Oracle® Solaris 11.3 での ZFS ファイルシステムの管理

Part No: E62701
2016 年 11 月

Part No: E62701

ドキュメントのアクセシビリティについて

オラクルのアクセシビリティについての詳細情報は、Oracle Accessibility ProgramのWeb サイト(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc)を参照してください。

Oracle Supportへのアクセス

サポートをご契約のお客様には、My Oracle Supportを通して電子支援サービスを提供しています。詳細情報は(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info) か、聴覚に障害のあるお客様は (http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs)を参照してください。

目次

このドキュメントの使用法

- **概要** - ZFS ファイルシステムを設定および管理する方法について説明します。
- **対象読者** – システム管理者。
- **前提知識** - Oracle Solaris または UNIX の基本的なシステム管理の経験、および一般的なファイルシステム管理の経験。

製品ドキュメントライブラリ

この製品および関連製品のドキュメントとリソースは http://www.oracle.com/pls/topic/lookup?ctx=E62101-01 で入手可能です。

フィードバック

このドキュメントに関するフィードバックを http://www.oracle.com/goto/docfeedback からお聞かせください。

♦ ♦ ♦ 第 1 章

1 Oracle Solaris ZFS ファイルシステムの概要

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。ここには、次の情報が含まれています。

Oracle Solaris 11.3 の ZFS での新機能

この Oracle Solaris リリースには、次の ZFS 機能が導入されています。

- デフォルトのユーザー割り当て制限またはデフォルトのグループ割り当て制限を設定して、多くの異なるユーザーによって使用される大規模な共通ファイルシステムのディスク領域を制限できます。詳細は、147 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。
- 再帰的な (-r) オプションを使用すると、再帰的な ZFS スナップショットの違いを表示できます。詳細は、175 ページの「再帰的な ZFS スナップショットの相違点の表示」を参照してください。
- コマンド `zpool monitor` を使用すると、ZFS ストリームの送受信、データのスクラブ、再同期化などの、ZFS データに対して開始された進行中の操作のステータスや進行状況をモニターできます。詳細は、189 ページの「ZFS プール操作をモニターする」を参照してください。
- データセットを圧縮するときに LZ4 アルゴリズムを使用できるようになりました。詳細は、154 ページの「ZFS ファイルシステムの圧縮」を参照してください。
- ZFS キャッシュデバイスがリブート後も保持されるようになりました。詳細は、34 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。

Oracle Solaris ZFS 機能

Oracle Solaris ZFS ファイルシステムは、ほかのファイルシステムにはない機能と利点を提供します。次の表は、ZFS ファイルシステムの機能を従来のファイルシステムと比較しています。

注記 - ZFS のファイルシステムと従来のファイルシステムの相違についての詳細は、Oracle Solaris ZFS and Traditional File System Differencesを参照してください。

表 1 ZFS ファイルシステムと従来のファイルシステムの比較

ZFS ファイルシステム	従来のファイルシステム
デバイス上に作成されたストレージプールの概念を使用します。より多くのデバイスがプールに追加されると、プールサイズが拡張されます。追加の領域は、ただちに使用可能になります。	1 つのデバイスおよびそのデバイスのサイズに制約されます。
ボリュームマネージャーは必要ありません。複数のデバイスにまたがるデータ冗長性のためのプールがコマンドによって構成されます。 管理を容易にするために、ユーザーまたはプロジェクトごとに 1 つのファイルシステムをサポートします。	データ冗長性の実現のために複数のデバイスを処理するにはボリュームマネージャーが必要になるため、管理の複雑さが増します。 1 つのファイルシステムを使用して複数のサブディレクトリを管理します。
コマンドを発行することによって多数のファイルシステムを設定および管理し、階層内の子孫ファイルシステムが継承できるプロパティーを直接適用します。`/etc/vfstab` 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マウントまたはアンマウントは、ファイルシステムプロパティーに基づいて自動的に行われます。 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であるスナップショットを迅速かつ容易に作成できます。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。	デバイスやサイズの制約のために管理が複雑です。たとえば、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。
メタデータは動的に割り当てられます。事前の割り当てまたは事前に定義された制限は設定されていません。サポートされるファイルシステムの数は、使用可能なディスク領域によってのみ制限されます。	メタデータの事前の割り当てにより、ファイルシステムの作成時に領域のコストが即座に必要になります。また、事前の割り当てによって、サポートできるファイルシステムの総数も事前に定義されます。
データ管理でデータの上書きではなく、書き込み時コピーのセマンティクスが使用される、トランザクションのセマンティクスを使用します。操作のどのシーケンスも、完全にコミットされるか、または完全に無視されるかのどちらかです。偶発的な停電またはシステムクラッシュが発生すると、最後に書き込まれた一部のデータは失われる可能性がありますが、ファイルシステムは常に整合性を維持し、破損することはありません。	所定の位置にあるデータを上書きします。ファイルシステムは脆弱であり、たとえば、データブロックが割り当てられてから、それがディレクトリにリンクされるまでの間にシステムで停電が発生した場合は、整合性のない状態になります。`fsck` コマンドなどのツールやジャーナリングでは必ずしも修正が保証されず、不要なオーバーヘッドが導入される場合があります。
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。すべての障害が検出され、回復を実行できます。	チェックサム検証 (提供されている場合) がブロック単位で実行されます。ブロック全体が間違った場所に書き込まれるといった特定の障害のために、間違っているが、チェックサムエラーにならないデータが生成される場合があります。

ZFS ファイルシステム	従来のファイルシステム
さまざまなレベルのデータ冗長性によって自己修復データをサポートします。不正なデータブロックは、別の冗長コピーからの正しいデータに置き換えることによって修復できます。	
NT 形式の ACL に似た、ZFS を保護するための NFSv4 仕様に基づいた ACL (アクセス制御リスト) モデル。このモデルは、はるかにきめ細かなアクセス権限のセットを提供します。より豊富な継承セマンティクスによって、アクセス権限がディレクトリ階層を通してどのように適用されるかが指定されます。	以前の Oracle Solaris リリースでは、ACL の実装は、UFS を保護するための POSIX ACL 仕様に基づいていました。

ZFS ストレージプールのコンポーネント

このセクションでは、ZFS プールを作成するために使用されるコンポーネントについて説明します。

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。管理、信頼性、およびパフォーマンスの観点からは、ディスク全体の使用が、ZFS を使用するためのもっとも簡単で、かつもっとも効率的な方法です。ZFS では、1 つの大きなスライスが含まれるようにディスク全体をフォーマットします。ディスクの特殊なフォーマットは必要ありません。ディスクスライスからのプールの構築、ハードウェア RAID アレイ内の LUN、ソフトウェアベースのボリュームマネージャーによって提供されるボリュームなどのその他の方法では、管理がますます複雑になり、最適とは言えないパフォーマンスが提供される可能性があります。

注意 - ストレージプールのスライスの管理が複雑になる可能性があるため、スライスの使用は避けてください。

`format` コマンドは、ディスクのパーティションテーブルを表示します。Oracle Solaris が、GPT 対応ファームウェアが搭載された SPARC® システムにインストールされている場合は、EFI (GPT) ラベルがディスクに適用されます。パーティションテーブルは、次の例のようになります。

```
Current partition table (original):
```

```
Total disk sectors available: 143358287 + 16384 (reserved sectors)

Part      Tag    Flag     First Sector         Size         Last Sector
0        usr    wm               256       68.36GB          143358320
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         143358321        8.00MB          143374704
```

Oracle Solaris が x86 ベースのシステムにインストールされている場合は、ほとんどの場合、EFI (GPT) ラベルがルートプールディスクに適用されます。パーティションテーブルは、次のようになります。

```
Current partition table (original):
Total disk sectors available: 27246525 + 16384 (reserved sectors)

Part      Tag    Flag     First Sector         Size         Last Sector
0  BIOS_boot    wm               256     256.00MB         524543
1        usr    wm            524544      12.74GB         27246558
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          27246559        8.00MB         27262942
```

この出力では、パーティション 0 (`BIOS boot`) に必要な GPT ブート情報が含まれています。パーティション 8 と同様に、パーティション 0 は管理を必要としないため、変更しないようにしてください。ルートファイルシステムは、パーティション 1 に含まれています。

注記 - EFI ラベルの詳細については、『Oracle Solaris 11.3 でのデバイスの管理』 の「EFI (GPT) ディスクラベル」を参照してください。

---

x86 ベースのシステムでは、有効な 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。Oracle Solaris `fdisk` パーティションの作成または変更について詳しくは、『Oracle Solaris 11.3 でのデバイスの管理』 の 「ディスクの構成」を参照してください。

ディスク名は一般に、`/dev/dsk/cNtNdN` の命名規則に従います。サードパーティー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には、手動でラベルを付け、そのディスクを ZFS に割り当てる必要があります。

ディスクは、フルパスか、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を使用して指定できます。次の例は、有効なディスク名を示しています。

- `c1t0d0`
- `/dev/dsk/c1t0d0`
- `/dev/tools/disk`

ZFS ストレージプール内のファイルを使用する

ZFS では、ファイルをストレージプール内の仮想デバイスとして使用できます。この方法を採用する場合は、すべてのファイルが完全パスとして指定され、サイズが少なくとも 64M バイトあることを確認してください。

この機能は、物理デバイスが不足している場合により複雑な ZFS 構成を実験するなどのテストに役立ちます。この機能を本番環境で使用しないでください。

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、別の ZFS プール上に作成されたファイルまたはボリュームによってバックアップされる ZFS プールを作成すると、システムのデッドロックまたはパニックが発生することがあります。

ZFS ストレージプールの冗長性機能

ストレージプールは、ZFS 冗長性を使用して構成するようにします。冗長性がないと、データを失うリスクは大きくなります。さらに、ZFS 冗長性がないと、プールはデータ不整合を報告することしかできず、これらの不整合を修復できません。ZFS は、ミラー化構成と RAID-Z 構成でデータ冗長性および自己修復プロパティーを提供します。

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。ミラー化構成は単純にすることも、各プール内に複数のミラーが存在するような複雑な構成にすることもできます。

単純または複雑なミラー化ストレージプールの作成については、32 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、次の耐障害性レベルを備えた RAID-Z 構成をサポートしています。

- シングルパリティー (`raidz` または `raidz1`) – RAID-5 に似ています。
- ダブルパリティー (`raidz2`) – RAID-6 に似ています。
- トリプルパリティー (`raidz3`) – 詳細は、`http://blogs.oracle.com/ahl/entry/triple_parity_raid_z` を参照してください。

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、ファイルシステムのメタデータに、可変幅の RAID ストライプを処理するためのベースとなるデータ冗長性モデルに関する十分な情報が含まれるようにしています。それにより、RAID-Z では、RAID-5の書き込みホールの問題などの、従来の RAID アルゴリズムで発生する問題が回避されます。

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し、不正なデータを正常なコピーに置き換えることによって修復します。

サイズが x のディスクを n 個備えた、p 個のパリティーディスクを含む RAID-Z 構成は、約 (n-p)*x バイトを保持でき、データの整合性が危険にさらされるまでに p 個のデバイスの障害に耐えることができます。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 (以下同様) 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。

ミラー化構成と同様に、RAID-Z 構成は単純にすることも、複雑にすることもどちらも可能です。

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。1 桁のディスクをグループ化した RAID-Z 構成の方が、一般にパフォーマンスは向上します。

詳細は、次のソースを参照してください。

- 32 ページの「RAID-Z ストレージプールを作成する」 – RAID-Z ストレージプールの作成に関する情報を提供します。
- `http://blogs.oracle.com/roch/entry/when_to_and_not_to`: – パフォーマンスやディスク容量の考慮事項に基づいてミラー化構成または RAID-Z 構成のどちらを選択するかに関するガイドラインを提供します。
- 第13章「Oracle Solaris ZFS の推奨されるプラクティス」 – RAID-Z ストレージプールの追加の推奨事項について説明しています。

ZFS ハイブリッドストレージプール

Oracle の Sun Storage 7000 製品シリーズで使用可能な ZFS ハイブリッドストレージプールは、DRAM、SSD、および HDD を組み合わせることによって、パフォーマンスの向上と容量の増加を実現しながら、消費電力を削減します。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。

詳細は、http://www.oracle.com/technetwork/documentation/old-unified-ss-1882427.html のドキュメントを参照してください。

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。データを配置する場所に関する決定は書き込み時に行われるため、固定幅ストライプは割り当て時に作成されません。

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 の効率的な使用を保証するには、同じタイプの最上位レベルの仮想デバイスを、各デバイスの冗長性レベルを同じにして使用します。2 方向のミラー構成と 3 方向の RAID-Z 構成を使用するなど、同じプール内で異なるタイプの仮想デバイス組み合わせることはしないでください。

♦ ♦ ♦ 第 2 章

Oracle Solaris ZFS の概要

この章では、基本的な Oracle Solaris ZFS 構成の設定に役立つ情報を提供します。以降の章では、さらに詳細な情報を提供します。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、次の情報について説明します。

ハードウェアおよびソフトウェアの要件

ZFS ソフトウェアを使用する前に、次のハードウェア要件を満たしていることを確認します。

- サポート対象の Oracle Solaris リリースを実行している SPARC® または x86 ベースのシステム。
- 7G バイト - 13G バイトの間のディスク領域。ZFS がディスク領域を使用する方法の詳細は、79 ページの「ZFS ルートプールの容量要件」を参照してください。
- 作業負荷をサポートできる十分なメモリー。
- ミラー化プール構成に対する複数のコントローラ。

さらに、ZFS 管理タスクを実行するには、次のいずれかのプロファイルが割り当てられた役割を引き受ける必要があります。

- 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

スーパーユーザー (`root`) アカウントを使用して ZFS を構成できますが、RBAC (役割によるアクセス制御) の役割を使用する方法が最適な方法です。役割の作成および割

り当ての詳細については、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』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10章「Oracle Solaris ZFS 委任管理」を参照してください。

ZFS 実装の計画

このセクションでは、ZFS を構成する前に考慮する必要がある要因について説明します。

ZFS コンポーネントの命名

データセットやプールなど、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、英数字および次の特殊文字だけです。
  - 下線 (_)
  - ハイフン (-)
  - コロン (:)
  - ピリオド (.)
  - 空白 (" ")

  注記 - タブとその他の空白は有効ではありません。
- プール名の先頭は文字である必要があり、プール名に含めることができるのは、英数字、下線 (_)、ダッシュ (-)、およびピリオド (.) のみです。プール名に関する次の制限事項に注意してください。
  - `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。

- コンポーネント名を空にすることはできません。

ストレージ要件の識別

プールはストレージの物理的特性を示します。ファイルシステムを作成する前に、プールを作成する必要があります。

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します。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たディスク上の個々のスライスではなくディスク全体を ZFS に割り当てます。

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、17 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。

データ冗長性のタイプの選択

ZFS は、データ冗長性のタイプを複数サポートしています。このタイプによって、プールが耐えることのできるハードウェア障害のタイプが決定されます。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。

ZFS 冗長性機能の詳細は、19 ページの「ZFS ストレージプールの冗長性機能」を参照してください。

ZFS ファイルシステム階層の決定

ファイルシステム階層はプールに作成されます。

階層を利用すれば、情報を簡単で、わかりやすくかつ機能的なメカニズムで編成できます。このセクションでは、階層の計画に関する固有の問題について説明します。

ファイルシステムの粒度の選択

ZFS では、階層に編成されたファイルシステムをサポートします。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 は、プロパ

ティーの継承をサポートします。そのため、階層を使用することによって、ファイルシステムのツリー全体でプロパティーをすばやく簡単に設定できます。

ZFS ファイルシステムを 1 つの場所で集中的に管理できます。1 つ 1 つのシステムは軽量であるため、ユーザーまたはプロジェクトごとにファイルシステムを確立できます。このモデルを使用することによって、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単位またはプロジェクト単位で制御できます。

ファイルシステムの管理方法の詳細については、第7章「Oracle Solaris ZFS ファイルシステムの管理」を参照してください。

ファイルシステムのグループ化

一般的な名前で類似した ZFS ファイルシステムを階層にグループ化できます。階層は、ファイルシステムとそのプロパティーを集中的に管理および制御する場所になります。

例1「ミラー化 ZFS ファイルシステムの構成」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置かれています。

ファイルシステムプロパティーの選択

ファイルシステムのほとんどの特性プロパティーはユーザーが決定します。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。

プロパティーの詳細については、105 ページの「ZFS のプロパティーの概要」を参照してください。

♦ ♦ ♦ 第 3 章

3 Oracle Solaris ZFS ストレージプールを作成および破棄する

この章では、Oracle Solaris で ZFS ストレージプールを作成および破棄する方法について説明します。内容は次のとおりです。

ZFS ストレージプールを作成する

このセクションでは、ストレージプールのいくつかの構成方法について説明します。ルートプールの詳細については、第6章「ZFS ルートプールの管理」を参照してください。

ストレージプールを作成する場合は、プールの仮想デバイスを構成します。仮想デバイスは、ストレージプールを作成するために使用され、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するディスクデバイスまたはファイルの内部表現です。プールでは、構成の最上位に任意の数の仮想デバイス (プールの最上位レベル *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、その構成はミラーデバイスまたは RAID-Z 仮想デバイスとしてのデータ冗長性を備えています。冗長性の利点を活かすために、冗長なストレージプールを作成するようにしてください。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。

冗長な構成を使用する場合でも、非エンタープライズグレードのハードウェアに、プールデータの通常のバックアップをスケジュールしてください。ZFS 冗長性を備えたストレージプールは、ハードウェア障害、電源障害、またはケーブルの切断による影響を免れません。通常のバックアップの実行によって、エンタープライズにデータ保護の別のレイヤーが追加されます。

ストレージプールを作成したら、次のコマンドを使用してその情報を表示できます。

```
# zpool status pool
```

`zpool status` コマンドで使用できるオプションの詳細は、60 ページの「ZFS ストレージプールのステータスのクエリー検索を行う」を参照してください。

ストレージプールの作成時に次の制限を守ってください。

- 既存のストレージプールの一部であるディスクのパーティションやラベルを変更しないでください。それ以外の場合は、OS を再インストールしなければならない場合があります。
- 別のストレージプールのコンポーネント (ファイルやボリュームなど) を含むストレージプールを作成しないでください。そのような構成はデッドロックを引き起こす可能性があります。
- システム全体で共有するプールを作成しないでください。そのような構成はサポートされていません。ZFS はクラスタファイルシステムではありません。

▼ システム上の ZFS を設定する方法

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は、23 ページの「ハードウェアおよびソフトウェアの要件」を参照してください。

2. **ZFS プールを作成します。**

   ```
   # zpool create pool keyword devices [keyword devices]
   ```

   pool
   ZFS プールの名前。プール名は、24 ページの「ZFS コンポーネントの命名」に記載されている命名規則に従う必要があります。

   keyword
   さらに、冗長性のタイプなどのプール構成、またはログデバイスやキャッシュが使用されるかどうかを指定します。

   冗長性タイプには、ミラー化構成のためのキーワード `mirror` か、あるいは RAID-Z 構成のためのキーワード、つまり、必要なパリティーに応じて、シングルパリティーの場合は `raidz` または `raidz1`、ダブルパリティーの場合は `raidz2`、トリプルパリティーの場合は `raidz3` のいずれかを使用します。

   devices
   プールに割り当てられるデバイスを指定します。使用中のデバイスを指定することや別のファイルシステムを含めることはできません。それらを指定すると、プールの作成は失敗します。

   デバイスの使用方法を決定する方法の詳細は、36 ページの「アクティブに使用されているデバイス」を参照してください。

3. **(オプション) システム上の ZFS プールのリストを表示します。**

```
# zpool list
```

4. **(オプション) プールのステータスを表示します。**

```
# zpool status pool
```

5. **ファイルシステム階層を構築します。**

   a. **基本ファイルシステムを作成します。**
   基本ファイルシステムは、後から作成する個々のファイルシステムのコンテナとして機能します。

   ```
   # zfs create pool/filesystem
   ```

   ここで、*filesystem* はファイルシステムの名前です。
   ファイルシステム名をコマンドで使用する場合は、常に階層の完全なパス (*pool*/*filesystem*) を含める必要があります。このルールは作成する後続の子ファイルシステムにも適用されます。

   b. **子ファイルシステムによって共有されるプロパティーを設定します。**

   ```
   # zfs set property=value pool/filesystem
   ```

   複数のシステムプロパティーを設定できます。

ヒント - 次の構文を使用すると、ファイルシステムの作成とそのプロパティーの設定を同時に実行できます。

```
# zfs create -o property=value [-o property=value] pool/filesystem
```

---

   c. **基本ファイルシステムの下にグループ化する個々のファイルシステムを作成します。**

   ```
   # zfs create pool/filesystem/fs1
   # zfs create pool/filesystem/fs2
   ...
   ```

   ここで、*fs1*、*fs2* などは、個別のファイルシステムを表します。

   d. **(オプション) 個々のファイルシステムに固有のプロパティーを設定します。**

   ```
   # zfs set property=value pool/filesystem/fs1
   ```

6. **(オプション) 最終結果を表示します。**

```
# zpool list
```

プールのステータスを確認する方法の詳細については、60 ページの「ZFS ストレージプールのステータスのクエリー検索を行う」を参照してください。

例 1　ミラー化 ZFS ファイルシステムの構成

次の例では、基本的な ZFS 構成は次の仕様でシステム上に作成されます。

- 2 台のディスク (`c1t0d0` と `c2t0d0`) が ZFS ファイルシステムに割り当てられます。
- プール `system1` はミラー化を使用します。
- ファイルシステム `home` がプール上に作成されます。
- `mountpoint`、`share.nfs`、および `compression` のプロパティーは `home` に設定されています。
- 2 つの子ファイルシステム `user1` および `user2` は `home` に作成されます。
- quota は `user2` に設定されます。このプロパティーは、プール全体の使用可能なディスク領域に関係なく `user2` が使用できるディスク領域を制限します。

例で使用されているコマンド `zfs get` はファイルシステムプロパティーを表示します。

```
# zpool create system1 mirror c1t0d0 c2t0d0
# zpool list
NAME       SIZE     ALLOC     FREE     CAP     HEALTH     ALTROOT
system1     80G     137K       80G      0%     ONLINE     -

# zpool status system1

pool: system1
state: ONLINE
scrub: none requested
config:

NAME          STATE     READ  WRITE  CKSUM
system1       ONLINE       0      0      0
   mirror-0   ONLINE       0      0      0
      c1t0d0  ONLINE       0      0      0
      c2t0d0  ONLINE       0      0      0

errors: No known data errors

# zfs create system1/home
# zfs set mountpoint=/export/zfs system1/home
# zfs set share.nfs=on system1/home
# zfs set compression=on system1/home

# zfs get compression system1/home
NAME             PROPERTY       VALUE     SOURCE
system1/home     compression    on        local

# zfs create system1/home/user1
# zfs create system1/home/user2

# zfs set quota=10G system1/home/user2

# zfs list
NAME                     USED  AVAIL  REFER  MOUNTPOINT
system1                 92.0K  67.0G   9.5K  /system1
system1/home            24.0K  67.0G     8K  /export/zfs
system1/home/user1         8K  67.0G     8K  /export/zfs/user1
```

```
system1/home/user2          8K  10.0G     8K  /export/zfs/user2
```

例 2 RAID-Z ZFS ファイルシステムの構成

この例では、RAID-Z ファイルシステムを作成し、それらの短縮形のデバイス名または完全なデバイス名のいずれかを使用してディスクを指定する方法を示します。ディスク `c6t0d0` は `/dev/dsk/c6t0d0` と同じです。

- 3 台のディスク `c4t0d0`、`c5t0d0`、および `c6t0d0` が ZFS ファイルシステムに割り当てられます。
- プール `rdpool` は RAID-Z シングルパリティー構成を使用します。
- ファイルシステム `base` はプール上に作成されます。

```
# zpool create rdpool raidz c4t0d0 c5t0d0 /dev/dsk/c6t0d0
# zpool list
NAME       SIZE     ALLOC     FREE     CAP     HEALTH     ALTROOT
rdpool     120G     205K      120G      0%     ONLINE     -

# zpool status -v rdpool
  pool: rdpool
 state: ONLINE
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        rdpool      ONLINE       0     0     0
          raidz-0   ONLINE       0     0     0
            c4t0d0  ONLINE       0     0     0
            c5t0d0  ONLINE       0     0     0
            c6t0d0  ONLINE       0     0     0

errors: No known data errors

# zfs create rdpool/base
# zfs set mountpoint=/export/zfs rdpool/base
# zfs set share.nfs=on rdpool/base
# zfs set compression=on rdpool/base

# zfs get compression rdpool/home
NAME               PROPERTY       VALUE     SOURCE
rdpool/base        compression    on        local

# zfs create rdpool/base/user1
# zfs create rdpool/base/user2

# zfs set quota=10G rdpool/base/user2

# zfs list
NAME                   USED  AVAIL  REFER  MOUNTPOINT
rdpool                  92.0K  67.0G   9.5K  /rdpool
rdpool/base             24.0K  67.0G     8K  /export/zfs
rdpool/base/user1          8K  67.0G     8K  /export/zfs/user1
rdpool/base/user2          8K  10.0G     8K  /export/zfs/user2
```

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` キーワードを使用します。複数のミラーを構成するには、コマンド行でキーワードを繰り返します。次のコマンドは 2 つの最上位仮想デバイスでプール `system1` を作成します。

```
zpool create system1 mirror c1d0 c2d0 mirror c3d0 c4d0
```

仮想デバイスは両方とも 2 方向のミラーです。データは、両方のミラー間で動的にストライプ化され、データは各ディスク間で適切に冗長となります。

推奨のミラー化構成の詳細については、第13章「Oracle Solaris ZFS の推奨されるプラクティス」を参照してください。

ZFS ミラー化構成で次の操作を実行できます。

- 異なるディスクセットで最上位レベルの別の仮想デバイスを追加します。41 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。
- 追加ディスクを接続します。44 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。
- ディスクを交換します。50 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。
- ディスクを切り離します。44 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。
- 新しい同一のプールを作成するためにミラー化構成を分割します。46 ページの「ミラー化されたストレージプールの分割による新しいプールの作成」を参照してください。

ミラー化されたストレージプールで ZFS を構成する方法の例については、例1「ミラー化 ZFS ファイルシステムの構成」を参照してください。

RAID-Z ストレージプールを作成する

RAID-Z 構成でストレージプールを作成するには、プールに使用するパリティーに基づいて RAID-Z キーワードの 1 つを使用します。

- シングルパリティー構成の場合、`raidz` または `raidz1`。
- ダブルパリティー構成の場合、`raidz2`。
- トリプルパリティー構成の場合、`raidz3`。

複数の RAID-Z 最上位レベルの仮想デバイスを作成するには、コマンド行でキーワードを繰り返します。次のコマンドは 1 つの最上位仮想デバイスでプール `rdpool` を作

成します。仮想デバイスは、9 個のディスクで構成されるトリプルパリティー RAID-Z 構成です。

```
zpool create rdpool raidz3 c0t0d0 c1t0d0 c2t0d0 c3t0d0 c4t0d0 \ c5t0d0
c6t0d0 c7t0d0 c8t0d0
```

RAID-Z ストレージプールで ZFS を構成する方法の例については、例2「RAID-Z ZFS ファイルシステムの構成」を参照してください。

ZFS RAID-Z 構成で次の操作を実行できます。

- 異なるディスクセットで最上位レベルの別の仮想デバイスを追加します。41 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。
- ディスクを交換します。50 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。

RAID-Z 構成で次の操作は実行できません。

- 追加ディスクを接続します。
- ディスクを切り離す。スペアディスクでディスクを交換する場合、またはスペアディスクを切り離す必要がある場合は除きます。
- ログデバイスまたはキャッシュデバイスではないデバイスを削除します。

ログデバイスを持つ ZFS ストレージプールを作成する

ZFS インテントログ (ZIL) は、同期トランザクションの POSIX 要件を満たしています。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 はメインプール内のブロックから割り当てられます。ただし、NVRAM や専用ディスクなど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。別個のログデバイスをデプロイすることによりパフォーマンスは向上しますが、デバイスタイプ、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応じて向上します。予備のパフォーマンス情報については、`http://blogs.oracle.com/perrin/entry/slog_blog_or_blogging_on` を参照してください。

RAID-Z ログデバイス用にではなく、冗長性のためだけにミラー化ログデバイスを構成できます。ミラー化されていないログデバイスで障害が発生した場合、ログブロッ

クの格納はストレージプールに戻されます。より大きなストレージプールの一部としてログデバイスの追加、交換、除去、接続、切り離し、インポート、およびエクスポートを実行できます。

ZFS ログデバイスに関する次の点を考慮してください。

- 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ログに記録されるデータは比較的少量です。ログのトランザクションまたは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半分にします。これは、格納できる潜在的なログ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。

ログデバイスでストレージプールを作成するには、`log` キーワードを使用します。次の例は、ミラー化ログデバイスで `datap` と呼ばれるミラー化ストレージプールを構成する方法を示します。

```
# zpool create datap mirror c0t5000C500335F95E3d0 c0t5000C500335F907Fd0 \
   mirror c0t5000C500335BD117d0 c0t5000C500335DC60Fd0 \
   log mirror c0t5000C500335E106Bd0 c0t5000C500335FC3E7d0

# 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
      c0t5000C500335F95E3d0 ONLINE       0      0      0
      c0t5000C500335F907Fd0 ONLINE       0      0      0
   mirror-1                 ONLINE       0      0      0
      c0t5000C500335BD117d0 ONLINE       0      0      0
      c0t5000C500335DC60Fd0 ONLINE       0      0      0
   logs
   mirror-2                 ONLINE       0      0      0
      c0t5000C500335E106Bd0 ONLINE       0      0      0
      c0t5000C500335FC3E7d0 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

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。これらのデバイスによって、ほぼ静的なコンテンツをランダムに読み取る作業負荷のパフォーマンスが大幅に向上します。

キャッシュデバイスを使用してストレージプールを構成するには、cache キーワードを使用します。次に例を示します。

```
# zpool create system1 mirror c2t0d0 c2t1d0 c2t3d0 cache c2t5d0 c2t8d0
# zpool status system1
pool: system1
state: ONLINE
scrub: none requested
config:

NAME           STATE     READ  WRITE  CKSUM
system1        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
```

例6「キャッシュデバイスの追加」に示されているように、作成中または作成後のいずれかに、プール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。ただし、ミラー化されたキャッシュデバイスを作成する、またはそれらを RAID-Z 構成の一部として作成することはできません。

注記 - 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。

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が容量の上限に達するまでの時間は、サイズによって異なります。次の例に示すように、`zpool iostat` コマンドを使用して、容量および読み取りをモニターします。

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

`zpool iostat` コマンドの詳細は、63 ページの「ZFS ストレージプールの入出力統計を表示する」を参照してください。

ストレージプール作成のドライランを行う

テスト目的の場合は、デバイスに実際に書き込まずに、プールの作成をシミュレーションできます。`zpool create -n` コマンドは、デバイスの使用中チェックと冗長性レベルの検証を実行し、そのプロセスで発生したエラーをすべて報告します。エラーが見つからなかった場合は、次の例のような出力が表示されます。

```
# zpool create -n system1 mirror c1t0d0 c1t1d0
```

```
would create 'system1' with the following layout:

system1
  mirror
    c1t0d0
    c1t1d0
```

注意 - 同じ構成に同じデバイスを 2 回指定したなどの一部のエラーは、プールを実際に作成しないと検出できません。そのため、実際のプール作成は、ドライランが成功した場合でも失敗する可能性があります。

ZFS ストレージプールの作成に関する問題への対応

このセクションでは、プール作成中のエラーの説明をデバイス、冗長性、またはマウントポイントに関連したエラーでグループ化しています。

一部のエラーメッセージでは、`-f` オプションを使用して報告されたエラーをオーバーライドすることが勧められます。しかし、通常、エラーはオーバーライドせずに、修復する必要があります。

アクティブに使用されているデバイス

特定のデバイスで ZFS プールを作成すると、ZFS は、これらのデバイスが ZFS 自体またはオペレーティングシステムのほかの部分によって使用されているかを最初に判別します。デバイスが使用されている場合、該当するエラーメッセージが表示されます。

プールの作成をもう一度試行する前に、次のメッセージで報告されるエラーを手動で訂正する必要があります。

`Mounted file system`

ディスクには、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。

`File system in /etc/vfstab`

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。

`Dedicated dump device`

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。

`Part of a ZFS pool`

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ます。そのプールが引き続き必要な場合、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ちます。プールを作成する -f オプションを使用すると、次の警告をオーバーライドできます。

`Contains a file system`

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か、使用されていません。

`Part of volume`

ディスクは Solaris Volume Manager ボリュームの一部です。

`Part of exported ZFS pool`

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 potentially active として報告されます。このディスクが別のシステムで使用されているネットワークに接続されたドライブである可能性があるためです。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。

冗長性レベルが一致しない

冗長性レベルの異なる仮想デバイスでプールを作成すると、次の例のようなエラーメッセージが表示されます。

