ストレージプールを作成する方法」のストレージ例では、使用可能な 2 台のディスクを基本的な方法でミラー化しています。

ZFS の複製機能の詳細については、69 ページの「ZFS ストレージプールの複製機能」を参照してください。

▼ ZFS ストレージプールを作成する方法

1 **root** になるか、適切な **ZFS** 権利プロファイルが割り当てられた **root** と同等の役割を引き受けます。

ZFS 権利プロファイルの詳細については、295 ページの「ZFS 権利プロファイル」を参照してください。

2 ストレージプールの名前を選択します。

この名前は、zpool または zfs コマンドの使用時にストレージプールを識別するために使用されます。ほとんどのシステムに必要なプールは 1 つだけです。任意の名前を選択できますが、51 ページの「ZFS コンポーネントに名前を付けるときの規則」の名前の付け方の規則に準拠している必要があります。

3 プールを作成します。

たとえば次のコマンドは、tank という名前のミラープールを作成します。

```
# zpool create tank mirror c1t0d0 c2t0d0
```

1 つ以上のデバイスに別のファイルシステムが含まれる場合、または 1 つ以上のデバイスが使用中である場合は、このコマンドを使ってプールを作成することはできません。

ストレージプールの作成方法の詳細については、72 ページの「ZFS ストレージプールを作成する」を参照してください。デバイスの使用状況を確認する方法の詳細については、78 ページの「使用中のデバイスを検出する」を参照してください。

4 結果を確認します。

プールが正常に作成されたかどうかは、zpool list コマンドを使って確認できます。

```
# zpool list
NAME                    SIZE    ALLOC   FREE    CAP  HEALTH     ALTROOT
tank                     80G    137K     80G     0%  ONLINE     -
```

プールの状態を確認する方法の詳細については、107 ページの「ZFS ストレージプールの状態のクエリー検索を行う」を参照してください。

ZFS ファイルシステム階層を作成する

ストレージプールを作成してデータを格納したあとで、ファイルシステムの階層を作成できます。階層を利用すれば、情報を簡単かつ強力に編成することができます。ファイルシステムを使用したことがあれば、階層も同じように操作できます。

ZFS では、ファイルシステムを階層に編成できます。各ファイルシステムの親は 1 つだけです。階層のルートは常にプールの名前です。ZFS では、この階層を利用してプロパティーを継承することができるので、ファイルシステムのツリー全体に共通のプロパティーをすばやく簡単に設定できます。

▼ ZFS ファイルシステム階層を決定する方法

1 ファイルシステムの構造を選択します。

ZFS の管理は、ファイルシステムに基づいて行います。このファイルシステムは、軽量で、簡単に作成できます。ファイルシステムは、ユーザーまたはプロジェクトご

とに作成することをお勧めします。このモデルを使えば、プロパティー、スナップショット、およびバックアップをユーザーまたはプロジェクト単位で制御することができます。

57 ページの「ZFS ファイルシステムを作成する方法」で、2 つの ZFS ファイルシステム `bonwick` および `billm` が作成されます。

ファイルシステムの管理方法の詳細については、第 6 章「Oracle Solaris ZFS ファイルシステムの管理」を参照してください。

2 属性が似ているファイルシステムをグループにまとめます。

ZFS では、ファイルシステムを階層に編成できます。そのため、属性が似ているファイルシステムをグループにまとめることがきます。このモデルを利用すれば、プロパティーの制御やファイルシステムの管理を一箇所で行うことができます。属性が似ているファイルシステムは、共通の名前の階層下に作成するようにしてください。

57 ページの「ZFS ファイルシステムを作成する方法」の例では、2 つのファイルシステムが `home` という名前のファイルシステムの下に配置されます。

3 ファイルシステムのプロパティーを選択します。

ファイルシステムの特性のほとんどは、プロパティーによって制御されます。ファイルシステムがマウントされる場所、共有される方法、圧縮を使用するかどうか、割り当て制限が有効かどうかなど、さまざまな動作がこれらのプロパティーによって制御されます。

57 ページの「ZFS ファイルシステムを作成する方法」の例では、すべてのホームディレクトリが `/export/zfs/` *user* にマウントされ、NFS を使って共有され、圧縮が有効になっています。さらに、ユーザー `bonwick` には 10G バイトの割り当て制限が適用されます。

プロパティーの詳細については、189 ページの「ZFS のプロパティーの紹介」を参照してください。

▼ ZFS ファイルシステムを作成する方法

1 **root** になるか、適切な **ZFS** 権利プロファイルが割り当てられた **root** と同等の役割を引き受けます。

ZFS 権利プロファイルの詳細については、295 ページの「ZFS 権利プロファイル」を参照してください。

2 必要な階層を作成します。

この例では、各ファイルシステムのコンテナとして機能するファイルシステムが作成されます。

```
# zfs create tank/home
```

3 継承されるプロパティーを設定します。

ファイルシステムの階層が確立されたら、すべてのユーザーの間で共有すべきプロパティーをすべて設定します。

```
# zfs set mountpoint=/export/zfs tank/home
# zfs set sharenfs=on tank/home
# zfs set compression=on tank/home
# zfs get compression tank/home
NAME             PROPERTY       VALUE                      SOURCE
tank/home        compression    on                         local
```

ファイルシステムの作成時にファイルシステムのプロパティーを設定できます。次に例を示します。

```
# zfs create -o mountpoint=/export/zfs -o sharenfs=on -o compression=on tank/home
```

プロパティーおよびプロパティーの継承の詳細については、189 ページの「ZFS のプロパティーの紹介」を参照してください。

次に、各ファイルシステムが、プール tank の home ファイルシステムの下にグループとしてまとめられます。

4 ファイルシステムを個別に作成します。

ファイルシステムが作成されている場合があり、それからプロパティーが home レベルで変更されている場合があります。すべてのプロパティーは、ファイルシステムの使用中に動的に変更できます。

```
# zfs create tank/home/bonwick
# zfs create tank/home/billm
```

これらのファイルシステムは、親ファイルシステムからプロパティー値を継承します。このため、/export/zfs/*user* に自動的にマウントされ、NFS を使って共有されます。/etc/vfstab や /etc/dfs/dfstab ファイルを編集する必要はありません。

ファイルシステムの作成方法の詳細については、186 ページの「ZFS ファイルシステムを作成する」を参照してください。

ファイルシステムのマウントおよび共有の詳細については、211 ページの「ZFS ファイルシステムをマウントおよび共有する」を参照してください。

5 ファイルシステム固有のプロパティーを設定します。

この例では、ユーザー bonwick に 10G バイトの割り当て制限が適用されます。このプロパティーを設定すると、プールで消費できる容量に関係なく、ユーザーが使用できる容量に制限が適用されます。

```
# zfs set quota=10G tank/home/bonwick
```

6 結果を確認します。

`zfs list` コマンドを使用して、利用できるファイルシステムの情報を確認します。

```
# zfs list
NAME                   USED  AVAIL  REFER  MOUNTPOINT
tank                  92.0K  67.0G   9.5K  /tank
tank/home             24.0K  67.0G     8K  /export/zfs
tank/home/billm          8K  67.0G     8K  /export/zfs/billm
tank/home/bonwick        8K  10.0G     8K  /export/zfs/bonwick
```

ユーザー `bonwick` が使用できる容量は 10G バイトだけですが、ユーザー `billm` はプール全体 (67G バイト) を使用できます。

ファイルシステムの状態の詳細については、204 ページの「ZFS ファイルシステムの情報のクエリー検索を行う」を参照してください。

ディスク領域がどのように使用および計算されるかの詳細については、62 ページの「ZFS のディスク領域の計上」を参照してください。

◆ ◆ ◆ 第 3 章

Oracle Solaris ZFS ファイルシステムと従来のファイルシステムの相違点

この章では、Oracle Solaris ZFS ファイルシステムと従来のファイルシステムの重要な相違点についていくつか説明します。これらの重要な相違点を理解していれば、従来のツールを使用して ZFS を操作するときの混乱を少なくすることができます。

この章は、次の節で構成されます。

ZFS ファイルシステムの構造

ファイルシステムはこれまで、1 つのデバイスの制約、したがってそのデバイスのサイズの制約を受けてきました。サイズの制限があるために、従来のファイルシステムの作成および再作成には時間がかかり、場合によっては難しい作業になります。従来のボリューム管理製品は、この作業の管理を支援するためのものです。

ZFS ファイルシステムは特定のデバイスに制限されないため、ディレクトリを作成する場合と同じようにすばやく簡単に作成できます。ZFS ファイルシステムは、格納先のストレージプールに割り当てられたディスク容量の範囲内で自動的に拡張されます。

1 つのファイルシステム (`/export/home` など) を作成して多数のユーザーサブディレクトリを管理する代わりに、ユーザーごとに 1 つのファイルシステムを作成できます。階層内に含まれる子孫ファイルシステムに継承可能なプロパティーを適用することで、多数のファイルシステムの設定や管理を容易に行えます。

ファイルシステム階層の作成方法を示す例については、56 ページの「ZFS ファイルシステム階層を作成する」を参照してください。

ZFS のディスク領域の計上

ZFS は、プールストレージの概念に基づいて構成されます。標準的なファイルシステムでは物理ストレージにマッピングされますが、ZFS ファイルシステムはすべてがプールの中にあって、プール内で使用可能なストレージを共有しています。このため、`df` などのユーティリティーから報告される使用可能なディスク領域は、ファイルシステムがアクティブでないときでも変化する可能性があります。これは、プール内のほかのファイルシステムがディスク領域を消費したり解放したりするためです。

ファイルシステムの最大サイズは、割り当て制限を使用して制限できます。割り当て制限の詳細については、219 ページの「ZFS ファイルシステムに割り当て制限を設定する」を参照してください。予約を使用すれば、指定されたディスク容量をファイルシステムに保証することができます。予約については、223 ページの「ZFS ファイルシステムに予約を設定する」を参照してください。このモデルは、NFS モデルによく似ています。つまり、複数のディレクトリが 1 つのファイルシステム (`/home` など) からマウントされます。

ZFS では、すべてのメタデータが動的に割り当てられます。ZFS 以外のほとんどのファイルシステムでは、多くのメタデータが事前に割り当てられます。そのため、ファイルシステムの作成時にこのメタデータの領域コストが即座に必要となります。これは、ファイルシステムでサポートされる合計ファイル数も、事前に決定されていることを意味します。ZFS では必要に応じてメタデータが割り当てられるので、初期領域を割り当てる必要がなく、ファイル数も使用可能なディスク領域に応じて制限されるだけです。`df -g` コマンドの出力は、ZFS と ZFS 以外のファイルシステムで解釈を変える必要があります。報告される `total files` は、プール内で使用できるストレージ容量に基づいて見積もった数値に過ぎません。

ZFS は、トランザクションファイルシステムです。ファイルシステムの変更のほとんどは、トランザクショングループに関連付けられ、ディスクに非同期にコミットされます。ディスクにコミットされる前の変更は、「保留状態の変更」と呼ばれます。ファイルまたはファイルシステムが使用するディスク領域、使用できるディスク領域、および参照するディスク領域の総計に、保留状態の変更は考慮されません。保留状態の変更は通常、数秒以内に計上されます。`fsync(3c)` や `O_SYNC` を使用してディスクへの変更をコミットしても、ディスク領域の使用状況の情報がすぐに更新されることが保証されているわけではありません。

`du` コマンドおよび `df` コマンドによって報告される ZFS ディスク領域の消費量については、次のリンクを参照してください。

`http://hub.opensolaris.org/bin/view/Community+Group+zfs/faq/#whydusize`

領域が不足した場合の動作

ZFS では、ファイルシステムのスナップショットを負荷をかけずに簡単に作成できます。スナップショットは、ほとんどの ZFS 環境でよく使用されます。ZFS スナップショットについては、第 7 章「Oracle Solaris ZFS のスナップショットとクローンの操作」を参照してください。

スナップショットが存在していると、ディスク領域を解放しようとするときに、予期しない動作が発生することがあります。適切なアクセス権が付与されている場合には、通常はファイルシステム全体からファイルを削除することで、ファイルシステムで利用できるディスク領域を増やすことができます。ただし、削除しようとするファイルがファイルシステムのスナップショットとして存在する場合には、そのファイルを削除してもディスク領域は解放されません。このファイルの使用するブロックは、スナップショットから引き続き参照されます。

つまり、ファイルを削除しているのに、さらに多くのディスク領域が使用されることがあります。新しい状態の名前空間を反映するために、新しいディレクトリの作成が必要になるためです。このため、ファイルを削除しようとすると、予期しない `ENOSPC` または `EDQUOT` エラーが返される可能性があります。

ZFS ファイルシステムをマウントする

ZFS を使えば複雑さが減り、管理が容易になります。たとえば、従来のファイルシステムでは、新しいファイルシステムを追加するたびに `/etc/vfstab` ファイルを編集する必要があります。ZFS では、データセットのプロパティーに基づいてファイルシステムのマウントとマウント解除を自動的に行うことで、この作業を不要にしました。`/etc/vfstab` ファイルの ZFS エントリを管理する必要はありません。

ZFS ファイルシステムのマウントおよび共有の詳細については、211 ページの「ZFS ファイルシステムをマウントおよび共有する」を参照してください。

従来のボリューム管理

46 ページの「プールされた ZFS ストレージ」で説明したように、ZFS ではボリュームマネージャーを個別に操作する必要はありません。ZFS は raw デバイス上で動作するため、論理ボリューム (ソフトウェアまたはハードウェア) で構成されるストレージプールを作成することもできます。ただし、この構成は推奨していません。ZFS は、raw 物理デバイスを使用するときに最適に動作するようになっているためです。論理ボリュームを使用すると、パフォーマンスまたは信頼性、あるいはその両方が損なわれることがあるため、使用するべきではありません。

新しい Solaris ACL モデル

以前のバージョンの Solaris OS では、主に POSIX ACL ドラフト仕様に基づく ACL 実装がサポートされていました。POSIX ドラフトベースの ACL は、UFS ファイルを保護するために使用されます。NFSv4 仕様に基づく新しい Solaris ACL モデルは、ZFS ファイルを保護するために使用されます。

新しい Solaris ACL モデルは、主に次の点が異なっています。

- このモデルは NFSv4 仕様に基づいており、NT 形式の ACL に似ています。
- このモデルは、より詳細なアクセス権を提供します。
- ACL は、`setfacl` や `getfacl` コマンドではなく、`chmod` および `ls` コマンドを使用して設定および表示します。
- ディレクトリのアクセス特権をどのようにサブディレクトリに適用するかを指定するために、より多くの継承セマンティクスを利用できます。

ZFS ファイルで ACL を使用するときの詳細については、第 8 章「ACL による Oracle Solaris ZFS ファイルの保護」を参照してください。

◆ ◆ ◆ 第 4 章

Oracle Solaris ZFS ストレージプールの管理

この章では、Oracle Solaris ZFS でストレージプールを作成および管理する方法について説明します。

この章は、次の節で構成されます。

ZFS ストレージプールのコンポーネント

以降の節では、次のストレージプールのコンポーネントについて詳しく説明します。

ZFS ストレージプール内でディスクを使用する

ストレージプールのもっとも基本的な要素は、物理ストレージです。128M バイト以上のサイズであれば、任意のブロック型デバイスを物理ストレージとして利用できます。このデバイスは通常、`/dev/dsk` ディレクトリとしてシステムから認識されるハードドライブです。

ディスク全体 (c1t0d0) または個別のスライス (c0t0d0s7) をストレージデバイスとして利用できます。推奨される操作モードは、ディスク全体を使用する方法です。この場合、ディスクが特別なフォーマットである必要はありません。ZFS によって、EFI ラベルを使用する 1 つの大きなスライスのディスクとしてフォーマットされます。この方法を使用した場合に、format コマンドで表示されるパーティションテーブルは、次のような内容になります。

```
Current partition table (original):
Total disk sectors available: 286722878 + 16384 (reserved sectors)

Part      Tag    Flag     First Sector         Size         Last Sector
  0        usr    wm                34      136.72GB          286722911
  1 unassigned    wm                 0           0                  0
  2 unassigned    wm                 0           0                  0
  3 unassigned    wm                 0           0                  0
  4 unassigned    wm                 0           0                  0
  5 unassigned    wm                 0           0                  0
  6 unassigned    wm                 0           0                  0
  8   reserved    wm         286722912        8.00MB          286739295
```

ディスク全体を使用するには、/dev/dsk/cNtNdN 命名規則を使用してディスクに名前を付ける必要があります。他社製のドライバの中には、異なる命名規則を使用したり、ディスクを /dev/dsk ディレクトリ以外の場所に配置するものがあります。これらのディスクを使用する場合は、ディスクの名前を手動で付けて、ZFS にスライスを渡す必要があります。

ディスク全体を使ってストレージプールを作成するときは、EFI ラベルが適用されます。EFI ラベルの詳細については、『Solaris のシステム管理 (デバイスとファイルシステム)』の「EFI ディスクラベル」を参照してください。

ZFS ルートプールに使用するディスクは、EFI ラベルではなく、SMI ラベルを使用して作成する必要があります。format - e コマンドを使用して、ディスクのラベルを SMI ラベルに変更することができます。

ディスクを指定するときには、フルパス (/dev/dsk/c1t0d0 など) または /dev/dsk ディレクトリ内のデバイス名で構成される短縮名 (c1t0d0 など) を使用できます。有効なディスク名の例を挙げます。

- c1t0d0
- /dev/dsk/c1t0d0
- /dev/foo/disk

物理ディスクの全体を使用するのが、ZFS ストレージプールを作成するためのもっとも簡単な方法です。ディスクスライス、ハードウェア RAID アレイ内の LUN、またはソフトウェアベースのボリュームマネージャーが提供するボリュームからプールを構築する場合、管理、信頼性、およびパフォーマンスの観点から ZFS 構成が次第により複雑になります。次の点を考慮すれば、ほかのハードウェアまたはソフトウェアストレージ解決策を使って ZFS を構成する方法を決定しやすくなる可能性があります。

- ハードウェア RAID アレイの LUN 上に ZFS 構成を構築する場合、ZFS の冗長機能とアレイが提供する冗長機能との関係を理解する必要があります。ある構成で十分な冗長性やパフォーマンスが得られても、別の構成ではそうならない可能性もあります。
- Solaris Volume Manager (SVM) や Veritas Volume Manager (VxVM) など、ソフトウェアベースのボリュームマネージャーが提供するボリュームを使って ZFS 用の論理デバイスを構築することも可能です。ただし、そうした構成は推奨されません。ZFS はこのようなデバイス上でも正しく動作しますが、最適なパフォーマンスが得られない場合があります。

ストレージプールの推奨事項に関するその他の情報は、次の ZFS ベストプラクティスのサイトを参照してください。

http://www.solarisinternals.com/wiki/index.php/ZFS_Best_Practices_Guide

ディスクは、パスとデバイス ID の両方で識別されます (利用できる場合)。デバイス ID 情報が利用可能なシステムでは、この識別方式を使うことで、ZFS を更新することなくデバイスを再構成できます。デバイス ID の生成および管理の方式はシステムごとに異なるため、コントローラ間でディスクを移動するなどのデバイス移動の前にまず、プールをエクスポートします。ファームウェアの更新やその他のハードウェア変更などのシステムイベントによって、ZFS ストレージプール内でデバイス ID が変化する場合があり、これが原因でデバイスが利用不能になる可能性があります。

ZFS ストレージプール内でスライスを使用する

ディスクスライスを使ってストレージプールを作成するときは、従来の Solaris VTOC (SMI) ラベルを使ってディスクに名前を付けることができます。

ブート可能な ZFS ルートプールの場合、プール内のディスクにはスライスが含まれ、ディスクには SMI ラベルが付けられていなければなりません。もっとも単純な構成としては、すべてのディスク容量をスライス 0 に割り当て、そのスライスをルートプールに使用します。

SPARC システムの 72G バイトのディスクに、68G バイトの使用可能領域がスライス 0 に配置されています。次の `format` の出力を参照してください。

```
# format
.
.
.
Specify disk (enter its number): 4
selecting c1t1d0
partition> p
Current partition table (original):
Total disk cylinders available: 14087 + 2 (reserved cylinders)
```

```
Part      Tag    Flag     Cylinders         Size            Blocks
  0       root    wm       0 - 14086       68.35GB    (14087/0/0) 143349312
  1 unassigned    wm       0                0         (0/0/0)             0
  2     backup    wm       0 - 14086       68.35GB    (14087/0/0) 143349312
  3 unassigned    wm       0                0         (0/0/0)             0
  4 unassigned    wm       0                0         (0/0/0)             0
  5 unassigned    wm       0                0         (0/0/0)             0
  6 unassigned    wm       0                0         (0/0/0)             0
  7 unassigned    wm       0                0         (0/0/0)             0
```

x86 システムの 72G バイトのディスクに、68G バイトの使用可能ディスク領域がスライス 0 に配置されています。次の `format` の出力を参照してください。少量のブート情報がスライス 8 に格納されています。スライス 8 は管理不要で、変更することはできません。

```
# format
.
.
.
selecting c1t0d0
partition> p
Current partition table (original):
Total disk cylinders available: 49779 + 2 (reserved cylinders)

Part      Tag    Flag     Cylinders         Size            Blocks
  0       root    wm       1 - 49778       68.36GB    (49778/0/0) 143360640
  1 unassigned    wu       0                0         (0/0/0)             0
  2     backup    wm       0 - 49778       68.36GB    (49779/0/0) 143363520
  3 unassigned    wu       0                0         (0/0/0)             0
  4 unassigned    wu       0                0         (0/0/0)             0
  5 unassigned    wu       0                0         (0/0/0)             0
  6 unassigned    wu       0                0         (0/0/0)             0
  7 unassigned    wu       0                0         (0/0/0)             0
  8       boot    wu       0 -     0        1.41MB    (1/0/0)          2880
  9 unassigned    wu       0                0         (0/0/0)             0
```

ZFS ストレージプール内のファイルを使用する

ZFS では、ストレージプール内の UFS ファイルを仮想デバイスとして使用することもできます。この機能は、本稼働環境で使用するのではなく、主にテストや簡単な実験のために使用します。これは、ファイルをどのように使用する場合でも、整合性を保つために背後のファイルシステムに依存しているためです。ZFS プールを UFS ファイルシステム上のファイルに基づいて作成する場合には、正確さと同期のセマンティクスを保証するために、UFS に暗黙に依存しています。

ただし、ZFS を初めて使用してみる場合や、十分な物理デバイスがない状況で複雑な構成を実験する場合には、これらのファイルが非常に便利なことがあります。すべてのファイルは、完全なパスで指定し、64M バイト以上のサイズにする必要があります。

ZFS ストレージプールの複製機能

ZFS には、ミラー化構成と RAID-Z 構成において、自己修復プロパティーに加えてデータ冗長性も用意されています。

ミラー化されたストレージプール構成

ストレージプール構成をミラー化するには、2 つのディスクが必要です。ディスクごとに個別のコントローラを割り当てることをお勧めします。ミラー化構成では、多数のディスクを使用できます。また、各プール内に複数のミラーを作成することもできます。概念的には、基本的なミラー化構成は次のようになります。

```
mirror c1t0d0 c2t0d0
```

概念的により複雑なミラー化構成は、次のようになります。

```
mirror c1t0d0 c2t0d0 c3t0d0 mirror c4t0d0 c5t0d0 c6t0d0
```

ミラー化されたストレージプールの作成方法については、72 ページの「ミラー化されたストレージプールを作成する」を参照してください。

RAID-Z ストレージプール構成

ZFS は、ミラー化ストレージプール構成のほかに、シングルパリティー、ダブルパリティー、またはトリプルパリティーの耐障害性を備えた RAID-Z 構成も提供します。シングルパリティーの RAID-Z (`raidz` または `raidz1`) は RAID-5 に似ています。ダブルパリティーの RAID-Z (`raidz2`) は RAID-6 に似ています。

RAIDZ-3 (`raidz3`) については、次のブログを参照してください。

`http://blogs.sun.com/ahl/entry/triple_parity_raid_z`

従来の RAID-5 に似たすべてのアルゴリズム (RAID-4、RAID-6、RDP、EVEN-ODD など) では、「RAID-5 書き込みホール」と呼ばれる問題が発生する可能性があります。RAID-5 ストライプが書き込まれている途中に電源が切断され、すべてのブロックがディスクにまだ書き込まれていない場合、パリティーとデータが同期されないままになり、あとでストライプ全体を書き込んで上書きしない限り、永久に使用できない状態になります。RAID-Z では、可変幅の RAID ストライプを使用し

て、すべての書き込みがストライプ全体を書き込むようになっています。ZFS では、ファイルシステムとデバイス管理を統合して、可変幅の RAID ストライプの処理に必要な基本となるデータ冗長モデルの情報をファイルシステムのメタデータに十分に取り込むことによって、この設計を実現しています。RAID-Z は、RAID-5 書き込みホールをソフトウェアだけで解決する、世界初のソリューションです。

RAID-Z 構成はサイズ X のディスク N 基とパリティーディスク P 基で構成されているので、約 (N-P)*X バイトを保管することができ、P 個のデバイスで障害が発生してもデータの完全性が低下することがありません。シングルパリティーの RAID-Z 構成には 2 基以上のディスク、ダブルパリティーの RAID-Z 構成には 3 基以上のディスクが必要になります。たとえば、3 つのディスクで構成されるシングルパリティー RAID-Z 構成の場合には、パリティーデータが占有するディスク領域は 3 つのディスクのいずれかです。それ以外の点では、RAID-Z 構成を作成するために特別なハードウェアは必要ありません。

3 つのディスクを使用する RAID-Z 構成の概念は、次のようになります。

```
raidz c1t0d0 c2t0d0 c3t0d0
```

概念的により複雑な RAID-Z 構成は、次のようになります。

```
raidz c1t0d0 c2t0d0 c3t0d0 c4t0d0 c5t0d0 c6t0d0 c7t0d0 raidz c8t0d0 c9t0d0 c10t0d0 c11t0d0
c12t0d0 c13t0d0 c14t0d0
```

多数のディスクを使用する RAID-Z 構成を作成している場合は、複数のグループにディスクを分割することを検討してください。たとえば、14 台のディスクを使用する RAID-Z 構成は、ディスク 7 台ずつの 2 つのグループに分割するほうが適切です。RAID-Z 構成をディスクグループに分割する場合には、その数を 1 桁にすると、パフォーマンスが向上します。

RAID-Z ストレージプールの作成方法については、74 ページの「RAID-Z ストレージプールを作成する」を参照してください。

パフォーマンスやディスク容量を考慮したうえでミラー化構成または RAID-Z 構成のどちらを選択するかについての詳細は、次のブログエントリを参照してください。

http://blogs.sun.com/roch/entry/when_to_and_not_to

RAID-Z ストレージプールの推奨事項に関するその他の情報は、次の ZFS ベストプラクティスのサイトを参照してください。

http://www.solarisinternals.com/wiki/index.php/ZFS_Best_Practices_Guide

ZFS ハイブリッドストレージプール

Oracle の Sun Storage 7000 製品シリーズで利用可能な ZFS ハイブリッドストレージプールは、パフォーマンスの向上および容量の増加と同時に電力消費の削減を実現するために、DRAM、SSD、および HDD を組み合わせた特殊なストレージプールです。この製品の管理インタフェースでは、ストレージプールの ZFS 冗長構成を選択したり、その他の構成オプションを容易に管理したりできます。

この製品の詳細については、『Sun Storage Unified Storage System Administration Guide』を参照してください。

冗長構成の自己修復データ

ZFS のミラー化構成または RAID-Z 構成は、自己修復データを備えています。

不正なデータブロックが検出されると、ZFS は別の冗長コピーから正しいデータを取得するだけでなく、不正なデータを適切なコピーで置き換えて修復も行います。

ストレージプール内の動的なストライプ

ZFS では、すべての最上位レベルの仮想デバイス間でデータが動的にストライプ化されます。データがどこに割り当てられるかは書き込み時に決定されるため、割り当て時には固定幅のストライプは作成されません。

新しい仮想デバイスがプールに追加されると、パフォーマンスとディスク領域割り当てポリシーを維持するために、データは新しいデバイスに順次割り当てられます。各仮想デバイスは、ほかのディスクデバイスまたはファイルを含むミラーまたは RAID-Z デバイスでもかまいません。この構成を使用すれば、プールの障害時の特性を柔軟に制御できます。たとえば、4 つのディスクから次の構成を作成できます。

- 動的なストライプを使用する 4 つのディスク
- 4 方向の RAID-Z 構成を 1 つ
- 動的なストライプを使用する 2 方向のミラーを 2 つ

ZFS では異なる種類の仮想デバイスを同じプール内で組み合わせることが可能ですが、そのような組み合わせは避けてください。たとえば、2 方向のミラー構成と 3 方向の RAID-Z 構成を含むプールを作成できます。ただし、耐障害性は、もっとも低い仮想デバイス (この場合は RAID-Z) と同じになります。最上位の仮想デバイスは同じ種類のデバイスを使用し、各デバイスで同じ冗長レベルにするのがもっとも良い方法です。

ZFS ストレージプールを作成および破棄する

以降の節では、ZFS ストレージプールを作成および破棄するさまざまなシナリオについて説明します。

プールはすばやく簡単に作成して破棄できるようになっています。ただし、これらの操作を実行するときには注意が必要です。使用中であることがわかっているデバイスについては、新しいプールで使用されないようにするためのチェックが実行されます。ただし、すでに使用されているデバイスを常に認識できるわけではありません。プールの破棄はプールの作成よりも簡単です。`zpool destroy` は、注意深く実行してください。この単純なコマンドは重大な結果をもたらします。

ZFS ストレージプールを作成する

ストレージプールを作成するには、`zpool create` コマンドを使用します。このコマンドの引数には、プール名および任意の数の仮想デバイスを指定します。プール名は、51 ページの「ZFS コンポーネントに名前を付けるときの規則」の規則に従って付ける必要があります。

基本的なストレージプールを作成する

次のコマンドでは、ディスク `c1t0d0` および `c1t1d0` で構成される、`tank` という名前の新しいプールが作成されます。

```
# zpool create tank c1t0d0 c1t1d0
```

ディスク全体を表すデバイス名は `/dev/dsk` ディレクトリに作成されます。適切な名前が自動的に割り当てられ、1 つの大きなスライスで構成されます。データは、両方のディスクに動的にストライプ化されます。

ミラー化されたストレージプールを作成する

ミラー化されたプールを作成するには、`mirror` キーワードと、ミラーを構成する任意の数のストレージデバイスを使用します。複数のミラーを指定する場合は、コマンド行で `mirror` キーワードを繰り返すことができます。次のコマンドでは、1 つのプールと 2 つの 2 方向ミラーが作成されます。

```
# zpool create tank mirror c1d0 c2d0 mirror c3d0 c4d0
```

2 番目の `mirror` キーワードでは、新しい最上位仮想デバイスを指定しています。データは両方のミラーにまたがって動的にストライプ化され、各ディスク間で適切に冗長化されます。

推奨されるミラー化構成の詳細については、次のサイトを参照してください。

http://www.solarisinternals.com/wiki/index.php/ZFS_Best_Practices_Guide

現時点では、ZFS ミラー化構成では次の操作がサポートされています。

- 最上位レベルの追加の仮想デバイス (vdev) 用の別のディスクセットを既存のミラー化構成に追加する。詳細は、82 ページの「ストレージプールにデバイスを追加する」を参照してください。
- 追加のディスクを既存のミラー化構成に接続する。あるいは、追加のディスクを複製されない構成に接続して、ミラー化構成を作成する。詳細は、87 ページの「ストレージプール内でデバイスを接続する/切り離す」を参照してください。
- 置き換えたあとのディスクのサイズが置き換える前のデバイスと等しいかそれ以上であれば、既存のミラー化構成内の 1 つ以上のディスクを置き換える。詳細は、96 ページの「ストレージプール内のデバイスを置き換える」を参照してください。
- 残りのデバイスがミラー化構成に十分な冗長性を備えているのであれば、その構成に含まれる 1 つのディスクを切り離す。詳細は、87 ページの「ストレージプール内でデバイスを接続する/切り離す」を参照してください。
- ディスクのうちの 1 つを切り離すことによってミラー化構成を分割し、新しい同一のプールを作成する。詳細は、89 ページの「ミラー化 ZFS ストレージプールを分割して新しいプールを作成する」を参照してください。

ログデバイスまたはキャッシュデバイス以外のデバイスは、ミラー化されたストレージプールから完全に削除できません。この機能については、RFE (改善要求) が提出されています。

ZFS ルートプールを作成する

ZFS ルートファイルシステムをインストールしてブートできます。ルートプールの構成に関する次の情報を確認してください。

- ルートプールに使用するディスクには VTOC (SMI) ラベルが必要です。また、ディスクスライスを使用してプールを作成する必要があります。
- ルートプールは、ミラー化構成または単一ディスク構成として作成する必要があります。`zpool add` コマンドを使ってディスクを追加し、複数のミラー化された最上位レベル仮想ディスクを作成することはできませんが、ミラー化された仮想デバイスを `zpool attach` コマンドを使って拡張することは可能です。
- RAID-Z やストライプ化構成はサポートされていません。
- ルートプールに別個のログデバイスを使用することはできません。
- ルートプールにサポートされていない構成を使用しようとすると、次のようなメッセージが表示されます。

```
ERROR: ZFS pool <pool-name> does not support boot environments
```

```
# zpool add -f rpool log c0t6d0s0
cannot add to 'rpool': root pool can not have multiple vdevs or separate logs
```

ZFS ルートファイルシステムのインストールと起動の詳細については、第 5 章「Oracle Solaris ZFS ルートファイルシステムのインストールと起動」を参照してください。

RAID-Z ストレージプールを作成する

シングルパリティー RAID-Z プールの作成方法は、ミラー化されたプールの作成方法と同じですが、`mirror` キーワードの代わりに `raidz` または `raidz1` を使用する点が異なります。次の例では、5 個のディスクで構成される 1 つの RAID-Z デバイスを使ってプールを作成する方法を示します。

```
# zpool create tank raidz c1t0d0 c2t0d0 c3t0d0 c4t0d0 /dev/dsk/c5t0d0
```

この例では、デバイスの短縮名または完全名を使ってディスクを指定できることを示しています。`/dev/dsk/c5t0d0` と `c5t0d0` はどちらも同じディスクを参照します。

プールの作成時に `raidz2` キーワードを使用するとダブルパリティーの、`raidz3` キーワードを使用するとトリプルパリティーの RAID-Z 構成を作成できます。次に例を示します。

```
# zpool create tank raidz2 c1t0d0 c2t0d0 c3t0d0 c4t0d0 c5t0d0
# zpool status -v tank
  pool: tank
 state: ONLINE
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        tank        ONLINE       0     0     0
          raidz2-0  ONLINE       0     0     0
            c1t0d0  ONLINE       0     0     0
            c2t0d0  ONLINE       0     0     0
            c3t0d0  ONLINE       0     0     0
            c4t0d0  ONLINE       0     0     0
            c5t0d0  ONLINE       0     0     0

errors: No known data errors

# zpool create tank raidz3 c0t0d0 c1t0d0 c2t0d0 c3t0d0 c4t0d0 c5t0d0 c6t0d0 c7t0d0
# zpool status -v tank
  pool: tank
 state: ONLINE
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        tank        ONLINE       0     0     0
          raidz3-0  ONLINE       0     0     0
            c0t0d0  ONLINE       0     0     0
            c1t0d0  ONLINE       0     0     0
```

```
            c2t0d0  ONLINE       0     0     0
            c3t0d0  ONLINE       0     0     0
            c4t0d0  ONLINE       0     0     0
            c5t0d0  ONLINE       0     0     0
            c6t0d0  ONLINE       0     0     0
            c7t0d0  ONLINE       0     0     0

errors: No known data errors
```

現時点では、ZFS RAID-Z 構成では次の操作がサポートされています。

- 最上位レベルの追加の仮想デバイス用の別のディスクセットを既存の RAID-Z 構成に追加する。詳細は、82 ページの「ストレージプールにデバイスを追加する」を参照してください。
- 置き換えたあとのディスクのサイズが置き換える前のデバイスと等しいかそれ以上であれば、既存の RAID-Z 構成内の 1 つ以上のディスクを置き換える。詳細は、96 ページの「ストレージプール内のデバイスを置き換える」を参照してください。

現時点では、RAID-Z 構成では次の操作がサポートされていません。

- 追加のディスクを既存の RAID-Z 構成に接続する。
- RAID-Z 構成からディスクを切り離す (スペアディスクによって置き換えられるディスクを切り離す場合を除く)。
- ログデバイスまたはキャッシュデバイス以外のデバイスは、RAID-Z 構成から完全に削除できません。この機能については、RFE (改善要求) が提出されています。

RAID-Z 構成の詳細については、69 ページの「RAID-Z ストレージプール構成」を参照してください。

ログデバイスを持つ ZFS ストレージプールを作成する

デフォルトでは、ZIL はメインプール内のブロックから割り当てられます。しかし、NVRAM や専用ディスクなどで、別個のインテントログデバイスを使用することにより、パフォーマンスを向上できる可能性があります。ZFS ログデバイスの詳細については、34 ページの「別個の ZFS ログデバイスの設定」を参照してください。

ZFS ログデバイスの設定は、ストレージプールの作成時または作成後に行えます。

次の例では、ミラー化ログデバイスを持つミラー化ストレージプールを作成する方法を示します。

```
# zpool create datap mirror c1t1d0 c1t2d0 mirror c1t3d0 c1t4d0 log mirror c1t5d0 c1t8d0
# zpool status datap
  pool: datap
 state: ONLINE
 scrub: none requested
config:
```

```
        NAME        STATE     READ WRITE CKSUM
        datap       ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c1t1d0  ONLINE       0     0     0
            c1t2d0  ONLINE       0     0     0
          mirror-1  ONLINE       0     0     0
            c1t3d0  ONLINE       0     0     0
            c1t4d0  ONLINE       0     0     0
        logs
          mirror-2  ONLINE       0     0     0
            c1t5d0  ONLINE       0     0     0
            c1t8d0  ONLINE       0     0     0

errors: No known data errors
```

ログデバイス障害からの回復の詳細については、例 11–2 を参照してください。

キャッシュデバイスを使用して ZFS ストレージプールを作成する

キャッシュデバイスを使用してストレージプールを作成して、ストレージプールデータをキャッシュすることができます。次に例を示します。

```
# zpool create tank mirror c2t0d0 c2t1d0 c2t3d0 cache c2t5d0 c2t8d0
# zpool status tank
  pool: tank
 state: ONLINE
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        tank        ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c2t0d0  ONLINE       0     0     0
            c2t1d0  ONLINE       0     0     0
            c2t3d0  ONLINE       0     0     0
        cache
          c2t5d0    ONLINE       0     0     0
          c2t8d0    ONLINE       0     0     0

errors: No known data errors
```

キャッシュデバイスを使用して ZFS ストレージプールを作成するかどうか決定する場合は、次の点を考慮してください。

- キャッシュデバイスを使用すると、ほぼ静的なコンテンツをランダムに読み込む作業負荷のパフォーマンスが大幅に向上します。
- `zpool iostat` コマンドを次のように使用して、容量と読み込みを監視できます。
- プールの作成時に単一または複数のキャッシュデバイスを追加できます。プールの作成後にキャッシュデバイスを追加または削除することもできます。詳細は、例 4–4 を参照してください。
- キャッシュデバイスは、ミラー化することも、RAID-Z 構成に含めることもできません。

- キャッシュデバイスで読み取りエラーが検出されると、ミラー化構成または RAID-Z 構成に含まれている可能性があるオリジナルのストレージプールデバイスに対して、その読み取り I/O が再発行されます。キャッシュデバイスの内容は、ほかのシステムキャッシュと同様に揮発的とみなされます。

ストレージプールの仮想デバイスの情報を表示する

個々のストレージプールには 1 つ以上の仮想デバイスが含まれます。「仮想デバイス」は、物理ストレージのレイアウトとストレージプールの障害時の特性を定義した、ストレージプールの内部表現です。つまり、仮想デバイスは、ストレージプールの作成に使用されるディスクデバイスまたはファイルを表しています。プールでは、構成の最上位に任意の数の仮想デバイス (「最上位レベル vdev」と呼ばれる) を含めることができます。

最上位の仮想デバイスに 2 つ以上の物理デバイスが含まれている場合、その構成はミラーまたは RAID-Z 仮想デバイスとしてのデータ冗長性を備えます。これらの仮想デバイスは、複数のディスク、ディスクスライス、またはファイルで構成されています。スペアは、プールで利用可能なホットスペアを追跡する特殊な仮想デバイスです。

次の例では、2 つの最上位仮想デバイスから成るプールを作成する方法を示します。仮想デバイスはそれぞれ 2 ディスクのミラーです。

```
# zpool create tank mirror c1d0 c2d0 mirror c3d0 c4d0
```

次の例では、4 ディスクから成る 1 つの最上位仮想デバイスで構成されたプールを作成する方法を示します。

```
# zpool create mypool raidz2 c1d0 c2d0 c3d0 c4d0
```

`zpool add` コマンドを使用して、このプールに別の最上位仮想デバイスを追加できます。次に例を示します。

```
# zpool add mypool raidz2 c2d1 c3d1 c4d1 c5d1
```

非冗長プールで使用されるディスク、ディスクスライス、またはファイルは、最上位の仮想デバイスとして機能します。ストレージプールには通常、複数の最上位レベルの仮想デバイスが含まれています。ZFS では、プール内のすべての最上位レベルの仮想デバイス間でデータが動的にストライプ化されます。

ZFS ストレージプールに含まれている仮想デバイスと物理デバイスは、`zpool status` コマンドで表示されます。次に例を示します。

```
# zpool status tank
  pool: tank
 state: ONLINE
```

```
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        tank        ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c0t1d0  ONLINE       0     0     0
            c1t1d0  ONLINE       0     0     0
          mirror-1  ONLINE       0     0     0
            c0t2d0  ONLINE       0     0     0
            c1t2d0  ONLINE       0     0     0
          mirror-2  ONLINE       0     0     0
            c0t3d0  ONLINE       0     0     0
            c1t3d0  ONLINE       0     0     0

errors: No known data errors
```

ZFS ストレージプールの作成エラーに対応する

さまざまな原因で、プールの作成エラーが発生することがあります。指定されたデバイスが存在しないなどの原因の明白なエラーもあれば、理由がはっきりしないエラーもあります。

使用中のデバイスを検出する

ZFS では、デバイスをフォーマットする前に、ディスクが ZFS またはオペレーティングシステムのほかの部分で使用されているかどうかを最初に確認します。ディスクが使用中の場合は、たとえば次のようなエラーが表示されます。

```
# zpool create tank c1t0d0 c1t1d0
invalid vdev specification
use '-f' to override the following errors:
/dev/dsk/c1t0d0s0 is currently mounted on /. Please see umount(1M).
/dev/dsk/c1t0d0s1 is currently mounted on swap. Please see swap(1M).
/dev/dsk/c1t1d0s0 is part of active ZFS pool zeepool. Please see zpool(1M).
```

エラーの中には `-f` オプションを使用することで無効にできるものもありますが、ほとんどのエラーは無効にできません。以降に示す条件の場合は `-f` オプションを指定しても無効にはできないため、手動で訂正する必要があります。

マウントされたファイルシステム	このディスクまたはそのスライスの 1 つに、現在マウントされているファイルシステムが含まれています。このエラーを訂正するには、`umount` コマンドを使用してください。
/etc/vfstab 内のファイルシステム	このディスクには、`/etc/vfstab` ファイルに指定されているファイルシステムが含まれていますが、そのファイルシステムが現在マウントされていません。このエラーを訂正するには、`/etc/vfstab` ファイルでその行をコメントにしてください。

専用のダンプデバイス	このディスクは、システム専用のダンプデバイスとして使用中です。このエラーを訂正するには、`dumpadm` コマンドを使用してください。
ZFS プールの一部	このディスクまたはファイルは、アクティブな ZFS ストレージプールに含まれています。このエラーを訂正するには、そのプールが不要であれば `zpool destroy` コマンドを使用して破棄してください。または、`zpool detach` コマンドを使用して、そのプールからディスクを切り離します。ディスクを切り離すことができるのは、ミラー化ストレージプールの場合のみです。

次の使用中チェックは警告として役に立つ情報ですが、`-f` オプションを使用して無効にすれば、プールを作成できます。

ファイルシステムを含んでいる	このディスクには既知のファイルシステムが含まれていますが、マウントされていないうえ、使用中のメッセージが表示されません。
ボリュームの一部	ディスクは Solaris Volume Manager ボリュームの一部です。
Live upgrade	このディスクは、Oracle Solaris Live Upgrade 用の代替ブート環境として使用中です。
エクスポートされた **ZFS** プール	このディスクは、エクスポートされたストレージプール、またはシステムから手動で削除されたストレージプールに含まれています。後者の場合、このプールは潜在的にアクティブとして報告されます。このディスクがネットワークに接続されたドライブとして別のシステムで使用されているかどうか、わからないためです。潜在的にアクティブなプールを無効にする場合には、注意が必要です。

次の例は、`-f` オプションの使用方法を示しています。

```
# zpool create tank c1t0d0
invalid vdev specification
use '-f' to override the following errors:
/dev/dsk/c1t0d0s0 contains a ufs filesystem.
# zpool create -f tank c1t0d0
```

できるだけ、`-f` オプションを使用してエラーを無効にする以外の方法でエラーを訂正するようにしてください。

複製レベルが一致しない

複製レベルの異なる仮想デバイスを使ってプールを作成することは、推奨されていません。zpool コマンドは、冗長レベルの一致しないプールが誤って作成されることを回避しようとします。このような構成のプールを作成しようとすると、次のようなエラーが表示されます。

```
# zpool create tank c1t0d0 mirror c2t0d0 c3t0d0
invalid vdev specification
use '-f' to override the following errors:
mismatched replication level: both disk and mirror vdevs are present
# zpool create tank mirror c1t0d0 c2t0d0 mirror c3t0d0 c4t0d0 c5t0d0
invalid vdev specification
use '-f' to override the following errors:
mismatched replication level: 2-way mirror and 3-way mirror vdevs are present
```

これらのエラーは -f オプションで無効化できますが、この方法はなるべく使用しないでください。このコマンドを使用してサイズの異なるデバイスで構成されるミラー化または RAID-Z プールを作成しようとした場合にも、警告が表示されます。この構成は可能ですが、冗長性のレベルが一致しないと、容量の大きいほうのデバイスに未使用のディスク領域が発生します。警告を無効化するには -f オプションが必要です。

ストレージプール作成のドライランを行う

プールの作成を試みると、さまざまな形態で予期しない失敗が起きる可能性があります。また、ディスクのフォーマットは好ましくない結果をもたらす可能性がある操作です。このような理由から、zpool create コマンドには、実際にはデバイスへの書き込みを行わずにプールの作成をシミュレートする -n オプションが追加で用意されています。この「ドライラン」オプションを指定すると、使用中のデバイスの確認と複製レベルの検証が行われ、処理中に発生したエラーがすべて報告されます。エラーが見つからない場合は、次のような出力が表示されます。

```
# zpool create -n tank mirror c1t0d0 c1t1d0
would create 'tank' with the following layout:

        tank
          mirror
            c1t0d0
            c1t1d0
```

一部のエラーは、プールを実際に作成しないと検出できません。たとえば、同じ構成に同じデバイスを 2 回指定していることがよくあります。このエラーは実際にデータを書き込まないと確実に検出できないため、zpool create -n コマンドでは成功が報告されるにもかかわらず、このオプションを指定せずにコマンドを実行するとプールの作成に失敗する可能性があります。

ストレージプールのデフォルトマウントポイント

プールが作成されるときに、最上位データセットのデフォルトマウントポイントは */pool-name* になります。このディレクトリは、存在しないディレクトリか、空のディレクトリにする必要があります。ディレクトリが存在しない場合は、自動的に作成されます。ディレクトリが空の場合は、root データセットが既存のディレクトリの最上位にマウントされます。別のデフォルトマウントポイントを使用してプールを作成する場合は、`-zpool create` コマンドの `m` オプションを使用します。次に例を示します。

```
# zpool create home c1t0d0
default mountpoint '/home' exists and is not empty
use '-m' option to provide a different default
# zpool create -m /export/zfs home c1t0d0
```

このコマンドを実行すると、`/export/zfs` をマウントポイントとして、新しいプール home および home データセットが作成されます。

マウントポイントの詳細については、212 ページの「ZFS マウントポイントを管理する」を参照してください。

ZFS ストレージプールを破棄する

プールを破棄するときは、`zpool destroy` コマンドを使用します。このコマンドを実行すると、マウント済みのデータセットがプールに含まれている場合でも、プールが破棄されます。

```
# zpool destroy tank
```

注意 – プールを破棄するときは、十分に注意してください。破棄するプールに間違いがないことを確認し、常にデータをコピーしておいてください。ほかのプールを間違って破棄してしまった場合は、そのプールの回復を試みることができます。詳細は、121 ページの「破棄された ZFS ストレージプールを回復する」を参照してください。

エラー状態のデバイスが含まれるプールを破棄する

プールを破棄するには、そのプールが有効でなくなったことを示すデータをディスクに書き込む必要があります。この状態情報が書き込まれたデバイスは、インポートを実行するときに、アクティブである可能性のあるプールとして表示されなくなります。1 つ以上のデバイスが使用できない状態のときでも、そのプールを破棄できます。ただし、これらの使用できないデバイスには、必要な状態情報は書き込まれません。

これらのデバイスは適切に修復された時点で、新しいプールの作成時に「潜在的にアクティブ」として報告されます。インポートするプールを検索するとき、それらのデバイスは有効なデバイスとして表示されます。エラー状態のデバイスが多いために、プール自体がエラー状態になる (最上位レベルの仮想デバイスがエラー状態になる) 場合は、このコマンドにより警告が出力され、-f オプションを指定しないとコマンドを完了できません。プールを開かないとデータがプールに格納されているかどうかがわからないときには、このオプションが必要になります。次に例を示します。

```
# zpool destroy tank
cannot destroy 'tank': pool is faulted
use '-f' to force destruction anyway
# zpool destroy -f tank
```

プールとデバイスの健全性の詳細については、113 ページの「ZFS ストレージプールの健全性状態を調べる」を参照してください。

インポートツールの詳細については、120 ページの「ZFS ストレージプールをインポートする」を参照してください。

ZFS ストレージプール内のデバイスを管理する

デバイスに関する基本情報のほとんどは、65 ページの「ZFS ストレージプールのコンポーネント」に記載してあります。プールを作成したあとに、いくつかのタスクを実行してプール内の物理デバイスを管理できます。

- 82 ページの「ストレージプールにデバイスを追加する」
- 87 ページの「ストレージプール内でデバイスを接続する/切り離す」
- 89 ページの「ミラー化 ZFS ストレージプールを分割して新しいプールを作成する」
- 93 ページの「ストレージプール内のデバイスをオンラインまたはオフラインにする」
- 95 ページの「ストレージプールデバイスのエラーをクリアーする」
- 96 ページの「ストレージプール内のデバイスを置き換える」
- 98 ページの「ストレージプールにホットスペアを指定する」

ストレージプールにデバイスを追加する

最上位レベルの新しい仮想デバイスを追加することで、プールにディスク領域を動的に追加できます。プール内のすべてのデータセットは、このディスク領域をすぐに利用できます。新しい仮想デバイスをプールに追加するときは、zpool add コマンドを使用します。次に例を示します。

```
# zpool add zeepool mirror c2t1d0 c2t2d0
```

仮想デバイスを指定する書式は zpool create コマンドの場合と同じです。デバイスは使用中かどうかを判断するために検査されます。また、このコマンドは -f オプションが指定されないかぎり冗長レベルを変更することはできません。ドライランを実行できるように、このコマンドも -n オプションをサポートしています。次に例を示します。

```
# zpool add -n zeepool mirror c3t1d0 c3t2d0
would update 'zeepool' to the following configuration:
      zeepool
        mirror
            c1t0d0
            c1t1d0
        mirror
            c2t1d0
            c2t2d0
        mirror
            c3t1d0
            c3t2d0
```

このコマンドの構文では、ミラー化されたデバイス c3t1d0 と c3t2d0 が zeepool プールの既存の構成に追加されます。

仮想デバイスがどのように検査されるかの詳細については、78 ページの「使用中のデバイスを検出する」を参照してください。

例 4–1　ZFS ミラー化構成にディスクを追加する

次の例では、Oracle の Sun Fire x4500 システムの既存の ZFS ミラー化構成に別のミラーが追加されます。

```
# zpool status tank
  pool: tank
 state: ONLINE
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        tank        ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c0t1d0  ONLINE       0     0     0
            c1t1d0  ONLINE       0     0     0
          mirror-1  ONLINE       0     0     0
            c0t2d0  ONLINE       0     0     0
            c1t2d0  ONLINE       0     0     0

errors: No known data errors
# zpool add tank mirror c0t3d0 c1t3d0
# zpool status tank
  pool: tank
 state: ONLINE
 scrub: none requested
```

例 4–1 ZFS ミラー化構成にディスクを追加する (続き)

```
config:

        NAME        STATE     READ WRITE CKSUM
        tank        ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c0t1d0  ONLINE       0     0     0
            c1t1d0  ONLINE       0     0     0
          mirror-1  ONLINE       0     0     0
            c0t2d0  ONLINE       0     0     0
            c1t2d0  ONLINE       0     0     0
          mirror-2  ONLINE       0     0     0
            c0t3d0  ONLINE       0     0     0
            c1t3d0  ONLINE       0     0     0

errors: No known data errors
```

例 4–2 RAID-Z 構成にディスクを追加する

同様にディスクを RAID-Z 構成に追加することができます。次の例は、3 つのディスクが含まれる 1 台の RAID-Z デバイスを持つストレージプールを、それぞれ 3 つのディスクが含まれる 2 台の RAID-Z デバイスを持つストレージプールに変換する方法を示しています。

```
# zpool status rzpool
  pool: rzpool
 state: ONLINE
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        rzpool      ONLINE       0     0     0
          raidz1-0  ONLINE       0     0     0
            c1t2d0  ONLINE       0     0     0
            c1t3d0  ONLINE       0     0     0
            c1t4d0  ONLINE       0     0     0

errors: No known data errors
# zpool add rzpool raidz c2t2d0 c2t3d0 c2t4d0
# zpool status rzpool
  pool: rzpool
 state: ONLINE
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        rzpool      ONLINE       0     0     0
          raidz1-0  ONLINE       0     0     0
            c1t0d0  ONLINE       0     0     0
            c1t2d0  ONLINE       0     0     0
            c1t3d0  ONLINE       0     0     0
          raidz1-1  ONLINE       0     0     0
            c2t2d0  ONLINE       0     0     0
            c2t3d0  ONLINE       0     0     0
```

例 4–2 RAID-Z 構成にディスクを追加する (続き)

```
            c2t4d0  ONLINE       0     0     0

errors: No known data errors
```

例 4–3 ミラー化ログデバイスを追加および削除する

次の例は、ミラー化ストレージプールにミラー化ログデバイスを追加する方法を示しています。ストレージプールでログデバイスを使用する方法の詳細については、34 ページの「別個の ZFS ログデバイスの設定」を参照してください。

```
# zpool status newpool
  pool: newpool
 state: ONLINE
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        newpool     ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c0t4d0  ONLINE       0     0     0
            c0t5d0  ONLINE       0     0     0

errors: No known data errors
# zpool add newpool log mirror c0t6d0 c0t7d0
# zpool status newpool
  pool: newpool
 state: ONLINE
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        newpool     ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c0t4d0  ONLINE       0     0     0
            c0t5d0  ONLINE       0     0     0
        logs
          mirror-1  ONLINE       0     0     0
            c0t6d0  ONLINE       0     0     0
            c0t7d0  ONLINE       0     0     0

errors: No known data errors
```

既存のログデバイスにログデバイスを接続して、ミラー化ログデバイスを作成できます。この操作は、ミラー化されていないストレージプール内にデバイスを接続する操作と同じです。

ログデバイスを削除するには、`zpool remove` コマンドを使用します。前の例のミラー化ログデバイスは、`mirror-1` 引数を指定することによって削除できます。次に例を示します。

例 4–3 ミラー化ログデバイスを追加および削除する (続き)

```
# zpool remove newpool mirror-1
# zpool status newpool
  pool: newpool
 state: ONLINE
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        newpool     ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c0t4d0  ONLINE       0     0     0
            c0t5d0  ONLINE       0     0     0

errors: No known data errors
```

プールの構成に含まれるログデバイスが 1 つだけの場合、デバイス名を指定することによってログデバイスを削除します。次に例を示します。

```
# zpool status pool
  pool: pool
 state: ONLINE
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        pool        ONLINE       0     0     0
          raidz1-0  ONLINE       0     0     0
            c0t8d0  ONLINE       0     0     0
            c0t9d0  ONLINE       0     0     0
        logs
          c0t10d0   ONLINE       0     0     0

errors: No known data errors
# zpool remove pool c0t10d0
```

例 4–4 キャッシュデバイスを追加および削除する

キャッシュデバイスを ZFS ストレージプールに追加したり、不要になったキャッシュデバイスを削除したりできます。

zpool add コマンドを使用して、キャッシュデバイスを追加します。次に例を示します。

```
# zpool add tank cache c2t5d0 c2t8d0
# zpool status tank
  pool: tank
 state: ONLINE
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        tank        ONLINE       0     0     0
```

例 4–4 キャッシュデバイスを追加および削除する (続き)

```
          mirror-0  ONLINE       0     0     0
            c2t0d0  ONLINE       0     0     0
            c2t1d0  ONLINE       0     0     0
            c2t3d0  ONLINE       0     0     0
        cache
          c2t5d0    ONLINE       0     0     0
          c2t8d0    ONLINE       0     0     0

errors: No known data errors
```

キャッシュデバイスは、ミラー化することも、RAID-Z 構成に含めることもできません。

zpool remove コマンドを使用して、キャッシュデバイスを削除します。次に例を示します。

```
# zpool remove tank c2t5d0 c2t8d0
# zpool status tank
 pool: tank
 state: ONLINE
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        tank        ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c2t0d0  ONLINE       0     0     0
            c2t1d0  ONLINE       0     0     0
            c2t3d0  ONLINE       0     0     0

errors: No known data errors
```

現時点では、zpool remove コマンドはホットスペア、ログデバイス、および キャッシュデバイスの削除のみをサポートしています。メインのミラー化プール構成に含まれて入るデバイスは、zpool detach コマンドを使用して削除することができます。非冗長デバイスおよび RAID-Z デバイスはプールから削除することはできません。

ZFS ストレージプールでキャッシュデバイスを使用する方法の詳細については、76 ページの「キャッシュデバイスを使用して ZFS ストレージプールを作成する」を参照してください。

ストレージプール内でデバイスを接続する/切り離す

zpool add コマンド以外に、zpool attach コマンドを使って、新しいデバイスを既存のミラー化されたデバイスまたはミラー化されていないデバイスに追加できます。

ディスクを切り離してミラー化ルートプールを作成する場合は、136 ページの「ミラー化ルートプールを作成する方法 (インストール後)」を参照してください。

ZFS ルートプールのディスクを置き換える場合は、178 ページの「ZFS ルートプールのディスクを置き換える方法」を参照してください。

例 4–5　2 方向ミラー化ストレージプールを 3 方向ミラー化ストレージプールに変換する

この例では、新しいデバイス `c2t1d0` を既存のデバイス `c1t1d0` に接続すると、既存の 2 方向ミラー `zeepool` が 3 方向ミラーに変換されます。

```
# zpool status zeepool
  pool: zeepool
 state: ONLINE
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        zeepool     ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c0t1d0  ONLINE       0     0     0
            c1t1d0  ONLINE       0     0     0

errors: No known data errors
# zpool attach zeepool c1t1d0 c2t1d0
# zpool status zeepool
  pool: zeepool
 state: ONLINE
 scrub: resilver completed after 0h0m with 0 errors on Fri Jan  8 12:59:20 2010
config:

        NAME        STATE     READ WRITE CKSUM
        zeepool     ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c0t1d0  ONLINE       0     0     0
            c1t1d0  ONLINE       0     0     0
            c2t1d0  ONLINE       0     0     0  592K resilvered

errors: No known data errors
```

たとえば、既存のデバイスが 3 方向ミラーの一部である場合は、新規デバイスを接続すると 4 方向ミラーが作成されます。どのような場合にも、新しいデバイスを接続すると、すぐに再同期化が開始されます。

例 4–6　非冗長な ZFS ストレージプールをミラー化された ZFS ストレージプールに変換する

また、`zpool attach` コマンドを使用して、非冗長なストレージプールを冗長なストレージプールに変換できます。次に例を示します。

```
# zpool create tank c0t1d0
# zpool status tank
  pool: tank
 state: ONLINE
```

例 4–6 非冗長な ZFS ストレージプールをミラー化された ZFS ストレージプールに変換する *(続き)*

```
 scrub: none requested
config:
        NAME        STATE     READ WRITE CKSUM
        tank        ONLINE       0     0     0
          c0t1d0    ONLINE       0     0     0

errors: No known data errors
# zpool attach tank c0t1d0 c1t1d0
# zpool status tank
  pool: tank
 state: ONLINE
 scrub: resilver completed after 0h0m with 0 errors on Fri Jan  8 14:28:23 2010
config:

        NAME        STATE     READ WRITE CKSUM
        tank        ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c0t1d0  ONLINE       0     0     0
            c1t1d0  ONLINE       0     0     0  73.5K resilvered

errors: No known data errors
```

`zpool detach` コマンドを使用して、ミラー化されたストレージプールからデバイスを切り離すことができます。次に例を示します。

```
# zpool detach zeepool c2t1d0
```

ただし、データの有効な複製がほかに存在しない場合、この操作は失敗します。次に例を示します。

```
# zpool detach newpool c1t2d0
cannot detach c1t2d0: only applicable to mirror and replacing vdevs
```

ミラー化 ZFS ストレージプールを分割して新しいプールを作成する

`zpool split` コマンドを使用すると、ミラー化された ZFS ストレージプールをバックアッププールとしてすばやく複製できます。

現時点では、この機能を使用してミラー化ルートプールを分割することはできません。

zpool split コマンドを使用してミラー化 ZFS ストレージプールからディスクを切り離し、切り離されたディスクのうちの 1 つを含む新しいプールを作成することができます。新しいプールの内容は、元のミラー化 ZFS ストレージプールと同じになります。

デフォルトでは、ミラー化プールに対して zpool split 操作を実行すると、最後のディスクが切り離され、新しく作成されるプールで使用されます。分割操作のあとで、新しいプールをインポートします。次に例を示します。

```
# zpool status tank
  pool: tank
 state: ONLINE
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        tank        ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c1t0d0  ONLINE       0     0     0
            c1t2d0  ONLINE       0     0     0

errors: No known data errors
# zpool split tank tank2
# zpool import tank2
# zpool status tank tank2
  pool: tank
 state: ONLINE
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        tank        ONLINE       0     0     0
          c1t0d0    ONLINE       0     0     0

errors: No known data errors

  pool: tank2
 state: ONLINE
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        tank2       ONLINE       0     0     0
          c1t2d0    ONLINE       0     0     0

errors: No known data errors
```

新しく作成されるプールでどのディスクを使用するかは、zpool split コマンドで指定できます。次に例を示します。

```
# zpool split tank tank2 c1t0d0
```

実際の分割操作が行われる前に、メモリー上のデータがミラー化ディスクに書き出されます。データが書き出されたあとで、ディスクがプールから切り離されて新し

いプール GUID を付与されます。新しいプール GUID が生成され、プールが分割されたのと同じシステム上でプールをインポートできるようになります。

分割されるプールのデータセットマウントポイントがデフォルトと異なっている場合に、新しいプールを同じシステム上に作成するには、`zpool split -R` オプションを使用して新しいプール用の代替ルートディレクトリを特定し、既存のマウントポイントと競合しないようにする必要があります。次に例を示します。

```
# zpool split -R /tank2 tank tank2
```

`zpool split -R` オプションを使用せずに新しいプールのインポートを試みたときにマウントポイントの競合を確認した場合は、`-R` オプションを使用して新しいプールをインポートしてください。新しいプールを別のシステムに作成する場合は、マウントポイントの競合が発生しないかぎり、代替ルートディレクトリの指定は不要です。

`zpool split` 機能を使用する前に、次の考慮事項を確認してください。

- RAID-Z 構成または複数のディスクから成る非冗長プールに対しては、この機能を使用できません。
- `zpool split` 操作を試みる前に、データおよびアプリケーションの操作を終了しておいてください。
- ディスクのフラッシュ書き込みキャッシュコマンドを無視するのではなく優先するようにディスクを設定しておくことは重要です。
- 再同期化が進行中の場合、プールを分割できません。
- ミラー化プールの分割は、プールが 2 台か 3 台のディスクで構成されているときに行うのが最適です。このとき、元のプール内の最後のディスクが新しく作成されるプールで使用されます。その後、`zpool attach` コマンドを使用して元のミラー化ストレージプールを再作成するか、または新しく作成したプールをミラー化ストレージプールに変換することができます。この機能を使用して「既存の」ミラー化プールから「新しい」ミラー化プールを作成する方法は現時点で存在しません。
- 既存のプールが 3 方向ミラーの場合、分割操作後に新しいプールに含まれるディスクは 1 台です。既存のプールが 2 台のディスクから成る 2 方向ミラーの場合の結果は、2 台のディスクから成る 2 つの非冗長プールになります。2 台の追加ディスクを接続して、非冗長プールをミラー化プールに変換することが必要になります。
- 分割操作中にデータの冗長性を維持するためのよい方法は、3 台のディスクで構成されるミラー化ストレージプールを分割し、分割操作後に元のプールが 2 台のミラー化ディスクで構成されるようにすることです。