```
# zpool create system1 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
```

異なるサイズのデバイスを使用してミラー化または RAID-Z プールを作成する場合に、同様のエラーメッセージが生成されます。

レベルが一致しない冗長性を維持すると、大きいデバイスで未使用のディスク領域が発生し、ZFS の使用が非効率的になります。これらのエラーはオーバーライドするのではなく、訂正する必要があります。

空でない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、最上位ファイルシステムのデフォルトマウントポイントは */pool-name* になります。このディレクトリが存在し、そこにデータが含まれる場合は、エラーが発生します。

異なるデフォルトのマウントポイントでプールを作成するには、`zpool create -m` *mountpoint* コマンドを使用します。例:

```
# zpool create system1 c1t0d0
default mountpoint '/system1' exists and is not empty
use '-m' option to provide a different default
# zpool create -m /export/zfs system1 c1t0d0
```

このコマンドを実行すると、`/export/zfs` をマウントポイントで新しいプール `system1` および `system1` ファイルシステムが作成されます。

マウントポイントの詳細については、131 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

プールにマウントされたデータセットが含まれている場合でも、`zpool destroy` *pool* コマンドを使用して、プールを破棄できます。

注意 - ZFS は、どのデバイスが使用中であるかを常に追跡することはできません。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。誤って違うプールを破棄してしまった場合、75 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。

プールを破棄しても、そのプールをインポートできます。そのため、機密データがプールの一部であったディスクに残る可能性があります。データを完全に破棄するには、破棄されたプールのすべてのディスクに対して `format` ユーティリティーの `analyze->purge` オプションなどの機能を使用します。

データの機密性を確保するには、暗号化された ZFS ファイルシステムを作成します。破棄したプールが回復されても、データは暗号鍵なしではアクセスできない状態の

ままになります。詳細については、154 ページの「ZFS ファイルシステムの暗号化」を参照してください。

使用可能デバイスと使用不可デバイスが混在するプールが破棄された場合、データは使用可能なディスクに書き込まれ、プールが有効ではなくなることが示されます。この状態情報は、インポートを実行するときに、デバイスが潜在的なプールとして表示されることを防ぎます。使用不可デバイスでも、プールを破棄できます。使用不可デバイスを修復すると、新しいプールの作成時に `potentially active` として報告されます。また、インポートするためにプールを検索する際に有効なデバイスとして表示されます。

使用不可デバイスの数が多くなると、プール自体が使用不可になることがあります。プールの最上位レベルの仮想デバイスの状態が `UNAVAIL` として報告されます。この場合、`zpool destroy -f` コマンドを使用することによってのみ、そのプールを破棄できます。

プールとデバイスの健全性の詳細については、66 ページの「ZFS ストレージプールの健全性ステータスを調べる」を参照してください。

♦ ♦ ♦ 第 4 章

Oracle Solaris ZFS ストレージプール内のデバイスを管理する

この章では、システム上の ZFS プールに使用する物理デバイスを管理するために実行できるさまざまなタスクを説明します。次のセクションで構成されています。

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。

追加する仮想デバイスは、既存の仮想デバイスと同じレベルの冗長性を持つ必要があります。ただし、`-f` オプションを使用することで冗長性のレベルを変更できます。

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。

```
# zpool add pool keyword devices
```

注記 - `zpool add -n` を使用すると、実際にデバイスを追加する前にドライランを実行できます。

例 3 ZFS ミラー化構成にディスクを追加する

次の例では、ミラーは、2 つの最上位ミラー化デバイスで構成される ZFS 構成に追加されます。

```
# zpool add mpool mirror c0t3d0 c1t3d0
# zpool status mpool
pool: mpool
state: ONLINE
scrub: none requested
config:

NAME           STATE     READ  WRITE  CKSUM
mpool          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     ミラー化されたデバイスを追加。
      c0t3d0   ONLINE       0      0      0
      c1t3d0   ONLINE       0      0      0

errors: No known data errors
```

例 4 RAID-Z 構成にディスクを追加する

この例では、3 台のディスクで構成される 1 台の RAID-Z デバイスを、3 台のディスクで構成される既存の RAID-Z ストレージプールに追加する方法を示します。

```
# 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
      c1t2d0   ONLINE       0      0      0
      c1t3d0   ONLINE       0      0      0
      c1t4d0   ONLINE       0      0      0
   raidz1-1    ONLINE       0      0      0     RAID-Z デバイスを追加。
      c2t2d0   ONLINE       0      0      0
      c2t3d0   ONLINE       0      0      0
      c2t4d0   ONLINE       0      0      0

errors: No known data errors
```

例 5 ミラー化されたログデバイスを追加する

次の例は、ミラー化ログデバイスをミラー化ストレージプールに追加する方法を示しています。

```
# 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                                          ミラー化されたログデバイスを追加。
   mirror-1    ONLINE       0      0      0
      c0t6d0   ONLINE       0      0      0
      c0t7d0   ONLINE       0      0      0

errors: No known data errors
```

ミラー化ログデバイスは、この例にある `mirror-1` などの識別子で示されます。例 7「ミラー化されたログデバイスを削除する」で示されているように、識別子はログデバイスを削除するときに役立ちます。

例 6 キャッシュデバイスの追加

次の例は、キャッシュデバイスをプールに追加する方法を示しています。

```
# zpool add system1 cache c2t5d0 c2t8d0
# zpool status system1
pool: system1
state: ONLINE
scrub: none requested
config:

NAME           STATE     READ  WRITE  CKSUM
system1        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                                      キャッシュデバイスを追加。
      c2t5d0   ONLINE       0      0      0
      c2t8d0   ONLINE       0      0      0

errors: No known data errors
```

ストレージプールからのデバイスの削除

例 7 ミラー化されたログデバイスを削除する

この例は、例5「ミラー化されたログデバイスを追加する」で作成されたログデバイス `mirror-1` を削除する方法を示します。ログデバイスに冗長性がない場合、`c0t6d0` などのデバイス名を参照してデバイスを削除します。

```
# 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
```

例 8 キャッシュデバイスの削除

この例は、例6「キャッシュデバイスの追加」で作成されたキャッシュデバイスを削除する方法を示します。

```
# zpool remove system1 c2t5d0 c2t8d0
# zpool status system1
pool: system1
state: ONLINE
scrub: none requested
config:

NAME           STATE     READ  WRITE  CKSUM
system1        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
```

ストレージプール内でデバイスを接続する/切り離す

既存の仮想デバイスに新しいデバイスを追加するには、次のコマンドを使用します。

```
# zpool attach pool existing-device new-device
```

次のいずれかの条件が適用される場合に、zpool detach コマンドを使用してデバイスを切り離すことができます。

- デバイスがミラー化されたプール構成に属している。
- RAID-Z 構成で、切り離されたデバイスがほかの物理デバイスまたはスペアで置き換えられる。

これらの条件が当てはまらない場合に、デバイスを切り離すと次のようなエラーが生成されます。

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

次の例は、zfs attach コマンドを適用する方法を示します。

例 9 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、mpool は既存の 2 方向のミラープールです。新しいデバイス c2t1d0 を既存のデバイス c1t1d0 に接続することによって、3 方向ミラープールに変換されます。新しく接続されたデバイスは、ただちに再同期化されます。

```
# zpool attach mpool c1t1d0 c2t1d0
# zpool status mpool
pool: mpool
state: ONLINE
scrub: resilver completed after 0h0m with 0 errors on Fri Jan  8 12:59:20 2010
config:

NAME           STATE     READ  WRITE  CKSUM
mpool          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     接続されたデバイスが 3 方向ミラープールを作成

errors: No known data errors
```

例 10 非冗長なストレージプールからミラー化されたストレージプールへの変換

zpool attach コマンドを使用すると、非冗長から冗長な構成にストレージプールまたはログデバイスを変換できます。

次の例は、冗長なプールに変換する前後の非冗長プール system1 のステータスを示します。

```
# zpool status system1
pool: system1
state: ONLINE
scrub: none requested
config:
NAME           STATE      READ  WRITE  CKSUM
system1        ONLINE        0      0      0
     c0t1d0    ONLINE        0      0      0
```

```
errors: No known data errors
# zpool attach system1 c0t1d0 c1t1d0
# zpool status system1
pool: system1
state: ONLINE
scrub: resilver completed after 0h0m with 0 errors on Fri Jan  8 14:28:23 2010
config:

NAME           STATE     READ  WRITE  CKSUM
system1        ONLINE       0      0      0     プールがミラー化される
   mirror-0    ONLINE       0      0      0
      c0t1d0   ONLINE       0      0      0
      c1t1d0   ONLINE       0      0      0  73.5K resilvered

errors: No known data errors
```

ミラー化されたストレージプールの分割による新しいプールの作成

`zpool split` コマンドを使用することによって、ミラー化された ZFS ストレージプールをすばやくクローニングできます。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。その後、同じシステムまたは別のシステムのいずれかに新しいプールをインポートできます。インポートツールの詳細については、72 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。

プールを分割するには、次のコマンドを使用します。

```
# zpool split pool new-pool [device]
```

デバイスを指定しないかぎり、`zpool split` コマンドはデフォルトで、新しく作成されるプールのために、プールの仮想デバイスの最後のディスクを切り離します。プールに複数の最上位レベルの仮想デバイスが存在する場合、これらのディスクから新しいプールを作成するため、このコマンドは各仮想デバイスからディスクを切り離します。

プールの分割は、ミラー化構成にのみ適用されます。RAID-Z で構成されたプールまたは非冗長プールは分割できません。

3 台のディスクで構成される単一の最上位レベルのデバイスのみが存在するプールを分割する場合、3 番目のディスクから作成した新しいプールは非冗長になります。残りのプールには、残りの 2 台のディスクによるデータの冗長性が保持されます。新しいプールを冗長な構成に変換するには、新しいデバイスをプールに接続します。

`zpool split` コマンドで ZFS プールを分割する方法の詳細な手順および例については、My Oracle Support (`https://support.oracle.com`) にログインし、「'zpool split' を使用してプールを分割する方法 (ドキュメント ID 1637715.1)」を参照してください。

ミラー化プールを分割する前に次の点を確認してください。

- データおよびアプリケーションの操作が静止されている。
- 進行中の再同期化がない。
- ハードウェアが正しく構成されている。ハードウェアのキャッシュフラッシュ設定の確認方法については、277 ページの「一般的なシステムプラクティス」を参照してください。

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ると、分割されたのと同じシステムにプールをインポートできるように、ディスクはプールから切り離されて新しいプール GUID が割り振られます。

分割されるプールのファイルシステム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。例:

```
# zpool split -R /system2 system1 system2
```

`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。

例 11 ミラー化された ZFS プールを分割する

次の例では、3 台のディスクから成る `poolA` というミラー化ストレージプールが分割されます。結果となる 2 つのプールは、2 台のディスクから成るミラー化プール `poolA` と、1 台のディスクから成る新しいプール `poolB` です。

```
# zpool status poolA
  pool: poolA
 state: ONLINE
  scan: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        poolA       ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c0t0d0  ONLINE       0     0     0
            c0t1d0  ONLINE       0     0     0
            c0t2d0  ONLINE       0     0     0

errors: No known data errors
# zpool split poolA poolB
# zpool import luna
# zpool status poolA poolB
  pool: luna
 state: ONLINE
  scan: none requested
config:
```

```
        NAME      STATE     READ WRITE CKSUM
        poolB     ONLINE       0     0     0
          c0t2d0  ONLINE       0     0     0

errors: No known data errors

  pool: poolA
 state: ONLINE
  scan: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        poolA       ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c0t0d0  ONLINE       0     0     0
            c0t1d0  ONLINE       0     0     0

errors: No known data errors
```

新しい構成では、ほかの操作を実行できます。たとえば、バックアップするために別のシステムに poolB をインポートできます。バックアップが完了したあとは、poolB を破棄し、poolA にディスクを再接続できます。

ストレージプールのデバイスをオフラインにするまたはオンラインに戻す

ストレージプールのデバイスが永続的に信頼できない状態になる、または完全に機能しなくなる場合、次のコマンドを使用してデバイスをオフラインにできます。

```
# zpool offline [option] pool device
```

ここで、*device* の名前には短縮名またはフルパスを指定できます。

デバイスをオフラインにすると、その OFFLINE 状態は永続的になります。非永続的な OFFLINE 状態の場合、デバイスを一時的にオフラインにする -t オプションを使用します。システムをリブートすると、デバイスは自動的に ONLINE 状態に戻ります。

注意 - プール自体が使用できなくなった場合などは、デバイスをオフラインにしないでください。たとえば、raidz1 構成内の 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。次のエラーメッセージが表示されます。

```
cannot offline c0t5000C500335F95E3d0: no valid replicas
```

OFFLINE 状態にしても、デバイスはプールから切り離されません。したがって、そのデバイスを別のプールに使用することはできません。それ以外の場合、次の例に似たエラーメッセージが生成されます。

device is part of exported or potentially active ZFS *pool*. Please see zpool(8)

別のプールでデバイスを使用するには、まずデバイスを ONLINE 状態に戻してから、デバイスが属するプールを破棄します。

プールを破棄しない場合は、オフラインにしたデバイスを同等のデバイスで置き換えます。置き換えられたデバイスは、別のプールで使用できるようになります。

注記 - デバイスを置き換えるために、そのデバイスをオフラインにする必要はありません。

デバイスをオンラインに戻すには、次のコマンドを使用します。

```
# zpool online [option] pool device
```

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。

状態が UNAVAIL であるデバイスをオンラインにしようとすると、次の例のような障害が発生したデバイスに関するメッセージが表示されます。

```
warning: device 'device' onlined, but remains in faulted state
use 'zpool clear' to restore a faulted device
```

障害が発生したデバイス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。

障害があるデバイスの置き換えの詳細は、247 ページの「見つからないデバイスまたは削除されたデバイスを解決する」を参照してください。

LUN を拡張するには、zpool online -e コマンドを使用します。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。プロパティーが無効な場合、このコマンドを使用して LUN を自動的に拡張します。LUN がオフラインまたはオンラインであるかどうかに関係なくこのコマンドを実行できます。

ストレージプールデバイスのエラーをクリアーする

接続が一時的に失われるなどのプールデバイスの障害は、zpool status 出力でも報告されるエラーの原因になることがあります。そのようなエラーをクリアするには、次のコマンドを使用します。

```
# zpool clear pool [devices]
```

デバイスが指定された場合は、コマンドはデバイスに関連付けられたエラーのみをクリアします。それ以外の場合、コマンドはプール内のすべてのデバイスエラーをクリアします。

zpool エラーのクリアの詳細は、253 ページの「一時的または永続的なデバイスエラーをクリアする」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。

```
# zpool replace pool replaced-device [new-device]
```

冗長プールの同じ位置に交換用デバイスをインストールする場合は、置き換えられるデバイスのみを指定すればよい場合があります。一部のハードウェアでは、ZFS は同じ場所に新しいデバイスを認識します。ただし、別の場所に交換用デバイスをインストール場合、交換されるデバイスと新しいデバイスの両方を指定する必要があります。

置き換えられるデバイスの自動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。さらに、一部のハードウェアタイプは `autoreplace` プールプロパティーをサポートします。このプロパティーが有効な場合に、同じ場所でデバイスを置き換えると、自動的にフォーマットおよび置き換えが実行されます。`zpool replace` コマンドの実行は不要です。

次のガイドラインに留意してください。

- ホットスペアデバイス (使用可能な場合) は、システムの稼動中に削除されるデバイスを自動的に置き換えます。障害の発生したディスクを置き換えたあとに、`zpool detach` コマンドを使用してスペアの切り離しが必要になる場合があります。ホットスペアの切り離しについては、53 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。
- 削除してからふたたび挿入されたデバイスは自動的にオンラインにされます。この場合、置き換えは行われません。再挿入されたデバイスがオンラインに戻されると、交換用のホットスペアデバイスが自動的に削除されます。

デバイスをより大きい容量を持つデバイスで置き換える場合は、プールに新しいデバイスを追加したあと、そのデバイスは自動的にはフルサイズに拡張されません。`autoexpand` プールプロパティーが交換用 LUN の自動拡張を決定します。デフォルトでは、このプロパティーは無効です。より大きい LUN をプールに追加する前後に、このプロパティーを有効にできます。

SATA ディスクを備えた一部のシステムでは、オフラインにする前にディスクを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、このセクションの最初の例 (例12「ミラー化プール内のデバイスを置き換える」) で説明したように `zpool replace` コマンドを実行するだけでかまいません。SATA ディスクの置き換えの例については、例58「ZFS ストレージプール内の SATA ディスクを置き換える」を参照してください。

新しいディスクへのデータの再同期化のため、ディスクの置き換えに時間がかかります。ディスクの置き換えの合間に `zpool scrub` コマンドを実行して、交換用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。

デバイスの置き換えの詳細は、247 ページの「見つからないデバイスまたは削除されたデバイスを解決する」および 251 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。

▼ ストレージプールのデバイスを置き換える方法

1. **必要に応じて、`OFFLINE` 状態にデバイスを切り替えます。**

   # **zpool offline** *pool device*

2. **新しいデバイスでデバイスを物理的に置き換えます。**

   冗長なストレージプールの場合、交換用のデバイスがプール内のもっとも小さいディスク以上であることを確認します。

3. **`format` コマンドを実行します。**

   出力で次の点を確認します。

   - 新しいデバイスがリストされていることを確認します。
   - 交換用のデバイスが `WWN` としてマークされているかどうかを確認し、デバイス ID が変更されたことを確認します。

4. **プール内のデバイスを置き換えます。**

   新しいデバイスに新しい ID がある場合は、コマンドにその ID を含めます。

   # **zpool replace** *pool replaced-device* **[***new-device-ID***]**

   注記 - 複数のデバイスを置き換える場合、次のデバイスを置き換える前に各デバイスが完全に再同期化されていることを確認します。

5. **必要に応じて、デバイスをオンラインにします。**

   # **zpool online** *pool new-device*

6. **デバイスに障害があることを示すエラーが報告される場合は、FMA 手順を実行します。**

   a. **`fmadm faulty` コマンドを実行します。**

   b. **出力の `Affects:` セクションから仮想デバイスのプール名および GUID を識別します。**

   c. **次のコマンドを実行して、前の手順の情報を指定します。**

      # **fmadm repaired zfs://pool=***name***/vdev=***guid*

例 12 ミラー化プール内のデバイスを置き換える

次の例では、プール system1 内の 2 つの 16G バイトディスクは 2 つの 72G バイトディスクで置き換えられます。ディスクの交換後に autoexpand プロパティーを有効にして、ディスクをその最大サイズまで拡張します。

```
# zpool status system1
  pool: system1
 state: ONLINE
 scrub: none requested
config:

        NAME         STATE     READ WRITE CKSUM
        system1      ONLINE       0     0     0
          mirror     ONLINE       0     0     0
            c1t16d0  ONLINE       0     0     0
            c1t17d0  ONLINE       0     0     0

# zpool list system1
NAME      SIZE   ALLOC   FREE    CAP  HEALTH  ALTROOT
system1  16.8G   76.5K  16.7G     0%  ONLINE  -
# zpool replace system1 c1t16d0 c1t1d0
# zpool replace system1 c1t17d0 c1t2d0
# zpool list system1
NAME      SIZE   ALLOC   FREE    CAP  HEALTH  ALTROOT
system1  16.8G   88.5K  16.7G     0%  ONLINE  -
# zpool set autoexpand=on system1
# zpool list system1
NAME      SIZE   ALLOC   FREE    CAP  HEALTH  ALTROOT
system1  68.2G    117K  68.2G     0%  ONLINE  -
```

ストレージプール内のホットスペアの操作

ホットスペア機能を使って、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交換するために使用するディスクを指定できます。ホットスペアデバイスは、スペアが障害の発生したデバイスを置き換えるまで、プール内で非アクティブです。

ストレージプールでのホットスペアの指定

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。

- プールを作成するとき。

  # **zpool create** *pool keyword devices* **spare** *devices*

- プールが作成されたあと。

  # **zpool add** *pool* **spare** *devices*

ホットスペアを削除するには次のコマンドを使用します。

```
# zpool remove pool spare-device
```

注記 - 現在プールによって使用されているホットスペアは削除できません。

ホットスペアデバイスは、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。それよりも小さいスペアデバイスをホットスペアとして指定することもできます。しかし、そのデバイスで障害の発生したデバイスの置き換えがアクティブにされたときに、操作は次のエラーメッセージで失敗します。

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

複数のシステムによるアクセスが表示されている場合でも、複数のプールまたは複数のシステムでスペアを共有しないでください。1 つのシステムだけで複数あるプールのすべてを制御しなければならない場合は、それらのプール間で共有されるようにディスクを構成できます。ただし、この方法には危険が伴います。たとえば、共有スペアを使用しているプール A がエクスポートされる場合、プール A のエクスポート中に、プール B がそのスペアを知らずに使用する可能性があります。プール A がインポートされると、両方のプールが同じディスクを使用しているためデータが破損する可能性があります。

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

次の方法でホットスペアをアクティブにします。

- 手動の交換 - `zpool replace` コマンドを実行して障害が発生したデバイスを置き換えます。障害が発生したディスクを置き換えるために新しいデバイスが挿入される場合、スペアを切り離すことによって新しいデバイスをアクティブにします。
- 自動交換 - FMA エージェントは障害を検出し、スペアが使用できるかどうかを判別して、障害の発生したデバイスを自動的に置き換えます。ホットスペアは `UNAVAIL` 状態のデバイスも置き換えます。

  プールの `autoreplace` プロパティーを `on` 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。

ホットスペアを非アクティブにするには、次のアクションのいずれかを実行します。

- 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にホットスペアを切り離す。例 13「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」を参照してください。
- 別のホットスペアと一時的または永続的に交換する。例14「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」を参照してください。

例 13 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

この例では、次の構成を前提としています。

- system1 の mirror-1 構成で、ディスク c0t5000C500335BA8C3d0 が失敗しました。次の部分的な出力には、mirror-1 のステータスが表示されます。

```
# zpool status system1
.
  mirror-1                 DEGRADED     0     0     0
    c0t5000C500335BD117d0  ONLINE       0     0     0
    c0t5000C500335BA8C3d0  UNAVAIL      0     0     0     障害の発生したディスク
```

- 障害の発生したディスクを置き換えるために、プールのスペア c0t5000C500335E106Bd0 は自動的にアクティブにされます。
- 新しいデバイス c0t5000C500335DC60Fd0 で障害が発生したディスクを物理的に置き換えます。

この例では、新しいデバイスでのプールの再構成から開始します。最初に、zpool replace を実行して、削除するデバイスについて ZFS に通知します。次に、必要に応じて、zpool detach を実行してスペアを非アクティブにし、そのスペアをスペアプールに戻します。この例では、最後に、新しい構成のステータスを表示し、51 ページの「ストレージプールのデバイスを置き換える方法」のステップ 6に示されているように、障害の発生したデバイスに対して適切な FMA の手順を実行します。

```
# zpool replace system1 c0t5000C500335BA8C3d0
# zpool detach system1 c0t5000C500335E106Bd0
# zpool status system1
.
.
  mirror-1                 ONLINE       0     0     0
    c0t5000C500335BD117d0  ONLINE       0     0     0
    c0t5000C500335DC60Fd0  ONLINE       0     0     0     交換デバイス
  spares
    c0t5000C500335E106Bd0    AVAIL                       非アクティブにするスペア

# fmadm faulty
# fmadm repaired zfs://pool=name/vdev=guid
```

例 14 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

新しい交換用デバイスの代わりに、スペアデバイスを永続的な交換用として使用できます。この場合、単純に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をあとで修復する場合、そのディスクを新しい指定スペアとしてプールに追加できます。

この例では、例13「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」と同じ前提を使用します。

- プール system1 の mirror-1 構成は機能低下状態にあります。

```
# zpool status system1
.
  mirror-1                 DEGRADED     0     0     0
    c0t5000C500335BD117d0  ONLINE       0     0     0
    c0t5000C500335BA8C3d0  UNAVAIL      0     0     0     障害の発生したディスク
```

- 障害の発生したディスクを置き換えるために、プールのスペア `c0t5000C500335E106Bd0` は自動的にアクティブにされます。

この例は、スペアで置き換えられた障害が発生したディスクを切り離すことから開始します。

```
# zpool detach system1 c0t5000C500335BA8C3d0
# zpool status system1
.
.
  mirror-1                 ONLINE       0     0     0
    c0t5000C500335BD117d0  ONLINE       0     0     0
    c0t5000C500335E106Bd0  ONLINE       0     0     0     障害の発生したディスクをスペアに交換

errors: No known data errors
```

その後、スペアデバイスとしてプールに修復したディスクを追加し直します。障害の発生したデバイスに対して適切な FMA の手順を実行して、手順を完了します。

```
# zpool add system1 spare c0t5000C500335BA8C3d0
# zpool status system1
.
.
  mirror-1                 ONLINE       0     0     0
    c0t5000C500335BD117d0  ONLINE       0     0     0
    c0t5000C500335E106Bd0  ONLINE       0     0     0     以前のスペア
  spares
    c0t5000C500335BA8C3d0    AVAIL                       スペアとして修復されるディスク

errors: No known data errors

# fmadm faulty
# fmadm repaired zfs://pool=name/vdev=guid
```

♦ ♦ ♦ 第 5 章

5 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管理する方法について説明します。次のセクションで構成されています。

ZFS ストレージプールのプロパティーの管理

ZFS プールのプロパティーを管理するには、次のコマンドを使用します。

- `zpool get all` *pool* – プールのすべてのプロパティーをその対応する値とともに一覧表示します。
- `zpool get` *property pool* – プールの指定されたプロパティーをその対応する値とともに一覧表示します。
- `zpool set` *property=value pool* – プールの指定されたプロパティーに値を割り当てます。

いっぱいになっているプールにプールプロパティー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。

```
# zpool set autoreplace=on system1
cannot set property for 'system1': out of space
```

プール容量の問題の回避方法については、第13章「Oracle Solaris ZFS の推奨されるプラクティス」を参照してください。

表 2 ZFS プールのプロパティーの説明

プロパティー名	タイプ	デフォルト値	説明
`allocated`	文字列	N/A	物理的に割り当てられているプール内のストレージ領域の量を識別する読み取り専用の値。

プロパティー名	タイプ	デフォルト値	説明
altroot	文字列	off	代替ルートディレクトリを識別します。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次のような状況、つまり、不明なプールを検査しているときにマウントポイントが信頼できない場合や、標準的な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。off に設定されている場合は、zpool replace コマンドを使用して、デバイスの交換を開始する必要があります。on に設定されている場合は、このプールに以前属していたデバイスと同じ物理的な場所で見つかった新しいデバイスはすべて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。
bootfs	ブール型	N/A	ルートプールのデフォルトのブート可能ファイルシステムを識別します。このプロパティーは、通常、インストールプログラムによって設定されます。
cachefile	文字列	N/A	プール構成情報がキャッシュされる場所を制御します。システムのブート時に、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、このプロパティーは使用されません。
capacity	数値	N/A	使用されるプール領域の割合 (%) を識別する読み取り専用の値。 このプロパティーの省略名は cap です。
dedupditto	文字列	N/A	複製解除されたブロックの参照カウントのしきい値を設定します。カウントがこのしきい値を超えると、そのブロックの別の ditto コピーが自動的に格納されます。
dedupratio	文字列	N/A	プールに対して達成された読み取り専用の複製解除率。乗数として表されます。
delegation	ブール型	on	特権のないユーザーにファイルシステムに対して定義されているアクセス権を許可できるかどうかを制御します。詳細は、第10章「Oracle Solaris ZFS 委任管理」を参照してください。
failmode	文字列	wait	壊滅的なプール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。このようなイベントの動作は、次の値のいずれかによって決定されます。 ■ wait – デバイス接続が復元され、zpool clear コマンドを使用してエラーがクリアされるまで、このプールへのすべての入出力要求をブロックします。この状態では、

プロパティー名	タイプ	デフォルト値	説明
			このプールへの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ことがあります。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 – 新しいどの書き込み入出力要求にも EIO エラーを返しますが、残りのどの正常なデバイスに対する読み取りも許可します。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	N/A	割り当てられていないプール内のブロック数を識別する読み取り専用の値。
guid	文字列	N/A	このプールの一意の識別子を識別する読み取り専用プロパティー。
health	文字列	N/A	このプールの現在の健全性を識別する読み取り専用プロパティーであり、ONLINE、DEGRADED、SUSPENDED、REMOVED、UNAVAIL のいずれかです。
listshares	文字列	off	このプール内の共有情報が zfs list コマンドで表示されるようにするかどうかを制御します。デフォルト値は off です。
listsnapshots	文字列	off	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 コマンドで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になっている場合は、zfs list -t snapshot コマンドでスナップショット情報を表示できます。
readonly	ブール型	off	プールを変更できるかどうかを指定します。このプロパティーは、プールが読み取り専用モードでインポートされている場合にのみ有効になります。有効になっている場合、プールを読み取り/書き込みモードで再インポートするまで、インテントログにのみ存在する同期データにはアクセスできなくなります。
size	数値	N/A	ストレージプールの合計サイズを識別する読み取り専用プロパティー。
version	数値	N/A	このプールの現在のディスク上バージョンを識別します。プールの更新には zpool upgrade コマンドによる方法を推奨しますが、このプロパティーを使用できるのは、下位互換性のために特定のバージョンが必要な場合です。このプロパティーは、1 から zpool upgrade -v コマンドによって報告される現在のバージョンまでの任意の数値に設定できます。

ZFS ストレージプールのステータス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ステータス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、および健全性ステータスです。このセクションでは、3 つのすべてのタイプのストレージプール情報について説明します。

ZFS ストレージプールについての情報を表示する

`zpool list` コマンドは、プールに関する基本情報を表示します。このコマンドは、次の方法で使用できます。

- オプションなし: `zpool list [`*pool*`]`
  プールを指定しない場合は、すべてのプールの情報が表示されます。
- オプション付き: `zpool list` *options* `[`*arguments*`]`

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表示する

`zpool list [`*pool*`]` コマンドは、次のプール情報を表示します。

`NAME`	プールの名前。
`SIZE`	このプールの合計サイズ。最上位レベルのすべての仮想デバイスの合計サイズに等しくなります。
`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ている物理容量。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。
`FREE`	このプール内の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。合計ディスク容量の割合 (%) として表されます。
`HEALTH`	このプールの現在の健全性ステータス。 プールの健全性の詳細については、66 ページの「ZFS ストレージプールの健全性ステータスを調べる」を参照してください。

ALTROOT　このプールの代替ルート (1 つ存在する場合)。

代替ルートプールの詳細については、236 ページの「代替ルート場所で ZFS プールを使用する」を参照してください。

次の例は、サンプルの zpool list コマンド出力を示しています。

```
# zpool list
NAME                    SIZE    ALLOC    FREE    CAP  HEALTH     ALTROOT
syspool1                80.0G   22.3G   47.7G    28%  ONLINE     -
syspool2                 1.2T    384G    816G    32%  ONLINE     -
```

特定のプールの統計を取得するには、このコマンドでプール名を指定します。

特定のストレージプールの統計を表示する

zpool list コマンドでオプションと引数を発行することによって、表示される特定のプール情報を選択できます。

-o オプションを使用すると、どの列が表示されるかをフィルタ処理できます。次の例は、各プールの名前とサイズのみを一覧表示する方法を示しています。

```
# zpool list -o name,size
NAME                     SIZE
syspool1                80.0G
syspool2                 1.2T
```

組み合わされた -Ho オプションを発行することによって、シェルスクリプトの一部として zpool list コマンドを使用できます。-H オプションは列見出しの表示を抑制し、代わりにタブで区切られたプール情報を表示します。例:

```
# zpool list -Ho name,size
syspool1   80.0G
syspool2  1.2T
```

-T オプションを使用すると、プールに関するタイムスタンプ付きの統計を収集できます。構文は次のとおりです。

```
# zpool list -T d interval [count]
```

d　日付を表示するときに標準の日付の形式を使用するように指定します。

interval　情報が表示される間隔 (秒単位) を指定します。

count　情報を報告する回数を指定します。*count* を指定しない場合は、Ctrl - C キーを押すまで、指定された間隔で情報が継続的にリフレッシュされます。

次の例では、報告の間隔を 3 秒にして、プール情報を 2 回表示します。この出力では、標準の形式を使用して日付を表示しています。

```
# zpool list -T d 3 2
Tue Nov  2 10:36:11 MDT 2010
NAME    SIZE  ALLOC   FREE    CAP  DEDUP  HEALTH  ALTROOT
pool   33.8G  83.5K  33.7G     0%  1.00x  ONLINE  -
rpool  33.8G  12.2G  21.5G    36%  1.00x  ONLINE  -
Tue Nov  2 10:36:14 MDT 2010
pool   33.8G  83.5K  33.7G     0%  1.00x  ONLINE  -
rpool  33.8G  12.2G  21.5G    36%  1.00x  ONLINE  -
```

物理的な場所によりプールデバイスを表示する

プールデバイスの物理的な場所に関する情報を表示するには、`zpool status -l` オプションを使用します。物理的な場所の情報を確認しておけば、ディスクを物理的に除去または交換する必要があるときに役立ちます。

さらに、`fmadm add-alias` コマンドを使って、環境内でディスクの物理的な位置を特定するのに役立つディスクの別名を含めることもできます。例:

```
# fmadm add-alias SUN-Storage-J4400.1002QCQ015 Lab10Rack5disk

# zpool status -l system1
  pool: system1
 state: ONLINE
  scan: scrub repaired 0 in 0h0m with 0 errors on Fri Aug  3 16:00:35 2012
config:

        NAME                                        STATE     READ WRITE CKSUM
        system1                                     ONLINE       0     0     0
          mirror-0                                  ONLINE       0     0     0
            /dev/chassis/Lab10Rack5.../DISK_02/disk  ONLINE       0     0     0
            /dev/chassis/Lab10Rack5.../DISK_20/disk  ONLINE       0     0     0
          mirror-1                                  ONLINE       0     0     0
            /dev/chassis/Lab10Rack5.../DISK_22/disk  ONLINE       0     0     0
            /dev/chassis/Lab10Rack5.../DISK_14/disk  ONLINE       0     0     0
          mirror-2                                  ONLINE       0     0     0
            /dev/chassis/Lab10Rack5.../DISK_10/disk  ONLINE       0     0     0
            /dev/chassis/Lab10Rack5.../DISK_16/disk  ONLINE       0     0     0
.
.
.
        spares
          /dev/chassis/Lab10Rack5.../DISK_17/disk    AVAIL
          /dev/chassis/Lab10Rack5.../DISK_12/disk    AVAIL

errors: No known data errors
```

ZFS ストレージプールのコマンド履歴を表示する

`zfs` および `zpool` コマンドの使用のログを表示するには、`zpool history` コマンドを使用します。このログは、プール状態情報を変更したり、エラー状態をトラブル

シューティングしたりするためにこれらのコマンドが正常に使用されたときに記録されます。

履歴ログに関する次の情報に注意してください。

- このログを無効にすることはできません。このログはディスク上に永続的に保存され、システムのリブート後も保持されます。
- 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。
- このログには管理が必要ないため、ログのサイズや場所をチューニングする必要はありません。

次の例は、プール `system1` に対する `zfs` および `zpool` コマンドの履歴を示しています。

```
# zpool history system1
2012-01-25.16:35:32 zpool create -f system1 mirror c3t1d0 c3t2d0 spare c3t3d0
2012-02-17.13:04:10 zfs create system1/test
2012-02-17.13:05:01 zfs snapshot -r system1/test@snap1
```

`-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。例:

```
# zpool history -l system1
History for 'system1':
2012-01-25.16:35:32 zpool create -f system1 mirror c3t1d0 c3t2d0 spare c3t3d0
[user root on host1:global]
2012-02-17.13:04:10 zfs create system1/test [user root on host1:global]
2012-02-17.13:05:01 zfs snapshot -r system1/test@snap1 [user root on host1:global]
```

`-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。例:

```
# zpool history -i system1
History for 'system1':
2012-01-25.16:35:32 zpool create -f system1 mirror c3t1d0 c3t2d0 spare c3t3d0
2012-01-25.16:35:32 [internal pool create txg:5] pool spa 33; zfs spa 33; zpl 5;
uts host1 5.11 11.1 sun4v
2012-02-17.13:04:10 zfs create system1/test
2012-02-17.13:04:10 [internal property set txg:66094] $share2=2 dataset = 34
2012-02-17.13:04:31 [internal snapshot txg:66095] dataset = 56
2012-02-17.13:05:01 zfs snapshot -r system1/test@snap1
2012-02-17.13:08:00 [internal user hold txg:66102] <.send-4736-1> temp = 1 ...
```

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。次の統計情報が報告されます。

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。
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

オプションなしで発行された場合、zpool iostat コマンドは、システム上のすべてのプールについてブート以降に累積された統計を表示します。例:

```
# zpool iostat
capacity     operations    bandwidth
pool        alloc   free   read  write   read  write
----------  -----  -----  -----  -----  -----  -----
rpool       6.05G  61.9G      0      0    786    107
system1     31.3G  36.7G      4      1   296K  86.1K
----------  -----  -----  -----  -----  -----  -----
```

これらの統計はブート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。例:

```
# zpool iostat system1 2
capacity     operations    bandwidth
pool           alloc   free   read  write   read  write
----------     -----   -----  ----  -----  -----  -----
system1        18.5G  49.5G      0    187      0  23.3M
system1        18.5G  49.5G      0    464      0  57.7M
system1        18.5G  49.5G      0    457      0  56.6M
system1        18.8G  49.2G      0    435      0  51.3M
```

この例では、Ctrl - C キーを押すまで、このコマンドによってプール system1 の使用統計が 2 秒ごとに表示されます。または、count 引数を追加で指定することもでき、その場合はコマンドが指定した数だけ繰り返されたあとで終了します。

たとえば、zpool iostat 2 3 の場合は、サマリーが 2 秒ごとに 3 回 (計 6 秒間) 出力されます。

仮想デバイスの入出力統計を一覧表示する

zpool iostat -v コマンドは、仮想デバイスの入出力統計を表示できます。異常に低速なデバイスを識別したり、ZFS によって生成された入出力の分布を監視したりするには、このコマンドを使用します。例:

```
# 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
----------  -----  -----  -----  -----  -----  -----
system1     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。

- ディスク領域使用に関する統計は、最上位レベルの仮想デバイスに対してのみ使用できます。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。
- 数値が、期待したほど正確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。この差は、特にプールが作成された直後に顕著です。それは、プール作成の一部として大量の入出力が直接ディスクに対して実行されますが、これがミラーレベルでは数に含まれないためです。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は、interval と count を使用できます。

プールの仮想デバイスに関する物理的な場所の情報も表示できます。次の例は、切り捨てられたサンプル出力を示しています。

```
# zpool iostat -lv
               capacity     operations    bandwidth
pool        alloc   free   read  write   read  write
----------  -----  -----  -----  -----  -----  -----
export      2.39T  2.14T     13     27  42.7K   300K
  mirror     490G   438G      2      5  8.53K  60.3K
    /dev/chassis/lab10rack15/SCSI_Device__2/disk      -   -    1    0  4.47K  60.3K
    /dev/chassis/lab10rack15/SCSI_Device__3/disk      -   -    1    0  4.45K  60.3K
  mirror     490G   438G      2      5  8.62K  59.9K
    /dev/chassis/lab10rack15/SCSI_Device__4/disk      -   -    1    0  4.52K  59.9K
    /dev/chassis/lab10rack15/SCSI_Device__5/disk      -   -    1    0  4.48K  59.9K
```

ZFS ストレージプールの健全性ステータスを調べる

`zpool status` コマンドを使用すると、プールとデバイスの健全性を表示できます。さらに、`fmd` コマンドはまた、プールとデバイスの潜在的な障害もシステムコンソールと `/var/adm/messages` ファイルに報告します。

このセクションでは、プールとデバイスの健全性を確認する方法についてのみ説明します。正常でないプールからのデータ回復については、第12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。

プールの健全性ステータスは、次の 4 つの状態のいずれかで表されます。

`DEGRADED`
: 1 つ以上のデバイスで障害が発生しているが、冗長構成のためにデータは引き続き使用可能な状態にあるプール。

`ONLINE`
: すべてのデバイスが正常に動作してるプール。

`SUSPENDED`
: デバイスの接続が復元されるのを待機しているプール。デバイスの問題が解決されるまで、`SUSPENDED` プールの状態は wait のままです。

`UNAVAIL`
: メタデータが壊れているか、1 つまたは複数のデバイスが使用できず、動作を継続するための複製が不足しているプール。

各プールデバイスは、次のいずれかの状態になることができます。

`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。別のデバイスでの以降の障害が回復不可能になることがあるため、プールの耐障害性が危険にさらされる可能性があります。
`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。
`ONLINE`	一時的なエラーがいくつか引き続き発生する可能性はありますが、デバイスまたは仮想デバイスは正常な動作状態にあります。
`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。

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。

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 UNAVAIL または OFFLINE の場合は、プールも UNAVAIL または SUSPENDED になります。UNAVAIL または SUSPENDED 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であった場合と同じレベルのデータ冗長性またはデータスループットを実現できない可能性があります。

zpool status コマンドはまた、次のように、再同期化およびスクラブ操作の状態も表示します。

- 再同期化またはスクラブ操作が進行中です。
- 再同期化またはスクラブ操作が完了しました。

  再同期化およびスクラブの完了メッセージはシステムのリブート後も保持されます。
- 操作が取り消されました。

ストレージプールの健全性ステータス

次のいずれかの zpool status コマンドオプションを使用すると、プールの健全性ステータスを確認できます。

- zpool status -x [*pool*] – エラーがあるか、またはそれ以外で使用できないプールのステータスのみを表示します。
- zpool status -v [*pool*] – プールとそのデバイスに関する詳細情報を提供する詳細出力を生成します。

  ONLINE 状態にないプールがある場合は、潜在的な問題がないかどうかを調査するようにしてください。

次の例は、プール system1 に関する詳細なステータスレポートを生成する方法を示しています。

```
# zpool status -v system1
  pool: system1
 state: DEGRADED
```

```
status: One or more devices are unavailable in response to persistent errors.
        Sufficient replicas exist for the pool to continue functioning in a
        degraded state.
action: Determine if the device needs to be replaced, and clear the errors
        using 'zpool clear' or 'fmadm repaired', or replace the device
        with 'zpool replace'.
  scan: scrub repaired 0 in 0h0m with 0 errors on Wed Jun 20 15:38:08 2012
config:

        NAME                      STATE     READ WRITE CKSUM
        system1                   DEGRADED     0     0     0
          mirror-0                DEGRADED     0     0     0
            c0t5000C500335F95E3d0 ONLINE       0     0     0
            c0t5000C500335F907Fd0 UNAVAIL      0     0     0
          mirror-1                ONLINE       0     0     0
            c0t5000C500335BD117d0 ONLINE       0     0     0
            c0t5000C500335DC60Fd0 ONLINE       0     0     0

device details:

        c0t5000C500335F907Fd0    UNAVAIL          cannot open
        status: ZFS detected errors on this device.
                The device was missing.
           see: URL to My Oracle Support knowledge article  for recovery


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、何らかの修正アクションが必要な潜在的なデバイス障害を示します。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。

この出力は、問題だけでなく、このプールの現在の状態の考えられる原因を識別します。この出力にはまた、こ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含むナレッジ記事へのリンクも含まれています。この出力から、どのデバイスが破損しているか、およびプールを修復する方法を特定できます。

UNAVAIL のプールとデータを診断および修復する方法の詳細は、第12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 ストレージプールのステータス情報を収集する

zpool status の間隔およびカウントオプションを使用して、ある期間にわたっての統計を収集できます。また、-T オプションを使用することによってタイムスタンプを表示できます。例:

```
# zpool status -T d 3 2
Wed Jun 20 16:10:09 MDT 2012
  pool: pond
 state: ONLINE
  scan: resilvered 9.50K in 0h0m with 0 errors on Wed Jun 20 16:07:34 2012
config:
```

```
        NAME                      STATE     READ WRITE CKSUM
        pond                      ONLINE       0     0     0
          mirror-0                ONLINE       0     0     0
            c0t5000C500335F95E3d0 ONLINE       0     0     0
            c0t5000C500335F907Fd0 ONLINE       0     0     0
          mirror-1                ONLINE       0     0     0
            c0t5000C500335BD117d0 ONLINE       0     0     0
            c0t5000C500335DC60Fd0 ONLINE       0     0     0

errors: No known data errors

  pool: rpool
 state: ONLINE
  scan: scrub repaired 0 in 0h11m with 0 errors on Wed Jun 20 15:08:23 2012
config:

        NAME                        STATE     READ WRITE CKSUM
        rpool                       ONLINE       0     0     0
          mirror-0                  ONLINE       0     0     0
            c0t5000C500335BA8C3d0s0 ONLINE       0     0     0
            c0t5000C500335FC3E7d0s0 ONLINE       0     0     0

errors: No known data errors
Wed Jun 20 16:10:12 MDT 2012

  pool: pond
 state: ONLINE
  scan: resilvered 9.50K in 0h0m with 0 errors on Wed Jun 20 16:07:34 2012
config:

        NAME                      STATE     READ WRITE CKSUM
        pond                      ONLINE       0     0     0
          mirror-0                ONLINE       0     0     0
            c0t5000C500335F95E3d0 ONLINE       0     0     0
            c0t5000C500335F907Fd0 ONLINE       0     0     0
          mirror-1                ONLINE       0     0     0
            c0t5000C500335BD117d0 ONLINE       0     0     0
            c0t5000C500335DC60Fd0 ONLINE       0     0     0

errors: No known data errors

  pool: rpool
 state: ONLINE
  scan: scrub repaired 0 in 0h11m with 0 errors on Wed Jun 20 15:08:23 2012
config:

        NAME                        STATE     READ WRITE CKSUM
        rpool                       ONLINE       0     0     0
          mirror-0                  ONLINE       0     0     0
            c0t5000C500335BA8C3d0s0 ONLINE       0     0     0
            c0t5000C500335FC3E7d0s0 ONLINE       0     0     0

errors: No known data errors
```

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。これらのストレージデバイスを元のシステムから切り離し、それらのデバイスを物理的にケー

ブル接続し直すか、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) を使用して宛先システムに再接続します。

ZFS では、各システムのアーキテクチャーエンディアンが異なる場合でも、あるシステムからプールをエクスポートし、それを宛先システムにインポートできます。異なるシステム上に存在する可能性のある異なるストレージプール間でのファイルシステムのレプリケーションまたは移行については、178 ページの「ZFS データの保存、送信、および受信」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

プールを移行するには、まずそのプールをエクスポートする必要があります。この操作により、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、エクスポートが完了したことを示すデータがディスクに書き込まれ、さらにその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。

そのプールを明示的にエクスポートせず、代わりにディスクを手動で削除した場合でも、結果として得られたプールを別のシステムに引き続きインポートできます。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。また、デバイスが存在しなくなっているため、元のシステムではこのプールが `UNAVAIL` 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、次のコマンドを使用します。

```
# zpool export [option] pool
```

このコマンドは、まずプール内のマウントされたファイルシステムをすべてアンマウントします。いずれかのファイルシステムのアンマウントに失敗した場合は、`-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。ただし、プール内の ZFS ボリュームが使用されている場合は、`-f` オプションが指定されていてもこの操作は失敗します。ZFS ボリュームを持つプールをエクスポートするには、まず、ボリュームのすべての使用者がアクティブでなくなっていることを確認します。

詳細は、227 ページの「ZFS ボリューム」を参照してください。

このコマンドが実行されると、このプールはシステム上で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があとで、ほかに動作中のデバイスがない状態でシステムに接続された場合、そのデバイスは潜在的にアクティブとして表示されます。

インポートできるストレージプールを判断する

システムからプールが削除されたら、ターゲットシステムにデバイスを接続できます。それらを同じデバイス名で接続する必要はありません。ZFS は、移動されたデバイスや名前が変更されたデバイスをすべて検出し、構成を適切に調整します。ZFS は、一部のデバイスのみが使用可能な状況にも対処できることに注意してください。ただし、プールの移行の成功は、すべてのデバイスの全体的な健全性に依存しています。

次の一般的なコマンド構文をプールのインポートのすべての操作に使用します。

```
# zpool import [options] [pool|ID-number]
```

インポートできる使用可能なプールを検出するには、プールを指定せずに `zpool import` コマンドを実行します。この出力で、各プールは名前と一意の数値識別子によって識別されます。インポートに使用できるプールが同じ名前を共有している場合は、数値識別子を使用して正しいプールをインポートします。

インポートされるプールに問題が存在する場合、このコマンド出力では、どのようなアクションを実行するのかを判断するのに役立つ適切な情報も提供されます。

次の例では、いずれかのデバイスが欠けていますが、ミラー化されたデータにアクセスできる状態のままであるため、このプールを引き続きインポートできます。

```
# zpool import
 pool: system1
    id: 4715259469716913940
 state: DEGRADED
status: One or more devices are unavailable.
action: The pool can be imported despite missing or damaged devices.  The
        fault tolerance of the pool may be compromised if imported.
config:

    system1                     DEGRADED
       mirror-0                 DEGRADED
         c0t5000C500335E106Bd0  ONLINE
         c0t5000C500335FC3E7d0  UNAVAIL  cannot open

device details:

         c0t5000C500335FC3E7d0     UNAVAIL   cannot open
         status: ZFS detected errors on this device.
                The device was missing.
```

次の例では、RAID-Z 仮想デバイスの 2 つのディスクが欠けているため、プールを再構築するための十分な冗長データが存在しません。使用可能なデバイスが不足していると、ZFS はプールをインポートできません。

```
# zpool import
pool: mothership
    id: 3702878663042245922
 state: UNAVAIL
status: One or more devices are unavailable.
action: The pool cannot be imported due to unavailable devices or data.
config:

        mothership     UNAVAIL  insufficient replicas
          raidz1-0     UNAVAIL  insufficient replicas
            c8t0d0     UNAVAIL  cannot open
            c8t1d0     UNAVAIL  cannot open
            c8t2d0     ONLINE
            c8t3d0     ONLINE

device details:

        c8t0d0    UNAVAIL          cannot open
        status: ZFS detected errors on this device.
                The device was missing.

        c8t1d0    UNAVAIL          cannot open
        status: ZFS detected errors on this device.
                The device was missing.
```

ZFS ストレージプールをインポートする

特定のプールをインポートするには、`zfs import` コマンドでプール名またはその数値識別子を指定します。さらに、プールのインポート中にその名前を変更することもできます。例:

```
# zpool import system1 mpool
```

このコマンドは、エクスポートされたプール `system1` をインポートし、その名前を `mpool` に変更します。新しいプール名は永続的な名前です。

注記 - プールの名前を直接変更することはできません。プールの名前は、そのプールのエクスポートおよびインポート中にのみ変更できます。それにより、ルートデータセットの名前も新しいプール名に変更されます。

注意 - インポート操作中に、そのプールが別のシステムで使用されている可能性がある場合は、警告が発生します。

```
cannot import 'pool': pool may be in use on another system
use '-f' to import anyway
```

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、異なる複数のシステム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の下にインポートすることもできます。詳細は、236 ページの「代替ルート場所で ZFS プールを使用する」を参照してください。

ログデバイスがないプールをインポートする

デフォルトでは、ログデバイスがないプールはインポートできません。zpool import -m コマンドを使用して、ログデバイスがないプールを強制的にインポートすることができます。

次の例では、プール dozer を最初にインポートしたときにミラー化ログがなかったことが出力に示されています。

```
# zpool import dozer
The devices below are missing, use '-m' to import the pool anyway:
mirror-1 [log]
c3t3d0
c3t4d0

cannot import 'dozer': one or more devices is currently unavailable
```

ミラー化ログのないプールのインポートを続行するには、-m オプションを使用します。

```
# zpool import -m dozer
# zpool status dozer
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: URL to My Oracle Support knowledge article
scan: scrub repaired 0 in 0h0m with 0 errors on Fri Oct 15 16:51:39 2010
config:

NAME                    STATE     READ  WRITE  CKSUM
dozer                   DEGRADED     0      0      0
   mirror-0             ONLINE       0      0      0
      c3t1d0            ONLINE       0      0      0
      c3t2d0            ONLINE       0      0      0
   logs
   mirror-1                 UNAVAIL      0     0     0  insufficient replicas
      13514061426445294202  UNAVAIL      0     0     0  was c3t3d0
      16839344638582008929  UNAVAIL      0     0     0  was c3t4d0
```

インポートされたプールは、DEGRADED 状態のままになります。出力の推奨事項に基づいて、欠落しているログデバイスを接続します。次に、zpool clear コマンドを実行してプールエラーをクリアします。

読み取り専用モードでプールをインポートする

あるプールがアクセス不可能なほど損傷している場合は、読み取り専用モードでインポートすると、そのプールのデータを回復できることがあります。例:

```
# zpool import -o readonly=on system1
# zpool scrub system1
cannot scrub system1: pool is read-only
```

プールが読み取り専用モードでインポートされるとき、次の条件が適用されます。

- すべてのファイルシステムおよびボリュームが読み取り専用モードでマウントされます。
- プールトランザクション処理が無効になります。インテントログ内の保留中の同期書き込みはすべて、そのプールが読み取り/書き込みモードでインポートされるまで実行されません。
- 読み取り専用のインポート中にプールプロパティーを設定しても無視されます。

プールは、エクスポートおよびインポートすることによって読み取り/書き込みモードに戻すことができます。例:

```
# zpool export system1
# zpool import system1
# zpool scrub system1
```

特定のデバイスパスを使用してプールをインポートする

デフォルトでは、`zpool import` コマンドは、`/dev/dsk` ディレクトリ内でのみデバイスを検索します。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。例:

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

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

デバイスが複数のディレクトリに存在する場合は、複数の `-d` オプションを指定できます。

次のコマンドは、プールの特定のデバイスの 1 つである `/dev/etc/c2t3d0` を識別することによって、プール `mpool` をインポートします。

```
# zpool import -d /dev/etc/c2t3d0 mpool
# zpool status mpool
pool: mpool
state: ONLINE
```

```
scan: resilvered 952K in 0h0m with 0 errors on Fri Jun 29 16:22:06 2012
config:

NAME        STATE     READ WRITE CKSUM
mpool       ONLINE       0     0     0
mirror-0  ONLINE       0     0     0
c2t3d0  ONLINE       0     0     0
c2t1d0  ONLINE       0     0     0
```

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。

次の例では、プール `system1` は破棄されたとして示されています。

```
# zpool import -D
  pool: system1
    id: 5154272182900538157
 state: ONLINE (DESTROYED)
action: The pool can be imported using its name or numeric identifier.
config:

        system1     ONLINE
          mirror-0  ONLINE
            c1t0d0  ONLINE
            c1t1d0  ONLINE
```

破棄されたプールを回復するには、`-D` オプションを使用してそのプールをインポートします。

```
# zpool import -D system1
# zpool status system1
  pool: system1
 state: ONLINE
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        system1     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
```

破棄されたプール内のいずれかのデバイスが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引き続き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。例:

```
# zpool import -D
  pool: dozer
    id: 4107023015970708695
 state: DEGRADED (DESTROYED)
status: One or more devices are unavailable.
```

```
action: The pool can be imported despite missing or damaged devices.  The
        fault tolerance of the pool may be compromised if imported.
config:

        dozer               DEGRADED
          raidz2-0          DEGRADED
            c8t0d0          ONLINE
            c8t1d0          ONLINE
            c8t2d0          ONLINE
            c8t3d0          UNAVAIL  cannot open
            c8t4d0          ONLINE

device details:

        c8t3d0    UNAVAIL          cannot open
        status: ZFS detected errors on this device.
                The device was missing.
# zpool import -Df dozer
# zpool status -x
  pool: dozer
 state: DEGRADED
status: One or more devices are unavailable in response to persistent errors.
        Sufficient replicas exist for the pool to continue functioning in a
        degraded state.
action: Determine if the device needs to be replaced, and clear the errors
        using 'zpool clear' or 'fmadm repaired', or replace the device
        with 'zpool replace'.
        Run 'zpool status -v' to see device specific details.
  scan: none requested
config:

        NAME                     STATE     READ WRITE CKSUM
        dozer                    DEGRADED     0     0     0
          raidz2-0               DEGRADED     0     0     0
            c8t0d0               ONLINE       0     0     0
            c8t1d0               ONLINE       0     0     0
            c8t2d0               ONLINE       0     0     0
            4881130428504041127  UNAVAIL      0     0     0
            c8t4d0               ONLINE       0     0     0

errors: No known data errors
# zpool online dozer c8t3d0
# 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

`zpool upgrade` コマンドを使用すると、ZFS ストレージプールを以前の Oracle Solaris リリースからアップグレードできます。

このコマンドを使用する前に、`zpool status` コマンドを使用して、これらのプールが現在システム上にあるバージョンより前の ZFS バージョンで構成されたかどうかをチェックします。また、次に示すように、`-v` オプションを使用して、システム上にある現在の ZFS バージョンの機能を表示することも考慮してください。

```
# zpool upgrade -v
```

機能のリストは、システム上の ZFS バージョン番号によって異なります。完全なリストについては、291 ページの「ZFS プールのバージョン」を参照してください。

プールをアップグレードし、最新の ZFS 機能を利用するには、-a オプションを使用します。

```
# zpool upgrade -a
```

プールをアップグレードすると、以前の ZFS バージョンを実行しているシステム上ではそれらのプールにアクセスできなくなります。

例 15 ZFS プールをアップグレードする

この例は、プールをアップグレードするためのアクションを示しています。

```
# zpool status
  pool: system1
 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
        system1     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

# zpool upgrade -v
This system is currently running ZFS pool version version-number.

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
.
.
    追加機能

# zpool upgrade -a
```

♦ ♦ ♦ 第 6 章

6 ZFS ルートプールの管理

この章では、Oracle Solaris ZFS ルートプールとそのコンポーネントを管理する方法について説明します。内容は次のとおりです。

- 79 ページの「ZFS ルートプールを構成するための要件」
- 82 ページの「ZFS ルートプールの管理」
- 90 ページの「ZFS スワップおよびダンプデバイスの管理」
- 95 ページの「ZFS ルートファイルシステムからのブート」

ルートプールの回復については、『Oracle Solaris 11.3 でのシステム復旧とクローン』を参照してください。

ZFS ルートプールを構成するための要件

ZFS は、Oracle Solaris のデフォルトのルートファイルシステムです。ルートプールはブート環境 (BE) を含んでおり、インストール中に自動的に作成されます。

ZFS ルートプールの容量要件

スワップダンプボリュームのサイズは、物理メモリーの量によって異なります。ブート可能な ZFS ルートファイルシステムのためのプール領域の最小量は、物理メモリーの量、使用可能なディスク容量、および作成される BE の数によって異なります。

23 ページの「ハードウェアおよびソフトウェアの要件」で推奨されている 7G バイトから 13G バイトの最小ディスク容量は、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 – インストールプログラムが作成するスワップおよびダンプボリュームのデフォルトサイズは、システムメモリーの量などの変数によって異なります。ダンプデバイスのサイズは、システムの動作状態に応じて、物理メモリーのサイズの約半分かそれ以上になります。

  スワップおよびダンプボリュームのサイズは、インストール中またはインストール後に調整できます。新しいサイズがシステム動作をサポートしている必要がありま

す。93 ページの「ZFS スワップおよびダンプデバイスのサイズ調整」を参照してください。

- ブート環境 – ZFS BE は、約 4G バイトから 6G バイトです。別の ZFS BE から複製される各 ZFS BE に追加ディスク容量は必要ありません。BE のサイズは、BE が更新されたときに、その更新に応じて増加します。同じルートプール内のすべての ZFS BE は、同じスワップおよびダンプデバイスを使用します。
- Oracle Solaris コンポーネント – OS イメージの一部である、`/var` を除くルートファイルシステムのすべてのサブディレクトリがルートファイルシステム内に存在する必要があります。スワップおよびダンプデバイスを除くすべての Oracle Solaris コンポーネントがルートプール内に存在する必要があります。

ZFS ルートプール構成の推奨事項

ZFS ルートプールを構成する場合は、次のガイドラインに従ってください。

- EFI (GPT) ラベル付きディスクを使用している場合は、ミラー化されたディスク全体にルートプールを作成します。SMI (VTOC) ラベル付きディスクを使用している場合は、ミラー化されたスライスにルートプールを作成します。

  ほとんどの場合、GPT 対応ファームウェアが搭載された x86 システムおよび SPARC システムでは EFI (GPT) ラベル付きディスクが使用されます。それ以外の場合、SPARC システムでは SMI (VTOC) ラベル付きディスクが使用されます。
- システムがブートできなくなる可能性があるため、初期インストールのあとに作成されたルートプールの名前を変更しないでください。
- シンプロビジョニングされた VMware デバイスをルートプールデバイスに使用しないでください。

ルートプールには、次の制限があります。

- ルートプールでは、RAID-Z またはストライプ化された構成はサポートされません。
- 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。
- ルートプール上に最上位レベルの仮想デバイスを複数構成することはできません。ただし、追加のデバイスを接続することによって、ミラー化ルートプールを拡張することは可能です。
- `gzip` および `lz4` 圧縮アルゴリズムはルートプールでサポートされていません。

ZFS ルートプールのインストール

『Oracle Solaris 11.3 システムのインストール』に記載されているように、Live Media、テキストインストーラ、または AI マニフェスト付きの Automated Installer (AI)

を使用して Oracle Solaris をインストールできます。3 つの方法はすべて、ZFS ルートプールを単一ディスクに自動的にインストールします。このインストールではまた、スワップおよびダンプデバイスもルートプールの ZFS ボリューム上に構成されます。

AI の方法では、ルートプールのインストールでのより高い柔軟性が提供されます。AI マニフェストでは、例16「ルートプールのインストールをカスタマイズするための AI マニフェストの変更」に示すように、ミラー化ルートプールを作成するために使用するディスクを指定したり、ZFS プロパティーを有効にしたりできます。

Oracle Solaris が完全にインストールされたら、次のアクションを実行します。

- インストールによってルートプールが単一ディスク上に作成された場合は、そのプールをミラー化構成に手動で変換します。83 ページの「ミラー化ルートプールを構成する方法 (SPARC または x86/VTOC)」を参照してください。
- ルートファイルシステムがいっぱいにならないようにするために、ZFS ルートファイルシステムに割り当て制限を設定します。現時点では、ファイルシステム全体のセーフティーネットとして予約されている ZFS ルートプール容量はありません。たとえば、ルートプールのディスクが 68G バイトの場合、ZFS ルートファイルシステム (`rpool/ROOT/solaris`) に 67G バイトの割り当て制限を設定して、ファイルシステム領域を 1G バイト残すことを検討してください。148 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。
- Oracle Solaris アーカイブユーティリティーを使用して、障害回復または移行の目的のためのルートプールの回復用のアーカイブを作成します。詳細は、『Oracle Solaris 11.3 でのシステム復旧とクローン』および `archiveadm(1M)` のマニュアルページを参照してください。

例 16 ルートプールのインストールをカスタマイズするための AI マニフェストの変更

この例は、次を実行するために AI マニフェストをカスタマイズする方法を示しています。

- `c1t0d0` と `c2t0d0` から成るミラー化ルートプールを作成します。
- ルートプールの `listsnaps` プロパティーを有効にします。

```
<target>
<disk whole_disk="true" in_zpool="rpool" in_vdev="mirrored">
<disk_name name="c1t0d0" name_type="ctd"/>
</disk>
<disk whole_disk="true" in_zpool="rpool" in_vdev="mirrored">
<disk_name name="c2t0d0" name_type="ctd"/>
</disk>
<logical>
<zpool name="rpool" is_root="true">
<vdev name="mirrored" redundancy="mirror"/>
<!--
...
-->
<filesystem name="export" mountpoint="/export"/>
<filesystem name="export/home"/>
<pool_options>
```

```
<option name="listsnaps" value="on"/>
</pool_options>
<be name="solaris"/>
</zpool>
</logical>
</target>
```

例 17 サンプルのルートプール構成

次の例は、カスタマイズされたマニフェストによる AI インストールのあとのミラー化ルートプールとファイルシステム構成を示しています。

```
# zpool status rpool
pool: rpool
state: ONLINE
scan: none requested
config:

NAME         STATE     READ WRITE CKSUM
rpool        ONLINE       0     0     0
   mirror-0  ONLINE       0     0     0
     c8t0d0  ONLINE       0     0     0
     c8t1d0  ONLINE       0     0     0

# zfs list
NAME                      USED  AVAIL  REFER  MOUNTPOINT
rpool                    11.8G  55.1G  4.58M  /rpool
rpool/ROOT               3.57G  55.1G    31K  legacy
rpool/ROOT/solaris       3.57G  55.1G  3.40G  /
rpool/ROOT/solaris/var    165M  55.1G   163M  /var
rpool/VARSHARE           42.5K  55.1G  42.5K  /var/share
rpool/dump               6.19G  55.3G  6.00G  -
rpool/export               63K  55.1G    32K  /export
rpool/export/home          31K  55.1G    31K  /export/home
rpool/swap               2.06G  55.2G  2.00G  -
```

ZFS ルートプールの管理

このセクションでは、ZFS ルートプールを管理するための手順について説明します。

▼ ミラー化ルートプール (SPARC または x86/EFI (GPT)) を構成する方法

この手順では、デフォルトのルートプールのインストールを冗長構成に変換する方法について説明します。この手順は、ディスクに EFI (GPT) ラベルが付いている、GPT 対応ファームウェアが搭載されたほとんどの x86 システムおよび SPARC システムに適用されます。