例 4–7 ミラー化された ZFS プールを分割する

次の例では、c1t0d0、c1t2d0、c1t3d0 の 3 台のディスクから成る、trinity という名前のミラー化ストレージプールを分割します。分割後の 2 つのプールは、ディスク c1t0d0 と c1t2d0 から成るミラー化プール trinity と、ディスク c1t3d0 から成る新しいプール neo になります。各プールの内容は同じです。

```
# zpool status trinity
  pool: trinity
 state: ONLINE
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        trinity     ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c1t0d0  ONLINE       0     0     0
            c1t2d0  ONLINE       0     0     0
            c1t3d0  ONLINE       0     0     0

errors: No known data errors
# zpool split trinity neo
# zpool import neo
# zpool status trinity neo
  pool: neo
 state: ONLINE
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        neo         ONLINE       0     0     0
          c1t3d0    ONLINE       0     0     0

errors: No known data errors

  pool: trinity
 state: ONLINE
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        trinity     ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c1t0d0  ONLINE       0     0     0
            c1t2d0  ONLINE       0     0     0

errors: No known data errors
```

ストレージプール内のデバイスをオンラインまたはオフラインにする

ZFS では、デバイスを個別にオフラインまたはオンラインにできます。ハードウェアが信頼できない場合または正しく機能しない場合でも、ZFS ではその状態を一時的な状態と見なして、デバイスからのデータの読み取りまたはデバイスへのデータの書き込みを続行します。一時的な状態でない場合には、デバイスをオフラインにして無視されるように設定できます。オフラインのデバイスには、要求はまったく送信されません。

注 – デバイスを置き換えるときに、オフラインにする必要はありません。

一時的にストレージを切断する必要がある場合は、`zpool offline` コマンドを使用できます。たとえば、ファイバチャネルスイッチのセットからアレイを物理的に切断し、そのアレイを別のセットに接続する必要がある場合は、ZFS ストレージプールで使用されているアレイから LUN をオフラインにできます。新しいスイッチのセットでアレイが再接続されて使用できるようになると、続いて同じ LUN をオンラインにできます。LUN がオフラインの間にストレージプールに追加されたデータは、オンラインに戻されたあとの LUN と再同期化されます。

このシナリオが可能になるのは、新しいスイッチにストレージが接続されたあとに、該当のシステムが場合によっては以前と異なるコントローラを介してそのストレージを検出でき、使用しているプールが RAID-Z 構成またはミラー構成として設定されている場合です。

デバイスをオフラインにする

`zpool offline` コマンドを使用して、デバイスをオフラインにできます。デバイスがディスクの場合は、パスまたは短縮名を使って指定できます。次に例を示します。

```
# zpool offline tank c1t0d0
bringing device c1t0d0 offline
```

デバイスをオフラインにするときは、次の点を考慮します。

- プールをオフラインにすることはできません。エラーになります。たとえば、`raidz1` 構成の 2 つのデバイスをオフラインにしたり、最上位レベルの仮想デバイスをオフラインにしたりすることはできません。

  ```
  # zpool offline tank c1t0d0
  cannot offline c1t0d0: no valid replicas
  ```
- デフォルトでは、`OFFLINE` 状態は持続的です。システムを再起動しても、デバイスはオフラインのままです。

デバイスを一時的にオフラインにするには、`zpool offline -t` オプションを使用します。次に例を示します。

```
# zpool offline -t tank c1t0d0
 bringing device 'c1t0d0' offline
```

システムを再起動すると、このデバイスは自動的に `ONLINE` 状態に戻ります。

- デバイスはオフラインになるとき、ストレージプールから切り離されません。オフラインのデバイスを別のプールで使用しようとすると、元のプールが破棄されたあとであっても、次のようなメッセージが表示されます。

  device `is part of exported or potentially active ZFS` *pool*`. Please see zpool(1M)`

  元のストレージプールを破棄したあとで、オフラインのデバイスを別のストレージプールで使用する場合は、まずデバイスをオンラインに戻してから、元のストレージプールを破棄します。

  元のストレージプールを破棄しないで、デバイスを別のストレージプールから使用する場合は、元のストレージプールにある既存のデバイスを別の類似したデバイスに置き換える方法もあります。デバイスを置き換える方法については、96 ページの「ストレージプール内のデバイスを置き換える」を参照してください。

オフラインのデバイスは、プール状態のクエリー検索を行うと、`OFFLINE` 状態として表示されます。プール状態のクエリー検索については、107 ページの「ZFS ストレージプールの状態のクエリー検索を行う」を参照してください。

デバイスの健全性の詳細については、113 ページの「ZFS ストレージプールの健全性状態を調べる」を参照してください。

デバイスをオンラインにする

デバイスをオフラインにしたあとで、`zpool online` コマンドを使ってデバイスをオンラインに戻すことができます。次に例を示します。

```
# zpool online tank c1t0d0
bringing device c1t0d0 online
```

デバイスがオンラインになると、プールに書き込まれたすべてのデータは、新しく使用可能になったデバイスと再同期化されます。デバイスをオンラインにする操作を利用して、ディスクを置き換えることはできません。デバイスをオフラインにしてから、そのデバイスを置き換えてオンラインにしようとしても、エラー状態のままです。

エラー状態のデバイスをオンラインにしようとすると、次のようなメッセージが表示されます。

```
# zpool online tank c1t0d0
warning: device 'c1t0d0' onlined, but remains in faulted state
```

```
use 'zpool replace' to replace devices that are no longer present
```

ディスクのエラーに関するメッセージがコンソールに表示されるか、/var/adm/messages ファイルに書き込まれる場合もあります。次に例を示します。

```
SUNW-MSG-ID: ZFS-8000-D3, TYPE: Fault, VER: 1, SEVERITY: Major
EVENT-TIME: Wed Jun 30 14:53:39 MDT 2010
PLATFORM: SUNW,Sun-Fire-880, CSN: -, HOSTNAME: neo
SOURCE: zfs-diagnosis, REV: 1.0
EVENT-ID: 504a1188-b270-4ab0-af4e-8a77680576b8
DESC: A ZFS device failed.  Refer to http://sun.com/msg/ZFS-8000-D3 for more information.
AUTO-RESPONSE: No automated response will occur.
IMPACT: Fault tolerance of the pool may be compromised.
REC-ACTION: Run 'zpool status -x' and replace the bad device.
```

エラー状態のデバイスを置き換える方法については、307 ページの「見つからないデバイスに関する問題を解決する」を参照してください。

zpool online -e コマンドを使用して LUN を拡張できます。デフォルトでは、プールに追加された LUN は、プールの autoexpand プロパティーが有効でない場合はその最大サイズにまで拡張されません。zpool online -e コマンドを使用すると、LUN がその時点でオンラインかオフラインかに関係なく、LUN を自動的に拡張できます。次に例を示します。

```
# zpool online -e tank c1t13d0
```

ストレージプールデバイスのエラーをクリアーする

障害のためにデバイスがオフラインになり、エラーが zpool status の出力に表示される場合は、zpool clear コマンドを使ってエラー数をクリアーできます。

引数を指定しないでこのコマンドを実行した場合は、プールに含まれるすべてのデバイスのエラーがクリアーされます。次に例を示します。

```
# zpool clear tank
```

1 つ以上のデバイスを指定してこのコマンドを実行した場合は、指定したデバイスに関連付けられたエラーだけがクリアーされます。次に例を示します。

```
# zpool clear tank c1t0d0
```

zpool エラーのクリアーについては、311 ページの「一時的なエラーを解消する」を参照してください。

ストレージプール内のデバイスを置き換える

`zpool replace` コマンドを使用して、ストレージプール内のデバイスを置き換えることができます。

冗長プール内の同じ場所にある別のデバイスでデバイスを物理的に置き換える場合は、置き換えられるデバイスを特定するだけで済むことがあります。ZFS はそのデバイスを、同じハードウェア上の同じ場所にある別のディスクであると認識します。たとえば、障害の発生したディスク (`c1t1d0`) を置き換える場合、そのディスクを取り除き、同じ場所でそれを置き換えるには、次の構文を使用します。

```
# zpool replace tank c1t1d0
```

ストレージプール内のデバイスを、物理的に異なる場所にあるディスクに置き換えようとしている場合は、両方のデバイスを指定する必要があります。次に例を示します。

```
# zpool replace tank c1t1d0 c1t2d0
```

ZFS ルートプールのディスクを置き換える場合は、178 ページの「ZFS ルートプールのディスクを置き換える方法」を参照してください。

ディスクを置き換えるための基本的な手順は次のとおりです。

- 必要に応じて、`zpool offline` コマンドでディスクをオフラインにします。
- 置き換えるディスクを取り外します。
- 交換用ディスクを挿入します。
- `zpool replace` コマンドを実行します。次に例を示します。

  ```
  # zpool replace tank c1t1d0
  ```
- `zpool online` コマンドでディスクをオンラインに戻します。

Sun Fire x4500 などの一部のシステムでは、ディスクをオフラインにする前に構成解除する必要があります。このシステム上の同じスロット位置にあるディスクを置き換えようとしている場合は、この節の最初の例で説明したように `zpool replace` コマンドを実行するだけで置き換えを実行できます。

Sun Fire X4500 システムでディスクを置き換える例については、例 11–1 を参照してください。

ZFS ストレージプール内のデバイスを置き換えるときは、次のことを考慮します。

- プールの autoreplace プロパティーをオンに設定した場合、そのプールに以前属していたデバイスと物理的に同じ位置に新しいデバイスが検出されると、そのデバイスが自動的にフォーマットされて置き換えられます。このプロパティーが有効なときは、zpool replace コマンドを使用する必要はありません。ハードウェアの種類によっては、この機能を使用できない場合があります。
- 交換用デバイスの容量が、ミラー化構成または RAID-Z 構成内でもっとも容量の小さいディスク以上である必要があります。
- 置き換える前のデバイスよりもサイズが大きい交換デバイスをプールに追加しても、プールは自動的にその最大サイズにまで拡張されません。プールの autoexpand プロパティーの値は、ディスクをプールに追加したときに、置き換え後の LUN がその最大サイズにまで拡張されるかどうかを決定します。デフォルトでは、autoexpand プロパティーは無効になっています。容量の大きい LUN をプールに追加する前後どちらでも、このプロパティーを有効にすることで LUN のサイズを拡張できます。

  次の例では、ミラー化プール内の 16G バイトのディスク 2 台を 72G バイトのディスク 2 台で置き換えます。ディスクの置き換え後に autoexpand プロパティーを有効にして、LUN をその最大サイズにまで拡張します。

```
# zpool create pool mirror c1t16d0 c1t17d0
# zpool status
  pool: pool
 state: ONLINE
 scrub: none requested
config:

        NAME         STATE     READ WRITE CKSUM
        pool         ONLINE       0     0     0
          mirror     ONLINE       0     0     0
            c1t16d0  ONLINE       0     0     0
            c1t17d0  ONLINE       0     0     0

zpool list pool
NAME   SIZE   ALLOC  FREE    CAP  HEALTH  ALTROOT
pool  16.8G  76.5K  16.7G     0%  ONLINE  -
# zpool replace pool c1t16d0 c1t1d0
# zpool replace pool c1t17d0 c1t2d0
# zpool list pool
NAME   SIZE   ALLOC  FREE    CAP  HEALTH  ALTROOT
pool  16.8G  88.5K  16.7G     0%  ONLINE  -
# zpool set autoexpand=on pool
# zpool list pool
NAME   SIZE   ALLOC  FREE    CAP  HEALTH  ALTROOT
pool  68.2G   117K  68.2G     0%  ONLINE  -
```

- 大規模プール内で多数のディスクを置き換える場合は、新しいディスク上にデータを再同期化するために時間がかかります。また、ディスクの置き換えの合間に zpool scrub コマンドを実行して、置き換えたあとのデバイスが動作可能なこと、およびデータが正しく書き込まれることを確認することもできます。

- 障害の発生したディスクがホットスペアに自動的に置き換えられた場合は、障害の発生したディスクが置き換えられたあとでスペアの切り離しが必要になることがあります。ホットスペアの切り離しについては、100 ページの「ストレージプール内のホットスペアをアクティブにする/非アクティブにする」を参照してください。

デバイスの置き換えの詳細については、307 ページの「見つからないデバイスに関する問題を解決する」および309 ページの「破損したデバイスを交換または修復する」を参照してください。

ストレージプールにホットスペアを指定する

ホットスペア機能を使って、1 つ以上のストレージプールで障害が発生したデバイスまたはエラー状態のデバイスを置き換えるために使用するディスクを指定できます。「ホットスペア」として指定したデバイスはプール内ではアクティブデバイスではありませんが、プールのアクティブデバイスで障害が発生した場合には、そのデバイスがホットスペアに自動的に置き換えられます。

次の方法を使って、デバイスをホットスペアとして指定できます。

- プール作成時に `zpool create` コマンドを使用する.
- プール作成後に `zpool add` コマンドを使用する.

次の例は、プールの作成時にデバイスをホットスペアとして指定する方法を示しています。

```
# zpool create trinity mirror c1t1d0 c2t1d0 spare c1t2d0 c2t2d0
# zpool status trinity
  pool: trinity
 state: ONLINE
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        trinity     ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c1t1d0  ONLINE       0     0     0
            c2t1d0  ONLINE       0     0     0
        spares
          c1t2d0    AVAIL
          c2t2d0    AVAIL

errors: No known data errors
```

次の例は、プールの作成後にプールに追加することによってホットスペアを指定する方法を示しています。

```
# zpool add neo spare c5t3d0 c6t3d0
# zpool status neo
  pool: neo
 state: ONLINE
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        neo         ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c3t3d0  ONLINE       0     0     0
            c4t3d0  ONLINE       0     0     0
        spares
          c5t3d0    AVAIL
          c6t3d0    AVAIL

errors: No known data errors
```

ホットスペアをストレージプールから削除するときは、`zpool remove` コマンドを使用します。次に例を示します。

```
# zpool remove zeepool c2t3d0
# zpool status zeepool
  pool: zeepool
 state: ONLINE
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        zeepool     ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c1t1d0  ONLINE       0     0     0
            c2t1d0  ONLINE       0     0     0
        spares
          c1t3d0    AVAIL

errors: No known data errors
```

ストレージプールが現在使用しているホットスペアは、削除できません。

ZFS ホットスペアを使用するときは、次の点を考慮してください。

- 現時点では、`zpool remove` コマンドはホットスペア、キャッシュデバイス、およびログデバイスを削除するときにのみ使用できます。
- ディスクをホットスペアとして追加するには、ホットスペアの容量が、プール内でもっとも容量の大きいディスク以上である必要があります。小さなディスクをスペアとしてプールに追加することも許可されています。ただし、小さなスペアディスクがアクティブになると、自動的または `zpool replace` コマンドにより、次のようなエラーで操作が失敗します。

  `cannot replace` *disk3* `with` *disk4*`: device is too small`

ストレージプール内のホットスペアをアクティブにする/非アクティブにする

ホットスペアをアクティブにするには、次のようにします。

- 手動で置き換える – `zpool replace` コマンドを使用して、ストレージプール内で障害の発生したデバイスをホットスペアで置き換えます。
- 自動的に置き換える – FMA エージェントは、エラー状態を検出すると、プールを検査して使用可能なホットスペアがあるかどうかを調べます。ある場合は、障害の発生したデバイスを使用可能なスペアに置き換えます。

  現在使用しているホットスペアで障害が発生した場合、FMA エージェントはそのスペアを切り離し、置き換えた状態を取り消します。続いてエージェントは、別のホットスペアが使用可能であれば、そのスペアを使ってデバイスを置き換えようとします。現時点では、デバイスがシステムから見えなくならないと ZFS 診断エンジンがエラー状態を生成しないので、この機能もその事実に制限されます。

  障害の発生したデバイスにアクティブなスペアがある場合にデバイスを物理的に交換するときは、`zpool detach` コマンドを使用して元のデバイスを再度アクティブにして、スペアを切り離すことができます。プールの `autoreplace` プロパティーをオンに設定した場合は、新しいデバイスが挿入されオンライン処理が完了すると、スペアは自動的に切り離されてスペアプールに戻されます。

`zpool replace` コマンドを使用して、デバイスをホットスペアに手動で置き換えることができます。例 4–8 を参照してください。

ホットスペアが使用可能な場合、エラー状態のデバイスは自動的に置き換えられます。次に例を示します。

```
# zpool status -x
  pool: zeepool
 state: DEGRADED
status: One or more devices could not be opened.  Sufficient replicas exist for
        the pool to continue functioning in a degraded state.
action: Attach the missing device and online it using 'zpool online'.
   see: http://www.sun.com/msg/ZFS-8000-2Q
 scrub: resilver completed after 0h0m with 0 errors on Mon Jan 11 10:20:35 2010
config:

        NAME          STATE     READ WRITE CKSUM
        zeepool       DEGRADED     0     0     0
          mirror-0    DEGRADED     0     0     0
            c1t2d0    ONLINE       0     0     0
            spare-1   DEGRADED     0     0     0
              c2t1d0  UNAVAIL      0     0     0  cannot open
              c2t3d0  ONLINE       0     0     0  88.5K resilvered
        spares
          c2t3d0      INUSE     currently in use

errors: No known data errors
```

現時点では、次の方法でホットスペアを非アクティブにできます。

- ストレージプールからホットスペアを削除する。
- 障害の発生したディスクを物理的に置き換えたあとでホットスペアを切り離す。例 4–9 を参照してください。
- ホットスペア内で一時的または永続的に交換を行う。例 4–10 を参照してください。

例 4–8 ディスクを手動でホットスペアと置き換える

この例では、`zpool replace` コマンドを使用して、ディスク `c2t1d0` をホットスペア `c2t3d0` と置き換えます。

```
# zpool replace zeepool c2t1d0 c2t3d0
# zpool status zeepool
  pool: zeepool
 state: ONLINE
 scrub: resilver completed after 0h0m with 0 errors on Wed Jan 20 10:00:50 2010
config:

        NAME          STATE     READ WRITE CKSUM
        zeepool       ONLINE       0     0     0
          mirror-0    ONLINE       0     0     0
            c1t2d0    ONLINE       0     0     0
            spare-1   ONLINE       0     0     0
              c2t1d0  ONLINE       0     0     0
              c2t3d0  ONLINE       0     0     0  90K resilvered
        spares
          c2t3d0      INUSE     currently in use

errors: No known data errors
```

その後、ディスク `c2t1d0` を切り離します。

```
# zpool detach zeepool c2t1d0
# zpool status zeepool
  pool: zeepool
 state: ONLINE
 scrub: resilver completed after 0h0m with 0 errors on Wed Jan 20 10:00:50 2010
config:

        NAME        STATE     READ WRITE CKSUM
        zeepool     ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c1t2d0  ONLINE       0     0     0
            c2t3d0  ONLINE       0     0     0  90K resilvered

errors: No known data errors
```

例 4–9 障害が発生したディスクの置き換え後にホットスペアを切り離す

次の例では、障害が発生したディスク (`c2t1d0`) を物理的に置き換えて、`zpool replace` コマンドを使って ZFS に通知します。

例 4–9 障害が発生したディスクの置き換え後にホットスペアを切り離す (続き)

```
# zpool replace zeepool c2t1d0
# zpool status zeepool
  pool: zeepool
 state: ONLINE
 scrub: resilver completed after 0h0m with 0 errors on Wed Jan 20 10:08:44 2010
config:

        NAME          STATE     READ WRITE CKSUM
        zeepool       ONLINE       0     0     0
          mirror-0    ONLINE       0     0     0
            c1t2d0    ONLINE       0     0     0
            spare-1   ONLINE       0     0     0
              c2t3d0  ONLINE       0     0     0  90K resilvered
              c2t1d0  ONLINE       0     0     0
        spares
          c2t3d0      INUSE     currently in use

errors: No known data errors
```

その後、zpool detach コマンドを使ってホットスペアをスペアプールに戻すことができます。次に例を示します。

```
# zpool detach zeepool c2t3d0
# zpool status zeepool
  pool: zeepool
 state: ONLINE
 scrub: resilver completed with 0 errors on Wed Jan 20 10:08:44 2010
config:

        NAME              STATE     READ WRITE CKSUM
        zeepool           ONLINE       0     0     0
          mirror          ONLINE       0     0     0
            c1t2d0        ONLINE       0     0     0
            c2t1d0        ONLINE       0     0     0
        spares
          c2t3d0          AVAIL

errors: No known data errors
```

例 4–10 障害が発生したディスクを切り離してホットスペアを使用する

障害が発生したディスクを、そのディスクを現在置き換えようとしているホットスペア内で一時的または永続的に交換することによって置き換えたい場合は、元の (障害が発生した) ディスクを切り離します。障害が発生したディスクが最終的に置き換えられたら、そのディスクをスペアとしてストレージプールに再び追加できます。次に例を示します。

```
# zpool status zeepool
  pool: zeepool
 state: DEGRADED
status: One or more devices could not be opened.  Sufficient replicas exist for
        the pool to continue functioning in a degraded state.
```

例 4–10 障害が発生したディスクを切り離してホットスペアを使用する (続き)

```
action: Attach the missing device and online it using 'zpool online'.
   see: http://www.sun.com/msg/ZFS-8000-2Q
 scrub: resilver in progress for 0h0m, 70.47% done, 0h0m to go
config:

        NAME          STATE     READ WRITE CKSUM
        zeepool       DEGRADED     0     0     0
          mirror-0    DEGRADED     0     0     0
            c1t2d0    ONLINE       0     0     0
            spare-1   DEGRADED     0     0     0
              c2t1d0  UNAVAIL      0     0     0  cannot open
              c2t3d0  ONLINE       0     0     0  70.5M resilvered
        spares
          c2t3d0      INUSE     currently in use

errors: No known data errors
# zpool detach zeepool c2t1d0
# zpool status zeepool
  pool: zeepool
 state: ONLINE
 scrub: resilver completed after 0h0m with 0 errors on Wed Jan 20 13:46:46 2010
config:

        NAME        STATE     READ WRITE CKSUM
        zeepool     ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c1t2d0  ONLINE       0     0     0
            c2t3d0  ONLINE       0     0     0  70.5M resilvered

errors: No known data errors
(Original failed disk c2t1d0 is physically replaced)
# zpool add zeepool spare c2t1d0
# zpool status zeepool
  pool: zeepool
 state: ONLINE
 scrub: resilver completed after 0h0m with 0 errors on Wed Jan 20 13:48:46 2010
config:

        NAME        STATE     READ WRITE CKSUM
        zeepool     ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c1t2d0  ONLINE       0     0     0
            c2t3d0  ONLINE       0     0     0  70.5M resilvered
        spares
          c2t1d0    AVAIL

errors: No known data errors
```

ZFS ストレージプールのプロパティーの管理

zpool get コマンドを使用して、プールのプロパティーの情報を表示できます。次に例を示します。

```
# zpool get all mpool
NAME  PROPERTY       VALUE       SOURCE
pool  size           68G         -
pool  capacity       0%          -
pool  altroot        -           default
pool  health         ONLINE      -
pool  guid           601891032394735745  default
pool  version        22          default
pool  bootfs         -           default
pool  delegation     on          default
pool  autoreplace    off         default
pool  cachefile      -           default
pool  failmode       wait        default
pool  listsnapshots  on          default
pool  autoexpand     off         default
pool  free           68.0G       -
pool  allocated      76.5K       -
```

ストレージプールのプロパティーは zpool set コマンドで設定できます。次に例を示します。

```
# zpool set autoreplace=on mpool
# zpool get autoreplace mpool
NAME  PROPERTY     VALUE    SOURCE
mpool autoreplace  on       default
```

表 4–1 ZFS プールのプロパティーの説明

プロパティー名	種類	デフォルト値	説明
allocated	文字列	なし	読み取り専用の値。物理的に割り当て済みであるプール内のストレージ領域の容量を示します。
altroot	文字列	off	代替ルートディレクトリを示します。設定されている場合、プール内のすべてのマウントポイントの先頭にこのディレクトリが付加されます。このプロパティーは、不明なプールを調べるときやマウントポイントが信頼できない場合、または通常のパスが有効でない代替ブート環境で使用できます。
autoreplace	ブール型	off	自動デバイス交換を制御します。オフに設定されている場合、zpool replace コマンドを使ってデバイス交換を開始する必要があります。オンに設定されている場合、そのプールに以前属していたデバイスと物理的に同じ位置にある新しいデバイスは、いずれも自動的にフォーマットされ、置き換えられます。このプロパティーの省略名は replace です。

表 4–1 ZFS プールのプロパティーの説明 (続き)

プロパティー名	種類	デフォルト値	説明
bootfs	ブール型	なし	ルートプールのデフォルトのブート可能データセットを示します。このプロパティーは通常、インストールプログラムとアップグレードプログラムによって設定されます。
cachefile	文字列	なし	プール構成情報がキャッシュされる場所を制御します。システムの起動時に、キャッシュ内のすべてのプールが自動的にインポートされます。ただし、インストール環境とクラスタ化環境では、プールが自動的にインポートされないようにするために、この情報を別の場所にキャッシュすることが必要になる場合もあります。プール構成情報を別の場所にキャッシュするようにこのプロパティーを設定できます。この情報は、あとから zpool import - c コマンドを使ってインポートできます。ほとんどの ZFS 構成では、このプロパティーは使用されません。
capacity	数値	なし	読み取り専用の値。使用されているプール領域の割合を示します。 このプロパティーの省略名は cap です。
委託	ブール型	on	データセットに定義されているアクセス権を非特権ユーザーに付与できるかどうかを制御します。詳細は、第 9 章「Oracle Solaris ZFS 委任管理」を参照してください。

表 4–1 ZFS プールのプロパティーの説明 (続き)

プロパティー名	種類	デフォルト値	説明
failmode	文字列	wait	プールに壊滅的な障害が発生した場合のシステムの動作を制御します。通常は、配下の 1 台以上のストレージデバイスへの接続が失われた場合や、プール内のすべてのデバイスに障害が発生した場合に、このような状況になります。そのような状況での動作は、次のいずれかの値によって決定されます。 ■ wait – デバイスへの接続を復元し、zpool clear コマンドでエラーをクリアーするまで、プールに対するすべての入出力要求をブロックします。この状態では、プールに対する入出力操作はブロックされますが、読み取り操作は成功する場合があります。デバイスの問題が解決されるまで、プールの状態は wait のままです。 ■ continue – 新しい書き込み入出力要求には EIO エラーを返しますが、正常な残りのデバイスに対する読み取りは許可します。まだディスクにコミットされていない書き込み要求はブロックされます。デバイスを再接続するか交換したあと、zpool clear コマンドでエラーを解決する必要があります。 ■ panic – コンソールにメッセージを出力し、システムクラッシュダンプを生成します。
free	文字列	なし	読み取り専用の値。まだ割り当てられていないプール内のブロック数を示します。
guid	文字列	なし	読み取り専用プロパティー。プールの一意の識別子を示します。
health	文字列	なし	読み取り専用プロパティー。プールの現在の健全性を ONLINE、DEGRADED、FAULTED、OFFLINE、REMOVED、または UNAVAIL のいずれかで示します。
listsnapshots	文字列	on	このプールに関連付けられているスナップショット情報が zfs list コマンドによって表示されるようにするかどうかを制御します。このプロパティーが無効な場合、zfs list -t snapshot コマンドを使用すればスナップショット情報を表示できます。
size	数値	なし	読み取り専用プロパティー。ストレージプールの合計サイズを示します。

表 4–1 ZFS プールのプロパティーの説明 (続き)

プロパティー名	種類	デフォルト値	説明
version	数値	なし	プールの現在のディスク上バージョンを示します。プールを更新する方法としては zpool upgrade コマンドをお勧めしますが、下位互換性のために特定のバージョンが必要な場合には、このプロパティーを使用できます。このプロパティーには、1 から zpool upgrade -v コマンドで報告される現在のバージョンまでの任意の数値を設定できます。

ZFS ストレージプールの状態のクエリー検索を行う

zpool list コマンドでは、いくつかの方法でプール状態に関する情報を要求できます。主に 3 つのカテゴリの情報を要求できます。 基本的な使用状況の情報、入出力統計、および健全性状態です。ここでは、3 種類のストレージプール情報のすべてについて説明します。

- 107 ページの「ZFS ストレージプールについての情報を表示する」
- 110 ページの「ZFS ストレージプールの入出力統計を表示する 」
- 113 ページの「ZFS ストレージプールの健全性状態を調べる」

ZFS ストレージプールについての情報を表示する

zpool list コマンドを使用して、プールに関する基本的な情報を表示できます。

すべてのストレージプールまたは特定のプールについての情報を一覧表示する

引数を指定しないで zpool list コマンドを実行すると、システム上のすべてのプールについて次の情報が表示されます。

```
# zpool list
NAME                    SIZE    ALLOC   FREE    CAP  HEALTH     ALTROOT
tank                   80.0G   22.3G   47.7G    28%  ONLINE     -
dozer                   1.2T    384G    816G    32%  ONLINE     -
```

このコマンド出力には、次の情報が表示されます。

NAME　プールの名前。

SIZE　プールの合計サイズ。最上位レベルにあるすべての仮想デバイスの合計サイズになります。

ALLOC　すべてのデータセットおよび内部メタデータに割り当てられた物理的容量。この容量は、ファイルシステムレベルで報告されるディスク容量とは異なります。

使用可能なファイルシステムの容量を確認する方法については、62 ページの「ZFS のディスク領域の計上」を参照してください。

FREE　プール内で割り当てられていない容量。

CAP (CAPACITY)　使用されているディスク容量。総ディスク容量に対するパーセントで表現されます。

HEALTH　プールの現在の健全性状態。

プールの健全性の詳細については、113 ページの「ZFS ストレージプールの健全性状態を調べる」を参照してください。

ALTROOT　プールの代替ルート (存在する場合)。

代替ルートプールの詳細については、294 ページの「ZFS 代替ルートプールを使用する」を参照してください。

プール名を指定して、特定のプールの統計を収集することもできます。次に例を示します。

```
# zpool list tank
NAME                    SIZE    ALLOC   FREE    CAP   HEALTH     ALTROOT
tank                   80.0G    22.3G   47.7G    28%  ONLINE     -
```

特定のストレージプールの統計を表示する

-o オプションを使用して、特定の統計を要求することができます。このオプションを使用して、カスタムレポートを出力したり、必要な情報をすばやく表示したりできます。たとえば、各プールの名前とサイズだけを表示する場合は、次の構文を使用します。

```
# zpool list -o name,size
NAME                     SIZE
tank                    80.0G
dozer                    1.2T
```

列の名前は、107 ページの「すべてのストレージプールまたは特定のプールについての情報を一覧表示する」で説明したプロパティーに対応しています。

ZFS ストレージプールの出力をスクリプトで使えるようにする

zpool list コマンドのデフォルト出力は、読みやすいように設計されているため、シェルスクリプトの一部として使いやすい状態ではありません。このコマンドをプログラムで使いやすくするために、-H オプションを使用して、列見出しを非表示にし、空白文字の代わりにタブでフィールドを区切ることができます。たとえば、システム上のすべてのプール名をリストとして要求するときは、次の構文を使用します。

```
# zpool list -Ho name
tank
dozer
```

別の例です。

```
# zpool list -H -o name,size
tank   80.0G
dozer  1.2T
```

ZFS ストレージプールのコマンド履歴を表示する

ZFS は、プールの状態に関する情報を変更する `zfs` コマンドと `zpool` コマンドが正常に実行された場合にだけ自動的にログを記録します。この情報は、`zpool history` コマンドを使用して表示することができます。

例えば、ルートプールに関するコマンド出力を表示する場合は、次の構文を使用します。

```
# zpool history
History for 'rpool':
2010-05-11.10:18:54 zpool create -f -o failmode=continue -R /a -m legacy -o
cachefile=/tmp/root/etc/zfs/zpool.cache rpool mirror c1t0d0s0 c1t1d0s0
2010-05-11.10:18:55 zfs set canmount=noauto rpool
2010-05-11.10:18:55 zfs set mountpoint=/rpool rpool
2010-05-11.10:18:56 zfs create -o mountpoint=legacy rpool/ROOT
2010-05-11.10:18:57 zfs create -b 8192 -V 2048m rpool/swap
2010-05-11.10:18:58 zfs create -b 131072 -V 1536m rpool/dump
2010-05-11.10:19:01 zfs create -o canmount=noauto rpool/ROOT/zfsBE
2010-05-11.10:19:02 zpool set bootfs=rpool/ROOT/zfsBE rpool
2010-05-11.10:19:02 zfs set mountpoint=/ rpool/ROOT/zfsBE
2010-05-11.10:19:03 zfs set canmount=on rpool
2010-05-11.10:19:04 zfs create -o mountpoint=/export rpool/export
2010-05-11.10:19:05 zfs create rpool/export/home
2010-05-11.11:11:10 zpool set bootfs=rpool rpool
2010-05-11.11:11:10 zpool set bootfs=rpool/ROOT/zfsBE rpool
```

システムでこれと同じような出力を利用して、エラー状況のトラブルシューティングのために実行された「実際の」ZFS コマンドセットを特定することができます。

履歴ログの特徴を次に示します。

- ログを無効にすることはできません。
- ログは永続的にディスクに保存されます。つまり、ログはシステムの再起動後も保持されます。
- ログはリングバッファーとして実装されます。最小サイズは 128K バイトです。最大サイズは 32M バイトです。
- 小さめのプールの場合、最大サイズはプールサイズの 1% を上限とします。このサイズはプールの作成時に自動的に決定されます。

- ログの管理は不要です。つまり、ログのサイズを調整したり、ログの場所を変更したりする必要はありません。

特定のストレージプールのコマンド履歴を確認するには、次のような構文を使用します。

```
# zpool history tank
History for 'tank':
2010-05-13.14:13:15 zpool create tank mirror c1t2d0 c1t3d0
2010-05-13.14:21:19 zfs create tank/snaps
2010-05-14.08:10:29 zfs create tank/ws01
2010-05-14.08:10:54 zfs snapshot tank/ws01@now
2010-05-14.08:11:05 zfs clone tank/ws01@now tank/ws01bugfix
```

-l オプションを使用して、ユーザー名、ホスト名、および操作が実行されたゾーンを含む長形式を表示します。次に例を示します。

```
# zpool history -l tank
History for 'tank':
2010-05-13.14:13:15 zpool create tank mirror c1t2d0 c1t3d0 [user root on neo]
2010-05-13.14:21:19 zfs create tank/snaps [user root on neo]
2010-05-14.08:10:29 zfs create tank/ws01 [user root on neo]
2010-05-14.08:10:54 zfs snapshot tank/ws01@now [user root on neo]
2010-05-14.08:11:05 zfs clone tank/ws01@now tank/ws01bugfix [user root on neo]
```

-i オプションを使用して、診断に利用できる内部イベント情報を表示します。次に例を示します。

```
# zpool history -i tank
2010-05-13.14:13:15 zpool create -f tank mirror c1t2d0 c1t23d0
2010-05-13.14:13:45 [internal pool create txg:6] pool spa 19; zfs spa 19; zpl 4;...
2010-05-13.14:21:19 zfs create tank/snaps
2010-05-13.14:22:02 [internal replay_inc_sync txg:20451] dataset = 41
2010-05-13.14:25:25 [internal snapshot txg:20480] dataset = 52
2010-05-13.14:25:25 [internal destroy_begin_sync txg:20481] dataset = 41
2010-05-13.14:25:26 [internal destroy txg:20488] dataset = 41
2010-05-13.14:25:26 [internal reservation set txg:20488] 0 dataset = 0
2010-05-14.08:10:29 zfs create tank/ws01
2010-05-14.08:10:54 [internal snapshot txg:53992] dataset = 42
2010-05-14.08:10:54 zfs snapshot tank/ws01@now
2010-05-14.08:11:04 [internal create txg:53994] dataset = 58
2010-05-14.08:11:05 zfs clone tank/ws01@now tank/ws01bugfix
```

ZFS ストレージプールの入出力統計を表示する

プールまたは特定の仮想デバイスの入出力統計を要求する場合は、zpool iostat コマンドを使用します。iostat コマンドと同様に、このコマンドでは、発生したすべての入出力アクティビティーの静的なスナップショットと、指定した間隔ごとに更新される統計を表示できます。次の統計が報告されます。

`alloc capacity` プールまたはデバイスに現在格納されているデータの量。この容量は、実装の内部的な詳細のために、実際のファイルシステムで利用できるディスク容量とわずかに異なります。

プール領域とデータセット領域の相違点の詳細については、62 ページの「ZFS のディスク領域の計上」を参照してください。

`free capacity` プールまたはデバイスで使用できるディスク容量。`used` 統計と同様に、この容量はデータセットで使用できるディスク容量と多少異なります。

`read operations` プールまたはデバイスに送信された入出力読み取り操作の数 (メタデータ要求を含む)。

`write operations` プールまたはデバイスに送信された入出力書き込み操作の数。

`read bandwidth` すべての読み取り操作 (メタデータを含む) の帯域幅。単位/秒として表現されます。

`write bandwidth` すべての書き込み操作の帯域幅。単位/秒として表現されます。

プール全体の入出力統計を一覧表示する

オプションを指定しないで `zpool iostat` コマンドを実行すると、システム上のすべてのプールを起動してから累積された統計が表示されます。次に例を示します。

```
# zpool iostat
               capacity     operations    bandwidth
pool        alloc   free   read  write   read  write
----------  -----  -----  -----  -----  -----  -----
rpool       6.05G  61.9G      0      0    786    107
tank        31.3G  36.7G      4      1   296K  86.1K
----------  -----  -----  -----  -----  -----  -----
```

これらの統計は起動してから累積されたものなので、プールのアイドル状態が相対的に多い場合には、帯域幅が低く表示されることがあります。間隔を指定すれば、帯域幅の現在の使用状況をより正確に表示できます。次に例を示します。

```
# zpool iostat tank 2
               capacity     operations    bandwidth
pool        alloc   free   read  write   read  write
----------  -----  -----  -----  -----  -----  -----
tank        18.5G  49.5G      0    187      0  23.3M
tank        18.5G  49.5G      0    464      0  57.7M
tank        18.5G  49.5G      0    457      0  56.6M
tank        18.8G  49.2G      0    435      0  51.3M
```

この例のコマンドでは、`tank` プールの使用状況の統計が 2 秒ごとに表示されます (Ctrl - C キーを押すと停止する)。`count` 引数を追加で指定することもできます。この場合は、コマンドが指定した数だけ繰り返されたあとに終了します。たとえ

ば、zpool iostat 2 3 の場合は、概要が 2 秒ごとに 3 回 (計 6 秒間) 出力されます。プールが 1 つだけの場合は、ひと続きの行に統計が表示されます。複数のプールがある場合は、各プールが分かれて見えるように、各プールの間に点線が挿入されます。

仮想デバイスの入出力統計を一覧表示する

zpool iostat コマンドでは、プール全体の入出力統計だけでなく、仮想デバイスの入出力統計を表示できます。このコマンドを使用して、速度が異常に遅いデバイスを検出することができます。また、ZFS が生成した入出力の分布を監視するといった使い方もできます。仮想デバイス全体のレイアウトおよびすべての入出力統計を要求する場合は、zpool iostat -v コマンドを使用します。次に例を示します。

```
# zpool iostat -v
               capacity     operations    bandwidth
pool        alloc   free   read  write   read  write
----------  -----  -----  -----  -----  -----  -----
rpool       6.05G  61.9G      0      0    785    107
  mirror    6.05G  61.9G      0      0    785    107
    c1t0d0s0    -      -      0      0    578    109
    c1t1d0s0    -      -      0      0    595    109
----------  -----  -----  -----  -----  -----  -----
tank        36.5G  31.5G      4      1   295K   146K
  mirror    36.5G  31.5G    126     45  8.13M  4.01M
    c1t2d0      -      -      0      3   100K   386K
    c1t3d0      -      -      0      3   104K   386K
----------  -----  -----  -----  -----  -----  -----
```

仮想デバイスの入出力統計を表示するときは、2 つの重要な点に注意してください。

- まず、ディスク容量の使用統計は、最上位レベルの仮想デバイスに対してのみ利用できます。ミラーおよび RAID-Z 仮想デバイスにディスク領域がどのように割り当てられるかは、実装に固有なので、1 つの数値として表現するのは簡単ではありません。
- 次に、予期したとおりの正確な数値にならないことがあります。特に、RAID-Z デバイスとミラー化されたデバイスの統計は、正確に一致することがありません。この相違は、プールが作成された直後に、特に顕著になります。プールが作成されるときに大量の入出力がディスクに直接実行されますが、これらがミラーレベルでは計上されないためです。時間の経過とともに、これらの数値はしだいに等しくなります。ただし、故障したデバイス、応答しないデバイス、またはオフラインのデバイスも、この対称性に影響する可能性があります。

仮想デバイスの統計を検査するときにも、同じオプション (間隔とカウント) を使用できます。

ZFS ストレージプールの健全性状態を調べる

ZFS では、プールとデバイスの健全性を検査する方法が統合されています。プールの健全性は、そのすべてのデバイスの状態から判断されます。この状態情報は、`zpool status` コマンドを使って表示されます。また、発生する可能性のあるプールとデバイスの障害も `fmd` によって報告され、システムコンソールに表示されるとともに `/var/adm/messages` ファイルに記録されます。

この節では、プールとデバイスの健全性を確認する方法について説明します。この章では、健全でないプールを修復または回復する方法については説明しません。障害追跡およびデータの回復については、第 11 章「Oracle Solaris ZFS のトラブルシューティングとプールの回復」を参照してください。

各デバイスは、次のいずれかの状態になることができます。

`ONLINE` デバイスまたは仮想デバイスは正常に動作しています。一時的なエラーがいくつか発生している可能性はありますが、それらを除けば正常に動作しています。

`DEGRADED` 仮想デバイスで障害が発生しましたが、デバイスはまだ動作しています。この状態は、ミラーデバイスまたは RAID-Z デバイスを構成するデバイスのうち、1 つ以上のデバイスが失われたときによく発生します。プールの耐障害性が損なわれる可能性があります。別のデバイスで続けて障害が発生した場合には、回復できない状態になることがあります。

`FAULTED` デバイスまたは仮想デバイスへのアクセスが完全にできない状態です。この状態は通常、このデバイスで大きな障害が発生していて、デバイスとの間でデータの送受信ができないことを示しています。最上位レベルの仮想デバイスがこの状態の場合には、そのプールへのアクセスはまったくできません。

`OFFLINE` 管理者がデバイスを明示的にオフラインにしています。

`UNAVAIL` デバイスまたは仮想デバイスを開くことができません。場合によっては、デバイスが `UNAVAIL` であるプールが `DEGRADED` モードで表示されることがあります。最上位レベルの仮想デバイスが `UNAVAIL` の場合は、そのプールのデバイスには一切アクセスできません。

`REMOVED` システムの稼働中にデバイスが物理的に取り外されました。デバイスの取り外しの検出はハードウェアに依存しており、一部のプラットフォームではサポートされていない場合があります。

プールの健全性は、最上位レベルのすべての仮想デバイスから判断されます。すべての仮想デバイスが `ONLINE` の場合は、プールも `ONLINE` になります。仮想デバイスのいずれかが `DEGRADED` または `UNAVAIL` の場合は、プールも `DEGRADED` になります。最上位レベルの仮想デバイスが `FAULTED` または `OFFLINE` の場合は、プールも `FAULTED` にな

ります。FAULTED 状態のプールには一切アクセスできません。必要なデバイスが接続または修復されるまで、データは回復できません。DEGRADED 状態のプールは引き続き動作しますが、プールがオンラインの場合と同じレベルのデータ冗長性やデータスループットを実現できない可能性があります。

ストレージプールの基本的な健全性状態

次のように zpool status コマンドを使用することにより、プールの健全性状態をすばやく確認できます。

```
# zpool status -x
all pools are healthy
```

プール名をコマンド構文に指定すれば、特定のプールを検査できます。ONLINE 状態ではないプールがある場合には、次の節で説明するように、問題が発生していないかどうかを調査するようにしてください。

詳細な健全性状態

-v オプションを使用すれば、より詳細な健全性の概要状態を要求することができます。次に例を示します。

```
# zpool status -v tank
  pool: tank
 state: DEGRADED
status: One or more devices could not be opened.  Sufficient replicas exist for
        the pool to continue functioning in a degraded state.
action: Attach the missing device and online it using 'zpool online'.
   see: http://www.sun.com/msg/ZFS-8000-2Q
 scrub: scrub completed after 0h0m with 0 errors on Wed Jan 20 15:13:59 2010
config:

        NAME        STATE     READ WRITE CKSUM
        tank        DEGRADED     0     0     0
          mirror-0  DEGRADED     0     0     0
            c1t0d0  ONLINE       0     0     0
            c1t1d0  UNAVAIL      0     0     0  cannot open

errors: No known data errors
```

この出力には、プールが現在の状態になった理由が詳細に表示されます。たとえば、問題に関するわかりやすい説明や、詳細な情報を入手するためのナレッジ記事へのリンクが表示されます。ナレッジ記事では、現在の問題から回復するための最良の方法に関する最新情報を提供しています。構成に関する詳細な情報を利用すれば、どのデバイスが損傷しているかや、プールをどのように修復するかを確認できます。

前の例では、エラー状態のデバイスを置き換えるべきです。デバイスを置き換えたあとに、zpool online コマンドを使用してデバイスをオンラインにします。次に例を示します。

```
# zpool online tank c1t0d0
Bringing device c1t0d0 online
# zpool status -x
all pools are healthy
```

autoreplace プロパティーがオンの場合、置き換えたデバイスをオンラインにする必要はない場合があります。

プールにオフラインのデバイスがある場合は、コマンドの出力から問題のプールを確認できます。次に例を示します。

```
# zpool status -x
  pool: tank
 state: DEGRADED
status: One or more devices has been taken offline by the administrator.
        Sufficient replicas exist for the pool to continue functioning in a
        degraded state.
action: Online the device using 'zpool online' or replace the device with
        'zpool replace'.
 scrub: resilver completed after 0h0m with 0 errors on Wed Jan 20 15:15:09 2010
config:

        NAME        STATE     READ WRITE CKSUM
        tank        DEGRADED     0     0     0
          mirror-0  DEGRADED     0     0     0
            c1t0d0  ONLINE       0     0     0
            c1t1d0  OFFLINE      0     0     0  48K resilvered

errors: No known data errors
```

READ 列と WRITE 列には、そのデバイスで発生した入出力エラーの数が表示されます。CKSUM 列には、そのデバイスで発生した訂正不可能なチェックサムエラーの数が表示されます。どちらのエラー数も、デバイス障害が発生する可能性があることを示し、その場合には訂正のための対応がいくつか必要になります。最上位レベルの仮想デバイスでエラー数があると報告された場合、データの一部にアクセスできないことがあります。

errors: フィールドは既知のデータエラーを示します。

前の出力例では、オフラインのデバイスでデータエラーは発生していません。

エラー状態のプールとデータを診断および修復する方法については、第 11 章「Oracle Solaris ZFS のトラブルシューティングとプールの回復」を参照してください。

ZFS ストレージプールを移行する

ストレージプールをシステム間で移動しなければならないことがあります。この作業を行うには、ストレージデバイスを元のシステムから切断して、移動先のシステムに再接続する必要があります。この作業は、ケーブルをデバイスに物理的に接続し直すか、または複数のポートを持つデバイス (SAN 上のデバイスなど) を使用する方法で、行うことができます。ZFS では、アーキテクチャーエンディアンの異なるシステム間でも、一方のマシンのプールをエクスポートして移行先のシステムにインポートできます。異なるストレージプール間 (異なるマシン上にある場合を含む) でファイルシステムを複製または移行する方法については、234 ページの「ZFS データを送信および受信する」を参照してください。

ZFS ストレージプールの移行を準備する

ストレージプールは、移行する準備ができていることを示すために、明示的にエクスポートすることをお勧めします。この操作を行うことで、書き込まれていないデータがすべてディスクにフラッシュされ、データがディスクに書き込まれてエクスポート済みであることが示され、プールに関するすべての情報がシステムから削除されます。

プールを明示的にエクスポートする代わりに、ディスクを手動で取り外した場合でも、そのプールを別のシステムにインポートすることはできます。ただし、最後の数秒間のデータトランザクションが失われる可能性があります。この場合、デバイスが存在しないために、プールが元のシステム上でエラー状態として表示されます。デフォルトでは、明示的にエクスポートしていないプールはインポート先のシステムでインポートできません。アクティブなプールを誤ってインポートしてしまうことを防ぐ (プールを構成するネットワークに接続されたストレージが別のシステムでまだ使用されていることがないようにする) には、この状態が必要になります。

ZFS ストレージプールをエクスポートする

プールをエクスポートするには、`zpool export` コマンドを使用します。次に例を示します。

```
# zpool export tank
```

このコマンドは、プールの中にマウントされたファイルシステムがある場合は、すべてのマウントを解除してから、次の処理を実行しようとします。いずれかのファイルシステムのマウント解除に失敗した場合は、-f オプションを使用して強制的にマウントを解除できます。次に例を示します。

```
# zpool export tank
cannot unmount '/export/home/eschrock': Device busy
# zpool export -f tank
```

このコマンドを実行したあとは、プール tank はシステムから認識されなくなります。

エクスポート時にデバイスが使用できない場合、それらのデバイスは明示的にエクスポートされたものとして識別できません。これらのデバイスのいずれかをあとでシステムに接続した場合には、動作中のデバイスがなくても「潜在的にアクティブ」として表示されます。

ZFS ボリュームがプール内で使用中である場合は、-f オプションを使用してもそのプールをエクスポートすることはできません。ZFS ボリュームが含まれているプールをエクスポートするには、最初にそのボリュームのコンシューマがすべてアクティブでなくなっていることを確認してください。

ZFS ボリュームの詳細については、285 ページの「ZFS ボリューム」を参照してください。

インポートできるストレージプールを判断する

プールをシステムから削除 (明示的にエクスポートするか、デバイスを強制的に取り外す) したあとで、それらのデバイスをインポート先のシステムに接続できます。ZFS では、一部のデバイスだけが利用可能である特定の状況を処理できますが、プールの移行が成功するかどうかはデバイスの全体的な健全性に依存します。また、デバイスは同じデバイス名で接続されている必要はありません。デバイスを移動した場合またはデバイスの名前を変更した場合には、それらが自動的に検出され、構成がそれに合わせて調整されます。インポートできるプールを確認するには、zpool import コマンドをオプションを指定しないで実行します。次に例を示します。

```
# zpool import
 pool: tank
    id: 11809215114195894163
 state: ONLINE
action: The pool can be imported using its name or numeric identifier.
config:

        tank        ONLINE
          mirror-0  ONLINE
```

```
            c1t0d0  ONLINE
            c1t1d0  ONLINE
```

この例では、プール tank をターゲットシステムにインポートできます。各プールは、名前および一意の数値識別子を使って識別されます。同じ名前の複数のプールがインポート可能な場合、数値識別子を使ってプールを区別することができます。

zpool status コマンドの出力と同様に、zpool import の出力にはナレッジ記事へのリンクが含まれています。この記事を参照して、プールのインポートを妨げている問題の修復手順に関する最新情報を入手します。この場合、ユーザーはプールを強制的にインポートできます。ただし、別のシステムがストレージネットワーク経由で使用しているプールをインポートすると、両方のシステムが同じストレージに書き込もうとするため、データの破壊とパニックが発生する可能性があります。プール内の一部のデバイスが使用できないが、使用可能なプールを提供するために十分な冗長データが存在する場合、そのプールは DEGRADED 状態であると表示されます。次に例を示します。

```
# zpool import
  pool: tank
    id: 11809215114195894163
 state: DEGRADED
status: One or more devices are missing from the system.
action: The pool can be imported despite missing or damaged devices.  The
        fault tolerance of the pool may be compromised if imported.
   see: http://www.sun.com/msg/ZFS-8000-2Q
config:

        NAME        STATE     READ WRITE CKSUM
        tank        DEGRADED     0     0     0
          mirror-0  DEGRADED     0     0     0
            c1t0d0  UNAVAIL      0     0     0  cannot open
            c1t3d0  ONLINE       0     0     0
```

この例では、最初のディスクが損傷しているか見つかりません。ただし、ミラー化されたデータにまだアクセスできるため、このプールをインポートすることはできます。エラー状態または見つからないデバイスの数が多すぎる場合、そのプールはインポートできません。次に例を示します。

```
# zpool import
  pool: dozer
    id: 9784486589352144634
 state: FAULTED
action: The pool cannot be imported. Attach the missing
        devices and try again.
   see: http://www.sun.com/msg/ZFS-8000-6X
config:
        raidz1-0       FAULTED
          c1t0d0       ONLINE
          c1t1d0       FAULTED
          c1t2d0       ONLINE
          c1t3d0       FAULTED
```

この例では、RAID-Z 仮想デバイスのうち、2 つのディスクが見つかりません。つまり、プールの再構築に必要な冗長データを利用できません。場合によっては、完全な構成を判断するために必要なデバイスが存在しないことがあります。この場合、ZFS ではほかにどのようなデバイスがプールを構成していたかを特定できませんが、その状況についてできるだけ多くの情報を報告しようとします。次に例を示します。

```
# zpool import
pool: dozer
    id: 9784486589352144634
 state: FAULTED
status: One or more devices are missing from the system.
action: The pool cannot be imported. Attach the missing
        devices and try again.
   see: http://www.sun.com/msg/ZFS-8000-6X
config:
        dozer          FAULTED   missing device
          raidz1-0     ONLINE
            c1t0d0     ONLINE
            c1t1d0     ONLINE
            c1t2d0     ONLINE
            c1t3d0     ONLINE
Additional devices are known to be part of this pool, though their
exact configuration cannot be determined.
```

ZFS ストレージプールを別のディレクトリからインポートする

デフォルトでは、`zpool import` コマンドは、`/dev/dsk` ディレクトリに含まれるデバイスだけを検索します。デバイスが別のディレクトリに存在するか、またはファイルに基づくプールを使用している場合は、`-d` オプションを使用して、代替ディレクトリを検索する必要があります。次に例を示します。

```
# zpool create dozer mirror /file/a /file/b
# zpool export dozer
# zpool import -d /file
  pool: dozer
    id: 7318163511366751416
 state: ONLINE
action: The pool can be imported using its name or numeric identifier.
config:

        dozer        ONLINE
          mirror-0   ONLINE
            /file/a  ONLINE
            /file/b  ONLINE
# zpool import -d /file dozer
```

デバイスが複数のディレクトリに存在する場合には、複数の `-d` オプションを指定できます。

ZFS ストレージプールをインポートする

インポートできるプールを確認したあとで、zpool import コマンドの引数にプールの名前または数値識別子を指定してインポートできます。次に例を示します。

```
# zpool import tank
```

インポートできるプールが複数存在し、それらが同じ名前を持っている場合でも、数値識別子を使ってインポートするプールを指定する必要があります。次に例を示します。

```
# zpool import
  pool: dozer
    id: 2704475622193776801
 state: ONLINE
action: The pool can be imported using its name or numeric identifier.
config:

        dozer       ONLINE
          c1t9d0    ONLINE

  pool: dozer
    id: 6223921996155991199
 state: ONLINE
action: The pool can be imported using its name or numeric identifier.
config:

        dozer       ONLINE
          c1t8d0    ONLINE
# zpool import dozer
cannot import 'dozer': more than one matching pool
import by numeric ID instead
# zpool import 6223921996155991199
```

プール名が既存のプール名と重複する場合は、別の名前でインポートできます。次に例を示します。

```
# zpool import dozer zeepool
```

このコマンドは、エクスポート済みのプール dozer を新しい名前 zeepool を使ってインポートします。

プールを明示的にエクスポートしていない場合は、別のシステムでまだ使用されているプールを誤ってインポートすることを防ぐためにインポートできません。-f フラグを使用する必要があります。次に例を示します。

```
# zpool import dozer
cannot import 'dozer': pool may be in use on another system
use '-f' to import anyway
# zpool import -f dozer
```

注 – あるシステムでアクティブになっているプールを別のシステムにインポートしようとしないでください。ZFS はネイティブのクラスタファイルシステム、分散ファイルシステム、または並列ファイルシステムではないため、異なる複数のホストからの同時アクセスには対応できません。

-R オプションを使用して、プールを代替ルートでインポートすることもできます。代替ルートプールの詳細については、294 ページの「ZFS 代替ルートプールを使用する」を参照してください。

破棄された ZFS ストレージプールを回復する

zpool import -D コマンドを使用して、破棄されたストレージプールを回復できます。次に例を示します。

```
# zpool destroy tank
# zpool import -D
  pool: tank
    id: 5154272182900538157
 state: ONLINE (DESTROYED)
action: The pool can be imported using its name or numeric identifier.
config:

        tank        ONLINE
          mirror-0  ONLINE
            c1t0d0  ONLINE
            c1t1d0  ONLINE
```

この zpool import の出力では、次の状態情報により、tank プールが破棄されたプールであることがわかります。

```
state: ONLINE (DESTROYED)
```

破棄されたプールを回復するには、回復するプールに対して zpool import -D コマンドを再度実行します。次に例を示します。

```
# zpool import -D tank
# zpool status tank
  pool: tank
 state: ONLINE
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        tank        ONLINE
          mirror-0  ONLINE
            c1t0d0  ONLINE
            c1t1d0  ONLINE
```

```
errors: No known data errors
```

破棄されたプール内のいずれかのデバイスがエラー状態または使用できない場合でも、-f オプションを含めることによって、破棄されたプールを別の方法で回復できることがあります。このような場合には、機能が低下したプールをインポートしてから、デバイスの障害の修正を試みます。次に例を示します。

```
# zpool destroy dozer
# zpool import -D
pool: dozer
    id: 13643595538644303788
 state: DEGRADED (DESTROYED)
status: One or more devices could not be opened.  Sufficient replicas exist for
        the pool to continue functioning in a degraded state.
action: Attach the missing device and online it using 'zpool online'.
   see: http://www.sun.com/msg/ZFS-8000-2Q
config:

        NAME         STATE     READ WRITE CKSUM
        dozer        DEGRADED     0     0     0
          raidz2-0   DEGRADED     0     0     0
            c2t8d0   ONLINE       0     0     0
            c2t9d0   ONLINE       0     0     0
            c2t10d0  ONLINE       0     0     0
            c2t11d0  UNAVAIL      0    35     1  cannot open
            c2t12d0  ONLINE       0     0     0

errors: No known data errors
# zpool import -Df dozer
# zpool status -x
  pool: dozer
 state: DEGRADED
status: One or more devices could not be opened.  Sufficient replicas exist for
        the pool to continue functioning in a degraded state.
action: Attach the missing device and online it using 'zpool online'.
   see: http://www.sun.com/msg/ZFS-8000-2Q
 scrub: scrub completed after 0h0m with 0 errors on Thu Jan 21 15:38:48 2010
config:

        NAME         STATE     READ WRITE CKSUM
        dozer        DEGRADED     0     0     0
          raidz2-0   DEGRADED     0     0     0
            c2t8d0   ONLINE       0     0     0
            c2t9d0   ONLINE       0     0     0
            c2t10d0  ONLINE       0     0     0
            c2t11d0  UNAVAIL      0    37     0  cannot open
            c2t12d0  ONLINE       0     0     0

errors: No known data errors
# zpool online dozer c2t11d0
Bringing device c2t11d0 online
# zpool status -x
all pools are healthy
```

ZFS ストレージプールをアップグレードする

Solaris 10 10/09 リリースなど、以前の Solaris リリースの ZFS ストレージプールがある場合には、`zpool upgrade` コマンドを使ってそのプールをアップグレードすれば、現行リリースのプール機能を利用することができます。また、古いバージョンのプールを実行している場合、`zpool status` コマンドによって通知されます。次に例を示します。

```
# zpool status
  pool: tank
 state: ONLINE
status: The pool is formatted using an older on-disk format.  The pool can
        still be used, but some features are unavailable.
action: Upgrade the pool using 'zpool upgrade'.  Once this is done, the
        pool will no longer be accessible on older software versions.
 scrub: none requested
config:
        NAME        STATE     READ WRITE CKSUM
        tank        ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c1t0d0  ONLINE       0     0     0
            c1t1d0  ONLINE       0     0     0
errors: No known data errors
```

次の構文を使って、特定のバージョンやサポートされるリリースに関する追加情報を確認できます。

```
# zpool upgrade -v
This system is currently running ZFS pool version 22.

The following versions are supported:

VER  DESCRIPTION
---  --------------------------------------------------------
 1   Initial ZFS version
 2   Ditto blocks (replicated metadata)
 3   Hot spares and double parity RAID-Z
 4   zpool history
 5   Compression using the gzip algorithm
 6   bootfs pool property
 7   Separate intent log devices
 8   Delegated administration
 9   refquota and refreservation properties
 10  Cache devices
 11  Improved scrub performance
 12  Snapshot properties
 13  snapused property
 14  passthrough-x aclinherit
 15  user/group space accounting
 16  stmf property support
 17  Triple-parity RAID-Z
 18  Snapshot user holds
 19  Log device removal
 20  Compression using zle (zero-length encoding)
```

```
 21  Reserved
 22  Received properties

For more information on a particular version, including supported releases,
see the ZFS Administration Guide.
```

これで、zpool upgrade コマンドを実行してすべてのプールをアップグレードできます。次に例を示します。

```
# zpool upgrade -a
```

注 – プールを新しい ZFS バージョンにアップグレードすると、古い ZFS バージョンを実行しているシステムのプールにアクセスできなくなります。

◆ ◆ ◆ 第 5 章

Oracle Solaris ZFS ルートファイルシステムのインストールと起動

この章では、Oracle Solaris ZFS ファイルシステムのインストールと起動の方法について説明します。Oracle Solaris Live Upgrade を使用して UFS ルートファイルシステムを ZFS ファイルシステムに移行する方法についても説明します。

この章は、次の節で構成されます。

このリリースで認識されている問題のリストについては、『Oracle Solaris 10 9/10 ご使用にあたって』を参照してください。

トラブルシューティングに関する最新情報については、次のサイトを参照してください。

http://www.solarisinternals.com/wiki/index.php/ZFS_Troubleshooting_Guide

Oracle Solaris ZFS ルートファイルシステムのインストールと起動 (概要)

Solaris 10 10/08 以降のリリースでは、次の方法で ZFS ルートファイルシステムからインストールと起動を行うことができます。

- 初期インストールを実行できます。その場合、ZFS がルートファイルシステムとして選択されます。
- Oracle Solaris Live Upgrade を使用して、UFS ルートファイルシステムを ZFS ルートファイルシステムに移行できます。また、Oracle Solaris Live Upgrade を使用して次のタスクを実行することもできます。
  - 既存の ZFS ルートプール内に新しいブート環境を作成する。
  - 新しい ZFS ルートプール内に新しいブート環境を作成する。
- Oracle Solaris JumpStart プロファイルを使用して、システムが ZFS ルートファイルシステムによって自動的にインストールされるようにすることができます。
- Solaris 10 10/09 以降のリリースでは、JumpStart プロファイルを使用して、システムが ZFS フラッシュアーカイブによって自動的にインストールされるようにすることができます。

SPARC システムまたは x86 システムを ZFS ルートファイルシステムでインストールするか、ZFS ルートファイルシステムに移行したあとは、システムは自動的に ZFS ルートファイルシステムから起動します。起動の変更に関する詳細は、170 ページの「ZFS ルートファイルシステムからの起動」を参照してください。

ZFS インストール機能

この Solaris リリースでは、次の ZFS インストール機能が用意されています。

- Solaris 対話式テキストインストーラを使用して、UFS または ZFS ルートファイルシステムをインストールできます。この Solaris リリースでも、デフォルトのファイルシステムは UFS です。対話式テキストインストーラは、次の方法で利用できます。
  - SPARC: Solaris インストール DVD から起動する場合は次の構文を使用します。

    ```
    ok boot cdrom - text
    ```
  - SPARC: ネットワークから起動する場合は次の構文を使用します。

    ```
    ok boot net - text
    ```
  - x86: テキストモードインストールのオプションを選択します。
- カスタム JumpStart プロファイルが提供する機能は次のとおりです。
  - ZFS ストレージプールを作成してブート可能な ZFS ファイルシステムを指定するプロファイルをセットアップすることができます。

    - ZFS ルートプールのフラッシュアーカイブを識別するプロファイルをセットアップすることができます。
- Oracle Solaris Live Upgrade を使用して、UFS ルートファイルシステムを ZFS ルートファイルシステムに移行できます。`lucreate` コマンドと `luactivate` コマンドが変更され、ZFS プールと ZFS ファイルシステムをサポートするようになりました。
- インストール時に 2 つのディスクを選択することにより、ミラー化された ZFS ルートプールを設定できます。また、インストール後に追加ディスクを接続することにより、ミラー化された ZFS ルートプールを作成できます。
- ZFS ルートプールの ZFS ボリュームにスワップデバイスとダンプデバイスが自動的に作成されます。

このリリースでは、次のインストール機能は用意されていません。

- 現時点では、ZFS ルートファイルシステムのインストールに GUI 機能は使用できません。
- ZFS ルートファイルシステムをインストールするための Oracle Solaris フラッシュインストール機能は、初期インストールオプションからフラッシュインストールオプションを選択しても、使用することはできません。ただし、ZFS ルートプールのフラッシュアーカイブを識別するための JumpStart プロファイルを作成することができます。詳細は、137 ページの「ZFS ルートファイルシステムのインストール (Oracle Solaris フラッシュアーカイブインストール)」を参照してください。
- 標準のアップグレードプログラムを使用して UFS ルートファイルシステムを ZFS ルートファイルシステムにアップグレードすることはできません。

ZFS をサポートするための Oracle Solaris インストールと Oracle Solaris Live Upgrade の要件

システムを ZFS ルートファイルシステムでインストールする場合や、UFS ルートファイルシステムを ZFS ルートファイルシステムに移行する場合は、次の要件が満たされていることを事前に確認してください。

Oracle Solaris リリースの要件

ZFS ルートファイルシステムのインストールと起動または ZFS ルートファイルシステムへの移行は、次の方法で行うことができます。

- ZFS ルートファイルシステムをインストールする – Solaris 10 10/08 以降のリリースで使用できます。
- Oracle Solaris Live Upgrade を使用して UFS ルートファイルシステムから ZFS ルートファイルシステムに移行する – Solaris 10 10/08 以降のリリースをインストールしてあるか、Solaris 10 10/08 以降のリリースにアップグレードしてあることが必要です。

一般的な ZFS ストレージプール要件

次の各節では、ZFS ルートプールの容量および構成の要件について説明します。

ZFS ストレージプールのディスク容量要件

ZFS ルート環境にはスワップデバイスおよびダンプデバイスとして別個のデバイスが必要なので、ZFS ルートファイルシステムに最小限必要なプール容量は、UFS ルートファイルシステムの場合よりも大きくなります。UFS ルートファイルシステムの場合、デフォルトではスワップデバイスとダンプデバイスは同一のデバイスです。

システムを ZFS ルートファイルシステムでインストールまたはアップグレードする場合、スワップ領域とダンプデバイスのサイズは、物理メモリーの容量によって決まります。ブート可能な ZFS ルートファイルシステムに最小限必要なプール容量は、物理メモリーの容量、利用可能なディスク容量、および作成するブート環境 (BE) の数によって決まります。

次の ZFS ストレージプールのディスク容量要件を確認してください。

- ZFS ルートファイルシステムのインストールに必要な最小メモリー容量は 768M バイトです。
- ZFS の全体的なパフォーマンスを向上させるには、1G バイトのメモリーを搭載することをお勧めします。
- 推奨される最小ディスク容量は 16G バイトです。ディスク容量は次のように消費されます。
  - スワップ領域とダンプデバイス – Solaris インストールプログラムによって作成されるスワップボリュームとダンプボリュームのデフォルトのサイズは、次のとおりです。
    - **Solaris** 初期インストール – 新しい ZFS ブート環境のスワップボリュームのデフォルトサイズは、物理メモリーのサイズの半分 (一般に 512M バイトから 2G バイトの範囲) として計算されます。スワップサイズは、初期インストール時に調整することができます。
    - ダンプボリュームのデフォルトのサイズは、`dumpadm` の情報と物理メモリーのサイズに基づいて、カーネルによって計算されます。ダンプサイズは、初期インストール時に調整することができます。
    - **Oracle Solaris Live Upgrade** – UFS ルートファイルシステムを ZFS ルートファイルシステムに移行する場合、ZFS BE のスワップボリュームのデフォルトサイズは、UFS BE のスワップデバイスのサイズとして計算されます。スワップボリュームのデフォルトサイズの計算では、UFS BE 内のすべてのスワップデバイスのサイズが合計され、そのサイズの ZFS ボリュームが ZFS BE 内に作成されます。UFS BE にスワップデバイスが定義されていない場合、スワップボリュームのデフォルトサイズは 512M バイトに設定されます。

- ZFS BE のダンプボリュームのデフォルトサイズは、物理メモリーのサイズの半分 (512M バイトから 2G バイトの間) に設定されます。

スワップボリュームとダンプボリュームのサイズを新しいサイズに調整することができます。ただし、システムの動作をサポートするサイズを選択する必要があります。詳細は、167 ページの「ZFS スワップデバイスおよびダンプデバイスのサイズを調整する」を参照してください。

- ブート環境 **(BE)** – 新しいスワップおよびダンプの容量要件、または調整したスワップおよびダンプのデバイスサイズのほかに、UFS BE から移行した ZFS BE には約 6G バイトが必要です。別の ZFS BE から複製された各 ZFS BE には、追加のディスク容量は必要ありませんが、パッチが適用されると BE のサイズが増加することを考慮してください。同じルートプール内の ZFS BE はすべて、同じスワップデバイスとダンプデバイスを使用します。
- **Solaris OS** コンポーネント – ルートファイルシステムの、OS イメージの一部となっているサブディレクトリのうち、`/var` 以外のものはすべて、ルートファイルシステムと同じデータセット内に存在している必要があります。さらに、スワップデバイスとダンプデバイス以外の Solaris OS コンポーネントはすべて、ルートプール内に存在している必要があります。

  さらに、`/var` ディレクトリまたはデータセットは単一のデータセットでなければならない、という制限もあります。たとえば、Oracle Solaris Live Upgrade を使って ZFS BE の移行やパッチ適用を行ったり、このプールの ZFS フラッシュアーカイブを作成したりする必要もある場合には、`/var/tmp` のような `/var` の下位データセットを作成することはできません。

たとえば、ブート可能な ZFS 環境には、12G バイトのディスク容量を備えたシステムでは小さすぎる可能性があります。UFS BE から移行した ZFS BE の場合、スワップデバイスとダンプデバイスにそれぞれ約 6G バイトのディスク容量が必要になるためです。

ZFS ストレージプールの構成要件

次の ZFS ストレージプール構成要件を確認してください。

- ルートプールに使用するプールには SMI ラベルが付いていなければなりません。ディスクスライスを使用して作成されたプールでは、この要件が満たされます。
- プールは、ディスクスライスとミラー化されているディスクスライスのいずれかに存在していなければなりません。Oracle Solaris Live Upgrade での移行時に、サポートされていないプール構成を使用しようとすると、次のようなメッセージが表示されます。

```
ERROR: ZFS pool name does not support boot environments
```

サポートされている ZFS ルートプール構成の詳細については、73 ページの「ZFS ルートプールを作成する」を参照してください。

- x86: Solaris `fdisk` パーティションがディスクに含まれている必要があります。Solaris `fdisk` パーティションは、x86 システムのインストール時に自動的に作成されます。Solaris `fdisk` パーティションの詳細については、『Solaris のシステム管理 (デバイスとファイルシステム)』の「fdisk パーティションの作成上のガイドライン」を参照してください。
- SPARC システムでも x86 システムでも、ZFS ルートプールで起動用として指定するディスクのサイズは 1T バイト以下でなければなりません。
- ルートプールで圧縮を有効にすることはできますが、ルートプールをインストールしたあとでないと有効にすることはできません。ルートプールのインストール時に圧縮を有効にする方法はありません。ルートプールでは `gzip` 圧縮アルゴリズムはサポートされていません。
- 初期インストールによるルートプールの作成後、あるいは Solaris Live Upgrade による ZFS ルートファイルシステムへの移行後に、ルートプールの名前を変更しないでください。ルートプールの名前を変更すると、システムが起動できなくなる可能性があります。

ZFS ルートファイルシステムのインストール (初期インストール)

この Solaris リリースでは、Solaris 対話式テキストインストーラを使用して初期インストールを実行し、ブート可能な ZFS ルートファイルシステムを含む ZFS ストレージプールを作成できます。既存の ZFS ストレージプールを ZFS ルートファイルシステムとして使用するには、Oracle Solaris Live Upgrade を使用して、既存の ZFS ストレージプール内で既存の UFS ルートファイルシステムを ZFS ファイルシステムに移行する必要があります。詳細は、144 ページの「UFS ルートファイルシステムから ZFS ルートファイルシステムへの移行 (Oracle Solaris Live Upgrade)」を参照してください。

ZFS ルートファイルシステムの初期インストールのあとでゾーンを構成し、システムにパッチやアップグレードを適用することを計画している場合は、151 ページの「ゾーンが含まれているシステムを Oracle Solaris Live Upgrade で移行またはアップグレードする (Solaris 10 10/08)」または 156 ページの「ゾーンが含まれているシステムを Oracle Solaris Live Upgrade で移行またはアップグレードする (Solaris 10 5/09 以降)」を参照してください。

ZFS ストレージプールがシステム上にすでに存在している場合、次のメッセージによってその旨が通知されます。ただし、ユーザーがそれらの既存プール内のディスクを新たに作成するストレージプール用として選択しないかぎり、それらのプールはそのまま残されます。

```
There are existing ZFS pools available on this system.  However, they can only be upgraded
using the Live Upgrade tools.  The following screens will only allow you to install a ZFS root system,
not upgrade one.
```

注意 – 既存のプールのディスクのいずれかを新しいプール用に選択すると、既存のプールは破棄されます。

初期インストールを実行して ZFS ストレージプールを作成する前に、127 ページの「ZFS をサポートするための Oracle Solaris インストールと Oracle Solaris Live Upgrade の要件」を参照してください。

例 5–1 ブート可能な ZFS ルートファイルシステムの初期インストール

Solaris 対話式テキストインストーラによるインストール処理は、基本的に以前の Solaris リリースの場合と同じですが、UFS または ZFS ルートファイルシステムの作成を求めるプロンプトが表示される点が異なります。このリリースでも、デフォルトのファイルシステムは UFS です。ZFS ルートファイルシステムを選択すると、ZFS ストレージプールの作成を求めるプロンプトが表示されます。ZFS ルートファイルシステムのインストール手順は次のとおりです。

1. ブート可能な ZFS ルートファイルシステムの作成に Solaris フラッシュインストールは使用できないため、Solaris の対話式インストール方法を選択します。ただし、ZFS フラッシュアーカイブを作成して JumpStart インストール時に使用することができます。詳細は、137 ページの「ZFS ルートファイルシステムのインストール (Oracle Solaris フラッシュアーカイブインストール)」を参照してください。

   Solaris 10 10/08 以降のリリースでは、Solaris 10 10/08 以降のリリースがすでにインストールされていれば、UFS ルートファイルシステムから ZFS ルートファイルシステムに移行することができます。ZFS ルートファイルシステムへの移行の詳細については、144 ページの「UFS ルートファイルシステムから ZFS ルートファイルシステムへの移行 (Oracle Solaris Live Upgrade)」を参照してください。
2. ZFS ルートファイルシステムを作成するには、ZFS オプションを選択します。次に例を示します。

```
Choose Filesystem Type

  Select the filesystem to use for your Solaris installation


            [ ] UFS
            [X] ZFS
```

3. インストールするソフトウェアを選択したあと、ZFS ストレージプールを作成するためのディスクの選択を求めるプロンプトが表示されます。この画面は、前の Solaris リリースとほぼ同じです。

```
Select Disks

  On this screen you must select the disks for installing Solaris software.
  Start by looking at the Suggested Minimum field; this value is the
  approximate space needed to install the software you've selected. For ZFS,
  multiple disks will be configured as mirrors, so the disk you choose, or the
```

例 5–1 ブート可能な ZFS ルートファイルシステムの初期インストール (続き)

```
slice within the disk must exceed the Suggested Minimum value.
NOTE: ** denotes current boot disk

Disk Device                                                 Available Space
=============================================================================
[X]    c1t0d0                                              69994 MB  (F4 to edit)
[ ]    c1t1d0                                              69994 MB
[-]    c1t2d0                                                  0 MB
[-]    c1t3d0                                                  0 MB

                                    Maximum Root Size:  69994 MB
                                    Suggested Minimum:   8279 MB
```

ZFS ルートプールに使用する 1 つ以上のディスクを選択できます。2 つのディスクを選択すると、ルートプールには 2 ディスク構成が設定されます。2 ディスクまたは 3 ディスクのミラー化プールが最適です。8 つのディスクがある場合にそれらすべてを選択すると、ルートプールでは 8 つのディスクが単一の大規模なミラーとして使用されます。この構成は最適ではありません。もう 1 つの方法は、初期インストールの完了後にミラー化ルートプールを作成することです。ルートプールでは RAID-Z プール構成はサポートされていません。ZFS ストレージプールの構成方法の詳細については、69 ページの「ZFS ストレージプールの複製機能」を参照してください。

4. ミラー化ルートプールを作成するために 2 つのディスクを選択するには、Ctrl キーを押しながら 2 番目のディスクをクリックします。次の例では、`c1t1d1` と `c1t2d0` の両方をルートプールディスクとして選択しています。両方のディスクには、SMI ラベルが付けられていて、スライス 0 が割り当てられていなければなりません。ディスクに SMI ラベルが付けられていない場合やディスクにスライスが含まれていない場合は、インストールプログラムを終了し、`format` ユーティリティを使用して、ディスクのラベルを変更し、パーティションを再設定してから、インストールプログラムを再起動してください。

```
Select Disks

  On this screen you must select the disks for installing Solaris software.
  Start by looking at the Suggested Minimum field; this value is the
  approximate space needed to install the software you've selected. For ZFS,
  multiple disks will be configured as mirrors, so the disk you choose, or the
  slice within the disk must exceed the Suggested Minimum value.
  NOTE: ** denotes current boot disk

  Disk Device                                                 Available Space
  =============================================================================
  [X]    c1t0d0                                              69994 MB
  [X]    c1t1d0                                              69994 MB  (F4 to edit)
  [-]    c1t2d0                                                  0 MB
  [-]    c1t3d0                                                  0 MB

                                      Maximum Root Size:  69994 MB
                                      Suggested Minimum:   8279 MB
```

例 5–1 ブート可能な ZFS ルートファイルシステムの初期インストール (続き)

「Available Space」欄が 0M バイトになっている場合、そのディスクには通常 EFI ラベルが付いています。EFI ラベルの付いたディスクを使用するには、インストールプログラムを終了し、`format -e` コマンドを使ってそのディスクに SMI ラベルを付け直したあと、インストールプログラムを再度起動します。

インストール中にミラー化ルートプールを作成しなかった場合も、インストール後にそのようなプールを容易に作成できます。これについては、136 ページの「ミラー化ルートプールを作成する方法 (インストール後)」を参照してください。

5. ZFS ルートプールに使用する 1 つ以上のディスクを選択したら、次のような画面が表示されます。

```
Configure ZFS Settings

  Specify the name of the pool to be created from the disk(s) you have chosen.
  Also specify the name of the dataset to be created within the pool that is
  to be used as the root directory for the filesystem.

              ZFS Pool Name: rpool
      ZFS Root Dataset Name: s10s_u9wos_08
      ZFS Pool Size (in MB): 69995
  Size of Swap Area (in MB): 2048
  Size of Dump Area (in MB): 1536
        (Pool size must be between 6231 MB and 69995 MB)

                        [X] Keep / and /var combined
                        [ ] Put /var on a separate dataset
```

この画面では、ZFS プールの名前、データセット名、プールサイズ、およびスワップデバイスとダンプデバイスのサイズを変更できます。変更するには、カーソル制御キーでエントリの中を移動し、デフォルトの値を新しい値で置き換えます。あるいは、デフォルト値をそのまま使用できます。また、`/var` ファイルシステムの作成およびマウントの方法を変更することもできます。

次の例では、ルートデータセットの名前が `zfsBE` に変更されます。

```
              ZFS Pool Name: rpool
      ZFS Root Dataset Name: zfsBE
      ZFS Pool Size (in MB): 69995
  Size of Swap Area (in MB): 2048
  Size of Dump Area (in MB): 1536
        (Pool size must be between 6231 MB and 69995 MB)

                        [X] Keep / and /var combined
                        [ ] Put /var on a separate dataset
```

6. この最後のインストール画面では、インストールプロファイルを変更できます。次に例を示します。

```
Profile

  The information shown below is your profile for installing Solaris software.
```

例 5–1 ブート可能な ZFS ルートファイルシステムの初期インストール (続き)

```
It reflects the choices you've made on previous screens.

============================================================================

              Installation Option: Initial
                      Boot Device: c1t0d0
            Root File System Type: ZFS
                  Client Services: None

                         Regions: North America
                   System Locale: C ( C )

                       Software: Solaris 10, Entire Distribution
                      Pool Name: rpool
            Boot Environment Name: zfsBE
                      Pool Size: 69995 MB
                  Devices in Pool: c1t0d0
                                   c1t1d0
```

7. インストールが完了したら、作成された ZFS ストレージプールおよびファイルシステムの情報を確認します。次に例を示します。

```
# zpool status
  pool: rpool
 state: ONLINE
 scrub: none requested
config:

        NAME          STATE     READ WRITE CKSUM
        rpool         ONLINE       0     0     0
          mirror-0    ONLINE       0     0     0
            c1t0d0s0  ONLINE       0     0     0
            c1t1d0s0  ONLINE       0     0     0

errors: No known data errors
# zfs list
NAME                 USED  AVAIL  REFER  MOUNTPOINT
rpool               8.03G  58.9G    96K  /rpool
rpool/ROOT          4.47G  58.9G    21K  legacy
rpool/ROOT/zfsBE    4.47G  58.9G  4.47G  /
rpool/dump          1.50G  58.9G  1.50G  -
rpool/export          44K  58.9G    23K  /export
rpool/export/home     21K  58.9G    21K  /export/home
rpool/swap          2.06G  61.0G    16K  -
```

このサンプルの `zfs list` の出力では、`rpool/ROOT` ディレクトリなどルートプールのコンポーネントが識別されています。デフォルトでは、これらにはアクセスできません。

8. 同じストレージプール内に別の ZFS ブート環境 (BE) を作成するには、`lucreate` コマンドを使用できます。次の例では、`zfs2BE` という名前の新しい BE が作成されます。`zfs list` の出力からわかるように、現在の BE の名前は `zfsBE` です。ただし、この現在の BE は、新しい BE が作成されるまで `lustatus` の出力に表示されません。

例 5–1 ブート可能な ZFS ルートファイルシステムの初期インストール (続き)

```
# lustatus
ERROR: No boot environments are configured on this system
ERROR: cannot determine list of all boot environment names
```

同じプール内に新しい ZFS BE を作成する場合は、次のような構文を使用します。

```
# lucreate -n zfs2BE
INFORMATION: The current boot environment is not named - assigning name <zfsBE>.
Current boot environment is named <zfsBE>.
Creating initial configuration for primary boot environment <zfsBE>.
The device </dev/dsk/c1t0d0s0> is not a root device for any boot environment; cannot get BE ID.
PBE configuration successful: PBE name <zfsBE> PBE Boot Device </dev/dsk/c1t0d0s0>.
Comparing source boot environment <zfsBE> file systems with the file
system(s) you specified for the new boot environment. Determining which
file systems should be in the new boot environment.
Updating boot environment description database on all BEs.
Updating system configuration files.
Creating configuration for boot environment <zfs2BE>.
Source boot environment is <zfsBE>.
Creating boot environment <zfs2BE>.
Cloning file systems from boot environment <zfsBE> to create boot environment <zfs2BE>.
Creating snapshot for <rpool/ROOT/zfsBE> on <rpool/ROOT/zfsBE@zfs2BE>.
Creating clone for <rpool/ROOT/zfsBE@zfs2BE> on <rpool/ROOT/zfs2BE>.
Setting canmount=noauto for </> in zone <global> on <rpool/ROOT/zfs2BE>.
Population of boot environment <zfs2BE> successful.
Creation of boot environment <zfs2BE> successful.
```

同じプール内で ZFS BE を作成する場合、ZFS のクローン機能とスナップショット機能を使ってその BE が即座に作成されます。Oracle Solaris Live Upgrade を使用した ZFS ルートへの移行の詳細については、144 ページの「UFS ルートファイルシステムから ZFS ルートファイルシステムへの移行 (Oracle Solaris Live Upgrade)」を参照してください。

9. 次に、新しいブート環境を確認します。次に例を示します。

```
# lustatus
Boot Environment           Is       Active Active    Can    Copy
Name                       Complete Now    On Reboot Delete Status
-------------------------- -------- ------ --------- ------ ----------
zfsBE                      yes      yes    yes       no     -
zfs2BE                     yes      no     no        yes    -
# zfs list
NAME                      USED  AVAIL  REFER  MOUNTPOINT
rpool                    8.03G  58.9G    97K  /rpool
rpool/ROOT               4.47G  58.9G    21K  legacy
rpool/ROOT/zfs2BE         116K  58.9G  4.47G  /
rpool/ROOT/zfsBE         4.47G  58.9G  4.47G  /
rpool/ROOT/zfsBE@zfs2BE  75.5K      -  4.47G  -
rpool/dump               1.50G  58.9G  1.50G  -
rpool/export               44K  58.9G    23K  /export
rpool/export/home          21K  58.9G    21K  /export/home
rpool/swap               2.06G  61.0G    16K  -
```

例 5–1 ブート可能な ZFS ルートファイルシステムの初期インストール (続き)

10. 代替 BE からブートするには、`luactivate` コマンドを使用します。SPARC システムで BE をアクティブにしたあと、起動デバイスに ZFS ストレージプールが含まれている場合は、利用可能な BE を `boot -L` コマンドで識別します。x86 システムから起動する場合は、起動する BE を GRUB メニューで識別します。

    たとえば、SPARC システムでは、`boot -L` コマンドを使用して利用可能な BE のリストを表示します。新しい BE `zfs2BE` から起動するには、オプション 2 を選択します。次に、表示された `boot -Z` コマンドを入力します。

```
ok boot -L
Executing last command: boot -L
Boot device: /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/disk@0  File and args: -L
1 zfsBE
2 zfs2BE
Select environment to boot: [ 1 - 2 ]: 2

To boot the selected entry, invoke:
boot [<root-device>] -Z rpool/ROOT/zfs2BE
ok boot -Z rpool/ROOT/zfs2BE
```

ZFS ファイルシステムの起動に関する詳細は、170 ページの「ZFS ルートファイルシステムからの起動」を参照してください。

▼ ミラー化ルートプールを作成する方法 (インストール後)

インストール中に ZFS ミラー化ルートプールを作成しなかった場合も、インストール後にそのようなプールを容易に作成できます。

ZFS ルートプールのディスクを置き換える方法については、178 ページの「ZFS ルートプールのディスクを置き換える方法」を参照してください。

1 ルートプールの現在の状態を表示します。

```
# zpool status rpool
  pool: rpool
 state: ONLINE
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        rpool       ONLINE       0     0     0
          c1t0d0s0  ONLINE       0     0     0

errors: No known data errors
```

2 ミラー化ルートプール構成にするために、**2** つ目のディスクを接続します。

```
# zpool attach rpool c1t0d0s0 c1t1d0s0
Please be sure to invoke installboot(1M) to make 'c1t1d0s0' bootable.
```

```
Make sure to wait until resilver is done before rebooting.
```

3 ルートプールの状態を表示し、再同期化が完了しているか確認します。

```
# zpool status rpool
  pool: rpool
 state: ONLINE
status: One or more devices is currently being resilvered.  The pool will
        continue to function, possibly in a degraded state.
action: Wait for the resilver to complete.
 scrub: resilver in progress for 0h1m, 24.26% done, 0h3m to go
config:

        NAME          STATE     READ WRITE CKSUM
        rpool         ONLINE       0     0     0
          mirror-0    ONLINE       0     0     0
            c1t0d0s0  ONLINE       0     0     0
            c1t1d0s0  ONLINE       0     0     0  3.18G resilvered

errors: No known data errors
```

上の出力の場合、再同期化処理は完了していません。次のようなメッセージが表示されたら、再同期化が完了しています。

```
scrub: resilver completed after 0h10m with 0 errors on Thu Mar 11 11:27:22 2010
```

4 再同期化の完了後、2 つ目のディスクにブートブロックを適用します。

```
sparc# installboot -F zfs /usr/platform/‘uname -i‘/lib/fs/zfs/bootblk /dev/rdsk/c1t1d0s0

x86# installgrub /boot/grub/stage1 /boot/grub/stage2 /dev/rdsk/c1t1d0s0
```

5 2 つ目のディスクから正常に起動できることを確認します。

6 システムが新しいディスクから自動的に起動するように設定します。そのためには、eeprom コマンドまたは SPARC ブート PROM の setenv コマンドを使用します。あるいは、PC BIOS を再設定します。

ZFS ルートファイルシステムのインストール (Oracle Solaris フラッシュアーカイブインストール)

Solaris 10 10/09 以降のリリースでは、フラッシュアーカイブは、UFS ルートファイルシステムまたは ZFS ルートファイルシステムを実行しているシステムで作成できます。ZFS ルートプールのフラッシュアーカイブには、スワップボリュームとダンプボリュームおよび任意の除外されたデータセットを除く、プール階層全体が含まれます。スワップボリュームとダンプボリュームは、フラッシュアーカイブのインストール時に作成されます。フラッシュアーカイブによるインストール方法は次のとおりです。

- ZFS ルートファイルシステムによるシステムのインストールとブートに使用できるフラッシュアーカイブを生成します。

- ZFS フラッシュアーカイブを使用して、システムの JumpStart インストールを実行します。ZFS フラッシュアーカイブを作成すると、個別のブート環境ではなく、ルートプール全体が複製されます。`flarcreate` コマンドと `flar` コマンドの `-D` オプションを使用すると、プール内の個別のデータセットを除外できます。

ZFS フラッシュアーカイブを使用してシステムをインストールする前に、次の制限事項を確認してください。

- サポートされているのは、ZFS フラッシュアーカイブの JumpStart インストールのみです。フラッシュアーカイブの対話式インストールオプションを使用して ZFS ルートファイルシステムを含んでいるシステムをインストールすることはできません。また、Oracle Solaris Live Upgrade によって ZFS BE をインストールするためにフラッシュアーカイブを使用することもできません。
- ZFS フラッシュアーカイブは、その作成元のシステムと同じアーキテクチャーのシステムにしかインストールできません。例えば、`sun4u` システムで作成されたアーカイブは、`sun4v` システムにはインストールできません。
- サポートされているのは、ZFS フラッシュアーカイブの完全な初期インストールのみです。ZFS ルートファイルシステムのさまざまなフラッシュアーカイブをインストールすることはできず、また、ハイブリッド UFS/ZFS アーカイブをインストールすることもできません。
- 従来どおり、UFS ルートファイルシステムをインストールするために使用できるのは既存の UFS フラッシュアーカイブのみです。ZFS ルートファイルシステムをインストールするために使用できるのは ZFS フラッシュアーカイブのみです。
- ルートプール全体 (ただし、明示的に除外されたデータセットを除く) がアーカイブされてインストールされますが、フラッシュアーカイブのインストール後に使用できるのは、アーカイブを作成時に起動されていた ZFS BE のみです。ただし、`flarcreate` または `flar` コマンドの `-R` *rootdir* オプションによってアーカイブされたプールを使用して、現在起動されているルートプール以外のルートプールをアーカイブすることができます。
- フラッシュアーカイブを使用して作成される ZFS ルートプールの名前は、マスタールートプールの名前と同じです。フラッシュアーカイブを作成するために使用するルートプールの名前は、新規作成するプールに割り当てる名前です。プール名の変更はサポートされていません。
- 個々のファイルを含める場合や除外する場合に使用する `flarcreate` および `flar` コマンドオプションは、ZFS フラッシュアーカイブではサポートされていません。データセットを ZFS フラッシュアーカイブから除外する場合はデータセット全体が除外されます。
- `flar info` コマンドは ZFS フラッシュアーカイブではサポートされていません。次に例を示します。

```
# flar info -l zfs10u8flar
ERROR: archive content listing not supported for zfs archives.
```

マスターシステムに Solaris 10 10/09 以降のリリースを新規インストールした後、または、マスターシステムを Solaris 10 10/09 以降のリリースにアップグレードした後に、ターゲットシステムのインストールに使用する ZFS フラッシュアーカイブを作成することができます。基本的な手順は次のとおりです。

- マスターシステムで Solaris 10 10/09 以降のリリースをインストールまたはアップグレードします。必要なカスタマイズを加えます。
- マスターシステムで `flarcreate` コマンドを使用して ZFS フラッシュアーカイブを作成します。ZFS フラッシュアーカイブには、スワップボリュームとダンプボリューム以外のルートプール内のすべてのデータセットが含まれます。
- インストールサーバーで、JumpStart プロファイルを作成してフラッシュアーカイブ情報を含めます。
- ZFS フラッシュアーカイブをターゲットシステムにインストールします。

フラッシュアーカイブによる ZFS ルートプールのインストールでサポートされているアーカイブオプションは、次のとおりです。

- `flarcreate` または `flar` コマンドを使用して、ZFS ルートプールを指定してフラッシュアーカイブを作成します。特に指定しない場合は、デフォルトのルートプールのフラッシュアーカイブが作成されます。
- `flarcreate -D` *dataset* を使用して、データセットを指定してフラッシュアーカイブから除外します。このオプションを複数回使用して複数のデータセットを除外することができます。

ZFS フラッシュアーカイブがインストールされると、システムが次のように構成されます。

- フラッシュアーカイブが作成されたシステム上のデータセット階層全体 (ただし、アーカイブの作成時に明示的に除外されたデータセットを除く) がターゲットシステム上で再作成されます。スワップボリュームおよびダンプボリュームは、フラッシュアーカイブに含まれません。
- ルートプールには、アーカイブを作成するために使用されたプールと同じ名前が付けられます。
- フラッシュアーカイブの作成時にアクティブだったブート環境が、展開先のシステムのアクティブなデフォルトの BE になります。

例 5–2　ZFS フラッシュアーカイブを使用してシステムをインストールする

マスターシステムに Solaris 10 10/09 以降のリリースを新規インストールした後、またはマスターシステムを Solaris 10 10/09 以降のリリースにアップグレードした後に、ZFS ルートプールのフラッシュアーカイブを作成します。次に例を示します。

```
# flarcreate -n zfsBE zfs10upflar
Full Flash
Checking integrity...
Integrity OK.
```

例 5–2　ZFS フラッシュアーカイブを使用してシステムをインストールする　(続き)

```
Running precreation scripts...
Precreation scripts done.
Determining the size of the archive...
The archive will be approximately 4.94GB.
Creating the archive...
Archive creation complete.
Running postcreation scripts...
Postcreation scripts done.

Running pre-exit scripts...
Pre-exit scripts done.
```

インストールサーバーとして使用されるシステム上で、任意のシステムをインストールするための JumpStart プロファイルを作成します。例えば、`zfs10upflar` アーカイブをインストールする場合は、次のようなプロファイルを使用します。

```
install_type flash_install
archive_location nfs system:/export/jump/zfs10upflar
partitioning explicit
pool rpool auto auto auto mirror c0t1d0s0 c0t0d0s0
```

ZFS ルートファイルシステムのインストール (Oracle Solaris JumpStart インストール)

ZFS ルートファイルシステムまたは UFS ルートファイルシステムをインストールするための JumpStart プロファイルを作成できます。

ZFS 固有のプロファイルには、新しいキーワード `pool` を含める必要があります。`pool` キーワードにより、新規ルートプールがインストールされ、新しいブート環境がデフォルトで作成されます。`bootenv` キーワードと `installbe` キーワード、および `bename` オプションと `dataset` オプションを使用して、ブート環境の名前を指定したり、別の `/var` データセットを作成したりできます。

JumpStart 機能の使用に関する一般的な情報については、『Oracle Solaris 10 9/10 インストールガイド (カスタム JumpStart/上級編)』を参照してください。

ZFS ルートファイルシステムの JumpStart インストールのあとでゾーンを構成し、システムにパッチやアップグレードを適用することを計画している場合は、151 ページの「ゾーンが含まれているシステムを Oracle Solaris Live Upgrade で移行またはアップグレードする (Solaris 10 10/08)」または 156 ページの「ゾーンが含まれているシステムを Oracle Solaris Live Upgrade で移行またはアップグレードする (Solaris 10 5/09 以降)」を参照してください。

ZFS 用の JumpStart キーワード

ZFS 固有のプロファイルでは、次のキーワードを使用できます。

`auto`
: プールのスライス、スワップボリューム、またはダンプボリュームのサイズを自動的に指定します。最小限のサイズを確保できることを確認するために、ディスクのサイズがチェックされます。最小限のサイズを確保できる場合は、ディスクや予約済みスライスのサイズなどの制約を考慮して、できる限り大きいプールサイズが割当てられます。

  たとえば、`c0t0d0s0` を指定する場合、`all` キーワードまたは `auto` キーワードを指定すると、ルートプールのスライスはできる限り大きいサイズで作成されます。または、スライス、スワップボリューム、またはダンプボリュームに特定のサイズを指定することができます。

  ZFS ルートプールに使用する場合、プールには未使用ディスク領域というものは存在しないため、`auto` キーワードは `all` キーワードと同様に機能します。

`bootenv`
: ブート環境の特性を特定します。

  ブート可能な ZFS ルート環境を作成するには、次の `bootenv` キーワード構文を使用します。

  `bootenv installbe bename` *BE-name* [*dataset mount-point*]

  `installbe`
  : `bename` オプションと *BE-name* エントリによって指定された新しい BE を作成し、インストールします。

  `bename` *BE-name*
  : インストールする *BE-name* を指定します。

    `bename` が `pool` キーワードとともに使用されている場合を除き、デフォルトの BE が作成されます。

  `dataset` *mount-point*
  : ルートデータセットとは別の `/var` データセットを指定するには、省略可能なキーワード `dataset` を使用します。現時点では、*mount-point* の値は `/var` に限られています。たとえば、別の `/var` データセットを指定する `bootenv` 構文の行は、次のようになります。

    ```
    bootenv installbe bename zfsroot dataset /var
    ```

`pool`
: 作成する新しいルートプールを定義します。次のキーワード構文を指定する必要があります。

  `pool` *poolname poolsize swapsize dumpsize vdevlist*

poolname 作成するプールの名前を指定します。プールは、指定されたプールサイズ (*size*) および指定された物理デバイス (*vdev*) で作成されます。`poolname` 値には、既存のプールの名前を指定しないようにしてください。既存のプールの名前を指定すると、既存のプールが上書きされます。

poolsize 作成するプールのサイズを指定します。指定できる値は `auto` または `existing` です。`auto` を指定すると、ディスクや予約済みスライスのサイズなどの制約を考慮して、できる限り大きいプールサイズが割当てられます。`existing` を指定すると、その名前で指定された既存のスライスの境界が維持され、上書きされませんg (G バイト) と指定した場合を除き、サイズの単位は M バイトと見なされます。

swapsize 作成するスワップボリュームのサイズを指定します。`auto` 値は、デフォルトのスワップサイズが使用されることを意味します。サイズを指定するには *size* 値を使用します。`g` (G バイト) と指定した場合を除き、サイズの単位は M バイトになります。

dumpsize 作成するダンプボリュームのサイズを指定します。`auto` 値は、デフォルトのスワップサイズが使用されることを意味します。サイズを指定するには *size* 値を使用します。`g` (G バイト) と指定した場合を除き、サイズの単位は M バイトと見なされます。

vdevlist プールの作成に使用する 1 つ以上のデバイスを指定します。*vdevlist* の書式は `zpool create` コマンドの書式と同じです。現時点では、複数のデバイスを指定する場合はミラー化構成だけがサポートされます。*vdevlist* に指定するデバイスは、ルートプール用のスライスにしてください。`any` という値を指定すると、インストールソフトウェアによって適切なデバイスが選択されます。

ディスクはいくつでもミラー化できますが、作成されるプールのサイズは、指定したディスクのうちで最小のディスクによって決定されます。ミラー化されたストレージプールの作成方法の詳細については、69 ページの「ミラー化されたストレージプール構成」を参照してください。

ZFS 用 JumpStart プロファイルの例

この節では、ZFS 固有の JumpStart プロファイルの例を紹介します。

次のプロファイルは、install_type *initial_install* で指定された初期インストールを、pool *newpool* で指定された新しいプールで実行します。auto キーワードにより、この新しいプールのサイズは自動的に、指定されたディスクのサイズになります。auto キーワードにより、スワップ領域とダンプデバイスのサイズは自動的に決められます。また、mirror キーワードにより、*c0t0d0s0* と *c0t1d0s0* で指定されたディスクのミラー化構成になります。ブート環境の特性は bootenv キーワードで設定されます。ここでは、キーワード installbe により新しい BE がインストールされ、*s10-xx* という bename が作成されます。

```
install_type initial_install
pool newpool auto auto auto mirror c0t0d0s0 c0t1d0s0
bootenv installbe bename s10-xx
```

次のプロファイルは、キーワード install_type *initial_install* で指定された初期インストールを、SUNWCall メタクラスタを対象として、newpool という新しいプールで実行します。このプールのサイズは 80G バイトです。このプールは、2G バイトのスワップボリュームと 2G バイトのダンプボリュームを含んで作成されます。また、80G バイトのプールを作成するのに十分なサイズの、利用可能な任意の 2 つのデバイスによるミラー化構成になります。そのような 2 つのデバイスを利用できない場合、インストールは失敗します。ブート環境の特性は bootenv キーワードで設定されます。ここでは、キーワード installbe により新しい BE がインストールされ、*s10–xx* という bename が作成されます。

```
install_type initial_install
cluster SUNWCall
pool newpool 80g 2g 2g mirror any any
bootenv installbe bename s10-xx
```

JumpStart インストールの構文を使用すれば、ZFS ルートプールも含まれているディスク上に、UFS ファイルシステムを維持したり作成したりできます。この構成は本稼働システムにはお勧めできませんが、ノートパソコンなどの小規模なシステムで移行を行う必要がある場合に使用できます。

ZFS の JumpStart に関する問題

ブート可能な ZFS ルートファイルシステムの JumpStart インストールを開始する前に、次の問題を考慮してください。

- 既存の ZFS ストレージプールを JumpStart インストールに使用して、ブート可能な ZFS ルートファイルシステムを作成することはできません。次のような構文を使用して、新しい ZFS ストレージプールを作成する必要があります。

  ```
  pool rpool 20G 4G 4G c0t0d0s0
  ```

- プールの作成には、ディスク全体ではなくスライスを使用する必要があります。詳細は、127 ページの「ZFS をサポートするための Oracle Solaris インストールと Oracle Solaris Live Upgrade の要件」を参照してください。たとえば、次の例の太字部分の構文は使用できません。

```
install_type initial_install
cluster SUNWCall
pool rpool all auto auto mirror c0t0d0 c0t1d0
bootenv installbe bename newBE
```

次の例の太字部分の構文は使用できます。

```
install_type initial_install
cluster SUNWCall
pool rpool all auto auto mirror c0t0d0s0 c0t1d0s0
bootenv installbe bename newBE
```

UFS ルートファイルシステムから ZFS ルートファイルシステムへの移行 (Oracle Solaris Live Upgrade)

Oracle Solaris Live Upgrade の UFS コンポーネント関連機能は引き続き使用可能で、以前の Solaris リリースと同様に動作します。

次の機能も使用可能です。

- UFS ルートファイルシステムを ZFS ルートファイルシステムに移行する場合は、既存の ZFS ストレージプールを -p オプションで指定する必要があります。
- UFS ルートファイルシステムのコンポーネントがさまざまなスライス上に存在する場合、それらは ZFS ルートプールに移行されます。
- Solaris 10 10/08 リリースでは、ゾーンが含まれているシステムを移行することはできますが、サポートされる構成は限られています。Solaris 10 5/09 以降のリリースでは、より多くのゾーン構成がサポートされています。詳細は、次の各節を参照してください。
  - 151 ページの「ゾーンが含まれているシステムを Oracle Solaris Live Upgrade で移行またはアップグレードする (Solaris 10 10/08)」
  - 156 ページの「ゾーンが含まれているシステムを Oracle Solaris Live Upgrade で移行またはアップグレードする (Solaris 10 5/09 以降)」

  ゾーンが含まれていないファイルシステムを移行する場合は、146 ページの「Oracle Solaris Live Upgrade を使用して ZFS ルートファイルシステムに移行する (ゾーンなし)」を参照してください。
- 同じプール内で新しい ZFS BE を作成する場合は、Oracle Solaris Live Upgrade で ZFS のスナップショットとクローンの機能を使用できます。したがって、以前の Solaris リリースと比べてはるかに高速に BE を作成できます。

Oracle Solaris インストールおよび Oracle Solaris Live Upgrade の機能の詳細については、『Oracle Solaris 10 9/10 インストールガイド (Solaris Live Upgrade とアップグレードの計画)』を参照してください。

UFS ルートファイルシステムを ZFS ルートファイルシステムに移行するための基本的な手順は次のとおりです。

- サポートされている任意の SPARC システムまたは x86 システムで、Solaris 10 10/08、Solaris 10 5/09、Solaris 10 10/09、または Oracle Solaris 10 9/10 リリースをインストールするか、標準のアップグレードプログラムを使用して以前の Solaris 10 リリースからアップグレードします。
- Solaris 10 10/08 以降のリリースを実行している場合は、ZFS ルートファイルシステム用の ZFS ストレージプールを作成します。
- Oracle Solaris Live Upgrade を使用して、UFS ルートファイルシステムを ZFS ルートファイルシステムに移行します。
- ZFS BE を `luactivate` コマンドでアクティブにします。

ZFS および Oracle Solaris Live Upgrade の要件については、127 ページの「ZFS をサポートするための Oracle Solaris インストールと Oracle Solaris Live Upgrade の要件」を参照してください。

Oracle Solaris Live Upgrade で ZFS に移行する際の問題

Oracle Solaris Live Upgrade を使用して UFS ルートファイルシステムを ZFS ルートファイルシステムに移行する前に、次に示す問題を確認してください。

- UFS から ZFS ルートファイルシステムへの移行には、Oracle Solaris インストール GUI の標準アップグレードオプションは使用できません。UFS ファイルシステムから移行するには、Oracle Solaris Live Upgrade を使用する必要があります。
- Oracle Solaris Live Upgrade 操作の前に、起動に使用する ZFS ストレージプールを作成する必要があります。また、現時点での起動に関する制限のため、ZFS ルートプールの作成には、ディスク全体ではなくスライスを使用する必要があります。次に例を示します。

  ```
  # zpool create rpool mirror c1t0d0s0 c1t1d0s0
  ```

  新しいプールを作成する前に、プールで使用するディスクに、EFI ラベルではなく SMI (VTOC) ラベルが付いていることを確認してください。ディスクに SMI ラベルを付け直した場合は、ラベル付け処理によってパーティション分割方式が変更されていないことを確認してください。ほとんどの場合、ルートプールに使用するスライスにディスク容量のすべてを割り当てるようにしてください。

- Oracle Solaris Live Upgrade を使用して、UFS BE を ZFS BE から作成することはできません。UFS BE を ZFS BE に移行し、UFS BE を維持する場合は、UFS BE または ZFS BE から起動できます。
- Oracle Solaris Live Upgrade 機能は名前の変更を検出できないため、`zfs rename` コマンドで ZFS BE の名前を変更しないでください。名前を変更すると、以降に実行する `ludelete` などのコマンドが失敗します。したがって、既存の BE を引き続き使用する場合は、ZFS プールまたはファイルシステムの名前を変更しないでください。
- 主 BE のクローンである代替 BE を作成するときに、`-f`、`-x`、`-y`、`-Y`、および `-z` オプションを使用して主 BE のファイルを含めたり除外したりすることはできません。ただし、次の場合には、ファイルを含めるオプションと除外するオプションを使用できます。

```
UFS -> UFS
UFS -> ZFS
ZFS -> ZFS (different pool)
```

- Oracle Solaris Live Upgrade を使用すると、UFS ルートファイルシステムを ZFS ルートファイルシステムにアップグレードできますが、ルート以外のファイルシステムまたは共有ファイルシステムはアップグレードできません。
- `lu` コマンドを使用して ZFS ルートファイルシステムの作成や移行を行うことはできません。

Oracle Solaris Live Upgrade を使用して ZFS ルートファイルシステムに移行する (ゾーンなし)

次の例では、UFS ルートファイルシステムを ZFS ルートファイルシステムに移行する方法を示します。

ゾーンが含まれているシステムを移行またはアップグレードする場合は、次の各節を参照してください。

- 151 ページの「ゾーンが含まれているシステムを Oracle Solaris Live Upgrade で移行またはアップグレードする (Solaris 10 10/08)」
- 156 ページの「ゾーンが含まれているシステムを Oracle Solaris Live Upgrade で移行またはアップグレードする (Solaris 10 5/09 以降)」

例 5–3 Oracle Solaris Live Upgrade を使用して、UFS ルートファイルシステムを ZFS ルートファイルシステムに移行する

次の例では、ZFS ルートファイルシステムの BE を UFS ルートファイルシステムから作成する方法を示します。現在の BE `ufsBE` は `-c` オプションで指定されています。この BE には UFS ルートファイルシステムが含まれています。`-c` オプション (省略可能) を指定しない場合、デフォルトではデバイス名が現在の BE の名前になります。新し

例 5–3 Oracle Solaris Live Upgrade を使用して、UFS ルートファイルシステムを ZFS ルートファイルシステムに移行する (続き)

い BE である zfsBE は、-n オプションによって識別されます。 lucreate 操作の前に ZFS ストレージプールが存在している必要があります。

ZFS ストレージプールは、アップグレード可能かつブート可能にするため、ディスク全体ではなくスライスを使って作成します。新しいプールを作成する前に、プールで使用するディスクに、EFI ラベルではなく SMI (VTOC) ラベルが付いていることを確認してください。ディスクに SMI ラベルを付け直した場合は、ラベル付け処理によってパーティション分割方式が変更されていないことを確認してください。ほとんどの場合、ルートプールに使用するスライスにディスク容量のすべてを割り当てるようにしてください。

```
# zpool create rpool mirror c1t2d0s0 c2t1d0s0
# lucreate -c ufsBE -n zfsBE -p rpool
Analyzing system configuration.
No name for current boot environment.
Current boot environment is named <ufsBE>.
Creating initial configuration for primary boot environment <ufsBE>.
The device </dev/dsk/c1t0d0s0> is not a root device for any boot environment; cannot get BE ID.
PBE configuration successful: PBE name <ufsBE> PBE Boot Device </dev/dsk/c1t0d0s0>.
Comparing source boot environment <ufsBE> file systems with the file
system(s) you specified for the new boot environment. Determining which
file systems should be in the new boot environment.
Updating boot environment description database on all BEs.
Updating system configuration files.
The device </dev/dsk/c1t2d0s0> is not a root device for any boot environment; cannot get BE ID.
Creating configuration for boot environment <zfsBE>.
Source boot environment is <ufsBE>.
Creating boot environment <zfsBE>.
Creating file systems on boot environment <zfsBE>.
Creating <zfs> file system for </> in zone <global> on <rpool/ROOT/zfsBE>.
Populating file systems on boot environment <zfsBE>.
Checking selection integrity.
Integrity check OK.
Populating contents of mount point </>.
Copying.
Creating shared file system mount points.
Creating compare databases for boot environment <zfsBE>.
Creating compare database for file system </rpool/ROOT>.
Creating compare database for file system </>.
Updating compare databases on boot environment <zfsBE>.
Making boot environment <zfsBE> bootable.
Creating boot_archive for /.alt.tmp.b-qD.mnt
updating /.alt.tmp.b-qD.mnt/platform/sun4u/boot_archive
Population of boot environment <zfsBE> successful.
Creation of boot environment <zfsBE> successful.
```

lucreate 操作が完了したら、lustatus コマンドを使用して BE の状態を表示します。次に例を示します。

例 5–3 Oracle Solaris Live Upgrade を使用して、UFS ルートファイルシステムを ZFS ルートファイルシステムに移行する *(続き)*

```
# lustatus
Boot Environment           Is       Active Active    Can    Copy
Name                       Complete Now    On Reboot Delete Status
-------------------------- -------- ------ --------- ------ ----------
ufsBE                      yes      yes    yes       no     -
zfsBE                      yes      no     no        yes    -
```

その後、ZFS コンポーネントのリストを確認します。次に例を示します。

```
# zfs list
NAME                   USED  AVAIL  REFER  MOUNTPOINT
rpool                 7.17G  59.8G  95.5K  /rpool
rpool/ROOT            4.66G  59.8G    21K  /rpool/ROOT
rpool/ROOT/zfsBE      4.66G  59.8G  4.66G  /
rpool/dump               2G  61.8G    16K  -
rpool/swap             517M  60.3G    16K  -
```

次に、luactivate コマンドを使用して、新しい ZFS BE をアクティブにします。次に例を示します。

```
# luactivate zfsBE
A Live Upgrade Sync operation will be performed on startup of boot environment <zfsBE>.

**********************************************************************

The target boot environment has been activated. It will be used when you
reboot. NOTE: You MUST NOT USE the reboot, halt, or uadmin commands. You
MUST USE either the init or the shutdown command when you reboot. If you
do not use either init or shutdown, the system will not boot using the
target BE.

**********************************************************************
.
.
.
Modifying boot archive service
Activation of boot environment <zfsBE> successful.
```

次に、システムを再起動して ZFS BE に切り替えます。

```
# init 6
```

ZFS BE がアクティブになっていることを確認します。

```
# lustatus
Boot Environment           Is       Active Active    Can    Copy
Name                       Complete Now    On Reboot Delete Status
-------------------------- -------- ------ --------- ------ ----------
ufsBE                      yes      no     no        yes    -
zfsBE                      yes      yes    yes       no     -
```

例 5–3 Oracle Solaris Live Upgrade を使用して、UFS ルートファイルシステムを ZFS ルートファイルシステムに移行する *(続き)*

UFS BE に切り替えなおす場合は、ZFS BE が起動されていた間に作成された ZFS ストレージプールをすべて再インポートする必要があります。これらは UFS BE で自動的には使用可能になりません。

UFS BE が必要でなくなった場合は、`ludelete` コマンドで削除できます。

例 5–4 Oracle Solaris Live Upgrade を使用して ZFS BE を ZFS BE から作成する

同じプール内で ZFS BE から ZFS BE を作成する操作には ZFS のスナップショットとクローンの機能が使用されるため、この操作は非常に高速です。現在の BE が同じ ZFS プールにある場合、`-p` オプションは省略されます。

ZFS BE が複数存在する場合は、次のようにして起動元の BE を選択します。

- SPARC: `boot -L` コマンドを使って使用可能な BE を確認し、`boot -Z` コマンドを使って起動元の BE を選択することができます。
- x86: GRUB メニューから BE を選択できます。

詳細は、例 5–9 を参照してください。

```
# lucreate -n zfs2BE
Analyzing system configuration.
No name for current boot environment.
INFORMATION: The current boot environment is not named - assigning name <zfsBE>.
Current boot environment is named <zfsBE>.
Creating initial configuration for primary boot environment <zfsBE>.
The device </dev/dsk/c1t0d0s0> is not a root device for any boot environment; cannot get BE ID.
PBE configuration successful: PBE name <zfsBE> PBE Boot Device </dev/dsk/c1t0d0s0>.
Comparing source boot environment <zfsBE> file systems with the file
system(s) you specified for the new boot environment. Determining which
file systems should be in the new boot environment.
Updating boot environment description database on all BEs.
Updating system configuration files.
Creating configuration for boot environment <zfs2BE>.
Source boot environment is <zfsBE>.
Creating boot environment <zfs2BE>.
Cloning file systems from boot environment <zfsBE> to create boot environment <zfs2BE>.
Creating snapshot for <rpool/ROOT/zfsBE> on <rpool/ROOT/zfsBE@zfs2BE>.
Creating clone for <rpool/ROOT/zfsBE@zfs2BE> on <rpool/ROOT/zfs2BE>.
Setting canmount=noauto for </> in zone <global> on <rpool/ROOT/zfs2BE>.
Population of boot environment <zfs2BE> successful.
Creation of boot environment <zfs2BE> successful.
```

例 5–5 ZFS BE をアップグレードする (`luupgrade`)

ZFS BE を追加のパッケージやパッチでアップグレードすることができます。

基本的な手順は次のとおりです。

例 5–5　ZFS BE をアップグレードする (luupgrade)　　(続き)

- 代替 BE を lucreate コマンドで作成します。
- 代替 BE をアクティブにし、そこから起動します。
- 主 ZFS BE を luupgrade コマンドでアップグレードして、パッケージやパッチを追加します。

```
# lustatus
Boot Environment           Is       Active Active    Can    Copy
Name                       Complete Now    On Reboot Delete Status
-------------------------- -------- ------ --------- ------ ----------
zfsBE                      yes      no     no        yes    -
zfs2BE                     yes      yes    yes       no     -
# luupgrade -p -n zfsBE -s /net/system/export/s10up/Solaris_10/Product SUNWchxge
Validating the contents of the media </net/install/export/s10up/Solaris_10/Product>.
Mounting the BE <zfsBE>.
Adding packages to the BE <zfsBE>.

Processing package instance <SUNWchxge> from </net/install/export/s10up/Solaris_10/Product>

Chelsio N110 10GE NIC Driver(sparc) 11.10.0,REV=2006.02.15.20.41
Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.

This appears to be an attempt to install the same architecture and
version of a package which is already installed.  This installation
will attempt to overwrite this package.

Using </a> as the package base directory.
## Processing package information.
## Processing system information.
   4 package pathnames are already properly installed.
## Verifying package dependencies.
## Verifying disk space requirements.
## Checking for conflicts with packages already installed.
## Checking for setuid/setgid programs.

This package contains scripts which will be executed with super-user
permission during the process of installing this package.

Do you want to continue with the installation of <SUNWchxge> [y,n,?] y
Installing Chelsio N110 10GE NIC Driver as <SUNWchxge>

## Installing part 1 of 1.
## Executing postinstall script.

Installation of <SUNWchxge> was successful.
Unmounting the BE <zfsBE>.
The package add to the BE <zfsBE> completed.
```

ゾーンが含まれているシステムを Oracle Solaris Live Upgrade で移行またはアップグレードする (Solaris 10 10/08)

Solaris 10 10/08 リリースでは、ゾーンが含まれているシステムを Oracle Solaris Live Upgrade で移行することはできますが、サポートされる構成は限られています。Solaris 10 5/09 以降のリリースをインストールする場合や Solaris 10 5/09 以降のリリースにアップグレードする場合は、より多くのゾーン構成がサポートされます。詳細は、156 ページの「ゾーンが含まれているシステムを Oracle Solaris Live Upgrade で移行またはアップグレードする (Solaris 10 5/09 以降)」を参照してください。

この節では、Oracle Solaris Live Upgrade によるアップグレードやパッチの適用が可能になるように、ゾーンが含まれているシステムを構成してインストールする方法について説明します。ゾーンが含まれていない ZFS ルートファイルシステムに移行する場合は、146 ページの「Oracle Solaris Live Upgrade を使用して ZFS ルートファイルシステムに移行する (ゾーンなし)」を参照してください。

Solaris 10 10/08 リリースで、ゾーンが含まれているシステムを移行する場合や、ゾーンが含まれているシステムを構成する場合は、次の手順を確認してください。

- 151 ページの「UFS 上にゾーンルートを持つ UFS ルートファイルシステムを ZFS ルートファイルシステムに移行する方法 (Solaris 10 10/08)」
- 153 ページの「ZFS 上にゾーンルートを持つ ZFS ルートファイルシステムを構成する方法 (Solaris 10 10/08)」
- 155 ページの「ZFS 上にゾーンルートを持つ ZFS ルートファイルシステムにアップグレードまたはパッチを適用する方法 (Solaris 10 10/08)」
- 175 ページの「正常な起動を妨げる ZFS マウントポイントの問題の解決 (Solaris 10 10/08)」

ZFS ルートファイルシステムが含まれているシステムでこれらの推奨手順に従ってゾーンを設定して、そのシステムで Oracle Solaris Live Upgrade を使用できるようにします。

▼ UFS 上にゾーンルートを持つ UFS ルートファイルシステムを ZFS ルートファイルシステムに移行する方法 (Solaris 10 10/08)

次の手順では、ゾーンがインストールされている UFS ルートファイルシステムを、アップグレードやパッチの適用が可能な ZFS ルートファイルシステムおよび ZFS ゾーンルート構成に移行する方法を説明します。

次の手順では、プール名の例として `rpool`、アクティブなブート環境の名前の例として `s10BE*` を使用しています。

1 システムで以前の **Solaris 10** リリースが稼働している場合は、**Solaris 10 10/08** リリースにアップグレードします。

Solaris 10 リリースが稼働しているシステムのアップグレードの詳細については、『Oracle Solaris 10 9/10 インストールガイド (Solaris Live Upgrade とアップグレードの計画)』を参照してください。

2 ルートプールを作成します。

```
# zpool create rpool mirror c0t1d0 c1t1d0
```

ルートプールの要件については、127 ページの「ZFS をサポートするための Oracle Solaris インストールと Oracle Solaris Live Upgrade の要件」を参照してください。

3 **UFS** 環境のゾーンが起動されることを確認します。

4 新しい **ZFS** ブート環境を作成します。

```
# lucreate -n s10BE2 -p rpool
```

このコマンドは、新しいブート環境のデータセットをルートプールに確立し、現在のブート環境をゾーンも含めてそれらのデータセットにコピーします。

5 新しい **ZFS** ブート環境をアクティブにします。

```
# luactivate s10BE2
```

これで、システムでは ZFS ルートファイルシステムが稼働していますが、UFS 上のゾーンルートはまだ UFS ルートファイルシステムにあります。UFS ゾーンをサポートされる ZFS 構成に完全に移行するには、次の手順が必要です。

6 システムを再起動します。

```
# init 6
```

7 ゾーンを **ZFS BE** に移行します。

a. ゾーンを起動します。

b. プール内に別の **ZFS BE** を作成します。

```
# lucreate s10BE3
```

c. 新しいブート環境をアクティブにします。

```
# luactivate s10BE3
```

d. システムを再起動します。

```
# init 6
```

この手順により、ZFS BE とゾーンが起動されることが確認されます。

8 発生する可能性のあるマウントポイントの問題をすべて解決します。

Oracle Solaris Live Upgrade のバグのため、アクティブでないブート環境は起動に失敗する場合があります。これは、ブート環境の ZFS データセットまたはゾーンの ZFS データセットに無効なマウントポイントが含まれているためです。

a. `zfs list` の出力を確認します。

正しくない一時的なマウントポイントを探します。次に例を示します。

```
# zfs list -r -o name,mountpoint rpool/ROOT/s10u6

NAME                                MOUNTPOINT
rpool/ROOT/s10u6                    /.alt.tmp.b-VP.mnt/
rpool/ROOT/s10u6/zones              /.alt.tmp.b-VP.mnt//zones
rpool/ROOT/s10u6/zones/zonerootA    /.alt.tmp.b-VP.mnt/zones/zonerootA
```

ルート ZFS BE (`rpool/ROOT/s10u6`) のマウントポイントは `/` となるべきです。

b. **ZFS BE** とそのデータセットのマウントポイントを設定しなおします。

次に例を示します。

```
# zfs inherit -r mountpoint rpool/ROOT/s10u6
# zfs set mountpoint=/ rpool/ROOT/s10u6
```

c. システムを再起動します。

GRUB メニューまたは OpenBoot PROM プロンプトで、特定のブート環境を起動するオプションが表示されたら、前の手順でマウントポイントを修正したブート環境を選択します。

▼ ZFS 上にゾーンルートを持つ ZFS ルートファイルシステムを構成する方法 (Solaris 10 10/08)

次の手順では、アップグレードやパッチの適用が可能な ZFS ルートファイルシステムおよび ZFS ゾーンルート構成を設定する方法を説明します。この構成では、ZFS ゾーンルートは ZFS データセットとして作成されます。

次の手順では、プール名の例として `rpool`、アクティブなブート環境の名前の例として `s10BE` を使用しています。ゾーンのデータセットの名前には、正当なデータセット名であればどのようなものでも使用できます。次の例では、ゾーンのデータセットの名前は `zones` になっています。

1 **Solaris** 対話式テキストインストーラまたは **Solaris JumpStart** インストール方法を使用して、システムを **ZFS** ルートでインストールします。

初期インストールまたは Solaris JumpStart インストールを使用して ZFS ルートファイルシステムをインストールする方法については、130 ページの「ZFS ルートファイル

システムのインストール (初期インストール)」または 140 ページの「ZFS ルートファイルシステムのインストール (Oracle Solaris JumpStart インストール)」を参照してください。

2 新しく作成したルートプールからシステムを起動します。

3 ゾーンルートをまとめるためのデータセットを作成します。

次に例を示します。

```
# zfs create -o canmount=noauto rpool/ROOT/s10BE/zones
```

canmount プロパティーに noauto 値を設定すると、Oracle Solaris Live Upgrade とシステムの起動コードの明示的なアクションによらないかぎり、データセットはマウントされなくなります。

4 新しく作成したゾーンデータセットをマウントします。

```
# zfs mount rpool/ROOT/s10BE/zones
```

データセットは /zones にマウントされます。

5 各ゾーンルートのデータセットを作成し、マウントします。

```
# zfs create -o canmount=noauto rpool/ROOT/s10BE/zones/zonerootA
# zfs mount rpool/ROOT/s10BE/zones/zonerootA
```

6 ゾーンルートディレクトリに適切なアクセス権を設定します。

```
# chmod 700 /zones/zonerootA
```

7 ゾーンを設定して、ゾーンパスを次のように設定します。

```
# zonecfg -z zoneA
    zoneA: No such zone configured
    Use 'create' to begin configuring a new zone.
    zonecfg:zoneA> create
    zonecfg:zoneA> set zonepath=/zones/zonerootA
```

次の構文を使用して、システムの起動時にゾーンが自動的に起動するように設定できます。

```
zonecfg:zoneA> set autoboot=true
```

8 ゾーンをインストールします。

```
# zoneadm -z zoneA install
```

9 ゾーンを起動します。

```
# zoneadm -z zoneA boot
```

▼ ZFS 上にゾーンルートを持つ ZFS ルートファイルシステムにアップグレードまたはパッチを適用する方法 (Solaris 10 10/08)

ZFS 上にゾーンルートを持つ ZFS ルートファイルシステムにアップグレードやパッチを適用する必要がある場合は、次の手順を使用します。このような更新には、システムのアップグレードの場合と、パッチの適用の場合があります。

次の手順では、アップグレードまたはパッチを適用するブート環境の名前の例として newBE を使用しています。

1 アップグレードまたはパッチを適用するブート環境を作成します。

```
# lucreate -n newBE
```

すべてのゾーンを含め、既存のブート環境が複製されます。元のブート環境の各データセットに対してデータセットが 1 つずつ作成されます。新しいデータセットは、現在のルートプールと同じプールに作成されます。

2 次のいずれかを選択して、システムをアップグレードするか新しいブート環境にパッチを適用します。

- システムをアップグレードします。

  ```
  # luupgrade -u -n newBE -s /net/install/export/s10u7/latest
  ```

  ここで、-s オプションは Solaris インストールメディアの場所を指定します。
- 新しいブート環境にパッチを適用します。

  ```
  # luupgrade -t -n newBE -t -s /patchdir 139147-02 157347-14
  ```

3 新しいブート環境をアクティブにします。

```
# luactivate newBE
```

4 新たにアクティブにしたブート環境から起動します。

```
# init 6
```

5 発生する可能性のあるマウントポイントの問題をすべて解決します。

Oracle Solaris Live Upgrade 機能のバグのため、アクティブでないブート環境は起動に失敗する場合があります。これは、ブート環境の ZFS データセットまたはゾーンの ZFS データセットに無効なマウントポイントが含まれているためです。

a. `zfs list` の出力を確認します。

正しくない一時的なマウントポイントを探します。次に例を示します。

```
# zfs list -r -o name,mountpoint rpool/ROOT/newBE

NAME                              MOUNTPOINT
rpool/ROOT/newBE                  /.alt.tmp.b-VP.mnt/
rpool/ROOT/newBE/zones            /.alt.tmp.b-VP.mnt/zones
```

```
rpool/ROOT/newBE/zones/zonerootA   /.alt.tmp.b-VP.mnt/zones/zonerootA
```

ルート ZFS BE (rpool/ROOT/newBE) のマウントポイントは / となるべきです。

b. **ZFS BE** とそのデータセットのマウントポイントを設定しなおします。

次に例を示します。

```
# zfs inherit -r mountpoint rpool/ROOT/newBE
# zfs set mountpoint=/ rpool/ROOT/newBE
```

c. システムを再起動します。

GRUB メニューまたは OpenBoot PROM プロンプトで、特定のブート環境を起動するオプションが表示されたら、前の手順でマウントポイントを修正したブート環境を選択します。

ゾーンが含まれているシステムを Oracle Solaris Live Upgrade で移行またはアップグレードする (Solaris 10 5/09 以降)

Solaris 10 10/08 以降のリリースでは、Oracle Solaris Live Upgrade 機能を使用して、ゾーンが含まれているシステムを移行またはアップグレードすることができます。Solaris 10 5/09 以降のリリースでは、疎ルートゾーン構成と完全ルートゾーン構成も Live Upgrade でサポートされます。

この節では、Solaris 10 5/09 以降のリリースで Oracle Solaris Live Upgrade によるアップグレードやパッチの適用が可能になるようにゾーンが含まれているシステムを構成する方法について説明します。ゾーンが含まれていない ZFS ルートファイルシステムに移行する場合は、146 ページの「Oracle Solaris Live Upgrade を使用して ZFS ルートファイルシステムに移行する (ゾーンなし)」を参照してください。

Solaris 10 5/09 以降のリリースで ZFS とゾーンに Oracle Solaris Live Upgrade を使用する場合は、次の点を考慮してください。

- Solaris 10 5/09 以降のリリースでサポートされるゾーン構成で Oracle Solaris Live Upgrade を使用するには、まず標準のアップグレードプログラムを使って Solaris 10 5/09 以降のリリースにシステムをアップグレードする必要があります。
- その後、Oracle Solaris Live Upgrade を使用して、ゾーンルートを持つ UFS ルートファイルシステムを ZFS ルートファイルシステムに移行するか、ZFS ルートファイルシステムとゾーンルートにアップグレードやパッチを適用することができます。
- サポートされていないゾーン構成を、以前の Solaris 10 リリースから直接 Solaris 10 5/09 以降のリリースに移行することはできません。

Solaris 10 5/09 以降のリリースでゾーンを含むシステムを移行または構成する場合は、次の情報を確認してください。

- 157 ページの「サポートされているゾーンルート構成を持つ ZFS の情報 (Solaris 10 5/09 以降)」
- 158 ページの「ZFS ルートファイルシステムとゾーンルートを持つ ZFS BE を作成する方法 (Solaris 10 5/09 以降)」
- 160 ページの「ゾーンルートを持つ ZFS ルートファイルシステムにアップグレードまたはパッチを適用する方法 (Solaris 10 5/09 以降)」
- 163 ページの「ゾーンルートを持つ UFS ルートファイルシステムを ZFS ルートファイルシステムに移行する方法 (Solaris 10 5/09 以降)」

サポートされているゾーンルート構成を持つ ZFS の情報 (Solaris 10 5/09 以降)

ゾーンが含まれているシステムを Oracle Solaris Live Upgrade を使用して移行またはアップグレードする前に、サポートされているゾーン構成を確認してください。