1. **(オプション) ルートプールの現在のステータスを表示します。**

```
# zpool status root-pool
```

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 root-pool current-disk new-disk
```

適切なディスクのラベル付けとブートブロックが自動的に適用されます。

3. **ルートプールのステータスを表示し、再同期化が完了しているか確認します。**

再同期化が完了している場合は、出力に次のようなメッセージが含まれています。

```
scan: resilvered 11.6G in 0h5m with 0 errors on Fri Jul 20 13:57:25 2014
```

4. **新しいディスクが現在のディスクより大きい場合は、ZFS の `autoexpand` プロパティーを有効にします。**

```
# zpool set autoexpand=on root-pool
```

次の例は、`autoexpand` プロパティーが有効になったあとの `rpool` のディスク領域の違いを示しています。

```
# zpool list rpool
NAME   SIZE  ALLOC   FREE  CAP  DEDUP  HEALTH  ALTROOT
rpool  29.8G   152K  29.7G   0%  1.00x  ONLINE  -

# zpool set autoexpand=on rpool

# zpool list rpool
NAME   SIZE  ALLOC  FREE  CAP  DEDUP  HEALTH  ALTROOT
rpool  279G   146K  279G   0%  1.00x  ONLINE  -
```

5. **新しいディスクから正常にブートできることを確認します。**

注記 - ZFS 構成が、ミラー化された iSCSI ターゲットに基づいて構築されたルートファイルシステムで構成されているとき、ブートディスクと同じ iSCSI ターゲットまたはセッション上で 2 番目の LUN を使用できない場合は、予期しない動作が発生することがあります。システムがブートされると、ブートプロセスによって、2 番目の iSCSI LUN を開くことに失敗し、ルートプールが縮退状態にあることが報告されます。ただし、このステータスは一時的です。この問題は、ZFS が迅速な再同期化を実行すると自動的に解決します。そのあと、2 番目の LUN がオンラインになり、ルートプールの状態もオンラインになります。

▼ ミラー化ルートプールを構成する方法 (SPARC または x86/VTOC)

この手順では、デフォルトのルートプールのインストールを冗長構成に変換する方法について説明します。この手順は、ディスクに SMI (VTOC) ラベルが付いている、

GPT 対応ファームウェアが搭載されていない特定の x86 システムおよび SPARC システムに適用されます。

始める前に ルートプールに接続する 2 番目のディスクを次のように準備します。

- SPARC: そのディスクに SMI (VTOC) ディスクラベルが付いており、スライス 0 にディスク容量の大部分が含まれていることを確認します。ディスクのラベルを変更し、スライス 0 を作成する必要がある場合は、『Oracle Solaris 11.3 でのデバイスの管理』 の 「ZFS ルートプール (VTOC) の交換方法」を参照してください。
- x86: ディスクに `fdisk` パーティション、SMI ディスクラベル、およびスライス 0 があることを確認してください。ディスクのパーティションを変更し、スライス 0 を作成する必要がある場合は、『Oracle Solaris 11.3 でのデバイスの管理』 の 「スライスまたはパーティションの変更」を参照してください。

1. **(オプション) ルートプールの現在のステータスを表示します。**

```
# zpool status root-pool
```

この構成では、ディスクのスライス 0 が次の `rpool` の例に示すように表示されます。

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

NAME        STATE     READ WRITE CKSUM
rpool       ONLINE       0     0     0
c2t0d0s0    ONLINE       0     0     0

errors: No known data errors
```

2. **ミラー化ルートプールを構成するために 2 番目のディスクを接続します。**

```
# zpool attach root-pool current-disk new-disk
```

ディスクを指定するときは、必ずスライスを含めてください (`c2t0d0s0` など)。適切なディスクのラベル付けとブートブロックが自動的に適用されます。

3. **ルートプールのステータスを表示し、再同期化が完了しているか確認します。**

再同期化が完了している場合は、出力に次のようなメッセージが含まれています。

```
scan: resilvered 11.6G in 0h5m with 0 errors on Fri Jul 20 13:57:25 2014
```

4. **新しいディスクが現在のディスクより大きい場合は、ZFS の `autoexpand` プロパティーを有効にします。**

```
# zpool set autoexpand=on root-pool
```

次の例は、`autoexpand` プロパティーが有効になったあとの `rpool` のディスク領域の違いを示しています。

```
# zpool list rpool
NAME   SIZE  ALLOC   FREE  CAP  DEDUP  HEALTH  ALTROOT
```

```
rpool  29.8G   152K  29.7G   0%  1.00x  ONLINE  -

# zpool set autoexpand=on rpool

# zpool list rpool
NAME   SIZE  ALLOC  FREE  CAP  DEDUP  HEALTH  ALTROOT
rpool  279G   146K  279G   0%  1.00x  ONLINE  -
```

5. **新しいディスクから正常にブートできることを確認します。**

注記 - ZFS 構成が、ミラー化された iSCSI ターゲットに基づいて構築されたルートファイルシステムで構成されているとき、ブートディスクと同じ iSCSI ターゲットまたはセッション上で 2 番目の LUN を使用できない場合は、予期しない動作が発生することがあります。システムがブートされると、ブートプロセスによって、2 番目の iSCSI LUN を開くことに失敗し、ルートプールが縮退状態にあることが報告されます。ただし、このステータスは一時的です。この問題は、ZFS が迅速な再同期化を実行すると自動的に解決します。そのあと、2 番目の LUN がオンラインになり、ルートプールの状態もオンラインになります。

▼ ZFS ブート環境を更新する方法

デフォルトでは、ZFS BE には `solaris` という名前が付けられます。`pkg update` コマンドは、現在の BE と更新された BE の間に大きな違いが存在する場合、新しい BE を作成し、自動的にアクティブ化することによって ZFS BE を更新します。

1. **(オプション) 現在のブート環境の構成を表示します。**

   BE の `Active` フィールドには、その BE がアクティブであることを示す `N`、システムのリブート後にアクティブになることを示す `R`、またはその両方 (`NR`) が表示されます。

```
# beadm list
BE       Active  Mountpoint  Space  Policy  Created
--       ------ ---------- ----- ------ -------
solaris  NR      /          3.82G  static  2012-07-19 13:44
```

2. **ZFS BE を更新します。**

```
# pkg update
.
DOWNLOAD                                  PKGS       FILES    XFER (MB)
Completed                                707/707 10529/10529  194.9/194.9
.
```

   `solaris-1` という新しい BE が自動的に作成されてアクティブになります。

3. **システムをリブートして BE のアクティブ化を完了します。その後、BE のステータスを確認します。**

```
# init 6
.
.
# beadm list
BE         Active  Mountpoint  Space   Policy  Created
--         ------ ---------- -----  ------ -------
solaris    -      -           46.95M   static  2014-07-20 10:25
solaris-1  NR      /           3.82G   static  2014-07-19 14:45
```

4. **新しい BE のブート時にエラーが発生した場合は、以前の BE を有効にしてブートします。**

```
# beadm activate solaris
# init 6
```

どの更新操作にも依存することなく既存のバックアップ BE をアクティブ化するには、同じ `beadm activate` *BE* コマンド構文を使用します。

▼ 代替 BE をマウントする方法

1. **管理者になります。**
2. **代替 BE をマウントします。**

```
# beadm mount alt-BE /mnt
```

3. **BE にアクセスします。**

```
# ls /mnt
```

4. **完了したら、代替 BE をアンマウントします。**

```
# beadm umount alt-BE
```

ZFS ルートプール内のディスクの交換

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、より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。非冗長プールでは、ディスクに障害が発生してシステムがブートしなくなった場合は、CD やネットワークなどの別のソースからブートします。次に、ルートプールディスクを交換します。

`zpool replace` を使用してディスクを交換したあと、ブートブロックを手動で適用する必要があります。

ミラー化ルートプール構成では、別のソースからブートしなくても、ディスクを交換できます。障害が発生したディスクを交換するには、`zpool replace` を使用します。または、追加のディスクがある場合は、`zpool attach` コマンドを使用できます。

SMI (VTOC) ラベルが付いているルートプールディスクを交換している場合は、次の要件を満たしていることを確認してください。

- SPARC: そのディスクに SMI (VTOC) ディスクラベルが付いており、スライス 0 にディスク容量の大部分が含まれていることを確認します。ディスクのラベルを変更し、スライス 0 を作成する必要がある場合は、『Oracle Solaris 11.3 でのデバイスの管理』 の 「ZFS ルートプール (VTOC) の交換方法」を参照してください。
- x86: ディスクに `fdisk` パーティション、SMI ディスクラベル、およびスライス 0 があることを確認してください。ディスクのパーティションを変更し、スライス 0 を作成する必要がある場合は、『Oracle Solaris 11.3 でのデバイスの管理』 の 「スライスまたはパーティションの変更」を参照してください。

▼ ZFS ルートプールのディスクを交換する方法

1. **交換用ディスクを物理的に接続します。**

2. **新しいディスクをルートプールに接続します。**

   ```
   # zpool attach root-pool current-disk new-disk
   ```

   ここで、*current-disk* は、この手順の最後に切り離される古いディスクになります。適切なディスクのラベル付けとブートブロックが自動的に適用されます。

   注記 - ディスクに SMI (VTOC) ラベルが付いている場合、ディスクを指定するときは、必ずスライスを含めてください (`c2t0d0s0` など)。

3. **ルートプールのステータスを表示し、再同期化が完了しているか確認します。**
   再同期化が完了している場合は、出力に次のようなメッセージが含まれています。

   ```
   scan: resilvered 11.6G in 0h5m with 0 errors on Fri Jul 20 13:57:25 2014
   ```

4. **新しいディスクから正常にブートできることを確認します。**

5. **正常にブートされたあと、古いディスクを切り離します。**

   ```
   # zpool detach root-pool old-disk
   ```

   注記 - ディスクに SMI (VTOC) ラベルが付いている場合、ディスクを指定するときは、必ずスライスを含めてください (`c2t0d0s0` など)。

6. **接続されているディスクが既存のディスクより大きい場合は、ZFS の `autoexpand` プロパティーを有効にします。**

```
# zpool set autoexpand=on root-pool
```

7. **システムを新しいディスクから自動的にブートするように設定します。**

   - SPARC: ブート PROM から `eeprom` コマンドまたは `setenv` コマンドを使用します。
   - x86: システム BIOS を再構成します。

例 18 ZFS ルートプール内のディスクの交換 (SPARC または x86/EFI (GPT)

この例では、`rpool` という名前のルートプール内の `c2t0d0` を交換します。ここでは、交換用ディスク `c2t1d0` がシステムに物理的に接続されたことを前提にしています。

```
# zpool attach rpool c2t0d0 c2t1d0
Make sure to wait until resilver is done before rebooting.

# zpool status rpool
pool: rpool
state: ONLINE
scan: resilvered 11.7G in 0h5m with 0 errors on Fri Jul 20 13:45:37 2012
config:

NAME         STATE   READ  WRITE  CKSUM
rpool        ONLINE     0      0      0
  mirror-0   ONLINE     0      0      0
    c2t0d0   ONLINE     0      0      0
    c2t1d0   ONLINE     0      0      0

errors: No known data errors
```

新しいディスク `c2t1d0` からのブートテストを完了したら、`c2t0d0` を切り離し、必要に応じて `autoexpand` プロパティーを有効にします。

```
# zpool detach c2t0d0

# zpool list rpool
NAME   SIZE  ALLOC   FREE  CAP  DEDUP  HEALTH  ALTROOT
rpool  29.8G   152K  29.7G   0%  1.00x  ONLINE  -

# zpool set autoexpand=on rpool

# zpool list rpool
NAME   SIZE  ALLOC  FREE  CAP  DEDUP  HEALTH  ALTROOT
rpool  279G   146K  279G   0%  1.00x  ONLINE  -
```

システムを新しいディスクから自動的にブートするように設定して操作を完了します。

例 19 ルートプール内の SATA ディスクの交換 (SPARC または x86/EFI (GPT))

SATA ディスクを備えたシステムでは、障害が発生したディスクを `zpool replace` コマンドで交換する前に、その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。例:

```
# zpool offline rpool c1t0d0
# cfgadm -c unconfigure c1::dsk/c1t0d0
```

障害が発生したディスク c1t0d0 を物理的に取り外し、交換用ディスク c1t0d0 を挿入します。一部のハードウェアでは、ディスクをオンラインにしたり、挿入された交換用ディスクを再構成したりする必要はありません。

```
# cfgadm -c configure c1::dsk/c1t0d0
# zpool online rpool c1t0d0
# zpool replace rpool c1t0d0
# zpool status rpool
```

x86 システムでは、再同期化が完了したあと、ブートブロックをインストールします。

```
x86# bootadm install-bootloader
```

例 20 ZFS ルートプール内のディスクの交換 (SPARC または x86/VTOC)

この例では、rpool という名前のルートプール内の c2t0d0s0 を交換します。ここでは、交換用ディスク c2t1d0s0 がシステムに物理的に接続されたことを前提にしています。

```
# zpool attach rpool c2t0d0s0 c2t1d0s0
Make sure to wait until resilver is done before rebooting.

# zpool status rpool
pool: rpool
state: ONLINE
scan: resilvered 11.7G in 0h5m with 0 errors on Fri Jul 20 13:45:37 2012
config:

NAME         STATE   READ  WRITE  CKSUM
rpool        ONLINE     0      0      0
  mirror-0   ONLINE     0      0      0
    c2t0d0s0 ONLINE     0      0      0
    c2t1d0s0 ONLINE     0      0      0

errors: No known data errors
```

新しいディスク c2t1d0s0 からのブートをテストします。c2t1d0s0 に障害が発生した場合は、古いディスク c2t0d0s0 からのブートもテストします。

```
ok boot /pci@1f,700000/scsi@2/disk@1,0

ok boot /pci@1f,700000/scsi@2/disk@0,0
```

ブートテストを完了したら、c2t0d0s0 を切り離し、必要に応じて autoexpand プロパティーを有効にします。

```
# zpool detach c2t0d0s0

# zpool list rpool
NAME   SIZE  ALLOC   FREE  CAP  DEDUP  HEALTH  ALTROOT
rpool  29.8G   152K  29.7G   0%  1.00x  ONLINE  -

# zpool set autoexpand=on rpool

# zpool list rpool
```

```
NAME   SIZE  ALLOC  FREE  CAP  DEDUP  HEALTH  ALTROOT
rpool  279G   146K  279G   0%  1.00x  ONLINE  -
```

システムを新しいディスクから自動的にブートするように設定して操作を完了します。

例 21 ルートプール内の SATA ディスクの交換 (SPARC または x86 (VTOC))

SATA ディスクを備えたシステムでは、障害が発生したディスクを `zpool replace` コマンドで交換する前に、その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。例:

```
# zpool offline rpool c1t0d0s0
# cfgadm -c unconfigure c1::dsk/c1t0d0
```

障害が発生したディスク `c1t0d0` を物理的に取り外し、交換用ディスク `c1t0d0` を挿入します。一部のハードウェアでは、ディスクをオンラインにしたり、挿入された交換用ディスクを再構成したりする必要はありません。

```
# cfgadm -c configure c1::dsk/c1t0d0
```

交換用ディスク `c1t0d0s0` に SMI ラベルが付いており、スライス 0 が含まれていることを確認したら、交換作業に進みます。

```
# zpool replace rpool c1t0d0s0
# zpool online rpool c1t0d0s0
# zpool status rpool
```

再同期化が完了したあと、ブートブロックをインストールします。

```
# bootadm install-bootloader
```

ZFS スワップおよびダンプデバイスの管理

インストールプロセスは、ZFS ルートプール内の ZFS ボリューム上にスワップ領域とダンプデバイスを自動的に作成します。

ダンプデバイスは、クラッシュダンプが保存されるディレクトリの領域が不足している場合や、`dumpadm -n` コマンド構文を実行した場合に使用されます。`-n` は、システムのリブート後に `savecore` を自動的に実行しないようにダンプ構成を変更します。

特定のシステムでは、現在の Oracle Solaris リリースにある遅延ダンプ機能を利用しています。この機能を使用すると、システムダンプがシステムのリブート後もメモリー内に保持されるため、システムのリブート後にクラッシュダンプを分析できます。詳細は、『Oracle Solaris 11.3 でのデバイスの管理』 の 「デバイスおよび Oracle Hardware Management Pack について」を参照してください。

スワップおよびダンプボリュームを管理する場合は、次のガイドラインに注意してください。

- ルート以外のプールでは、スワップおよびダンプボリュームは、単一ディスク構成またはミラー化構成のどちらでもサポートされます。
- スワップ領域とダンプデバイスに個別の ZFS ボリュームを使用する必要があります。
- スパースボリュームはスワップボリュームではサポートされません。
- 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。

スワップおよびダンプ情報の表示

スワップ領域を表示するには、`swap -l` コマンドを使用します。例:

```
# swap -l
swapfile                  dev    swaplo   blocks     free
/dev/zvol/dsk/rpool/swap 145,2        16 16646128 16646128
```

ダンプ構成を表示するには、`dumpadm` コマンドを使用します。例:

```
# dumpadm
Dump content: kernel pages
Dump device: /dev/zvol/dsk/rpool/dump (dedicated)
Savecore directory: /var/crash/
Savecore enabled: yes
Save compressed: on
```

ルート以外のプールにスワップまたはダンプボリュームを手動で作成することもできます。ルート以外のプールにダンプデバイスを作成したあと、`dump -d` コマンドを実行して、そのデバイスをリセットすることも必要です。

次の例では、ルート以外のプール `bpool` 上にダンプデバイスが作成されます。

```
# zfs create -V 10g bpool/dump2
# dumpadm -d /dev/zvol/dsk/bpool/dump2
Dump content      : kernel with ZFS metadata
Dump device       : /dev/zvol/dsk/bpool/dump2 (dedicated)
Savecore directory: /var/crash
Savecore enabled  : yes
Save compressed   : on
```

ダンプデバイスとスワップデバイスの詳細は、『Oracle Solaris 11.3 でのファイルシステムの管理』 の 第 3 章,「追加スワップ空間の構成」および『Oracle Solaris 11.3 でのシステム管理のトラブルシューティング』を参照してください。

▼ スワップボリュームを作成する方法

この手順は、ルートプールとルート以外のプールの両方に適用されます。さらに多くのスワップ領域を必要としているが、既存のスワップデバイスがビジー状態である場合は、この同じ手順を使用して別のスワップボリュームを追加するだけです。

1. **スワップボリュームを作成します。**

   ```
   # zfs create -V size new-pool/swap
   ```

2. **テキストエディタを使用して、新しいスワップデバイスの `/etc/vfstab` エントリを更新します。**

   サンプルエントリについては、例22「スワップボリュームの手動での作成」を参照してください。

3. **既存のアクティブなスワップボリュームから新しいスワップボリュームに切り替える場合は、新しいスワップボリュームをアクティブ化します。**

   ```
   # swap -a path-to-new-swap-volume
   ```

4. **必要に応じて、システムをリブートします。**

例 22 スワップボリュームの手動での作成

この例では、プール `rpool` 内に 4G バイトの新しいスワップボリュームを作成します。この新しいスワップボリュームは、既存のスワップボリュームを置き換えることを目的にしています。

```
# zfs create -V 4g rpool2/swap2
# vi /etc/vfstab
/dev/zvol/dsk/rpool2/swap    -    -    swap -    no    -    vfstab エントリ

# swap -a /dev/zvol/dsk/rpool/swap2
```

▼ ダンプボリュームを作成する方法

この手順は、ルートプールまたはルート以外のプールのどちらを使用している場合にも適用されます。

1. **ダンプボリュームを作成します。**

   ```
   # zfs create -V size new-pool/dump
   ```

2. **ダンプデバイスをリセットします。**

   ```
   # dumpadm -d dump-path
   ```

3. **必要に応じて、システムをリブートします。**

例 23 ダンプボリュームの手動での作成

この例では、プール `rpool` 内に 4G バイトの新しいダンプボリュームを作成します。

```
# zfs create -V 4g rpool2/dump
```

```
# dumpadm -d /dev/zvol/dsk/rpool2/dump
```

ZFS スワップおよびダンプデバイスのサイズ調整

インストール後に、スワップおよび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。または、スワップおよびダンプボリュームの再作成が必要になることもあります。

デフォルトでは、スワップサイズに *n* ブロックを指定するとき、スワップファイルの先頭ページは自動的にスキップされます。このため、割り当てられる実際のサイズは、*n*-1 ブロックです。スワップファイルサイズを異なる方法で構成するには、`swaplow` オプションを `swap` コマンドに付けて使用します。`swap` コマンドのオプションの詳細については、swap(1M) マニュアルページを参照してください。

アクティブなシステムでスワップデバイスを削除する方法については、『Oracle Solaris 11.3 でのファイルシステムの管理』 の 「Oracle Solaris ZFS ルート環境でスワップ空間を追加する方法」を参照してください。

次の例は、さまざまな状況の下で既存のスワップおよびダンプデバイスを調整する方法を示しています。

例 24 ダンプデバイスの `volsize` プロパティーのリセット

大きなダンプデバイスのサイズ変更には時間がかかる場合があることに注意してください。

```
# zfs set volsize=2G rpool/dump
# zfs get volsize rpool/dump
NAME         PROPERTY  VALUE       SOURCE
rpool/dump   volsize   2G          -
```

例 25 ただちに使用するためのスワップボリュームのサイズ変更

この例は、スワップサイズを調整するための 2 つの方法を示しています。

最初の方法では、システムをリブートしなくてもスワップボリュームを調整できます。

```
# swap -l
swapfile                     dev    swaplo    blocks     free
/dev/zvol/dsk/rpool/swap   303,1         8   2097144  2097144
# zfs get volsize rpool/swap
NAME          PROPERTY   VALUE    SOURCE
rpool/swap    volsize       1G    local
# zfs set volsize=2g rpool/swap
# swap -l
swapfile                     dev    swaplo   blocks     free
```

```
/dev/zvol/dsk/rpool/swap   303,1         8  2097144  2097144
/dev/zvol/dsk/rpool/swap   303,1   2097160  2097144  2097144
```

スワップサイズを調整するためのこの 2 番目の方法では、新しいサイズを表示するにはシステムをリブートする必要があります。

```
# swap -d /dev/zvol/dsk/rpool/swap
# zfs set volsize=2G rpool/swap
# swap -a /dev/zvol/dsk/rpool/swap
# init 6
```

ZFS ダンプデバイスの問題のトラブルシューティング

このセクションでは、ダンプデバイスに関連した特定の問題と考えられる解決策について説明します。

- ダンプデバイスのサイズが小さすぎる。

  ダンプデバイスをリセットしたときに、出力に次のようなメッセージが含まれています。

  ```
  dumpadm: dump device dump-path is too small to hold a system dump
  ```

  このエラーを解決するには、ダンプデバイスのサイズを増やします。93 ページの「ZFS スワップおよびダンプデバイスのサイズ調整」を参照してください。

  スワップおよびダンプサイズの詳細は、『Oracle Solaris 11.3 でのファイルシステムの管理』 の 「スワップ空間の計画」を参照してください。
- ダンプデバイスが無効になっている。

  必要に応じて、新しいダンプデバイスを作成し、`dumpadm -d` コマンドを使用してそれを有効にします。92 ページの「ダンプボリュームを作成する方法」を参照してください。
- ルート以外のプールにダンプデバイスを追加できない。

  ダンプデバイスをリセットしたときに、出力に次のようなメッセージが含まれています。

  ```
  dump is not supported on device 'dump-path':
  'pool' has multiple top level vdevs
  ```

  最上位レベルのデバイスが複数存在するプールへのダンプデバイスの追加はサポートされていません。代わりに、ルートプールにダンプデバイスを追加してください。ルートプールでは、最上位レベルが 1 つである構成のみがサポートされます。
- クラッシュダンプが自動的に作成されなかった。

  この場合は、`savecore` コマンドを使用してクラッシュダンプを保存します。

ZFS ルートファイルシステムからのブート

SPARC システムと x86 システムの両方は、ブートアーカイブを使用してブートします。ブートアーカイブは、ブート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。ブート用に選択されたルートファイルシステムには、ブートアーカイブとカーネルファイルの両方のパス名が含まれています。

ZFS ブートの場合は、デバイス指定子によって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識別されます。ストレージプールには、複数のブート可能 ZFS ルートファイルシステムが含まれていることがあります。そのため、ブートデバイスと、そのブートデバイスによって識別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、ZFS ブートプロセスは、そのプールの `bootfs` プロパティーで定義されているファイルシステムを使用します。ただし、デフォルトのファイルシステムをオーバーライドできます。SPARC システムでは、`boot -z` コマンドを使用し、代替ブート可能ファイルシステムを指定できます。x86 システムでは、BIOS から代替ブートデバイスを選択できます。

`zpool replace` コマンドを使用してルートプールディスクを交換する場合は、交換用ディスクにブート情報をインストールする必要があります。ただし、ルートプールに追加のディスクを接続するだけの場合は、ブート情報のインストールは必要ありません。

ブート情報をインストールするには、次のいずれかの方法で `bootadm` コマンドを使用します。

- 既存のルートプールのディスクにブート情報をインストールするには、次のコマンドを使用します。

  ```
  # bootadm install-bootloader
  ```

- 代替プールにブート情報をインストールするには、次のコマンドを使用します。

  ```
  # bootadm install-bootloader -P alt-root-pool
  ```

- x86 システムの場合のみ、GRUB レガシーブートローダーをインストールする場合は、次のコマンドを使用します。

  ```
  x86# installgrub /boot/grub/stage1 /boot/grub/stage2 /dev/rdsk/c0t1d0s0
  ```

代替ルートプールディスクからのブート

注記 - このセクションの情報は、ミラー化ルートプールにのみ適用されます。

代替ルートプールディスクからブートする場合は、必要に応じてどのディスクからでもブートできるように、ルートプールのすべてのディスクが接続され、オンラインになっていることを確認してください。ほとんどのシステムでは、切り離されているディスクから直接ブートしたり、現在オフラインであるアクティブなルートプールディスクからブートしたりすることはできません。

SPARC システム上の代替ブートディスク

ミラー化ルートプール内のプライマリディスクは通常、デフォルトのブートデバイスです。別のデバイスからブートするには、ブートするためのコマンドを発行するときに、そのディスクを指定する必要があります。

デフォルトのブートデバイスを変更する場合は、まず、目的のデバイスを選択するためにプールの構成を表示します。次に、`OK` プロンプトで、システムの PROM を選択したデバイスで更新します。システムをブートし、選択したデバイスがアクティブなブートデバイスであることを確認します。

次の例では、`c1t1d0` をデフォルトのブートデバイスとして割り当てます。

```
# 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
    c1t0d0  ONLINE    0     0     0
    c1t1d0  ONLINE    0     0     0
...

ok boot /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/disk@1
```

システムがリブートされたあと、どのアクティブなブートデバイスがシステム内に存在するかを確認します。

```
# prtconf -vp | grep bootpath
bootpath:  '/pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/disk@1,0:a'
```

x86 システム上の代替ブートディスク

ブートディスクの順序が正しく設定されている、最新の BIOS が搭載された x86 ベースのシステムでは、プライマリルートプールディスクが切り離されているか、オフラインであるか、または使用できない場合、システムは自動的に 2 番目のデバイスからブートします。このようなシステムでは、次の例に示すように、どのアクティブなブートデバイスがシステム内に存在するかを確認するだけで済みます。

```
# 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 システムで ZFS ルートファイルシステムからブートする

複数の ZFS BE が存在するシステムでは、`beadm activate` コマンドを使用して、任意の BE からブートできます。インストールプロセスと `beadm` アクティブ化プロセスはどちらも、`bootfs` プロパティーを自動的に設定します。

デフォルトでは、`bootfs` プロパティーは、*/pool-name*`/boot/menu.lst` ファイル内のブート可能ファイルシステムエントリを識別します。ただし、`menu.lst` エントリには、プール内の代替ファイルシステムを指定する `bootfs` コマンドを含めることができます。そのため、`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のエントリが含まれている場合があります。

ZFS ルートファイルシステムがインストールされると、次の例のようなエントリが `menu.lst` ファイルに追加されます。

```
title release-version SPARC
bootfs rpool/ROOT/solaris
```

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。

```
title release-version SPARC
bootfs rpool/ROOT/solaris
title solaris
bootfs rpool/ROOT/solaris2
```

▼ SPARC: ブートのためのブート環境を選択する方法

1. **ZFS BE がアクティブ化されたら、ZFS プール内のブート可能ファイルシステムのリストを表示します。**

   ```
   # boot -L
   ```

2. **そのリスト内のブート可能ファイルシステムの 1 つを選択します。**

   そのファイルシステムをブートするための詳細な手順が表示されます。

3. **手順に従って、選択したファイルシステムをブートします。**

4. **特定の ZFS ファイルシステムをブートするには、boot -Z** *file system* **コマンドを使用します。**

5. **(オプション) 選択した BE をリブートのあとも永続的にするには、その BE をアクティブ化します。**

例 26 特定の ZFS ブート環境からブートする

システムのブートデバイスの ZFS ストレージプール内に複数の ZFS BE が存在する場合は、`beadm activate` コマンドを使用してデフォルトの BE を指定します。

この例では、`beadm` により、次の使用可能な ZFS BE が一覧表示されます。

```
# beadm list
BE        Active Mountpoint Space Policy Created
--        ------ ---------- ----- ------ -------
solaris   NR     /          3.80G static 2012-07-20 10:25
solaris-2 -      -          7.68M static 2012-07-19 13:44
```

特定の BE を選択するには、`boot -L` コマンドを使用します。例:

```
ok boot -L
Boot device: /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/disk@0,0:a  File and args: -L
1 release-version SPARC
2 solaris
Select environment to boot: [ 1 - 2 ]: 1

To boot the selected entry, invoke:
boot [<root-device>] -Z rpool/ROOT/solaris-2

Program terminated
ok boot -Z rpool/ROOT/solaris-2
```

選択した BE から自動的にブートするには、その BE をアクティブ化します。

x86 システムで ZFS ルートファイルシステムからブートする

Oracle Solaris 11.1 から、x86 ベースのシステムは GRUB2 とともにインストールされます。`menu.lst` ファイルは、`/rpool/boot/grub/grub.cfg` ファイルに置き換えられています。このファイルを手動で編集してはいけません。代わりに、`bootadm` サブコマンドを使用して、メニューエントリを追加、変更、および削除します。

注記 - システムの Oracle Solaris バージョンが引き続き従来の GRUB を使用している場合は、`menu.lst` ファイル内の ZFS ルートファイルシステムエントリが記載されている Booting From a ZFS Root File System on an x86 Based System を参照してください。

GRUB メニュー項目の変更の詳細は、『Oracle Solaris 11.3 システムのブートとシャットダウン』を参照してください。

x86: ルートファイルシステムの表示

GRUB2 システムで ZFS ルートファイルシステムからブートするとき、ブートデバイスは次のように指定されます。

```
# bootadm list-menu
the location of the boot loader configuration files is: /rpool/boot/grub
default 0
console text
timeout 30
0 release-version
```

x86: ZFS ルートファイルシステムの高速リブート

高速リブート機能は、x86 システム上で数秒間のうちにリブートする機能を提供します。高速リブート機能により、BIOS およびブートローダーによって発生する可能性のある長い遅延を回避して、新しいカーネルにリブートすることができます。

`beadm activate` コマンドで BE 間を移行するときは、引き続き `init 6` コマンドを使用する必要があります。その他のシステム動作については、必要に応じて `reboot` コマンドを使用します。

ZFS ルート環境での回復のためのブート

破損したブートローダーの問題、root パスワードの問題、または不正なシェルを解決するために、システムのブートが必要になることがあります。これらの各ケースの回復手順については、『Oracle Solaris 11.3 システムのブートとシャットダウン』 の 「復旧目的のシステムのシャットダウンおよびブート」を参照してください。

ルートプール内のディスクを交換する必要がある場合は、86 ページの「ZFS ルートプール内のディスクの交換」を参照してください。完全なシステム (ベアメタル) 回復を実行する必要がある場合は、『Oracle Solaris 11.3 でのシステム復旧とクローン』を参照してください。

♦ ♦ ♦ 第 7 章

7 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。

この章では、次の内容について説明します。

注記 - 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、またはボリュームを指す総称として使用されます。

ZFS ファイルシステムの概要

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。ZFS ファイルシステムは動的な作成および破棄が可能で、ベース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。これらのファイルシステムは非常に軽量であり、ZFS での管理の中心点であることから、多数作成する可能性があります。

ZFS ファイルシステムは zfs コマンドを使用して管理できます。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、クローン、およびボリューム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8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、227 ページの「ZFS ボリューム」を参照してください。

zfs コマンドの呼び出しには、常にファイルシステムの名前が必要です。ファイルシステム名は、次のように、プールの名前から始まるパス名として指定されます。

pool-name/[*dataset-path*]/*filesystem-name*

プール名およびデータセットパスは、階層内でのファイルシステムの場所を識別します。名前の最後の部分はファイルシステムの名前を識別します。このファイルシステム名は、24 ページの「ZFS コンポーネントの命名」にある命名要件を満たしている必要があります。たとえば、tank/home/jeff というファイルシステム名は、tank プールの /home データベースパスにある jeff という名前の ZFS ファイルシステムを示します。

ZFS ファイルシステムの作成、破棄、および名前変更を行う

このセクションの内容は次のとおりです。

▼ ZFS ファイルシステムの作成方法

zfs create コマンドによって、新しく作成されたファイルシステムが自動的にマウントされます (ファイルシステムが正常に作成されている場合)。デフォルトでは、ファイルシステムは create サブコマンドで指定したデータセット名を使用して、*/dataset* としてマウントされます。この例では、新しく作成した jeff ファイルシステムは /tank/home/jeff 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、131 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。

注記 - ZFS ファイルシステムの暗号化は、ファイルシステムの作成時に有効にする必要があります。ZFS ファイルシステムの暗号化の詳細については、154 ページの「ZFS ファイルシステムの暗号化」を参照してください。

`zfs create` コマンドの詳細は、`zfs(1M)` のマニュアルページを参照してください。

1. **ルートになります。**
2. **ZFS ファイルシステムを作成します。**

例 27 単純な ZFS ファイルシステムを作成する

次の例では、`jeff` 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/jeff
```