- **UFS** ルートファイルシステムを **ZFS** ルートファイルシステムに移行する – 次のゾーンルート構成がサポートされています。
  - UFS ルートファイルシステムのディレクトリ内
  - UFS ルートファイルシステムのマウントポイントのサブディレクトリ内
  - UFS ルートファイルシステムのディレクトリ内または UFS ルートファイルシステムのマウントポイントのサブディレクトリ内にゾーンルートを含む UFS ルートファイルシステム、およびゾーンルートを含む ZFS 非ルートプール

  次の UFS/ゾーン構成はサポートされません: ゾーンルートをマウントポイントとして持つ UFS ルートファイルシステム。
- **ZFS** ルートファイルシステムを移行またはアップグレードする – 次のゾーンルート構成がサポートされています。
  - ZFS ルートプールのデータセット内。場合により、Oracle Solaris Live Upgrade 操作の前にゾーンルートのデータセットが用意されていないときは、ゾーンルートのデータセット (`zoneds`) が Oracle Solaris Live Upgrade によって作成されます。
  - ZFS ルートファイルシステムのサブディレクトリ内
  - ZFS ルートファイルシステムの外部にあるデータセット内
  - ZFS ルートファイルシステムの外部にあるデータセットのサブディレクトリ内
  - 非ルートプールのデータセット内。次の例では、`zonepool/zones` はゾーンルートを含むデータセットであり、`rpool` は ZFS BE を含んでいます。

    ```
    zonepool
    zonepool/zones
    zonepool/zones/myzone
    ```

```
rpool
rpool/ROOT
rpool/ROOT/myBE
```

次の構文を使用すると、Oracle Solaris Live Upgrade によって zonepool のゾーンおよび rpool BE のスナップショットが作成され、複製が行われます。

```
# lucreate -n newBE
```

newBE BE が rpool/ROOT/newBE 内に作成されます。newBE をアクティブにすると、 zonepool のコンポーネントにアクセスできるようになります。

前述の例では、/zonepool/zones がサブディレクトリーであり、別個のデータセットではない場合には、それは、ルートプール rpool のコンポーネントとして、Live Upgrade によって移行されます。

- **UFS** および **ZFS** のゾーンの移行またはアップグレードに関する情報 – UFS 環境または ZFS 環境の移行またはアップグレードに影響を与える可能性のある次の考慮事項を確認してください。
  - 151 ページの「ゾーンが含まれているシステムを Oracle Solaris Live Upgrade で移行またはアップグレードする (Solaris 10 10/08)」 の説明に従って Solaris 10 10/08 リリースでゾーンを構成し、Solaris 10 5/09 以降のリリースにアップグレードした場合は、ZFS ルートファイルシステムへの移行や、Solaris Live Upgrade による Solaris 10 5/09 以降のリリースへのアップグレードが可能なはずです。
  - zones/zone1 や zones/zone1/zone2 のような入れ子のディレクトリ内にゾーンルートを作成しないでください。そうしないと、起動時にマウントが失敗する可能性があります。

▼ ZFS ルートファイルシステムとゾーンルートを持つ ZFS BE を作成する方法 (Solaris 10 5/09 以降)

Solaris 10 5/09 以降のリリースの初期インストールを実行したあとで、この手順を使用して ZFS ルートファイルシステムを作成します。また、luupgrade 機能を使って Solaris 10 5/09 以降のリリースに ZFS ルートファイルシステムをアップグレードしたあとも、この手順を使用します。この手順を使用して作成した ZFS BE には、あとでアップグレードやパッチを適用できます。

次の手順で例として使用する Oracle Solaris 10 9/10 システムは、ZFS ルートファイルシステムと、/rpool/zones にゾーンルートのデータセットを持っています。zfs2BE という名前の ZFS BE が作成されますが、あとでそのアップグレードやパッチの適用を行うことができます。

1 既存の **ZFS** ファイルシステムを確認します。

```
# zfs list
NAME                   USED  AVAIL  REFER  MOUNTPOINT
```

```
rpool                  7.26G  59.7G    98K  /rpool
rpool/ROOT             4.64G  59.7G    21K  legacy
rpool/ROOT/zfsBE       4.64G  59.7G  4.64G  /
rpool/dump             1.00G  59.7G  1.00G  -
rpool/export             44K  59.7G    23K  /export
rpool/export/home        21K  59.7G    21K  /export/home
rpool/swap              1G  60.7G    16K  -
rpool/zones            633M  59.7G   633M  /rpool/zones
```

2 ゾーンがインストールされ起動されていることを確認します。

```
# zoneadm list -cv
  ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              native   shared
   2 zfszone          running    /rpool/zones                   native   shared
```

3 **ZFS BE** を作成します。

```
# lucreate -n zfs2BE
Analyzing system configuration.
No name for current boot environment.
INFORMATION: The current boot environment is not named - assigning name <zfsBE>.
Current boot environment is named <zfsBE>.
Creating initial configuration for primary boot environment <zfsBE>.
The device </dev/dsk/c1t0d0s0> is not a root device for any boot environment; cannot get BE ID.
PBE configuration successful: PBE name <zfsBE> PBE Boot Device </dev/dsk/c1t0d0s0>.
Comparing source boot environment <zfsBE> file systems with the file
system(s) you specified for the new boot environment. Determining which
file systems should be in the new boot environment.
Updating boot environment description database on all BEs.
Updating system configuration files.
Creating configuration for boot environment <zfs2BE>.
Source boot environment is <zfsBE>.
Creating boot environment <zfs2BE>.
Cloning file systems from boot environment <zfsBE> to create boot environment <zfs2BE>.
Creating snapshot for <rpool/ROOT/zfsBE> on <rpool/ROOT/zfsBE@zfs2BE>.
Creating clone for <rpool/ROOT/zfsBE@zfs2BE> on <rpool/ROOT/zfs2BE>.
Setting canmount=noauto for </> in zone <global> on <rpool/ROOT/zfs2BE>.
Population of boot environment <zfs2BE> successful.
Creation of boot environment <zfs2BE> successful.
```

4 **ZFS BE** をアクティブにします。

```
# lustatus
Boot Environment           Is       Active Active    Can    Copy
Name                       Complete Now    On Reboot Delete Status
-------------------------- -------- ------ --------- ------ ----------
zfsBE                      yes      yes    yes       no     -
zfs2BE                     yes      no     no        yes    -
# luactivate zfs2BE
A Live Upgrade Sync operation will be performed on startup of boot environment <zfs2BE>.
.
.
.
# init 6
```

5 新しい **BE** に **ZFS** ファイルシステムとゾーンが作成されていることを確認します。

```
# zfs list
NAME                              USED  AVAIL  REFER  MOUNTPOINT
rpool                            7.38G  59.6G    98K  /rpool
rpool/ROOT                       4.72G  59.6G    21K  legacy
rpool/ROOT/zfs2BE                4.72G  59.6G  4.64G  /
rpool/ROOT/zfs2BE@zfs2BE         74.0M      -  4.64G  -
rpool/ROOT/zfsBE                 5.45M  59.6G  4.64G  /.alt.zfsBE
rpool/dump                       1.00G  59.6G  1.00G  -
rpool/export                       44K  59.6G    23K  /export
rpool/export/home                  21K  59.6G    21K  /export/home
rpool/swap                          1G  60.6G    16K  -
rpool/zones                      17.2M  59.6G   633M  /rpool/zones
rpool/zones-zfsBE                 653M  59.6G   633M  /rpool/zones-zfsBE
rpool/zones-zfsBE@zfs2BE         19.9M      -   633M  -
# zoneadm list -cv
  ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              native   shared
   - zfszone          installed  /rpool/zones                   native   shared
```

▼ ゾーンルートを持つ **ZFS** ルートファイルシステムにアップグレードまたはパッチを適用する方法 **(Solaris 10 5/09** 以降**)**

Solaris 10 5/09 以降のリリースで、ゾーンルートを持つ ZFS ルートファイルシステムにアップグレードやパッチを適用する必要がある場合は、次の手順を使用します。このような更新には、システムのアップグレードの場合と、パッチの適用の場合があります。

次の手順では、アップグレードまたはパッチを適用するブート環境の名前の例として `zfs2BE` を使用しています。

1 既存の **ZFS** ファイルシステムを確認します。

```
# zfs list
NAME                              USED  AVAIL  REFER  MOUNTPOINT
rpool                            7.38G  59.6G   100K  /rpool
rpool/ROOT                       4.72G  59.6G    21K  legacy
rpool/ROOT/zfs2BE                4.72G  59.6G  4.64G  /
rpool/ROOT/zfs2BE@zfs2BE         75.0M      -  4.64G  -
rpool/ROOT/zfsBE                 5.46M  59.6G  4.64G  /
rpool/dump                       1.00G  59.6G  1.00G  -
rpool/export                       44K  59.6G    23K  /export
rpool/export/home                  21K  59.6G    21K  /export/home
rpool/swap                          1G  60.6G    16K  -
rpool/zones                      22.9M  59.6G   637M  /rpool/zones
rpool/zones-zfsBE                 653M  59.6G   633M  /rpool/zones-zfsBE
rpool/zones-zfsBE@zfs2BE         20.0M      -   633M  -
```

2 ゾーンがインストールされ起動されていることを確認します。

```
# zoneadm list -cv
  ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              native   shared
   5 zfszone          running    /rpool/zones                   native   shared
```

3 アップグレードまたはパッチを適用する **ZFS BE** を作成します。

```
# lucreate -n zfs2BE
Analyzing system configuration.
Comparing source boot environment <zfsBE> file systems with the file
system(s) you specified for the new boot environment. Determining which
file systems should be in the new boot environment.
Updating boot environment description database on all BEs.
Updating system configuration files.
Creating configuration for boot environment <zfs2BE>.
Source boot environment is <zfsBE>.
Creating boot environment <zfs2BE>.
Cloning file systems from boot environment <zfsBE> to create boot environment <zfs2BE>.
Creating snapshot for <rpool/ROOT/zfsBE> on <rpool/ROOT/zfsBE@zfs2BE>.
Creating clone for <rpool/ROOT/zfsBE@zfs2BE> on <rpool/ROOT/zfs2BE>.
Setting canmount=noauto for </> in zone <global> on <rpool/ROOT/zfs2BE>.
Creating snapshot for <rpool/zones> on <rpool/zones@zfs10092BE>.
Creating clone for <rpool/zones@zfs2BE> on <rpool/zones-zfs2BE>.
Population of boot environment <zfs2BE> successful.
Creation of boot environment <zfs2BE> successful.
```

4 次のいずれかを選択して、システムをアップグレードするか新しいブート環境にパッチを適用します。

- システムをアップグレードします。

  ```
  # luupgrade -u -n zfs2BE -s /net/install/export/s10up/latest
  ```

  ここで、-s オプションは Solaris インストールメディアの場所を指定します。

  この処理には非常に長い時間がかかることがあります。

  luupgrade 処理の詳細な例については、例 5–6 を参照してください。

- 新しいブート環境にパッチを適用します。

  ```
  # luupgrade -t -n zfs2BE -t -s /patchdir patch-id-02 patch-id-04
  ```

5 新しいブート環境をアクティブにします。

```
# lustatus
Boot Environment           Is       Active Active    Can    Copy
Name                       Complete Now    On Reboot Delete Status
-------------------------- -------- ------ --------- ------ ----------
zfsBE                      yes      yes    yes       no     -
zfs2BE                     yes      no     no        yes    -
# luactivate zfs2BE
A Live Upgrade Sync operation will be performed on startup of boot environment <zfs2BE>.
.
.
.
```

6 新たにアクティブにしたブート環境から起動します。

```
# init 6
```

例 5–6 ゾーンルートを持つ ZFS ルートファイルシステムを Oracle Solaris 10 9/10 の ZFS ルートファイルシステムにアップグレードする

この例では、Solaris 10 10/09 システムに作成された、非ルートプール内に ZFS ルートファイルシステムとゾーンルートを持つ ZFS BE (zfsBE) を、Oracle Solaris 10 9/10 リリースにアップグレードします。この処理には長い時間がかかることがあります。その後、アップグレードした BE (zfs2BE) をアクティブにします。アップグレードを行う前に、ゾーンがインストールされ起動されていることを確認してください。

この例では、zonepool プール、/zonepool/zones データセット、および zfszone ゾーンが、次のようにして作成されます。

```
# zpool create zonepool mirror c2t1d0 c2t5d0
# zfs create zonepool/zones
# chmod 700 zonepool/zones
# zonecfg -z zfszone
zfszone: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:zfszone> create
zonecfg:zfszone> set zonepath=/zonepool/zones
zonecfg:zfszone> verify
zonecfg:zfszone> exit
# zoneadm -z zfszone install
cannot create ZFS dataset zonepool/zones: dataset already exists
Preparing to install zone <zfszone>.
Creating list of files to copy from the global zone.
Copying <8960> files to the zone.
.
.
.

# zoneadm list -cv
  ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              native   shared
   2 zfszone          running    /zonepool/zones                native   shared

# lucreate -n zfsBE
.
.
.
# luupgrade -u -n zfsBE -s /net/install/export/s10up/latest
40410 blocks
miniroot filesystem is <lofs>
Mounting miniroot at </net/system/export/s10up/latest/Solaris_10/Tools/Boot>
Validating the contents of the media </net/system/export/s10up/latest>.
The media is a standard Solaris media.
The media contains an operating system upgrade image.
The media contains <Solaris> version <10>.
Constructing upgrade profile to use.
Locating the operating system upgrade program.
Checking for existence of previously scheduled Live Upgrade requests.
Creating upgrade profile for BE <zfsBE>.
Determining packages to install or upgrade for BE <zfsBE>.
```

```
Performing the operating system upgrade of the BE <zfsBE>.
CAUTION: Interrupting this process may leave the boot environment unstable
or unbootable.
Upgrading Solaris: 100% completed
Installation of the packages from this media is complete.
Updating package information on boot environment <zfsBE>.
Package information successfully updated on boot environment <zfsBE>.
Adding operating system patches to the BE <zfsBE>.
The operating system patch installation is complete.
INFORMATION: The file </var/sadm/system/logs/upgrade_log> on boot
environment <zfsBE> contains a log of the upgrade operation.
INFORMATION: The file </var/sadm/system/data/upgrade_cleanup> on boot
environment <zfsBE> contains a log of cleanup operations required.
INFORMATION: Review the files listed above. Remember that all of the files
are located on boot environment <zfsBE>. Before you activate boot
environment <zfsBE>, determine if any additional system maintenance is
required or if additional media of the software distribution must be
installed.
The Solaris upgrade of the boot environment <zfsBE> is complete.
Installing failsafe
Failsafe install is complete.
# luactivate zfsBE
# init 6
# lustatus
Boot Environment           Is       Active Active    Can    Copy
Name                       Complete Now    On Reboot Delete Status
-------------------------- -------- ------ --------- ------ ----------
zfsBE                      yes      no     no        yes    -
zfs2BE                     yes      yes    yes       no     -
# zoneadm list -cv
  ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              native   shared
   - zfszone          installed  /zonepool/zones                native   shared
```

▼ ゾーンルートを持つ UFS ルートファイルシステムを ZFS ルートファイルシステムに移行する方法 (Solaris 10 5/09 以降)

UFS ルートファイルシステムとゾーンルートを持つシステムを Solaris 10 5/09 以降のリリースに移行するには、次の手順を使用します。その後、Oracle Solaris Live Upgrade を使用して ZFS BE を作成します。

次の手順では、UFS BE の名前の例として `c0t1d0s0`、UFS ゾーンルートとして `zonepool/zfszone`、ZFS ルート BE として `zfsBE` を使用しています。

1 システムで以前の **Solaris 10** リリースが稼働している場合は、**Solaris 10 5/09** 以降のリリースにアップグレードします。

Solaris 10 リリースが稼働しているシステムのアップグレードについては、『Oracle Solaris 10 9/10 インストールガイド (Solaris Live Upgrade とアップグレードの計画)』を参照してください。

2 ルートプールを作成します。

ルートプールの要件については、127 ページの「ZFS をサポートするための Oracle Solaris インストールと Oracle Solaris Live Upgrade の要件」を参照してください。

3 **UFS** 環境のゾーンが起動されることを確認します。

```
# zoneadm list -cv
  ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              native   shared
   2 zfszone          running    /zonepool/zones                native   shared
```

4 新しい **ZFS** ブート環境を作成します。

```
# lucreate -c c1t1d0s0 -n zfsBE -p rpool
```

このコマンドは、新しいブート環境のデータセットをルートプールに確立し、現在のブート環境をゾーンも含めてそれらのデータセットにコピーします。

5 新しい **ZFS** ブート環境をアクティブにします。

```
# lustatus
Boot Environment           Is       Active Active    Can    Copy
Name                       Complete Now    On Reboot Delete Status
-------------------------- -------- ------ --------- ------ ----------
c1t1d0s0                   yes      no     no        yes    -
zfsBE                      yes      yes    yes       no     -         #
luactivate zfsBE
A Live Upgrade Sync operation will be performed on startup of boot environment <zfsBE>.
.
.
.
```

6 システムを再起動します。

```
# init 6
```

7 新しい **BE** に **ZFS** ファイルシステムとゾーンが作成されていることを確認します。

```
# zfs list
NAME                                USED  AVAIL  REFER  MOUNTPOINT
rpool                              6.17G  60.8G    98K  /rpool
rpool/ROOT                         4.67G  60.8G    21K  /rpool/ROOT
rpool/ROOT/zfsBE                   4.67G  60.8G  4.67G  /
rpool/dump                         1.00G  60.8G  1.00G  -
rpool/swap                          517M  61.3G    16K  -
zonepool                            634M  7.62G    24K  /zonepool
zonepool/zones                      270K  7.62G   633M  /zonepool/zones
zonepool/zones-c1t1d0s0             634M  7.62G   633M  /zonepool/zones-c1t1d0s0
zonepool/zones-c1t1d0s0@zfsBE       262K      -   633M  -
# zoneadm list -cv
  ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              native   shared
   - zfszone          installed  /zonepool/zones                native   shared
```

例 5–7 ゾーンルートを持つ UFS ルートファイルシステムを ZFS ルートファイルシステムに移行する

この例では、UFS ルートファイルシステムとゾーンルート (/uzone/ufszone) およびルート以外の ZFS プール (pool) とゾーンルート (/pool/zfszone) を持つ Oracle Solaris 10 9/10 システムを、ZFS ルートファイルシステムに移行します。移行を行う前に、ZFS ルートプールが作成されていることと、ゾーンがインストールされ起動されていることを確認してください。

```
# zoneadm list -cv
  ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              native   shared
   2 ufszone          running    /uzone/ufszone                 native   shared
   3 zfszone          running    /pool/zones/zfszone            native   shared

# lucreate -c ufsBE -n zfsBE -p rpool
Analyzing system configuration.
No name for current boot environment.
Current boot environment is named <zfsBE>.
Creating initial configuration for primary boot environment <zfsBE>.
The device </dev/dsk/c1t0d0s0> is not a root device for any boot environment; cannot get BE ID.
PBE configuration successful: PBE name <ufsBE> PBE Boot Device </dev/dsk/c1t0d0s0>.
Comparing source boot environment <ufsBE> file systems with the file
system(s) you specified for the new boot environment. Determining which
file systems should be in the new boot environment.
Updating boot environment description database on all BEs.
Updating system configuration files.
The device </dev/dsk/c1t1d0s0> is not a root device for any boot environment; cannot get BE ID.
Creating configuration for boot environment <zfsBE>.
Source boot environment is <ufsBE>.
Creating boot environment <zfsBE>.
Creating file systems on boot environment <zfsBE>.
Creating <zfs> file system for </> in zone <global> on <rpool/ROOT/zfsBE>.
Populating file systems on boot environment <zfsBE>.
Checking selection integrity.
Integrity check OK.
Populating contents of mount point </>.
Copying.
Creating shared file system mount points.
Copying root of zone <ufszone> to </.alt.tmp.b-EYd.mnt/uzone/ufszone>.
Creating snapshot for <pool/zones/zfszone> on <pool/zones/zfszone@zfsBE>.
Creating clone for <pool/zones/zfszone@zfsBE> on <pool/zones/zfszone-zfsBE>.
Creating compare databases for boot environment <zfsBE>.
Creating compare database for file system </rpool/ROOT>.
Creating compare database for file system </>.
Updating compare databases on boot environment <zfsBE>.
Making boot environment <zfsBE> bootable.
Creating boot_archive for /.alt.tmp.b-DLd.mnt
updating /.alt.tmp.b-DLd.mnt/platform/sun4u/boot_archive
Population of boot environment <zfsBE> successful.
Creation of boot environment <zfsBE> successful.
# lustatus
Boot Environment           Is       Active Active    Can    Copy
Name                       Complete Now    On Reboot Delete Status
-------------------------- -------- ------ --------- ------ ----------
```

```
ufsBE                      yes      yes    yes       no     -
zfsBE                      yes      no     no        yes    -
# luactivate zfsBE
.
.
.
# init 6
.
.
.
# zfs list
NAME                                 USED  AVAIL  REFER  MOUNTPOINT
pool                                 628M  66.3G    19K  /pool
pool/zones                           628M  66.3G    20K  /pool/zones
pool/zones/zfszone                  75.5K  66.3G   627M  /pool/zones/zfszone
pool/zones/zfszone-ufsBE             628M  66.3G   627M  /pool/zones/zfszone-ufsBE
pool/zones/zfszone-ufsBE@zfsBE        98K      -   627M  -
rpool                               7.76G  59.2G    95K  /rpool
rpool/ROOT                          5.25G  59.2G    18K  /rpool/ROOT
rpool/ROOT/zfsBE                    5.25G  59.2G  5.25G  /
rpool/dump                          2.00G  59.2G  2.00G  -
rpool/swap                           517M  59.7G    16K  -
# zoneadm list -cv
  ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              native   shared
   - ufszone          installed  /uzone/ufszone                 native   shared
   - zfszone          installed  /pool/zones/zfszone            native   shared
```

スワップデバイスおよびダンプデバイスの ZFS サポート

Solaris OS の初期インストール中、あるいは UFS ファイルシステムからの Oracle Solaris Live Upgrade 移行の実行後に、ZFS ルートプールの ZFS ボリュームにスワップ領域が作成されます。次に例を示します。

```
# swap -l
swapfile                  dev  swaplo  blocks   free
/dev/zvol/dsk/rpool/swap 256,1      16 4194288 4194288
```

Solaris OS の初期インストールまたは UFS ファイルシステムからの Oracle Solaris Live Upgrade の際に、ZFS ルートプールの ZFS ボリュームにダンプデバイスが作成されます。ダンプデバイスは一般に、インストール時に自動的に設定されるため、管理の必要はありません。次に例を示します。

```
# dumpadm
      Dump content: kernel pages
       Dump device: /dev/zvol/dsk/rpool/dump (dedicated)
Savecore directory: /var/crash/t2000
  Savecore enabled: yes
   Save compressed: on
```

ダンプデバイスを無効にして削除した場合、ダンプデバイスを作成し直したあと、dumpadm コマンドを使ってデバイスを有効にする必要があります。ほとんどの場合、zfs コマンドを使ってダンプデバイスのサイズを調整するだけですみます。

インストールプログラムによって作成されるスワップボリュームとダンプボリュームのサイズについては、127 ページの「ZFS をサポートするための Oracle Solaris インストールと Oracle Solaris Live Upgrade の要件」を参照してください。

スワップボリュームのサイズとダンプボリュームのサイズはどちらも、インストール中またはインストール後に調整することができます。詳細は、167 ページの「ZFS スワップデバイスおよびダンプデバイスのサイズを調整する」を参照してください。

ZFS のスワップデバイスとダンプデバイスを操作する場合には、次の問題を考慮してください。

- スワップ領域とダンプデバイスには別個の ZFS ボリュームを使用する必要があります。
- 現時点では、ZFS ファイルシステムでスワップファイルを使用することはできません。
- システムのインストール後またはアップグレード後にスワップ領域やダンプデバイスを変更する必要がある場合は、以前の Solaris 10 リリースと同様に swap コマンドと dumpadm コマンドを使用します。詳細は、『Solaris のシステム管理 (デバイスとファイルシステム)』の第 20 章「追加スワップ空間の構成 (手順)」および『Solaris のシステム管理 (上級編)』の第 17 章「システムクラッシュ情報の管理 (手順)」を参照してください。

詳細は、次の章を参照してください。

- 167 ページの「ZFS スワップデバイスおよびダンプデバイスのサイズを調整する」
- 169 ページの「ZFS ダンプデバイスの問題のトラブルシューティング」

ZFS スワップデバイスおよびダンプデバイスのサイズを調整する

ZFS ルートのインストールでは、スワップデバイスとダンプデバイスのサイズの決定方法が異なるため、インストール前、インストール中、またはインストール後にスワップデバイスとダンプデバイスのサイズの調整が必要になることがあります。

- スワップボリュームとダンプボリュームのサイズは、初期インストール時に調整することができます。詳細は、例 5–1 を参照してください。

- Oracle Solaris Live Upgrade 操作を実行する前に、スワップボリュームとダンプボリュームを作成し、それらのサイズを設定することができます。次に例を示します。

  1. ストレージプールを作成します。

     ```
     # zpool create rpool mirror c0t0d0s0 c0t1d0s0
     ```

  2. ダンプデバイスを作成します。

     ```
     # zfs create -V 2G rpool/dump
     ```

  3. ダンプデバイスを有効にします。

     ```
     # dumpadm -d /dev/zvol/dsk/rpool/dump
           Dump content: kernel pages
            Dump device: /dev/zvol/dsk/rpool/dump (dedicated)
     Savecore directory: /var/crash/t2000
       Savecore enabled: yes
        Save compressed: on
     ```

  4. 次のいずれかを選択してスワップ領域を作成します。

     - SPARC: スワップ領域を作成します。ブロックサイズを 8K バイトに設定します。

       ```
       # zfs create -V 2G -b 8k rpool/swap
       ```

     - x86: スワップ領域を作成します。ブロックサイズを 4K バイトに設定します。

       ```
       # zfs create -V 2G -b 4k rpool/swap
       ```

  5. 新しいスワップデバイスを追加または変更したときは、スワップボリュームを有効にしてください。
  6. スワップボリュームのエントリを `/etc/vfstab` ファイルに追加します。

  Oracle Solaris Live Upgrade では、既存のスワップボリュームとダンプボリュームのサイズは変更されません。

- ダンプデバイスの `volsize` プロパティーは、システムのインストール後に再設定することができます。次に例を示します。

  ```
  # zfs set volsize=2G rpool/dump
  # zfs get volsize rpool/dump
  NAME         PROPERTY  VALUE       SOURCE
  rpool/dump   volsize   2G          -
  ```

- スワップボリュームのサイズを変更することはできますが、CR 6765386 を組み込むまでは、まずスワップデバイスを削除することをお勧めします。そのあとで再作成してください。次に例を示します。

  ```
  # swap -d /dev/zvol/dsk/rpool/swap
  # zfs volsize=2G rpool/swap
  # swap -a /dev/zvol/dsk/rpool/swap
  ```

アクティブなシステムからスワップデバイスを削除する方法については、次のサイトを参照してください。

http://www.solarisinternals.com/wiki/index.php/ZFS_Troubleshooting_Guide

- JumpStart プロファイルのスワップボリュームとダンプボリュームのサイズは、次のようなプロファイル構文を使用して調整することができます。

```
install_type initial_install
cluster SUNWCXall
pool rpool 16g 2g 2g c0t0d0s0
```

このプロファイルでは、2 つの 2g エントリによって、スワップボリュームとダンプボリュームのサイズがそれぞれ 2G バイトに設定されます。

- インストール済みのシステムのスワップ領域を増やす必要がある場合は、スワップボリュームを追加するだけです。次に例を示します。

```
# zfs create -V 2G rpool/swap2
```

その後、新しいスワップボリュームをアクティブにします。次に例を示します。

```
# swap -a /dev/zvol/dsk/rpool/swap2
# swap -l
swapfile                  dev  swaplo   blocks   free
/dev/zvol/dsk/rpool/swap  256,1      16 1058800 1058800
/dev/zvol/dsk/rpool/swap2 256,3      16 4194288 4194288
```

最後に、2 つ目のスワップボリュームのエントリを /etc/vfstab ファイルに追加します。

ZFS ダンプデバイスの問題のトラブルシューティング

システムクラッシュダンプの取得やダンプデバイスのサイズ変更で問題が発生した場合には、次の項目を確認してください。

- クラッシュダンプが自動的に作成されなかった場合は、savecore コマンドを使ってクラッシュダンプを保存することができます。
- ZFS ルートファイルシステムの初期インストール時や ZFS ルートファイルシステムへの移行時に、ダンプボリュームが自動的に作成されます。ダンプボリュームのデフォルトサイズが小さすぎる場合には、ほとんどの場合、ダンプボリュームのサイズを調整するだけですみます。たとえば、大量のメモリーが搭載されたシステムでは、次のようにダンプボリュームのサイズを 40G バイトに増やします。

```
# zfs set volsize=40G rpool/dump
```

大きなサイズのダンプボリュームのサイズ変更処理には、長い時間がかかる可能性があります。

何らかの理由で、ダンプデバイスを手動で作成したあとでそのデバイスを有効化する必要がある場合には、次のような構文を使用します。

```
# dumpadm -d /dev/zvol/dsk/rpool/dump
      Dump content: kernel pages
       Dump device: /dev/zvol/dsk/rpool/dump (dedicated)
Savecore directory: /var/crash/t2000
  Savecore enabled: yes
```

- 128G バイト以上のメモリーが搭載されたシステムでは、デフォルトで作成されるダンプデバイスよりも大きいダンプデバイスが必要となります。ダンプデバイスが小さすぎて既存のクラッシュダンプを取得できない場合には、次のようなメッセージが表示されます。

  ```
  # dumpadm -d /dev/zvol/dsk/rpool/dump
  dumpadm: dump device /dev/zvol/dsk/rpool/dump is too small to hold a system dump
  dump size 36255432704 bytes, device size 34359738368 bytes
  ```

  スワップデバイスやダンプデバイスのサイジングについては、『Solaris のシステム管理 (デバイスとファイルシステム)』の「スワップ空間の計画」を参照してください。

- 現在のところ、複数の最上位デバイスを含むプールにダンプデバイスを追加することはできません。次のようなメッセージが表示されます。

```
# dumpadm -d /dev/zvol/dsk/datapool/dump
dump is not supported on device '/dev/zvol/dsk/datapool/dump': 'datapool' has multiple top level vdevs
```

ダンプデバイスは、最上位デバイスを複数持つことのできないルートプールに追加してください。

ZFS ルートファイルシステムからの起動

SPARC システムと x86 システムの両方で、ブートアーカイブによる新しい形式の起動方法が使用されます。ブートアーカイブは、起動に必要なファイルを含んだファイルシステムイメージです。ZFS ルートファイルシステムからシステムが起動される際には、ブートアーカイブとカーネルファイルのパス名が、起動用に選択されたルートファイルシステム内で解決されます。

インストールのためにシステムを起動する場合は、インストール処理の全体にわたって RAM ディスクがルートファイルシステムとして使用されます。

ZFS では、単一のルートファイルシステムではなくストレージプールが起動デバイス指定子で指定されるため、ZFS ファイルシステムからの起動は UFS ファイルシステムからの起動とは異なります。ストレージプールには、複数の「ブート可能なデータセット」または ZFS ルートファイルシステムが含まれている場合があります。ZFS から起動する場合は、起動デバイスと、起動デバイスによって指定されたプール内のルートファイルシステムを指定する必要があります。

デフォルトでは、プールの bootfs プロパティーで指定されているデータセットが、起動用に選択されます。別のブート可能データセットを boot -Z コマンドに指定することで、このデフォルトの選択を無効にできます。

ミラー化された ZFS ルートプールの代替ディスクから起動する

ミラー化された ZFS ルートプールは、システムのインストール時に作成するか、インストール後にディスクを接続することによって作成することができます。詳細については、次のトピックを参照してください。

- 130 ページの「ZFS ルートファイルシステムのインストール (初期インストール)」
- 136 ページの「ミラー化ルートプールを作成する方法 (インストール後)」

ミラー化された ZFS ルートプールに関して、次に示す既知の問題を確認してください。

- CR 6668666 – ミラー内のほかのディスクで起動できるようにするには、追加で接続したディスクにブート情報を installboot コマンドまたは installgrub コマンドでインストールする必要があります。ミラー化された ZFS ルートプールを初期インストールで作成した場合は、この手順は不要です。たとえば、ミラーに追加した 2 番目のディスクが c0t1d0s0 であった場合、installboot コマンドまたは installgrub コマンドの構文は次のようになります。
  - SPARC:

```
sparc# installboot -F zfs /usr/platform/‘uname -i‘/lib/fs/zfs/bootblk /dev/rdsk/c0t1d0s0
```

  - x86:

```
x86# installgrub /boot/grub/stage1 /boot/grub/stage2 /dev/rdsk/c0t1d0s0
```

- ミラー化された ZFS ルートプールのさまざまなデバイスから起動することができます。ハードウェア構成によっては、別の起動デバイスを指定するには、PROM または BIOS の更新が必要になる場合があります。

  たとえば、次のプール内のどちらかのディスク (c1t0d0s0 または c1t1d0s0) から起動できます。

```
# zpool status
  pool: rpool
 state: ONLINE
 scrub: none requested
config:

        NAME          STATE     READ WRITE CKSUM
        rpool         ONLINE       0     0     0
          mirror-0    ONLINE       0     0     0
            c1t0d0s0  ONLINE       0     0     0
            c1t1d0s0  ONLINE       0     0     0
```

- SPARC: ok プロンプトで代替ディスクを入力します。

  ```
  ok boot /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/disk@0
  ```

  システムが再起動したら、アクティブな起動デバイスを確認します。次に例を示します。

  ```
  SPARC# prtconf -vp | grep bootpath
         bootpath:  '/pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/disk@0,0:a'
  ```

- x86: ミラー化された ZFS ルートプールの代替ディスクを、適切な BIOS メニューで選択します。

  続いて、次のような構文を使って、代替ディスクから起動されていることを確認します。

  ```
  x86# prtconf -v|sed -n '/bootpath/,/value/p'
         name='bootpath' type=string items=1
             value='/pci@0,0/pci8086,25f8@4/pci108e,286@0/disk@0,0:a'
  ```

SPARC: ZFS ルートファイルシステムから起動する

複数の ZFS BE が存在する SPARC システムでは、`luactivate` コマンドを使用することによって、任意の BE から起動できます。

Solaris OS インストールおよび Oracle Solaris Live Upgrade の処理中に、ZFS ルートファイルシステムが `bootfs` プロパティーで自動的に指定されます。

ブート可能なデータセットがプール内に複数存在する場合があります。デフォルトでは、*/pool-name*`/boot/menu.lst` ファイルのブート可能データセットのエントリは、プールの `bootfs` プロパティーで指定されます。ただし、`menu.lst` のエントリに `bootfs` コマンドを含めて、プールの代替データセットを指定することもできます。このように、`menu.lst` ファイルには、プール内の複数のルートファイルシステムに対応するエントリが含まれている場合があります。

システムを ZFS ルートファイルシステムでインストールするか、ZFS ルートファイルシステムに移行すると、次のようなエントリが `menu.lst` ファイルに追加されます。

```
title zfsBE
bootfs rpool/ROOT/zfsBE
title zfs2BE
bootfs rpool/ROOT/zfs2BE
```

新しい BE を作成すると、`menu.lst` ファイルが自動的に更新されます。

SPARC システムでは、2 つの新しい起動オプションを使用できます。

- その BE がアクティブになったあと、boot -L コマンドを使用して ZFS プール内のブート可能なデータセットのリストを表示できます。その後、ブート可能なデータセットの 1 つをリストで選択できます。そのデータセットを起動するための詳細な手順が表示されます。手順に従って、選択したデータセットを起動できます。
- boot -Z *dataset* コマンドを使用して、特定の ZFS データセットを起動できます。

例 5–8 SPARC: 特定の ZFS ブート環境から起動する

システムの起動デバイス上の ZFS ストレージプールに複数の ZFS BE が存在する場合は、luactivate コマンドを使用してデフォルトの BE を指定できます。

たとえば、次の ZFS BE が lustatus の出力のとおり使用可能であるとします。

```
# lustatus
Boot Environment           Is       Active Active    Can    Copy
Name                       Complete Now    On Reboot Delete Status
-------------------------- -------- ------ --------- ------ ----------
zfsBE                      yes      no     no        yes    -
zfs2BE                     yes      yes    yes       no     -
```

SPARC システム上に ZFS BE が複数存在している場合、boot -L コマンドを使用すれば、デフォルト BE とは異なる BE から起動することができます。ただし、boot -L セッションから起動された BE がデフォルト BE としてリセットされることはなく、bootfs プロパティーも更新されません。boot -L セッションから起動された BE をデフォルト BE にするには、その BE を luactivate コマンドでアクティブにする必要があります。

次に例を示します。

```
ok boot -L
Rebooting with command: boot -L
Boot device: /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/disk@0  File and args: -L

1 zfsBE
2 zfs2BE
Select environment to boot: [ 1 - 2 ]: 1
To boot the selected entry, invoke:
boot [<root-device>] -Z rpool/ROOT/zfsBE

Program terminated
ok boot -Z rpool/ROOT/zfsBE
```

例 5–9 SPARC: ZFS ファイルシステムをフェイルセーフモードで起動する

SPARC システムでは、/platform/‘uname -i‘/failsafe にあるフェイルセーフアーカイブから、次のように起動できます。

```
ok boot -F failsafe
```

例 5–9 SPARC: ZFS ファイルシステムをフェイルセーフモードで起動する (続き)

特定の ZFS ブート可能データセットのフェイルセーフアーカイブを起動するには、次のような構文を使用します。

```
ok boot -Z rpool/ROOT/zfsBE -F failsafe
```

x86: ZFS ルートファイルシステムから起動する

Solaris OS インストール処理中または Oracle Solaris Live Upgrade 操作中に、ZFS を自動的に起動するための次のようなエントリが */pool-name*/boot/grub/menu.lst ファイルに追加されます。

```
title Solaris 10 9/10  X86
findroot (rootfs0,0,a)
kernel$ /platform/i86pc/multiboot -B $ZFS-BOOTFS
module /platform/i86pc/boot_archive
title Solaris failsafe
findroot (rootfs0,0,a)
kernel /boot/multiboot kernel/unix -s -B console=ttya
module /boot/x86.miniroot-safe
```

GRUB によって起動デバイスとして識別されたデバイスに ZFS ストレージプールが含まれている場合、menu.lst ファイルを使用して GRUB メニューが作成されます。

複数の ZFS BE が存在する x86 システムでは、BE を GRUB メニューから選択できます。このメニューエントリに対応するルートファイルシステムが ZFS データセットである場合は、次のオプションが追加されます。

```
-B $ZFS-BOOTFS
```

例 5–10 x86: ZFS ファイルシステムを起動する

ZFS ファイルシステムからシステムを起動する場合は、boot -B $ZFS-BOOTFS パラメータを GRUB メニューエントリの kernel 行または module 行に記述して、ルートデバイスを指定します。このパラメータ値は、-B オプションで指定されるすべてのパラメータと同様に、GRUB によってカーネルに渡されます。次に例を示します。

```
title Solaris 10 9/10  X86
findroot (rootfs0,0,a)
kernel$ /platform/i86pc/multiboot -B $ZFS-BOOTFS
module /platform/i86pc/boot_archive
title Solaris failsafe
findroot (rootfs0,0,a)
kernel /boot/multiboot kernel/unix -s -B console=ttya
module /boot/x86.miniroot-safe
```

例 5–11　x86: ZFS ファイルシステムをフェイルセーフモードで起動する

x86 のフェイルセーフアーカイブは /boot/x86.miniroot-safe です。GRUB メニューで Solaris フェイルセーフエントリを選択することによって起動できます。次に例を示します。

```
title Solaris failsafe
findroot (rootfs0,0,a)
kernel /boot/multiboot kernel/unix -s -B console=ttya
module /boot/x86.miniroot-safe
```

正常な起動を妨げる ZFS マウントポイントの問題の解決 (Solaris 10 10/08)

アクティブなブート環境を変更するための最適な方法は、luactivate コマンドを使用することです。不適切なパッチや構成エラーが原因でアクティブな環境の起動に失敗する場合、別の環境から起動する唯一の方法は、起動時にその環境を選択することです。x86 システムでは GRUB メニューから代替 BE を選択でき、SPARC システムでは PROM から明示的に代替 BE を起動できます。

Solaris 10 10/08 リリースの Oracle Solaris Live Upgrade のバグのため、アクティブでないブート環境は起動に失敗する場合があります。これは、ブート環境の ZFS データセットまたはゾーンの ZFS データセットに無効なマウントポイントが含まれているためです。同じバグのため、BE に別の /var データセットがある場合は、BE をマウントすることもできなくなります。

ゾーンのデータセットに無効なマウントポイントが含まれている場合は、次の手順を実行してマウントポイントを修正することができます。

▼ ZFS マウントポイントの問題を解決する方法

1. フェイルセーフアーカイブからシステムを起動します。
2. プールをインポートします。
   次に例を示します。

   ```
   # zpool import rpool
   ```

3. 正しくない一時的なマウントポイントを探します。
   次に例を示します。

   ```
   # zfs list -r -o name,mountpoint rpool/ROOT/s10u6

       NAME                              MOUNTPOINT
       rpool/ROOT/s10u6                  /.alt.tmp.b-VP.mnt/
       rpool/ROOT/s10u6/zones            /.alt.tmp.b-VP.mnt//zones
   ```

```
rpool/ROOT/s10u6/zones/zonerootA   /.alt.tmp.b-VP.mnt/zones/zonerootA
```

ルート BE (rpool/ROOT/s10u6) のマウントポイントは / となるべきです。

/var のマウントの問題が原因で起動に失敗する場合は、/var データセットについて同様に、正しくない一時的なマウントポイントを探します。

4 **ZFS BE とそのデータセットのマウントポイントを設定しなおします。**

次に例を示します。

```
# zfs inherit -r mountpoint rpool/ROOT/s10u6
# zfs set mountpoint=/ rpool/ROOT/s10u6
```

5 システムを再起動します。

GRUB メニューまたは OpenBoot PROM プロンプトで、特定のブート環境を起動するオプションが表示されたら、前の手順でマウントポイントを修正したブート環境を選択します。

ZFS ルート環境での回復のための起動

失われたルートパスワードやそれに似た問題から回復する目的でシステムを起動する必要がある場合は、次の手順を使用します。

エラーの深刻度に応じてフェイルセーフモードの起動、代替メディアからの起動のいずれかを行う必要があります。一般に、フェイルセーフモードを起動すれば、失われたルートパスワードや未知のルートパスワードを回復することができます。

- 176 ページの「ZFS フェイルセーフモードを起動する方法」
- 177 ページの「代替メディアから ZFS を起動する方法」

ルートプールまたはルートプールのスナップショットを回復する必要がある場合は、178 ページの「ZFS ルートプールまたはルートプールのスナップショットを回復する」を参照してください。

▼ ZFS フェイルセーフモードを起動する方法

1 フェイルセーフモードを起動します。

SPARC システムの場合:

```
ok boot -F failsafe
```

x86 システムの場合、GRUB プロンプトからフェイルセーフモードを選択します。

2 プロンプトが表示されたら、**ZFS BE** を /a にマウントします。

.
.

```
.
ROOT/zfsBE was found on rpool.
Do you wish to have it mounted read-write on /a? [y,n,?] y
mounting rpool on /a
Starting shell.
```

3 /a/etc ディレクトリに移動します。

```
# cd /a/etc
```

4 必要であれば、**TERM** タイプを設定します。

```
# TERM=vt100
# export TERM
```

5 passwd または shadow ファイルを修正します。

```
# vi shadow
```

6 システムを再起動します。

```
# init 6
```

▼ 代替メディアから ZFS を起動する方法

システムの正常な起動を妨げる問題やその他の何らかの深刻な問題が発生した場合には、ネットワークインストールサーバーまたは Solaris インストール CD から起動し、ルートプールをインポートし、ZFS BE をマウントし、問題の解決を試みる必要があります。

1 インストール **CD** またはネットワークから起動します。

- SPARC:

  ```
  ok boot cdrom -s
  ok boot net -s
  ```

  -s オプションを使用しない場合は、インストールプログラムを終了する必要があります。

- x86: ネットワーク起動、ローカル CD からの起動、のいずれかのオプションを選択します。

2 ルートプールをインポートし、代替マウントポイントを指定します。次に例を示します。

```
# zpool import -R /a rpool
```

3 **ZFS BE** をマウントします。次に例を示します。

```
# zfs mount rpool/ROOT/zfsBE
```

4 /a ディレクトリから **ZFS BE** の内容にアクセスします。

```
# cd /a
```

5 システムを再起動します。

```
# init 6
```

ZFS ルートプールまたはルートプールのスナップショットを回復する

ここでは、次のタスクを実行する方法について説明します。

- 178 ページの「ZFS ルートプールのディスクを置き換える方法」
- 180 ページの「ルートプールのスナップショットを作成する方法」
- 182 ページの「ZFS ルートプールを再作成しルートプールのスナップショットを復元する方法」
- 183 ページの「フェイルセーフブートからルートプールのスナップショットをロールバックする方法」

▼ ZFS ルートプールのディスクを置き換える方法

次の理由により、ルートプールのディスクの置き換えが必要になることがあります。

- ルートプールが小さすぎるため、小さいディスクを大きいディスクに置き換えたい。
- ルートプールのディスクに障害が発生している。非冗長プールでディスクに障害が発生してシステムが起動しない場合は、CD やネットワークなどの代替メディアから起動したあとでルートプールのディスクを置き換える必要があります。

ミラー化ルートプール構成では、代替メディアからの起動を行わずにディスク交換を試みることができます。`zpool replace` コマンドを使用すれば、障害が発生したデバイスを置き換えることができます。あるいは追加ディスクがある場合には、`zpool attach` コマンドを使用できます。追加ディスクの接続やルートプールディスクの切り離しの例については、この節に含まれる手順を参照してください。

一部のハードウェアでは、故障したディスクを交換するための `zpool replace` 操作を試みる前に、ディスクをオフラインにして構成解除する必要があります。次に例を示します。

```
# zpool offline rpool c1t0d0s0
# cfgadm -c unconfigure c1::dsk/c1t0d0
```

```
<Physically remove failed disk c1t0d0>
<Physically insert replacement disk c1t0d0>
# cfgadm -c configure c1::dsk/c1t0d0
# zpool replace rpool c1t0d0s0
# zpool online rpool c1t0d0s0
# zpool status rpool
<Let disk resilver before installing the boot blocks>
SPARC# installboot -F zfs /usr/platform/'uname -i'/lib/fs/zfs/bootblk /dev/rdsk/c1t0d0s0
x86# installgrub /boot/grub/stage1 /boot/grub/stage2 /dev/rdsk/c1t9d0s0
```

一部のハードウェアでは、交換用ディスクの装着後にそのディスクをオンラインにしたり再構成を行ったりする必要がありません。

交換用ディスクからの起動をテストできるように、また、交換用ディスクに障害が発生した場合に既存のディスクから手動で起動できるように、現在のディスクと新しいディスクの起動デバイスのパス名を特定する必要があります。次の手順の例では、現在のルートプールディスク (c1t10d0s0) のパス名は次のとおりです。

```
/pci@8,700000/pci@3/scsi@5/sd@a,0
```

交換用起動ディスク (c1t9d0s0) のパス名は次のとおりです。

```
/pci@8,700000/pci@3/scsi@5/sd@9,0
```

1. 交換用ディスク (新しいディスク) を物理的に接続します。
2. 新しいディスクに SMI ラベルが付けられていてスライス 0 があることを確認してください。

   ルートプールに使用するディスクのラベルを変更する方法については、次のサイトを参照してください。

   http://www.solarisinternals.com/wiki/index.php/ZFS_Troubleshooting_Guide
3. 新しいディスクをルートプールに接続します。

   次に例を示します。

   ```
   # zpool attach rpool c1t10d0s0 c1t9d0s0
   ```
4. ルートプールのステータスを確認します。

   次に例を示します。

   ```
   # zpool status rpool
     pool: rpool
    state: ONLINE
   status: One or more devices is currently being resilvered.  The pool will
           continue to function, possibly in a degraded state.
   action: Wait for the resilver to complete.
    scrub: resilver in progress, 25.47% done, 0h4m to go
   config:

           NAME          STATE     READ WRITE CKSUM
   ```

```
        rpool          ONLINE       0     0     0
          mirror-0     ONLINE       0     0     0
            c1t10d0s0  ONLINE       0     0     0
            c1t9d0s0   ONLINE       0     0     0

errors: No known data errors
```

5 ディスクの再同期化が完了したら、新しいディスクにブートブロックを適用します。

次のような構文を使用します。

- SPARC:

  ```
  # installboot -F zfs /usr/platform/‘uname -i‘/lib/fs/zfs/bootblk /dev/rdsk/c1t9d0s0
  ```

- x86:

  ```
  # installgrub /boot/grub/stage1 /boot/grub/stage2 /dev/rdsk/c1t9d0s0
  ```

6 新しいディスクから起動できることを確認します。

たとえば、SPARC システムの場合、次のような構文を使用します。

```
ok boot /pci@8,700000/pci@3/scsi@5/sd@9,0
```

7 新しいディスクからシステムが起動した場合は、古いディスクを切り離します。

次に例を示します。

```
# zpool detach rpool c1t10d0s0
```

8 システムが新しいディスクから自動的に起動するように設定します。そのためには、eeprom コマンドまたは **SPARC** ブート **PROM** の setenv コマンドを使用するか、**PC BIOS** を再設定します。

▼ ルートプールのスナップショットを作成する方法

回復に利用できるようにルートプールのスナップショットを作成することができます。ルートプールのスナップショットを作成するための最適な方法は、ルートプールの再帰的なスナップショットを実行することです。

次の手順に従って、再帰的なルートプールスナップショットを作成し、そのスナップショットをリモートシステムでプール内のファイルとして保存します。ルートプールで障害が発生した場合には、NFS を使ってリモートデータセットをマウントし、そのスナップショットファイルを受信して作成し直したプール内に格納することができます。代わりに、ルートプールスナップショットをリモートシステムのプール内の実際のスナップショットとして保存することもできます。修復対象のシステムを Solaris OS ミニルートから起動する一方で ssh を構成または rsh を使用する必要があるので、リモートシステムとのスナップショットの送受信は、やや複雑です。

ルートプールスナップショットをリモートで保存して回復する方法、およびルートプールの回復に関する最新情報については、次のサイトを参照してください。

http://www.solarisinternals.com/wiki/index.php/ZFS_Troubleshooting_Guide

ファイルまたはスナップショットとしてリモートに格納されたスナップショットの内容を確認することは、ルートプールを回復する際の重要なステップです。プールの構成が変更された場合や Solaris OS をアップグレードした場合など、定期的にいずれかの方法を使ってスナップショットを作成し直すべきです。

次の手順では、zfsBE ブート環境からシステムを起動します。

1 リモートシステム上で、スナップショットを格納するためのプールとファイルシステムを作成します。

次に例を示します。

```
remote# zfs create rpool/snaps
```

2 ファイルシステムをローカルシステムと共有します。

次に例を示します。

```
remote# zfs set sharenfs='rw=local-system,root=local-system' rpool/snaps
# share
-@rpool/snaps   /rpool/snaps   sec=sys,rw=local-system,root=local-system   ""
```

3 ルートプールの再帰的なスナップショットを作成します。

```
local# zfs snapshot -r rpool@0804
local# zfs list
NAME                       USED  AVAIL  REFER  MOUNTPOINT
rpool                     6.17G  60.8G    98K  /rpool
rpool@0804                    0      -    98K  -
rpool/ROOT                4.67G  60.8G    21K  /rpool/ROOT
rpool/ROOT@0804               0      -    21K  -
rpool/ROOT/zfsBE          4.67G  60.8G  4.67G  /
rpool/ROOT/zfsBE@0804      386K      -  4.67G  -
rpool/dump                1.00G  60.8G  1.00G  -
rpool/dump@0804               0      -  1.00G  -
rpool/swap                 517M  61.3G    16K  -
rpool/swap@0804               0      -    16K  -
```

4 ルートプールのスナップショットをリモートシステムに送信します。

次に例を示します。

```
local# zfs send -Rv rpool@0804 > /net/remote-system/rpool/snaps/rpool.0804
sending from @ to rpool@0804
sending from @ to rpool/swap@0804
sending from @ to rpool/ROOT@0804
sending from @ to rpool/ROOT/zfsBE@0804
sending from @ to rpool/dump@0804
```

▼ ZFS ルートプールを再作成しルートプールのスナップショットを復元する方法

この手順では、次の条件を前提としています。

- ZFS ルートプールを回復できない。
- ZFS ルートプールのスナップショットがリモートシステム上に保存されており、NFS で共有されている。

手順はすべてローカルシステム上で実行します。

1. **CD/DVD またはネットワークから起動します。**
   - SPARC: 次のいずれかのブート方法を選択します。

     ```
     ok boot net -s
     ok boot cdrom -s
     ```

     -s オプションを使用しない場合は、インストールプログラムを終了する必要があります。
   - x86: DVD またはネットワークから起動するオプションを選択します。その後、インストールプログラムを終了します。

2. **リモートのスナップショットのデータセットをマウントします。**
   次に例を示します。

   ```
   # mount -F nfs remote-system:/rpool/snaps /mnt
   ```

   ネットワークサービスを構成していない場合は、*remote-system* の IP アドレスを指定する必要があります。

3. **ルートプールのディスクが置き換えられ、ZFS で使用可能なディスクラベルを含んでいない場合は、ディスクのラベルを変更する必要があります。**
   ディスクのラベル変更の詳細は、次の Web サイトを参照してください。

   http://www.solarisinternals.com/wiki/index.php/ZFS_Troubleshooting_Guide

4. **ルートプールを再作成します。**
   次に例を示します。

   ```
   # zpool create -f -o failmode=continue -R /a -m legacy -o cachefile=
   /etc/zfs/zpool.cache rpool c1t1d0s0
   ```

5. **ルートプールのスナップショットを復元します。**
   この手順には時間がかかることがあります。次に例を示します。

   ```
   # cat /mnt/rpool.0804 | zfs receive -Fdu rpool
   ```

-u オプションを使用すると、復元されたアーカイブは zfs receive 処理の完了時にマウントされません。

6 ルートプールのデータセットが復元されていることを確認します。

次に例を示します。

```
# zfs list
NAME                       USED  AVAIL  REFER  MOUNTPOINT
rpool                     6.17G  60.8G    98K  /a/rpool
rpool@0804                    0      -    98K  -
rpool/ROOT                4.67G  60.8G    21K  /legacy
rpool/ROOT@0804               0      -    21K  -
rpool/ROOT/zfsBE          4.67G  60.8G  4.67G  /a
rpool/ROOT/zfsBE@0804      398K      -  4.67G  -
rpool/dump                1.00G  60.8G  1.00G  -
rpool/dump@0804               0      -  1.00G  -
rpool/swap                 517M  61.3G    16K  -
rpool/swap@0804               0      -    16K  -
```

7 ルートプールの **BE** に bootfs プロパティーを設定します。

次に例を示します。

```
# zpool set bootfs=rpool/ROOT/zfsBE rpool
```

8 新しいディスクに起動ブロックをインストールします。

SPARC:

```
# installboot -F zfs /usr/platform/‘uname -i‘/lib/fs/zfs/bootblk /dev/rdsk/c1t1d0s0
```

x86:

```
# installgrub /boot/grub/stage1 /boot/grub/stage2 /dev/rdsk/c1t1d0s0
```

9 システムを再起動します。

```
# init 6
```

▼ フェイルセーフブートからルートプールのスナップショットをロールバックする方法

この手順では、ルートプールの既存のスナップショットを利用できることを前提としています。この例は、それらはローカルシステム上で使用可能となっています。

```
# zfs snapshot -r rpool@0804
# zfs list
NAME                       USED  AVAIL  REFER  MOUNTPOINT
rpool                     6.17G  60.8G    98K  /rpool
rpool@0804                    0      -    98K  -
rpool/ROOT                4.67G  60.8G    21K  /rpool/ROOT
rpool/ROOT@0804               0      -    21K  -
```

```
rpool/ROOT/zfsBE           4.67G  60.8G  4.67G  /
rpool/ROOT/zfsBE@0804       398K      -  4.67G  -
rpool/dump                 1.00G  60.8G  1.00G  -
rpool/dump@0804               0      -  1.00G  -
rpool/swap                  517M  61.3G    16K  -
rpool/swap@0804               0      -    16K  -
```

1 システムをシャットダウンし、フェイルセーフモードで起動します。

```
ok boot -F failsafe
ROOT/zfsBE was found on rpool.
Do you wish to have it mounted read-write on /a? [y,n,?] y
mounting rpool on /a

Starting shell.
```

2 ルートプールの各スナップショットをロールバックします。

```
# zfs rollback rpool@0804
# zfs rollback rpool/ROOT@0804
# zfs rollback rpool/ROOT/zfsBE@0804
```

3 再起動してマルチユーザーモードにします。

```
# init 6
```

◆ ◆ ◆ 第 6 章

Oracle Solaris ZFS ファイルシステムの管理

この章では、Oracle Solaris ZFS ファイルシステムの管理について詳しく説明します。ファイルシステムの階層レイアウト、プロパティーが継承されること、およびマウントポイント管理および共有が自動的に行われることなどについて、それらの概念を説明しています。

この章は、次の節で構成されます。

ZFS ファイルシステムの管理 (概要)

ZFS ファイルシステムは、ストレージプールの最上位に構築されます。ファイルシステムは動的に作成および破棄することができ、基礎となるディスク領域を割り当てたりフォーマットしたりする必要はありません。ファイルシステムが非常に軽量であることと、ZFS はファイルシステムに基づいて管理することから、作成されるファイルシステムの数は多くなる傾向があります。

ZFS ファイルシステムの管理には、`zfs` コマンドを使用します。`zfs` コマンドには、ファイルシステムに特定の操作を実行するために一連のサブコマンドが用意されています。この章では、これらのサブコマンドについて詳細に説明します。スナップショット、ボリューム、およびクローンもこのコマンドを使って管理しますが、これらの機能についてはこの章では簡単に取り上げるだけにとどめます。スナップショットおよびクローンの詳細については、第 7 章「Oracle Solaris ZFS のスナップショットとクローンの操作」を参照してください。ZFS ボリュームの詳細については、285 ページの「ZFS ボリューム」を参照してください。

注– 「データセット」という用語は、この章ではファイルシステム、スナップショット、クローン、またはボリュームの総称として使用します。

ZFS ファイルシステムの作成、破棄、および名前変更を行う

ZFS ファイルシステムは、`zfs create` および `zfs destroy` コマンドを使って作成および破棄できます。`zfs rename` コマンドを使用して、ZFS ファイルシステムの名前を変更できます。

ZFS ファイルシステムを作成する

ZFS ファイルシステムの作成には、`zfs create` コマンドを使用します。`create` サブコマンドの引数は 1 つだけです。作成するファイルシステムの名前です。ファイルシステム名は、次のようにプール名から始まるパス名として指定します。

pool-name/[filesystem-name/]filesystem-name

パスのプール名と最初のファイルシステム名は、新規ファイルシステムを階層内のどこに作成するかを示しています。パスの最後にある名前は、作成するファイルシステムの名前です。ファイルシステム名は、51 ページの「ZFS コンポーネントに名前を付けるときの規則」の命名要件に従って付ける必要があります。

次の例では、`bonwick` という名前のファイルシステムが `tank/home` ファイルシステムに作成されます。

```
# zfs create tank/home/bonwick
```

新しく作成するファイルシステムが正常に作成されると、自動的にマウントされます。ファイルシステムは、デフォルトでは `create` サブコマンドのファイルシステム名に指定したパスを使って、*/dataset* としてマウントされます。この例では、新しく作成した `bonwick` ファイルシステムは `/tank/home/bonwick` にマウントされます。自動的に管理されるマウントポイントの詳細については、212 ページの「ZFS マウントポイントを管理する」を参照してください。

`zfs create` コマンドの詳細については、`zfs(1M)` を参照してください。

ファイルシステムの作成時にファイルシステムのプロパティーを設定できます。

次の例では、tank/home ファイルシステム用に /export/zfs というマウントポイントが作成されます。

```
# zfs create -o mountpoint=/export/zfs tank/home
```

ファイルシステムのプロパティーの詳細については、189 ページの「ZFS のプロパティーの紹介」を参照してください。

ZFS ファイルシステムを破棄する

ZFS ファイルシステムを破棄するには、zfs destroy コマンドを使用します。破棄されたファイルシステムは、マウントおよび共有が自動的に解除されます。自動的に管理されるマウントおよび共有の詳細については、212 ページの「自動マウントポイント」を参照してください。

次の例では、tabriz ファイルシステムが破棄されます。

```
# zfs destroy tank/home/tabriz
```

注意 – destroy サブコマンドでは、確認を求めるプロンプトは表示されません。慎重に使用してください。

破棄するファイルシステムがビジー状態でマウントを解除できない場合、zfs destroy コマンドは失敗します。アクティブなファイルシステムを破棄する場合は、-f オプションを使用します。このオプションは慎重に使用してください。アクティブなファイルシステムのマウント解除、共有解除、および破棄も実行することができ、その場合はアプリケーションが予期しない動作をすることがあります。

```
# zfs destroy tank/home/ahrens
cannot unmount 'tank/home/ahrens': Device busy

# zfs destroy -f tank/home/ahrens
```

zfs destroy コマンドは、ファイルシステムに子孫が存在する場合にも失敗します。ファイルシステムとそのすべての子孫を再帰的に破棄するときは、-r オプションを使用します。再帰的な破棄を実行すると、スナップショットも破棄されるので、このオプションは慎重に使用してください。

```
# zfs destroy tank/ws
cannot destroy 'tank/ws': filesystem has children
use '-r' to destroy the following datasets:
tank/ws/billm
tank/ws/bonwick
tank/ws/maybee

# zfs destroy -r tank/ws
```

破棄するファイルシステムに間接的な依存関係が存在する場合は、再帰的な破棄コマンドでも失敗します。破棄する階層の外部に複製されたファイルシステムなど、すべての依存関係を強制的に破棄する場合は、-R オプションを使用する必要があります。このオプションは、慎重に使用してください。

```
# zfs destroy -r tank/home/schrock
cannot destroy 'tank/home/schrock': filesystem has dependent clones
use '-R' to destroy the following datasets:
tank/clones/schrock-clone

# zfs destroy -R tank/home/schrock
```

注意 – zfs destroy コマンドの -f、-r、または -R オプションでは、確認を求めるプロンプトは表示されないので、これらのオプションは慎重に使用してください。

スナップショットおよびクローンの詳細については、第 7 章「Oracle Solaris ZFS のスナップショットとクローンの操作」を参照してください。

ZFS ファイルシステムの名前を変更する

zfs rename コマンドを使用して、ファイルシステムの名前を変更できます。rename サブコマンドを使用すれば、次の操作を実行できます。

- ファイルシステムの名前を変更する。
- ZFS 階層内でファイルシステムの場所を移動する。
- ファイルシステムの名前を変更して、ZFS 階層内で場所を移動する。

次の例では、rename サブコマンドを使ってファイルシステムの名前を kustarz から kustarz_old に変更しています。

```
# zfs rename tank/home/kustarz tank/home/kustarz_old
```

次の例では、zfs rename を使用してファイルシステムの場所を移動する方法を示しています。

```
# zfs rename tank/home/maybee tank/ws/maybee
```

この例では、maybee ファイルシステムの場所が tank/home から tank/ws に移動します。名前の変更を使ってファイルの場所を移動するときは、新しい場所は同じプールの中にする必要があり、新しいファイルシステムを格納するために十分なディスク領域が存在している必要があります。割り当て制限に達したなどの理由で新しい場所のディスク容量が不足していると、rename 操作は失敗します。

割り当て制限の詳細については、218 ページの「ZFS の割り当て制限と予約を設定する」を参照してください。

rename を実行すると、ファイルシステムおよび子孫のファイルシステム (存在する場合) をマウント解除して再マウントしようとする処理が行われます。アクティブなファイルシステムのマウントを解除できない場合、rename コマンドは失敗します。この問題が発生した場合は、ファイルシステムを強制的にマウント解除する必要があります。

スナップショットの名前を変更する方法については、229 ページの「ZFS スナップショットの名前を変更する」を参照してください。

ZFS のプロパティーの紹介

ファイルシステム、ボリューム、スナップショット、およびクローンの動作を制御するときには、主にプロパティーという機構を使用します。この節で定義しているプロパティーは、特に説明している場合を除いて、すべての種類のデータセットに適用されます。

- 198 ページの「ZFS の読み取り専用のネイティブプロパティー」
- 200 ページの「設定可能な ZFS ネイティブプロパティー」
- 203 ページの「ZFS ユーザープロパティー」

プロパティーは、ネイティブプロパティーとユーザー定義プロパティーの 2 種類に分けられます。ネイティブプロパティーは、内部の統計情報をエクスポートするか、ZFS ファイルシステムの動作を制御します。また、ネイティブプロパティーは設定可能なプロパティーまたは読み取り専用のプロパティーのどちらかです。ユーザープロパティーは ZFS ファイルシステムの動作には影響しませんが、これらを使用すると、使用環境内で意味を持つようにデータセットに注釈を付けることができます。ユーザープロパティーの詳細については、203 ページの「ZFS ユーザープロパティー」を参照してください。

設定可能なプロパティーのほとんどは、継承可能なプロパティーでもあります。継承可能なプロパティーとは、親データセットに設定されるとそのすべての子孫に伝達されるプロパティーのことです。

継承可能なプロパティーには必ず、どのようにしてプロパティーが取得されたかを示すソースが関連付けられています。プロパティーのソースには、次の値が記述される可能性があります。

`local`	そのプロパティーが `zfs set` コマンドを使用して明示的にデータセットに設定されたことを示しています。206 ページの「ZFS プロパティーを設定する」を参照してください。
`inherited from` *dataset-name*	そのプロパティーが、指定された祖先から継承されたことを示しています。
`default`	そのプロパティーの値が、継承されたのでもローカルで設定されたのでもないことを示しています。こ

のソースは、このプロパティーがソース `local` として設定された祖先が存在しないことを示しています。

次の表には、ZFS ファイルシステムの読み取り専用のネイティブプロパティーと設定可能なネイティブプロパティーの両方を示しています。読み取り専用のネイティブプロパティーには、そのことを記載しています。この表に示すそれ以外のプロパティーは、すべて設定可能なプロパティーです。ユーザープロパティーについては、203 ページの「ZFS ユーザープロパティー」を参照してください。

表 6–1 ZFS のネイティブプロパティーの説明

プロパティー名	種類	デフォルト値	説明
`aclinherit`	文字列	`secure`	ファイルとディレクトリが作成されるときに ACL エントリをどのように継承するかを制御します。値は、`discard`、`noallow`、`secure`、および `passthrough` です。これらの値については、249 ページの「ACL プロパティー」を参照してください。
`aclmode`	文字列	`groupmask`	`chmod` を実行するときに ACL エントリをどのように変更するかを制御します。値は、`discard`、`groupmask`、および `passthrough` です。これらの値については、249 ページの「ACL プロパティー」を参照してください。
`atime`	ブール型	`on`	ファイルを読み取るときにファイルのアクセス時刻を更新するかどうかを制御します。このプロパティーをオフに設定すると、ファイルを読み取るときに書き込みトラフィックが生成されなくなるため、パフォーマンスが大幅に向上する可能性があります。ただし、メールプログラムなどのユーティリティーが予期しない動作をすることがあります。
`available`	数値	なし	読み取り専用プロパティー。データセットおよびそのすべての子が利用できるディスク容量を調べます。プール内でほかのアクティビティーが実行されていないことを前提とします。ディスク容量はプール内で共有されるため、プールの物理サイズ、割り当て制限、予約、プール内のほかのデータセットなどのさまざまな要因によって、利用できる容量が制限されることがあります。 このプロパティーの省略名は `avail` です。 ディスク領域の計上の詳細については、62 ページの「ZFS のディスク領域の計上」を参照してください。

表 6–1 ZFS のネイティブプロパティーの説明 (続き)

プロパティー名	種類	デフォルト値	説明
canmount	ブール型	on	ファイルシステムが zfs mount コマンドを使ってマウントできるかどうかを制御します。このプロパティーはどのファイルシステムにも設定可能で、プロパティー自体は継承可能ではありません。ただし、このプロパティーを off に設定するとマウントポイントを子孫のファイルシステムに継承できますが、ファイルシステム自体がマウントされることはありません。 noauto オプションを設定すると、データセットのマウントおよびマウント解除は明示的に実行することが必要になります。データセットの作成時やインポート時に、データセットが自動的にマウントされることはありません。また、zfs mount-a コマンドでマウントされることや、zfs unmount-a コマンドでマウント解除されることもありません。 詳細は、201 ページの「canmount プロパティー」を参照してください。
チェックサム	文字列	on	データの整合性を検証するために使用するチェックサムを制御します。デフォルト値は on で、適切なアルゴリズム (現在は fletcher4) が自動的に選択されます。値は、on、off、fletcher2、fletcher4、および sha256 です。値を off にすると、ユーザーデータの完全性チェックが無効になります。値を off にすることは推奨されていません。
compression	文字列	off	データセットに対する圧縮を有効または無効にします。値は on、 off、lzjb、gzip、および gzip-*N* です。現時点では、このプロパティーを lzjb、gzip、または gzip-*N* に設定することは、このプロパティーを on に設定することと同じ効果を持ちます。既存のデータを持つファイルシステムで compression を有効にした場合は、新しいデータのみが圧縮されます。既存のデータは圧縮されないまま残されます。 このプロパティーの省略名は compress です。

表 6–1 ZFS のネイティブプロパティーの説明 (続き)

プロパティー名	種類	デフォルト値	説明
compressratio	数値	なし	読み取り専用プロパティー。データセットに適用された圧縮率を調べます。乗数で表現されます。zfs set compression=on *dataset* コマンドを実行すると、圧縮を有効にできます。 値は、すべてのファイルの論理サイズおよび参照する物理データの量から計算されます。これには、compression プロパティーを使用して明示的に圧縮されたデータセットも含まれます。
copies	数値	1	ファイルシステムごとのユーザーデータのコピー数を設定します。使用できる値は 1、2、または 3 です。これらのコピーは、プールレベルの冗長性を補うものです。ユーザーデータの複数のコピーで使用されるディスク領域は、対応するファイルとデータセットから取られるため、割り当て制限と予約にとって不利に働きます。また、複数のコピーを有効にすると、used プロパティーが更新されます。既存のファイルシステムでこのプロパティーを変更しても、新たに書き出されるデータが影響を受けるだけなので、ファイルシステムの作成時にこのプロパティーの設定を検討してください。
creation	文字列	なし	読み取り専用プロパティー。このデータセットが作成された日時を調べます。
devices	ブール型	on	ファイルシステムのデバイスファイルを開けるかどうかを制御します。
exec	ブール型	on	ファイルシステムに含まれるプログラムの実行を許可するかどうかを制御します。off に設定した場合は、PROT_EXEC による mmap(2) 呼び出しも許可されません。
mounted	ブール型	なし	読み取り専用のプロパティー。ファイルシステム、クローン、またはスナップショットが現在マウントされているかどうかを調べます。このプロパティーは、ボリュームには適用されません。値には yes または no を指定できます。

表 6–1 ZFS のネイティブプロパティーの説明 (続き)

プロパティー名	種類	デフォルト値	説明
mountpoint	文字列	なし	このファイルシステムで使用されるマウントポイントを制御します。ファイルシステムの mountpoint プロパティーを変更すると、そのマウントポイントを継承するファイルシステムおよびそのすべての子孫がマウント解除されます。新しい値が legacy の場合は、マウントが解除されたままになります。それ以外のときは、プロパティーの古い値が legacy または none だった場合、またはプロパティーが変更される前にマウントされていた場合は、自動的に再マウントされます。また、共有されていたすべてのファイルシステムは、共有が解除されてから新しい場所で共有されます。 このプロパティーの使用方法の詳細については、212 ページの「ZFS マウントポイントを管理する」を参照してください。
primarycache	文字列	all	一次キャッシュ (ARC) にキャッシュされる内容を制御します。設定できる値は、all、none、および metadata です。all に設定すると、ユーザーデータとメタデータの両方がキャッシュされます。none に設定すると、ユーザーデータも、メタデータも、キャッシュされません。metadata に設定すると、メタデータだけがキャッシュされます。
origin	文字列	なし	複製されたファイルシステムまたはボリュームのための読み取り専用プロパティー。どのスナップショットからクローンが作成されたかを調べます。クローンが存在する場合には、-r や -f オプションを使用しても、作成元は破棄できません。 複製されていないファイルシステムの origin は、none になります。

表 6–1 ZFS のネイティブプロパティーの説明 (続き)

プロパティー名	種類	デフォルト値	説明
quota	数値 (または none)	none	データセットおよびその子孫が消費できるディスク容量を制限します。このプロパティーは、使用されるディスク容量に強い制限値を適用します。容量には、子孫 (ファイルシステムやスナップショットを含む) が使用するすべての容量も含まれます。割り当て制限がすでに設定されているデータセットの子孫に割り当て制限を設定した場合は、祖先の割り当て制限は上書きされずに、制限が追加されます。ボリュームには割り当て制限を設定できません。volsize プロパティーが暗黙的な割り当て制限として機能します。 割り当て制限の設定については、219 ページの「ZFS ファイルシステムに割り当て制限を設定する」を参照してください。
readonly	ブール型	off	データセットを変更できるかどうかを制御します。on に設定すると、変更できなくなります。 このプロパティーの省略名は rdonly です。
recordsize	数値	128K	ファイルシステムに格納するファイルの推奨ブロックサイズを指定します。 このプロパティーの省略名は recsize です。詳細については、202 ページの「recordsize プロパティー」を参照してください。
referenced	数値	なし	読み取り専用プロパティー。データセットからアクセスできるデータの量を調べます。プール内のほかのデータセットで共有されるデータも含まれることがあります。 スナップショットまたはクローンを作成したときには、それらの作成元のファイルシステムまたはスナップショットと同じディスク領域を最初は参照しています。内容が同じであるためです。 このプロパティーの省略名は refer です。
refquota	数値 (または none)	none	1 つのデータセットが消費できるディスク容量を設定します。このプロパティーにより、使用される容量に対して強い制限値が設定されます。この強い制限値には、スナップショットやクローンなどの子孫で使用されるディスク容量は含まれません。

表 6–1 ZFS のネイティブプロパティーの説明 (続き)

プロパティー名	種類	デフォルト値	説明
refreservation	数値 (または none)	none	データセットに保証される最小ディスク容量を設定します。この容量には、スナップショットやクローンなどの子孫は含まれません。使用しているディスク容量がこの値を下回っているデータセットは、refreservation に指定された容量を使用していると見なされます。refreservation 予約は、親データセットが使用するディスク容量に計上されるので、親データセットの割り当て制限と予約を減らすことになります。 refreservation を設定すると、スナップショットを作成できるのは、データセットの *referenced* の現在のバイト数を格納できる十分な空きプール領域が、この予約容量のほかに存在する場合だけになります。 このプロパティーの省略名は refreserv です。
reservation	数値 (または none)	none	データセットおよびその子孫に保証される最小ディスク容量を設定します。使用しているディスク容量がこの値を下回っているデータセットは、予約に指定された容量を使用していると見なされます。予約は、親データセットが使用するディスク容量に計上されるので、親データセットの割り当て制限と予約を減らすことになります。 このプロパティーの省略名は reserv です。 詳細については、223 ページの「ZFS ファイルシステムに予約を設定する」を参照してください。
secondarycache	文字列	all	二次キャッシュ (L2ARC) にキャッシュされる内容を制御します。設定できる値は、all、none、および metadata です。all に設定すると、ユーザーデータとメタデータの両方がキャッシュされます。none に設定すると、ユーザーデータも、メタデータも、キャッシュされません。metadata に設定すると、メタデータだけがキャッシュされます。
setuid	ブール型	on	ファイルシステムで setuid ビットを考慮するかどうかを制御します。

表 6–1 ZFS のネイティブプロパティーの説明 (続き)

プロパティー名	種類	デフォルト値	説明
shareiscsi	文字列	off	ZFS ボリュームが iSCSI ターゲットとして共有されるかどうかを制御します。プロパティーの値は on、off、および type=disk です。ファイルシステムに shareiscsi=on と設定して、そのファイルシステム内のすべての ZFS ボリュームがデフォルトで共有されるようにすることをお勧めします。ただし、このプロパティーをファイルシステムに設定しても、直接的な効果は何も得られません。
sharenfs	文字列	off	ファイルシステムを NFS 経由で使用できるかどうか、およびどのオプションを使用するかを制御します。on に設定した場合は、zfs share コマンドがオプションなしで呼び出されます。または、このプロパティーの内容に対応するオプションを使って、zfs share コマンドが呼び出されます。off に設定した場合は、従来の share と unshare コマンド、および dfstab ファイルを使用してファイルシステムが管理されます。 ZFS ファイルシステムの共有の詳細については、216 ページの「ZFS ファイルシステムを共有および共有解除する」を参照してください。
snapdir	文字列	hidden	ファイルシステムのルートから .zfs ディレクトリを見えるようにするかどうかを制御します。スナップショットの使用方法の詳細については、225 ページの「ZFS スナップショットの概要」を参照してください。
type	文字列	なし	読み取り専用プロパティー。データセットの種類を調べます。filesystem (ファイルシステムまたはクローン)、volume、または snapshot のいずれかになります。
used	数値	なし	読み取り専用プロパティー。データセットおよびそのすべての子孫が消費するディスク容量を調べます。 詳細については、199 ページの「used プロパティー」を参照してください。
usedbychildren	数値	off	このデータセットの子によって使用されるディスク領域の量を特定する読み取り専用プロパティー。この領域は、データセットのすべての子が破棄されると、解放されます。プロパティーの省略名は usedchild です。

表 6–1 ZFS のネイティブプロパティーの説明 (続き)

プロパティー名	種類	デフォルト値	説明
usedbydataset	数値	off	データセット自体によって使用されるディスク領域の量を特定する読み取り専用プロパティー。この領域は、最初にあらゆるスナップショットが破棄されてから refreservation 予約がすべて削除された後に、データセットが破棄されると、解放されます。プロパティーの省略名は usedds です。
usedbyrefreservation	数値	off	データセットに設定されている refreservation によって使用されるディスク領域の量を特定する読み取り専用プロパティー。この領域は、refreservation が削除されると、解放されます。プロパティーの省略名は usedrefreserv です。
usedbysnapshots	数値	off	データセットのスナップショットによって消費されるディスク領域の量を特定する読み取り専用プロパティー。特に、このディスク領域は、このデータセットのすべてのスナップショットが破棄されると、解放されます。この値はスナップショットの used プロパティーの値を単純に合計した結果ではないことに注意してください。複数のスナップショットで共有されている容量も存在するためです。プロパティーの省略名は usedsnap です。
version	数値	なし	ファイルシステムのディスク上バージョンを識別します。プールのバージョンとは無関係です。このプロパティーは、サポートされるソフトウェアリリースから入手可能な最近のバージョンにしか設定できません。詳細については、zfs upgrade コマンドを参照してください。
volsize	数値	なし	ボリュームの場合に、ボリュームの論理サイズを指定します。 詳細については、202 ページの「volsize プロパティー」を参照してください。

表 6–1 ZFS のネイティブプロパティーの説明 (続き)

プロパティー名	種類	デフォルト値	説明
volblocksize	数値	8K バイト	ボリュームの場合に、ボリュームのブロックサイズを指定します。ボリュームが書き込まれたあとに、ブロックサイズを変更することはできません。ブロックサイズはボリュームを作成するときに設定してください。ボリュームのデフォルトブロックサイズは、8K バイトです。512 バイトから 128K バイトの範囲で、任意の 2 の累乗を指定できます。 このプロパティーの省略名は volblock です。
zoned	ブール型	なし	データセットが非大域ゾーンに追加されているかどうかを指定します。このプロパティーが設定されている場合、そのマウントポイントは大域ゾーンで考慮されません。ZFS では、このようなファイルシステムを要求されても、マウントすることはできません。ゾーンを最初にインストールしたときには、追加されたすべてのファイルシステムにこのプロパティーが設定されます。 ゾーンがインストールされている環境で ZFS を使用する方法の詳細については、288 ページの「ゾーンがインストールされている Solaris システムで ZFS を使用する」を参照してください。
xattr	ブール型	on	このファイルシステムの拡張属性を有効 (on)、無効 (off) のいずれにするかを示します。

ZFS の読み取り専用のネイティブプロパティー

読み取り専用のネイティブプロパティーは、取得はできますが設定はできません。読み取り専用のネイティブプロパティーは継承されません。一部のネイティブプロパティーは、特定の種類のデータセットに固有です。このような場合は、データセットの種類について、表 6–1 の説明の中で記載しています。

読み取り専用のネイティブプロパティーをここに示します。説明は、表 6–1 を参照してください。

- available
- compressratio
- creation
- mounted
- origin

- referenced
- type
- used

  詳細については、199 ページの「used プロパティー」を参照してください。
- usedbychildren
- usedbydataset
- usedbyrefreservation
- usedbysnapshots

used 、referenced、available プロパティーなど、ディスク領域の計上の詳細については、62 ページの「ZFS のディスク領域の計上」を参照してください。

used プロパティー

used プロパティーは読み取り専用のプロパティーであり、このデータセットとそのすべての子孫が消費するディスク容量を特定します。この値は、データの割り当て制限および予約を対象にして確認されます。使用されるディスク領域にデータセットの予約は含まれませんが、子孫のデータセットがある場合はそれらの予約も考慮されます。データセットがその親から継承して消費するディスク容量、およびデータセットが再帰的に破棄されるときに解放されるディスク容量は、使用済み領域および予約の中で大きな割合を占めます。

スナップショットを作成したときは、それらのディスク領域は最初はスナップショットとファイルシステムの間で共有されます。それまでに作成したスナップショットと領域が共有されることもあります。ファイルシステムが変化していくにつれて、それまで共有されていたディスク領域がスナップショット固有になり、スナップショットが使用する領域に計上されます。スナップショットが使用するディスク領域には、その固有データが計上されます。また、スナップショットを削除すると、ほかのスナップショットに固有の (および使用される) ディスク容量を増やすことができます。スナップショットと領域の詳細については、63 ページの「領域が不足した場合の動作」を参照してください。

使用済み、使用可能、参照済みの各ディスク容量には、保留状態の変更は含まれません。保留状態の変更は通常、数秒以内に計上されます。fsync(3c) や O_SYNC 関数を使用してディスクへの変更をコミットしても、ディスク領域の使用状況の情報がすぐに更新されることが保証されているわけではありません。

usedbychildren、usedbydataset、usedbyrefreservation、および usedbysnapshots プロパティーの情報は、zfs list - o space コマンドを使用して表示することができます。これらのプロパティーを使用して、used プロパティーを、子孫によって消費されるディスク領域に分解することができます。詳細は、表 6–1 を参照してください。

設定可能な ZFS ネイティブプロパティー

設定可能なネイティブプロパティーとは、値の取得および設定ができるプロパティーのことです。設定可能なネイティブプロパティーは、`zfs set` コマンド (説明は 206 ページの「ZFS プロパティーを設定する」を参照) または `zfs create` コマンド (説明は 186 ページの「ZFS ファイルシステムを作成する」を参照) を使って設定します。設定可能なネイティブプロパティーは、割り当て制限と予約を除いて継承されます。割り当て制限と予約の詳細については、218 ページの「ZFS の割り当て制限と予約を設定する」を参照してください。

一部の設定可能なネイティブプロパティーは、特定の種類のデータセットに固有です。このような場合は、データセットの種類について、表 6–1 の説明の中で記載しています。特に記載している場合を除いて、プロパティーはすべての種類のデータセットに適用されます。 つまり、ファイルシステム、ボリューム、クローン、およびスナップショットに適用されます。

次のプロパティーは設定可能です。説明は、表 6–1 を参照してください。

- `aclinherit`

  詳細については、249 ページの「ACL プロパティー」を参照してください。
- `aclmode`

  詳細については、249 ページの「ACL プロパティー」を参照してください。
- `atime`
- `canmount`
- チェックサム
- `compression`
- `copies`
- `devices`
- `exec`
- `mountpoint`
- `primarycache`
- `quota`
- `readonly`
- `recordsize`

  詳細については、202 ページの「`recordsize` プロパティー」を参照してください。
- `refquota`
- `refreservation`
- `reservation`

- secondarycache
- shareiscsi
- sharenfs
- setuid
- snapdir
- version
- volsize

  詳細については、202 ページの「volsize プロパティー」を参照してください。
- volblocksize
- zoned
- xattr

canmount プロパティー

canmount プロパティーを off に設定した場合は、zfs mount または zfs mount -a コマンドを使ってファイルシステムをマウントすることはできません。このプロパティーを off に設定することは、mountpoint プロパティーを none に設定することに似ていますが、継承可能な通常の mountpoint プロパティーをデータセットが引き続き保持する点が異なります。たとえば、このプロパティーを off に設定して、子孫のファイルシステム用に継承可能なプロパティーを確立できますが、親ファイルシステム自体がマウントされることもなければ、ユーザーがそれにアクセスすることもできません。この場合、親のファイルシステムは「コンテナ」の役目を果たしているため、そのコンテナにプロパティーを設定することはできますが、コンテナ自体にはアクセスできません。

次の例では、userpool が作成され、その canmount プロパティーが off に設定されます。子孫のユーザーファイルシステムのマウントポイントは、1 つの共通したマウントポイント /export/home に設定されます。親のファイルシステムに設定されたプロパティーは子孫のファイルシステムに継承されますが、親のファイルシステム自体がマウントされることはありません。

```
# zpool create userpool mirror c0t5d0 c1t6d0
# zfs set canmount=off userpool
# zfs set mountpoint=/export/home userpool
# zfs set compression=on userpool
# zfs create userpool/user1
# zfs create userpool/user2
# zfs mount
userpool/user1                  /export/home/user1
userpool/user2                  /export/home/user2
```

canmount プロパティーを noauto に設定した場合、データセットは明示的にマウントする必要があり、自動的にはマウントできません。この値設定は Oracle Solaris

アップグレードソフトウェアで使用され、アクティブなブート環境に属するデータセットだけが起動時にマウントされるようにします。

recordsize プロパティー

recordsize プロパティーは、ファイルシステムに格納するファイルの推奨ブロックサイズを指定します。

このプロパティーは、レコードサイズが固定されているファイルにアクセスするデータベースワークロードだけで使用するように設計されています。ZFS では、標準的なアクセスパターンに最適化された内部アルゴリズムに従って、ブロックサイズが自動的に調整されます。作成されるファイルのサイズが大きく、それらのファイルにさまざまなパターンの小さなブロック単位でアクセスするデータベースの場合には、このようなアルゴリズムが最適でないことがあります。recordsize にデータベースのレコードサイズ以上の値を設定すると、パフォーマンスが大きく向上することがあります。このプロパティーを汎用目的のファイルシステムに使用することは、パフォーマンスが低下する可能性があるため、できるだけ避けてください。指定するサイズは、512 バイト - 128K バイトの 2 の累乗にしてください。ファイルシステムの recordsize 値を変更した場合、そのあとに作成されたファイルだけに適用されます。既存のファイルには適用されません。

このプロパティーの省略名は recsize です。

volsize プロパティー

volsize プロパティーはボリュームの論理サイズを指定します。デフォルトでは、ボリュームを作成するときに、同じ容量の予約が設定されます。volsize への変更があった場合には、予約にも同様の変更が反映されます。これらのチェックは、予期しない動作が起きないようにするために使用されます。ボリュームで使用できる容量が指定した容量より少ない場合には、ボリュームがどのように使用されるかによって異なりますが、定義されていない動作が実行されたりデータが破損したりする可能性があります。このような影響は、ボリュームの使用中にボリュームサイズを変更した場合にも発生することがあります。特に、サイズを縮小した場合にはその可能性が高くなります。ボリュームサイズを調整するときは、特に注意してください。

推奨される方法ではありませんが、zfs create -V に -s フラグを指定するか、またはボリュームの作成後に予約を変更すると、疎ボリュームを作成できます。「疎ボリューム」とは、予約がボリュームサイズと等しくないボリュームのことです。疎ボリュームの場合、volsize を変更しても予約には反映されません。

ボリュームの使用方法の詳細については、285 ページの「ZFS ボリューム」を参照してください。

ZFS ユーザープロパティー

ZFS は、ネイティブプロパティーに加えて、任意のユーザープロパティーもサポートします。ユーザープロパティーは ZFS の動作には影響しませんが、これらを使用すると、使用環境内で意味のある情報をデータセットに注釈として付けることができます。

ユーザープロパティーの名前は、次の規則に適合している必要があります。

- ネイティブプロパティーと区別するためのコロン(':') を含んでいる必要がある。
- 小文字の英字、数字、または次の句読文字を含んでいる必要がある。「:」、「+」、「.」、「_」。
- ユーザープロパティー名の最大長は、256 文字である。

想定されている規則では、プロパティー名は次の 2 つの部分に分割しますが、この名前空間は ZFS によって強制されているものではありません。

module:property

ユーザープロパティーをプログラムで使用する場合、プロパティー名の *module* 要素には、逆順にした DNS ドメイン名を使用してください。これは、それぞれ単独で開発された 2 つのパッケージが、異なる目的で同じプロパティー名を使用する可能性を減らすためです。`com.sun.` で始まるプロパティー名は、Oracle Corporation が使用するために予約されています。

ユーザープロパティーの値は次の規則に準拠する必要があります。

- 常に継承され、決して検証されることのない任意の文字列から構成されている必要がある。
- ユーザープロパティー値の最大長は、1024 文字である。

次に例を示します。

```
# zfs set dept:users=finance userpool/user1
# zfs set dept:users=general userpool/user2
# zfs set dept:users=itops userpool/user3
```

プロパティーを処理するコマンド (`zfs list`、`zfs get`、`zfs set` など) はすべて、ネイティブプロパティーとユーザープロパティーの両方の操作に使用できます。

次に例を示します。

```
zfs get -r dept:users userpool
NAME            PROPERTY    VALUE           SOURCE
userpool        dept:users  all             local
userpool/user1  dept:users  finance         local
userpool/user2  dept:users  general         local
userpool/user3  dept:users  itops           local
```

ユーザープロパティーを消去するには、zfs inherit コマンドを使用します。次に例を示します。

```
# zfs inherit -r dept:users userpool
```

プロパティーがどの親のデータセットにも定義されていない場合は、完全に削除されます。

ZFS ファイルシステムの情報のクエリー検索を行う

zfs list コマンドを使って、データセット情報を表示してクエリー検索を行うことができます。さらに、必要に応じてその操作を拡張することができます。この節では、基本的なクエリーと複雑なクエリーについて説明します。

基本的な ZFS 情報を表示する

zfs list コマンドをオプションなしで使用すると、基本的なデータセット情報を表示できます。このコマンドでは、システム上のすべてのデータセットの名前と、それらの used、available、 referenced、および mountpoint プロパティーの値が表示されます。これらのプロパティーの詳細については、189 ページの「ZFS のプロパティーの紹介」を参照してください。

次に例を示します。

```
# zfs list
NAME                   USED  AVAIL  REFER  MOUNTPOINT
pool                   476K  16.5G    21K  /pool
pool/clone              18K  16.5G    18K  /pool/clone
pool/home              296K  16.5G    19K  /pool/home
pool/home/marks        277K  16.5G   277K  /pool/home/marks
pool/home/marks@snap      0      -   277K  -
pool/test               18K  16.5G    18K  /test
```

このコマンドを使用するときに、コマンド行にデータセット名を指定すれば、特定のデータセットを表示することもできます。また、-r オプションを使って、そのデータセットのすべての子孫を再帰的に表示することもできます。次に例を示します。

```
# zfs list -r pool/home/marks
NAME                   USED  AVAIL  REFER  MOUNTPOINT
pool/home/marks        277K  16.5G   277K  /pool/home/marks
pool/home/marks@snap      0      -   277K  -
```

zfs list コマンドは、ファイルシステムのマウントポイントとともに使用することができます。次に例を示します。

```
# zfs list /pool/home/marks
NAME              USED  AVAIL  REFER  MOUNTPOINT
pool/home/marks   277K  16.5G   277K  /pool/home/marks
```

次の例は、tank/home/chua およびそのすべての子孫データセットの基本情報を表示する方法を示しています。

```
# zfs list -r tank/home/chua
NAME                         USED  AVAIL  REFER  MOUNTPOINT
tank/home/chua                  26.0K  4.81G  10.0K  /tank/home/chua
tank/home/chua/projects       16K  4.81G   9.0K  /tank/home/chua/projects
tank/home/chua/projects/fs1    8K  4.81G     8K  /tank/home/chua/projects/fs1
tank/home/chua/projects/fs2    8K  4.81G     8K  /tank/home/chua/projects/fs2
```

zfs list コマンドの追加情報については、zfs(1M) を参照してください。

複雑な ZFS クエリーを作成する

-o、-t、および -H オプションを使用して、zfs list の出力をカスタマイズすることができます。

-o オプションと必要なプロパティーのコンマ区切りのリストを使用すれば、プロパティー値の出力をカスタマイズできます。任意のデータセットプロパティーを有効な引数として指定できます。サポートされているすべてのデータセットプロパティーのリストは、189 ページの「ZFS のプロパティーの紹介」を参照してください。また、定義されているプロパティー以外に、-o オプションのリストにリテラル name を指定すれば、出力にデータセットの名前が表示されるはずです。

次の例では、zfs list と一緒に sharenfs と mountpoint プロパティー値を使用して、データセット名を表示しています。

```
# zfs list -o name,sharenfs,mountpoint
NAME                  SHARENFS         MOUNTPOINT
tank                  off              /tank
tank/home             on               /tank/home
tank/home/ahrens      on               /tank/home/ahrens
tank/home/bonwick     on               /tank/home/bonwick
tank/home/chua        on               /tank/home/chua
tank/home/eschrock    on               legacy
tank/home/moore       on               /tank/home/moore
tank/home/tabriz      ro               /tank/home/tabriz
```

-t オプションを使用すれば、表示するデータセットの種類を指定できます。次の表は、有効な種類について説明しています。

表 6–2 ZFS データセットの種類

種類	説明
filesystem	ファイルシステムとクローン

表 6–2 ZFS データセットの種類 (続き)

種類	説明
ボリューム	ボリューム
snapshot	スナップショット

-t オプションには、表示するデータセットの種類をコンマ区切りのリストとして指定します。次の例では、-t オプションと - o オプションを同時に使用して、すべてのファイルシステムの名前と used プロパティーを表示しています。

```
# zfs list -t filesystem -o name,used
NAME              USED
pool              476K
pool/clone         18K
pool/home         296K
pool/home/marks   277K
pool/test          18K
```

-H オプションを使用すると、生成される出力から zfs list ヘッダーを省略できます。-H オプションを使用した場合、空白はすべてタブ文字で置き換えられます。このオプションは、スクリプトで使えるようにする場合など、解析しやすい出力を必要とするときに利用できます。次の例では、zfs list コマンドと -H オプションを使用して生成される出力を示しています。

```
# zfs list -H -o name
pool
pool/clone
pool/home
pool/home/marks
pool/home/marks@snap
pool/test
```

ZFS プロパティーを管理する

データセットプロパティーの管理には、zfs コマンドの set、inherit、および get サブコマンドを使用します。

- 206 ページの「ZFS プロパティーを設定する」
- 207 ページの「ZFS プロパティーを継承する」
- 208 ページの「ZFS プロパティーのクエリー検索を行う」

ZFS プロパティーを設定する

zfs set コマンドを使用して、任意の設定可能なデータセットプロパティーを変更できます。あるいは、zfs create コマンドを使用して、データセットの作成時にプロパティーを設定できます。設定可能なデータセットプロパティーのリストは、200 ページの「設定可能な ZFS ネイティブプロパティー」を参照してください。

zfs set コマンドには、*property=value* の形式のプロパティー/値のシーケンスを指定したあと、続けてデータセット名を指定します。各 zfs set 呼び出しで設定または変更できるプロパティーは、1 つだけです。

次の例では、tank/home の atime プロパティーを off に設定しています。

```
# zfs set atime=off tank/home
```

また、どのファイルシステムプロパティーもファイルシステムの作成時に設定できます。次に例を示します。

```
# zfs create -o atime=off tank/home
```

数値プロパティー値を指定する際には、理解が容易な次の接尾辞を使用できます (サイズの小さい順): BKMGTPEZ。これらのすべての接尾辞のあとに、オプションの b (バイト) を続けて指定することができます。ただし、B 接尾辞のあとには指定できません。もともとバイトを表しているためです。次の例にある 4 つの zfs set 呼び出しは、すべて同じ数値を表現しています。つまり、tank/home/marks ファイルシステムの quota プロパティーに 50G バイトの値を設定しています。

```
# zfs set quota=50G tank/home/marks
# zfs set quota=50g tank/home/marks
# zfs set quota=50GB tank/home/marks
# zfs set quota=50gb tank/home/marks
```

数値でないプロパティーの値では、大文字と小文字が区別されるので、小文字を使用する必要があります。ただし、mountpoint および sharenfs は例外です。これらのプロパティーの値には、大文字と小文字を混在させることができます。

zfs set コマンドの詳細については、zfs(1M) を参照してください。

ZFS プロパティーを継承する

割り当て制限と予約を除いて、すべての設定可能なプロパティーは、親データセットから値を継承します。ただし、子孫データセットに対して割り当て制限または予約が明示的に設定されている場合は継承されません。継承するプロパティーについて、明示的な値が祖先に設定されていない場合は、プロパティーのデフォルト値が使用されます。zfs inherit コマンドを使用して、プロパティー値を消去することができます。その場合は、親データセットの値を継承することになります。

次の例では、zfs set コマンドを使用して tank/home/bonwick ファイルシステムの圧縮を有効にしています。次に、zfs inherit を使用して、compression プロパティーをクリアしています。この結果、このプロパティーはデフォルト値の off を継承します。home と tank の compression プロパティーはローカルに設定されていないため、デフォルト値が使用されます。圧縮が両方とも有効になっていた場合は、すぐ上の祖先 (この例では home) に設定されている値が使用されます。

```
# zfs set compression=on tank/home/bonwick
# zfs get -r compression tank
NAME             PROPERTY      VALUE                     SOURCE
tank             compression   off                       default
tank/home        compression   off                       default
tank/home/bonwick compression   on                       local
# zfs inherit compression tank/home/bonwick
# zfs get -r compression tank
NAME             PROPERTY      VALUE                     SOURCE
tank             compression   off                       default
tank/home        compression   off                       default
tank/home/bonwick compression  off                       default
```

-r オプションを指定すると、inherit サブコマンドが再帰的に適用されます。次の例では、このコマンドによって、compression プロパティーの値が tank/home およびそのすべての子孫に継承されます。

```
# zfs inherit -r compression tank/home
```

注 – -r オプションを使用すると、すべての子孫のデータセットに割り当てられている現在のプロパティー設定が消去されることに注意してください。

---

zfs inherit コマンドの詳細については、zfs(1M) を参照してください。

ZFS プロパティーのクエリー検索を行う

プロパティー値のクエリー検索を行うもっとも簡単な方法は、zfs list コマンドを使用することです。詳細については、204 ページの「基本的な ZFS 情報を表示する」を参照してください。ただし、複雑なクエリーを使用する場合およびスクリプトで使用する場合は、より詳細な情報をカスタマイズした書式で渡すために zfs get コマンドを使用します。

zfs get コマンドを使用して、任意のデータセットプロパティーを取得できます。次の例は、データセット上の 1 つのプロパティー値を取得する方法を示しています。

```
# zfs get checksum tank/ws
NAME             PROPERTY       VALUE                     SOURCE
tank/ws          checksum       on                        default
```

4 番目の列 SOURCE は、このプロパティー値の起点を示します。次の表は、表示される可能性のあるソース値を定義したものです。

表 6–3 出力される可能性のある SOURCE 値 (zfs get コマンド)

ソース値	説明
default	このプロパティー値は、このデータセットまたはその祖先 (存在する場合) で明示的に設定されたことが一度もありません。このプロパティーのデフォルト値が使用されています。
inherited from *dataset-name*	このプロパティー値は、*dataset-name* に指定されている親データセットから継承されます。
local	このプロパティー値は、zfs set を使って、このデータセットに明示的に設定されました。
temporary	このプロパティー値は、zfs mount - o オプションを使って設定され、マウントの有効期間だけ有効です。一時的なマウントプロパティーの詳細については、215 ページの「一時的なマウントプロパティーを使用する」を参照してください。
- (なし)	このプロパティーは読み取り専用です。値は ZFS によって生成されます。

特殊キーワード all を使って、すべてのデータセットプロパティー値を取得できます。all キーワードの使用例を次に示します。

```
# zfs get all tank/home
NAME       PROPERTY              VALUE                  SOURCE
tank/home  type                  filesystem             -
tank/home  creation              Tue Jun 29 11:44 2010  -
tank/home  used                  21K                    -
tank/home  available             66.9G                  -
tank/home  referenced            21K                    -
tank/home  compressratio         1.00x                  -
tank/home  mounted               yes                    -
tank/home  quota                 none                   default
tank/home  reservation           none                   default
tank/home  recordsize            128K                   default
tank/home  mountpoint            /tank/home             default
tank/home  sharenfs              off                    default
tank/home  checksum              on                     default
tank/home  compression           off                    default
tank/home  atime                 on                     default
tank/home  devices               on                     default
tank/home  exec                  on                     default
tank/home  setuid                on                     default
tank/home  readonly              off                    default
tank/home  zoned                 off                    default
tank/home  snapdir               hidden                 default
tank/home  aclmode               groupmask              default
tank/home  aclinherit            restricted             default
tank/home  canmount              on                     default
tank/home  shareiscsi            off                    default
tank/home  xattr                 on                     default
tank/home  copies                1                      default
tank/home  version               4                      -
```

```
tank/home  utf8only              off                    -
tank/home  normalization         none                   -
tank/home  casesensitivity       sensitive              -
tank/home  vscan                 off                    default
tank/home  nbmand                off                    default
tank/home  sharesmb              off                    default
tank/home  refquota              none                   default
tank/home  refreservation        none                   default
tank/home  primarycache          all                    default
tank/home  secondarycache        all                    default
tank/home  usedbysnapshots       0                      -
tank/home  usedbydataset         21K                    -
tank/home  usedbychildren        0                      -
tank/home  usedbyrefreservation  0                      -
tank/home  logbias               latency                default
```

注 – Oracle Solaris 10 リリースでは Oracle Solaris SMB サービスがサポートされていないため、Oracle Solaris 10 リリースの `casesensitivity`、`nbmand`、`normalization`、`sharesmb`、`utf8only`、および `vscan` プロパティーは完全には機能しません。

`zfs get` に `-s` オプションを使用すると、表示するプロパティーをソースの種類ごとに指定できます。このオプションには、必要なソースの種類をコンマ区切りのリストとして指定します。指定したソースの種類のプロパティーだけが表示されます。有効なソースの種類は、`local`、`default`、`inherited`、`temporary`、および `none` です。次の例では、`pool` 上にローカルに設定されているすべてのプロパティーを表示しています。

```
# zfs get -s local all pool
NAME             PROPERTY      VALUE                     SOURCE
pool             compression   on                        local
```

前述のどのオプションの場合にも、`-r` オプションを使用して、指定したデータセットのすべての子に設定されている特定のプロパティーを再帰的に表示することができます。次の例では、`tank` に含まれるすべてのデータセットに設定されている、すべての一時的なプロパティーが再帰的に表示されます。

```
# zfs get -r -s temporary all tank
NAME             PROPERTY       VALUE                      SOURCE
tank/home          atime          off                      temporary
tank/home/bonwick  atime          off                      temporary
tank/home/marks    atime          off                      temporary
```

`zfs get` コマンドでは、ターゲットのファイルシステムを指定せずにプロパティー値のクエリーを行うことが可能です。これは、すべてのプールやファイルシステムがコマンドの処理対象となることを意味します。次に例を示します。

```
# zfs get -s local all
tank/home               atime          off                    local
tank/home/bonwick       atime          off                    local
tank/home/marks         quota          50G                    local
```

zfs get コマンドの詳細については、zfs(1M) を参照してください。

スクリプトで使用できるように ZFS プロパティーのクエリー検索を行う

zfs get コマンドでは、スクリプトで使用できるように設計された -H および - o オプションを利用できます。-H オプションを使用すると、ヘッダー情報を省略し、空白をタブ文字で置き換えることができます。空白が揃うことで、データが見やすくなります。-o オプションを使用すると、次の方法で出力をカスタマイズできます。

- リテラル name は、189 ページの「ZFS のプロパティーの紹介」節で定義したプロパティーのコンマ区切りリストと組み合わせて使用できます。
- 出力対象となるリテラルフィールド name、value、property、および source のコンマ区切りリストのあとに、空白 1 つと引数 1 つ。この引数は、プロパティーのコンマ区切りリストとなります。

次の例では、-zfs get の -H および o オプションを使用して、1 つの値を取得する方法を示しています。

```
# zfs get -H -o value compression tank/home
on
```

-p オプションを指定すると、数値が正確な値として出力されます。たとえば、1M バイトは 1000000 として出力されます。このオプションは、次のように使用できます。

```
# zfs get -H -o value -p used tank/home
182983742
```

前述のどのオプションの場合にも、-r オプションを使用して、要求した値をすべての子孫について再帰的に取得できます。次の例では、-H、-o、および -r オプションを使用して、export/home およびその子孫のデータセット名と used プロパティーの値を取得しています。ヘッダー出力は省略されています。

```
# zfs get -H -o name,value -r used export/home
export/home             5.57G
export/home/marks       1.43G
export/home/maybee      2.15G
```

ZFS ファイルシステムをマウントおよび共有する

この節では、ZFS でマウントポイントと共有ファイルシステムをどのように管理するかについて説明します。

- 212 ページの「ZFS マウントポイントを管理する」
- 214 ページの「ZFS ファイルシステムをマウントする」
- 215 ページの「一時的なマウントプロパティーを使用する」

- 216 ページの「ZFS ファイルシステムをマウント解除する」
- 216 ページの「ZFS ファイルシステムを共有および共有解除する」

ZFS マウントポイントを管理する

デフォルトで、ZFS ファイルシステムは作成時に自動的にマウントされます。この節で説明するように、ユーザーはファイルシステムの特定のマウントポイント動作を決定することができます。

`zpool create` の `-m` オプションを使用すれば、プールを作成するときにプールのデータセットのデフォルトマウントポイントを設定することもできます。プールの作成の詳細については、72 ページの「ZFS ストレージプールを作成する」を参照してください。

すべての ZFS ファイルシステムは、ZFS の起動時にサービス管理機能 (SMF) の `svc://system/filesystem/local` サービスを使用してマウントされます。ファイルシステムは、*/path* にマウントされます。*path* はファイルシステムの名前です。

デフォルトのマウントポイントを上書きするには、`zfs set` コマンド使って `mountpoint` プロパティーを特定のパスに設定します。ZFS では、`zfs mount -a` コマンドが呼び出されるときに、指定されたマウントポイントが必要に応じて自動的に作成され、関連付けられたファイルシステムが自動的にマウントされます。`/etc/vfstab` ファイルを編集する必要はありません。

`mountpoint` プロパティーは継承されます。たとえば、`pool/home` の `mountpoint` プロパティーが `/export/stuff` に設定されている場合、`pool/home/user` は `mountpoint` プロパティー値の `/export/stuff/user` を継承します。

ファイルシステムがマウントされないようにするには、`mountpoint` プロパティーを `none` に設定します。さらに、`canmount` プロパティーを使えば、ファイルシステムをマウント可能にするかどうかを制御できます。`canmount` プロパティーの詳細については、201 ページの「canmount プロパティー」を参照してください。

また、従来のマウントインタフェース経由でファイルシステムを明示的に管理することもできます。それには、`zfs set` を使って `mountpoint` プロパティーを `legacy` に設定します。このようにすると、ファイルシステムが自動的にマウントおよび管理されなくなります。代わりに、`mount` や `umount` コマンドなどのレガシーツールと、`/etc/vfstab` ファイルを使用する必要があります。レガシーマウントの詳細については、213 ページの「レガシーマウントポイント」を参照してください。

自動マウントポイント

- `mountpoint` プロパティーを `legacy` または `none` から特定のパスに変更すると、ZFS はそのファイルシステムを自動的にマウントします。

- ファイルシステムが ZFS によって管理されているのにそのマウントが現在解除されている場合は、mountpoint プロパティーを変更しても、そのファイルシステムのマウントは解除されたままになります。

mountpoint プロパティーが legacy に設定されていないデータセットは、すべて ZFS によって管理されます。次の例では、作成されたデータセットのマウントポイントが ZFS によって自動的に管理されます。

```
# zfs create pool/filesystem
# zfs get mountpoint pool/filesystem
NAME             PROPERTY      VALUE                     SOURCE
pool/filesystem  mountpoint    /pool/filesystem          default
# zfs get mounted pool/filesystem
NAME             PROPERTY      VALUE                     SOURCE
pool/filesystem  mounted       yes                       -
```

次の例に示すように、mountpoint プロパティーを明示的に設定することもできます。

```
# zfs set mountpoint=/mnt pool/filesystem
# zfs get mountpoint pool/filesystem
NAME             PROPERTY      VALUE                     SOURCE
pool/filesystem  mountpoint    /mnt                      local
# zfs get mounted pool/filesystem
NAME             PROPERTY      VALUE                     SOURCE
pool/filesystem  mounted       yes                       -
```

mountpoint プロパティーを変更すると、ファイルシステムが古いマウントポイントから自動的にマウント解除されて、新しいマウントポイントに再マウントされます。マウントポイントのディレクトリは必要に応じて作成されます。ファイルシステムがアクティブであるためにマウント解除できない場合は、エラーが報告され、手動で強制的にマウント解除する必要があります。

レガシーマウントポイント

mountpoint プロパティーを legacy に設定することで、ZFS ファイルシステムをレガシーツールを使って管理することができます。レガシーファイルシステムは、mount と umount コマンド、および /etc/vfstab ファイルを使用して管理する必要があります。レガシーファイルシステムは、ZFS が起動するときに自動的にマウントされません。ZFS の mount および umount コマンドは、この種類のデータセットでは使用できません。次の例では、ZFS データセットをレガシーモードで設定および管理する方法を示しています。

```
# zfs set mountpoint=legacy tank/home/eschrock
# mount -F zfs tank/home/eschrock /mnt
```

起動時にレガシーファイルシステムを自動的にマウントするには、/etc/vfstab ファイルにエントリを追加する必要があります。次に、/etc/vfstab ファイル内のエントリの例を示します。

```
#device         device        mount           FS      fsck    mount   mount
#to mount       to fsck       point           type    pass    at boot options
#

tank/home/eschrock -        /mnt           zfs        -        yes        -
```

device to fsck エントリと fsck pass エントリは - に設定されていますが、これは、fsck コマンドが ZFS ファイルシステムで使用できないからです。ZFS データの整合性の詳細については、47 ページの「トランザクションのセマンティクス」を参照してください。

ZFS ファイルシステムをマウントする

ZFS では、ファイルシステムが作成されるときまたはシステムが起動するときに、ファイルシステムが自動的にマウントされます。zfs mount コマンドを使用する必要があるのは、マウントオプションを変更したりファイルシステムを明示的にマウントまたはマウント解除したりする必要がある場合だけです。

zfs mount コマンドを引数なしで実行すると、現在マウントされているファイルシステムのうち、ZFS が管理しているファイルシステムがすべて表示されます。レガシー管理されているマウントポイントは表示されません。次に例を示します。

```
# zfs mount
tank                            /tank
tank/home                       /tank/home
tank/home/bonwick               /tank/home/bonwick
tank/ws                         /tank/ws
```

-a オプションを使用すると、ZFS が管理しているファイルシステムをすべてマウントできます。レガシー管理されているファイルシステムはマウントされません。次に例を示します。

```
# zfs mount -a
```

デフォルトでは、空でないディレクトリの最上位にマウントすることは許可されません。空でないディレクトリの最上位に強制的にマウントする場合は、-O オプションを使用する必要があります。次に例を示します。

```
# zfs mount tank/home/lalt
cannot mount '/export/home/lalt': directory is not empty
use legacy mountpoint to allow this behavior, or use the -O flag
# zfs mount -O tank/home/lalt
```

レガシーマウントポイントは、レガシーツールを使って管理する必要があります。ZFS ツールを使用しようとすると、エラーになります。次に例を示します。

```
# zfs mount pool/home/billm
cannot mount 'pool/home/billm': legacy mountpoint
use mount(1M) to mount this filesystem
# mount -F zfs tank/home/billm
```

ファイルシステムがマウントされるときに使用されるマウントオプションは、データセットに関連付けられたプロパティー値に基づいて決まります。プロパティーとマウントオプションは、次のような関係になっています。

表 6–4 ZFS のマウント関連プロパティーとマウントオプション

プロパティー	マウントオプション
atime	atime/noatime
devices	devices/nodevices
exec	exec/noexec
nbmand	nbmand/nonbmand
readonly	ro/rw
setuid	setuid/nosetuid
xattr	xattr/noaxttr

マウントオプション nosuid は、nodevices,nosetuid の別名です。

一時的なマウントプロパティーを使用する

前節で説明したどのマウントオプションの場合にも、-zfs mount コマンドと o オプションを使って明示的に設定されている場合には、関連するプロパティー値が一時的に上書きされます。これらのプロパティー値は zfs get コマンドを実行すると temporary として報告されますが、ファイルシステムがマウント解除されるときに元の値に戻ります。データセットがマウントされるときにプロパティー値を変更した場合は、変更がすぐに有効になり、一時的な設定がすべて上書きされます。

次の例では、tank/home/perrin ファイルシステムに読み取り専用マウントオプションが一時的に設定されます。ファイルシステムがマウント解除されているものと仮定しています。

```
# zfs mount -o ro tank/home/perrin
```

現在マウントされているファイルシステムのプロパティー値を一時的に変更するときは、特別な remount オプションを使用する必要があります。次の例では、現在マウントされているファイルシステムの atime プロパティーを一時的に off に変更しています。

```
# zfs mount -o remount,noatime tank/home/perrin
# zfs get atime tank/home/perrin
NAME              PROPERTY      VALUE                     SOURCE
tank/home/perrin atime         off                       temporary
```

zfs mount コマンドの詳細については、zfs(1M) を参照してください。

ZFS ファイルシステムをマウント解除する

zfs unmount サブコマンドを使用して、ZFS ファイルシステムをマウント解除できます。unmount コマンドの引数として、マウントポイントまたはファイルシステム名のいずれかを指定できます。

次の例では、ファイルシステム名を使ってファイルシステムをマウント解除しています。

```
# zfs unmount tank/home/tabriz
```

次の例では、マウントポイントを使ってファイルシステムをマウント解除しています。

```
# zfs unmount /export/home/tabriz
```

ファイルシステムがビジー状態の場合には、unmount コマンドは失敗します。ファイルシステムを強制的にマウント解除する場合は、-f オプションを使用できます。アクティブに使用されているファイルシステムを強制的にマウント解除する場合は、十分に注意してください。アプリケーションが予期しない動作を行うことがあります。

```
# zfs unmount tank/home/eschrock
cannot unmount '/export/home/eschrock': Device busy
# zfs unmount -f tank/home/eschrock
```

下位互換性を提供するために、従来の umount コマンドを使用して ZFS ファイルシステムをマウント解除することもできます。次に例を示します。

```
# umount /export/home/bob
```

zfs umount コマンドの詳細については、zfs(1M) を参照してください。

ZFS ファイルシステムを共有および共有解除する

ZFS でファイルシステムを自動的に共有するには、sharenfs プロパティーを設定します。このプロパティーを使用すれば、新規ファイルシステムを共有するときに /etc/dfs/dfstab ファイルを変更する必要はありません。sharenfs プロパティーは、コンマ区切りのオプションリストとして、share コマンドに渡されます。値 on は、デフォルトの共有オプションの別名で、すべてのユーザーに read/write アクセス権を提供します。値 off を指定すると、ファイルシステムが ZFS によって管理されなくなり、従来の方法 (/etc/dfs/dfstab ファイルなど) で共有できるようになります。sharenfs プロパティーが off でないすべてのファイルシステムは、起動時に共有されます。

共有セマンティクスを制御する

デフォルトでは、すべてのファイルシステムは共有されていません。新規ファイルシステムを共有する場合は、次のような zfs set 構文を使用します。

```
# zfs set sharenfs=on tank/home/eschrock
```

sharenfs プロパティーは継承されます。継承したプロパティーが off でないファイルシステムは、作成時に自動的に共有されます。次に例を示します。

```
# zfs set sharenfs=on tank/home
# zfs create tank/home/bricker
# zfs create tank/home/tabriz
# zfs set sharenfs=ro tank/home/tabriz
```

tank/home/bricker と tank/home/tabriz は、tank/home から sharenfs プロパティーを継承するため、最初は書き込み可能として共有されます。このプロパティーが ro (読み取り専用) に設定されたあとは、tank/home に設定されている sharenfs プロパティーに関係なく、tank/home/tabriz は読み取り専用として共有されます。

ZFS ファイルシステムの共有を解除する

ほとんどのファイルシステムは、起動、作成、および破棄されるときに自動的に共有または共有解除されますが、場合によってはファイルシステムの共有を明示的に解除しなければならないことがあります。このような場合は、zfs unshare コマンドを使用します。次に例を示します。

```
# zfs unshare tank/home/tabriz
```

このコマンドを実行すると、tank/home/tabriz ファイルシステムの共有が解除されます。システムの上のすべての ZFS ファイルシステムを共有解除する場合は、-a オプションを使用する必要があります。

```
# zfs unshare -a
```

ZFS ファイルシステムを共有する

通常の操作にはほとんどの場合、起動時や作成時のファイルシステムの共有に関する ZFS の自動動作で十分です。なんらかの理由でファイルシステムの共有を解除する場合でも、zfs share コマンドを使用すれば再度共有できます。次に例を示します。

```
# zfs share tank/home/tabriz
```

- a オプションを使用すれば、システム上のすべての ZFS ファイルシステムを共有できます。

```
# zfs share -a
```

従来の共有の動作

sharenfs プロパティーが off に設定された場合は、ZFS はどのような場合にもファイルシステムを共有または共有解除することがありません。この値を使用すれば、/etc/dfs/dfstab ファイルなどの従来の方法でファイルシステムの共有を管理することができます。

従来の mount コマンドと異なり、従来の share および unshare コマンドは ZFS ファイルシステムでも使用できます。このため、sharenfs プロパティーのオプションとは異なるオプションを使って、ファイルシステムを手動で共有することもできます。この管理モデルは推奨されていません。ZFS を使用して NFS 共有を完全に管理するか、または /etc/dfs/dfstab ファイルを使用して完全に管理する方法を選択してください。ZFS 管理モデルは、従来のモデルより少ない操作で簡単に管理できるように設計されています。

ZFS の割り当て制限と予約を設定する

quota プロパティーを使用して、ファイルシステムが使用できるディスク容量を制限できます。また、reservation プロパティーを使用して、指定されたディスク容量をファイルシステムが使用できることを保証することもできます。これらのプロパティーは、設定したデータセットとそのデータセットのすべての子孫に適用されます。

つまり、割り当て制限を tank/home データセットに設定した場合は、tank/home およびそのすべての子孫が使用するディスク容量の合計がその割り当て制限を超えることができなくなります。同様に、tank/home に予約を設定した場合は、tank/home およびそのすべての子孫がその予約を利用することになります。データセットとそのすべての子孫が使用するディスク容量は、used プロパティーによって報告されます。

refquota プロパティーと refreservation プロパティーは、スナップショットやクローンなどの子孫で消費されるディスク容量を計上せずにファイルシステムの容量を管理するために使用されます。

この Solaris リリースでは、特定のユーザーまたはグループが所有するファイルによって消費されるディスク領域の量に割り当て制限を設定することができます。ファイルシステムバージョン 4 より古いファイルシステム上のボリューム、またはバージョン 15 より古いプール上のボリュームには、ユーザーおよびグループの割り当て制限プロパティーを設定できません。

ファイルシステムを管理するために、割り当て制限と予約の機能としてどれがもっとも役立つかを判断するには、次の点を考慮してください。

- quota プロパティーと reservation プロパティーは、データセットとその子孫が消費するディスク容量を管理する場合に便利です。

- refquota プロパティーと refreservation プロパティーは、データセットが消費するディスク容量を管理する場合に適しています。
- refquota または refreservation プロパティーに、quota または reservation プロパティーより大きい値を設定しても、何の効果もありません。quota プロパティーまたは refquota プロパティーを設定した場合、どちらかの値を超えるような操作は失敗します。refquota より大きい値の quota 値を超える場合もあります。たとえば、スナップショットのブロックの一部が変更された場合は、refquota を超える前に実際に quota を超える可能性があります。
- ユーザーおよびグループの割り当てを制限することによって、大学などのような、多数のユーザーアカウントが存在する環境でディスクスペースを簡単に管理できるようになります。

割り当て制限と予約の設定方法の詳細については、219 ページの「ZFS ファイルシステムに割り当て制限を設定する」および223 ページの「ZFS ファイルシステムに予約を設定する」を参照してください。

ZFS ファイルシステムに割り当て制限を設定する

ZFS ファイルシステムの割り当て制限は、zfs set および zfs get コマンドを使用して設定および表示できます。次の例では、10G バイトの割り当て制限が tank/home/bonwick に設定されます。

```
# zfs set quota=10G tank/home/bonwick
# zfs get quota tank/home/bonwick
NAME              PROPERTY      VALUE                     SOURCE
tank/home/bonwick quota         10.0G                     local
```

割り当て制限を設定すると、zfs list および df コマンドの出力も変化します。次に例を示します。

```
# zfs list
NAME                   USED  AVAIL  REFER  MOUNTPOINT
tank/home             16.5K  33.5G  8.50K  /export/home
tank/home/bonwick     15.0K  10.0G  8.50K  /export/home/bonwick
tank/home/bonwick/ws  6.50K  10.0G  8.50K  /export/home/bonwick/ws
# df -h /export/home/bonwick
Filesystem             size   used  avail capacity  Mounted on
tank/home/bonwick       10G     8K    10G     1%    /export/home/bonwick
```

tank/home は 33.5G バイトのディスク容量を使用できますが、tank/home/bonwick と tank/home/bonwick/ws は 10G バイトしか使用できません。これは、tank/home/bonwick に割り当て制限が設定されているためです。

割り当て制限には、データセットが現在使用している容量より少ない容量を設定することはできません。次に例を示します。

```
# zfs set quota=10K tank/home/bonwick
cannot set quota for 'tank/home/bonwick': size is less than current used or
reserved space
```

データセットに refquota を設定して、データセットが消費できるディスク容量を制限できます。この強い制限値には、子孫が消費するディスク容量は含まれません。次に例を示します。

```
# zfs set refquota=10g students/studentA
# zfs list
NAME                 USED  AVAIL  REFER  MOUNTPOINT
profs                106K  33.2G    18K  /profs
students            57.7M  33.2G    19K  /students
students/studentA   57.5M  9.94G  57.5M  /students/studentA
# zfs snapshot students/studentA@today
# zfs list
NAME                       USED  AVAIL  REFER  MOUNTPOINT
profs                      106K  33.2G    18K  /profs
students                  57.7M  33.2G    19K  /students
students/studentA         57.5M  9.94G  57.5M  /students/studentA
students/studentA@today       0      -  57.5M  -
```

利便性を高めるために、データセットに別の割り当て制限を設定して、スナップショットで消費されるディスク容量を管理することもできます。次に例を示します。

```
# zfs set quota=20g students/studentA
# zfs list
NAME                       USED  AVAIL  REFER  MOUNTPOINT
profs                      106K  33.2G    18K  /profs
students                  57.7M  33.2G    19K  /students
students/studentA         57.5M  9.94G  57.5M  /students/studentA
students/studentA@today       0      -  57.5M  -
```

このシナリオでは、studentA が refquota (10G バイト) の強い制限に到達する可能性がありますが、スナップショットが存在している場合でも回復のためにファイルを削除することができます。

上の例では、2 つの割り当て制限 (10G バイトと 20G バイト) の小さいほうが、zfs list 出力に表示されています。両方の割り当て制限を表示するには、zfs get コマンドを使用します。次に例を示します。

```
# zfs get refquota,quota students/studentA
NAME               PROPERTY  VALUE              SOURCE
students/studentA  refquota  10G                local
students/studentA  quota     20G                local
```

ZFS ファイルシステムでユーザーおよびグループの割り当て制限を設定する

ユーザー割り当て制限またはグループ割り当て制限を設定するには、それぞれ zfs userquota コマンドまたは zfs groupquota コマンドを使用します。次に例を示します。

```
# zfs create students/compsci
# zfs set userquota@student1=10G students/compsci
# zfs create students/labstaff
# zfs set groupquota@staff=20GB students/labstaff
```

現在のユーザーまたはグループの割り当て制限が次のように表示されます。

```
# zfs get userquota@student1 students/compsci
NAME              PROPERTY            VALUE               SOURCE
students/compsci  userquota@student1  10G                 local
# zfs get groupquota@staff students/labstaff
NAME               PROPERTY          VALUE             SOURCE
students/labstaff  groupquota@staff  20G               local
```

次のプロパティーのクエリーによって、ユーザーまたはグループの全般的なディスク領域使用状況を表示することができます。

```
# zfs userspace students/compsci
TYPE        NAME      USED  QUOTA
POSIX User  root      227M   none
POSIX User  student1  455M    10G
# zfs groupspace students/labstaff
TYPE         NAME   USED  QUOTA
POSIX Group  root   217M   none
POSIX Group  staff  217M    20G
```

個々のユーザーやグループのディスク領域の使用状況を特定するには、次のプロパティーのクエリーを行います。

```
# zfs get userused@student1 students/compsci
NAME              PROPERTY           VALUE              SOURCE
students/compsci  userused@student1  455M               local
# zfs get groupused@staff students/labstaff
NAME               PROPERTY         VALUE            SOURCE
students/labstaff  groupused@staff  217M             local
```

`zfs get all` *dataset* コマンドを使用しても、ユーザーおよびグループの割り当て制限プロパティーは表示されず、その他のすべてのファイルシステムプロパティーの一覧が表示されるだけです。

ユーザー割り当て制限またはグループ割り当て制限は、次のようにして解除することができます。

```
# zfs set userquota@user1=none students/compsci
# zfs set groupquota@staff=none students/labstaff
```

ZFS ファイルシステムのユーザーおよびグループ割り当て制限で提供される機能は、次のとおりです。

- 親ファイルシステムで設定されたユーザー割り当て制限またはグループ割り当て制限は、自動的には子孫のファイルシステムに継承されません。

- ただし、ユーザーまたはグループの割り当て制限が設定されているファイルシステムのクローンまたはスナップショットを作成した場合には、それらの割り当て制限が適用されます。同様に、`zfs send` コマンド (`-R` オプションなしでも可) を使用してストリームを作成した場合にも、ユーザーまたはグループの割り当て制限がファイルシステムに組み込まれます。
- 非特権ユーザーは、自身のディスク領域使用状況のみを確認することができます。root ユーザー、または `userused` 権限や `groupused` 権限を持っているユーザーは、あらゆるユーザーまたはグループのディスク領域アカウント情報にアクセスすることができます。
- `userquota` および `groupquota` プロパティーは、ZFS ボリューム、バージョン 4 よりも古いファイルシステム、またはバージョン 15 よりも古いプールでは設定できません。

ユーザーまたはグループの割り当て制限が適用されるのが数秒遅れることがあります。そのような遅延が発生する場合は、割り当て制限を超えているのでこれ以上は書き込みが許可されないことが `EDQUOT` エラーメッセージによって通知される前にユーザーが自身の割り当て制限を超えている可能性があります。

従来の `quota` コマンドを使用して、NFS 環境 (例えば、ZFS ファイルシステムがマウントされているものなど) におけるユーザーの割り当て制限を確認することができます。ユーザーが割り当て制限を超えてる場合は、何もオプションを指定しなくても、`quota` コマンドだけで、出力情報が表示されます。次に例を示します。

```
# zfs set userquota@student1=10m students/compsci
# zfs userspace students/compsci
TYPE        NAME      USED  QUOTA
POSIX User  root      227M   none
POSIX User  student1  455M    10M
# quota student1
Block limit reached on /students/compsci
```

ユーザーの割り当て制限をリセットして制限を超えることがないようにする場合は、`quota -v` コマンドを使用してユーザーの割り当てを確認することができます。次に例を示します。

```
# zfs set userquota@student1=10GB students/compsci
# zfs userspace students/compsci
TYPE        NAME      USED  QUOTA
POSIX User  root      227M   none
POSIX User  student1  455M    10G
# quota student1
# quota -v student1
Disk quotas for student1 (uid 201):
Filesystem     usage  quota  limit    timeleft  files  quota  limit    timeleft
/students/compsci
              466029 10485760 10485760
```

ZFS ファイルシステムに予約を設定する

ZFS の「予約」とは、データセットが使用できることを保証された、プールから割り当てられたディスク領域のことです。つまり、プールで現在使用できないディスク容量をデータセットのディスク容量として予約することはできません。未処理の使用されていない予約の合計容量が、プールで消費されていないディスク容量を超えることはできません。ZFS の予約は、`zfs set` および `zfs get` コマンドを使用して設定および表示できます。次に例を示します。

```
# zfs set reservation=5G tank/home/moore
# zfs get reservation tank/home/moore
NAME             PROPERTY     VALUE   SOURCE
tank/home/moore  reservation  5G      local
```

予約を設定すると、`zfs list` コマンドの出力が変化する可能性があります。次に例を示します。

```
# zfs list
NAME                   USED  AVAIL  REFER  MOUNTPOINT
tank/home             5.00G  33.5G  8.50K  /export/home
tank/home/moore       15.0K  33.5G  8.50K  /export/home/moore
```

`tank/home` は 5G バイトのディスク容量を使用していますが、`tank/home` とそのすべての子孫が参照しているディスク容量の合計は 5G バイト未満です。使用されている容量には、`tank/home/moore` に予約されている容量が反映されます。予約は、親データセットが使用しているディスク容量の計算時に計上されるので、親データセットの割り当て制限または予約、あるいはその両方を減らすことになります。

```
# zfs set quota=5G pool/filesystem
# zfs set reservation=10G pool/filesystem/user1
cannot set reservation for 'pool/filesystem/user1': size is greater than
available space
```

データセットは、予約より多くのディスク容量を使用できます。ただし、プールの中で予約されていない領域があり、データセットが現在使用している容量が割り当て制限に達していないことが条件です。データセットは、別のデータセットに予約されているディスク容量を使用することはできません。

予約は加算されません。つまり、`zfs set` をもう一度呼び出して予約を設定しても、既存の予約に新しい予約が追加されることはありません。代わりに、既存の予約が 2 番目の予約で置き換えられます。次に例を示します。

```
# zfs set reservation=10G tank/home/moore
# zfs set reservation=5G tank/home/moore
# zfs get reservation tank/home/moore
NAME             PROPERTY      VALUE                      SOURCE
tank/home/moore  reservation   5.00G                      local
```

`refreservation` 予約を設定すると、スナップショットとクローンで消費されるディスク容量は含めずに、データセットのディスク容量を保証することができま

す。この予約は、親データセットの使用済み容量の計算時に計上されるので、親データセットの割り当て制限と予約を減らすことになります。次に例を示します。

```
# zfs set refreservation=10g profs/prof1
# zfs list
NAME                     USED  AVAIL  REFER  MOUNTPOINT
profs                   10.0G  23.2G    19K  /profs
profs/prof1               10G  33.2G    18K  /profs/prof1
```