例 28 ファイルシステムプロパティーを使用して ZFS ファイルシステム階層を作成する

ファイルシステムが作成されるときにファイルシステム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、105 ページの「ZFS のプロパティーの概要」を参照してください。

▼ ZFS ファイルシステムを破棄する方法

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。デフォルトでは、データセットのすべてのスナップショットが破棄されます。破棄されたファイルシステムは、自動的にアンマウントおよび共有解除されます。自動的に管理されるマウントまたは自動的に管理される共有の詳細については、132 ページの「自動マウントポイント」を参照してください。

1. **ルートになります。**
2. **ZFS ファイルシステムを破棄します。**

```
# zfs destroy tank/home/mark
```

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。

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。

例 29 アクティブな ZFS ファイルシステムを破棄する

破棄するファイルシステムがビジー状態でアンマウントできない場合、`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アンマウント、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 -f tank/home/matt
```

例 30 子孫のある ZFS ファイルシステムを破棄する

`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。

```
# zfs destroy tank/ws
cannot destroy 'tank/ws': filesystem has children
use '-r' to destroy the following datasets:
tank/ws/jeff
tank/ws/bill
tank/ws/mark
# zfs destroy -r tank/ws
```

例 31 依存関係のある ZFS ファイルシステムを破棄する

破棄されるファイルシステムが間接的な依存関係を持っている場合は、再帰的な destroy コマンドでさえも失敗します。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。このオプションは特に慎重に使用してください。

```
# zfs destroy -r tank/home/eric
cannot destroy 'tank/home/eric': filesystem has dependent clones
use '-R' to destroy the following datasets:
tank//home/eric-clone
# zfs destroy -R tank/home/eric
```

▼ ZFS ファイルシステムの名前を変更する方法

`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ると、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します。

- ZFS 階層内でファイルシステムの場所を移動します。
- 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します。

`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) をアンマウント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をアンマウントできない場合、`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アンマウントする必要があります。XREF!!!!

1. **ルートになります。**

2. **ZFS ファイルシステムの名前を変更します。**
   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 `eric` から `eric_old` に変更しています。

例 32 ZFS ファイルシステムの場所を移動する

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/mark tank/ws/mark
```

この例では、`mark` 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、147 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 のプロパティーの概要

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メカニズムを使用します。このセクションで定義されているプロパティーは、ほかに説明のある場合を除いて、すべての種類のデータセットに適用されます。

- 115 ページの「ZFS の読み取り専用のネイティブプロパティー」
- 116 ページの「設定可能な ZFS ネイティブプロパティー」
- 122 ページの「ZFS の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、内部の統計情報を提供するか、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、122 ページの「ZFS の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、親ファイルシステム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。

すべての継承可能なプロパティーには、プロパティーがどのように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を使用できます。

`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。126 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、122 ページの「ZFS のユーザープロパティー」を参照してください。

表 3 ZFS のネイティブプロパティーの説明

プロパティー名	タイプ	デフォルト値	説明
`aclinherit`	文字列	`secure`	ファイルやディレクトリが作成されたときに ACL エントリがどのように継承されるかを制御します。値は、`discard`、`noallow`、`secure`、および `passthrough` です。これらの値については、199 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` 操作中に ACL エントリがどのように変更されるかを制御します。値は、`discard`、`groupmask`、および `passthrough` です。これらの値については、199 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`	ファイルが読み取られたときに、そのファイルのアクセス時間が更新されるかどうかを制御します。このプロパティーを

プロパティー名	タイプ	デフォルト値	説明
			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	N/A	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場合に、ファイルシステムおよびそのすべての子が使用できるディスク容量を識別する読み取り専用プロパティー。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、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。
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、ファイルシステムは明示的なマウントおよびアンマウントのみ可能になります。ファイルシステムの作成時やインポート時に、ファイルシステム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 コマンドでアンマウントされることもありません。 詳細は、117 ページの「canmount プロパティー」を参照してください。
casesensitivity	文字列	mixed	このプロパティーは、ファイルシステムで使用するファイル名照合アルゴリズムで、大文字と小文字を区別する (casesensitive) か、区別しない (caseinsensitive) か、または両方の照合方式の組み合わせを許可する (mixed) かを指定します。通常、UNIX および POSIX のファイルシステムでは、ファイル名の大文字と小文字を区別します。 このプロパティーの値が mixed の場合、ファイルシステムが要求に応じて大文字と小文字を区別する照合も区別しない照合もサポートできることを示します。現在、混合動作をサポートするファイルシステムで大文字と小文字を区別しない照合が可能なのは、Oracle Solaris SMB サーバー製品に限られています。mixed 値の使用方法については、117 ページの「casesensitivity プロパティー」を参照してください。 casesensitivity プロパティー設定には関係なく、ファイルシステムは、ファイルを作成するために指定された名前の大文字または小文字を保持します。このプロパティーは、ファイルシステムの作成後には変更できません。
checksum	文字列	on	データの整合性を検証するために使用され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、

プロパティー名	タイプ	デフォルト値	説明
			off、fletcher2、fletcher4、sha256、および sha256+mac 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。これらの値は、on、off、lzjb、lz4、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となります。 このプロパティーの省略名は compress です。
compressratio	数値	N/A	データセットに適用される圧縮率を識別する読み取り専用プロパティー。乗数で表現されます。zfs set compression=on *dataset* コマンドを使用すると、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しく書き込まれたデータにしか影響しないため、このプロパティーの設定はファイルシステムの作成時に検討します。
creation	文字列	N/A	データセットが作成された日付と時間を識別する読み取り専用プロパティー。
dedup	文字列	off	ZFS ファイルシステムの重複したデータを削除する機能を制御します。設定できる値は、on、 off、verify、および sha256[,verify] です。デフォルトの複製解除のチェックサムは sha256 です。 詳細は、119 ページの「dedup プロパティー」を参照してください。
defaultgroupquota	文字列	なし	デフォルトのグループ割り当て制限を設定します。値は、明示的なグループ割り当て制限が指定されていないすべてのグループに適用されます。デフォルト値は none です。詳細は、151 ページの「デフォルトのユーザーおよびグループの割り当て制限を設定する」を参照してください。
defaultuserquota	文字列	なし	デフォルトのユーザー割り当て制限を設定します。値は、明示的なユーザー割り当て制限が指定されていないすべてのユーザーに適用されます。デフォルト値は none です。詳細は、151 ページの「デフォルトのユーザーおよびグループの割り当て制限を設定する」を参照してください。

プロパティー名	タイプ	デフォルト値	説明
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くことができるかどうかを制御します。
encryption	ブール型	off	ファイルシステムが暗号化されるかどうかを制御します。暗号化されたファイルシステムでは、データはエンコードされ、ファイルシステムの所有者はデータにアクセスするときに鍵が必要になります。
exec	ブール型	on	ファイルシステム内のプログラムの実行を許可するかどうかを制御します。また、off に設定されているとき、mmap(2) に PROT_EXEC を付けた呼び出しは禁止されます。
keychangedate	文字列	なし	指定されたファイルシステムで zfs key -c 操作からラッピング鍵を最後に変更した日付を識別する読み取り専用プロパティー。鍵の変更操作が発生していない場合、このプロパティーの値はファイルシステムの作成日と同じです。
keysource	文字列	なし	ファイルシステム鍵をラップする鍵の形式と場所を識別します。有効なプロパティー値は、raw、hex、passphrase,prompt、または *file* です。zfs key -l コマンドを使用してファイルシステムを作成、マウント、またはロードするときに、この鍵が存在している必要があります。新しいファイルシステムで暗号化を有効にする場合、デフォルトの keysource は passphrase,prompt です。
keystatus	文字列	なし	ファイルシステムの暗号化鍵ステータスを識別する読み取り専用プロパティー。ファイルシステムの鍵が使用できるかどうかは、available か unavailable で示されます。暗号化が有効になっていないファイルシステムの場合、none が表示されます。
logbias	文字列	latency	ZFS がこのファイルシステムに対する同期要求を最適化する方法を制御します。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。
mlslabel	文字列	なし	マルチレベルファイルシステムでの mlslabel プロパティーの動作については、multilevel プロパティーを参照してください。次の mlslabel の説明は非マルチレベルファイルシステムに適用されます。 ファイルシステムをゾーンにマウントできるかどうかを判別する応答性ラベルを提供します。ラベル付きのファイルシステムがラベル付きのゾーンに一致する場合、ファイルシステムは、そのラベル付きのゾーンからマウントしてアクセスできます。デフォルト値は none です。このプロパティーは、適切な権限を使用してのみ変更できます。
mounted	ブール型	N/A	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示す読み取り専用のプロパティー。このプロパティーは、ボリュームには適用されません。値は、yes または no です。
mountpoint	文字列	N/A	このファイルシステムに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

プロパティー名	タイプ	デフォルト値	説明
			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アンマウントされます。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、131 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。
multilevel	ブール型	off	マルチレベルファイルシステム内のオブジェクトは、自動的に生成される明示的な機密ラベル属性で個別にラベル付けされています。このラベル属性を変更してオブジェクトに再度適切なラベル付けを行うには、setlabel または setflabel インタフェースを使用します。デフォルトはオフです。 ルートファイルシステム、Oracle Solaris ゾーンファイルシステム、またはパッケージ化された Solaris コードを含むファイルシステムは、マルチレベルにしないようにしてください。 マルチレベルファイルシステムの mlslabel プロパティーにはいくつかの違いがあります。mlslabel 値は、ファイルシステム内のオブジェクトに対して使用可能な最上位のラベルを定義します。mlslabel 値よりも上位のラベルでファイルを作成 (またはそのレベルにファイルを再度ラベル付け) しようとしても許可されません。mlslabel 値に基づいたマウントポリシーはマルチレベルファイルシステムには適用されません。 マルチレベルファイルシステムの場合、ファイルシステムの作成時に mlslabel プロパティーを明示的に設定できます。それ以外の場合は、ADMIN_HIGH というデフォルトの mlslabel プロパティーが自動的に作成されます。マルチレベルファイルシステムを作成したあとで mlslabel プロパティーを変更できますが、それを下位のラベルに設定したり、none に設定したり、削除したりすることはできません。
nbmand	ブール型	off	nbmand (非ブロッキング強制) ロックでファイルシステムがマウントされるかどうかを制御します。このプロパティーは SMB クライアント専用です。このプロパティーの変更は、ファイルシステムをアンマウントしてから再マウントした場合にのみ反映されます。
normalization	文字列	なし	このプロパティーは、2 つのファイル名が比較される場合に常にファイルシステムがファイル名の unicode 正規化を実行するかどうか、およびどの正規化アルゴリズムが使用されるかを示します。保存されるファイル名が変更されることはなく、名前の正規化は比較処理の一部として実行されます。このプロパティーが none 以外の有効な値に設定されており、utf8only プロパティーが指定されなかった場合、utf8only プロパティーは自動的に on に設定されます。normalization プロパティーのデフォルト値は none で

プロパティー名	タイプ	デフォルト値	説明
			す。このプロパティーは、ファイルシステムの作成後には変更できません。
origin	文字列	N/A	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。 クローニングされていないファイルシステムの作成元は none になります。
primarycache	文字列	all	プライマリキャッシュ (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、メタデータのみがキャッシュされます。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。
quota	数値 (または none)	none	ファイルシステムと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。割り当てがすでに設定されているファイルシステムの子孫に割り当てを設定した場合は、祖先の割り当てはオーバーライドされずに、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、148 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 に設定されているときは、変更を行うことができません。 このプロパティーの省略名は rdonly です。
recordsize	数値	128K	ファイルシステム内の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、121 ページの「recordsize プロパティー」を参照してください。
referenced	数値	N/A	データセットからアクセスできるデータの量を識別する読み取り専用プロパティーであり、これはプール内のほかのデータセットとともに共有される場合と共有されない場合があります。 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

プロパティー名	タイプ	デフォルト値	説明
			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`	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、親データセットの使用済み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。
`rekeydate`	文字列	N/A	このファイルシステムで `zfs key -K` または `zfs clone -K` 操作からデータ暗号化鍵を最後に変更した日付を示す読み取り専用プロパティー。`rekey` 操作が実行されていない場合は、このプロパティーの値は `creation` 日と同じです。
`reservation`	数値 (または none)	`none`	ファイルシステム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場合、ファイルシステムは予約に指定された容量を使用しているように扱われます。予約は、親ファイルシステムのディスク容量に計上されるため、親ファイルシステム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、152 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。
`rstchown`	ブール型	on	ファイルシステムの所有者がファイルの所有権の変更を許可できるかどうかを示します。デフォルトでは `chown` 操作を制限します。`rstchown` が off に設定されていると、`chown` 操作に対する `PRIV_FILE_CHOWN_SELF` 権限がユーザーに与えられます。
`secondarycache`	文字列	`all`	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、メタデータのみがキャッシュされます。

プロパティー名	タイプ	デフォルト値	説明
setuid	ブール型	on	setuid ビットをファイルシステムで考慮するかどうかを制御します。
shadow	文字列	None	ZFS ファイルシステムを *URI* で記述されたファイルシステムの *shadow* として識別します。このプロパティーを設定すると、URI で識別されたファイルシステムから shadow ファイルシステムにデータが移行されます。完全な移行のためには、移行されるファイルシステムが読み取り専用である必要があります。
share.nfs	文字列	off	ZFS ファイルシステムの NFS 共有が作成および公開されるかどうか、およびどのようなオプションが使用されるかを制御します。また、zfs share および zfs unshare コマンドを使用して、NFS 共有を公開および非公開にすることもできます。zfs share コマンドを使用して NFS 共有を公開するには、NFS 共有プロパティーも設定されている必要があります。NFS 共有プロパティーの設定の詳細については、136 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。 ZFS ファイルシステムの共有の詳細は、136 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。
share.smb	文字列	off	ZFS ファイルシステムの SMB 共有を作成および公開するかどうか、およびどのようなオプションを使用するかを制御します。また、zfs share および zfs unshare コマンドを使用して、SMB 共有を公開および非公開にすることもできます。zfs share コマンドを使用して SMB 共有を公開するには、SMB 共有プロパティーも設定されている必要があります。SMB 共有プロパティーの設定の詳細については、136 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については、167 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。
sync	文字列	standard	ファイルシステムのトランザクションの同期動作を決定します。指定できる値は次のとおりです。 ■ standard。デフォルト値であり、この値に設定すると、fsync、O_DSYNC、O_SYNC などの、ファイルシステムの同期トランザクションがインテントログに書き込まれます。 ■ always。すべてのファイルシステムトランザクションが書き込まれ、呼び戻すシステムコールによって安定したストレージに書き出されるようにします。この値には、パフォーマンス上の大きなデメリットがあります。 ■ disabled。同期要求が無効になります。ファイルシステムトランザクションは、次のトランザクショングループの確定時にのみ安定したストレージに確定されます。これには数秒かかる場合があります。この値はパフォーマンスがもっとも高くなり、プールを破壊してしまうリスクもありません。

プロパティー名	タイプ	デフォルト値	説明
			注意 - ZFS がデータベースや NFS 操作などのアプリケーションの同期トランザクション要求を無視することになるので、この `disabled` 値は非常に危険です。現在有効なルートまたは `/var` ファイルシステムでこの値を設定すると、予期しない動作、アプリケーションデータの損失、リプレー攻撃に対する脆弱性の増大という結果を招く可能性があります。関連するすべてのリスクを完全に理解している場合にのみ、この値を使用してください。
`type`	文字列	N/A	`filesystem` (ファイルシステムまたはクローン)、`volume`、または `snapshot` としてデータセットの種類を識別する読み取り専用プロパティー。
`used`	数値	N/A	データセットおよびそのすべての子孫によって消費されたディスク容量を識別する読み取り専用プロパティー。 詳細については、115 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容量を識別する読み取り専用プロパティー。これは、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`	数値	`off`	データセット自体によって使用されるディスク容量を識別する読み取り専用プロパティー。このディスク領域は、最初にすべての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あとに、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`	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容量を識別する読み取り専用プロパティー。このディスク領域は、`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`	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容量を識別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。このプロパティーの省略名は `usedsnap` です。
`utf8only`	ブール型	`Off`	このプロパティーは、ファイルシステムが UTF-8 文字コードセットに存在しない文字を含むファイル名を拒否するかどうかを示します。このプロパティーが明示的に `off` に設定されている場合、`normalization` プロパティーを明示的に設定しないか、または `none` に設定する必要があります。`utf8only` プロパティーのデフォルト値は `off` です。このプロパティーは、ファイルシステムの作成後には変更できません。
`version`	数値	N/A	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。

プロパティー名	タイプ	デフォルト値	説明
volblocksize	数値	8 KB	ボリュームに対して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 - 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。
volsize	数値	N/A	ボリュームに対してボリュームの論理サイズを指定します。 詳細については、121 ページの「volsize プロパティー」を参照してください。
vscan	ブール型	Off	ファイルを開いたり閉じたりするときに、ウイルスがないか通常ファイルがスキャンされるかどうかを制御します。このプロパティーを有効にする以外に、サードパーティー製のウイルススキャンソフトウェアを所有している場合は、ウイルススキャンが行われるようにウイルススキャンサービスも有効にしておく必要があります。デフォルト値は off です。
xattr	ブール型	on	このファイルシステムに対して拡張属性を有効 (on) にするか、無効 (off) にするかを示します。
zoned	ブール型	N/A	ファイルシステムが非大域ゾーンに追加されているかどうかを示します。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、230 ページの「ゾーンがインストールされている Oracle Solaris システムで ZFS を使用する」を参照してください。

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3の説明の中で記載しています。

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使用するディスク容量、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、使用済み領域と予約のどちらか大きい方になります。

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。

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。詳細については、表3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 126 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 102 ページの「ZFS ファイルシステムの作成方法」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。割り当て制限および予約の詳細については、147 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3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。

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 に設定する場合は、mountpoint プロパティーを none に設定する場合に似ていますが、継承可能な通常の mountpoint プロパティーをファイルシステム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、コンテナでプロパティーを設定できるように親ファイルシステムはコンテナとして機能しますが、コンテナ自体には決して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 に設定することは、ファイルシステムは自動的ではなく、明示的なマウントのみが可能になることを意味します。

casesensitivity プロパティー

このプロパティーは、ファイルシステムで使用するファイル名照合アルゴリズムで、大文字と小文字を区別する (casesensitive) か、区別しない (caseinsensitive) か、または両方の照合方式の組み合わせを許可する (mixed) かを指定します。

大文字と小文字を区別しない照合要求が、両方の照合方式が混在するファイルシステムで行われた場合、その動作は通常、純粋に大文字と小文字を区別しないファイルシステムで予想される動作と同じです。異なる点は、両方の照合方式が混在するファイルシステムが、大文字と小文字を区別する見方からは一意だが大文字と小文字を区別しない見方からは一意ではない複数の名前を持つディレクトリを含むことができるという点です。

たとえば、ディレクトリには foo、Foo、および FOO というファイルを含めることができます。foo のありうる形式 (たとえば、foo、FOO、FoO、 fOo など) のいずれかに

大文字と小文字を区別しないで一致するものを求める要求が行われた場合、照合アルゴリズムにより、既存の 3 つのファイルのいずれかが一致した結果として選択されます。一致した結果としてどのファイルがアルゴリズムに選択されるかは厳密には保証されませんが、`foo` のすべての形式に一致する結果として同じファイルが選択されるということは保証されます。`foo`、`FOO` 、`foO`、`Foo` などに大文字と小文字を区別しないで一致した結果として選択されるファイルは、ディレクトリが変更されないかぎり常に同じです。

`utf8only`、`normalization`、および `casesensitivity` プロパティーはまた、ZFS 委任管理を使用して権限のないユーザーに割り当てることができる新しいアクセス権を提供します。詳細については、216 ページの「ZFS アクセス権の委任」を参照してください。

`copies` プロパティー

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 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。例:

```
# 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。

注記 - 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。

dedup プロパティー

dedup プロパティーは、重複したデータをファイルシステムから削除するかどうかを制御します。ファイルシステムで dedup プロパティーが有効になっている場合、重複データブロックが同期的に削除されます。この結果、一意のデータだけが格納され、共通のコンポーネントがファイル間で共有されます。

次の考慮事項を確認するまで、本稼働システムにあるファイルシステムでは、dedup プロパティーを有効にしないでください。

1. 複製解除による領域の節約がデータに有益であるかどうかを判断します。zdb -S コマンドを使用して、プールでの複製解除の有効化による領域の節約の可能性をシミュレートすることができます。このコマンドは静止しているプール上で実行する必要があります。重複除去できないデータの場合は、重複除去を有効にしても意味がありません。例:

```
# zdb -S tank
Simulated DDT histogram:
bucket              allocated                       referenced
______   ______________________________   ______________________________
refcnt   blocks   LSIZE   PSIZE   DSIZE   blocks   LSIZE   PSIZE   DSIZE
------   ------   -----   -----   -----   ------   -----   -----   -----
1         2.27M    239G    188G    194G    2.27M    239G    188G    194G
2          327K   34.3G   27.8G   28.1G     698K   73.3G   59.2G   59.9G
4         30.1K   2.91G   2.10G   2.11G     152K   14.9G   10.6G   10.6G
8         7.73K    691M    529M    529M    74.5K   6.25G   4.79G   4.80G
16          673   43.7M   25.8M   25.9M    13.1K    822M    492M    494M
32          197   12.3M   7.02M   7.03M    7.66K    480M    269M    270M
64           47   1.27M    626K    626K    3.86K    103M   51.2M   51.2M
128          22    908K    250K    251K    3.71K    150M   40.3M   40.3M
256           7    302K     48K   53.7K    2.27K   88.6M   17.3M   19.5M
512           4    131K   7.50K   7.75K    2.74K    102M   5.62M   5.79M
2K            1      2K      2K      2K    3.23K   6.47M   6.47M   6.47M
8K            1    128K      5K      5K    13.9K   1.74G   69.5M   69.5M
Total     2.63M    277G    218G    225G    3.22M    337G    263G    270G

dedup = 1.20, compress = 1.28, copies = 1.03, dedup * compress / copies = 1.50
```

   推定される dedup 比率が 2 より大きい場合は、dedup によって領域が節約される可能性があります。

   上記の例では重複除去比は 2 よりも小さいので、重複除去はお勧めしません。
2. システムに dedup をサポートするための十分なメモリーがあることを確認してください。
   - コア内の各 dedup テーブルエントリは、およそ 320 バイトです。

- 割り当てられているブロック数に 320 を掛けます。例:

```
in-core DDT size = 2.63M x 320 = 841.60M
```

3. dedup のパフォーマンスは、複製解除テーブルがメモリーに入る場合に最適になります。dedup テーブルをディスクに書き込む必要がある場合は、パフォーマンスが低下します。たとえば、重複除去を有効にして大きなファイルシステムを削除すると、システムが上記のメモリー要件を満たしていない場合、システムパフォーマンスが大幅に低下します。

`dedup` が有効な場合、`dedup` チェックサムアルゴリズムによって `checksum` プロパティーがオーバーライドされます。プロパティー値を `verify` に設定することは、`sha256,verify` を指定することと同等です。プロパティーが `verify` に設定されており、2 つのブロックに同じ署名がある場合、ZFS は既存のブロックとバイト単位の比較を行なって、内容が同一であることを確認します。

このプロパティーは、ファイルシステムごとに有効にできます。例:

```
# zfs set dedup=on tank/home
```

`zfs get` コマンドを使用して、 `dedup` プロパティーが設定されているかどうかを判別できます。

複製解除はファイルシステムプロパティーとして設定されますが、その適用範囲はプール全体に及びます。たとえば、複製解除比を指定できます。例:

```
# zpool list tank
NAME    SIZE  ALLOC   FREE    CAP  DEDUP  HEALTH  ALTROOT
rpool   136G  55.2G  80.8G    40%  2.30x  ONLINE  -
```

`DEDUP` 列には、どれだけの複製解除が行われたかが示されます。いずれかのファイルシステムで `dedup` プロパティーが有効になっていない場合、または `dedup` プロパティーがファイルシステムで少し前まで有効であった場合、`DEDUP` 比は `1.00x` です。

`zpool get` コマンドを使用して、`dedupratio` プロパティーの値を判定できます。例:

```
# zpool get dedupratio export
NAME   PROPERTY    VALUE  SOURCE
rpool  dedupratio  3.00x  -
```

このプールプロパティーは、このプールでどれだけのデータ複製解除が達成されたかを示します。

encryption プロパティー

暗号化プロパティーを使用して、ZFS ファイルシステムを暗号化できます。詳細については、154 ページの「ZFS ファイルシステムの暗号化」を参照してください。

recordsize プロパティー

recordsize プロパティーは、ファイルシステム内の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 - 1M 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。

share.smb プロパティー

このプロパティーは、Oracle Solaris SMB サービスとの ZFS ファイルシステムの共有を有効にし、使用されるオプションを指定します。

プロパティーを off から on に変更すると、そのプロパティーを継承するすべての共有が現在のオプションで再共有されます。このプロパティーが off に設定されている場合、このプロパティーを継承する共有は共有解除されます。share.smb プロパティーの使用の例については、136 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。

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、特に注意するようにしてください。

ボリュームの使用方法の詳細については、227 ページの「ZFS ボリューム」を参照してください。

ZFS の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名は次の規則に準拠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* コンポーネントには、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.oracle.` 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。

例:

```
# 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。

例:

```
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クリアするには、`zfs inherit` コマンドを使用します。例:

```
# 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。このセクションでは、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、105 ページの「ZFS のプロパティーの概要」を参照してください。

例:

```
# zfs list
users                   2.00G  64.9G    32K  /users
users/home              2.00G  64.9G    35K  /users/home
users/home/cindy         548K  64.9G   548K  /users/home/cindy
users/home/mark         1.00G  64.9G  1.00G  /users/home/mark
users/home/neil         1.00G  64.9G  1.00G  /users/home/neil
```

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。例:

```
# zfs list -t all -r users/home/mark
NAME                        USED  AVAIL  REFER  MOUNTPOINT
users/home/mark            1.00G  64.9G  1.00G  /users/home/mark
users/home/mark@yesterday      0      -  1.00G  -
users/home/mark@today          0      -  1.00G  -
```

`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。例:

```
# zfs list /user/home/mark
NAME              USED  AVAIL  REFER  MOUNTPOINT
users/home/mark  1.00G  64.9G  1.00G  /users/home/mark
```

次の例は、tank/home/gina およびそのすべての子孫ファイルシステムに関する基本情報を表示する方法を示しています。

```
# zfs list -r users/home/gina
NAME                          USED  AVAIL  REFER  MOUNTPOINT
users/home/gina               2.00G  62.9G    32K  /users/home/gina
users/home/gina/projects      2.00G  62.9G    33K  /users/home/gina/projects
users/home/gina/projects/fs1  1.00G  62.9G  1.00G  /users/home/gina/projects/fs1
users/home/gina/projects/fs2  1.00G  62.9G  1.00G  /users/home/gina/projects/fs2
```

zfs list コマンドの詳細は、zfs(1M) のマニュアルページ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、105 ページの「ZFS のプロパティーの概要」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 share.nfs および 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 -r -o name,share.nfs,mountpoint users/home
NAME                          NFS       MOUNTPOINT
users/home                    on        /users/home
users/home/cindy              on        /users/home/cindy
users/home/gina               on        /users/home/gina
users/home/gina/projects      on        /users/home/gina/projects
users/home/gina/projects/fs1  on        /users/home/gina/projects/fs1
users/home/gina/projects/fs2  on        /users/home/gina/projects/fs2
users/home/mark               on        /users/home/mark
users/home/neil               on        /users/home/neil
```

-t オプションを使用して、表示するデータセットのタイプを指定できます。次の表は、有効な種類について説明しています。

表 4 ZFS オブジェクトの種類

タイプ	説明
filesystem	ファイルシステムとクローン
volume	ボリューム

タイプ	説明
share	ファイルシステム共有
snapshot	スナップショット

-t オプションには、表示されるデータセットのタイプのコンマ区切りリストを指定します。次の例は、-t および -o オプションを使用して、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ます。

```
# zfs list -r -t filesystem -o name,used users/home
NAME                           USED
users/home                    4.00G
users/home/cindy               548K
users/home/gina               2.00G
users/home/gina/projects      2.00G
users/home/gina/projects/fs1  1.00G
users/home/gina/projects/fs2  1.00G
users/home/mark               1.00G
users/home/neil               1.00G
```

-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。次の例は、zfs list コマンドに -H オプションを付けて使用した場合に生成される出力を示しています。

```
# zfs list -r -H -o name users/home
users/home
users/home/cindy
users/home/gina
users/home/gina/projects
users/home/gina/projects/fs1
users/home/gina/projects/fs2
users/home/mark
users/home/neil
```

不完全な ZFS データセットを一覧表示する

不完全なデータセットは、zfs receive を実行することによって開始されたデータセット転送が中断されたときに作成されます。不完全なデータセットを表示するには、zfs list -I コマンドを使用できます。各データセットの状態は、receiving (受信中) または resumable (再開可能) のいずれかです。-I オプションの引数は、all、resumable、または receiving です。

```
# zfs list -I all
NAME            USED  AVAIL  REFER  TYPE    STATE
users/home/dst  189M   910G   189M  volume  resumable
```

次のコマンドを使用して、再開可能なデータセットの名前のみ表示できます。

```
# # zfs list -HI resumable
users/home/dst
```

zfs list を使用して解析可能な出力を作成する

zfs list -o コマンドで -p オプションを使用して、マシンによる解析が可能な数値出力を作成できます。例:

```
# zfs list -o guid users/home
GUID
3.30E
# zfs list -po guid users/home
GUID
3807001345661527925
```

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。

- 126 ページの「ZFS プロパティーを設定する」
- 127 ページの「ZFS のプロパティーの継承」
- 128 ページの「ZFS プロパティーのクエリー検索」

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、116 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。zfs set の各呼び出しでは、プロパティーを 1 つだけ設定または変更できます。

次の例では、tank/home の atime を off に設定します。

```
# 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。例:

```
# 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、users/home/mark ファイルシステムの quota プロパティーに 20G バイトの値を設定しています。

```
# zfs set quota=20G users/home/mark
# zfs set quota=20g users/home/mark
# zfs set quota=20GB users/home/mark
# zfs set quota=20gb users/home/mark
```

使用率が 100% のファイルシステムにプロパティー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。

```
# zfs set quota=20gb users/home/mark
cannot set property for '/users/home/mark': out of space
```

数値以外のプロパティーの値は大文字と小文字が区別され、mountpoint を除いて小文字である必要があります。このプロパティーの値は、大文字と小文字を混在させることができます。

zfs set コマンドの詳細は、zfs(1M) のマニュアルページを参照してください。

ZFS のプロパティーの継承

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、親ファイルシステムから値を継承します。ただし、子孫ファイルシステム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、プロパティー値をクリアし、値が親ファイルシステムから継承されるようにすることができます。

次の例では、zfs set コマンドを使用して tank/home/jeff 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/jeff
# zfs get -r compression tank/home
NAME                  PROPERTY     VALUE     SOURCE
tank/home             compression  off       default
tank/home/eric        compression  off       default
tank/home/eric@today  compression  -         -
tank/home/jeff        compression  on        local
# zfs inherit compression tank/home/jeff
# zfs get -r compression tank/home
NAME                  PROPERTY     VALUE     SOURCE
tank/home             compression  off       default
```

```
tank/home/eric        compression  off       default
tank/home/eric@today  compression  -         -
tank/home/jeff        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
```