同じデータセットに予約を設定して、データセットの容量とスナップショットの容量を確保することもできます。次に例を示します。

```
# zfs set reservation=20g profs/prof1
# zfs list
NAME                     USED  AVAIL  REFER  MOUNTPOINT
profs                   20.0G  13.2G    19K  /profs
profs/prof1               10G  33.2G    18K  /profs/prof1
```

通常の予約は、親の使用済み容量の計算時に計上されます。

上の例では、2 つの割り当て制限 (10G バイトと 20G バイト) の小さいほうが、`zfs list` 出力に表示されています。両方の割り当て制限を表示するには、`zfs get` コマンドを使用します。次に例を示します。

```
# zfs get reservation,refreserv profs/prof1
NAME         PROPERTY        VALUE        SOURCE
profs/prof1  reservation     20G          local
profs/prof1  refreservation  10G          local
```

`refreservation` を設定すると、スナップショットを作成できるのは、データセットの *referenced* の現在のバイト数を格納できるだけの未予約プール領域が、この予約容量のほかに存在する場合だけになります。

◆ ◆ ◆ 第 7 章

Oracle Solaris ZFS のスナップショットとクローンの操作

この章では、Oracle Solaris ZFS のスナップショットとクローンを作成して管理する方法について説明します。スナップショットの保存についての情報も示します。

この章は、次の節で構成されます。

ZFS スナップショットの概要

「スナップショット」とは、ファイルシステムまたはボリュームの読み取り専用コピーのことです。スナップショットはほとんど瞬間的に作成することができ、最初はプール内で追加のディスク領域を消費しません。ただし、アクティブなデータセット内のデータが変化していくにつれて、ディスク領域が消費されるようになります。古いデータを引き続き参照し、そのディスク領域を解放しないためです。

ZFS スナップショットには次の特長があります。

- システムの再起動後も残ります。
- スナップショットの理論上の最大数は、2^{64} です。
- スナップショットは個別のバッキングストアを使用しません。スナップショットは、作成元のファイルシステムまたはボリュームと同じストレージプールのディスク領域を直接使用します。

- 再帰的なスナップショットは、1 つの原子動作としてすばやく作成されます。スナップショットは、まとめて (一度にすべて) 作成されるか、まったく作成されないかのどちらかです。原子スナップショット動作の利点は、子孫ファイルシステムにまたがる場合でも、常にある一貫した時間のスナップショットデータが取得されることです。

ボリュームのスナップショットに直接アクセスすることはできませんが、それらの複製、バックアップ、ロールバックなどを行うことはできます。ZFS スナップショットのバックアップの詳細については、234 ページの「ZFS データを送信および受信する」を参照してください。

- 226 ページの「ZFS スナップショットを作成および破棄する」
- 229 ページの「ZFS スナップショットを表示してアクセスする」
- 231 ページの「ZFS スナップショットにロールバックする」

ZFS スナップショットを作成および破棄する

スナップショットは、`zfs snapshot` コマンドを使って作成します。引数として、作成するスナップショットの名前だけを指定できます。スナップショット名は次のように指定します。

filesystem@snapname
volume@snapname

スナップショット名は、51 ページの「ZFS コンポーネントに名前を付けるときの規則」の命名要件に従って付ける必要があります。

次の例では、`tank/home/ahrens` のスナップショットが `friday` という名前で作成されます。

```
# zfs snapshot tank/home/ahrens@friday
```

すべての子孫ファイルシステムのスナップショットを作成するには、`-r` オプションを使用します。次に例を示します。

```
# zfs snapshot -r tank/home@now
# zfs list -t snapshot
NAME                        USED  AVAIL  REFER  MOUNTPOINT
rpool/ROOT/zfs2BE@zfs2BE   78.3M      -  4.53G  -
tank/home@now                 0      -    26K  -
tank/home/ahrens@now          0      -   259M  -
tank/home/anne@now            0      -   156M  -
tank/home/bob@now             0      -   156M  -
tank/home/cindys@now          0      -   104M  -
```

スナップショットには、変更できるプロパティーはありません。また、データセットのプロパティーをスナップショットに適用することもできません。次に例を示します。

```
# zfs set compression=on tank/home/ahrens@now
cannot set compression property for 'tank/home/ahrens@now': snapshot
properties cannot be modified
```

スナップショットを破棄するには、`zfs destroy` コマンドを使用します。次に例を示します。

```
# zfs destroy tank/home/ahrens@now
```

データセットのスナップショットが存在する場合、そのデータセットを破棄することはできません。次に例を示します。

```
# zfs destroy tank/home/ahrens
cannot destroy 'tank/home/ahrens': filesystem has children
use '-r' to destroy the following datasets:
tank/home/ahrens@tuesday
tank/home/ahrens@wednesday
tank/home/ahrens@thursday
```

また、スナップショットからクローンが作成されている場合は、スナップショットを破棄する前にクローンを破棄する必要があります。

`destroy` サブコマンドの詳細については、187 ページの「ZFS ファイルシステムを破棄する」を参照してください。

ZFS スナップショットの保持

異なる自動スナップショットポリシーを実装しており、送信側にもう存在しないという理由で古いスナップショットが `zfs receive` によって意図せず破棄されてしまう場合、スナップショット保持機能の使用を検討することができます。

スナップショットを「保持」すると、そのスナップショットは破棄されなくなります。また、この機能と `zfs destroy -d` コマンドを使用することにより、最後のクローンの消去を保留しながら、クローンが存在するスナップショットを削除できます。個々のスナップショットには、初期値が 0 のユーザー参照カウントが関連付けられます。このカウントは、スナップショットの保持を設定するたびに 1 増加し、保持を解除するたびに 1 減少します。

以前の Solaris リリースでは、スナップショットを破棄するには、スナップショットにクローンが存在しない状態で `zfs destroy` コマンドを使用する必要がありました。この Solaris リリースでは、さらにスナップショットのユーザー参照カウントが 0 である必要があります。

1 つのスナップショットまたはスナップショットの集合を保持できます。たとえば次の構文は、保持タグ `keep` を `tank/home/cindys/snap@1` に付与します。

```
# zfs hold keep tank/home/cindys@snap1
```

`-r` オプションを使用すると、すべての子孫ファイルシステムのスナップショットを再帰的に保持できます。次に例を示します。

```
# zfs snapshot -r tank/home@now
# zfs hold -r keep tank/home@now
```

この構文は、単一の参照 keep を特定のスナップショットまたはスナップショットの集合に追加します。個々のスナップショットには独自のタグ名前空間があり、その空間内で保持タグが一意である必要があります。スナップショットに保持が設定されている場合、保持されたそのスナップショットを zfs destroy コマンドを使って破棄しようとしても失敗します。次に例を示します。

```
# zfs destroy tank/home/cindys@snap1
cannot destroy 'tank/home/cindys@snap1': dataset is busy
```

保持されたスナップショットを破棄する場合は、-d オプションを使用します。次に例を示します。

```
# zfs destroy -d tank/home/cindys@snap1
```

保持されたスナップショットの一覧を表示するには、zfs holds コマンドを使用します。次に例を示します。

```
# zfs holds tank/home@now
NAME           TAG   TIMESTAMP
tank/home@now  keep  Thu Jul 15 11:25:39 2010

# zfs holds -r tank/home@now
NAME                   TAG   TIMESTAMP
tank/home/cindys@now  keep  Thu Jul 15 11:25:39 2010
tank/home/mark@now    keep  Thu Jul 15 11:25:39 2010
tank/home@now         keep  Thu Jul 15 11:25:39 2010
```

zfs release コマンドを使用すると、保持されたスナップショットまたはスナップショットの集合を解放することができます。次に例を示します。

```
# zfs release -r keep tank/home@now
```

スナップショットが解放されたら、zfs destroy コマンドを使用してスナップショットを破棄できます。次に例を示します。

```
# zfs destroy -r tank/home@now
```

スナップショットの保持情報を示す 2 つの新しいプロパティーがあります。

- zfs destroy -d コマンドを使ってスナップショットの遅延破棄が予約されている場合、defer_destroy プロパティーがオンになります。それ以外の場合、このプロパティーはオフです。
- userrefs プロパティーの値は、このスナップショットに設定されている保持の数に設定されます。この数のことをユーザー参照カウントとも呼びます。

ZFS スナップショットの名前を変更する

スナップショットの名前を変更することはできますが、名前を変更するときはそれらが作成された同じプールとデータセットの中で行う必要があります。次に例を示します。

```
# zfs rename tank/home/cindys@083006 tank/home/cindys@today
```

また、次のショートカット構文は前の構文と同等です。

```
# zfs rename tank/home/cindys@083006 today
```

次のようなスナップショット名の変更操作はサポートされていません。ターゲットのプールとファイルシステムの名前が、スナップショットの作成されたプールとファイルシステムと異なるためです。

```
# zfs rename tank/home/cindys@today pool/home/cindys@saturday
cannot rename to 'pool/home/cindys@today': snapshots must be part of same
dataset
```

zfs rename -r コマンドを使用すると、スナップショットの名前を再帰的に変更することができます。次に例を示します。

```
# zfs list
NAME                          USED  AVAIL  REFER  MOUNTPOINT
users                         270K  16.5G    22K  /users
users/home                     76K  16.5G    22K  /users/home
users/home@yesterday            0      -    22K  -
users/home/markm               18K  16.5G    18K  /users/home/markm
users/home/markm@yesterday      0      -    18K  -
users/home/marks               18K  16.5G    18K  /users/home/marks
users/home/marks@yesterday      0      -    18K  -
users/home/neil                18K  16.5G    18K  /users/home/neil
users/home/neil@yesterday       0      -    18K  -
# zfs rename -r users/home@yesterday @2daysago
# zfs list -r users/home
NAME                         USED  AVAIL  REFER  MOUNTPOINT
users/home                    76K  16.5G    22K  /users/home
users/home@2daysago            0      -    22K  -
users/home/markm              18K  16.5G    18K  /users/home/markm
users/home/markm@2daysago      0      -    18K  -
users/home/marks              18K  16.5G    18K  /users/home/marks
users/home/marks@2daysago      0      -    18K  -
users/home/neil               18K  16.5G    18K  /users/home/neil
users/home/neil@2daysago       0      -    18K  -
```

ZFS スナップショットを表示してアクセスする

listsnapshots プールプロパティーを使用すれば、zfs list 出力でのスナップショット表示を有効または無効にすることができます。このプロパティーは、デフォルトで有効になっています。

このプロパティーを無効にした場合、zfs list -t snapshot コマンドを使用すればスナップショット情報を表示できます。あるいは、listsnapshots プールプロパティーを有効にします。次に例を示します。

```
# zpool get listsnapshots tank
NAME  PROPERTY       VALUE      SOURCE
tank  listsnapshots  on        default
# zpool set listsnapshots=off tank
# zpool get listsnapshots tank
NAME  PROPERTY       VALUE      SOURCE
tank  listsnapshots  off         local
```

ファイルシステムのスナップショットには、ルートの .zfs/snapshot ディレクトリからアクセスできます。たとえば、tank/home/ahrens が /home/ahrens にマウントされている場合は、tank/home/ahrens@thursday スナップショットのデータには、/home/ahrens/.zfs/snapshot/thursday ディレクトリからアクセスできます。

```
# ls /tank/home/ahrens/.zfs/snapshot
tuesday wednesday thursday
```

スナップショットの一覧は次の方法で表示できます。

```
# zfs list -t snapshot
NAME                         USED  AVAIL  REFER  MOUNTPOINT
pool/home/anne@monday          0      -   780K  -
pool/home/bob@monday           0      -  1.01M  -
tank/home/ahrens@tuesday    8.50K     -   780K  -
tank/home/ahrens@wednesday  8.50K     -  1.01M  -
tank/home/ahrens@thursday      0      -  1.77M  -
tank/home/cindys@today      8.50K     -   524K  -
```

特定のファイルシステムのために作成したスナップショットの一覧は、次の方法で表示できます。

```
# zfs list -r -t snapshot -o name,creation tank/home
NAME                  CREATION
tank/home@now         Wed Jun 30 16:16 2010
tank/home/ahrens@now  Wed Jun 30 16:16 2010
tank/home/anne@now    Wed Jun 30 16:16 2010
tank/home/bob@now     Wed Jun 30 16:16 2010
tank/home/cindys@now  Wed Jun 30 16:16 2010
```

ZFS スナップショットのディスク領域の計上

スナップショットを作成したときは、そのディスク領域は最初はスナップショットとファイルシステムの間で共有されます。それまでに作成したスナップショットと領域が共有されることもあります。ファイルシステムが変化していくにつれて、それまで共有されていたディスク領域がスナップショット固有になり、スナップショットの used プロパティーに計上されます。また、スナップショットを削除すると、ほかのスナップショットに固有の (および使用される) ディスク容量を増やすことができます。

スナップショット領域の referenced プロパティーの値は、スナップショットを作成したときのファイルシステムのプロパティーと同じです。

used プロパティーの値がどのように消費されているかについて、さらに詳細な情報を確認することができます。新しい読み取り専用ファイルシステムプロパティーは、クローン、ファイルシステム、およびボリュームに関するディスク領域使用状況を示します。次に例を示します。

```
$ zfs list -o space
# zfs list -ro space tank/home
NAME                  AVAIL   USED  USEDSNAP  USEDDS  USEDREFRESERV  USEDCHILD
tank/home             66.3G   675M         0     26K              0       675M
tank/home@now             -      0         -       -              -          -
tank/home/ahrens      66.3G   259M         0    259M              0          0
tank/home/ahrens@now      -      0         -       -              -          -
tank/home/anne        66.3G   156M         0    156M              0          0
tank/home/anne@now        -      0         -       -              -          -
tank/home/bob         66.3G   156M         0    156M              0          0
tank/home/bob@now         -      0         -       -              -          -
tank/home/cindys      66.3G   104M         0    104M              0          0
tank/home/cindys@now      -      0         -       -              -          -
```

これらのプロパティーについては、表 6–1 を参照してください。

ZFS スナップショットにロールバックする

zfs rollback コマンドを使用すると、特定のスナップショットが作成された時点よりもあとにファイルシステムに対して行われたすべての変更を破棄できます。ファイルシステムは、そのスナップショットが作成されたときの状態に戻ります。デフォルトでは、このコマンドを使って、最新のスナップショット以外のスナップショットにロールバックすることはできません。

それより前のスナップショットにロールバックするには、中間にあるスナップショットをすべて破棄する必要があります。-r オプションを指定すれば、古いスナップショットを破棄できます。

中間にあるスナップショットのクローンが存在する場合は、-R オプションを指定してクローンも破棄する必要があります。

注– ロールバックするファイルシステムが現在マウントされている場合は、そのマウントが解除されてから再度マウントされます。ファイルシステムのマウントを解除できない場合は、ロールバックに失敗します。必要に応じて -f オプションを指定すると、ファイルシステムのマウントが強制的に解除されます。

次の例では、tank/home/ahrens ファイルシステムが tuesday スナップショットにロールバックされます。

```
# zfs rollback tank/home/ahrens@tuesday
cannot rollback to 'tank/home/ahrens@tuesday': more recent snapshots exist
use '-r' to force deletion of the following snapshots:
tank/home/ahrens@wednesday
tank/home/ahrens@thursday
# zfs rollback -r tank/home/ahrens@tuesday
```

この例では、スナップショット wednesday および thursday が破棄されます。これらよりも古いスナップショット tuesday にロールバックされるためです。

```
# zfs list -r -t snapshot -o name,creation tank/home/ahrens
NAME                  CREATION
tank/home/ahrens@now  Wed Jun 30 16:16 2010
```

ZFS クローンの概要

「クローン」とは、書き込み可能なボリュームまたはファイルシステムのことで、最初の内容は作成元のデータセットと同じです。スナップショットの場合と同様に、クローンは瞬間的に作成され、最初は追加のディスク領域を消費しません。また、クローンのスナップショットを作成することもできます。

クローンは、スナップショットだけから作成できます。スナップショットが複製されるときに、クローンとスナップショットの間に暗黙の依存関係が作成されます。クローンはデータセット階層内の別の場所に作成されますが、クローンが存在する間は元のスナップショットを破棄することはできません。この依存関係は、origin プロパティーからわかります。そのような依存関係が存在する場合には、zfs destroy コマンドを実行すると表示されます。

クローンには、作成元のデータセットのプロパティーは継承されません。zfs get および zfs set コマンドを使用して、複製したデータセットのプロパティーを表示して変更することができます。ZFS データセットのプロパティーの設定方法の詳細については、206 ページの「ZFS プロパティーを設定する」を参照してください。

クローンのすべてのディスク領域は最初は元のスナップショットと共有されるため、used プロパティーの初期値はゼロになります。クローンに変更が加えられるにつれて、使用されるディスク領域が多くなります。元のスナップショットの used プロパティーには、クローンが消費するディスク領域は含まれません。

- 232 ページの「ZFS クローンを作成する」
- 233 ページの「ZFS クローンを破棄する」
- 233 ページの「ZFS ファイルシステムを ZFS クローンで置き換える」

ZFS クローンを作成する

クローンを作成するには、zfs clone コマンドを使用します。クローンをどのスナップショットから作成するかを指定し、新しいファイルシステムまたはボリュームの名前を指定します。新しいファイルシステムまたはボリュームは、ZFS 階

層内の任意の場所に配置できます。新しいデータセットは、クローンの作成元になったスナップショットと同じ種類 (ファイルシステムやボリュームなど) です。クローンを作成するためのファイルシステムは、基にするファイルシステムスナップショットがあるプールに存在している必要があります。

次の例では、tank/home/ahrens/bug123 という名前の新しいクローンが作成されます。最初の内容は、スナップショット tank/ws/gate@yesterday と同じです。

```
# zfs snapshot tank/ws/gate@yesterday
# zfs clone tank/ws/gate@yesterday tank/home/ahrens/bug123
```

次の例では、スナップショット projects/newproject@today から複製されたワークスペースが、一時的なユーザーのために projects/teamA/tempuser という名前で作成されます。次に、複製されたワークスペースにプロパティーが設定されます。

```
# zfs snapshot projects/newproject@today
# zfs clone projects/newproject@today projects/teamA/tempuser
# zfs set sharenfs=on projects/teamA/tempuser
# zfs set quota=5G projects/teamA/tempuser
```

ZFS クローンを破棄する

ZFS クローンを破棄するには、zfs destroy コマンドを使用します。次に例を示します。

```
# zfs destroy tank/home/ahrens/bug123
```

親のスナップショットを破棄するときは、その前にクローンを破棄する必要があります。

ZFS ファイルシステムを ZFS クローンで置き換える

zfs promote コマンドを使えば、アクティブな ZFS ファイルシステムをそのファイルシステムのクローンで置き換えることができます。この機能を使ってファイルシステムの複製と置換を実行でき、「作成元」のファイルシステムが、指定されたファイルシステムのクローンになります。さらに、この機能を使えば、クローンの作成元となるファイルシステムを破棄することもできます。クローンの移行促進を行わない限り、アクティブクローンの元のファイルシステムを破棄することはできません。クローンの破棄に関する詳細については、233 ページの「ZFS クローンを破棄する」を参照してください。

次の例では、tank/test/productA ファイルシステムがクローンされたあと、クローンファイルシステム tank/test/productAbeta が元の tank/test/productA ファイルシステムになっています。

```
# zfs create tank/test
# zfs create tank/test/productA
# zfs snapshot tank/test/productA@today
# zfs clone tank/test/productA@today tank/test/productAbeta
# zfs list -r tank/test
NAME                        USED  AVAIL  REFER  MOUNTPOINT
tank/test                   104M  66.2G    23K  /tank/test
tank/test/productA          104M  66.2G   104M  /tank/test/productA
tank/test/productA@today       0      -   104M  -
tank/test/productAbeta         0  66.2G   104M  /tank/test/productAbeta
# zfs promote tank/test/productAbeta
# zfs list -r tank/test
NAME                           USED  AVAIL  REFER  MOUNTPOINT
tank/test                      104M  66.2G    24K  /tank/test
tank/test/productA                0  66.2G   104M  /tank/test/productA
tank/test/productAbeta         104M  66.2G   104M  /tank/test/productAbeta
tank/test/productAbeta@today      0      -   104M  -
```

この zfs list の出力では、元の productA ファイルシステムのディスク領域計上情報が、productAbeta ファイルシステムのものに置き換わっています。

ファイルシステムの名前を変更することで、クローンの置換処理を完了することができます。次に例を示します。

```
# zfs rename tank/test/productA tank/test/productAlegacy
# zfs rename tank/test/productAbeta tank/test/productA
# zfs list -r tank/test
```

また、旧バージョンのファイルシステムを削除することもできます。次に例を示します。

```
# zfs destroy tank/test/productAlegacy
```

ZFS データを送信および受信する

zfs send コマンドを実行すると、スナップショットのストリーム表現が作成され、標準出力に書き込まれます。デフォルトでは、完全なストリームが生成されます。この出力は、ファイルまたは別のシステムにリダイレクトできます。zfs receive コマンドを実行すると、ストリームに内容が指定されているスナップショットが作成され、標準入力に渡されます。ストリーム全体を受信する場合、新しいファイルシステムも作成されます。これらのコマンドを使えば、ZFS スナップショットデータを送信したり、ZFS スナップショットデータやファイルシステムを受信したりできます。次の節の例を参照してください。

- 236 ページの「ZFS スナップショットを送信する」
- 237 ページの「ZFS スナップショットを受信する」
- 238 ページの「複雑な ZFS スナップショットストリームを送信および受信する」
- 235 ページの「ほかのバックアップ製品を使用して ZFS データを保存する」

ZFS データを保存するために、次のバックアップ方法が用意されています。

- 企業向けバックアップ製品 – 次の機能が必要な場合は、企業向けバックアップソリューションを検討してください。
  - ファイルごとの復元
  - バックアップメディアの検証
  - メディアの管理
- ファイルシステムのスナップショットとスナップショットのロールバック – ファイルシステムのコピーを作成して、必要に応じて以前のバージョンのファイルシステムに戻す作業を簡単に実行するには、`zfs snapshot` および `zfs rollback` コマンドを使用します。たとえば、以前のバージョンのファイルシステムからファイルを復元するために、この方法を使用できます。

  スナップショットの作成およびロールバックの詳細については、225 ページの「ZFS スナップショットの概要」を参照してください。
- スナップショットの保存 – `zfs send` および `zfs receive` コマンドを使用して、ZFS スナップショットの送信と受信を行います。スナップショットから次のスナップショットまでの増分変更を保存することができますが、ファイルを個別に復元することはできません。ファイルシステムのスナップショット全体を復元する必要があります。これらのコマンドでは、ZFS データを保存するための完全なバックアップソリューションは提供されません。
- リモート複製 – あるシステムのファイルシステムを別のシステムにコピーするには、`zfs send` および `zfs receive` コマンドを使用します。この処理は、WAN 経由でデバイスをミラー化する従来のボリューム管理製品とは異なります。特殊な設定やハードウェアは必要ありません。ZFS ファイルシステムを複製する利点は、ファイルシステムを別のシステムのストレージプール上に再作成し、その新しく作成したプールに同じファイルシステムデータを格納しながら RAID-Z などの別の構成レベルを指定できることです。
- アーカイブユーティリティー – `tar`、`cpio`、`pax`、サードパーティーバックアップ製品などのアーカイブユーティリティーを使って ZFS データを保存します。現時点では、`tar` と `cpio` では NFSv4 方式の ACL を正しく変換できますが、`pax` では変換できません。

ほかのバックアップ製品を使用して ZFS データを保存する

`zfs send` および `zfs receive` コマンド以外に、`tar` や `cpio` コマンドなどのアーカイブユーティリティーを使用して、ZFS ファイルを保存することもできます。これらのユーティリティーは、ZFS ファイル属性と ACL を保存して復元します。`tar` コマンドと `cpio` コマンドの適切なオプションを確認してください。

ZFS およびサードパーティーバックアップ製品に関する問題の最新情報については、『Solaris 10 ご使用にあたって』または次のサイトの ZFS FAQ を参照してください。

http://hub.opensolaris.org/bin/view/Community+Group+zfs/faq/#backupsoftware

ZFS スナップショットを送信する

zfs send コマンドを使用して、スナップショットストリームのコピーを送信し、バックアップデータの格納に使用する別のプール (同じシステム上または別のシステム上にある) でそのスナップショットストリームを受信することができます。たとえば、別のプール上のスナップショットストリームを同じシステムに送信するには、次のような構文を使用します。

```
# zfs send tank/data@snap1 | zfs recv spool/ds01
```

zfs receive コマンドの別名として、zfs recv を 使用できます。

スナップショットストリームを別のシステムに送信する場合は、zfs send の出力を ssh コマンドにパイプします。次に例を示します。

```
host1# zfs send tank/dana@snap1 | ssh host2 zfs recv newtank/dana
```

完全なストリームを送信するときは、対象のファイルシステムが存在してはいけません。

zfs send -i オプションを使用すれば、増分データを送信できます。次に例を示します。

```
host1# zfs send -i tank/dana@snap1 tank/dana@snap2 | ssh host2 zfs recv newtank/dana
```

最初の引数 (snap1) には、以前のスナップショットを指定します。2 番目の引数 (snap2) には、それよりあとのスナップショットを指定します。この場合は、増分データの受信を正常に行うために newtank/dana ファイルシステムがあらかじめ存在している必要があります。

増分ソース *snap1* は、スナップショット名の最後の構成要素だけで指定できます。このショートカットは、*snap1* には @ 記号のあとの名前を指定するだけでよいことを意味し、この場合 snap1 は *snap2* と同じファイルシステムから作成されたものと見なされます。次に例を示します。

```
host1# zfs send -i snap1 tank/dana@snap2 > ssh host2 zfs recv newtank/dana
```

このショートカット構文は、前の例の増分構文と同等です。

異なるファイルシステム snapshot1 から増分ストリームを生成しようとすると、次のメッセージが表示されます。

```
cannot send 'pool/fs@name': not an earlier snapshot from the same fs
```

多数のコピーを保管する必要がある場合は、gzip コマンドを使って ZFS スナップショットのストリーム表現を圧縮することを検討してください。次に例を示します。

```
# zfs send pool/fs@snap | gzip > backupfile.gz
```

ZFS スナップショットを受信する

ファイルシステムのスナップショットを受信するときは、次の重要な点に留意してください。

- スナップショットとファイルシステムの両方が受信されます。
- ファイルシステムとその子孫のすべてのファイルシステムがマウント解除されます。
- ファイルシステムが受信されている間は、それらにアクセスできません。
- 受信される元のファイルシステムは、その転送中に存在してはいけません。
- ファイルシステム名がすでに存在する場合は、zfs rename コマンドを使ってファイルシステムの名前を変更できます。

次に例を示します。

```
# zfs send tank/gozer@0830 > /bkups/gozer.083006
# zfs receive tank/gozer2@today < /bkups/gozer.083006
# zfs rename tank/gozer tank/gozer.old
# zfs rename tank/gozer2 tank/gozer
```

対象のファイルシステムに変更を加え、新たに増分スナップショットを送信する場合は、まず受信側のファイルシステムをロールバックする必要があります。

次のような例を考えます。まず、次のようにファイルシステムに変更を加えます。

```
host2# rm newtank/dana/file.1
```

次に、tank/dana@snap3 の増分を送信します。ただし、新しい増分スナップショットを受信するには、まず受信側のファイルシステムをロールバックする必要があります。または、-F オプションを使用すれば、ロールバック手順を実行する必要がなくなります。次に例を示します。

```
host1# zfs send -i tank/dana@snap2 tank/dana@snap3 | ssh host2 zfs recv -F newtank/dana
```

増分スナップショットを受信するときは、対象のファイルシステムが存在している必要があります。

ファイルシステムに変更を加えたあとで、新しい増分スナップショットを受信するために受信側のファイルシステムのロールバックを行わない場合、または -F オプションを使用しない場合は、次のようなメッセージが表示されます。

```
host1# zfs send -i tank/dana@snap4 tank/dana@snap5 | ssh host2 zfs recv newtank/dana
cannot receive: destination has been modified since most recent snapshot
```

-F オプションが正常に実行される前に、次の検査が行われます。

- 最新のスナップショットが増分ソースと一致しない場合は、ロールバックも受信も完了せず、エラーメッセージが返される。
- zfs receive コマンドで指定された増分ソースと一致しない異なるファイルシステムの名前を間違って指定した場合は、ロールバックも受信も完了せず、次のエラーメッセージが返される。

  ```
  cannot send 'pool/fs@name': not an earlier snapshot from the same fs
  ```

複雑な ZFS スナップショットストリームを送信および受信する

この節では、zfs send -I および -R オプションを使用して、より複雑なスナップショットストリームを送受信する方法について説明します。

複雑な ZFS スナップショットストリームを送受信するときは、次の点に留意してください。

- 1 つのスナップショットのすべての増分ストリームを累積スナップショットに送信する場合は、zfs send -I オプションを使用します。または、元のスナップショットからの増分ストリームを送信してクローンを作成する場合にも、このオプションを使用します。増分ストリームを受け入れるには、元のスナップショットが受信側にすでに存在している必要があります。
- すべての子孫ファイルシステムの複製ストリームを送信する場合は、zfs send -R オプションを使用します。複製ストリームの受信時には、すべてのプロパティー、スナップショット、下位ファイルシステム、およびクローンが維持されます。
- 増分複製ストリームを送信するには、両方のオプションを使用します。
  - プロパティーの変更は保持され、スナップショットおよびファイルシステムの rename 操作と destroy 操作も保持されます。
  - 複製ストリームの受信時に zfs recv -F が指定されていない場合、データセットの destroy 操作は無視されます。この場合の zfs recv -F 構文は、「必要に応じてロールバックする」という意味も持っています。
  - (zfs send -R ではない) ほかの -i または -I の場合と同様に、-I を使用すると、snapA から snapD までのすべてのスナップショットが送信されます。-i を使用すると、(すべての子孫の) snapD だけが送信されます。
- このような新しい種類の zfs send ストリームを受信するには、そのストリームを送信できるソフトウェアバージョンが受信側のシステムで稼働している必要があります。ストリームのバージョンは 1 増やされています。

ただし、新しいソフトウェアバージョンを使用して古いプールバージョンのストリームにアクセスすることはできます。たとえば、新しいオプションで作成されたストリームを、バージョン 3 プールに対して送受信することができます。ただし、新しいオプションで送信されたストリームを受信するには、最近のソフトウェアが稼働している必要があります。

例 7–1 複雑な ZFS スナップショットストリームを送信および受信する

zfs send -I オプションを使用すると、一連の増分スナップショットを結合して 1 つのスナップショットを作成できます。次に例を示します。

```
# zfs send -I pool/fs@snapA pool/fs@snapD > /snaps/fs@all-I
```

次に、snapB、snapC、および snapD を削除します。

```
# zfs destroy pool/fs@snapB
# zfs destroy pool/fs@snapC
# zfs destroy pool/fs@snapD
```

結合されたスナップショットを受信するには、次のコマンドを使用します。

```
# zfs receive -d -F pool/fs < /snaps/fs@all-I
# zfs list
NAME                     USED  AVAIL  REFER  MOUNTPOINT
pool                     428K  16.5G    20K  /pool
pool/fs                   71K  16.5G    21K  /pool/fs
pool/fs@snapA             16K      -  18.5K  -
pool/fs@snapB             17K      -    20K  -
pool/fs@snapC             17K      -  20.5K  -
pool/fs@snapD               0      -    21K  -
```

zfs send -I コマンドを使用すると、スナップショットとクローンスナップショットを結合して、結合されたデータセットを作成することもできます。次に例を示します。

```
# zfs create pool/fs
# zfs snapshot pool/fs@snap1
# zfs clone pool/fs@snap1 pool/clone
# zfs snapshot pool/clone@snapA
# zfs send -I pool/fs@snap1 pool/clone@snapA > /snaps/fsclonesnap-I
# zfs destroy pool/clone@snapA
# zfs destroy pool/clone
# zfs receive -F pool/clone < /snaps/fsclonesnap-I
```

zfs send -R コマンドを使用すると、ZFS ファイルシステムおよび指定されたスナップショットまでのすべての子孫ファイルシステムを複製できます。このストリームの受信時には、すべてのプロパティー、スナップショット、子孫ファイルシステム、およびクローンが維持されます。

例 7–1 複雑な ZFS スナップショットストリームを送信および受信する (続き)

次の例では、ユーザーのファイルシステムのスナップショットが作成されます。すべてのユーザースナップショットから 1 つの複製ストリームが作成されます。次に、元のファイルシステムおよびスナップショットが破棄されてから回復されます。

```
# zfs snapshot -r users@today
# zfs list
NAME                 USED  AVAIL  REFER  MOUNTPOINT
users                187K  33.2G    22K  /users
users@today             0      -    22K  -
users/user1           18K  33.2G    18K  /users/user1
users/user1@today       0      -    18K  -
users/user2           18K  33.2G    18K  /users/user2
users/user2@today       0      -    18K  -
users/user3           18K  33.2G    18K  /users/user3
users/user3@today       0      -    18K  -
# zfs send -R users@today > /snaps/users-R
# zfs destroy -r users
# zfs receive -F -d users < /snaps/users-R
# zfs list
NAME                 USED  AVAIL  REFER  MOUNTPOINT
users                196K  33.2G    22K  /users
users@today             0      -    22K  -
users/user1           18K  33.2G    18K  /users/user1
users/user1@today       0      -    18K  -
users/user2           18K  33.2G    18K  /users/user2
users/user2@today       0      -    18K  -
users/user3           18K  33.2G    18K  /users/user3
users/user3@today       0      -    18K  -
```

次の例では、`zfs send -R` コマンドを使用して、`users` データセットとその子孫を複製し、複製したストリームを別のプール `users2` に送信します。

```
# zfs create users2 mirror c0t1d0 c1t1d0
# zfs receive -F -d users2 < /snaps/users-R
# zfs list
NAME                 USED  AVAIL  REFER  MOUNTPOINT
users                224K  33.2G    22K  /users
users@today             0      -    22K  -
users/user1           33K  33.2G    18K  /users/user1
users/user1@today     15K      -    18K  -
users/user2           18K  33.2G    18K  /users/user2
users/user2@today       0      -    18K  -
users/user3           18K  33.2G    18K  /users/user3
users/user3@today       0      -    18K  -
users2                188K  16.5G    22K  /users2
users2@today            0      -    22K  -
users2/user1          18K  16.5G    18K  /users2/user1
users2/user1@today      0      -    18K  -
users2/user2          18K  16.5G    18K  /users2/user2
users2/user2@today      0      -    18K  -
users2/user3          18K  16.5G    18K  /users2/user3
users2/user3@today      0      -    18K  -
```

ZFS データのリモート複製

zfs send および zfs recv コマンドを使用して、あるシステムのスナップショットのストリーム表現を別のシステムに離れた場所からコピーできます。次に例を示します。

```
# zfs send tank/cindy@today | ssh newsys zfs recv sandbox/restfs@today
```

このコマンドは、tank/cindy@today スナップショットのデータを送信し、そのデータを sandbox/restfs ファイルシステムに受信します。このコマンドは、restfs@today スナップショットを newsys システム上にも作成します。この例のユーザーは、リモートシステム上で ssh を使用するように設定されています。

◆ ◆ ◆ 第 8 章

ACL による Oracle Solaris ZFS ファイルの保護

この章では、アクセス制御リスト (ACL) を使用して ZFS ファイルを保護する方法について説明します。ACL は、標準の UNIX アクセス権よりも詳細なアクセス権を提供します。

この章は、次の節で構成されます。

新しい Solaris ACL モデル

以前のバージョンの Solaris OS では、主に POSIX ドラフト ACL 仕様に基づく ACL 実装がサポートされていました。POSIX ドラフトに基づく ACL は、UFS ファイルを保護するために使用され、NFSv4 より前のバージョンの NFS によって変換されます。

NFSv4 を導入したことにより、NFSv4 が提供する UNIX クライアントと UNIX 以外のクライアントとの間の相互運用性を、新しい ACL モデルを使って完全にサポートできるようになりました。NFSv4 仕様で定義されている新しい ACL 実装は、NT 方式の ACL を使用して、より豊かなセマンティクスを実現します。

新しい ACL モデルの主な相違点は、次のとおりです。

- 新しい ACL モデルは NFSv4 仕様に基づいており、NT 形式の ACL に似ています。
- 新しいモデルは、より詳細なアクセス権を提供します。詳細は、表 8–2 を参照してください。
- ACL は、`setfacl` や `getfacl` コマンドではなく、`chmod` および `ls` コマンドを使用して設定および表示します。

- 新しいモデルは、ディレクトリからサブディレクトリへとアクセス特権が適用されていく方法に対して、より豊富な継承セマンティクスを提供します。詳細については、248 ページの「ACL 継承」を参照してください。

どちらの ACL モデルを使った場合でも、標準のファイルアクセス権の場合より詳細にアクセス権を制御できます。新しい ACL は、POSIX ドラフト ACL と同様に、複数のアクセス制御エントリ (ACE) で構成されています。

POSIX ドラフトに基づく ACL では、1 つのエントリを使用して、許可するアクセス権と拒否するアクセス権を定義します。新しい ACL モデルでは、アクセス権を検査するために、2 種類の ACE が利用されます。 `ALLOW` と `DENY` です。つまり、どちらの ACE にもアクセス権が定義されていますが、その ACE に定義されていないアクセス権については、その ACE だけを使ってアクセス権を許可または拒否するかを推論することはできません。

NFSv4 ACL と POSIX ドラフト ACLとの 間の変換は、次のように行われます。

- ACL に対応するユーティリティー (`cp`、`mv`、`tar`、`cpio`、`rcp` コマンドなど) を使用している場合は、ACL が含まれる UFS ファイルを ZFS ファイルシステムに転送するときに、POSIX ドラフト ACL が同等の NFSv4 ACL に変換されます。
- 一部の NFSv4 ACL は、POSIX ドラフト ACL に変換されます。NFSv4 ACL が POSIX ドラフト ACL に変換されない場合は、次のようなメッセージが表示されます。

```
# cp -p filea /var/tmp
cp: failed to set acl entries on /var/tmp/filea
```

- 最新の Solaris リリースを実行するシステム上で ACL の保持オプション (`tar -p` または `cpio -P`) を使って UFS `tar` または `cpio` アーカイブを作成した場合でも、以前の Solaris リリースを実行するシステム上でそのアーカイブを展開したときは、ACL が失われます。

  すべてのファイルが正しいファイルモードで展開されますが、ACL エントリは無視されます。
- `ufsrestore` コマンドを使って ZFS ファイルシステムにデータを復元することができます。元のデータに POSIX ドラフト ACL が含まれている場合、それらは NFSv4 ACL に変換されます。
- UFS ファイルに NFSv4 ACL を設定しようとすると、次のようなメッセージが表示されます。

```
chmod: ERROR: ACL type's are different
```

- ZFS ファイルに POSIX ドラフト ACL を設定しようとすると、次のようなメッセージが表示されます。

```
# getfacl filea
File system doesn't support aclent_t style ACL's.
See acl(5) for more information on Solaris ACL support.
```

ACL およびバックアップ製品に関するその他の制限については、235 ページの「ほかのバックアップ製品を使用して ZFS データを保存する」を参照してください。

ACL を設定する構文の説明

2 つの基本 ACL 形式を次に示します。

簡易 **ACL** を設定する構文

この ACL は、従来の UNIX の `owner/group/other` エントリを表現しているだけという点で、「簡易な」 ACL です。

`chmod [options] A[index]{+|=}owner@ |group@ |everyone@:`
access-permissions`/...[:`*inheritance-flags*`]: deny | allow` *file*

`chmod [options] A-owner@, group@, everyone@:`*access-permissions*
`/...[:`*inheritance-flags*`]:deny | allow` *file ...*

`chmod [options] A[index]-` *file*

非簡易 **ACL** を設定する構文

`chmod [options] A[index]{+|=}user|group:name:`*access-permissions*
`/...[:`*inheritance-flags*`]:deny | allow` *file*

`chmod [options] A-user|group:name:`*access-permissions* `/...[:`*inheritance-flags*`]:deny | allow` *file ...*

`chmod [options] A[index]-` *file*

`owner@`, `group@`, `everyone@`
: 簡易 ACL 構文の *ACL-entry-type* を指定します。ACL エントリタイプについては、表 8–1 を参照してください。

user または *group*`:`*ACL-entry-ID*`=`*username* または *groupname*
: 明示的な ACL 構文の *ACL-entry-type* を指定します。ユーザーとグループの *ACL-entry-type* には、*ACL-entry-ID* と、*username* または *groupname* も含める必要があります。ACL エントリタイプについては、表 8–1 を参照してください。

access-permissions`/.../`
: 許可または拒否するアクセス権を指定します。ACL アクセス特権については、表 8–2 を参照してください。

inheritance-flags
: ACL 継承フラグのオプションリストを指定します。ACL 継承フラグについては、表 8–3 を参照してください。

`deny` | `allow`
: そのアクセス権を許可するかまたは拒否するかを指定します。

次の例では、*ACL-entry-ID* の値は関係ありません。

```
group@:write_data/append_data/execute:deny
```

次の例では、*ACL-entry-ID* が含まれています。特定のユーザー (*ACL-entry-type*) を ACL に含めるためです。

```
0:user:gozer:list_directory/read_data/execute:allow
```

ACL エントリが表示されるときは、次のようになります。

```
2:group@:write_data/append_data/execute:deny
```

この例の **2** は *index-ID* 指定と呼ばれますが、これにより、所有者、特定の UID、グループ、および全員用として複数のエントリを含む可能性のある比較的大きな ACL 内の ACL エントリが識別されます。`chmod` コマンドと一緒に *index-ID* を指定すれば、ACL のどの部分を変更するかを指定できます。たとえば次のように、`chmod` コマンドの構文で `A3` と指定して、インデックス ID 3 を特定することができます。

```
chmod A3=user:venkman:read_acl:allow filename
```

ACL エントリタイプは、所有者やグループなどの ACL 表現です。次の表の説明を参照してください。

表 **8–1** ACL エントリタイプ

ACL エントリタイプ	説明
`owner@`	オブジェクトの所有者に許可するアクセス権を指定します。
`group@`	オブジェクトを所有するグループに許可するアクセス権を指定します。
`everyone@`	ほかのどの ACL エントリにも一致しないすべてのユーザーまたはグループに許可するアクセス権を指定します。
`user`	ユーザー名を使って、オブジェクトに追加するユーザーに許可するアクセス権を指定します。このエントリには *ACL-entry-ID* を含める必要があります。*username* または *userID* を指定します。値が有効な数値 UID または *username* でない場合、その ACL エントリタイプは無効です。
`group`	グループ名を使って、オブジェクトに追加するグループに許可するアクセス権を指定します。このエントリには *ACL-entry-ID* を含める必要があります。*groupname* または *groupID* を指定します。値が有効な数値 GID または *groupname* でない場合、その ACL エントリタイプは無効です。

ACL アクセス特権について、次の表で説明します。

表 **8–2** ACL アクセス特権

アクセス特権	アクセス特権のコンパクト表現	説明
`add_file`	`w`	ディレクトリに新しいファイルを追加するためのアクセス権。

表 8–2 ACL アクセス特権 (続き)

アクセス特権	アクセス特権のコンパクト表現	説明
add_subdirectory	p	ディレクトリ上でサブディレクトリを作成するためのアクセス権。
append_data	p	プレースホルダー。現時点では実装されていません。
delete	d	ファイルを削除するためのアクセス権。
delete_child	D	ディレクトリ内のファイルまたはディレクトリを削除するためのアクセス権。
execute	x	ファイルを実行するためのアクセス権またはディレクトリの内容を検索するためのアクセス権。
list_directory	r	ディレクトリの内容を表示するためのアクセス権。
read_acl	c	ACL (ls) を読み取るためのアクセス権。
read_attributes	a	ファイルの基本属性 (ACL 以外) を読み取るためのアクセス権。基本属性は、stat レベルの属性と考えてください。このアクセスマスクビットを許可したエンティティーは、ls(1) および stat(2) を実行できる状態になります。
read_data	r	ファイルの内容を読み取るためのアクセス権。
read_xattr	R	ファイルの拡張属性を読み取るためのアクセス権。または、ファイルの拡張属性ディレクトリの検索を実行するためのアクセス権。
synchronize	s	プレースホルダー。現時点では実装されていません。
write_xattr	W	拡張属性を作成するためのアクセス権。または、拡張属性ディレクトリに書き込むためのアクセス権。 このアクセス権を許可したユーザーは、ファイルの拡張属性ディレクトリを作成できます。属性ファイルのアクセス権を使って、その属性にユーザーがアクセスできるかどうかを制御します。
write_data	w	ファイルの内容を変更または置き換えるためのアクセス権。
write_attributes	A	ファイルまたはディレクトリに関連付けられたタイムスタンプを任意の値に変更する権限。
write_acl	C	ACL の書き込みを行うアクセス権、または chmod コマンドを使って ACL を変更するアクセス権。

表 8–2 ACL アクセス特権 (続き)

アクセス特権	アクセス特権のコンパクト表現	説明
write_owner	o	ファイルの所有者またはグループを変更するためのアクセス権。つまり、ファイルに対して chown または chgrp コマンドを実行することができます。 ファイルの所有権を取得するためのアクセス権。または、ファイルのグループ所有権をユーザーが所属するグループに変更するためのアクセス権。ファイルまたはグループの所有権を任意のユーザーまたはグループに変更する場合は、PRIV_FILE_CHOWN 権限が必要です。

ACL 継承

ACL 継承を使用する目的は、親ディレクトリの既存のアクセス権を考慮しながら、意図した ACL を新しく作成するファイルまたはディレクトリが継承できるようにすることです。

デフォルトでは、ACL は伝達されません。ディレクトリに非簡易 ACL を設定した場合でも、その ACL はそれ以降に作成されるディレクトリには継承されません。ACL を継承する場合は、ファイルまたはディレクトリにそのことを 指定する必要があります。

オプションの継承フラグについて、次の表で説明します。

表 8–3 ACL 継承フラグ

継承フラグ	継承フラグのコンパクト表現	説明
file_inherit	f	親ディレクトリから ACL を継承しますが、適用対象はそのディレクトリのファイルのみとなります。
dir_inherit	d	親ディレクトリから ACL を継承しますが、適用対象はそのディレクトリのサブディレクトリのみとなります。
inherit_only	i	親ディレクトリから ACL を継承しますが、新しく作成したファイルまたはサブディレクトリにのみ適用され、そのディレクトリ自体には適用されません。このフラグを使用する場合は、何を継承するかを指定するために、file_inherit フラグまたは dir_inherit フラグ、あるいはその両方を指定する必要があります。

表 8–3 ACL 継承フラグ (続き)

継承フラグ	継承フラグのコンパクト表現	説明
no_propagate	n	親ディレクトリの ACL をそのディレクトリの第 1 レベルの内容にのみ継承します。第 2 レベル以降の内容には継承しません。このフラグを使用する場合は、何を継承するかを指定するために、file_inherit フラグまたは dir_inherit フラグ、あるいはその両方を指定する必要があります。
-	なし	アクセス権は付与されていません。

また、aclinherit ファイルシステムプロパティーを使用して、デフォルトの ACL 継承ポリシーをファイルシステムに設定することもできます。ポリシーの厳密度はプロパティーによって異なります。詳細については、次の節を参照してください。

ACL プロパティー

ZFS ファイルシステムには、ACL に関連するプロパティーが 2 つ用意されています。

- aclinherit – このプロパティーを使って、ACL 継承の動作を決定します。次の値を使用できます。
  - discard – 新しいオブジェクトの場合に、ファイルまたはディレクトリを作成するときに ACL エントリは継承されません。新しいファイルまたはディレクトリの ACL は、そのファイルまたはディレクトリのアクセス権と等価です。
  - noallow – 新しいオブジェクトの場合に、継承可能な ACL エントリのうち、アクセスタイプが deny のエントリだけが継承されます。
  - restricted – 新しいオブジェクトの場合に、ACL エントリが継承されるときに、write_owner および write_acl アクセス権が取り除かれます。
  - passthrough – プロパティーの値が passthrough に設定されている場合、作成されるファイルのアクセス権は継承可能な ACE によって決定されます。アクセス権に影響を与える継承可能な ACE が存在しない場合、アクセス権はアプリケーションから要求されたアクセス権に従って設定されます。
  - passthrough-x - このプロパティー値のセマンティクスは次の点を除き passthrough と同じです。passthrough-x を有効にした場合、ファイル作成モードおよびそのモードに影響を与える継承可能な ACE で実行権が設定されている場合に限り、ファイルが実行 (x) 権付きで作成されます。

  aclinherit プロパティーのデフォルト値は、restricted です。
- aclmode – このプロパティーを指定すると、ファイルが最初に作成されるとき、または chmod コマンドを使ってファイルまたはディレクトリのアクセス権が変更されるときに、ACL の動作が変更されます。次の値を使用できます。

- discard – ファイルまたはディレクトリのモードを定義するために必要なエントリを除いて、すべての ACL エントリが取り除かれます。
- groupmask – ユーザーまたはグループの ACL アクセス権の強度が、グループのアクセス権より弱くなるように変更されます。ただし、そのファイルまたはディレクトリの所有者と同じ UID を持つユーザーエントリは変更されません。さらに、その ACL アクセス権の強度が、所有者のアクセス権より弱くなるように変更されます。
- passthrough – owner@、group@、または everyone@ 以外の ACE は、chmod 操作時に変更されることはありません。owner@、group@、または everyone@ の ACE は無効になり、chmod 操作で要求されたファイルモードが設定されます。

aclmode プロパティーのデフォルト値は、groupmask です。

ZFS ファイルに ACL を設定する

ZFS と一緒に実装される ACL は、ACL エントリで構成されます。ZFS の ACL モデルは「純粋」です。つまり、すべてのファイルに ACL が含まれます。この ACL は、従来の UNIX の owner/group/other エントリを表現しているだけという点で、全体的に見れば簡易な ACLです。

ファイルのアクセス権を変更した場合には、それに応じてファイルの ACL が更新されます。また、ファイルまたはディレクトリへのアクセスをユーザーに許可するための非簡易 ACL を削除しても、グループまたは全員にアクセスを許可するファイルまたはディレクトリのアクセス権ビットが設定されている場合には、そのユーザーはそのファイルまたはディレクトリに引き続きアクセスできます。アクセス制御に関するすべての決定は、ファイルまたはディレクトリの ACL に表現されているアクセス権によって制御されます。

ZFS ファイルの ACL アクセス権に関する主な規則は、次のとおりです。

- ZFS では、ACL に指定されている順序に従って、上から順番に ACL エントリが処理されます。
- ACL エントリが処理されるのは、アクセスを要求したユーザーが ACL エントリに設定されているユーザーと一致した場合だけです。
- いったん付与した allow アクセス権は、その ACL アクセス権セットの後続の ACL deny エントリで拒否することはできません。
- ファイルの所有者には write_acl アクセス権が無条件で付与されます。そのアクセス権を明示的に拒否した場合でも付与されます。それ以外のユーザーの場合は、指定していないアクセス権はすべて拒否されます。

  deny アクセス権が設定されている場合、またはファイルのアクセス権が失われている場合でも、ファイルの所有者またはスーパーユーザーに許可されるアクセス要求は、特権サブシステムによって決められます。この機構によって、ファイル

の所有者が所有しているファイルから拒否されることがなくなり、スーパーユーザーがファイルを回復するために変更できるようになります。

ディレクトリに非簡易 ACL を設定しても、その ACL はそのディレクトリの子に自動的に継承されることはありません。非簡易 ACL を設定し、それがそのディレクトリの子に継承されるようにする場合は、ACL 継承フラグを使用する必要があります。詳細については、表 8–3 および258 ページの「ZFS ファイルの ACL 継承を冗長形式で設定する」を参照してください。

新しいファイルを作成すると、umask の値に応じて、次のようなデフォルトの簡易 ACL が適用されます。

```
$ ls -v file.1
-rw-r--r--   1 root     root      206663 May 20 14:09 file.1
     0:owner@:execute:deny
     1:owner@:read_data/write_data/append_data/write_xattr/write_attributes
         /write_acl/write_owner:allow
     2:group@:write_data/append_data/execute:deny
     3:group@:read_data:allow
     4:everyone@:write_data/append_data/write_xattr/execute/write_attributes
         /write_acl/write_owner:deny
     5:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
         :allow
```

この例では、ACL エントリがユーザーカテゴリ (owner@、group@、everyone@) ごとに 2 つずつ存在しています。1 つは deny アクセス権のエントリ、もう 1 つは allow アクセス権のエントリです。

このファイル ACL について説明します。

`0:owner@` この所有者は、このファイルでの実行権を拒否されます (execute:deny)。

`1:owner@` この所有者は、このファイルの内容を読み取って変更することができます (read_data/write_data/append_data)。この所有者は、タイムスタンプ、拡張属性、ACL などのファイル属性を変更することもできます (write_xattr/write_attributes /write_acl)。さらに、この所有者はファイルの所有権を変更できます (write_owner:allow)。

`2:group@` このグループは、このファイルを変更および実行するアクセス権を拒否されます (write_data/append_data/execute:deny)。

`3:group@` このグループには、ファイルの読み取りアクセス権が付与されます (read_data:allow)。

`4:everyone@` ファイルの所有者でもファイルの所有グループのメンバーでもないユーザーはすべて、このファイルの内容を実行および変更するアクセス権、およびこのファイルの属性を変更するアクセス権を拒否されます (write_data/append_data/write_xattr/execute/write_attributes/write_acl/write_owner:deny)。

5:everyone@ ファイルの所有者でもファイルの所有グループのメンバーでもないユーザーはすべて、このファイルおよびこのファイルの属性の読み取りアクセス権を付与されます (read_data/read_xattr/read_attributes/read_acl/synchronize:allow)。synchronize の許可アクセス権は、現在のところ実装されていません。

新しいディレクトリを作成すると、umask の値に応じて、次のようなデフォルトのディレクトリ ACL が適用されます。

```
$ ls -dv dir.1
drwxr-xr-x   2 root     root           2 May 20 14:11 dir.1
     0:owner@::deny
     1:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
         /append_data/write_xattr/execute/write_attributes/write_acl
         /write_owner:allow
     2:group@:add_file/write_data/add_subdirectory/append_data:deny
     3:group@:list_directory/read_data/execute:allow
     4:everyone@:add_file/write_data/add_subdirectory/append_data/write_xattr
         /write_attributes/write_acl/write_owner:deny
     5:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
         /read_acl/synchronize:allow
```

このディレクトリ ACL について説明します。

0:owner@ この所有者の deny リストは、このディレクトリについて空です (::deny)。

1:owner@ この所有者は、ディレクトリの内容を読み取って変更すること (list_directory/read_data/add_file/write_data/add_subdirectory/append_data)、内容を検索すること (execute)、およびタイムスタンプ、拡張属性、ACL などのディレクトリ属性を変更すること (write_xattr/write_attributes/write_acl) が許可されます。また、この所有者はディレクトリの所有権を変更できます (write_owner:allow)。

2:group@ このグループは、ディレクトリの内容を追加または変更できません (add_file/write_data/add_subdirectory/append_data:deny)。

3:group@ このグループは、ディレクトリの内容を表示して読み取ることができます。また、このグループには、ディレクトリの内容を検索するアクセス権が許可されます (list_directory/read_data/execute:allow)。

4:everyone@ ファイルの所有者でもファイルの所有グループのメンバーでもないユーザーはすべて、ディレクトリの内容の追加または変更を行うアクセス権を拒否されます (add_file/write_data/add_subdirectory/append_data)。さら

に、ディレクトリの任意の属性を変更するアクセス権も拒否されます (write_xattr/write_attributes/write_acl/write_owner:deny)。

5:everyone@ ファイルの所有者でもファイルの所有グループのメンバーでもないユーザーはすべて、ディレクトリの内容と属性に対する読み取りアクセス権と実行権を付与されます (list_directory/read_data/read_xattr/execute/read_attributes/read_acl/synchronize:allow)。synchronize の許可アクセス権は、現在のところ実装されていません。

ZFS ファイルの ACL を冗長形式で設定および表示する

chmod コマンドを使用して、ZFS ファイルの ACL を変更できます。次の chmod 構文では、ACL を変更するために *acl-specification* を使って ACL の形式を指定しています。*acl-specification* については、245 ページの「ACL を設定する構文の説明」を参照してください。

- ACL エントリを追加する
  - ユーザーの ACL エントリを追加する

    ```
    % chmod A+acl-specification filename
    ```
  - *index-ID* を使用して ACL エントリを追加する

    ```
    % chmod Aindex-ID+acl-specification filename
    ```

    この構文では、指定した *index-ID* の位置に新しい ACL エントリが挿入されます。
- ACL エントリを置き換える

  ```
  % chmod A=acl-specification filename
  % chmod Aindex-ID=acl-specification filename
  ```
- ACL エントリを削除する
  - *index-ID* を使用して ACL エントリを削除する

    ```
    % chmod Aindex-ID- filename
    ```
  - ユーザーを使用して ACL エントリを削除する

    ```
    % chmod A-acl-specification filename
    ```
  - 非簡易 ACL をファイルからすべて削除する

    ```
    % chmod A- filename
    ```

ls - v コマンドを使用することで、詳細な ACL 情報が表示されます。次に例を示します。

```
# ls -v file.1
-rw-r--r--   1 root     root      206663 May 20 14:09 file.1
     0:owner@:execute:deny
     1:owner@:read_data/write_data/append_data/write_xattr/write_attributes
         /write_acl/write_owner:allow
     2:group@:write_data/append_data/execute:deny
     3:group@:read_data:allow
     4:everyone@:write_data/append_data/write_xattr/execute/write_attributes
         /write_acl/write_owner:deny
     5:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
         :allow
```

コンパクト形式の ACL の使用方法については、265 ページの「ZFS ファイルの ACL をコンパクト形式で設定および表示する」を参照してください。

例 8–1 ZFS ファイルの簡易 ACL を変更する

この節では、簡易 ACL を設定して表示する例を紹介します。

次の例では、簡易 ACL が `file.1` にあります。

```
# ls -v file.1
-rw-r--r--   1 root     root      206663 May 20 15:03 file.1
     0:owner@:execute:deny
     1:owner@:read_data/write_data/append_data/write_xattr/write_attributes
         /write_acl/write_owner:allow
     2:group@:write_data/append_data/execute:deny
     3:group@:read_data:allow
     4:everyone@:write_data/append_data/write_xattr/execute/write_attributes
         /write_acl/write_owner:deny
     5:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
         :allow
```

次の例では、`write_data` アクセス権が `group@` に付与されます。

```
# chmod A2=group@:append_data/execute:deny file.1
# chmod A3=group@:read_data/write_data:allow file.1
# ls -v file.1
-rw-rw-r--   1 root     root      206663 May 20 15:03 file.1
     0:owner@:execute:deny
     1:owner@:read_data/write_data/append_data/write_xattr/write_attributes
         /write_acl/write_owner:allow
     2:group@:append_data/execute:deny
     3:group@:read_data/write_data:allow
     4:everyone@:write_data/append_data/write_xattr/execute/write_attributes
         /write_acl/write_owner:deny
     5:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
         :allow
```

次の例では、`file.1` へのアクセス権の設定が `644` に戻されます。

```
# chmod 644 file.1
# ls -v file.1
-rw-r--r--   1 root     root      206663 May 20 15:03 file.1
     0:owner@:execute:deny
```

例 8–1 ZFS ファイルの簡易 ACL を変更する (続き)

```
     1:owner@:read_data/write_data/append_data/write_xattr/write_attributes
         /write_acl/write_owner:allow
     2:group@:write_data/append_data/execute:deny
     3:group@:read_data:allow
     4:everyone@:write_data/append_data/write_xattr/execute/write_attributes
         /write_acl/write_owner:deny
     5:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
         :allow
```

例 8–2 ZFS ファイルに非簡易 ACL を設定する

この節では、非簡易 ACL を設定して表示する例を紹介します。

次の例では、read_data/execute アクセス権が、test.dir ディレクトリのユーザー gozer に追加されます。

```
# chmod A+user:gozer:read_data/execute:allow test.dir
# ls -dv test.dir
drwxr-xr-x+  2 root     root           2 May 20 15:09 test.dir
     0:user:gozer:list_directory/read_data/execute:allow
     1:owner@::deny
     2:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
         /append_data/write_xattr/execute/write_attributes/write_acl
         /write_owner:allow
     3:group@:add_file/write_data/add_subdirectory/append_data:deny
     4:group@:list_directory/read_data/execute:allow
     5:everyone@:add_file/write_data/add_subdirectory/append_data/write_xattr
         /write_attributes/write_acl/write_owner:deny
     6:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
         /read_acl/synchronize:allow
```

次の例では、read_data/execute アクセス権がユーザー gozer から削除されます。

```
# chmod A0- test.dir
# ls -dv test.dir
drwxr-xr-x   2 root     root           2 May 20 15:09 test.dir
     0:owner@::deny
     1:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
         /append_data/write_xattr/execute/write_attributes/write_acl
         /write_owner:allow
     2:group@:add_file/write_data/add_subdirectory/append_data:deny
     3:group@:list_directory/read_data/execute:allow
     4:everyone@:add_file/write_data/add_subdirectory/append_data/write_xattr
         /write_attributes/write_acl/write_owner:deny
     5:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
         /read_acl/synchronize:allow
```

例 8–3 ACL を使用して ZFS ファイルのアクセス権を操作する

これらの例では、ACL を設定してから、ファイルまたはディレクトリのアクセス権を変更するまでの操作を説明します。

例 8–3　ACL を使用して ZFS ファイルのアクセス権を操作する　(続き)

次の例では、簡易 ACL が file.2 にあります。

```
# ls -v file.2
-rw-r--r--   1 root     root        3103 May 20 15:23 file.2
     0:owner@:execute:deny
     1:owner@:read_data/write_data/append_data/write_xattr/write_attributes
         /write_acl/write_owner:allow
     2:group@:write_data/append_data/execute:deny
     3:group@:read_data:allow
     4:everyone@:write_data/append_data/write_xattr/execute/write_attributes
         /write_acl/write_owner:deny
     5:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
         :allow
```

次の例では、ACL allow アクセス権が everyone@ から削除されます。

```
# chmod A5- file.2
# ls -v file.2
-rw-r-----+  1 root     root        3103 May 20 15:23 file.2
     0:owner@:execute:deny
     1:owner@:read_data/write_data/append_data/write_xattr/write_attributes
         /write_acl/write_owner:allow
     2:group@:write_data/append_data/execute:deny
     3:group@:read_data:allow
     4:everyone@:write_data/append_data/write_xattr/execute/write_attributes
         /write_acl/write_owner:deny
```

この出力では、ファイルのアクセス権が 644 から 640 に再設定されています。everyone@ の読み取りアクセス権は、everyone@ の ACL allow アクセス権が削除されるときに、ファイルのアクセス権から事実上削除されています。

次の例では、既存の ACL が everyone@ の read_data/write_data アクセス権に置き換わります。

```
# chmod A=everyone@:read_data/write_data:allow file.3
# ls -v file.3
-rw-rw-rw-+  1 root     root        6986 May 20 15:25 file.3
     0:everyone@:read_data/write_data:allow
```

この出力では、chmod 構文を使って、owner@、group@、および everyone@ が読み取りまたは書き込みできるように、既存の ACL を read_data/write_data:allow アクセス権に事実上置き換えています。このモデルでは、everyone@ を使って、すべてのユーザーまたはグループへのアクセス権を指定しています。所有者とグループのアクセス権を上書きする owner@ と group@ の ACL エントリがないので、アクセス権は 666 に設定されます。

次の例では、既存の ACL がユーザー gozer の読み取りアクセス権に置き換わります。

例 8–3 ACL を使用して ZFS ファイルのアクセス権を操作する (続き)

```
# chmod A=user:gozer:read_data:allow file.3
# ls -v file.3
----------+  1 root     root        6986 May 20 15:25 file.3
     0:user:gozer:read_data:allow
```

この出力では、従来のファイルアクセス権コンポーネントを表す owner@、group@、または everyone@ の ACL エントリがないので、ファイルアクセス権は 000 になります。ファイルの所有者は、次のようにアクセス権 (および ACL) を再設定することで、この問題を解決できます。

```
# chmod 655 file.3
# ls -v file.3
-rw-r-xr-x+  1 root     root        6986 May 20 15:25 file.3
     0:user:gozer::deny
     1:user:gozer:read_data:allow
     2:owner@:execute:deny
     3:owner@:read_data/write_data/append_data/write_xattr/write_attributes
         /write_acl/write_owner:allow
     4:group@:write_data/append_data:deny
     5:group@:read_data/execute:allow
     6:everyone@:write_data/append_data/write_xattr/write_attributes
         /write_acl/write_owner:deny
     7:everyone@:read_data/read_xattr/execute/read_attributes/read_acl
         /synchronize:allow
```

例 8–4 ZFS ファイルの簡易 ACL を復元する

chmod コマンドを使用して、ファイルまたはディレクトリの非簡易 ACL をすべて削除し、ファイルまたはディレクトリの簡易 ACL を復元することができます。

次の例では、2 つの非簡易 ACL が test5.dir にあります。

```
# ls -dv test5.dir
drwxr-xr-x+  2 root     root           2 May 20 15:32 test5.dir
     0:user:lp:read_data:file_inherit:deny
     1:user:gozer:read_data:file_inherit:deny
     2:owner@::deny
     3:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
         /append_data/write_xattr/execute/write_attributes/write_acl
         /write_owner:allow
     4:group@:add_file/write_data/add_subdirectory/append_data:deny
     5:group@:list_directory/read_data/execute:allow
     6:everyone@:add_file/write_data/add_subdirectory/append_data/write_xattr
         /write_attributes/write_acl/write_owner:deny
     7:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
         /read_acl/synchronize:allow
```

次の例では、ユーザー gozer と lp の非簡易 ACL が削除されます。owner@、group@、および everyone@ の残りの ACL には、6 個のデフォルト値が設定されています。