注記 - -r オプションを使用すると、すべての子孫のファイルシステムに割り当てられている現在のプロパティー設定がクリアされることに注意してください。

zfs inherit コマンドの詳細は、zfs(1M) のマニュアルページを参照してください。

ZFS プロパティーのクエリー検索

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、123 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、複雑なクエリーおよびスクリプトについては、zfs get コマンドを使用して、より詳しい情報をカスタマイズされた形式で提供してください。

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 5 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。一時的なマウントポイント

ソース値	説明
	プロパティーの詳細については、135 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  aclinherit            restricted             default
tank/home  aclmode               discard                default
tank/home  atime                 on                     default
tank/home  available             274G                   -
tank/home  canmount              on                     default
tank/home  casesensitivity       mixed                  -
tank/home  checksum              on                     default
tank/home  compression           off                    default
tank/home  compressratio         1.00x                  -
tank/home  copies                1                      default
tank/home  creation              Tue Jul 30 10:08 2013  -
tank/home  dedup                 off                    default
tank/home  defaultgroupquota     none                   -
tank/home  defaultuserquota      none                   -
tank/home  devices               on                     default
tank/home  encryption            off                    -
tank/home  exec                  on                     default
tank/home  keychangedate         -                      default
tank/home  keysource             none                   default
tank/home  keystatus             none                   -
tank/home  logbias               latency                default
tank/home  mlslabel              none                   -
tank/home  mounted               yes                    -
tank/home  mountpoint            /tank/home             default
tank/home  multilevel            off                    -
tank/home  nbmand                off                    default
tank/home  normalization         none                   -
tank/home  primarycache          all                    default
tank/home  quota                 none                   default
tank/home  readonly              off                    default
tank/home  recordsize            128K                   default
tank/home  referenced            31K                    -
tank/home  refquota              none                   default
tank/home  refreservation        none                   default
tank/home  rekeydate             -                      default
tank/home  reservation           none                   default
tank/home  rstchown              on                     default
tank/home  secondarycache        all                    default
tank/home  setuid                on                     default
tank/home  shadow                none                   -
tank/home  share.*               ...                    default
tank/home  snapdir               hidden                 default
tank/home  sync                  standard               default
tank/home  type                  filesystem             -
tank/home  used                  31K                    -
tank/home  usedbychildren        0                      -
tank/home  usedbydataset         31K                    -
tank/home  usedbyrefreservation  0                      -
tank/home  usedbysnapshots       0                      -
tank/home  utf8only              off                    -
```

```
tank/home  version              6                    -
tank/home  vscan                off                  default
tank/home  xattr                on                   default
tank/home  zoned                off                  default
```

zfs get の -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タイプで指定できます。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。次の例は、tank/ws で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ます。

```
# zfs get -s local all tank/ws
NAME     PROPERTY             VALUE                  SOURCE
tank/ws  compression          on                     local
```

前述のどのオプションの場合にも、-r オプションを組み合わせて、指定したファイルシステム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できます。次の例では、tank/home に含まれるすべてのファイルシステムについての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/home
NAME             PROPERTY       VALUE                    SOURCE
tank/home          atime          off                    temporary
tank/home/jeff     atime          off                    temporary
tank/home/mark     quota          20G                    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。例:

```
# zfs get -s local all
tank/home               atime          off                  local
tank/home/jeff          atime          off                  local
tank/home/mark          quota          20G                  local
```

zfs get コマンドの詳細は、zfs(1M) のマニュアルページ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 オプションを使用して、次の方法で出力をカスタマイズできます。

- リテラル name は、105 ページの「ZFS のプロパティーの概要」セクション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 およびその子孫のファイルシステム名と 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
```

ZFS ファイルシステムをマウントする

このセクションでは、ZFS でファイルシステムをマウントする方法について説明します。

- 131 ページの「ZFS マウントポイントを管理する」
- 133 ページの「ZFS ファイルシステムをマウントする」
- 135 ページの「一時的なマウントプロパティーを使用する」
- 135 ページの「ZFS ファイルシステムをアンマウントする」

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。このセクションで説明するように、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ファイルシステム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。プールの作成方法については、27 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 のブート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* の下にマウントされます。ここで、*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オーバーライドするには、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 では指定されたマウントポイントを必要な場合に自動的に作成し、関連付けられたファイルシステムを自動的にマウントします。

ZFS ファイルシステムは、/etc/vfstab ファイルの編集を必要とすることなく、ブート時に自動的にマウントされます。

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、117 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、133 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現在アンマウントされている場合は、mountpoint プロパティーを変更しても、そのファイルシステムはアンマウント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ファイルシステムは、すべて ZFS によって管理されます。次の例では、作成されたファイルシステム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 プロパティーが変更されると、ファイルシステムは古いマウントポイントから自動的にアンマウントされ、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。ZFS がアクティブであるためにファイルシステムをアンマウントできない場合は、エラーが報告され、手動での強制マウントが必要になります。

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 がブート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、この種類のファイルシステムでは使用できません。次の例では、ZFS ファイルシステム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/eric
# mount -F zfs tank/home/eschrock /mnt
```

ブート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。次の例は、/etc/vfstab ファイルのエントリがどのようなものかを示しています。

```
#device         device        mount           FS      fsck    mount   mount
#to mount       to fsck       point           type    pass    at boot options
#

tank/home/eric  -  /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 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ブートするときに、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アンマウント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。例:

```
# zfs mount | grep tank/home
zfs mount | grep tank/home
tank/home                       /tank/home
tank/home/jeff                  /tank/home/jeff
```

-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。例:

```
# zfs mount -a
```

デフォルトでは、ZFS は、空でないディレクトリの最上位へのマウントを許可しません。例:

```
# zfs mount tank/home/lori
cannot mount 'tank/home/lori': filesystem already mounted
```

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。例:

```
# zfs mount tank/home/bill
cannot mount 'tank/home/bill': legacy mountpoint
use mount(8) 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、ファイルシステムに関連付けられたプロパティー値に基づいてマウントオプションのセットが使用されます。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 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。

NFSv4 ミラーマウント機能を使用して、NFS マウント済みの ZFS ホームディレクトリをより適切に管理できます。

NFS サーバー上にファイルシステムが作成されると、NFS クライアントは新しく作成されたこれらのファイルシステムを、親ファイルシステムの既存マウント内で自動的に検出することができます。

たとえば、サーバー neo がすでに tank ファイルシステムを共有しており、クライアント zee がそれをマウントしている場合、サーバー上に /tank/baz が作成されると、それはクライアント上で自動的に認識されます。

```
zee# mount neo:/tank /mnt
zee# ls /mnt
baa    bar

neo# zfs create tank/baz

zee% ls /mnt
baa    bar    baz
zee% ls /mnt/baz
file1    file2
```

一時的なマウントプロパティーを使用する

前セクション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アンマウントされるときに元の値に戻ります。ファイルシステム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/neil 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アンマウントされているものと仮定しています。

```
# zfs mount -o ro users/home/neil
```

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 users/home/neil
NAME             PROPERTY  VALUE  SOURCE
users/home/neil  atime     off    temporary
# zfs get atime users/home/perrin
```

zfs mount コマンドの詳細は、zfs(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アンマウントする

zfs unmount サブコマンドを使用して、ZFS ファイルシステムをアンマウントできます。unmount コマンドには、マウントポイントまたはファイルシステム名を引数として指定できます。

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アンマウントしています。

```
# zfs unmount users/home/mark
```

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アンマウントしています。

```
# zfs unmount /users/home/mark
```

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アンマウントする場合は、-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アンマウントする場合は、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/eric
cannot unmount '/tank/home/eric': Device busy
# zfs unmount -f tank/home/eric
```

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アンマウントすることもできます。例:

```
# umount /tank/home/bob
```

zfs umount コマンドの詳細は、zfs(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共有および共有解除する

Oracle Solaris 11.1 リリースでは、ZFS プロパティーの継承を活用することで、ZFS 共有の管理を簡素化しています。プールバージョン 34 が動作しているプールで新しい共有構文が有効になっています。

次に示すのは、NFS および SMB のファイルシステムパッケージです。

- NFS クライアントおよびサーバーパッケージ
  - service/file-system/nfs (サーバー)
  - service/file-system/nfs (クライアント)

  その他の NFS 構成情報については、『Oracle Solaris 11.3 でのネットワークファイルシステムの管理』を参照してください。
- SMB クライアントおよびサーバーパッケージ
  - service/file-system/smb (サーバー)
  - service/file-system/smb (クライアント)

  SMB パスワード管理を含む、その他の SMB 構成情報については、『Managing SMB File Sharing and Windows Interoperability in Oracle Solaris 11.3』 の 「Managing SMB Mounts in Your Local Environment」 を参照してください。

ファイルシステムごとに複数の共有を定義できます。共有名は、各共有を一意に識別します。ファイルシステム内の特定のパスを共有するために使用されるプロパティーを定義できます。デフォルトでは、すべてのファイルシステムが共有されません。通常、共有が作成されるまで、NFS サーバーサービスは開始されません。有効な共有

が作成されると、NFS サービスは自動的に開始されます。ZFS ファイルシステムの mountpoint プロパティーが legacy に設定されている場合、レガシー share コマンドを使用することによってのみファイルシステムを共有できます。

- share.nfs プロパティーは、以前のリリースの sharenfs プロパティーを置き換えて、NFS 共有を定義および公開します。
- share.smb プロパティーは、以前のリリースの sharesmb プロパティーを置き換えて、SMB 共有を定義および公開します。
- sharenfs プロパティーと sharesmb プロパティーは、どちらもshare.nfs プロパティーと sharenfs プロパティーの別名です。
- ブート時のファイルシステムの共有に /etc/dfs/dfstab ファイルは使用されなくなりました。これらのプロパティーを設定すると、ファイルシステムが自動的に共有されます。システムのリブート時にファイルシステムが自動的に共有されるように、SMF は ZFS または UFS 共有情報を管理します。この機能は、sharenfs または sharesmb プロパティーが off に設定されていないすべてのファイルシステムがブート時に共有されることを意味します。
- sharemgr インタフェースは使用できなくなりました。レガシー share コマンドは、レガシー共有の作成に引き続き使用できます。次の例を参照してください。
- share -a コマンドは、以前の share -ap コマンドに似ており、ファイルシステムの共有は永続的です。share -p オプションは使用できなくなりました。

たとえば、tank/home ファイルシステムを共有する場合は、次のような構文を使用します。

```
# zfs set share.nfs=on tank/home
```

前の例では、tank/home ファイルシステムに対して share.nfs プロパティーが on に設定されており、share.nfs プロパティー値はすべての子孫ファイルシステムに継承されます。例:

```
# zfs create tank/home/userA
# zfs create tank/home/userB
```

追加のプロパティー値を指定したり、既存のファイルシステム共有の既存のプロパティー値を変更したりすることもできます。例:

```
# zfs set share.nfs.nosuid=on tank/home/userA
# zfs set share.nfs=on tank/home/userA
```

旧バージョンの ZFS 共有の構文

Oracle Solaris 11 の構文は引き続きサポートされているため、2 つのステップでファイルシステムを共有できます。この構文は、すべてのプールバージョンでサポートされています。

- まず、zfs set share コマンドを使用して ZFS ファイルシステムの NFS または SMB 共有を作成します。

```
# zfs create rpool/fs1
# zfs set share=name=fs1,path=/rpool/fs1,prot=nfs rpool/fs1
name=fs1,path=/rpool/fs1,prot=nfs
```

- 次に、sharenfs または sharesmb プロパティーを on に設定して共有を公開します。例:

```
# zfs set sharenfs=on rpool/fs1
# grep fs1 /etc/dfs/sharetab
/rpool/fs1      fs1     nfs     sec=sys,rw
```

ファイルシステム共有は、レガシー zfs get share コマンドを使用して表示できます。

```
# zfs get share rpool/fs1
NAME       PROPERTY  VALUE  SOURCE
rpool/fs1  share     name=fs1,path=/rpool/fs1,prot=nfs  local
```

また、ファイルシステムを共有するための share コマンドは、Oracle Solaris 10 リリースの構文と同様に、ファイルシステム内のディレクトリを共有するためにも引き続きサポートされています。たとえば、ZFS ファイルシステムを共有するには、次のように行います。

```
# share -F nfs /tank/zfsfs
# grep zfsfs /etc/dfs/sharetab
/tank/zfsfs    tank_zfsfs     nfs     sec=sys,rw
```

上の構文は UFS ファイルシステムの共有と同じです。

```
# share -F nfs /ufsfs
# grep ufsfs /etc/dfs/sharetab
/ufsfs          -               nfs     rw
/tank/zfsfs     tank_zfsfs      nfs     rw
```

新しい ZFS 共有構文

zfs set コマンドは、NFS または SMB プロトコルを介して ZFS ファイルシステムを共有および公開するために使用します。あるいは、ファイルシステムの作成時に share.nfs または share.smb プロパティーを設定することもできます。

たとえば、tank/sales ファイルシステムを作成および共有します。デフォルトの共有アクセス権は、全員に対する読み取り/書き込みです。share.nfs プロパティーは子孫のファイルシステムに継承されるため、子孫の tank/sales/logs ファイルシステムも自動的に共有され、tank/sales/log ファイルシステムは読み取り専用アクセスに設定されます。

```
# zfs create -o share.nfs=on tank/sales
# zfs create -o share.nfs.ro=\* tank/sales/logs
# zfs get -r share.nfs tank/sales
NAME             PROPERTY   VALUE  SOURCE
tank/sales       share.nfs  on     local
tank/sales%      share.nfs  on     inherited from tank/sales
```

```
tank/sales/log   share.nfs  on     inherited from tank/sales
tank/sales/log%  share.nfs  on     inherited from tank/sales
```

次のように、共有ファイルシステムの特定のシステムにルートアクセスすることができます。

```
# zfs set share.nfs=on tank/home/data
# zfs set share.nfs.sec.default.root=neo.daleks.com tank/home/data
```

プロパティーごとの継承による ZFS 共有

最新のプールバージョン 34 にアップグレードされているプールでは、ZFS プロパティーの継承を使用して共有の維持を容易にする新しい共有構文を使用できます。各共有特性は、別々の share プロパティーになります。それらの share プロパティーは、share. 接頭辞で始まる名前によって識別されます。share プロパティーの例には、share.desc、share.nfs.nosuid、および share.smb.guestok などがあります。

share.nfs プロパティーは NFS 共有が有効であるかどうかを制御します。share.smb プロパティーは SMB 共有が有効であるかどうかを制御します。新しいプールでは、sharenfs は share.nfs の別名であり、sharesmb は share.smb の別名であるため、レガシー sharenfs および sharesmb プロパティー名は引き続き使用できます。tank/home ファイルシステムを共有する場合、次のような構文を使用します。

```
# zfs set share.nfs=on tank/home
```

この例では、share.nfs プロパティー値はすべての子孫ファイルシステムに継承されます。例:

```
# zfs create tank/home/userA
# zfs create tank/home/userB
# grep tank/home /etc/dfs/sharetab
/tank/home      tank_home       nfs     sec=sys,rw
/tank/home/userA        tank_home_userA nfs     sec=sys,rw
/tank/home/userB        tank_home_userB nfs     sec=sys,rw
```

古いプールでの ZFS 共有の継承

古いプールでは、sharenfs および sharesmb プロパティーのみが子孫ファイルシステムによって継承されます。他の共有特性は、共有ごとに .zfs/shares ファイルに格納され、継承されません。

特別な規則は、sharenfs または sharesmb を親から継承する新しいファイルシステムを作成したときは必ず、sharenfs または sharesmb 値からそのファイルシステムのデフォルトの共有が作成されるというものです。sharenfs が単に on のときは、子孫ファイルシステムで作成されるデフォルトの共有にはデフォルトの NFS 特性のみが含まれることに注意してください。例:

```
# zpool get version tank
NAME  PROPERTY  VALUE  SOURCE
```

```
tank  version   33     default
# zfs create -o sharenfs=on tank/home
# zfs create tank/home/userA
# grep tank/home /etc/dfs/sharetab
/tank/home      tank_home       nfs     sec=sys,rw
/tank/home/userA        tank_home_userA nfs     sec=sys,r
```

ZFS 名前付き共有

名前付き共有を作成することもできます。これにより、SMB 環境でアクセス権およびプロパティーを設定する際の柔軟性が向上します。例:

```
# zfs share -o share.smb=on tank/workspace%myshare
```

前の例では、`zfs share` コマンドによって、`tank/workspace` ファイルシステムの `myshare` という SMB 共有が作成されます。このファイルシステムの `.zfs/shares` ディレクトリを介して SMB 共有にアクセスしたり、特定のアクセス権や ACL を表示または設定したりできます。各 SMB 共有は、個別の `.zfs/shares` ファイルで表されます。例:

```
# ls -lv /tank/workspace/.zfs/shares
-rwxrwxrwx+  1 root     root           0 May 15 10:31 myshare
0:everyone@:read_data/write_data/append_data/read_xattr/write_xattr
/execute/delete_child/read_attributes/write_attributes/delete
/read_acl/write_acl/write_owner/synchronize:allow
```

名前付き共有は親ファイルシステムから共有プロパティーを継承します。前の例で `share.smb.guestok` プロパティーを親ファイルシステムに追加した場合、このプロパティーは名前付き共有に継承されます。例:

```
# zfs get -r share.smb.guestok tank/workspace
NAME                    PROPERTY           VALUE  SOURCE
tank/workspace          share.smb.guestok  on     inherited from tank
tank/workspace%myshare  share.smb.guestok  on     inherited from tank
```

名前付き共有は、NFS 環境でファイルシステムのサブディレクトリに対して共有を定義するときに役立つことがあります。例:

```
# zfs create -o share.nfs=on -o share.nfs.anon=99 -o share.auto=off tank/home
# mkdir /tank/home/userA
# mkdir /tank/home/userB
# zfs share -o share.path=/tank/home/userA tank/home%userA
# zfs share -o share.path=/tank/home/userB tank/home%userB
# grep tank/home /etc/dfs/sharetab
/tank/home/userA        userA   nfs     anon=99,sec=sys,rw
/tank/home/userB        userB   nfs     anon=99,sec=sys,rw
```

上記の例は、ファイルシステムの `share.auto` を off に設定すると、そのファイルシステムの自動共有のみが off になり、他のすべてのプロパティー継承は変更されないことも示しています。他のほとんどの共有プロパティーと異なり、`share.auto` プロパティーは継承可能ではありません。

名前付き共有は、公開 NFS 共有の作成時にも使用します。公開共有は、名前付きの NFS 共有でのみ作成できます。例:

```
# zfs create -o mountpoint=/pub tank/public
# zfs share -o share.nfs=on -o share.nfs.public=on tank/public%pubshare
# grep pub /etc/dfs/sharetab
/pub    pubshare        nfs     public,sec=sys,rw
```

NFS および SMB 共有プロパティーの詳細は、share_nfs(1M) および share_smb(1M) のマニュアルページを参照してください。

ZFS 自動共有

自動共有が作成されると、ファイルシステム名から一意のリソース名が構築されます。構築される名前はファイルシステム名のコピーですが、リソース名では使用できない文字がファイルシステム名に含まれている場合、それらは下線 (_) で置き換えられます。たとえば、data/home/john のリソース名は data_home_john になります。

share.autoname プロパティー名を設定すると、自動共有の作成時にファイルシステム名を特定の名前で置き換えることができます。この特定の名前は、継承の際に先頭部分のファイルシステム名を置き換えるためにも使用されます。例:

```
# zfs create -o share.smb=on -o share.autoname=john data/home/john
# zfs create data/home/john/backups
# grep john /etc/dfs/sharetab
/data/home/john john    smb
/data/home/john/backups john_backups    smb
```

まだ共有されていないファイルシステムでレガシー share コマンドまたは zfs set share コマンドを使用すると、その share.auto 値は自動的に off に設定されます。レガシーコマンドは常に名前付き共有を作成します。この特別な規則によって、自動共有が作成中の名前付き共有を妨害するのを防ぐことができます。

ZFS 共有情報を表示する

ファイル共有プロパティーの値を表示するには、zfs get コマンドを使用します。次の例は、単一ファイルシステムの share.nfs プロパティーを表示する方法を示しています。

```
# zfs get share.nfs tank/sales
NAME        PROPERTY   VALUE  SOURCE
tank/sales  share.nfs  on     local
```

次の例は、子孫ファイルシステムの share.nfs プロパティーを表示する方法を示しています。

```
# zfs get -r share.nfs tank/sales
NAME             PROPERTY   VALUE  SOURCE
tank/sales       share.nfs  on     local
tank/sales%      share.nfs  on     inherited from tank/sales
tank/sales/log   share.nfs  on     inherited from tank/sales
tank/sales/log%  share.nfs  on     inherited from tank/sales
```

zfs get all コマンド構文では、共有プロパティーの拡張情報は利用できません。

NFS または SMB 共有に関する詳細を表示するには、次の構文を使用します。

```
# zfs get share.nfs.all tank/sales
NAME        PROPERTY             VALUE  SOURCE
tank/sales  share.nfs.aclok      off    default
tank/sales  share.nfs.anon              default
tank/sales  share.nfs.charset.*  ...    default
tank/sales  share.nfs.cksum             default
tank/sales  share.nfs.index             default
tank/sales  share.nfs.log               default
tank/sales  share.nfs.noaclfab   off    default
tank/sales  share.nfs.nosub      off    default
tank/sales  share.nfs.nosuid     off    default
tank/sales  share.nfs.public     -      -
tank/sales  share.nfs.sec               default
tank/sales  share.nfs.sec.*      ...    default
```

共有プロパティーの数が多いため、デフォルト以外の値でプロパティーを表示することを検討してください。例:

```
# zfs get -e -s local,received,inherited share.all tank/home
NAME       PROPERTY           VALUE  SOURCE
tank/home  share.auto         off    local
tank/home  share.nfs          on     local
tank/home  share.nfs.anon     99     local
tank/home  share.protocols    nfs    local
tank/home  share.smb.guestok  on     inherited from tank
```

ZFS 共有プロパティー値を変更する

共有プロパティー値を変更するには、ファイルシステム共有で新規または変更されたプロパティーを指定します。たとえば、ファイルシステムの作成時に読み取り専用プロパティーを設定した場合、そのプロパティーを off に設定できます。

```
# zfs create -o share.nfs.ro=\* tank/data
# zfs get share.nfs.ro tank/data
NAME       PROPERTY              VALUE  SOURCE
tank/data  share.nfs.sec.sys.ro  *     local
# zfs set share.nfs.ro=none tank/data
# zfs get share.nfs.ro tank/data
NAME       PROPERTY              VALUE  SOURCE
tank/data  share.nfs.sec.sys.ro  off    local
```

SMB 共有を作成した場合は、NFS 共有プロトコルを追加することもできます。例:

```
# zfs set share.smb=on tank/multifs
# zfs set share.nfs=on tank/multifs
# grep multifs /etc/dfs/sharetab
/tank/multifs   tank_multifs    nfs     sec=sys,rw
/tank/multifs   tank_multifs    smb     -
```

SMB プロトコルを削除します。

```
# zfs set share.smb=off tank/multifs
# grep multifs /etc/dfs/sharetab
/tank/multifs   tank_multifs    nfs     sec=sys,rw
```

名前付き共有の名前を変更できます。例:

```
# zfs share -o share.smb=on tank/home/abc%abcshare
# grep abc /etc/dfs/sharetab
/tank/home/abc  abcshare        smb     -
# zfs rename tank/home/abc%abcshare tank/home/abc%a1share
# grep abc /etc/dfs/sharetab
/tank/home/abc  a1share        smb     -
```

ZFS 共有の公開と非公開

名前付き共有を破棄しないで一時的に共有解除するには、`zfs unshare` コマンドを使用します。例:

```
# zfs unshare tank/home/abc%a1share
# grep abc /etc/dfs/sharetab
#
# zfs share tank/home/abc%a1share
# grep abc /etc/dfs/sharetab
/tank/home/abc  a1share smb     -
```

`zfs unshare` コマンドを発行すると、すべてのファイルシステム共有が共有解除されます。これらの共有は、そのファイルシステムに対して `zfs share` コマンドを発行するか、そのファイルシステムに対して `share.nfs` または `share.smb` プロパティーを設定するまで共有解除のままです。

`zfs unshare` コマンドを発行しても定義された共有は削除されず、次回そのファイルシステムに対して `zfs share` コマンドを発行するか、そのファイルシステムに対して `share.nfs` または `share.smb` プロパティーを設定したときに再度共有されます。

ZFS 共有を削除する

ファイルシステム共有を共有解除するには、`share.nfs` または `share.smb` プロパティーを off に設定します。例:

```
# zfs set share.nfs=off tank/multifs
# grep multifs /etc/dfs/sharetab
#
```

名前付き共有を完全に削除するには、`zfs destroy` コマンドを使用します。例:

```
# zfs destroy tank/home/abc%a1share
```

非大域ゾーン内の ZFS ファイル共有

Oracle Solaris 11 以降、非大域ゾーン内で NFS 共有を作成して公開できます。

- ZFS ファイルシステムは、非大域ゾーンでマウントされて使用できるようになれば、そのゾーン内で共有できます。
- ファイルシステムは、非大域ゾーンに委任されておらず、かつ非大域ゾーンにマウントされていない場合に、大域ゾーン内で共有できます。ファイルシステムを非大

域ゾーンに追加した場合、レガシー share コマンドを使用することによってのみそれを共有できます。

たとえば、/export/home/data および /export/home/data1 ファイルシステムは、zfszone で使用できます。

```
zfszone# share -F nfs /export/home/data
zfszone#  cat /etc/dfs/sharetab

zfszone# zfs set share.nfs=on tank/zones/export/home/data1
zfszone# cat /etc/dfs/sharetab
```

ZFS 共有のマイグレーション/移行に関する問題

次の移行の問題を確認してください。

- **古い共有プロパティーを使ったファイルシステムのインポート** - プールをインポートする場合、または Oracle Solaris 11 より前に作成されたファイルシステムストリームを受け取る場合、sharenfs および sharesmb プロパティーのプロパティー値にすべての共有プロパティーが直接含まれています。ほとんどの場合、これらのレガシー共有プロパティーは、各ファイルシステムが共有されるとすぐに、同等の名前付き共有セットに変換されます。ほとんどの場合、インポート操作によってマウントおよび共有がトリガーされるため、名前付き共有への変換はインポートプロセス中に直接行われます。
- **Oracle Solaris 11 からのアップグレード** - 名前付き共有は新しい形式に変換されるため、プールをバージョン 34 にアップグレードしたあとの最初のファイルシステムの共有には長い時間がかかることがあります。アップグレードプロセスによって作成された名前付き共有は正しいものですが、共有プロパティー継承を利用できない可能性があります。
  - 共有プロパティー値を表示します。

    ```
    # zfs get share.nfs filesystem
    # zfs get share.smb filesystem
    ```

  - 古い BE でブートする場合は、sharenfs および sharesmb プロパティーを元の値にリセットします。
- **Oracle Solaris からのアップグレード 11** - Oracle Solaris 11 および 11.1 では、sharenfs および sharesmb プロパティーには off と on の値しか設定できません。これらのプロパティーは、共有特性の定義に使用できなくなりました。

  ブート時のファイルシステムの共有に /etc/dfs/dfstab ファイルは使用されなくなりました。ブート時に、有効なファイルシステム共有を含むマウント済みの ZFS ファイルシステムがすべて自動的に共有されます。共有は、sharenfs または sharesmb が on に設定されると有効になります。

  sharemgr インタフェースは使用できなくなりました。レガシー share コマンドは、レガシー共有の作成に引き続き使用できます。share -a コマンドは、以前の

share -ap コマンドに似ており、ファイルシステムの共有は永続的です。share -p オプションは使用できなくなりました。

- **システムのアップグレード** - このリリースでのプロパティーの変更により、Oracle Solaris 11 BE でブートすると、ZFS 共有が不正になります。ZFS 以外の共有は影響を受けません。古い BE でブートすることを計画している場合は、ZFS 共有構成を復元できるように、pkg update 操作の前にまず、既存の共有構成のコピーを保存します。

  古い BE で、sharemgr show -vp コマンドを使用して、すべての共有およびそれらの構成を一覧表示します。

  共有プロパティー値を表示するには、次のコマンドを使用します。

  ```
  # zfs get sharenfs filesystem
  # zfs get sharesmb filesystem
  ```

  古い BE に戻す場合は、sharenfs および sharesmb プロパティーと、sharemgr で定義されたすべての共有を元の値にリセットします。
- **旧バージョンの共有解除動作** - unshare -a コマンドまたは unshareall コマンドを使用すると、ファイルシステムの共有が解除されますが、SMF 共有リポジトリは更新されません。既存の共有を再度共有しようとすると、共有リポジトリで競合がチェックされ、エラーが表示されます。

ZFS ファイルシステムの共有の問題のトラブルシューティング

次の共有エラー状況を確認してください。

- **新しい共有または以前の共有が共有されない**
  - **プールおよびファイルシステムバージョンが最新であることを確認する** - share.nfs または share.smb プロパティーを設定することによって新しい共有が共有されない場合は、プールバージョンが 34、ファイルシステムバージョンが 6 であることを確認してください。
  - **共有は NFS サービスが起動する前に存在する必要がある**- NFS サーバーサービスは、ファイルシステムが共有されるまで実行されません。最初に NFS 共有を作成し、共有へのリモートアクセスを試してください。
  - **既存の共有を含むシステムはアップグレードされたが、共有が利用できない** - 既存の共有を含むシステムはアップグレードされるが、共有を再共有しようとすると失敗します。share.auto プロパティーが無効になっているため、共有は共有されていない可能性があります。share.auto が off に設定されている場合は、名前付き共有のみが利用可能であり、以前の共有構文との互換性が,適用されます。既存の共有は次のようになります。

    ```
    # zfs get share
    NAME                          PROPERTY  VALUE  SOURCE
    ```

```
tank/data                 share     name=data,path=/tank/data,prot=nfs  local
```

1. share.auto プロパティーが有効になっていることを確認します。有効でない場合は、有効にします。

   ```
   # zfs get -r share.auto tank/data
   # zfs set share.auto=on tank/data
   ```

2. ファイルシステムを再共有します。

   ```
   # zfs set -r share.nfs=on tank/data
   ```

3. 前述のコマンドを正常に実行するには、その前に名前付き共有を削除して再作成する必要がある場合もあります。

   ```
   # zfs list -t share -Ho name -r tank/data | xargs -n1 zfs destroy
   ```

4. 必要に応じて、名前付き共有を再作成します。

   ```
   # zfs create -o share.nfs=on tank/data%share
   ```

- **名前付き共有を含む共有プロパティーがスナップショットに含まれていない** - 共有プロパティーおよび .zfs/shares ファイルの扱いが、zfs clone 操作と zfs send 操作では異なります。.zfs/shares ファイルはスナップショットに含まれており、zfs clone および zfs send 操作で保存されます。zfs send 操作と zfs receive 操作中のプロパティーの動作については、184 ページの「ZFS スナップショットストリームに異なるプロパティー値を適用する」を参照してください。クローン操作後、プロパティーが ZFS ファイルシステム階層内のクローンの新しい位置から継承されるのに対し、ファイルはすべてクローン前のスナップショットからのものです。
- **名前付き共有要求が失敗する** - 名前付き共有を作成するための要求が共有が自動共有と競合するために失敗したときは、auto.share プロパティーを無効にする必要がある場合があります。
- **共有を含むプールが以前にエクスポートされた** - プールが読み取り専用でインポートされると、そのプロパティーもそのファイルも変更できないため、新しい共有の作成は失敗します。プールがエクスポートされる前に共有が存在していた場合、可能であれば既存の共有特性が使用されます。

次の表は、既知の共有状態と、必要に応じてそれらの解決方法を示しています。

共有状態	説明	解決方法
INVALID	共有が内部で一貫性がないか、別の共有と競合しているために無効になっています。	次のコマンドを使用して、無効な共有を再度共有してみます。 `# zfs share FS%share` このコマンドを使用すると、共有のどの側面が検証に失敗しているのかに関するエラーメッセージが表示されます。これを訂正してから、共有を再試行してください。
SHARED	共有は共有されています。	何も必要ありません。

共有状態	説明	解決方法
UNSHARED	共有は有効ですが、共有が解除されています。	zfs share コマンドを使用して、個々の共有または親ファイルシステムのどちらかを再度共有します。
UNVALIDATED	共有はまだ検証されていません。その共有を含むファイルシステムが共有可能な状態でない可能性があります。たとえば、マウントされていなかったり、現在のゾーン以外のゾーンに委任されていたりします。あるいは、目的の共有を表す ZFS プロパティーが作成されてはいますが、まだ有効な共有として検証されていません。	zfs share コマンドを使用して、個々の共有または親ファイルシステムを再度共有します。ファイルシステムそのものが共有可能である場合、再共有の試みは共有に成功 (状態が shared に遷移) するか、共有に失敗 (状態が invalid に遷移) するかのどちらかになります。あるいは、share -A コマンドを使用して、マウント済みのすべてのファイルシステム内のすべての共有を一覧表示することもできます。これにより、マウント済みのファイルシステム内のすべての共有が unshared (有効だがまだ共有されていない) と invalid のどちらかとして解決されます。

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。どちらのプロパティーも、設定対象のファイルシステムとそのファイルシステム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 ファイルシステムに設定した場合は、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。ファイルシステム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 Oracle 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、ファイルシステム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、ファイルシステム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。
- 多くの異なるユーザーのための大規模なファイルシステム上に割り当て制限を設定するための便利な方法として、デフォルトのユーザーまたはグループ割り当て制限の設定があります。

割り当て制限と予約の設定方法の詳細については、148 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152 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、tank/home/jeff で 10G バイトの割り当て制限が設定されます。

```
# zfs set quota=10G tank/home/jeff
# zfs get quota tank/home/jeff
NAME            PROPERTY  VALUE  SOURCE
tank/home/jeff  quota     10G    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。例:

```
# zfs list -r tank/home
NAME                USED  AVAIL  REFER  MOUNTPOINT
tank/home          1.45M  66.9G    36K  /tank/home
tank/home/eric      547K  66.9G   547K  /tank/home/eric
tank/home/jeff      322K  10.0G   291K  /tank/home/jeff
tank/home/jeff/ws    31K  10.0G    31K  /tank/home/jeff/ws
tank/home/lori      547K  66.9G   547K  /tank/home/lori
tank/home/mark       31K  66.9G    31K  /tank/home/mark
# df -h /tank/home/jeff
Filesystem             Size  Used Avail Use% Mounted on
tank/home/jeff          10G  306K   10G   1% /tank/home/jeff
```

tank/home は 66.9G バイトのディスク容量を使用できますが、tank/home/jeff と tank/home/jeff/ws は、tank/home/jeff の割り当て制限のため、10G バイトのディスク容量しか使用できません。

ファイルシステムに refquota を設定して、ファイルシステムが消費できるディスク容量を制限できます。この制限には、子孫によって消費されたディスク容量は含まれません。たとえば、studentA の 10G バイトの割り当て制限は、スナップショットによって消費される容量によって影響されません。

```
# zfs set refquota=10g students/studentA
# zfs list -t all -r students
NAME                          USED  AVAIL  REFER  MOUNTPOINT
students                      150M  66.8G    32K  /students
students/studentA             150M  9.85G   150M  /students/studentA
students/studentA@yesterday      0      -   150M  -
# zfs snapshot students/studentA@today
# zfs list -t all -r students
NAME                          USED  AVAIL  REFER  MOUNTPOINT
students                      150M  66.8G    32K  /students
students/studentA             150M  9.90G   100M  /students/studentA
students/studentA@yesterday  50.0M      -   150M  -
students/studentA@today          0      -   100M  -
```

さらに利便性を高めるために、ファイルシステム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。例:

```
# zfs set quota=20g students/studentA
# zfs list -t all -r students
NAME                          USED  AVAIL  REFER  MOUNTPOINT
students                      150M  66.8G    32K  /students
students/studentA             150M  9.90G   100M  /students/studentA
students/studentA@yesterday  50.0M      -   150M  -
students/studentA@today          0      -   100M  -
```

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、studentA は、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。例:

```
# zfs get refquota,quota students/studentA
NAME               PROPERTY  VALUE              SOURCE
students/studentA  refquota  10G                local
students/studentA  quota     20G                local
```

ファイルシステムの割り当て制限を強制的に適用すると、数秒間の遅延が発生する可能性あります。この遅延が発生する場合は、ファイルシステムが割り当て制限を超えていることをシステムが通知し、EDQUOT エラーメッセージで追加の書き込みを拒否する前に、ユーザーがファイルシステム割り当て制限を超えている可能性があります。

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。例:

```
# zfs create students/compsci
# zfs set userquota@student1=10G students/compsci
# zfs create students/labstaff
# zfs set groupquota@labstaff=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@labstaff students/labstaff
NAME               PROPERTY             VALUE                SOURCE
students/labstaff  groupquota@labstaff  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      350M   none
POSIX User  student1  426M    10G
# zfs groupspace students/labstaff
TYPE         NAME      USED  QUOTA
POSIX Group  labstaff  250M    20G
POSIX Group  root      350M   none
```

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  550M               local
# zfs get groupused@labstaff students/labstaff
NAME               PROPERTY            VALUE                SOURCE
students/labstaff  groupused@labstaff  250                  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@student1=none students/compsci
# zfs set groupquota@labstaff=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、ユーザーが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ユーザー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。例:

```
# zfs set userquota@student1=10m students/compsci
# zfs userspace students/compsci
TYPE        NAME      USED  QUOTA
POSIX User  root      350M   none
POSIX User  student1  550M    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。例:

```
# zfs set userquota@student1=10GB students/compsci
# zfs userspace students/compsci
TYPE        NAME      USED  QUOTA
POSIX User  root      350M   none
POSIX User  student1  550M    10G
# quota student1
# quota -v student1
Disk quotas for student1 (uid 102):
Filesystem     usage  quota  limit    timeleft  files  quota  limit    timeleft
/students/compsci
563287 10485760 10485760          -      -      -      -          -
```

デフォルトのユーザーおよびグループの割り当て制限を設定する

Oracle Solaris 11.3 リリース以降では、特定の割り当て制限が定義されていないすべてのユーザーに自動的に適用されるデフォルトのユーザー割り当て制限またはデフォルトのグループ割り当て制限を設定できます。特定のユーザーおよびグループ割り当て制限と同様に、デフォルトのユーザーおよびグループ割り当て制限も子孫ファイルシステムには継承できません。さらに、デフォルトのユーザーまたはグループ割り当て制限が最上位ファイルシステム上に設定されている場合、子孫ファイルシステムで消費された容量は最上位ファイルシステムのデフォルトの割り当て制限に計上されません。

大規模な共有ファイルシステムでは、デフォルトのユーザー割り当て制限を設定できます。例:

```
# zfs set defaultuserquota=30gb students/labstaff/admindata
```

大規模な共有ファイルシステムでデフォルトのユーザー割り当て制限を使用すると、個々のユーザー割り当て制限を指定することなく、容量の増大を制限できます。また、だれが最上位ファイルシステムを使用しているかをモニターすることもできます。

```
# zfs userspace students/labstaff/admindata
TYPE        NAME     USED  QUOTA  SOURCE
POSIX User  admin1  2.00G    30G  default
POSIX User  admin2  4.00G    30G  default
POSIX User  root       3K    30G  default
```

上の例では、既存の割り当て制限が設定されていない各ユーザーに `students/labstaff/admindata` 内の 30G バイトのディスク容量が許可されます。この動作を `students/labstaff/admindata` に 30G バイトのファイルシステム割り当て制限を設定した場合と比較すると、30G バイトの累積の割り当て制限が、既存の割り当て制限が設定されていなかったすべてのユーザーに適用されます。

デフォルトのグループ割り当て制限も同様に設定できます。たとえば、次の構文は、`students/math` ファイルシステム上に 120G バイトの割り当て制限を設定します。`zfs groupspace` コマンドを使用すると、デフォルトのグループ割り当て制限で最上位ファイルシステムの使用状況を追跡できます。

```
# zfs set defaultgroupquota=120g students/math
# zfs groupspace students/math
TYPE         NAME       USED  QUOTA  SOURCE
POSIX Group  root         6K   120G  default
POSIX Group  students  40.0G   120G  default
```

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。例:

```
# zfs set reservation=5G tank/home/bill
# zfs get reservation tank/home/bill
NAME            PROPERTY     VALUE   SOURCE
tank/home/bill  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。例:

```
# zfs list -r tank/home
NAME                USED  AVAIL  REFER  MOUNTPOINT
tank/home          5.00G  61.9G    37K  /tank/home
```

```
tank/home/bill       31K  66.9G    31K  /tank/home/bill
tank/home/jeff      337K  10.0G   306K  /tank/home/jeff
tank/home/lori      547K  61.9G   547K  /tank/home/lori
tank/home/mark       31K  61.9G    31K  /tank/home/mark
```

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。使用される容量には、tank/home/bill 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、親ファイルシステム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。例:

```
# zfs set reservation=10G tank/home/bill
# zfs set reservation=5G tank/home/bill
# zfs get reservation tank/home/bill
NAME            PROPERTY     VALUE   SOURCE
tank/home/bill  reservation  5G      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。例:

```
# 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。例:

```
# 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。例:

```
# 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。

ZFS ファイルシステムの圧縮

圧縮は、データがより少ないディスク容量を使用して格納されるプロセスです。次の圧縮アルゴリズムを使用できます。

- `gzip` - 標準の UNIX 圧縮。
- `gzip-`*N* - 特定の gzip レベルを選択します。`gzip-1` は、もっとも高速な gzip 圧縮を提供します。`gzip-9` は、最適なデータ圧縮を提供します。`gzip-6` がデフォルトです。
- `lz4` - より優れた圧縮をより低い CPU オーバーヘッドで提供します
- `lzjb` - 優れた圧縮と同時に、最適なパフォーマンスを提供します
- `zle` - 長さ 0 のエンコーディングが 0 の大きなブロックを含むデータセットに有効です

注記 - 現時点では、`lz4` および `gzip` のどちらの圧縮アルゴリズムもルートプールでサポートされていません。

圧縮の ZFS プロパティーを設定することによって、特定の圧縮アルゴリズムを選択できます。LZ4 アルゴリズムを使用するには、次のようなコマンドを使用します。

```
# zfs set compress=lz4 pool/fs
```

ZFS ファイルシステムの暗号化

暗号化とは機密保護のためにデータをエンコードするプロセスで、エンコードされたデータにデータ所有者がアクセスするには鍵が必要になります。ZFS 暗号化を使用する利点は次のとおりです。

- ZFS 暗号化は ZFS コマンドセットと統合されています。ほかの ZFS 操作と同様に、鍵の変更や再入力などの暗号化操作は、オンラインで実行されます。

- 既存のストレージプールがアップグレードされていれば、それを使用できます。特定のファイルシステムの暗号化には柔軟性があります。
- データは、CCM および GCM 操作モードで、鍵の長さが 128、192、および 256 の AES (Advanced Encryption Standard) を使用して暗号化されます。
- ZFS 暗号化は、Oracle Solaris 暗号化フレームワークを使用します。このため自動的に、暗号化アルゴリズムのすべての使用可能なハードウェアアクセラレーションまたは最適化されたソフトウェア実装にアクセスできます。
- 現時点では、単独のファイルシステムであっても、ZFS ルートファイルシステムまたはその他の OS コンポーネント (`/var` ディレクトリなど) を暗号化することはできません。
- ZFS 暗号化は子孫のファイルシステムに継承できます。
- 一般ユーザーは、`create`、`mount`、`keysource`、`checksum`、`encryption` アクセス権が割り当てられている場合に、暗号化されたファイルを作成して鍵の操作を管理できます。

ZFS ファイルシステムが作成されるときに暗号化ポリシーを設定できますが、そのポリシーを変更することはできません。たとえば、`tank/home/darren` ファイルシステムは、暗号化プロパティーを有効にして作成されています。デフォルトの暗号化ポリシーでは、最低 8 文字の長さが必要なパスフレーズの入力が求められます。

```
# zfs create -o encryption=on tank/home/darren
Enter passphrase for 'tank/home/darren': xxxxxxx
Enter again: xxxxxxxx
```

ファイルシステムの暗号化が有効になっていることを確認します。例:

```
# zfs get encryption tank/home/darren
NAME              PROPERTY    VALUE        SOURCE
tank/home/darren  encryption  on           local
```

ファイルシステムの暗号化の値が `on` になっている場合、デフォルトの暗号化アルゴリズムは `aes-128-ccm` です。

ラッピング鍵は、実際のデータ暗号化鍵を暗号化するために使用されます。ラッピング鍵は、上記の例のように、暗号化したファイルシステムの作成時に、`zfs` コマンドからカーネルに渡されます。ラッピング鍵は、ファイルにあるか (生または 16 進数形式)、パスフレーズから派生します。

ラッピング鍵の形式と場所は、`keysource` プロパティーで次のように指定されます。

```
keysource=format,location
```

- 形式は次のいずれかになります。
  - `raw` – 生の鍵バイト
  - `hex` – 16 進数の鍵文字列
  - `passphrase` – 鍵を生成する文字列
- 場所は次のいずれかになります。

- `prompt` – ファイルシステムの作成またはマウント時に鍵またはパスフレーズの入力が要求される
- `file:///`*filename* – ファイルシステム内の鍵またはパスフレーズファイルの場所
- `pkcs11` – PKCS#11 トークンでの鍵またはパスフレーズの場所を記述した URI
- `https://`*location* – セキュアなサーバー上の鍵またはパスフレーズファイルの場所。この方法を使用して鍵情報を平文で転送することは推奨されていません。URL に `GET` を付けると、`keysource` プロパティーの形式部分でリクエストされた内容に従って、鍵の値またはパスフレーズのみが返されます。

  `keysource` で `https://` ロケータを使用する場合は、ZFS サーバーが提示する証明書が `libcurl` および OpenSSL で信頼されているものである必要があります。独自のトラストアンカーまたは自己署名付き証明書を、`/etc/openssl/certs` にある証明書ストアに追加します。PEM 形式の証明書を `/etc/certs/CA` ディレクトリに配置し、次のコマンドを実行します。

  ```
  # svcadm refresh ca-certificates
  ```

`keysource` 形式が *passphrase* の場合、ラッピング鍵はパスフレーズから派生します。それ以外の場合、`keysource` プロパティー値は、生のバイトまたは 16 進数形式で、実際のラッピング鍵を示します。パスフレーズがファイルに格納されているか、入力が要求される生のバイトストリームに格納されているか (これはおそらくスクリプト処理にのみ適しています) を指定できます。

ファイルシステムの `keysource` プロパティー値が `passphrase` を指定している場合、ラッピング鍵は、`PKCS#5 PBKD2` と、ファイルシステムごとにランダムに生成されたソルトを使用して、パスフレーズから派生します。したがって、子孫のファイルシステムで使用された場合、同じパスフレーズでも異なるラッピング鍵が生成されます。

ファイルシステムの暗号化ポリシーは、子孫のファイルシステムによって継承され、削除することはできません。例:

```
# zfs snapshot tank/home/darren@now
# zfs clone tank/home/darren@now tank/home/darren-new
Enter passphrase for 'tank/home/darren-new': xxxxxxx
Enter again: xxxxxxxx
# zfs set encryption=off tank/home/darren-new
cannot set property for 'tank/home/darren-new': 'encryption' is readonly
```

暗号化された、または暗号化されていない ZFS ファイルシステムをコピーまたは移行する必要がある場合、次の点を考慮してください。

- 現在、暗号化されていないデータセットストリームを送信することはできず、受信側のプールのデータセットで暗号化を有効にしている場合でも、これを暗号化されたストリームとして受信することもできません。
- 次のコマンドを使用して、暗号化されていないデータを、暗号化を有効にしているプール/ファイルシステムに移行できます。
  - `cp -r`

  - `find | cpio`
  - `tar`
  - `rsync`
- 複製された暗号化ファイルシステムストリームは、暗号化されたファイルシステムで受信でき、データは暗号化されたままです。詳細については、例36「暗号化された ZFS ファイルシステムを送受信する」を参照してください。

暗号化された ZFS ファイルシステムの鍵を変更する

`zfs key -c` コマンドを使用して、暗号化されたファイルシステムのラッピング鍵を変更できます。ブート時にファイルシステムの鍵 (`zfs key -l`) を明示的に読み込むか、ファイルシステム (`zfs mount` *filesystem*) をマウントすることによって、既存のラッピング鍵を最初に読み込んでいる必要があります。例:

```
# zfs key -c tank/home/darren
Enter new passphrase for 'tank/home/darren': xxxxxxxx
Enter again: xxxxxxxx
```

次の例では、ラッピング鍵が変更され、ラッピング鍵がファイルから取得されることを指定するように `keysource` プロパティー値が変更されます。

```
# zfs key -c -o keysource=raw,file:///media/stick/key tank/home/darren
```

暗号化されたファイルシステムのデータ暗号化鍵は、`zfs key -K` コマンドを使用して変更できますが、新しい暗号化鍵は新しく書き込むデータにのみ使用されます。この機能は、データ暗号化鍵の制限時間に関する NIST 800-57 ガイドラインを遵守するために使用できます。例:

```
# zfs key -K tank/home/darren
```

上記の例では、データ暗号化鍵は表示されず、ユーザーが直接管理することもできません。さらに、鍵変更操作を実行するには `keychange` の委任が必要です。

次の暗号化アルゴリズムを使用できます。

- `aes-128-ccm`, `aes-192-ccm`, `aes-256-ccm`
- `aes-128-gcm`, `aes-192-gcm`, `aes-256-gcm`

ZFS `keysource` プロパティーは、ファイルシステムのデータ暗号化鍵をラップする鍵の形式と場所を指定します。例:

```
# zfs get keysource tank/home/darren
NAME              PROPERTY   VALUE              SOURCE
tank/home/darren  keysource  passphrase,prompt  local
```

ZFS `rekeydate` プロパティーは、前回の `zfs key -K` 操作の日付を特定します。例:

```
# zfs get rekeydate tank/home/darren
NAME              PROPERTY   VALUE                  SOURCE
tank/home/darren  rekeydate  Wed Jul 25 16:54 2012  local
```

暗号化したファイルシステムの `creation` プロパティーと `rekeydate` プロパティーの値が同じであれば、`zfs key -K` 操作でファイルシステムの鍵が再入力されていることは決してありません。

ZFS 暗号化鍵の管理

ZFS 暗号化鍵は、集中管理の場所が必要な場合、ローカルシステムとリモートのどちらかで、ユーザーのニーズに応じて、さまざまな方法で管理できます。

- **ローカル** – 上記の例は、ラッピング鍵がパスフレーズプロンプトまたはローカルシステム上のファイルに格納されている未処理の鍵のどちらにもなり得ることを示しています。
- **リモート** – 鍵情報をリモートで格納するには、Oracle Key Manager のような集中管理された鍵管理システムを使用するか、http または https URI での単純な GET リクエストをサポートする Web サービスを使用します。PKCS#11 トークンを使用すれば、Oracle Solaris システムで Oracle Key Manager の鍵情報にアクセスできます。

ZFS 暗号化鍵の管理の詳細は、次を参照してください `http://www.oracle.com/technetwork/articles/servers-storage-admin/manage-zfs-encryption-1715034.html`

Oracle Key Manager を使用した鍵情報の管理については、次を参照してください。

`http://docs.oracle.com/cd/E24472_02/`

ZFS 鍵操作アクセス権を委任する

鍵操作を委任するための次のアクセス権に関する説明を確認してください。

- `zfs key -l` コマンドと `zfs key -u` コマンドを使用してファイルシステム鍵をロードまたはアンロードする場合、`key` アクセス権が必要になります。多くの場合、`mount` アクセス権も必要になります。
- `zfs key -c` コマンドと `zfs key -K` コマンドを使用してファイルシステム鍵を変更する場合、`keychange` アクセス権が必要になります。

鍵の使用 (ロードまたはアンロード) と鍵の変更に、別々のアクセス権を委任することを検討してください。これにより 2 人鍵操作モデルを使用できます。たとえば、どのユーザーが鍵を使用でき、どのユーザーが鍵を変更できるかを決めます。または、鍵の変更には両方のユーザーが同席する必要があります。このモデルを使用すると、キーエスクローシステムを構築することもできます。

暗号化した ZFS ファイルシステムをマウントする

暗号化した ZFS ファイルシステムをマウントしようとする場合には、次の考慮事項を確認してください。

- 暗号化したファイルシステム鍵がブート中に使用できない場合、ファイルシステムは自動的にはマウントされません。たとえば、暗号化ポリシーが `passphrase,prompt` に設定されているファイルシステムは、ブートプロセスを中断してパスフレーズの入力を要求できないため、ブート中にマウントされません。
- 暗号化ポリシーが `passphrase,prompt` に設定されているファイルシステムをブート時にマウントする場合は、`zfs mount` コマンドで明示的にマウントしてパスフレーズを指定するか、`zfs key -l` コマンドを使用して、システムのブート後に鍵の入力を要求する必要があります。

  例:

  ```
  # zfs mount -a
  Enter passphrase for 'tank/home/darren': xxxxxxxx
  Enter passphrase for 'tank/home/ws': xxxxxxxx
  Enter passphrase for 'tank/home/mark': xxxxxxxx
  ```

- 暗号化したファイルシステムの `keysource` プロパティーが別のファイルシステム内のファイルを指している場合、特にファイルがリムーバブルメディアにある場合には、ファイルシステムのマウント順序が、暗号化されたファイルシステムがブート時にマウントされるかどうかに影響を与える可能性があります。

暗号化された ZFS ファイルシステムをアップグレードする

Oracle Solaris 11 システムを 11.1 にアップグレードする前に、暗号化されたファイルシステムがマウントされていることを確認してください。暗号化されたファイルシステムをマウントして、入力を要求されたらパスフレーズを指定します。

```
# zfs mount -a
Enter passphrase for 'pond/amy': xxxxxxxx
Enter passphrase for 'pond/rory': xxxxxxxx
# zfs mount | grep pond
pond                            /pond
pond/amy                        /pond/amy
pond/rory                       /pond/rory
```

その後、暗号化されたファイルシステムをアップグレードします。

```
# zfs upgrade -a
```

アンマウントされている、暗号化された ZFS ファイルシステムをアップグレードしようとすると、次のようなメッセージが表示されます。

```
# zfs upgrade -a
```

```
cannot set property for 'pond/amy': key not present
```

また、zpool status 出力には破壊されたデータが表示されることがあります。

```
# zpool status -v pond
.
.
.
pond/amy:<0x1>
pond/rory:<0x1>
```

上のエラーが発生した場合は、上述のとおりに暗号化されたファイルシステムを再マウントします。その後、プールのエラーをスクラブして、クリアします。

```
# zpool scrub pond
# zpool clear pond
```

ファイルシステムのアップグレードの詳細は、165 ページの「ZFS ファイルシステムをアップグレードする」を参照してください。

ZFS の圧縮、複製解除、暗号化のプロパティー間の関連

ZFS の圧縮、複製解除、および暗号化のプロパティーを使用する場合は、次の考慮事項を確認してください。

- ファイルを作成するときに、データは圧縮され、暗号化され、チェックサムが計算されます。続いて、可能な場合はデータが複製解除されます。
- ファイルを読み取るときには、チェックサムが検証され、データが復号化されます。続いて、必要に応じてデータが解凍されます。
- 暗号化され、クローンも作成されたファイルシステム上で dedup プロパティーが有効になっており、このクローン上では zfs key -K コマンドも zfs clone -K コマンドも使用されていない場合は、可能であれば、すべてのクローンのデータが複製解除されます。

ZFS ファイルシステムを暗号化する例

例 33 生の鍵を使用して ZFS ファイルシステムを暗号化する

次の例では、pktool コマンドを使用して aes-256-ccm 暗号化鍵が生成され、/cindykey.file のファイルに書き込まれます。

```
# pktool genkey keystore=file outkey=/cindykey.file keytype=aes keylen=256
```

続いて、tank/home/cindy ファイルシステムが作成されるときに、/cindykey.file が指定されます。

```
# zfs create -o encryption=aes-256-ccm -o keysource=raw,file:///cindykey.file
tank/home/cindy
```

例 34 別の暗号化アルゴリズムで ZFS ファイルシステムを暗号化する

ZFS ストレージプールを作成し、そのストレージプール内のすべてのファイルシステムに暗号化アルゴリズムを継承させることができます。この例では、users プールが作成され、users/home ファイルシステムが作成され、パスフレーズを使用して暗号化されます。デフォルトの暗号化アルゴリズムは aes-128-ccm です。

続いて、users/home/mark ファイルシステムが作成され、aes-256-ccm 暗号化アルゴリズムを使用して暗号化されます。

```
# zpool create -O encryption=on users mirror c0t1d0 c1t1d0 mirror c2t1d0 c3t1d0
Enter passphrase for 'users': xxxxxxxx
Enter again: xxxxxxxx
# zfs create users/home
# zfs get encryption users/home
NAME        PROPERTY    VALUE        SOURCE
users/home  encryption  on           inherited from users
# zfs create -o encryption=aes-256-ccm users/home/mark
# zfs get encryption users/home/mark
NAME               PROPERTY    VALUE        SOURCE
users/home/mark    encryption  aes-256-ccm  local
```

例 35 暗号化した ZFS ファイルシステムのクローンを作成する

クローンファイルシステムが、元のスナップショットと同じファイルシステムの keysource プロパティーを継承する場合、新しい keysource は不要であり、keysource=passphrase,prompt の場合でも新しいパスフレーズの入力を要求されることはありません。クローンには同じ keysource が使用されます。例:

デフォルトでは、暗号化されたファイルシステムの子孫のクローンを作成するときに、鍵の入力を要求されません。

```
# zfs create -o encryption=on tank/ws
Enter passphrase for 'tank/ws': xxxxxxxx
Enter again: xxxxxxxx
# zfs create tank/ws/fs1
# zfs snapshot tank/ws/fs1@snap1
# zfs clone tank/ws/fs1@snap1 tank/ws/fs1clone
```

クローンファイルシステムの新しい鍵を作成する場合、zfs clone -K コマンドを使用します。

子孫の暗号化されたファイルシステムではなく、暗号化されたファイルシステムのクローンを作成する場合は、新しい鍵を入力するように要求されます。例:

```
# zfs create -o encryption=on tank/ws
Enter passphrase for 'tank/ws': xxxxxxxx
Enter again: xxxxxxxx
# zfs snapshot tank/ws@1
# zfs clone tank/ws@1 tank/ws1clone
Enter passphrase for 'tank/ws1clone': xxxxxxxx
Enter again: xxxxxxxx
```

例 36 暗号化された ZFS ファイルシステムを送受信する

次の例では、tank/home/darren@snap1 スナップショットが、暗号化された /tank/home/darren ファイルシステムから作成されます。続いて、暗号化プロパティーを有効にしてスナップショットが bpool/snaps に送信されます。このため、結果として受信されたデータは暗号化されています。ただし、送信プロセス中、tank/home/darren@snap1 ストリームは暗号化されていません。

```
# zfs get encryption tank/home/darren
NAME              PROPERTY    VALUE        SOURCE
tank/home/darren  encryption  on           local
# zfs snapshot tank/home/darren@snap1
# zfs get encryption bpool/snaps
NAME          PROPERTY    VALUE        SOURCE
bpool/snaps  encryption  on           inherited from bpool
# zfs send tank/home/darren@snap1 | zfs receive bpool/snaps/darren1012
# zfs get encryption bpool/snaps/darren1012
NAME                     PROPERTY    VALUE        SOURCE
bpool/snaps/darren1012  encryption  on           inherited from bpool
```

この場合、受信した暗号化されたファイルシステムに対して、新しい鍵が自動的に生成されます。

ZFS ファイルシステムを移行する

ローカルまたはリモート ZFS または UFS ファイルシステムをターゲット ZFS ファイルシステムに移行するには、シャドウ移行を使用します。ターゲットファイルシステムはシャドウファイルシステムとも呼ばれます。

シャドウ移行を管理するには、次のコマンドを使用します。

- shadowadm コマンドは、シャドウ移行の停止、再開、または取り消しを行います。
- オプションが指定された shadowstat コマンドは、システム上で実行されている移行をモニターします。移行の進行状況をモニターするには、shadowstat コマンドをオプションなしで使用します。表示された情報は、Ctrl - C キーを入力するまで継続的に更新されます。

  このコマンドの -E および -e オプションは特に役立ちます。

  - 現在実行中のすべての移行を一覧表示し、エラーのために完了できなかった移行を識別するには、-E オプションを使用します。
  - 特定の移行を表示し、その移行がエラーのために失敗したかどうかを確認するには、-e オプションを使用します。

これらのコマンドを使用している例については、例37「ファイルシステムの移行開始とモニター」を参照してください。

注意 - ファイルシステムを移行しているときは、次の規則に従ってください。

- 移行中は、ファイルシステムへのデータの追加または削除を行わないでください。そうしないと、これらの変更が移行から除外されます。
- 移行が進行中のときは、シャドウファイルシステムの `mountpoint` プロパティーを変更しないでください。

▼ 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

1. **リモート NFS サーバーからデータを移行している場合は、リモートシステムとローカルシステムの両方でネームサービス情報にアクセスできることを確認してください。**
   NFS を使用した大規模な移行の場合、データの一部で移行テストを行なって、UID、GUID、および ACL 情報が正しく移行されるか確認してみることもできます。

2. **必要に応じて、ターゲットシステムにシャドウ移行パッケージをインストールします。**

   ```
   # pkg install shadow-migration
   ```

3. **shadowd サービスを有効にします。**

   ```
   # svcadm enable shadowd
   ```

4. **移行するローカルまたはリモートのファイルシステムを読み取り専用に設定します。**

   - ローカルの ZFS ファイルシステムを移行している場合、これを読み取り専用に設定します。例:

     ```
     # zfs set share.nfs=on /export/home/ufsdata
     ```

   - リモートファイルシステムを移行している場合は、それを読み取り専用として共有します。例:

     ```
     # share set share.nfs.ro='*'/export/home/ufsdata
     ```

5. **移行されるファイルシステムへの shadow ファイルの設定中に、ターゲットファイルシステムを作成します。**

   注記 - 新しいターゲット ZFS ファイルシステムは完全に空である必要があります。そうでない場合は、移行が開始されません。

   移行しているシステムのタイプに応じて、`shadow` 設定を指定します。

- ローカルファイルシステムを移行している場合は、ソースパスを指定します。

  例:

  ```
  # zfs create -o shadow=file:///west/home/data users/home/shadow
  ```

- NFS ファイルシステムを移行している場合は、ソースホストの名前とパスを指定します。

  例:

  ```
  # zfs create -o shadow=nfs://neo/export/home/ufsdata users/home/shadow2
  ```

6. **(オプション) 移行の進行状況をチェックするには、`shadowstat` コマンドを発行します。**

   移行をモニターする方法の例については、例37「ファイルシステムの移行開始とモニター」を参照してください。

注記 - ネットワーク帯域幅によっては、NFS 経由のファイルシステムデータの移行は低速になる場合があります。移行中にシステムがリブートされた場合、その移行は、システムブートが完了したあとに続行されます。

例 37 ファイルシステムの移行開始とモニター

この例では、複数の移行が開始されます。`shadowadm` コマンドが進行中の移行を一覧表示するのに対して、`shadowstat` コマンドはそれらの進行状況をモニターします。

```
# zfs create -o shadow=nfs://system2/pool/data/jsmith/archive rpool/data/copyarchive
# shadowadm list
PATH                                        STATE
/rpool/data/copyarchive                     ACTIVE

# zfs create -o shadow=nfs://system2/pool/data/jsmith/datlogs rpool/data/logcopy
# shadowadm list
PATH                                        STATE
/rpool/data/copyarchive                     ACTIVE
/rpool/data/logcopy                         ACTIVE

# shadowstat
                                    EST
                          BYTES     BYTES           ELAPSED
DATASET                   XFRD      LEFT    ERRORS  TIME
rpool/data/copyarchive    34.4M     3.37G   -       00:00:36
rpool/data/logcopy        1.12K     155K    1       (completed)     エラーが検出される。
rpool/data/copyarchive    34.5M     3.37G   -       00:00:37
rpool/data/logcopy        1.12K     155K    1       (completed)
rpool/data/copyarchive    35.0M     3.37G   -       00:00:38
rpool/data/logcopy        1.12K     155K    1       (completed)
rpool/data/copyarchive    35.2M     3.37G   -       00:00:39
rpool/data/logcopy        1.12K     155K    1       (completed)
^C
```

前の `shadowstat` の出力は、`rpool/data/logcopy` への移行時のエラーを示しています。

shadowstat の -E および -e コマンドオプションの次の出力は、ソケットの移行がサポートされていないために rpool/data/logcopy への移行を完了できなかったことを示しています。この shadowadm コマンドは、移行を取り消します。

```
# shadowstat -E
rpool/data/copyarchive:
No errors encountered.
rpool/data/logcopy:
PATH                                                    ERROR
errdir/cups-socket                                      Operation not supported

# shadowstat -e /rpool/data/logcopy
rpool/data/logcopy:
PATH                                                    ERROR
errdir/cups-socket                                      Operation not supported

# shadowadm cancel /rpool/data/logcopy
```