例 8–4 ZFS ファイルの簡易 ACL を復元する (続き)

```
# chmod A- test5.dir
# ls -dv test5.dir
drwxr-xr-x   2 root     root           2 May 20 15:32 test5.dir
     0:owner@::deny
     1:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
         /append_data/write_xattr/execute/write_attributes/write_acl
         /write_owner:allow
     2:group@:add_file/write_data/add_subdirectory/append_data:deny
     3:group@:list_directory/read_data/execute:allow
     4:everyone@:add_file/write_data/add_subdirectory/append_data/write_xattr
         /write_attributes/write_acl/write_owner:deny
     5:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
         /read_acl/synchronize:allow
```

ZFS ファイルの ACL 継承を冗長形式で設定する

ファイルやディレクトリの ACL を継承するかどうかやその継承方法を指定できます。デフォルトでは、ACL は伝達されません。ディレクトリに非簡易 ACL を設定した場合でも、その ACL はそれ以降に作成されるディレクトリには継承されません。ACL を継承する場合は、ファイルまたはディレクトリにそのことを指定する必要があります。

また、次の 2 つの ACL プロパティーをファイルシステムにグローバルに設定できます。aclinherit と aclmode です。デフォルトでは、aclinherit は restricted に、aclmode は groupmask に設定されています。

詳細については、248 ページの「ACL 継承」を参照してください。

例 8–5 デフォルトの ACL 継承を許可する

デフォルトでは、ACL はディレクトリ階層に伝達されません。

次の例では、read_data/write_data/execute の非簡易 ACL が、test.dir ディレクトリのユーザー gozer に適用されます。

```
# chmod A+user:gozer:read_data/write_data/execute:allow test.dir
# ls -dv test.dir
drwxr-xr-x+  2 root     root           2 May 20 15:41 test.dir
     0:user:gozer:list_directory/read_data/add_file/write_data/execute:allow
     1:owner@::deny
     2:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
         /append_data/write_xattr/execute/write_attributes/write_acl
         /write_owner:allow
     3:group@:add_file/write_data/add_subdirectory/append_data:deny
     4:group@:list_directory/read_data/execute:allow
     5:everyone@:add_file/write_data/add_subdirectory/append_data/write_xattr
         /write_attributes/write_acl/write_owner:deny
```

例 8–5 デフォルトの ACL 継承を許可する (続き)

```
     6:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
         /read_acl/synchronize:allow
```

test.dir サブディレクトリが作成されても、ユーザー gozer の ACE は伝達されません。サブディレクトリ上でユーザー gozer に許可されているアクセス権がファイル所有者、グループメンバー、または everyone@ としてのアクセス権の場合には、このユーザーはサブディレクトリにしかアクセスできません。次に例を示します。

```
# mkdir test.dir/sub.dir
# ls -dv test.dir/sub.dir
drwxr-xr-x   2 root     root           2 May 20 15:42 test.dir/sub.dir
     0:owner@::deny
     1:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
         /append_data/write_xattr/execute/write_attributes/write_acl
         /write_owner:allow
     2:group@:add_file/write_data/add_subdirectory/append_data:deny
     3:group@:list_directory/read_data/execute:allow
     4:everyone@:add_file/write_data/add_subdirectory/append_data/write_xattr
         /write_attributes/write_acl/write_owner:deny
     5:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
         /read_acl/synchronize:allow
```

例 8–6 ファイルとディレクトリの ACL 継承を許可する

次の一連の例では、file_inherit フラグが設定されているときに適用されるファイルとディレクトリの ACE を示しています。

次の例では、test2.dir ディレクトリ上のファイルへの read_data/write_data アクセス権がユーザー gozer に追加されます。このユーザーは、新しく作成されたすべてのファイルに読み取りアクセスできるようになります。

```
# chmod A+user:gozer:read_data/write_data:file_inherit:allow test2.dir
# ls -dv test2.dir
drwxr-xr-x+  2 root     root           2 May 20 15:50 test2.dir
     0:user:gozer:read_data/write_data:file_inherit:allow
     1:owner@::deny
     2:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
         /append_data/write_xattr/execute/write_attributes/write_acl
         /write_owner:allow
     3:group@:add_file/write_data/add_subdirectory/append_data:deny
     4:group@:list_directory/read_data/execute:allow
     5:everyone@:add_file/write_data/add_subdirectory/append_data/write_xattr
         /write_attributes/write_acl/write_owner:deny
     6:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
         /read_acl/synchronize:allow
```

次の例では、ユーザー gozer のアクセス権が、新しく作成されたファイル test2.dir/file.2 に適用されます。ACL 継承が許可されているので (read_data:file_inherit:allow)、ユーザー gozer は新しく作成されたすべてのファイルの内容を読み取ることができます。

例 8–6 ファイルとディレクトリの ACL 継承を許可する (続き)

```
# touch test2.dir/file.2
# ls -v test2.dir/file.2
-rw-r--r--+  1 root     root           0 May 20 15:51 test2.dir/file.2
     0:user:gozer:write_data:deny
     1:user:gozer:read_data/write_data:allow
     2:owner@:execute:deny
     3:owner@:read_data/write_data/append_data/write_xattr/write_attributes
         /write_acl/write_owner:allow
     4:group@:write_data/append_data/execute:deny
     5:group@:read_data:allow
     6:everyone@:write_data/append_data/write_xattr/execute/write_attributes
         /write_acl/write_owner:deny
     7:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
         :allow
```

このファイルの aclmode プロパティーがデフォルト値 groupmask に設定されているため、ユーザー gozer には file.2 への write_data アクセス権は割り当てられません。これは、ファイルのグループアクセス権が許可していないためです。

inherit_only アクセス権は、file_inherit または dir_inherit フラグが設定されているときに適用されます。このアクセス権は、ディレクトリ階層に ACL を伝達するために使用します。この場合、ユーザー gozer のアクセス権の許可または拒否は、ファイル所有者またはファイルのグループ所有者のメンバーである場合を除いて、everyone@ アクセス権に基づいてのみ行われます。次に例を示します。

```
# mkdir test2.dir/subdir.2
# ls -dv test2.dir/subdir.2
drwxr-xr-x+  2 root     root           2 May 20 15:52 test2.dir/subdir.2
     0:user:gozer:list_directory/read_data/add_file/write_data:file_inherit
         /inherit_only:allow
     1:owner@::deny
     2:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
         /append_data/write_xattr/execute/write_attributes/write_acl
         /write_owner:allow
     3:group@:add_file/write_data/add_subdirectory/append_data:deny
     4:group@:list_directory/read_data/execute:allow
     5:everyone@:add_file/write_data/add_subdirectory/append_data/write_xattr
         /write_attributes/write_acl/write_owner:deny
     6:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
         /read_acl/synchronize:allow
```

次の例では、file_inherit と dir_inherit フラグが両方設定されているときに適用される、ファイルとディレクトリの ACL を示しています。

次の例では、ユーザー gozer に読み取り、書き込み、および実行権が付与されます。これらのアクセス権は、新しく作成されたファイルとディレクトリに継承されます。

```
# chmod A+user:gozer:read_data/write_data/execute:file_inherit/dir_inherit:allow
test3.dir
```

例 8–6 ファイルとディレクトリの ACL 継承を許可する (続き)

```
# ls -dv test3.dir
drwxr-xr-x+  2 root     root           2 May 20 15:53 test3.dir
     0:user:gozer:list_directory/read_data/add_file/write_data/execute
         :file_inherit/dir_inherit:allow
     1:owner@::deny
     2:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
         /append_data/write_xattr/execute/write_attributes/write_acl
         /write_owner:allow
     3:group@:add_file/write_data/add_subdirectory/append_data:deny
     4:group@:list_directory/read_data/execute:allow
     5:everyone@:add_file/write_data/add_subdirectory/append_data/write_xattr
         /write_attributes/write_acl/write_owner:deny
     6:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
         /read_acl/synchronize:allow

# touch test3.dir/file.3
# ls -v test3.dir/file.3
-rw-r--r--+  1 root     root           0 May 20 15:58 test3.dir/file.3
     0:user:gozer:write_data/execute:deny
     1:user:gozer:read_data/write_data/execute:allow
     2:owner@:execute:deny
     3:owner@:read_data/write_data/append_data/write_xattr/write_attributes
         /write_acl/write_owner:allow
     4:group@:write_data/append_data/execute:deny
     5:group@:read_data:allow
     6:everyone@:write_data/append_data/write_xattr/execute/write_attributes
         /write_acl/write_owner:deny
     7:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
         :allow

# mkdir test3.dir/subdir.1
# ls -dv test3.dir/subdir.1
drwxr-xr-x+  2 root     root           2 May 20 15:59 test3.dir/subdir.1
     0:user:gozer:list_directory/read_data/add_file/write_data/execute
         :file_inherit/dir_inherit/inherit_only:allow
     1:user:gozer:add_file/write_data:deny
     2:user:gozer:list_directory/read_data/add_file/write_data/execute:allow
     3:owner@::deny
     4:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
         /append_data/write_xattr/execute/write_attributes/write_acl
         /write_owner:allow
     5:group@:add_file/write_data/add_subdirectory/append_data:deny
     6:group@:list_directory/read_data/execute:allow
     7:everyone@:add_file/write_data/add_subdirectory/append_data/write_xattr
         /write_attributes/write_acl/write_owner:deny
     8:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
         /read_acl/synchronize:allow
```

これらの例では、group@ および everyone@ の親ディレクトリのアクセス権によって、書き込みアクセス権と実行権が拒否されます。このため、ユーザー gozer は書き込みアクセス権と実行権が拒否されます。デフォルトの aclinherit プロパティーは restricted です。つまり、write_data および execute アクセス権が継承されません。

例 8–6 ファイルとディレクトリの ACL 継承を許可する (続き)

次の例では、ユーザー gozer に読み取り、書き込み、および実行権が付与されます。これらのアクセス権は、新しく作成されたファイルに継承されます。ただしそれらは、ディレクトリの後続の内容には伝達されません。

```
# chmod A+user:gozer:read_data/write_data/execute:file_inherit/no_propagate:allow
test4.dir
# ls -dv test4.dir
drwxr-xr-x+  2 root     root           2 May 20 16:02 test4.dir
     0:user:gozer:list_directory/read_data/add_file/write_data/execute
         :file_inherit/no_propagate:allow
     1:owner@::deny
     2:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
         /append_data/write_xattr/execute/write_attributes/write_acl
         /write_owner:allow
     3:group@:add_file/write_data/add_subdirectory/append_data:deny
     4:group@:list_directory/read_data/execute:allow
     5:everyone@:add_file/write_data/add_subdirectory/append_data/write_xattr
         /write_attributes/write_acl/write_owner:deny
     6:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
         /read_acl/synchronize:allow
```

次の例に示すように、新しいサブディレクトリを作成しても、ユーザー gozer のファイルへの read_data/write_data/execute アクセス権は、その新しい sub4.dir ディレクトリには伝達されません。

```
mkdir test4.dir/sub4.dir
# ls -dv test4.dir/sub4.dir
drwxr-xr-x   2 root     root           2 May 20 16:03 test4.dir/sub4.dir
     0:owner@::deny
     1:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
         /append_data/write_xattr/execute/write_attributes/write_acl
         /write_owner:allow
     2:group@:add_file/write_data/add_subdirectory/append_data:deny
     3:group@:list_directory/read_data/execute:allow
     4:everyone@:add_file/write_data/add_subdirectory/append_data/write_xattr
         /write_attributes/write_acl/write_owner:deny
     5:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
         /read_acl/synchronize:allow
```

次の例に示すように、ユーザー gozer のファイルへの read_data/write_data/execute アクセス権は、新しく作成したファイルに伝達されます。

```
# touch test4.dir/file.4
# ls -v test4.dir/file.4
-rw-r--r--+  1 root     root           0 May 20 16:04 test4.dir/file.4
     0:user:gozer:write_data/execute:deny
     1:user:gozer:read_data/write_data/execute:allow
     2:owner@:execute:deny
     3:owner@:read_data/write_data/append_data/write_xattr/write_attributes
         /write_acl/write_owner:allow
     4:group@:write_data/append_data/execute:deny
```

例 8–6 ファイルとディレクトリの ACL 継承を許可する (続き)

```
     5:group@:read_data:allow
     6:everyone@:write_data/append_data/write_xattr/execute/write_attributes
         /write_acl/write_owner:deny
     7:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
         :allow
```

例 8–7 aclmode プロパティーを passthrough に設定した場合の ACL 継承

次の例からわかるように、tank/cindys ファイルシステムの aclmode プロパティーを passthrough に設定すると、ユーザー gozer は、新しく作成された file.4 の test4.dir ディレクトリに適用された ACL を継承します。

```
# zfs set aclmode=passthrough tank/cindys
# touch test4.dir/file.4
# ls -v test4.dir/file.4
-rw-r--r--+  1 root     root           0 May 20 16:08 test4.dir/file.4
     0:user:gozer:write_data/execute:deny
     1:user:gozer:read_data/write_data/execute:allow
     2:owner@:execute:deny
     3:owner@:read_data/write_data/append_data/write_xattr/write_attributes
         /write_acl/write_owner:allow
     4:group@:write_data/append_data/execute:deny
     5:group@:read_data:allow
     6:everyone@:write_data/append_data/write_xattr/execute/write_attributes
         /write_acl/write_owner:deny
     7:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
         :allow
```

この出力は、親ディレクトリ test4.dir に設定されている read_data/write_data/execute:allow:file_inherit/dir_inherit ACL が、ユーザー gozer にそのまま渡されることを示しています。

例 8–8 aclmode プロパティーを discard に設定した場合の ACL 継承

ファイルシステムの aclmode プロパティーが discard に設定されている場合には、ディレクトリのアクセス権が変更されたときに、ACL が破棄される可能性があります。次に例を示します。

```
# zfs set aclmode=discard tank/cindys
# chmod A+user:gozer:read_data/write_data/execute:dir_inherit:allow test5.dir
# ls -dv test5.dir
drwxr-xr-x+  2 root     root           2 May 20 16:09 test5.dir
     0:user:gozer:list_directory/read_data/add_file/write_data/execute
         :dir_inherit:allow
     1:owner@::deny
     2:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
         /append_data/write_xattr/execute/write_attributes/write_acl
         /write_owner:allow
     3:group@:add_file/write_data/add_subdirectory/append_data:deny
     4:group@:list_directory/read_data/execute:allow
     5:everyone@:add_file/write_data/add_subdirectory/append_data/write_xattr
```

例 8–8 aclmode プロパティーを discard に設定した場合の ACL 継承 (続き)

```
        /write_attributes/write_acl/write_owner:deny
     6:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
        /read_acl/synchronize:allow
```

あとでディレクトリのアクセス権をより厳格に設定することにした場合は、非簡易 ACL は破棄されます。次に例を示します。

```
# chmod 744 test5.dir
# ls -dv test5.dir
drwxr--r--   2 root     root           2 May 20 16:09 test5.dir
     0:owner@::deny
     1:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
        /append_data/write_xattr/execute/write_attributes/write_acl
        /write_owner:allow
     2:group@:add_file/write_data/add_subdirectory/append_data/execute:deny
     3:group@:list_directory/read_data:allow
     4:everyone@:add_file/write_data/add_subdirectory/append_data/write_xattr
        /execute/write_attributes/write_acl/write_owner:deny
     5:everyone@:list_directory/read_data/read_xattr/read_attributes/read_acl
        /synchronize:allow
```

例 8–9 aclinherit プロパティーを noallow に設定した場合の ACL 継承

次の例では、ファイルに継承される 2 つの非簡易 ACL が設定されます。一方の ACL では read_data アクセス権が許可され、もう一方の ACL では read_data アクセス権が拒否されます。この例では、1 つの chmod コマンドに 2 つの ACE を指定できることも示しています。

```
# zfs set aclinherit=noallow tank/cindys
# chmod A+user:gozer:read_data:file_inherit:deny,user:lp:read_data:file_inherit:allow
test6.dir
# ls -dv test6.dir
drwxr-xr-x+  2 root     root           2 May 20 16:11 test6.dir
     0:user:gozer:read_data:file_inherit:deny
     1:user:lp:read_data:file_inherit:allow
     2:owner@::deny
     3:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
        /append_data/write_xattr/execute/write_attributes/write_acl
        /write_owner:allow
     4:group@:add_file/write_data/add_subdirectory/append_data:deny
     5:group@:list_directory/read_data/execute:allow
     6:everyone@:add_file/write_data/add_subdirectory/append_data/write_xattr
        /write_attributes/write_acl/write_owner:deny
     7:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
        /read_acl/synchronize:allow
```

次の例に示すように、新しいファイルが作成されると、read_data アクセス権を許可する ACL が破棄されます。

```
# touch test6.dir/file.6
# ls -v test6.dir/file.6
```

例 8–9 aclinherit プロパティーを noallow に設定した場合の ACL 継承 (続き)

```
-rw-r--r--   1 root     root           0 May 20 16:13 test6.dir/file.6
     0:owner@:execute:deny
     1:owner@:read_data/write_data/append_data/write_xattr/write_attributes
         /write_acl/write_owner:allow
     2:group@:write_data/append_data/execute:deny
     3:group@:read_data:allow
     4:everyone@:write_data/append_data/write_xattr/execute/write_attributes
         /write_acl/write_owner:deny
     5:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
         :allow
```

ZFS ファイルの ACL をコンパクト形式で設定および表示する

ZFS ファイルのアクセス権をコンパクト形式で設定および表示できます。コンパクト形式では、一意の 14 個の文字を使ってアクセス権を表現します。アクセス権を表現するためのコンパクト形式の文字については、表 8–2 および表 8–3 を参照してください。

ファイルとディレクトリのコンパクト形式の ACL リストは、`ls -V` コマンドを使用して表示できます。次に例を示します。

```
# ls -V file.1
-rw-r--r--   1 root     root      206663 Jun 17 10:07 file.1
           owner@:--x-----------:------:deny
           owner@:rw-p---A-W-Co-:------:allow
           group@:-wxp----------:------:deny
           group@:r-------------:------:allow
        everyone@:-wxp---A-W-Co-:------:deny
        everyone@:r-----a-R-c--s:------:allow
```

このコンパクト形式の ACL 出力について説明します。

`owner@` この所有者は、このファイルを実行するアクセス権を拒否されます (`x=execute`)。

`owner@` この所有者は、このファイルの内容を読み取って変更することができます (`rw=read_data/write_data`、`p=append_data`)。この所有者は、タイムスタンプ、拡張属性、ACL などのファイルの属性を変更することもできます (`A=write_xattr`、`W=write_attributes`、`C=write_acl`)。さらに、この所有者はファイルの所有権を変更できます (`o=write_owner`)。

`group@` このグループは、このファイルを変更および実行するアクセス権を拒否されます (`write_data`、`p=append_data`、`x=execute`)。

`group@` このグループには、このファイルを読み取るアクセス権が付与されます (`r=read_data`)。

`everyone@` ファイルの所有者でもファイルの所有グループのメンバーでもないユーザーはすべて、このファイルの内容を実行および変更するアクセス権、およびこのファイルの属性を変更するアクセス権を拒否されます (`w=write_data`、`x=execute`、`p =append_data`、`A=write_xattr`、`W=write_attributes`、`C=write_acl`、`o=write_owner`)。

`everyone@` ファイルの所有者でもファイルの所有グループのメンバーでもないユーザーはすべて、このファイルおよびこのファイルの属性の読み取りアクセス権を付与されます (`r=read_data`、`a=append_data`、`R=read_xattr`、`c=read_acl`、`s=synchronize`)。`synchronize` の許可アクセス権は、現在のところ実装されていません。

コンパクト形式の ACL には、冗長形式の ACL と比べて次の利点があります。

- アクセス権を `chmod` コマンドに指定するときに、順対応引数として指定できます。
- アクセス権がないことを示すハイフン (-) 文字は、削除してもかまいません。必要な文字だけ指定します。
- アクセス権と継承フラグは、同じ方法で設定します。

冗長形式の ACL の使用方法については、253 ページの「ZFS ファイルの ACL を冗長形式で設定および表示する」を参照してください。

例 8–10 コンパクト形式で ACL を設定および表示する

次の例では、簡易 ACL が `file.1` にあります。

```
# ls -V file.1
-rw-r--r--   1 root     root      206663 Jun 17 10:07 file.1
            owner@:--x-----------:------:deny
            owner@:rw-p---A-W-Co-:------:allow
            group@:-wxp----------:------:deny
            group@:r-------------:------:allow
         everyone@:-wxp---A-W-Co-:------:deny
         everyone@:r-----a-R-c--s:------:allow
```

次の例では、`read_data/execute` アクセス権が、`file.1` のユーザー `gozer` に追加されます。

```
# chmod A+user:gozer:rx:allow file.1
# ls -V file.1
-rw-r--r--+  1 root     root      206663 Jun 17 10:07 file.1
        user:gozer:r-x-----------:------:allow
            owner@:--x-----------:------:deny
            owner@:rw-p---A-W-Co-:------:allow
            group@:-wxp----------:------:deny
            group@:r-------------:------:allow
```

例 8–10 コンパクト形式で ACL を設定および表示する (続き)

```
         everyone@:-wxp---A-W-Co-:------:deny
         everyone@:r-----a-R-c--s:------:allow
```

また、所定の場所 (この例では 4) に新しい ACL エントリを挿入しても、ユーザー gozer に同じアクセス権を追加することができます。これにより、場所 4–6 の既存の ACL が下に移動します。次に例を示します。

```
# chmod A4+user:gozer:rx:allow file.1
# ls -V file.1
-rw-r--r--+  1 root     root      206663 Jun 17 10:16 file.1
            owner@:--x-----------:------:deny
            owner@:rw-p---A-W-Co-:------:allow
            group@:-wxp----------:------:deny
            group@:r-------------:------:allow
        user:gozer:r-x-----------:------:allow
         everyone@:-wxp---A-W-Co-:------:deny
         everyone@:r-----a-R-c--s:------:allow
```

次の例では、ユーザー gozer に読み取り、書き込み、および実行権が付与されます。これらのアクセス権は、新しく作成されたファイルとディレクトリに継承されます。

```
# chmod A+user:gozer:rwx:fd:allow dir.2
# ls -dV dir.2
drwxr-xr-x+  2 root     root           2 Jun 17 10:19 dir.2
        user:gozer:rwx-----------:fd----:allow
            owner@:--------------:------:deny
            owner@:rwxp---A-W-Co-:------:allow
            group@:-w-p----------:------:deny
            group@:r-x-----------:------:allow
         everyone@:-w-p---A-W-Co-:------:deny
         everyone@:r-x---a-R-c--s:------:allow
```

ls -V の出力にあるアクセス権と継承フラグをコンパクト形式の chmod にカット&ペーストすることもできます。たとえば、ユーザー gozer に割り当てられている dir.2 ディレクトリのアクセス権と継承フラグを同じ dir.2 上のユーザー cindys に複製するには、そのアクセス権と継承フラグ (rwx-----------:fd----:allow) を chmod コマンドに次のようにコピー&ペーストします。

```
# chmod A+user:cindys:rwx-----------:fd----:allow dir.2
# ls -dV dir.2
drwxr-xr-x+  2 root     root           2 Jun 17 10:19 dir.2
       user:cindys:rwx-----------:fd----:allow
        user:gozer:rwx-----------:fd----:allow
            owner@:--------------:------:deny
            owner@:rwxp---A-W-Co-:------:allow
            group@:-w-p----------:------:deny
            group@:r-x-----------:------:allow
         everyone@:-w-p---A-W-Co-:------:deny
         everyone@:r-x---a-R-c--s:------:allow
```

例 8–11 aclinherit プロパティーを passthrough に設定した場合の ACL 継承

aclinherit プロパティーが passthrough に設定されているファイルシステムは、継承時に ACL エントリに加えられた変更を除く、継承可能なすべての ACL エントリを継承します。このプロパティーが passthrough に設定されている場合、作成されるファイルのアクセス権は継承可能な ACL によって決定されます。アクセス権に影響を与える継承可能な ACE が存在しない場合、アクセス権はアプリケーションから要求されたアクセス権に従って設定されます。

次の例では、コンパクト形式の ACL 構文を使用して、aclinherit プロパティーを passthrough に設定することによってアクセス権を継承する方法を示します。

次の例では、ACL を test1.dir ディレクトリに設定して継承を強制します。この構文によって新しく作成されたファイルには、owner@、group@、および everyone@ ACL エントリが作成されます。新しく作成されたディレクトリには、owner@、group@、および everyone@ ACL エントリが継承されます。これらに加え、ディレクトリには、新しく作成されるディレクトリとファイルに ACE を伝達する 6 つの ACE も継承されます。

```
# zfs set aclinherit=passthrough tank/cindys
# pwd
/tank/cindys
# mkdir test1.dir

# chmod A=owner@:rwxpcCosRrWaAdD:fd:allow,group@:rwxp:fd:allow,everyone@::fd:allow
test1.dir
# ls -Vd test1.dir
drwxrwx---+  2 root     root           2 Jun 17 10:37 test1.dir
            owner@:rwxpdDaARWcCos:fd----:allow
            group@:rwxp----------:fd----:allow
         everyone@:--------------:fd----:allow
```

次の例では、新しく作成されるファイルに継承されるように指定されていた ACL が、新しく作成されたファイルに継承されます。

```
# cd test1.dir
# touch file.1
# ls -V file.1
-rwxrwx---+  1 root     root           0 Jun 17 10:38 file.1
            owner@:rwxpdDaARWcCos:------:allow
            group@:rwxp----------:------:allow
         everyone@:--------------:------:allow
```

次の例では、新しく作成されたディレクトリに、このディレクトリへのアクセスを制御する ACE と、この新しく作成されたディレクトリの子にあとで伝達するための ACE が継承されます。

```
# mkdir subdir.1
# ls -dV subdir.1
drwxrwx---+  2 root     root           2 Jun 17 10:39 subdir.1
```

例 8–11 aclinherit プロパティーを passthrough に設定した場合の ACL 継承 (続き)

```
            owner@:rwxpdDaARWcCos:fdi---:allow
            owner@:rwxpdDaARWcCos:------:allow
            group@:rwxp----------:fdi---:allow
            group@:rwxp----------:------:allow
         everyone@:--------------:fdi---:allow
         everyone@:--------------:------:allow
```

-di-- エントリと f-i--- エントリは継承の伝達に関するもので、アクセス制御時には考慮されません。次の例では、簡易 ACL の設定されたファイルが別のディレクトリ作成されます。このディレクトリには、継承される ACE はありません。

```
# cd /tank/cindys
# mkdir test2.dir
# cd test2.dir
# touch file.2
# ls -V file.2
-rw-r--r--   1 root     root           0 Jun 17 10:40 file.2
            owner@:--x-----------:------:deny
            owner@:rw-p---A-W-Co-:------:allow
            group@:-wxp----------:------:deny
            group@:r-------------:------:allow
         everyone@:-wxp---A-W-Co-:------:deny
         everyone@:r-----a-R-c--s:------:allow
```

例 8–12 aclinherit プロパティーを passthrough-x に設定した場合の ACL 継承

aclinherit プロパティーが passthrough-x に設定されていると、ファイル作成モードおよびそのモードに影響する継承可能な ACE モードで実行権が設定されている場合に限り、owner@、group@、または everyone@ の実行 (x) 権を使用してファイルが作成されます。

次の例では、aclinherit プロパティーを passthrough-x に設定して実行権を継承する方法を示します。

```
# zfs set aclinherit=passthrough-x tank/cindys
```

次の ACL は /tank/cindys/test1.dir で設定されており、owner@、 group@、および everyone@ のファイルに対する実行可能 ACL 継承を提供します。

```
# chmod A=owner@:rwxpcCosRrWaAdD:fd:allow,group@:rwxp:fd:allow,everyone@::fd:allow test1.dir
# ls -Vd test1.dir
drwxrwx---+  2 root     root           2 Jun 17 10:41 test1.dir
            owner@:rwxpdDaARWcCos:fd----:allow
            group@:rwxp----------:fd----:allow
         everyone@:--------------:fd----:allow
```

要求されたアクセス権 0666 を使用してファイル (file1) が作成されます。この結果、アクセス権 0660 が設定されます。作成モードで要求していないため、実行権は継承されません。

例 8–12 aclinherit プロパティーを passthrough-x に設定した場合の ACL 継承 (続き)

```
# touch test1.dir/file1
# ls -V test1.dir/file1
-rw-rw----+  1 root     root           0 Jun 17 10:42 test1.dir/file1
            owner@:rw-pdDaARWcCos:------:allow
            group@:rw-p----------:------:allow
         everyone@:--------------:------:allow
```

次に、t という実行可能ファイルが、cc コンパイラを使用して testdir ディレクトリーに作成されます。

```
# cc -o t t.c
# ls -V t
-rwxrwx---+  1 root     root        7396 Jun 17 10:50 t
                owner@:rwxpdDaARWcCos:------:allow
                group@:rwxp----------:------:allow
             everyone@:--------------:------:allow
```

cc が要求したアクセス権は 0777 であるため、アクセス権は 0770 になります。その結果、owner@、group@、および everyone@ エントリから実行権が継承されます。

◆ ◆ ◆ 第 9 章

Oracle Solaris ZFS 委任管理

この章では、ZFS 委任管理を使用して、特権のないユーザーが ZFS 管理タスクを実行できるようにする方法について説明します。

この章は、次の節で構成されます。

ZFS 委任管理の概要

ZFS 委任管理を使用すると、細かく調整したアクセス権を、特定のユーザー、グループ、または全員に割り当てることができます。次の 2 種類の委任アクセス権がサポートされています。

- 作成、破棄、マウント、スナップショットといった個別のアクセス権を明示的に委任できます。
- 「アクセス権セット」と呼ばれるアクセス権の集まりを定義できます。アクセス権セットはあとで更新することができ、そのセットの使用者は自動的に変更内容を取得します。アクセス権セットは @ 記号で始まり、64 文字以下の長さに制限されています。@ 記号に続くセット名の残り部分の文字には、通常の ZFS ファイルシステム名と同じ制限事項が適用されます。

ZFS 委任管理では、RBAC セキュリティーモデルに似た機能が提供されます。ZFS 委任を使用すると、ZFS ストレージプールおよびファイルシステムの管理に次のような利点が得られます。

- ZFS ストレージプールの移行時には常にアクセス権も移行されます。

- 動的継承により、ファイルシステム間でアクセス権をどのように伝達するかを制御できます。
- ファイルシステムの作成者だけがそのファイルシステムを破棄できるように設定することができます。
- アクセス権を特定のファイルシステムに委任できます。新しく作成されるファイルシステムは、アクセス権を自動的に取得できます。
- NFS の管理が容易になります。たとえば、明示的なアクセス権を持っているユーザーは、NFS 経由でスナップショットを作成し、適切な `.zfs/snapshot` ディレクトリに保存できます。

委任管理を使用して ZFS タスクを分散することを検討してください。RBAC を使用して一般的な Oracle Solaris 管理タスクを管理する方法については、『Solaris のシステム管理 (セキュリティサービス)』のパート III「役割、権利プロファイル、特権」を参照してください。

ZFS 委任アクセス権を無効にする

委任管理機能を制御するには、プールの `delegation` プロパティーを使用します。次に例を示します。

```
# zpool get delegation users
NAME  PROPERTY    VALUE       SOURCE
users  delegation  on          default
# zpool set delegation=off users
# zpool get delegation users
NAME  PROPERTY    VALUE       SOURCE
users  delegation  off         local
```

デフォルトでは、`delegation` プロパティーは有効になっています。

ZFS アクセス権の委任

`zfs allow` コマンドを使用して、ZFS データセットに対するアクセス権を root 以外のユーザーに次の方法で委任できます。

- 個別のアクセス権をユーザー、グループ、または全員に委任できます。
- 個別のアクセス権の集まりを「アクセス権セット」としてユーザー、グループ、または全員に委任できます。
- アクセス権は、現在のデータセットだけにローカルで委任するか、現在のデータセットのすべての子孫に委任することができます。

次の表では、委任できる操作と、委任された操作の実行に必要な依存するアクセス権について説明します。

アクセス権 (サブコマンド)	説明	依存関係
allow	所有しているアクセス権を別のユーザーに付与するアクセス権。	許可しようとしているアクセス権自体を持っていることも必要です。
クローン	データセットのスナップショットのいずれかを複製するアクセス権。	元のファイルシステムで create アクセス権と mount アクセス権を持っていることも必要です。
create	子孫のデータセットを作成するアクセス権。	mount アクセス権を持っていることも必要です。
destroy	データセットを破棄するアクセス権。	mount アクセス権を持っていることも必要です。
mount	データセットのマウントとマウント解除、およびボリュームのデバイスリンクの作成と破棄を行うアクセス権。	
promote	クローンをデータセットに昇格させるアクセス権。	元のファイルシステムで mount アクセス権と promote アクセス権を持っていることも必要です。
receive	zfs receive コマンドで子孫のファイルシステムを作成するアクセス権。	mount アクセス権と create アクセス権を持っていることも必要です。
rename	データセットの名前を変更するアクセス権。	新しい親で create アクセス権と mount アクセス権を持っていることも必要です。
rollback	スナップショットをロールバックするアクセス権。	
send	スナップショットストリームを送信するアクセス権。	
share	データセットを共有および共有解除するアクセス権。	
snapshot	データセットのスナップショットを作成するアクセス権。	

次の一連のアクセス権を委任できますが、アクセス、読み取り、および変更のアクセス権に限定されることがあります。

- groupquota
- groupused
- userprop
- userquota
- userused

また、次の ZFS プロパティーの管理をルート以外のユーザーに委任できます。

- aclinherit
- aclmode
- atime
- canmount
- casesensitivity
- チェックサム
- compression
- copies
- devices
- exec
- mountpoint
- nbmand
- normalization
- primarycache
- quota
- readonly
- recordsize
- refreservation
- reservation
- secondarycache
- setuid
- shareiscsi
- sharenfs
- sharesmb
- snapdir
- utf8only
- version
- volblocksize
- volsize
- vscan
- xattr
- zoned

これらのプロパティーの一部は、データセットの作成時にのみ設定できます。これらのプロパティーについては、189 ページの「ZFS のプロパティーの紹介」を参照してください。

ZFS アクセス権の委任 (zfs allow)

zfs allow の構文を次に示します。

```
zfs allow -[ldugecs] everyone|user|group[,...] perm|@setname,...] filesystem| volume
```

次の zfs allow 構文 (太字) は、アクセス権の委任先を示しています。

zfs allow [-uge]|user|group|everyone [,...] *filesystem* | *volume*

複数のエンティティーをコンマ区切りのリストとして指定できます。-uge オプションが指定されていない場合、引数はキーワード everyone、ユーザー名、最後にグループ名という優先順位で解釈されます。「everyone」という名前のユーザーまたはグループを指定するには、-u オプションまたは -g オプションを使用します。ユーザーと同じ名前のグループを指定するには、-g オプションを使用します。-c オプションは作成時のアクセス権を委任します。

次の zfs allow 構文 (太字) は、アクセス権およびアクセス権セットの指定方法を示しています。

zfs allow [-s] ... perm|@setname [,...] *filesystem* | *volume*

複数のアクセス権をコンマ区切りのリストとして指定できます。アクセス権の名前は、ZFS のサブコマンドおよびプロパティーと同じです。詳細は、前の節を参照してください。

アクセス権を「アクセス権セット」にまとめ、-s オプションで指定できます。アクセス権セットは、指定のファイルシステムとその子孫に対してほかの zfs allow コマンドで使用できます。アクセス権セットは動的に評価されるため、セットに加えられた変更はすぐに更新されます。アクセス権セットは ZFS ファイルシステムと同じ命名要件に従いますが、名前はアットマーク記号 (@) で始まり、64 文字以下の長さでなければなりません。

次の zfs allow 構文 (太字) は、アクセス権の委任方法を示しています。

zfs allow [-ld] ... ... *filesystem* | *volume*

-l オプションは、アクセス権が指定のデータセットだけに許可されることを示します。-d オプションも指定されている場合を除き、子孫には許可されません。-d オプションは、アクセス権が子孫のデータセットだけに許可されることを示します。-l オプションも指定されている場合を除き、このデータセットには許可されません。どちらのオプションも指定されていない場合は、ファイルシステムまたはボリュームおよびそのすべての子孫にアクセス権が許可されます。

ZFS 委任アクセス権を削除する (zfs unallow)

以前に委任したアクセス権を zfs unallow コマンドで削除できます。

たとえば、create、destroy、mount、および snapshot アクセス権を次のように委任したとします。

```
# zfs allow cindys create,destroy,mount,snapshot tank/cindys
# zfs allow tank/cindys
-------------------------------------------------------------
```

```
Local+Descendent permissions on (tank/cindys)
        user cindys create,destroy,mount,snapshot
--------------------------------------------------------------
```

これらのアクセス権を削除するには、次の構文を使用します。

```
# zfs unallow cindys tank/cindys
# zfs allow tank/cindys
```

ZFS アクセス権を委任する (例)

例 9–1 個別のユーザーにアクセス権を委任する

create アクセス権と mount アクセス権を個別のユーザーに委任する場合は、そのユーザーが配下のマウントポイントに対するアクセス権を持っていることを確認する必要があります。

たとえば、ユーザー marks に create アクセス権と mount アクセス権を tank ファイルシステムに関して委任するには、まず次のようにアクセス権を設定します。

```
# chmod A+user:marks:add_subdirectory:fd:allow /tank
```

その後、zfs allow コマンドを使用して create、destroy、および mount アクセス権を委任します。次に例を示します。

```
# zfs allow marks create,destroy,mount tank
```

これで、ユーザー marks は tank ファイルシステム内に自分のファイルシステムを作成できるようになります。次に例を示します。

```
# su marks
marks$ zfs create tank/marks
marks$ ^D
# su lp
$ zfs create tank/lp
cannot create 'tank/lp': permission denied
```

例 9–2 グループに create および destroy アクセス権を委任する

次の例では、ファイルシステムを設定して、staff グループの任意のメンバーが tank ファイルシステムでファイルシステムの作成とマウント、および各自のファイルシステムの破棄を実行できるようにする方法を示します。ただし、staff グループのメンバーであっても、ほかのメンバーのファイルシステムを破棄することはできません。

```
# zfs allow staff create,mount tank
# zfs allow -c create,destroy tank
# zfs allow tank
--------------------------------------------------------------
```

例 9–2 グループに create および destroy アクセス権を委任する (続き)

```
Create time permissions on (tank)
        create,destroy
Local+Descendent permissions on (tank)
        group staff create,mount
-------------------------------------------------------------
# su cindys
cindys% zfs create tank/cindys
cindys% exit
# su marks
marks% zfs create tank/marks/data
marks% exit
cindys% zfs destroy tank/marks/data
cannot destroy 'tank/mark': permission denied
```

例 9–3 正しいファイルシステムレベルでアクセス権を委任する

ユーザーにアクセス権を委任する場合は、必ず正しいファイルシステムレベルで委任してください。たとえば、ユーザー marks には create、destroy、および mount アクセス権が、ローカルおよび子孫のファイルシステムに関して委任されています。ユーザー marks には tank ファイルシステムのスナップショットを作成するローカルアクセス権が委任されていますが、自分のファイルシステムのスナップショットを作成することは許可されていません。したがって、このユーザーには snapshot アクセス権が正しいファイルシステムレベルで委任されていません。

```
# zfs allow -l marks snapshot tank
# zfs allow tank
-------------------------------------------------------------
Local permissions on (tank)
        user marks snapshot
Local+Descendent permissions on (tank)
        user marks create,destroy,mount
-------------------------------------------------------------
# su marks
marks$ zfs snapshot tank/@snap1
marks$ zfs snapshot tank/marks@snap1
cannot create snapshot 'mark/marks@snap1': permission denied
```

ユーザー marks に子孫ファイルシステムレベルのアクセス権を委任するには、zfs allow -d オプションを使用します。次に例を示します。

```
# zfs unallow -l marks snapshot tank
# zfs allow -d marks snapshot tank
# zfs allow tank
-------------------------------------------------------------
Descendent permissions on (tank)
        user marks snapshot
Local+Descendent permissions on (tank)
        user marks create,destroy,mount
-------------------------------------------------------------
# su marks
$ zfs snapshot tank@snap2
cannot create snapshot 'tank@snap2': permission denied
$ zfs snapshot tank/marks@snappy
```

例 9–3 正しいファイルシステムレベルでアクセス権を委任する (続き)

これで、ユーザー marks は tank ファイルシステムレベルの下のスナップショットだけを作成できるようになります。

例 9–4 複雑な委任アクセス権を定義して使用する

特定のアクセス権をユーザーやグループに委任できます。たとえば、次の zfs allow コマンドでは、特定のアクセス権が staff グループに委任されます。また、destroy アクセス権と snapshot アクセス権が tank ファイルシステムの作成後に委任されます。

```
# zfs allow staff create,mount tank
# zfs allow -c destroy,snapshot tank
# zfs allow tank
-------------------------------------------------------------
Create time permissions on (tank)
        destroy,snapshot
Local+Descendent permissions on (tank)
        group staff create,mount
-------------------------------------------------------------
```

ユーザー marks は staff グループのメンバーなので、tank 内にファイルシステムを作成できます。また、ユーザー marks は、tank/marks2 のスナップショットを作成するための特定のアクセス権を持っているため、そのようなスナップショットを作成できます。次に例を示します。

```
# su marks
$ zfs create tank/marks2
$ zfs allow tank/marks2
-------------------------------------------------------------
Local permissions on (tank/marks2)
        user marks destroy,snapshot
-------------------------------------------------------------
Create time permissions on (tank)
        destroy,snapshot
Local+Descendent permissions on (tank)
        group staff create
        everyone mount
-------------------------------------------------------------
```

ただし、ユーザー marks は tank/marks でスナップショットを作成するための特定のアクセス権を持っていないため、そのようなスナップショットは作成できません。次に例を示します。

```
$ zfs snapshot tank/marks2@snap1
$ zfs snapshot tank/marks@snappp
cannot create snapshot 'tank/marks@snappp': permission denied
```

この例では、ユーザー marks は自身のホームディレクトリで create アクセス権を持っていますが、これは、このユーザーがスナップショットを作成できることを意味します。このシナリオは、ファイルシステムを NFS マウントする場合に役立ちます。

例 9–4 複雑な委任アクセス権を定義して使用する (続き)

```
$ cd /tank/marks2
$ ls
$ cd .zfs
$ ls
snapshot
$ cd snapshot
$ ls -l
total 3
drwxr-xr-x   2 marks    staff          2 Dec 15 13:53 snap1
$ pwd
/tank/marks2/.zfs/snapshot
$ mkdir snap2
$ zfs list
NAME                   USED  AVAIL  REFER  MOUNTPOINT
tank                   264K  33.2G  33.5K  /tank
tank/marks            24.5K  33.2G  24.5K  /tank/marks
tank/marks2             46K  33.2G  24.5K  /tank/marks2
tank/marks2@snap1     21.5K      -  24.5K  -
tank/marks2@snap2         0      -  24.5K  -
$ ls
snap1  snap2
$ rmdir snap2
$ ls
snap1
```

例 9–5 ZFS 委任アクセス権セットを定義して使用する

次の例では、アクセス権セット @myset を作成し、グループ staff にこのアクセス権セットと rename アクセス権を tank ファイルシステムに関して委任する方法を示します。ユーザー cindys は staff グループのメンバーであり、tank にファイルシステムを作成するアクセス権を持っています。ただし、ユーザー lp は tank にファイルシステムを作成するアクセス権を持っていません。

```
# zfs allow -s @myset create,destroy,mount,snapshot,promote,clone,readonly tank
# zfs allow tank
-------------------------------------------------------------
Permission sets on (tank)
        @myset clone,create,destroy,mount,promote,readonly,snapshot
-------------------------------------------------------------
# zfs allow staff @myset,rename tank
# zfs allow tank
-------------------------------------------------------------
Permission sets on (tank)
        @myset clone,create,destroy,mount,promote,readonly,snapshot
Local+Descendent permissions on (tank)
        group staff @myset,rename
# chmod A+group:staff:add_subdirectory:fd:allow tank
# su cindys
cindys% zfs create tank/data
Cindys% zfs allow tank
-------------------------------------------------------------
Permission sets on (tank)
        @myset clone,create,destroy,mount,promote,readonly,snapshot
Local+Descendent permissions on (tank)
```

例 9–5 ZFS 委任アクセス権セットを定義して使用する (続き)

```
        group staff @myset,rename
------------------------------------------------------------
cindys% ls -l /tank
total 15
drwxr-xr-x   2 cindys   staff          2 Aug  8 14:10 data
cindys% exit
# su lp
$ zfs create tank/lp
cannot create 'tank/lp': permission denied
```

ZFS 委任アクセス権を表示する (例)

次のコマンドを使用して、アクセス権を表示できます。

 # zfs allow *dataset*

このコマンドでは、指定されたデータセットに設定または許可されているアクセス権が表示されます。出力には、次の構成要素が含まれています。

- アクセス権セット
- 個々のアクセス権または作成時のアクセス権
- ローカルのデータセット
- ローカルおよび子孫のデータセット
- 子孫のデータセットのみ

例 9–6 基本的な委任管理アクセス権を表示する

次の出力は、ユーザー cindys が tank/cindys ファイルシステムに対して create、destroy、mount、snapshot のアクセス権を持っていることを示しています。

```
# zfs allow tank/cindys
       -------------------------------------------------------------
       Local+Descendent permissions on (tank/cindys)
               user cindys create,destroy,mount,snapshot
```

例 9–7 複雑な委任管理アクセス権を表示する

次の例の出力は、pool/fred ファイルシステムと pool ファイルシステムに対する次のようなアクセス権を示しています。

pool/fred ファイルシステムに対しては次のとおりです。

- 次の 2 つのアクセス権セットが定義されています。
  - @eng (create, destroy , snapshot, mount, clone , promote, rename)
  - @simple (create, mount)

例 9–7 複雑な委任管理アクセス権を表示する (続き)

- 作成時のアクセス権が @eng アクセス権セットと mountpoint プロパティーに対して設定されています。この作成時のアクセス権により、データセットが作成されたあとで @eng アクセス権セットと mountpoint プロパティーを設定するアクセス権が委任されます。
- ユーザー tom には @eng アクセス権セット、ユーザー joe には create、destroy、および mount アクセス権が、ローカルファイルシステムに関して委任されています。
- ユーザー fred には @basic アクセス権セットと share および rename アクセス権が、ローカルおよび子孫のファイルシステムに関して委任されています。
- ユーザー barney と staff グループには @basic アクセス権セットが、子孫のファイルシステムに関してのみ委任されています。

pool ファイルシステムに対しては次のとおりです。

- アクセス権セット @simple (create、destroy、mount) が定義されています。
- グループ staff には @simple アクセス権セットが、ローカルファイルシステムに関して付与されています。

この例の出力を次に示します。

```
$ zfs allow pool/fred
-------------------------------------------------------------------------------
Permission sets on (pool/fred)
        @eng create,destroy,snapshot,mount,clone,promote,rename
        @simple create,mount
Create time permissions on (pool/fred)
        @eng,mountpoint
Local permissions on (pool/fred)
        user tom @eng
        user joe create,destroy,mount
Local+Descendent permissions on (pool/fred)
        user fred @basic,share,rename
Descendent permissions on (pool/fred)
        user barney @basic
        group staff @basic
-------------------------------------------------------------------------------
Permission sets on (pool)
        @simple create,destroy,mount
Local permissions on (pool)
        group staff @simple
-------------------------------------------------------------------------------
```

委任された ZFS アクセス権を削除する (例)

zfs unallow コマンドを使用して、委任したアクセス権を削除できます。たとえば、ユーザー cindys は tank/cindys ファイルシステムで作成、破棄、マウント、およびスナップショット作成を行うアクセス権を持っています。

```
# zfs allow cindys create,destroy,mount,snapshot tank/cindys
       # zfs allow tank/cindys
       -------------------------------------------------------------
       Local+Descendent permissions on (tank/cindys)
               user cindys create,destroy,mount,snapshot
       -------------------------------------------------------------
```

次の zfs unallow 構文では、ユーザー cindys の snapshot アクセス権が tank/cindys ファイルシステムから削除されます。

```
# zfs unallow cindys snapshot tank/cindys
# zfs allow tank/cindys
-------------------------------------------------------------
Local+Descendent permissions on (tank/cindys)
        user cindys create,destroy,mount
-------------------------------------------------------------
cindys% zfs create tank/cindys/data
cindys% zfs snapshot tank/cindys@today
cannot create snapshot 'tank/cindys@today': permission denied
```

別の例として、ユーザー marks は tank/marks ファイルシステムで次のアクセス権を持っています。

```
# zfs allow tank/marks
-------------------------------------------------------------
Local+Descendent permissions on (tank/marks)
        user marks create,destroy,mount
-------------------------------------------------------------
```

次の zfs unallow 構文を使用すると、ユーザー marks のすべてのアクセス権が tank/marks ファイルシステムから削除されます。

```
# zfs unallow marks tank/marks
```

次の zfs unallow 構文では、tank ファイルシステムのアクセス権セットが削除されます。

```
# zfs allow tank
-------------------------------------------------------------
Permission sets on (tank)
        @myset clone,create,destroy,mount,promote,readonly,snapshot
Create time permissions on (tank)
        create,destroy,mount
Local+Descendent permissions on (tank)
        group staff create,mount
-------------------------------------------------------------
# zfs unallow -s @myset tank
```

```
$ zfs allow tank
-------------------------------------------------------------
Create time permissions on (tank)
        create,destroy,mount
Local+Descendent permissions on (tank)
        group staff create,mount
-------------------------------------------------------------
```

◆ ◆ ◆ 第 10 章

Oracle Solaris ZFS の高度なトピック

この章では、ZFS ボリューム、ゾーンがインストールされた Solaris システムで ZFS を使用する方法、ZFS 代替ルートプール、および ZFS 権利プロファイルについて説明します。

この章は、次の節で構成されます。

ZFS ボリューム

ZFS ボリュームとは、ブロックデバイスを表すデータセットです。ZFS ボリュームは、`/dev/zvol/{dsk,rdsk}/pool` ディレクトリのデバイスとして識別されます。

次の例では、5G バイトの ZFS ボリューム `tank/vol` が作成されます。

```
# zfs create -V 5gb tank/vol
```

ボリュームの作成時には、予期しない動作が発生しないよう、予約が自動的にボリュームの初期サイズに設定されます。たとえば、ボリュームのサイズを縮小すると、データが破壊される可能性があります。ボリュームのサイズを変更するときは、注意深く行う必要があります。

また、サイズが変化するボリュームのスナップショットを作成する場合は、スナップショットをロールバックしたり、スナップショットからのクローンを作成しようとすると、不一致が発生する可能性があります。

ボリュームに適用可能なファイルシステムプロパティーについては、表 6–1 を参照してください。

ゾーンがインストールされた Solaris システムを使用している場合は、非大域ゾーンの中で ZFS ボリュームを作成または複製することはできません。そうしようとしても失敗します。ZFS ボリュームを大域ゾーンで使用する方法については、290 ページの「ZFS ボリュームを非大域ゾーンに追加する」を参照してください。

ZFS ボリュームをスワップデバイスまたはダンプデバイスとして使用する

ZFS ルートファイルシステムをインストールするとき、または UFS ルートファイルシステムから移行するときに、ZFS ルートプールの ZFS ボリュームにスワップデバイスが作成されます。次に例を示します。

```
# swap -l
swapfile                  dev    swaplo   blocks     free
/dev/zvol/dsk/rpool/swap 253,3        16  8257520  8257520
```

ZFS ルートファイルシステムをインストールするとき、または UFS ルートファイルシステムから移行するときに、ZFS ルートプールの ZFS ボリュームにダンプデバイスが作成されます。ダンプデバイスを設定したあとは、ダンプデバイスの管理は不要です。次に例を示します。

```
# dumpadm
      Dump content: kernel pages
       Dump device: /dev/zvol/dsk/rpool/dump (dedicated)
Savecore directory: /var/crash/t2000
  Savecore enabled: yes
```

システムのインストール後またはアップグレード後にスワップ領域やダンプデバイスを変更する必要がある場合は、以前の Solaris 10 リリースと同様に `swap` コマンドと `dumpadm` コマンドを使用します。追加のスワップボリュームを作成する必要がある場合は、特定のサイズの ZFS ボリュームを作成してから、そのデバイスでスワップを有効にします。次に例を示します。

```
# zfs create -V 2G rpool/swap2
# swap -a /dev/zvol/dsk/rpool/swap2
# swap -l
swapfile                   dev  swaplo blocks   free
/dev/zvol/dsk/rpool/swap  256,1      16 2097136 2097136
/dev/zvol/dsk/rpool/swap2 256,5      16 4194288 4194288
```

ZFS ファイルシステム上のファイルには、スワップしないでください。ZFS スワップファイルの構成はサポートされていません。

スワップボリュームとダンプボリュームのサイズの調整については、167 ページの「ZFS スワップデバイスおよびダンプデバイスのサイズを調整する」を参照してください。

ZFS ボリュームを Solaris iSCSI ターゲットとして使用する

ボリュームに shareiscsi プロパティーを設定すれば、簡単に ZFS ボリュームを iSCSI ターゲットとして作成できます。次に例を示します。

```
# zfs create -V 2g tank/volumes/v2
# zfs set shareiscsi=on tank/volumes/v2
# iscsitadm list target
Target: tank/volumes/v2
    iSCSI Name: iqn.1986-03.com.sun:02:984fe301-c412-ccc1-cc80-cf9a72aa062a
    Connections: 0
```

iSCSI ターゲットが作成されたら、iSCSI イニシエータを設定します。Solaris iSCSI ターゲットおよびイニシエータの詳細については、『Solaris のシステム管理 (デバイスとファイルシステム)』の第 14 章「Oracle Solaris iSCSI ターゲットおよびイニシエータの構成 (手順)」を参照してください。

注 – また、Solaris iSCSI ターゲットは、iscsitadm コマンドを使って作成および管理することもできます。ZFS ボリュームに shareiscsi プロパティーを設定した場合は、iscsitadm コマンドを使用して同じターゲットデバイスをまた作成しないでください。そうしないと、同じデバイスに対して重複したターゲット情報が作成されてしまいます。

---

iSCSI ターゲットとしての ZFS ボリュームは、ほかの ZFS データセットとまったく同じように管理されます。ただし、iSCSI ターゲットでは、名前の変更、エクスポート、およびインポートの操作が少し異なります。

- ZFS ボリュームの名前を変更しても、iSCSI ターゲットの名前は変わりません。次に例を示します。

  ```
  # zfs rename tank/volumes/v2 tank/volumes/v1
  # iscsitadm list target
  Target: tank/volumes/v1
      iSCSI Name: iqn.1986-03.com.sun:02:984fe301-c412-ccc1-cc80-cf9a72aa062a
      Connections: 0
  ```

- 共有 ZFS ボリュームが含まれるプールをエクスポートすると、ターゲットが削除されます。共有 ZFS ボリュームが含まれるプールをインポートすると、ターゲットが共有されます。次に例を示します。

  ```
  # zpool export tank
  # iscsitadm list target
  # zpool import tank
  # iscsitadm list target
  Target: tank/volumes/v1
      iSCSI Name: iqn.1986-03.com.sun:02:984fe301-c412-ccc1-cc80-cf9a72aa062a
      Connections: 0
  ```

iSCSI ターゲットの構成情報はすべてデータセット内に格納されます。NFS 共有ファイルシステムと同様に、別のシステム上にインポートされる iSCSI ターゲットは正しく共有されます。

ゾーンがインストールされている Solaris システムで ZFS を使用する

以降の節では、Oracle Solaris ゾーンを備えたシステムで ZFS を使用する方法について説明します。

- 289 ページの「ZFS ファイルシステムを非大域ゾーンに追加する」
- 290 ページの「データセットを非大域ゾーンに委任する」
- 290 ページの「ZFS ボリュームを非大域ゾーンに追加する」
- 291 ページの「ZFS ストレージプールをゾーンで使用する」
- 291 ページの「ZFS プロパティーをゾーンで管理する」
- 292 ページの「zoned プロパティーについて」

ZFS ルートファイルシステムがインストールされたシステムにゾーンを構成し、Oracle Solaris Live Upgrade で移行やパッチの適用を行う方法については、151 ページの「ゾーンが含まれているシステムを Oracle Solaris Live Upgrade で移行またはアップグレードする (Solaris 10 10/08)」または 156 ページの「ゾーンが含まれているシステムを Oracle Solaris Live Upgrade で移行またはアップグレードする (Solaris 10 5/09 以降)」を参照してください。

ZFS データセットをゾーンに関連付けるときは、次の点に留意してください。

- ZFS ファイルシステムまたは ZFS クローンを非大域ゾーンに追加できますが、管理者制御を委任しても委任しなくてもかまいません。
- ZFS ボリュームをデバイスとして非大域ゾーンに追加できます。
- この時点で、ZFS スナップショットをゾーンに関連付けることはできません。

以降の節では、ZFS データセットはファイルシステムまたはクローンを指します。

データセットを追加すると、非大域ゾーンは大域ゾーンとディスク領域を共有できます。ただし、ゾーン管理者は、配下のファイルシステム階層でプロパティーを制御したり、新しいファイルシステムを作成したりすることはできません。この動作は、ほかの種類のファイルシステムをゾーンに追加する場合と同じであり、共通のディスク領域を共有することが目的の場合にのみ使用してください。

ZFS では、データセットを非大域ゾーンに委任して、データセットとそのすべての子を完全に制御する権限をゾーン管理者に渡すこともできます。ゾーン管理者は、そのデータセット内でファイルシステムやクローンを作成および破棄したり、データセットのプロパティーを変更したりできます。ゾーン管理者は、委任されたデータセットに設定された最上位の割り当て制限を超過するなど、ゾーンに追加されていないデータセットに影響を与えることはできません。

Oracle Solaris ゾーンがインストールされたシステム上で ZFS を操作する場合には、次の点を考慮してください。

- 非大域ゾーンに追加する ZFS ファイルシステムでは、`mountpoint` プロパティーを `legacy` に設定する必要があります。
- CR 6449301 のため、非大域ゾーンが構成されている場合は、非大域ゾーンに ZFS データセットを追加しないでください。代わりに、ゾーンのインストール後に ZFS データセットを追加してください。
- ソース `zonepath` とターゲット `zonepath` がどちらも ZFS ファイルシステム上に存在し、同じプール内にある場合、`zoneadm clone` は自動的に ZFS クローンを使ってゾーンを複製するようになりました。`zoneadm clone` コマンドは、ソース `zonepath` の ZFS スナップショットを作成し、ターゲット `zonepath` を設定します。`zfs clone` コマンドを使用してゾーンを複製することはできません。詳細は、『Oracle Solaris のシステム管理 (Oracle Solaris コンテナ : 資源管理と Oracle Solaris ゾーン)』のパート II「ゾーン」を参照してください。
- ZFS ファイルシステムを非大域ゾーンに委任する場合は、Oracle Solaris Live Upgrade を使用する前にそのファイルシステムを該当の非大域ゾーンから削除する必要があります。削除しないと、読み取り専用ファイルシステムエラーのため、Oracle Live Upgrade が失敗します。

ZFS ファイルシステムを非大域ゾーンに追加する

大域ゾーンと領域を共有する必要がある場合は、ZFS ファイルシステムを汎用のファイルシステムとして追加して、その目的のためだけに使用できます。 非大域ゾーンに追加する ZFS ファイルシステムでは、`mountpoint` プロパティーを `legacy` に設定する必要があります。

`zonecfg` コマンドの `add fs` サブコマンドを使用することで、ZFS ファイルシステムを非大域ゾーンに追加できます。

次の例では、大域ゾーンの大域ゾーン管理者が、ZFS ファイルシステムを非大域ゾーンに追加しています。

```
# zonecfg -z zion
zonecfg:zion> add fs
zonecfg:zion:fs> set type=zfs
zonecfg:zion:fs> set special=tank/zone/zion
zonecfg:zion:fs> set dir=/export/shared
zonecfg:zion:fs> end
```

この構文では、ZFS ファイルシステム `tank/zone/zion` がすでに構成済みの `zion` ゾーンに追加され、`/export/shared` にマウントされます。ファイルシステムの `mountpoint` プロパティーは、`legacy` に設定する必要があります。別の場所にすでにマウントされているファイルシステムは追加できません。ゾーン管理者は、ファイルシステム内でファイルを作成および破棄することができます。ファイルシステム

を別の場所に再マウントすることはできません。また、ゾーン管理者がファイルシステムのプロパティー (atime、readonly、compression など) を変更することもできません。大域ゾーン管理者は、ファイルシステムのプロパティーの設定および制御を担当します。

zonecfg コマンドの詳細および zonecfg を使用したリソースタイプの設定の詳細については、『Oracle Solaris のシステム管理 (Oracle Solaris コンテナ : 資源管理と Oracle Solaris ゾーン)』のパート II「ゾーン」を参照してください。

データセットを非大域ゾーンに委任する

ストレージの管理をゾーンに委任するという主要目的を果たせるよう、ZFS では、zonecfg コマンドの add dataset サブコマンドを使用してデータセットを非大域ゾーンに追加することができます。

次の例では、大域ゾーンの大域ゾーン管理者が、ZFS ファイルシステムを非大域ゾーンに委任しています。

```
# zonecfg -z zion
zonecfg:zion> add dataset
zonecfg:zion:dataset> set name=tank/zone/zion
zonecfg:zion:dataset> end
```

ファイルシステムを追加する場合と異なり、この構文を実行すると、ZFS ファイルシステム tank/zone/zion がすでに構成済みの zion ゾーンから見えるようになります。ゾーン管理者は、ファイルシステムのプロパティーを設定したり、子孫ファイルシステムを作成したりできます。また、ゾーン管理者は、スナップショットやクローンを作成し、およびファイルシステム階層全体を制御することができます。

Oracle Solaris Live Upgrade を使って非大域ゾーンを含む ZFS BE をアップグレードする場合には、まず委任されたデータセットをすべて削除してください。削除しないと、読み取り専用ファイルシステムエラーで Oracle Solaris Live Upgrade が失敗します。次に例を示します。

```
zonecfg:zion>
zonecfg:zion> remove dataset name=tank/zone/zion
zonecfg:zion1> exit
```

ゾーンでどのような操作が許可されるかの詳細については、291 ページの「ZFS プロパティーをゾーンで管理する」を参照してください。

ZFS ボリュームを非大域ゾーンに追加する

zonecfg コマンドの add dataset サブコマンドを使用して、ZFS ボリュームを非大域ゾーンに追加することはできません。ただし、zonecfg コマンドの add device サブコマンドを使って、ボリュームをゾーンに追加することはできます。

次の例では、大域ゾーンの大域ゾーン管理者が、ZFS ボリュームを非大域ゾーンに追加しています。

```
# zonecfg -z zion
zion: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:zion> create
zonecfg:zion> add device
zonecfg:zion:device> set match=/dev/zvol/dsk/tank/vol
zonecfg:zion:device> end
```

この構文では、tank/vol ボリュームが zion ゾーンに追加されます。raw ボリュームをゾーンに追加する操作は、そのボリュームが物理ディスクに対応していない場合でも、潜在的なセキュリティー上の危険を伴います。特に、ゾーン管理者が作成したファイルシステムの形式が正しくない場合には、マウントしようとするときにファイルシステムでパニックが発生します。デバイスをゾーンに追加することおよびそれに関連するセキュリティー上の危険の詳細については、292 ページの「zoned プロパティーについて」を参照してください。

デバイスをゾーンに追加する方法の詳細については、『Oracle Solaris のシステム管理 (Oracle Solaris コンテナ : 資源管理と Oracle Solaris ゾーン)』のパート II「ゾーン」を参照してください。

ZFS ストレージプールをゾーンで使用する

ZFS ストレージプールをゾーンの内部で作成または変更することはできません。委任管理モデルを使用することで、大域ゾーン内の物理ストレージデバイスの制御と仮想ストレージの制御をすべて非大域ゾーンで行うことができます。プールレベルのデータセットをゾーンに追加することはできますが、デバイスを作成したり、追加したり、削除したりするなど、プールの物理特性を変更するコマンドはゾーンの内部から実行することはできません。zonecfg コマンドの add device サブコマンドを使用して物理デバイスをゾーンに追加する場合でも、ファイルを使用する場合でも、zpool コマンドを使用してゾーンの内部に新しいプールを作成することはできません。

ZFS プロパティーをゾーンで管理する

データセットをゾーンに委任したあとで、ゾーン管理者は特定のデータセットプロパティーを制御できます。ゾーンに委任したデータセットのすべての祖先は、読み取り専用データセットとして表示されます。ただし、データセット自体およびそのすべての子孫は書き込み可能です。たとえば、次のような構成を考えてみます。

```
global# zfs list -Ho name
tank
tank/home
```

```
tank/data
tank/data/matrix
tank/data/zion
tank/data/zion/home
```

tank/data/zion をゾーンに追加した場合には、各データセットのプロパティーは次のようになります。

データセット	表示可能	書き込み可能	不変のプロパティー
tank	はい	いいえ	-
tank/home	いいえ	-	-
tank/data	はい	いいえ	-
tank/data/matrix	いいえ	-	-
tank/data/zion	はい	はい	sharenfs、zoned、quota、reservation
tank/data/zion/home	はい	はい	sharenfs、zoned

tank/zone/zion のすべての親は読み取り専用として表示され、すべての子孫は書き込み可能になり、親階層に含まれないデータセットは完全に非表示になります。非大域ゾーンは NFS サーバーとして動作できないため、ゾーン管理者が sharenfs プロパティーを変更することはできません。次の節で説明するように、zoned プロパティーを変更するとセキュリティー上の危険にさらされるため、ゾーン管理者はこの操作を行えません。

ゾーンの特権ユーザーは、その他の設定可能なプロパティーはすべて変更できます。ただし、quota プロパティーと reservation プロパティーは除きます。大域ゾーン管理者は、この動作を利用して、非大域ゾーンで使用されるすべてのデータセットが使用するディスク容量を制御できます。

また、データセットを非大域ゾーンに委任したあとに、大域ゾーン管理者が sharenfs および mountpoint プロパティーを変更することもできません。

zoned プロパティーについて

データセットを非大域ゾーンに委任するときに、特定のプロパティーが大域ゾーンのコンテキストで解釈されないように、データセットに特別な設定を行う必要があります。データセットが非大域ゾーンに委任され、ゾーン管理者の制御下に入ると、その内容は信頼できる状態ではなくなります。どのファイルシステムにも該当することですが、setuid バイナリやシンボリックリンクなどの安全性に問題のある内容が含まれていることがあります。これらは、大域ゾーンのセキュリティーを低下させる可能性があります。また、mountpoint プロパティーは、大域ゾーンのコン

テキストでは解釈できません。さらに、ゾーン管理者が大域ゾーンの名前空間を操作してしまう可能性もあります。後者の問題に対処するために、ZFS では zoned プロパティーを使って、データセットがある時点で非大域ゾーンに委任されていることを示しています。

zoned プロパティーはブール値で、ZFS データセットを含むゾーンが最初に起動するときに自動的にオンに設定されます。ゾーン管理者が、このプロパティーを手動でオンに設定する必要はありません。zoned プロパティーを設定した場合、そのデータセットを大域ゾーンでマウントしたり共有したりすることはできません。次の例では、tank/zone/zion はゾーンに委任されていますが、tank/zone/global は追加されていません。

```
# zfs list -o name,zoned,mountpoint -r tank/zone
NAME                  ZONED  MOUNTPOINT
tank/zone/global        off  /tank/zone/global
tank/zone/zion           on  /tank/zone/zion
# zfs mount
tank/zone/global           /tank/zone/global
tank/zone/zion             /export/zone/zion/root/tank/zone/zion
```

mountpoint プロパティーと、tank/zone/zion データセットが現在マウントされているディレクトリとが異なっていることに注意してください。mountpoint プロパティーには、データセットがシステム上で現在マウントされている場所ではなく、ディスクに格納されているプロパティーが反映されます。

データセットがゾーンから削除されたり、ゾーンが破棄されたりした場合でも、zoned プロパティーが自動的に消去されることはありません。これらの操作に関連するセキュリティー上の危険が潜在的に存在するために、このような動作になっています。信頼されないユーザーがデータセットとその子孫へのアクセスを完了してしまっているので、mountpoint プロパティーが不正な値に設定されたり、ファイルシステムに setuid バイナリが存在したりする可能性があります。

意図しないセキュリティー上の危険を防ぐために、データセットをなんらかの方法で再利用する場合には、大域ゾーン管理者が zoned プロパティーを手動で消去する必要があります。zoned プロパティーを off に設定する前に、データセットおよびそのすべての子孫の mountpoint プロパティーが適切な値に設定されていること、および setuid バイナリが存在しないことを確認するか、または setuid プロパティーを無効に設定します。

セキュリティーが脆弱なままでないことを確認したあとで、zfs set または zfs inherit コマンドを使用して zoned プロパティーをオフに設定できます。データセットがゾーンで使用されているときに zoned プロパティーをオフに設定すると、システムが予期しない動作をする可能性があります。このプロパティーを変更するのは、データセットが非大域ゾーンで使用されていないことを確認した場合にのみ行なってください。

ZFS 代替ルートプールを使用する

プールが作成されると、そのプールはデフォルトでホストシステムに関連付けられます。ホストシステムでは、プールに関する情報を管理しているので、プールが使用できなくなったときにそのことを自動的に検出することができます。この情報は、通常の操作では有効な情報ですが、代替メディアから起動するときまたはリムーバブルメディアにプールを作成するときには障害になることがあります。この問題を解決するために、ZFS には「代替ルート」プール機能が用意されています。代替ルートプールは、システムの再起動後には有効でなくなり、すべてのマウントポイントはプールのルートへの相対パスに変更されます。

ZFS 代替ルートプールを作成する

代替ルートプールを作成する理由としてもっとも一般的なのは、リムーバブルメディアでの使用です。このような場合には、必要なファイルシステムは通常 1 つだけなので、ターゲットシステムでユーザーが選択した場所にマウントする必要があります。`zpool create -R` オプションを使用して代替ルートプールを作成すると、ルートファイルシステムのマウントポイントは代替ルート値と同じ `/` に自動的に設定されます。

次の例では、`morpheus` という名前のプールが 代替ルートパスとしての `/mnt` に作成されます。

```
# zpool create -R /mnt morpheus c0t0d0
# zfs list morpheus
NAME                    USED  AVAIL  REFER  MOUNTPOINT
morpheus               32.5K  33.5G     8K  /mnt
```

ファイルシステムが 1 つだけで (`morpheus`)、そのマウントポイントがプールの代替ルート `/mnt` であることに注意してください。ディスクに格納されているマウントポイントは、実際に `/` になっています。`/mnt` のフルパスは、プール作成のこの初期コンテキストでのみ解釈されます。その後、このファイルシステムをエクスポートし、それを別のシステム上の任意の代替ルートプールの下で、`-R` *alternate root value* 構文を使ってインポートすることができます。

```
# zpool export morpheus
# zpool import morpheus
cannot mount '/': directory is not empty
# zpool export morpheus
# zpool import -R /mnt morpheus
# zfs list morpheus
NAME                    USED  AVAIL  REFER  MOUNTPOINT
morpheus               32.5K  33.5G     8K  /mnt
```

代替ルートプールをインポートする

代替ルートを使って、プールをインポートすることもできます。この機能は、回復を行う状況で利用できます。つまり、マウントポイントを現在のルートのコンテキストではなく、修復を実行できるように一時的なディレクトリとして解釈するような状況で利用できます。前節で説明したように、この機能はリムーバブルメディアをマウントするときにも使用できます。

次の例では、morpheus という名前のプールが代替ルートパスとしての /mnt にインポートされます。この例では、morpheus がすでにエクスポート済みであることを前提としています。

```
# zpool import -R /a pool
# zpool list morpheus
NAME   SIZE   ALLOC  FREE    CAP  HEALTH  ALTROOT
pool  44.8G    78K  44.7G     0%  ONLINE  /a
# zfs list pool
NAME   USED  AVAIL  REFER  MOUNTPOINT
pool  73.5K  44.1G    21K  /a/pool
```

ZFS 権利プロファイル

スーパーユーザー (root) アカウントを使用しないで ZFS 管理タスクを実行する必要がある場合は、次のいずれかのプロファイルが割り当てられた役割引き受けて ZFS 管理タスクを実行できます。

- ZFS ストレージ管理 – ZFS ストレージプール内でデバイスを作成、破棄、および操作できます
- ZFS ファイルシステム管理 – ZFS ファイルシステムを作成、破棄、および変更できます

役割の作成または割り当ての詳細については、『Solaris のシステム管理 (セキュリティサービス)』を参照してください。

RBAC の役割を使用して ZFS ファイルシステムを管理するほかに、ZFS 委任管理を使用して ZFS 管理タスクを分散することも検討できます。詳細は、第 9 章「Oracle Solaris ZFS 委任管理」を参照してください。

◆ ◆ ◆ 第 11 章

Oracle Solaris ZFS のトラブルシューティングとプールの回復

この章では、ZFS の障害をどのように識別し、そこから回復するかについて説明します。また、障害の発生を防ぐ方法についても説明します。

この章は、次の節で構成されます。

ZFS の障害を識別する

ZFS では、ファイルシステムとボリュームマネージャーが統合されているために、多くの異なる障害が存在します。この章では、さまざまな障害の概要を説明してから、実行しているシステムでそれらをどのように識別するかについて説明します。この章の最後では、問題を修復する方法について説明します。ZFS で発生する可能性がある基本的なエラーには、次の 3 種類があります。

1 つのプールで 3 つのすべてのエラーが発生することもあります。このため、完全な修復作業を行うには、1 つのエラーを検出して訂正したら、次のエラーの対処に進む必要があります。

ZFS ストレージプール内でデバイスが見つからない

デバイスがシステムから完全に削除されると、ZFS はそのデバイスを開けないことを検出し、REMOVED 状態にします。この削除が原因でプール全体が使用できない状態になるかどうかは、そのプールのデータ複製レベルによって決まります。ミラー化されたデバイスまたは RAID-Z デバイスにあるディスクが取り外されても、そのプールには引き続きアクセスできます。プールは FAULTED 状態になる可能性があります。この場合、次の条件のもとでは、デバイスが再接続されるまでどのデータにもアクセスできません。

- ミラーのすべてのコンポーネントが削除される場合
- RAID-Z (raidz1) デバイス内の複数のデバイスが削除される場合
- 単一ディスク構成で最上位レベルのデバイスが削除される場合

ZFS ストレージプール内のデバイスが損傷している

「損傷している」という用語には、発生する可能性のあるさまざまなエラーが含まれます。たとえば、次のようなものがあります。

- ディスクまたはコントローラが不良であるために、一時的な入出力エラーが発生する
- 宇宙線が原因で、ディスク上のデータが破壊される
- ドライバのバグが原因で、間違った場所からデータが転送されたり、間違った場所にデータが転送されたりする
- ユーザーが誤って物理デバイスの一部を上書きしてしまう

これらのエラーは、ある場合には一時的に発生します。たとえば、コントローラに問題があるときは、入出力が無作為にエラーになります。また、ディスク上の破壊のように、損傷が永続することもあります。ただし、損傷が永続的だからといって、そのエラーが再度発生する可能性が高いことには必ずしもなりません。たとえば、管理者が誤ってディスクの一部を上書きしてしまった場合には、ハードウェア障害のようなことは発生していないので、そのデバイスを置き換える必要はありません。デバイスの問題を正確に識別するのは簡単なことではありません。詳細については、あとで説明します。

ZFS データが破壊している

データの破壊が発生するのは、1 つ以上のデバイスエラー (1 つ以上のデバイスが見つからないか、損傷している) が最上位レベルの仮想デバイスに影響するときで

す。たとえば、データは破壊されていないけれども、一方のミラーに大量のデバイスエラーが発生する場合があります。もう一方のミラーの正確に同じ場所にエラーが発生した場合は、データが破壊されたことになります。

データの破壊は常に永続的であり、修復時は特に注意する必要があります。配下のデバイスを修復または置き換えても、元のデータは永久に失われています。このような状況では、ほとんどの場合、バックアップからデータを復元する必要があります。データエラーは発生するたびに記録されます。次の節で説明するように、定期的にプールをスクラブすることでデータエラーを制御できます。破壊されたブロックを削除すると、次のスクラブ処理で破壊が存在しないことが認識され、すべてのエラー追跡がシステムから削除されます。

ZFS ファイルシステムの整合性をチェックする

`fsck` に相当するユーティリティーは、ZFS には存在しません。このユーティリティーは従来から、ファイルシステムの修復と検証という 2 つの目的に利用されてきました。

ファイルシステムの修復

従来のファイルシステムのデータ書き込み方法は、本質的に予期しない障害によってファイルシステムの不一致が発生しやすい性質を持っています。従来のファイルシステムはトランザクション方式ではないので、参照されないブロックや不正なリンクカウントなど、ファイルシステム構造の矛盾が発生する可能性があります。ジャーナリングを導入することでこれらの問題のいくつかは解決されますが、ログをロールバックできないときには別の問題が発生する可能性があります。データの不一致が ZFS 構成内のディスク上で発生するとすれば、それはハードウェア障害が発生した場合か、ZFS ソフトウェアにバグが存在する場合だけです。ただし、ハードウェア障害の場合は、プールに冗長性があるはずです。

`fsck` ユーティリティーは、UFS ファイルシステムに固有の既知の問題を修復します。ZFS ストレージプールの問題の大半は一般に、ハードウェアまたは電源の障害に関連しています。冗長プールを利用することで、多くの問題を回避できます。ハードウェアの障害または電源の停止が原因でプールが損傷している場合は、321 ページの「ZFS ストレージプール全体の損傷を修復する」を参照してください。

プールに冗長性がない場合は、ファイルシステムの破壊によってデータの一部またはすべてにアクセスできなくなるリスクが常に存在します。

ファイルシステムの検証

fsck ユーティリティーには、ファイルシステムの修復を実行する以外に、ディスク上のデータに問題がないことを検証する機能があります。この作業では従来から、ファイルシステムのマウントを解除し、fsck ユーティリティーを実行する必要があります。処理中は、多くのシステムでシングルユーザーモードになります。このシナリオで発生するダウンタイムの長さは、チェックするファイルシステムのサイズに比例します。ZFS では、必要なチェックを実行するためのユーティリティーを明示的に使用する代わりに、すべての不一致を定期的にチェックする機構が用意されています。この機能は「スクラブ」と呼ばれ、メモリーやほかのシステム内で、ハードウェアまたはソフトウェア障害が発生する前にエラーを検出および回避する手段として一般的に使用されます。

ZFS データのスクラブを制御する

スクラブを行なっているときまたは必要なファイルにアクセスしているときにエラーが発生した場合には、そのエラーが内部でログに記録されるので、そのプールで認識されているすべてのエラーの概要をすぐに確認できます。

ZFS データの明示的なスクラブ

データの完全性をもっとも簡単にチェックする方法は、プールに含まれるすべてのデータのスクラブを明示的に開始することです。この処理では、プールに含まれるすべてのデータを 1 回たどってみて、すべてのブロックが読み取り可能であることを確認します。スクラブは、デバイスが実現できる最大速度で進行します。ただし、入出力が発生する場合には、その優先順位は通常の操作よりも低くなります。この操作によって、パフォーマンスが低下することがあります。ただし、スクラブの実行中でも、プールのデータはそのまま使用することができ、応答時間もほとんど変わらないはずです。明示的なスクラブを開始するには、zpool scrub コマンドを使用します。次に例を示します。

```
# zpool scrub tank
```

現在のスクラブ操作の状態は、zpool status コマンドを使用して表示できます。次に例を示します。

```
# zpool status -v tank
  pool: tank
 state: ONLINE
 scrub: scrub completed after 0h7m with 0 errors on Tue Tue Feb  2 12:54:00 2010
config:
        NAME        STATE     READ WRITE CKSUM
        tank        ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c1t0d0  ONLINE       0     0     0
            c1t1d0  ONLINE       0     0     0

errors: No known data errors
```

一度に実行できるスクラブ操作は、各プールで 1 つだけです。

-s オプションを使用すれば、進行中のスクラブ操作を中止できます。次に例を示します。

```
# zpool scrub -s tank
```

ほとんどの場合、データの完全性を保証するスクラブ操作は、完了するまで続けるようにしてください。操作によってシステム性能に影響が出る場合は、ユーザー自身の判断でスクラブ操作を中止してください。

定期的にスクラブを実行すると、システム上のすべてのディスクへの継続的な入出力が保証されます。定期的なスクラブには、電源管理がアイドル状態のディスクを低電力モードにすることができなくなるという副作用があります。システムによる入出力がほとんど常に実行されている場合や、電力消費を気にする必要がない場合には、この問題は無視しても問題ありません。

zpool status の出力の解釈の詳細については、107 ページの「ZFS ストレージプールの状態のクエリー検索を行う」を参照してください。

ZFS データのスクラブと再同期化

デバイスを置き換えると、再同期化処理が開始されて、正常なコピーのデータが新しいデバイスに移動します。この処理は、ディスクのスクラブの一種です。このため、このような処理をプールで実行できるのは、その時点で 1 つだけです。スクラブ処理の実行中に再同期化を実行すると、進行中のスクラブは中断されて、再同期化の完了後に再開されます。

再同期化の詳細については、317 ページの「再同期化の状態を表示する」を参照してください。

ZFS の問題を解決する

ここでは、ZFS ファイルシステムまたはストレージプールで発生する問題を識別して解決する方法について説明します。

次の機能を使用して、ZFS 構成で発生した問題を識別することができます。

- zpool status コマンドを使用すると、ZFS ストレージプールについての詳細な情報を表示できます。
- プールおよびデバイスの障害が ZFS/FMA の診断メッセージで報告されます。
- zpool history コマンドを使用すると、プール状態の情報を変更した以前の ZFS コマンドを表示できます。

ZFS のほとんどのトラブルシューティングで、zpool status コマンドを使用します。このコマンドを実行すると、システム上のさまざまな障害が分析され、もっとも重大な問題が識別されます。さらに、推奨する処置と、詳細情報が掲載されたナレッジ記事へのリンクが提示されます。プールで複数の問題が発生している可能性がある場合でも、このコマンドで識別できる問題は 1 つだけです。たとえば、データ破壊のエラーは一般に、いずれかのデバイスで障害が発生したことを示唆しますが、障害が発生したデバイスを置き換えても、データ破壊の問題がすべて解決するとは限りません。

また、ZFS 診断エンジンはプールの障害とデバイスの障害を診断し、報告します。これらの障害に関連するチェックサム、入出力、デバイス、およびプールのエラーも報告されます。fmd で報告される ZFS 障害は、コンソールとシステムメッセージファイルに表示されます。ほとんどの fmd メッセージで、zpool status コマンドを実行して詳細な回復の手順を確認することを求めています。

基本的な回復方法は次のとおりです。

- 該当する場合、zpool history コマンドを使って、エラーシナリオに至る前に実行された ZFS コマンドを特定します。次に例を示します。

  ```
  # zpool history tank
  History for 'tank':
  2010-07-15.12:06:50 zpool create tank mirror c0t1d0 c0t2d0 c0t3d0
  2010-07-15.12:06:58 zfs create tank/erick
  2010-07-15.12:07:01 zfs set checksum=off tank/erick
  ```

  この出力では、tank/erick ファイルシステムのチェックサムが無効になっています。この構成はお勧めできません。
- システムコンソールまたは /var/adm/messages ファイルに表示される fmd メッセージからエラーを識別します。
- zpool status -x コマンドを使って、詳細な修復手順を確認します。
- 次の手順を実行して、障害を修復します。
  - 障害の発生したデバイスまたは見つからないデバイスを置き換えて、オンラインにします。
  - 障害の発生した構成または破壊されたデータをバックアップから復元します。
  - zpool status -x コマンドを使用して回復を確認します。
  - 復元した構成のバックアップを作成します (該当する場合)。

この節では、発生する可能性がある障害の種類を診断するために、zpool status の出力を解釈する方法について説明します。ほとんどの作業はコマンドによって自動的に実行されますが、障害を診断するうえで、どのような問題が識別されるかを正確に理解しておくことは重要です。以降の節では、発生する可能性のあるさまざまな問題を修復する方法について説明します。

ZFS ストレージプールに問題があるかどうかを確認する

システムになんらかの既知の問題が存在するかどうかを確認するもっとも簡単な方法は、zpool status -x コマンドを使用することです。このコマンドでは、問題が発生しているプールの説明だけが出力されます。健全性に問題があるプールがシステムに存在しない場合、コマンドは次の出力を表示します。

```
# zpool status -x
all pools are healthy
```

-x フラグを指定しないでこのコマンドを実行した場合は、すべてのプールが健全である場合でも、すべてのプール (コマンド行で特定のプールを指定した場合は、要求したプール) のすべての状態が表示されます。

zpool status コマンドのコマンド行オプションの詳細については、107 ページの「ZFS ストレージプールの状態のクエリー検索を行う」を参照してください。

zpool status の出力を確認する

zpool status の完全な出力は次のようになります。

```
# zpool status tank
# zpool status tank
  pool: tank
 state: DEGRADED
status: One or more devices could not be opened.  Sufficient replicas exist for
        the pool to continue functioning in a degraded state.
action: Attach the missing device and online it using 'zpool online'.
   see: http://www.sun.com/msg/ZFS-8000-2Q
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        tank        DEGRADED     0     0     0
          mirror-0  DEGRADED     0     0     0
            c1t0d0  ONLINE       0     0     0
            c1t1d0  UNAVAIL      0     0     0  cannot open

errors: No known data errors
```

この出力の説明を次に示します。

プールの全般的な状態情報

zpool status 出力のこのセクションは、次のフィールドで構成されます。一部の項目は、プールに問題がある場合にのみ表示されます。

pool　　プールの名前を示します。

`state` プールの現在の健全性を示します。この情報は、プールが必要な複製レベルを提供できるかどうかだけを示しています。

`status` プールで発生している問題の説明です。エラーが検出されない場合は、このフィールドは省略されます。

`action` エラーを修復するために推奨される処置。エラーが検出されない場合は、このフィールドは省略されます。

`see` 詳細な修復情報が掲載されているナレッジ記事を紹介します。オンラインの記事はこのガイドよりも頻繁に更新されます。そのため、最新の修復手順については常にオンラインの記事を参照してください。エラーが検出されない場合は、このフィールドは省略されます。

`scrub` スクラブ操作の現在の状態が出力されます。前回のスクラブが完了した日付と時刻、進行中のスクラブ、スクラブが要求されていないかどうかなどが出力されます。

`errors` 既知のデータエラー、または既知のデータエラーが存在しないことが出力されます。

プール構成情報

`zpool status` 出力の `config` フィールドには、プール内のデバイスの構成、デバイスの状態、およびデバイスから生成されたエラーが出力されます。次のいずれかの状態になる可能性があります。`ONLINE`、`FAULTED`、`DEGRADED`、`UNAVAIL`、`OFFLINE` です。`ONLINE` 以外のいずれかの状態の場合は、プールの耐障害性が危殆化しています。

構成出力の 2 番目のセクションには、エラー統計が表示されます。これらのエラーは、3 つのカテゴリに分けられます。

- `READ` – 読み取り要求を実行したときに発生した入出力エラー
- `WRITE` – 書き込み要求を実行したときに発生した入出力エラー
- `CKSUM` – チェックサムエラー。読み取り要求の結果として、破壊されたデータがデバイスから返されたことを意味する

これらのエラーを使って、損傷が永続的かどうかを判断できます。入出力エラーが少数の場合は、機能が一時的に停止している可能性があります。入出力エラーが大量の場合は、デバイスに永続的な問題が発生している可能性があります。これらのエラーは、アプリケーションによって解釈されるデータ破壊に対応していないことがあります。デバイスが冗長構成になっている場合は、デバイスの訂正できないエラーが表示されることがあります。ただし、ミラーまたは RAID-Z デバイスレベルではエラーは表示されません。そのような場合、ZFS は正常なデータの取得に成功し、既存の複製から損傷したデータの回復を試みたことになります。

これらのエラーを解釈する方法の詳細については、309 ページの「デバイス障害の種類を確認する」を参照してください。

さらに、zpool status 出力の最終列には、補足情報が表示されます。この情報は、state フィールドの情報を補足するもので、障害の診断に役立ちます。デバイスが FAULTED の場合は、このフィールドにはデバイスがアクセスできない状態かどうか、またはデバイス上のデータが破壊されているかどうかが表示されます。デバイスで再同期化が実行されている場合、このフィールドには現在の進行状況が表示されます。

再同期化の進行状況を監視する方法の詳細については、317 ページの「再同期化の状態を表示する」を参照してください。

スクラブの状態

zpool status 出力のスクラブセクションには、すべての明示的なスクラブ操作の現在の状態が説明されます。この情報は、システム上でなんらかのエラーが検出されているかどうかを示すものではありません。ただし、この情報を使って、データ破壊エラーの報告が正確かどうかを判断できます。前回のスクラブが最近実行されている場合には、既知のデータ破壊が発生していれば、高い確率でそのとき検出されている可能性があります。

スクラブ完了メッセージはシステムの再起動後も残ります。

データスクラブおよびこの情報の解釈方法の詳細については、299 ページの「ZFS ファイルシステムの整合性をチェックする」を参照してください。

データ破壊エラー

zpool status コマンドでは、既知のエラーが発生している場合に、それらがプールに関連するものであるかどうかも出力されます。これらのエラーは、データのスクラブ中または通常の操作中に検出されている可能性があります。ZFS では、プールに関連するすべてのデータエラーの持続的なログを管理しています。システムの完全なスクラブが終了するたびに、このログのローテーションが行われます。

データ破壊エラーは、常に致命的です。このエラーが発生している場合は、プールのデータが破壊されたために、1 つ以上のアプリケーションで入出力エラーが発生したことになります。冗長なプール内でデバイスエラーが発生してもデータは破壊されないので、このログの一部として記録されません。デフォルトでは、検出されたエラーの数だけが表示されます。エラーおよびその詳細の完全なリストは、zpool status -v オプションを使用すれば表示できます。次に例を示します。

```
# zpool status -v
  pool: tank
 state: UNAVAIL
status: One or more devices are faulted in response to IO failures.
```

```
action: Make sure the affected devices are connected, then run 'zpool clear'.
   see: http://www.sun.com/msg/ZFS-8000-HC
 scrub: scrub completed after 0h0m with 0 errors on Tue Feb  2 13:08:42 2010
config:

        NAME        STATE     READ WRITE CKSUM
        tank        UNAVAIL      0     0     0  insufficient replicas
          c1t0d0    ONLINE       0     0     0
          c1t1d0    UNAVAIL      4     1     0  cannot open

errors: Permanent errors have been detected in the following files:

/tank/data/aaa
/tank/data/bbb
/tank/data/ccc
```

同様のメッセージは、システムコンソールで fmd を実行した場合にも、また /var/adm/messages ファイルにも表示されます。fmdump コマンドを使って、これらのメッセージを追跡することもできます。

データ破壊エラーの解釈の詳細については、319 ページの「データ破壊の種類を確認する」を参照してください。

ZFS エラーメッセージのシステムレポート

ZFS では、プール内のエラーを継続的に追跡するだけでなく、そのようなイベントが発生したときに syslog メッセージを表示することもできます。次のような場合に、イベントを生成して管理者に通知します。

- デバイス状態の移行 – デバイスが FAULTED になると、プールの耐障害性が危殆化する可能性があることを示すメッセージがログに記録されます。あとでデバイスがオンラインになり、プールの健全性が復元した場合にも、同様のメッセージが送信されます。
- データの破壊 – データの破壊が検出された場合には、破壊が検出された日時と場所を示すメッセージがログに記録されます。このメッセージがログに記録されるのは、はじめて検出されたときだけです。それ以降のアクセスについては、メッセージは生成されません。
- プールの障害とデバイスの障害 – プールの障害またはデバイスの障害が発生した場合には、障害マネージャーデーモンが syslog メッセージおよび fmdump コマンドを使用してこれらのエラーを報告します。

ZFS がデバイスエラーを検出してそれを自動的に回復した場合には、通知は行われません。このようなエラーでは、プールの冗長性またはデータの完全性の障害は発生しません。また、このようなエラーは通常、ドライバの問題が原因で発生しており、ドライバ自身のエラーメッセージも出力されます。

損傷した ZFS 構成を修復する

ZFS では、アクティブなプールとその構成のキャッシュをルートファイルシステム上で管理しています。このキャッシュファイルが破壊された場合、またはこのファイルがなんらかの形でディスクに保管されている構成情報と同期しなくなった場合には、そのプールを開くことができなくなります。ZFS ではこのような状況を回避しようとしますが、配下のストレージの特性から、なんらかの破壊は常に発生する可能性があります。こうした状況になると、ほかの方法で使用できるとしても、通常はプールがシステムに表示されなくなります。また、この状況から構成が不完全であること、つまり、最上位レベルの仮想デバイスが見つからない (その数は不明) ことがわかる場合もあります。どちらの場合でも、なんらかの方法でプールを見ることができる場合には、プールをエクスポートして再度インポートする方法で、構成を回復することができます。

プールのインポートとエクスポートについては、116 ページの「ZFS ストレージプールを移行する」を参照してください。

見つからないデバイスに関する問題を解決する

デバイスを開けない場合には、`zpool status` の出力に `UNAVAIL` 状態が表示されます。この状態は、プールにはじめてアクセスしたときにデバイスを開けなかったか、またはそのデバイスがそれ以降使用できない状態であることを示しています。このデバイスが原因で、最上位レベルの仮想デバイスが使用できない場合、そのプールの内容にはアクセスできません。または、プールの耐障害性が危殆化している可能性があります。どちらの場合でも、通常の動作に戻すために必要な操作は、そのデバイスをシステムに再接続することだけです。

たとえば、デバイス障害が発生したあとに、 `fmd` から次のようなメッセージが表示される場合があります。

```
SUNW-MSG-ID: ZFS-8000-FD, TYPE: Fault, VER: 1, SEVERITY: Major
EVENT-TIME: Thu Jun 24 10:42:36 PDT 2010
PLATFORM: SUNW,Sun-Fire-T200, CSN: -, HOSTNAME: neo2
SOURCE: zfs-diagnosis, REV: 1.0
EVENT-ID: a1fb66d0-cc51-cd14-a835-961c15696fcb
DESC: The number of I/O errors associated with a ZFS device exceeded
acceptable levels.  Refer to http://sun.com/msg/ZFS-8000-FD for more information.
AUTO-RESPONSE: The device has been offlined and marked as faulted.  An attempt
will be made to activate a hot spare if available.
IMPACT: Fault tolerance of the pool may be compromised.
REC-ACTION: Run 'zpool status -x' and replace the bad device.
```

デバイスの問題と解決策についてより詳細な情報を表示するには、`zpool status -x` コマンドを使用します。次に例を示します。

```
# zpool status -x
  pool: tank
 state: DEGRADED
status: One or more devices could not be opened.  Sufficient replicas exist for
        the pool to continue functioning in a degraded state.
action: Attach the missing device and online it using 'zpool online'.
   see: http://www.sun.com/msg/ZFS-8000-2Q
 scrub: scrub completed after 0h0m with 0 errors on Tue Feb  2 13:15:20 2010
config:

        NAME        STATE     READ WRITE CKSUM
        tank        DEGRADED     0     0     0
          mirror-0  DEGRADED     0     0     0
            c1t0d0  ONLINE       0     0     0
            c1t1d0  UNAVAIL      0     0     0  cannot open

errors: No known data errors
```

この出力から、見つからない c1t1d0 デバイスが機能していないことを確認できます。デバイスで障害が発生していると判断した場合は、デバイスを置き換えます。

次に、zpool online コマンドを使用して、置き換えたデバイスをオンラインにします。次に例を示します。

```
# zpool online tank c1t1d0
```

最後のステップでは、デバイスを置き換えたプールの健全性を確認します。次に例を示します。

```
# zpool status -x tank
pool 'tank' is healthy
```

デバイスを物理的に再接続する

見つからないデバイスを再接続するための正確な手順は、そのデバイスごとに異なります。デバイスがネットワークに接続されているドライブの場合は、ネットワークへの接続を復元するべきです。デバイスが USB デバイスなどのリムーバブルメディアである場合は、システムに再接続するべきです。デバイスがローカルディスクである場合は、コントローラに障害が発生していたために、デバイスがシステムから見えない状態になっていた可能性があります。この場合は、コントローラを置き換えれば、ディスクが再び使用できる状態になるはずです。ハードウェアの種類と構成によっては、ほかの問題が存在する可能性もあります。ドライブに障害が発生してシステムから認識されなくなった場合には、デバイスが損傷していると見なすべきです。309 ページの「破損したデバイスを交換または修復する」の手順に従ってください。

デバイスが使用できることを ZFS に通知する

デバイスをシステムに再接続したあとも、デバイスが使用できるようになったことが自動的に検出されないこともあります。プールでそれまで障害が発生した場合、または接続手続きの一部としてシステムが再起動された場合には、プールを開こうとするときにすべてのデバイスが自動的に再スキャンされます。システムの稼働中にプールの機能が低下したのでデバイスを置き換えた場合には、デバイスが使用できるようになって再度開ける状態になったことを、`zpool online` コマンドを使って ZFS に通知する必要があります。次に例を示します。

```
# zpool online tank c0t1d0
```

デバイスをオンラインする方法の詳細については、94 ページの「デバイスをオンラインにする」を参照してください。

破損したデバイスを交換または修復する

この節では、デバイスの障害の種類を確認し、一時的なエラーを消去し、デバイスを置き換える方法について説明します。

デバイス障害の種類を確認する

「損傷したデバイス」という用語は定義があいまいですが、発生する可能性のあるいくつかの状況はこの用語で説明できます。

- ビットの腐敗 – 時間の経過とともに、磁力の影響や宇宙線などのさまざまなことが原因で、ディスクに格納されているビットが反転してしまうことがあります。このようなことはあまり発生しませんが、発生した場合には、大規模なまたは長期間稼働するシステムでデータが破壊する可能性は十分にあります。
- 間違った方向への読み取りまたは書き込み – ファームウェアのバグまたはハードウェア障害のために、ブロック全体の読み取りまたは書き込みで、ディスク上の不正な場所を参照してしまうことがあります。これらのエラーは通常、一時的です。ただし、エラーの数が多い場合には、ドライブの障害が発生している可能性があります。
- 管理者エラー – 管理者が意図せずにディスクの一部を不正なデータで上書きする (ディスクの一部に `/dev/zero` をコピーするなど) ことで、ディスクが永続的に破壊されてしまう場合があります。これらのエラーは常に一時的です。
- 一時的な機能停止– ディスクが一定期間使用できなくなり、入出力に失敗することがあります。この状況は通常、ネットワークに接続されたデバイスに発生しますが、ローカルディスクでも一時的に機能が停止することがあります。これらのエラーは、一時的な場合と、そうでない場合があります。

- 不良または信頼性の低いハードウェア – この状況は、ハードウェアの障害によって引き起こされるさまざまな問題の総称です。問題の例としては、断続的な入出力エラー、不規則な破壊を引き起こす転送エラー、その他のさまざまな障害があります。これらのエラーは通常永続的です。
- オフラインのデバイス – デバイスがオフラインである場合は、そのデバイスに障害が発生していると判断した管理者がデバイスをこの状態にしたと推定されます。管理者は、デバイスをこの状態にしたうえで、この推定が正しいかどうかを判断できます。

デバイスのどこに問題があるかを正確に判断することは、難しい作業です。最初に行うことは、`zpool status` 出力のエラー数を調べることです。次に例を示します。

```
# zpool status -v tpool
  pool: tpool
 state: ONLINE
status: One or more devices has experienced an error resulting in data
        corruption.  Applications may be affected.
action: Restore the file in question if possible.  Otherwise restore the
        entire pool from backup.
   see: http://www.sun.com/msg/ZFS-8000-8A
 scrub: scrub completed after 0h0m with 2 errors on Tue Jul 13 11:08:37 2010
config:

        NAME        STATE     READ WRITE CKSUM
        tpool       ONLINE       2     0     0
          c1t1d0    ONLINE       2     0     0
          c1t3d0    ONLINE       0     0     0
errors: Permanent errors have been detected in the following files:

        /tpool/words
```

エラーは、入出力エラーとチェックサムエラーに分かれます。どちらのエラーも、発生している可能性のある障害の種類を示している可能性があります。通常の処理で発生するエラーの数は、少ない (長い時間にほんの数個) と予測されます。大量のエラーが表示される場合、この状況はデバイス障害がすぐに発生する可能性または完全なデバイス障害が発生する可能性を示しています。ただし、管理者のミスが原因で大量のエラーが表示される可能性もあります。別の情報源は、`syslog` システムログです。このログに大量の SCSI ドライバまたはファイバチャネルドライバのメッセージが記録される場合、この状況は重大なハードウェアの問題が発生している可能性を示しています。`syslog` メッセージが生成されない場合、損傷は一時的であると思われます。

最後の手順は次の質問に答えることです。

このデバイスでもう一度エラーが発生する可能性がありますか。

一度だけ発生するエラーは「一時的」と考えられ、潜在的な障害を示していません。ハードウェア障害の可能性がある持続的または重大なエラーは、「致命的」と考えられます。エラーの種類を特定する作業は、ZFS で現在利用できる自動化ソフト

ウェアの範囲を超えているため、管理者自身が手動で行う必要があります。エラーの種類を特定したあとで、それに対応する処置を採ることができます。一時的なエラーを解消したり、致命的なエラーが起こっているデバイスを置き換えたりします。これらの修復手順については、次の節で説明します。

一時的であると考えられるデバイスエラーでも、それらがプール内のデータの訂正不可能なエラーを発生させていることがあります。このようなエラーについては、配下のデバイスが健全であると判断されている場合、または別の機会に修復されている場合でも、特別な修復手順が必要になります。データエラーの修復の詳細については、318 ページの「損傷したデータを修復する」を参照してください。

一時的なエラーを解消する

デバイスエラーが一時的と考えられる場合、つまりデバイスの今後の健全性に影響しないと考えられる場合は、デバイスエラーを安全に解消することで、致命的なエラーが発生していないと示すことができます。RAID-Z デバイスまたはミラーデバイスのエラー数を消去するには、`zpool clear` コマンドを使用します。次に例を示します。

```
# zpool clear tank c1t1d0
```

この構文を実行すると、すべてのデバイスエラーと、デバイスに関連付けられたすべてのデータエラー数が消去されます。

プール内の仮想デバイスに関連付けられているすべてのエラーを消去し、プールに関連付けられているすべてのデータエラー数を消去するには、次の構文を使用します。

```
# zpool clear tank
```

プールエラーの消去の詳細については、95 ページの「ストレージプールデバイスのエラーをクリアーする」を参照してください。

ZFS ストレージプール内のデバイスを置き換える

デバイスの損傷が永続的である場合、または永続的な損傷が今後発生する可能性がある場合には、そのデバイスを置き換える必要があります。デバイスを置き換えられるかどうかは、構成によって異なります。

デバイスを置き換えられるかどうかを確認する

デバイスを置き換えるには、プールが ONLINE 状態である必要があります。デバイスは冗長構成の一部であるか、健全 (ONLINE 状態) である必要があります。デバイスが冗長構成の一部である場合は、正常なデータを取得するための十分な複製が存在している必要があります。4 方向ミラーの 2 台のディスク に障害が発生している場合は、健全な複製を入手できるので、どちらのディスクも置き換えることができます。ただし、4 方向 RAID-Z (raidz1) 仮想デバイス内の 2 つのディスクで障害が発生した場合は、データを入手するために必要な複製がないため、どちらのディスクも置き換えることができません。デバイスが損傷していてもオンラインである場合には、プールの状態が FAULTED でない限り、デバイスを置き換えることができます。ただし、損傷を受けたデバイス上の壊れたデータは、正常なデータが格納されている複製が必要な数だけ存在しない場合には、新しいデバイスにコピーされます。

次の構成で、c1t1d0 ディスクは置き換えることができます。プール内のすべてのデータは正常な複製 c1t0d0 からコピーされます。

```
    mirror            DEGRADED
    c1t0d0             ONLINE
    c1t1d0             FAULTED
```

c1t0d0 ディスクも置き換えることができますが、正常な複製を入手できないため、データの自己修復は行われません。

次の構成では、障害が発生したどのディスクも置き換えることができません。プール自体に障害が発生しているため、ONLINE 状態のディスクも置き換えることができません。

```
    raidz             FAULTED
    c1t0d0            ONLINE
    c2t0d0            FAULTED
    c3t0d0            FAULTED
    c4t0d0            ONLINE
```

次の構成の最上位レベルのディスクは、どちらも置き換えることができます。ただし、ディスクに不正なデータが存在する場合は、それらが新しいディスクにコピーされます。

```
c1t0d0         ONLINE
c1t1d0         ONLINE
```

どちらかのディスクで障害が発生している場合は、プール自体に障害が発生していることになるため、置き換えを実行できません。

置き換えることができないデバイス

デバイスが失われたためにプールが障害状態になった場合、または非冗長な構成でデバイスに大量のデータエラーが含まれている場合は、そのデバイスを安全に置き

換えることはできません。十分な冗長性がない場合、損傷したデバイスの修復に使用する正常なデータは存在しません。この場合は、プールを破棄して構成を再作成したのちに、データをバックアップコピーから復元するのが唯一の選択肢です。

プール全体を復元する方法の詳細については、321 ページの「ZFS ストレージプール全体の損傷を修復する」を参照してください。

ZFS ストレージプール内のデバイスを置き換える

置き換えられるデバイスであることを確認したあとで、`zpool replace` コマンドを使ってデバイスを置き換えます。損傷したデバイスを別のデバイスに置き換える場合は、次のような構文を使用します。

```
# zpool replace tank c1t1d0 c2t0d0
```

このコマンドを実行すると、損傷したデバイスまたはプール内のほかのデバイス (冗長な構成の場合) から新しいデバイスにデータが移行されます。コマンドが完了すると、損傷したデバイスが構成から切り離され、そのデバイスをシステムから取り外せる状態になります。1 つの場所ですでにデバイスを取り外して新しいデバイスに置き換えている場合には、1 つのデバイス形式のコマンドを使用します。次に例を示します。

```
# zpool replace tank c1t1d0
```

このコマンドにフォーマットされていないディスクを指定すると、そのディスクが適切な状態にフォーマットされたのち、残りの構成からデータが再同期化されます。

`zpool replace` コマンドの詳細については、96 ページの「ストレージプール内のデバイスを置き換える」を参照してください。

例 11–1　ZFS ストレージプール内のデバイスを置き換える

次の例では、Oracle の Sun Fire x4500 システム上のミラー化ストレージプール `tank` 内のデバイス (`c1t3d0`) を置き換える方法を示します。ディスク `c1t3d0` を同じ位置 (`c1t3d0`) で新しいディスクに置き換えるには、ディスクを置き換える前に構成解除する必要があります。基本的な手順は次のとおりです。

- 置き換えるディスク (`c1t3d0`) をオフラインにします。現在使用中のディスクを構成解除することはできません。
- `cfgadm` コマンドを使用して、構成解除するディスク (`c1t3d0`) を識別し、このディスクを構成解除します。このミラー化構成にオフラインのディスクが存在するプールの機能は低下しますが、プールは引き続き使用可能です。
- ディスク (`c1t3d0`) を物理的に交換します。障害の発生したドライブを物理的に取り外す前に、青色の `Ready to Remove` (取り外し準備完了) LED が点灯していることを確認してください。

例 11–1 ZFS ストレージプール内のデバイスを置き換える (続き)

- ディスク (c1t3d0) を再構成します。
- 新しいディスク (c1t3d0) をオンラインにします。
- zpool replace コマンドを実行してディスク (c1t3d0) を置き換えます。

注 – あらかじめプールの autoreplace プロパティーをオンに設定してあった場合、そのプールに以前属していたデバイスと物理的に同じ位置に新しいデバイスが検出されると、そのデバイスは自動的にフォーマットされ、zpool replace コマンドを使用せずに置き換えられます。ハードウェアによっては、この機能はサポートされない場合があります。

- 障害の発生したディスクがホットスペアに自動的に置き換えられる場合は、障害の発生したディスクが置き換えられたあとでホットスペアの切り離しが必要になることがあります。たとえば、障害の発生したディスクが置き換えられたあとも c2t4d0 がアクティブなホットスペアになっている場合は、切り離してください。

```
# zpool detach tank c2t4d0
```

次の例では、ZFS ストレージプール内のディスクを置き換える手順を示します。

```
# zpool offline tank c1t3d0
# cfgadm | grep c1t3d0
sata1/3::dsk/c1t3d0            disk         connected    configured   ok
# cfgadm -c unconfigure sata1/3
Unconfigure the device at: /devices/pci@0,0/pci1022,7458@2/pci11ab,11ab@1:3
This operation will suspend activity on the SATA device
Continue (yes/no)? yes
# cfgadm | grep sata1/3
sata1/3                        disk         connected    unconfigured ok
<Physically replace the failed disk c1t3d0>
# cfgadm -c configure sata1/3
# cfgadm | grep sata1/3
sata1/3::dsk/c1t3d0            disk         connected    configured   ok
# zpool online tank c1t3d0
# zpool replace tank c1t3d0
# zpool status tank
  pool: tank
 state: ONLINE
 scrub: resilver completed after 0h0m with 0 errors on Tue Feb  2 13:17:32 2010
config:

        NAME        STATE     READ WRITE CKSUM
        tank        ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c0t1d0  ONLINE       0     0     0
            c1t1d0  ONLINE       0     0     0
          mirror-1  ONLINE       0     0     0
            c0t2d0  ONLINE       0     0     0
```

例 11–1 ZFS ストレージプール内のデバイスを置き換える (続き)

```
            c1t2d0  ONLINE       0     0     0
          mirror-2  ONLINE       0     0     0
            c0t3d0  ONLINE       0     0     0
            c1t3d0  ONLINE       0     0     0

errors: No known data errors
```

上記の zpool の出力で、新しいディスクと古いディスクの両方が *replacing* 見出しの下に表示される場合があります。次に例を示します。

```
replacing     DEGRADED     0     0    0
  c1t3d0s0/o  FAULTED      0     0    0
  c1t3d0      ONLINE       0     0    0
```

このテキストは、置き換え処理および新しいディスクの再同期化が進行中であることを示しています。

ディスク (c1t3d0) を別のディスク (c4t3d0) で置き換える場合は、zpool replace コマンドの実行だけが必要です。次に例を示します。

```
# zpool replace tank c1t3d0 c4t3d0
# zpool status
  pool: tank
 state: DEGRADED
 scrub: resilver completed after 0h0m with 0 errors on Tue Feb  2 13:35:41 2010
config:

        NAME             STATE     READ WRITE CKSUM
        tank             DEGRADED     0     0     0
          mirror-0       ONLINE       0     0     0
            c0t1d0       ONLINE       0     0     0
            c1t1d0       ONLINE       0     0     0
          mirror-1       ONLINE       0     0     0
            c0t2d0       ONLINE       0     0     0
            c1t2d0       ONLINE       0     0     0
          mirror-2       DEGRADED     0     0     0
            c0t3d0       ONLINE       0     0     0
            replacing    DEGRADED     0     0     0
              c1t3d0     OFFLINE      0     0     0
              c4t3d0     ONLINE       0     0     0

errors: No known data errors
```

ディスクの置き換えが完了するまでに zpool status コマンドを数回実行する必要がある場合があります。

```
# zpool status tank
  pool: tank
 state: ONLINE
 scrub: resilver completed after 0h0m with 0 errors on Tue Feb  2 13:35:41 2010
config:
```

例 11–1 ZFS ストレージプール内のデバイスを置き換える (続き)

```
        NAME          STATE     READ WRITE CKSUM
        tank          ONLINE       0     0     0
          mirror-0    ONLINE       0     0     0
            c0t1d0    ONLINE       0     0     0
            c1t1d0    ONLINE       0     0     0
          mirror-1    ONLINE       0     0     0
            c0t2d0    ONLINE       0     0     0
            c1t2d0    ONLINE       0     0     0
          mirror-2    ONLINE       0     0     0
            c0t3d0    ONLINE       0     0     0
            c4t3d0    ONLINE       0     0     0
```

例 11–2 障害が発生したログデバイスを交換する

次の例では、ストレージプール `pool` で障害が発生したログデバイス `c0t5d0` を回復する方法を示します。基本的な手順は次のとおりです。

- `zpool status -x` の出力と FMA 診断メッセージを確認します (次のサイトの説明を参照)。

  http://www.sun.com/msg/ZFS-8000-K4

- 障害が発生したログデバイスを物理的に交換します。
- 新しいログデバイスをオンラインにします。
- プールのエラー状況がクリアされます。

```
# zpool status -x
  pool: pool
 state: FAULTED
status: One or more of the intent logs could not be read.
        Waiting for adminstrator intervention to fix the faulted pool.
action: Either restore the affected device(s) and run 'zpool online',
        or ignore the intent log records by running 'zpool clear'.
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        pool        FAULTED      0     0     0 bad intent log
          mirror    ONLINE       0     0     0
            c0t1d0  ONLINE       0     0     0
            c0t4d0  ONLINE       0     0     0
        logs        FAULTED      0     0     0 bad intent log
          c0t5d0    UNAVAIL      0     0     0 cannot open
<Physically replace the failed log device>
# zpool online pool c0t5d0
# zpool clear pool

# zpool status -x
  pool: pool
 state: FAULTED
status: One or more of the intent logs could not be read.
        Waiting for adminstrator intervention to fix the faulted pool.
```

例 11–2 障害が発生したログデバイスを交換する (続き)

```
action: Either restore the affected device(s) and run 'zpool online',
        or ignore the intent log records by running 'zpool clear'.
 scrub: none requested
config:

        NAME          STATE     READ WRITE CKSUM
        pool          FAULTED      0     0     0 bad intent log
          mirror-0    ONLINE       0     0     0
            c0t1d0    ONLINE       0     0     0
            c0t4d0    ONLINE       0     0     0
        logs          FAULTED      0     0     0 bad intent log
          c0t5d0      UNAVAIL      0     0     0 cannot open
<Physically replace the failed log device>
# zpool online pool c0t5d0
# zpool clear pool
```

再同期化の状態を表示する

デバイスを置き換えるときには、デバイスのサイズとプールに含まれるデータの量によっては、かなり長い時間がかかることがあります。あるデバイスのデータを別のデバイスに移動する処理は「再同期化」と呼ばれ、`zpool status` コマンドを使って監視できます。

従来のファイルシステムでは、ブロックレベルでデータが再同期化されます。ZFS では、ボリュームマネージャーの論理階層がなくなり、より強力な制御された方法で再同期化できます。この機能の主な利点として、次の 2 点を挙げることができます。

- ZFS では、最小限の必要なデータ量だけが再同期化されます。デバイスの完全な置き換えとは異なり、短時間の停止の場合は、わずか数分または数秒でディスク全体を再同期化できます。ディスク全体を置き換えるときは、再同期化処理にかかる時間は、ディスク上で使用されているデータ量に比例します。500G バイトのディスクを置き換えるときでも、プールで使用されているディスク容量が数 G バイトであれば、数秒で完了できます。
- 再同期化は、割り込み可能で安全です。システムの電源が切れるか、またはシステムが再起動した場合には、再同期化処理は中断した場所から正確に再開されます。手動で介入する必要はありません。

再同期化処理を表示するには、`zpool status` コマンドを使用します。次に例を示します。

```
# zpool status tank
  pool: tank
 state: DEGRADED
status: One or more devices is currently being resilvered.  The pool will
        continue to function, possibly in a degraded state.
action: Wait for the resilver to complete.
```

```
 scrub: resilver in progress for 0h0m, 22.60% done, 0h1m to go
config:
        NAME                  STATE     READ WRITE CKSUM
        tank                DEGRADED     0     0     0
          mirror-0          DEGRADED     0     0     0
            replacing-0     DEGRADED     0     0     0
              c1t0d0        UNAVAIL      0     0     0  cannot open
              c2t0d0        ONLINE       0     0     0  85.0M resilvered
            c1t1d0          ONLINE       0     0     0

errors: No known data errors
```

この例では、ディスク c1t0d0 が c2t0d0 に置き換わります。状態が replacing の仮想デバイスが構成に存在しているので、このイベントは状態出力で監視されます。このデバイスは実際のデバイスではなく、このデバイスを使ってプールを作成することもできません。このデバイスは、再同期化処理を表示し、置き換え中のデバイスを識別するためだけに使用されます。

再同期化が現在進行しているプールの状態は、すべて ONLINE または DEGRADED 状態になります。これは、再同期化処理が完了するまで、必要とする冗長レベルをそのプールで提供できないためです。システムへの影響を最小限に抑えるために、再同期化は最大限の速度で処理されます。ただし、その入出力の優先順位は、ユーザーが要求した入出力より常に低く設定されます。再同期化が完了すると、新しい完全な構成に戻ります。次に例を示します。

```
# zpool status tank
  pool: tank
 state: ONLINE
 scrub: resilver completed after 0h1m with 0 errors on Tue Feb  2 13:54:30 2010
config:

        NAME        STATE     READ WRITE CKSUM
        tank        ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c2t0d0  ONLINE       0     0     0  377M resilvered
            c1t1d0  ONLINE       0     0     0

errors: No known data errors
```

プールは ONLINE に戻り、元の障害が発生したディスク (c1t0d0) は構成から削除されています。

損傷したデータを修復する

ここでは、データ破壊の種類を確認する方法と、破壊したデータを修復する (可能な場合) 方法について説明します。

- 319 ページの「データ破壊の種類を確認する」
- 320 ページの「破壊されたファイルまたはディレクトリを修復する」

- 321 ページの「ZFS ストレージプール全体の損傷を修復する」

ZFS では、データ破壊のリスクを最小限に抑えるために、チェックサム、冗長性、および自己修復データが使用されます。それでも、プールが冗長でない場合、プールの機能が低下しているときに破壊が発生した場合、または予期しないことが一度に起こってデータの複数のコピーが破壊された場合は、データの破壊が発生することがあります。どのような原因であったとしても、結果は同じです。 データが破壊され、その結果アクセスできなくなっています。対処方法は、破壊されたデータの種類とその相対的な価値により異なります。破壊されるデータは、大きく 2 つの種類に分けられます。

- プールメタデータ – ZFS では、プールを開いてデータセットにアクセスするために、一定量のデータを解析する必要があります。これらのデータが破壊された場合には、プール全体またはデータセット階層の一部が使用できなくなります。
- オブジェクトデータ – この場合、破壊は特定のファイルまたはディレクトリに限定されます。この問題が発生すると、そのファイルまたはディレクトリの一部がアクセスできなくなる可能性があります。この問題が原因で、オブジェクトも一緒に破壊されることがあります。

データの検証は、通常の操作中およびスクラブ時に行われます。プールデータの完全性を検証する方法については、299 ページの「ZFS ファイルシステムの整合性をチェックする」を参照してください。

データ破壊の種類を確認する

デフォルトでは、`zpool status` コマンドでは、破壊が発生したことだけが報告され、破壊が発生した場所は報告されません。次に例を示します。

```
# zpool status monkey
  pool: monkey
 state: ONLINE
status: One or more devices has experienced an error resulting in data
        corruption.  Applications may be affected.
action: Restore the file in question if possible.  Otherwise restore the
        entire pool from backup.
   see: http://www.sun.com/msg/ZFS-8000-8A
 scrub: scrub completed after 0h0m with 8 errors on Tue Jul 13 13:17:32 2010
config:

        NAME        STATE     READ WRITE CKSUM
        monkey      ONLINE       8     0     0
          c1t1d0    ONLINE       2     0     0
          c2t5d0    ONLINE       6     0     0

errors: 8 data errors, use '-v' for a list
```

特定の時刻にエラーが発生したことだけが、エラーごとに報告されます。各エラーが現在もシステムに存在するとは限りません。通常の状況下では、これが当て

はまります。なんらかの一時的な機能停止によって、データが破壊されることがあります。それらは、機能停止が終了したあとで自動的に修復されます。プール内のすべてのアクティブなブロックを検査するために、プールのスクラブは完全に実行されることが保証されています。このため、スクラブが完了するたびに、エラーログがリセットされます。エラーが存在しないことを確認したので、スクラブが完了するのを待っている必要がない場合には、`zpool online` コマンドを使ってプール内のすべてのエラーをリセットします。

データ破壊がプール全体のメタデータで発生している場合は、出力が少し異なります。次に例を示します。

```
# zpool status -v morpheus
  pool: morpheus
    id: 1422736890544688191
 state: FAULTED
status: The pool metadata is corrupted.
action: The pool cannot be imported due to damaged devices or data.
   see: http://www.sun.com/msg/ZFS-8000-72
config:

        morpheus    FAULTED   corrupted data
          c1t10d0   ONLINE
```

プール全体が破壊された場合、プールは必要な冗長レベルを提供できないため、`FAULTED` 状態になります。

破壊されたファイルまたはディレクトリを修復する

ファイルまたはディレクトリが破壊されても、破壊の種類によってはシステムがそのまま動作する場合があります。データの正常なコピーがシステムに存在しなければ、どの損傷も事実上修復できません。貴重なデータの場合は、影響を受けたデータをバックアップから復元する必要があります。このような場合でも、プール全体を復元しなくても破壊から回復できる場合があります。

ファイルデータブロックの中で損傷が発生した場合は、ファイルを安全に削除することができるため、システムのエラーを解消できます。永続的なエラーが発生しているファイル名のリストを表示するには、`zpool status -v` コマンドを使用します。次に例を示します。

```
# zpool status -v
  pool: monkey
 state: ONLINE
status: One or more devices has experienced an error resulting in data
        corruption.  Applications may be affected.
action: Restore the file in question if possible.  Otherwise restore the
        entire pool from backup.
```

```
   see: http://www.sun.com/msg/ZFS-8000-8A
 scrub: scrub completed after 0h0m with 8 errors on Tue Jul 13 13:17:32 2010
config:

        NAME        STATE     READ WRITE CKSUM
        monkey      ONLINE       8     0     0
          c1t1d0    ONLINE       2     0     0
          c2t5d0    ONLINE       6     0     0

errors: Permanent errors have been detected in the following files:

/monkey/a.txt
/monkey/bananas/b.txt
/monkey/sub/dir/d.txt
monkey/ghost/e.txt
/monkey/ghost/boo/f.txt
```

永続的なエラーが発生しているファイル名のリストは、次のようになります。

- ファイルへの完全なパスが見つかり、データセットがマウントされている場合は、ファイルへの完全なパスが表示されます。次に例を示します。

  ```
  /monkey/a.txt
  ```

- ファイルへの完全なパスは見つかったが、データセットがマウントされていない場合は、前にスラッシュ (/) が付かず、後ろにファイルへのデータセット内のパスが付いたデータセット名が表示されます。次に例を示します。

  ```
  monkey/ghost/e.txt
  ```

- エラーにより、または `dnode_t` の場合のようにオブジェクトに実際のファイルパスが関連付けられていないことにより、ファイルパスに対するオブジェクト番号を正常に変換できない場合は、後ろにオブジェクト番号の付いたデータセット名が表示されます。次に例を示します。

  ```
  monkey/dnode:<0x0>
  ```

- メタオブジェクトセット (MOS) のオブジェクトが破壊された場合は、後ろにオブジェクト番号の付いた `<metadata>` という特別なタグが表示されます。

ディレクトリまたはファイルのメタデータの中で破壊は発生している場合には、そのファイルを別の場所に移動するしかありません。任意のファイルまたはディレクトリを不便な場所に安全に移動することができ、そこで元のオブジェクトを復元することができます。

ZFS ストレージプール全体の損傷を修復する

プールのメタデータが損傷していて、その損傷によりプールを開けないかインポートできない場合の選択肢には、次のものがあります。

- zpool clear - F コマンドまたは zpool import -F コマンドを使用して、プールの回復を試みます。これらのコマンドは、プールに対する直近数回のトランザクションをロールバックして、プールを正常な状態に戻すことを試みます。zpool status コマンドを使用すると、損傷したプールと推奨される回復手順を確認できます。次に例を示します。

```
# zpool status
  pool: tpool
 state: FAULTED
status: The pool metadata is corrupted and the pool cannot be opened.
action: Recovery is possible, but will result in some data loss.
        Returning the pool to its state as of Wed Jul 14 11:44:10 2010
        should correct the problem.  Approximately 5 seconds of data
        must be discarded, irreversibly.  Recovery can be attempted
        by executing 'zpool clear -F tpool'.  A scrub of the pool
        is strongly recommended after recovery.
   see: http://www.sun.com/msg/ZFS-8000-72
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
         tpool      FAULTED      0     0     1  corrupted data
          c1t1d0    ONLINE       0     0     2
          c1t3d0    ONLINE       0     0     4
```

これまでに説明した回復プロセスでは、次のコマンドを使用します。

```
# zpool clear -F tpool
```

損傷したストレージプールをインポートしようとすると、次のようなメッセージが表示されます。

```
# zpool import tpool
cannot import 'tpool': I/O error
        Recovery is possible, but will result in some data loss.
        Returning the pool to its state as of Wed Jul 14 11:44:10 2010
        should correct the problem.  Approximately 5 seconds of data
        must be discarded, irreversibly.  Recovery can be attempted
        by executing 'zpool import -F tpool'.  A scrub of the pool
        is strongly recommended after recovery.
```

これまでに説明した回復プロセスでは、次のコマンドを使用します。

```
# zpool import -F tpool
Pool tpool returned to its state as of Wed Jul 14 11:44:10 2010.
Discarded approximately 5 seconds of transactions
```

損傷したプールが zpool.cache ファイルに存在する場合、システムの起動時に問題が検出され、損傷したプールが zpool status コマンドで報告されます。プールが zpool.cache ファイルに存在しない場合、プールをインポートすることも開くこともできず、プールをインポートしようとするとプールの損傷を知らせるメッセージが表示されます。

- これまでに説明した回復方法によってプールを回復できない場合は、プールとそのすべてのデータをバックアップコピーから復元する必要があります。そのために使用する方法は、プールの構成とバックアップ方法によって大きく異なります。最初に、`zpool status` コマンドを使用して表示された構成を保存しておき、プールを破棄したあとで構成を再作成できるようにします。次に、`zpool destroy -f` コマンドを使用してプールを破棄します。また、データセットのレイアウトやローカルに設定されたさまざまなプロパティーが記述されているファイルを別の安全な場所に保存します。これは、プールがアクセスできない状態になった場合に、これらの情報にアクセスできなくなるためです。プールを破棄したあとに、プールの構成とデータセットのレイアウトを使用して、完全な構成を再構築できます。次に、なんらかのバックアップまたは採用している復元方法を使って、データを生成することができます。

起動できないシステムを修復する

ZFS は、エラーが発生した場合でも、堅牢で安定した状態であるように設計されています。それでも、ソフトウェアのバグや予期しない異常な操作のために、プールにアクセスするときにシステムでパニックが発生することがあります。各プールは起動処理のときに開く必要があるので、このような障害が発生すると、システムがパニックと再起動のループに入ってしまうことになります。この状況から回復するには、起動時にどのプールも探さないように ZFS を設定する必要があります。

ZFS では、利用できるプールとその構成の内部キャッシュを `/etc/zfs/zpool.cache` で管理しています。このファイルの場所と内容は非公開で、変更される可能性があります。システムを起動できなくなった場合は、`-m milestone=none` 起動オプションを使用して、マイルストーン `none` で起動します。システムが起動したら、ルートファイルシステムを書き込み可能として再マウントしてから、`/etc/zfs/zpool.cache` ファイルの名前を変更するかこのファイルを別の場所に移動します。これらの操作によって、システムに存在するすべてのプールがキャッシュから消去されるので、問題の原因となっている正常でないプールにアクセスしようとしなくなります。この状態になったら、`svcadm milestone all` コマンドを実行して、通常のシステムの状態に戻ることができます。代替ルートから起動して修復を行う場合にも、同じような工程を使用できます。

システムが起動したあとで、`zpool import` コマンドを使ってプールをインポートしてみることができます。ただし、このコマンドを実行すると、起動で発生したエラーと同じエラーが発生する可能性があります。これは、プールにアクセスするときに起動時と同じ方法が使用されているためです。複数のプールがシステムに存在する場合は、次の手順を実行します。

- すでに説明したように、`zpool.cache` ファイルの名前を変更するか、このファイルを別の場所に移動します。

- どのプールに問題が発生している可能性があるかを調べるために、致命的エラーが報告されているプールを `fmdump -eV` コマンドで表示します。
- `fmdump` の出力に示された問題のあるプールを除き、プールを 1 つずつインポートします。

◆ ◆ ◆ 付 録 A

Oracle Solaris ZFS バージョンの説明

この付録では、利用可能な ZFS のバージョン、各バージョンの機能、および Solaris OS の各リリースで提供される ZFS のバージョンと機能について説明します。

この付録は、次の節で構成されます。

ZFS バージョンの概要

Solaris の各リリースで利用可能な特定の ZFS バージョンを使用することにより、プールやファイルシステムに関する新しい ZFS の機能が導入され、利用できるようになります。`zpool upgrade` または `zfs upgrade` を使用すると、プールまたはファイルシステムのバージョンが、現在実行中の Solaris リリースで提供されるバージョンよりも古いかどうかを識別できます。これらのコマンドを使用して、プールまたはファイルシステムのバージョンをアップグレードすることもできます。

`zpool upgrade` および `zfs upgrade` コマンドの使用方法については、33 ページの「ZFS ファイルシステムをアップグレードする (`zfs upgrade`)」および123 ページの「ZFS ストレージプールをアップグレードする」を参照してください。

ZFS プールのバージョン

次の表に、Solaris の各リリースで利用可能な ZFS プールのバージョンの一覧を示します。

バージョン	Solaris 10	説明
1	Solaris 10 6/06	初期バージョンの ZFS
2	Solaris 10 11/06	Ditto ブロック (複製されたメタデータ)
3	Solaris 10 11/06	ホットスペアおよびダブルパリティー RAID-Z
4	Solaris 10 8/07	`zpool history`
5	Solaris 10 10/08	`gzip` 圧縮アルゴリズム
6	Solaris 10 10/08	`bootfs` プールプロパティー
7	Solaris 10 10/08	別のインテントログデバイス
8	Solaris 10 10/08	委任管理
9	Solaris 10 10/08	`refquota` および `refreservation` プロパティー
10	Solaris 10 5/09	キャッシュデバイス
11	Solaris 10 10/09	スクラブパフォーマンスの向上
12	Solaris 10 10/09	スナップショットプロパティー
13	Solaris 10 10/09	`snapused` プロパティー
14	Solaris 10 10/09	`aclinherit passthrough-x` プロパティー
15	Solaris 10 10/09	ユーザーおよびグループの領域の計上
16	Solaris 10 9/10	`stmf` プロパティーのサポート
17	Solaris 10 9/10	トリプルパリティー RAID-Z
18	Solaris 10 9/10	ユーザーによるスナップショットの保持
19	Solaris 10 9/10	ログデバイスの削除
20	Solaris 10 9/10	`zle` (長さゼロのエンコード) を使用した圧縮
21	Solaris 10 9/10	予約済み
22	Solaris 10 9/10	受信プロパティー

ZFS ファイルシステムのバージョン

次の表に、Solaris の各リリースで利用可能な ZFS ファイルシステムのバージョンの一覧を示します。

バージョン	Solaris 10	説明
1	Solaris 10 6/06	初期バージョンの ZFS ファイルシステム
2	Solaris 10 10/08	拡張されたディレクトリエントリ
3	Solaris 10 10/08	大文字小文字の区別の廃止とファイルシステム一意識別子 (FUID)
4	Solaris 10 10/09	`userquota` および `groupquota` プロパティー

索引

よ

り

ろ

わ