次の出力は、完了に向けて続行されている rpool/data/copyarchive への移行に関する情報を示しています。

```
# shadowadm list
PATH                                        STATE
/rpool/data/copyarchive                     ACTIVE shadowstat
                                      EST
                                BYTES BYTES           ELAPSED
DATASET                         XFRD  LEFT    ERRORS  TIME
rpool/data/copyarchive          251M  3.16G   -       00:01:27
rpool/data/copyarchive          251M  3.16G   -       00:01:28
rpool/data/copyarchive          252M  3.16G   -       00:01:29
^C

# shadowstat
No migrations in progress.
# shadowadm list
# exit
```

ZFS ファイルシステムをアップグレードする

以前の Oracle Solaris リリースからの ZFS ファイルシステムである場合、最新リリースのファイルシステム機能を利用するために、zfs upgrade コマンドを使用してファイルシステムをアップグレードできます。さらに、このコマンドは、ファイルシステムが古いバージョンを実行しているときに通知します。

たとえば、このファイルシステムの現在のバージョンが 5 だとします。

```
# zfs upgrade
This system is currently running ZFS filesystem version 5.

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

ファイルシステムの各バージョンで使用可能な機能を識別するには、次のコマンドを使用します。

```
# zfs upgrade -v
The following filesystem versions are supported:

VER  DESCRIPTION
---  --------------------------------------------------------
1   Initial ZFS filesystem version
2   Enhanced directory entries
3   Case insensitive and File system unique identifier (FUID)
4   userquota, groupquota properties
5   System attributes
6   Multilevel file system support

For more information on a particular version, including supported releases,
see the ZFS Administration Guide.
```

暗号化されたファイルシステムのアップグレードについては、159 ページの「暗号化された ZFS ファイルシステムをアップグレードする」を参照してください。

♦ ♦ ♦ 第 8 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。また、スナップショットの保存についての情報も提供します。

この章の内容は次のとおりです。

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。しかし、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、スナップショットは古いデータを引き続き参照し、ディスク容量を解放しないため、ディスク領域を消費します。

「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。クローンは、スナップショットからしか作成できません。

ZFS スナップショットには次の特長があります。

- システムのリブート後も残ります。
- 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、178 ページの「ZFS データの保存、送信、および受信」を参照してください。

このセクションの内容は次のとおりです。

- 168 ページの「ZFS スナップショットを作成および破棄する」
- 171 ページの「ZFS スナップショットを表示してアクセスする」
- 173 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` コマンドまたは `zfs snap` 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では、次の規則のいずれかが使用されます。

- *filesystem@snapname*
- *volume@snapname*

スナップショット名は、24 ページの「ZFS コンポーネントの命名」の命名要件に従って付ける必要があります。

次の例では、`friday` という名前を付けた `system1/home/cindy` のスナップショットを作成します。

```
# zfs snapshot system1/home/cindy@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。例:

```
# zfs snapshot -r system1/home@snap1
```

```
# zfs list -t snapshot -r system1/home
NAME                       USED  AVAIL   REFER  MOUNTPOINT
system1/home@snap1            0      -   2.11G  -
system1/home/cindy@snap1      0      -    115M  -
system1/home/lori@snap1       0      -   2.00G  -
system1/home/mark@snap1       0      -   2.00G  -
system1/home/tim@snap1        0      -   57.3M  -
```

スナップショットには変更可能なプロパティーはなく、スナップショットにはデータセットのプロパティーを適用できません。例:

```
# zfs set compression=on system1/home/cindy@friday
cannot set property for 'system1/home/cindy@friday':
this property can not be modified for snapshots
```

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。例:

```
# zfs destroy system1/home/cindy@friday
```

データセットのスナップショットが存在する場合は、データセットを破棄できません。例:

```
# zfs destroy system1/home/cindy
cannot destroy 'system1/home/cindy': filesystem has children
use '-r' to destroy the following datasets:
system1/home/cindy@tuesday
system1/home/cindy@wednesday
system1/home/cindy@thursday
```

さらに、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、103 ページの「ZFS ファイルシステムを破棄する方法」を参照してください。

ZFS スナップショットを保持する

別の自動スナップショットまたはデータ保持のポリシーが原因で、古いスナップショットが不注意に破棄されることがあります。削除されたスナップショットが進行中の ZFS 送受信操作の一部である場合、操作は失敗する可能性があります。このシナリオを回避するには、スナップショットの保留を検討してください。

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 Oracle Solaris リリースでは、スナップショットにクローンが存在しない場合に `zfs destroy` コマンドを使用すると、スナップショットが破棄される可能性がありま

した。この Oracle 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 system1/home/cindy/snap@1 に保持タグ keep を配置します。

```
# zfs hold keep system1/home/cindy@snap1
```

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するには、-r オプションを使用します。例:

```
# zfs snapshot -r system1/home@now
# zfs hold -r keep system1/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。例:

```
# zfs destroy system1/home/cindy@snap1
cannot destroy 'system1/home/cindy@snap1': dataset is busy
```

保持されたスナップショットを破棄するには、-d オプションを使用します。例:

```
# zfs destroy -d system1/home/cindy@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。例:

```
# zfs holds system1/home@now
NAME              TAG   TIMESTAMP
system1/home@now  keep  Fri Aug  3 15:15:53 2012

# zfs holds -r system1/home@now
NAME                     TAG   TIMESTAMP
system1/home/cindy@now  keep  Fri Aug  3 15:15:53 2012
system1/home/lori@now   keep  Fri Aug  3 15:15:53 2012
system1/home/mark@now   keep  Fri Aug  3 15:15:53 2012
system1/home/tim@now    keep  Fri Aug  3 15:15:53 2012
system1/home@now        keep  Fri Aug  3 15:15:53 2012
```

スナップショットまたはスナップショットセット上の保持を解除するには、zfs release コマンドを使用します。例:

```
# zfs release -r keep system1/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。例:

```
# zfs destroy -r system1/home@now
```

2 つのプロパティーでスナップショットの保持情報が識別されます。

- zfs destroy -d コマンドを使ってスナップショットに遅延破棄のマークが付けられている場合は、defer_destroy プロパティーが on に設定されます。それ以外の場合、プロパティーは off に設定されます。

- userrefs プロパティーは、このスナップショット上の保持数に設定されます。

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は変更できますが、スナップショットが作成されたときと同じプールおよびデータセット内に限ります。例:

```
# zfs rename system1/home/cindy@snap1 system1/home/cindy@today
```

また、次のショートカット構文を使用することもできます。

```
# zfs rename system1/home/cindy@snap1 today
```

次のようなスナップショットの rename 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 system1/home/cindy@today pool/home/cindy@saturday
cannot rename to 'pool/home/cindy@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。例:

```
# zfs list -t snapshot -r users/home
NAME                          USED  AVAIL  REFER  MOUNTPOINT
users/home@now               23.5K      -  35.5K  -
users/home@yesterday             0      -    38K  -
users/home/lori@yesterday        0      -  2.00G  -
users/home/mark@yesterday        0      -  1.00G  -
users/home/neil@yesterday        0      -  2.00G  -
# zfs rename -r users/home@yesterday @2daysago
# zfs list -t snapshot -r users/home
NAME                         USED  AVAIL  REFER  MOUNTPOINT
users/home@now              23.5K      -  35.5K  -
users/home@2daysago             0      -    38K  -
users/home/lori@2daysago        0      -  2.00G  -
users/home/mark@2daysago        0      -  1.00G  -
users/home/neil@2daysago        0      -  2.00G  -
```

ZFS スナップショットを表示してアクセスする

デフォルトでは、スナップショットはすでに zfs list 出力には表示されません。スナップショット情報を表示するには、zfs list -t snapshot コマンドを使用する必要があります。代わりに、listsnapshots プールプロパティーを有効にすることもできます。例:

```
# zpool get listsnapshots system1
NAME     PROPERTY       VALUE      SOURCE
```

```
system1  listsnapshots  off        default
# zpool set listsnapshots=on system1
# zpool get listsnapshots system1
NAME     PROPERTY       VALUE      SOURCE
system1  listsnapshots  on         local
```

ファイルシステムのスナップショットは、ファイルシステムのルート内の .zfs/snapshot ディレクトリに配置されています。たとえば、system1/home/cindy が /home/cindy にマウントされている場合、/home/cindy/.zfs/snapshot/thursday ディレクトリにある system1/home/cindy@thursday スナップショットデータにアクセスできます。

```
# ls /system1/home/cindy/.zfs/snapshot
thursday   tuesday    wednesday
```

次の例は、スナップショットを一覧表示する方法を示しています。

```
# zfs list -t snapshot -r system1/home
NAME                          USED  AVAIL  REFER  MOUNTPOINT
system1/home/cindy@tuesday     45K      -  2.11G  -
system1/home/cindy@wednesday   45K      -  2.11G  -
system1/home/cindy@thursday      0      -  2.17G  -
```

次の例は、特定のファイルシステム用に作成されたスナップショットを一覧表示する方法を示しています。

```
# zfs list -r -t snapshot -o name,creation system1/home
NAME                          CREATION
system1/home/cindy@tuesday    Fri Aug  3 15:18 2012
system1/home/cindy@wednesday  Fri Aug  3 15:19 2012
system1/home/cindy@thursday   Fri Aug  3 15:19 2012
system1/home/lori@today       Fri Aug  3 15:24 2012
system1/home/mark@today       Fri Aug  3 15:24 2012
```

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すると、最初はそのディスク容量がスナップショットとファイルシステム間で共有されます。ただし、以前のスナップショットと共有される可能性もあります。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。

スナップショットの容量 referenced プロパティーの値は、スナップショットが作成されたときのファイルシステムの値と同じです。

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。例:

```
$ zfs list -o space -r rpool
```

```
NAME                     AVAIL   USED  USEDSNAP  USEDDS  USEDREFRESERV  USEDCHILD
rpool                     124G  9.57G         0    302K              0      9.57G
rpool/ROOT                124G  3.38G         0     31K              0      3.38G
rpool/ROOT/solaris        124G  20.5K         0       0              0      20.5K
rpool/ROOT/solaris/var    124G  20.5K         0   20.5K              0          0
rpool/ROOT/solaris-1      124G  3.38G     66.3M   3.14G              0       184M
rpool/ROOT/solaris-1/var  124G   184M     49.9M    134M              0          0
rpool/VARSHARE            124G  39.5K         0   39.5K              0          0
rpool/dump                124G  4.12G         0   4.00G           129M          0
rpool/export              124G    63K         0     32K              0        31K
rpool/export/home         124G    31K         0     31K              0          0
rpool/swap                124G  2.06G         0   2.00G          64.7M          0
```

これらのプロパティーについては、zfs(1M) のマニュアルページで used プロパティー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。

以前のスナップショットにロールバックするには、`-r` オプションを指定して、中間のスナップショットをすべて破棄する必要があります。

中間の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します。

注記 - 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、アンマウントしてから再マウントする必要があります。ファイルシステムをアンマウントできない場合は、ロールバックに失敗します。必要に応じて `-f` オプションを使用すると、ファイルシステムが強制的にアンマウントされます。

次の例では、`system1/home/cindy` ファイルシステムは `tuesday` スナップショットまでロールバックされます。

```
# zfs rollback system1/home/cindy@tuesday
cannot rollback to 'system1/home/cindy@tuesday': more recent snapshots exist
use '-r' to force deletion of the following snapshots:
system1/home/cindy@wednesday
system1/home/cindy@thursday
# zfs rollback -r system1/home/cindy@tuesday
```

次の例では、ファイルシステムが以前の `tuesday` スナップショットまでロールバックされるため、`wednesday` および `thursday` スナップショットは破棄されます。

```
# zfs list -r -t snapshot -o name,creation system1/home/cindy
```

```
NAME                        CREATION
system1/home/cindy@tuesday  Fri Aug  3 15:18 2012
```

ZFS スナップショットの相違点の識別 (zfs diff)

zfs diff コマンドを使用して、ZFS スナップショットの相違点を判別できます。

たとえば、次の 2 つのスナップショットが作成されるものとします。

```
$ ls /system1/home/tim
fileA
$ zfs snapshot system1/home/tim@snap1
$ ls /system1/home/tim
fileA  fileB
$ zfs snapshot system1/home/tim@snap2
```

2 つのスナップショットの相違点を識別するには、次の例のような構文を使用します。

```
$ zfs diff system1/home/tim@snap1 system1/home/tim@snap2
M       /system1/home/tim/
+       /system1/home/tim/fileB
```

出力で、M はディレクトリが変更されたことを示します。+ は、後者のスナップショットに fileB が存在していることを示します。

次の出力の R は、スナップショットのファイルの名前が変更されたことを示しています。

```
$ mv /system1/cindy/fileB /system1/cindy/fileC
$ zfs snapshot system1/cindy@snap2
$ zfs diff system1/cindy@snap1 system1/cindy@snap2
M       /system1/cindy/
R       /system1/cindy/fileB -> /system1/cindy/fileC
```

次の表は、zfs diff コマンドによって識別されるファイルまたはディレクトリの変更を要約したものです。

ファイルまたはディレクトリの変更	識別子
ファイルまたはディレクトリが変更されたかファイルまたはディレクトリのリンクが変更されました	M
ファイルまたはディレクトリは古いスナップショットに存在するが、最近のスナップショットには存在しません	–
ファイルまたはディレクトリは最近のスナップショットに存在するが、古いスナップショットには存在しません	+
ファイルまたはディレクトリの名前が変更されました	R

詳細については、zfs(1M) のマニュアルページを参照してください。

再帰的な ZFS スナップショットの相違点の表示

`zfs diff` コマンドを使用して別々のスナップショットを比較すると、新しいファイルシステムやディレクトリなどの高レベルの相違点が表示されます。たとえば、`sales` ファイルシステムには、それぞれの子孫ファイルシステム内にファイルを持つ `data` および `logs` という 2 つの子孫ファイルシステムが含まれています。

```
# zfs list -r sales
NAME         USED  AVAIL  REFER  MOUNTPOINT
sales       1.75M  66.9G    33K  /sales
sales/data   806K  66.9G   806K  /sales/data
sales/logs   806K  66.9G   806K  /sales/logs
```

`sales@snap1` と `sales@snap2` との間で、高レベルの相違点を表示できます。ここで、主な相違点は `sales/logs` ファイルシステムの追加です。

```
# zfs diff sales@snap1 sales@snap2
M       /sales/
+       /sales/logs
```

次のような構文を使用すると、スナップショットの相違点 (ファイル名を含む) を再帰的に識別できます。

```
# zfs diff -r -E sales@snap1
D       /sales/ (sales)
+       /sales/data
D       /sales/data/    (sales/data)
+       /sales/data/dfile.1
+       /sales/data/dfile.2
+       /sales/data/dfile.3
# zfs diff -r -E sales@snap2
D       /sales/ (sales)
+       /sales/data
+       /sales/logs
D       /sales/logs/    (sales/logs)
+       /sales/logs/lfile.1
+       /sales/logs/lfile.2
+       /sales/logs/lfile.3
D       /sales/data/    (sales/data)
+       /sales/data/dfile.1
+       /sales/data/dfile.2
+       /sales/data/dfile.3
```

この出力では、`D` で始まり、(*name*) で終わる行がファイルシステム (データセット) およびマウントポイントを示しています。

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、スナップショットからしか作成できません。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。クローンはファイルシステム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。

クローンでは、作成元のデータセットのプロパティーが継承されません。`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。詳細については、126 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。

このセクションの内容は次のとおりです。

- 176 ページの「ZFS クローンを作成する」。
- 177 ページの「ZFS クローンを破棄する」。
- 177 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。クローンの作成元となるスナップショット、および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すると、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、初期の内容がスナップショット `system1/ws/gate@yesterday` と同じ `system1/home/matt/bug123` という名前の新しいクローンを作成します。

```
# zfs snapshot system1/ws/gate@yesterday
# zfs clone system1/ws/gate@yesterday system1/home/matt/bug123
```

次の例では、スナップショット `projects/newproject@today` からクローンが作成されたワークスペースを一時ユーザー用に `projects/teamA/tempuser` として作成します。次に、クローンが作成されたワークスペース上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 share.nfs=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。例:

```
# zfs destroy system1/home/matt/bug123
```

親の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。

ZFS ファイルシステムを ZFS クローンで置き換える

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には、`zfs promote` コマンドを使用します。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、クローンの作成元であ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。詳細については、177 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、`system1/test/productA` ファイルシステムのクローンが作成されたあとに、クローンファイルシステム `system1/test/productAbeta` が元の `system1/test/productA` ファイルシステムになります。

```
# zfs create system1/test
# zfs create system1/test/productA
# zfs snapshot system1/test/productA@today
# zfs clone system1/test/productA@today system1/test/productAbeta
# zfs list -r system1/test
NAME                          USED  AVAIL  REFER  MOUNTPOINT
system1/test                  104M  66.2G    23K  /system1/test
system1/test/productA         104M  66.2G   104M  /system1/test/productA
system1/test/productA@today      0      -   104M  -
system1/test/productAbeta        0  66.2G   104M  /system1/test/productAbeta
# zfs promote system1/test/productAbeta
# zfs list -r system1/test
NAME                             USED  AVAIL  REFER  MOUNTPOINT
system1/test                     104M  66.2G    24K  /system1/test
system1/test/productA               0  66.2G   104M  /system1/test/productA
system1/test/productAbeta        104M  66.2G   104M  /system1/test/productAbeta
system1/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。例:

```
# zfs rename system1/test/productA system1/test/productAlegacy
# zfs rename system1/test/productAbeta system1/test/productA
# zfs list -r system1/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。例:

```
# zfs destroy system1/test/productAlegacy
```

ZFS データの保存、送信、および受信

`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。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することもできます。

さらに、このリリースでは ZFS データの転送を再開する機能が組み込まれています。特に、ネットワーク障害またはシステム障害が原因で大量の ZFS データの転送が中断されることがあります。すべてのデータを再送信しなければならない事態を避けるため、`zfs send` および `zfs receive` コマンドに、ZFS データの送信を再開するための `-c` オプションを付けて実行できます。詳細は、183 ページの「再開可能レプリケーションの使用」を参照してください。

このセクションの内容は次のとおりです。

ZFS データを保存する際は、次のバックアップ解決策に注意してください。

- **企業向けバックアップ製品** –これらの製品には、次のような機能が備わっています。
  - ファイルごとのレプリケーション
  - 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、必要に応じて、以前のファイルシステムバージョンに戻します。

  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、167 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` コマンドを使用すると、スナップショット間の増分変更を保存できますが、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。
- **リモートレプリケーション** – あるシステムから別のシステムにファイルシステムをコピーします。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。特殊な構成やハードウェアは必要ありません。`zfs send` および `zfs receive` コマンドを使用して ZFS ファイルシステムをレプリケートすると、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ます。
- 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 スナップショットストリームのタイプ

`zfs send` コマンドを使用して、1 つ以上のスナップショットのストリームを作成できます。続いて、`zfs receive` コマンドを使用することにより、このスナップショットストリームを使用して、ZFS ファイルシステムまたはボリュームを再作成できます。

スナップショットストリームを作成する際に使用された `zfs send` オプションによって、生成されるストリーム形式のタイプが決まります。

- 完全なストリーム – データセットが作成された時間から指定されたスナップショットまで、すべてのデータセットの内容から構成されます。

  `zfs send` コマンドで生成されたデフォルトのストリームが完全なストリームです。これには、1 つのファイルシステムまたはボリュームから指定されたスナップショットまで含まれます。ストリームには、コマンドで指定されたスナップショット以外のスナップショットは含まれません。
- 増分ストリーム – あるスナップショットと別のスナップショットの差から構成されます。

ストリームパッケージとは、1 つ以上の完全ストリームまたは増分ストリームを含むストリームタイプです。ストリームパッケージのタイプには、次のものがあります。

- レプリケーションストリームパッケージ - 指定したデータセットとその子孫で構成されます。すべての中間スナップショットを含みます。クローンの作成元であるデータセットがコマンド行で指定されたスナップショットの子孫でない場合は、この作成元のデータセットがストリームパッケージに含まれません。ストリームを受信するには、元のデータセットが受信先のストレージプールに存在する必要があります。

  次に示すデータセットとそれらの作成元の一覧が、表示される順序で作成されたと仮定します。

```
NAME                    ORIGIN
pool/a                  -
pool/a/1                -
pool/a/1@clone          -
pool/b                  -
pool/b/1                pool/a/1@clone
pool/b/1@clone2         -
pool/b/2                pool/b/1@clone2
pool/b@pre-send         -
pool/b/1@pre-send       -
pool/b/2@pre-send       -
pool/b@send             -
pool/b/1@send           -
pool/b/2@send           -
```

  次の構文を使用して、レプリケーションストリームパッケージが作成されたと仮定します。

```
# zfs send -R pool/b@send ....
```

  このパッケージは、次の完全ストリームおよび増分ストリームで構成されます。

```
TYPE    SNAPSHOT                INCREMENTAL FROM
full    pool/b@pre-send         -
incr    pool/b@send             pool/b@pre-send
incr    pool/b/1@clone2         pool/a/1@clone
incr    pool/b/1@pre-send       pool/b/1@clone2
incr    pool/b/1@send           pool/b/1@send
incr    pool/b/2@pre-send       pool/b/1@clone2
incr    pool/b/2@send           pool/b/2@pre-send
```

  この出力では、レプリケーションストリームパッケージに `pool/a/1@clone` スナップショットが含まれていません。したがって、このレプリケーションストリームパッケージは、すでに `pool/a/1@clone` スナップショットがあるプールでのみ受信できます。

- **再帰的ストリームパッケージ** – 指定したデータセットとその子孫で構成されます。複製ストリームパッケージとは異なり、ストリームに含まれる複製されたデータセットの複製元でないかぎり、中間スナップショットは含まれません。デフォルトでは、データセットの作成元がコマンドで指定されたスナップショットの子孫でない場合の動作は、レプリケーションストリームと同様です。自己完結型の再帰的ストリームは、外部の依存関係を持たないことに注意してください。

  再帰的ストリームパッケージを作成するには、次の例のような構文を使用します。

  ```
  # zfs send -r pool/b@send ...
  ```

  このパッケージ例は、次の完全ストリームおよび増分ストリームで構成されます。

  ```
  TYPE    SNAPSHOT                INCREMENTAL FROM
  full    pool/b@send             -
  incr    pool/b/1@clone2         pool/a/1@clone
  incr    pool/b/1@send           pool/b/1@clone2
  incr    pool/b/2@send           pool/b/1@clone2
  ```

  この出力では、再帰的ストリームパッケージに `pool/a/1@clone` スナップショットが含まれていません。したがって、レプリケーションストリームパッケージと同様に、この再帰的ストリームパッケージは、すでに `pool/a/1@clone` スナップショットがあるプールでのみ受信できます。この動作は、前述の複製ストリームパッケージの場合と似ています。
- **自己完結型の再帰的ストリームパッケージ** - このタイプのパッケージは、ストリームパッケージに含まれないどのデータセットにも依存しません。再帰的ストリームパッケージを作成するには、次の例のような構文を使用します。

  ```
  # zfs send -rc pool/b@send ...
  ```

  このパッケージ例は、次の完全ストリームおよび増分ストリームで構成されます。

  ```
  TYPE    SNAPSHOT                INCREMENTAL FROM
  full    pool/b@send             -
  full    pool/b/1@clone2
  incr    pool/b/1@send           pool/b/1@clone2
  incr    pool/b/2@send           pool/b/1@clone2
  ```

  このような自己完結型の再帰的ストリームは、`pool/b/1@clone2` スナップショットの完全なストリームを含んでいるため、外的な依存関係なしに `pool/b/1` スナップショットを受信できます。

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、異なる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、次の例のようなコマンドを使用します。

```
# zfs send ool/dana@snap1 | zfs recv spool/ds01
```

ヒント - `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。例:

```
sys1# zfs send ool/dana@snap1 | ssh sys2 zfs recv pool/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。

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。例:

```
# zfs send pool/fs@snap | gzip > backupfile.gz
```

例 38 増分 ZFS データを送信する

`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。例:

```
sys1# zfs send -i ool/dana@snap1 system1/dana@snap2 | ssh system2 zfs recv ool/dana
```

最初の引数 (`snap1`) は以前のスナップショットで、2 番目の引数 (`snap2`) はそれよりあとのスナップショットです。この場合は、増分データの受信を正常に行うために `ool/dana` ファイルシステムがすでに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コンポーネントとして指定できます。その後、*snap1* には `@` 記号のあとの名前のみを指定する必要があります。これにより、*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。例:

```
sys1# zfs send -i snap1 ool/dana@snap2 | ssh system2 zfs recv ool/dana
```

このショートカット構文は、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
```

元の受信側ファイルシステム内のファイル情報にアクセスすると、増分スナップショットの受信操作が失敗して、次のようなメッセージが表示される可能性があります。

```
cannot receive incremental stream of pool/dana@snap2 into pool/dana:
most recent snapshot of pool/dana@snap2 does not match incremental source
```

元の受信側ファイルシステム内のファイル情報にアクセスする必要がある場合で、増分スナップショットを受信側ファイルシステムで受信する必要もある場合は、atime プロパティーを off に設定することを考慮してください。

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。例:

```
# zfs send pool/fs@snap | gzip > backupfile.gz
```

例 39 Oracle Solaris 12.0 データセットから ZFS データを送信する

出力データストリーム内でレコード単位のチェックサムを使用する機能は、デフォルトで使用可能になっています。データを古いシステムに転送するには、nocheck 引数を使用してこの機能を無効にする必要があります。

```
# zfs send -s nocheck pool/dana@snap1 | zfs recv pool/ds01
```

再開可能レプリケーションの使用

zfs receive を使用したデータの転送が中断された場合、プロセスを再開できます。たとえば、このコマンドで転送を開始したとします。

```
system1# zfs send pool/dana@snap1 | ssh system2 zfs recv pool/dana
```

転送が中断された場合、データセットが不完全になります。次の一連のコマンドを使用して転送を再開できます。

```
system1# ssh system2 zfs receive -C pool/dana | zfs send -C pool/dana@snap1 | \
     ssh system2 zfs receive pool/dana
```

不完全なデータセットを表示するには、zfs list -I コマンドを使用します。125 ページの「不完全な ZFS データセットを一覧表示する」を参照してください。

ZFS スナップショットを受信する

ファイルシステムスナップショットを作成するときは、次のキーポイントに注意してください。

- 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アンマウントされます。
- 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。
- ターゲットシステム上に、受信されるソースファイルシステムと同じ名前を持つファイルシステムが存在してはいけません。ターゲットシステム上にファイルシステム名がすでに存在する場合は、ファイルシステム名を変更してください。

例:

```
# zfs send system1/gozer@0830 > /bkups/gozer.083006
# zfs receive system1/gozer2@today < /bkups/gozer.083006
# zfs rename system1/gozer system1/gozer.old
# zfs rename system1/gozer2 system1/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。

```
sys2# rm newsys/dana/file.1
```

次に、system1/dana@snap3 の増分送信を実行します。新しい増分スナップショットを受信したり、-F オプションを使用してロールバック手順を排除したりするには、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。例:

```
sys1# zfs send -i system1/dana@snap2 system1/dana@snap3 | ssh sys2 zfs recv -F newsys/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 オプションを使用する場合は、次の例のようなメッセージが表示されます。

```
sys1# zfs send -i system1/dana@snap4 system1/dana@snap5 | ssh sys2 zfs recv newsys/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
  ```

ZFS スナップショットストリームに異なるプロパティー値を適用する

特定のファイルシステムプロパティー値を含む ZFS スナップショットストリームを送信したあとに、スナップショットストリームの受信時に別のローカルプロパティー値を指定するか、またはスナップショットストリームの受信時に元のプロパティー値が使用されるように指定すると、元のファイルシステムを再作成できます。さらに、ス

ナップショットストリームを受信したときにファイルシステムプロパティーを無効にすることもできます。

- ローカルのプロパティー値を受信した値 (存在する場合) に戻すには、zfs inherit -S コマンドを使用します。プロパティーに受信した値がない場合、-S オプションの動作は、オプションを付けなかった場合と同じです。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 で表示される列を指定できます。新しい非デフォルトの RECEIVED 列を含めるには、-o オプションを使用します。RECEIVED を含むすべての列を含めるには、-o all オプションを使用します。
- -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には、-p オプションを使用します。
- 送信されたスナップショット名の最後の要素を使用して新しいスナップショット名を指定するには、-e オプションを使用します。

  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  -
```

元のプロパティー値の保持

場合によっては、送信ストリーム内のファイルシステムプロパティーが受信側のファイルシステムに該当しなかったり、mountpoint プロパティー値などのローカルファイルシステムプロパティーが復元を妨害したりすることがあります。

たとえば、system1/data というファイルシステムで compression プロパティーが無効になっていると仮定します。system1/data ファイルシステムのスナップショットが、プロパティー (-p オプション) を指定してバックアッププールに送信され、compression プロパティーが有効な状態で受信されます。

```
# zfs get compression system1/data
```

```
NAME          PROPERTY     VALUE     SOURCE
system1/data  compression  off       default
# zfs snapshot system1/data@snap1
# zfs send -p system1/data@snap1 | zfs recv -o compression=on -d bpool
# zfs get -o all compression bpool/data
NAME        PROPERTY     VALUE     RECEIVED  SOURCE
bpool/data  compression  on        off       local
```

この例では、スナップショットが bpool に受信されたとき、compression プロパティーは有効になります。したがって、bpool/data では、compression 値は on です。

このスナップショットストリームが復元目的で restorepool という新規プールに送信される場合、元のスナップショットプロパティーをすべて保持することが必要なことがあります。この場合、元のスナップショットプロパティーを復元するために zfs send -b オプションを使用します。例:

```
# zfs send -b bpool/data@snap1 | zfs recv -d restorepool
# zfs get -o all compression restorepool/data
NAME              PROPERTY     VALUE     RECEIVED  SOURCE
restorepool/data  compression  off       off       received
```

この例では、compression 値は off です。これは、元の system1/data ファイルシステムからのスナップショット圧縮値を表します。

元のプロパティー値を無効にする

スナップショットストリーム内にローカルファイルシステムのプロパティー値があり、スナップショットストリームを受信したときにこのプロパティーを無効にする場合、zfs receive -x オプションを使用します。たとえば次のコマンドでは、すべてのファイルシステムプロパティーを予約した状態で home ディレクトリファイルシステムの再帰的なスナップショットストリームをバックアッププールに送信しますが、割り当て制限プロパティー値は設定されません。

```
# zfs send -R system1/home@snap1 | zfs recv -x quota bpool/home
# zfs get -r quota bpool/home
NAME                    PROPERTY  VALUE  SOURCE
bpool/home              quota     none   local
bpool/home@snap1        quota     -      -
bpool/home/lori         quota     none   default
bpool/home/lori@snap1   quota     -      -
bpool/home/mark         quota     none   default
bpool/home/mark@snap1   quota     -      -
```

再帰的なスナップショットが -x オプションで受信されなかった場合、割り当て制限プロパティーは受信側ファイルシステム内で設定されます。

```
# zfs send -R system1/home@snap1 | zfs recv bpool/home
# zfs get -r quota bpool/home
NAME                    PROPERTY  VALUE  SOURCE
bpool/home              quota     none   received
bpool/home@snap1        quota     -      -
bpool/home/lori         quota     10G    received
bpool/home/lori@snap1   quota     -      -
bpool/home/mark         quota     10G    received
```

```
bpool/home/mark@snap1  quota     -      -
```

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

このセクションでは、-I および -R オプションを zfs send コマンドに使用して、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には、-I オプションを使用します。このオプションを使用すると、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こともできます。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レプリケーションストリームを送信するには、-R オプションを使用します。レプリケーションストリームが受信されると、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、およびクローンが保持されます。
- -c オプションを付けずに zfs send -r コマンドまたは zfs send -R コマンドを使用してパッケージストリームを送信すると、一部の状況ではクローンの origin が省略されます。詳細は、179 ページの「ZFS スナップショットストリームのタイプ」を参照してください。
- 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。
  - レプリケーションストリームの受信時に -F オプション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。したがって、必要に応じて、受信操作を取り消して、ファイルシステムを以前の状態に復元できます。
  - 増分ストリームを送信するときに -I を使用すると、snapA と snapD 間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 スナップショットのみが送信されます。
- このようなタイプの 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動作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。

例 40 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

-I オプションを使用すると、増分スナップショットの A グループを結合して 1 つのスナップショットを作成できます。例:

```
# zfs send -I pool/fs@snapA pool/fs@snapD > /snaps/fs@all-I
```

次に、増分スナップショット 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  -
```

-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。例:

```
# 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 ファイルシステムおよびすべての子孫ファイルシステムを指定されたスナップショットまでレプリケートするには、-R オプションを使用します。このストリームが受信されると、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、およびクローンが保持されます。

次の例では、ユーザーファイルシステムのスナップショットを作成します。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、-R コマンドを使用して、users ファイルシステムとその子孫をレプリケートし、レプリケート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リモートでコピーできます。例:

```
# zfs send system1/cindy@today | ssh newsys zfs recv sandbox/restfs@today
```

このコマンドは、system1/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 を使用するように構成されています。

ZFS プール操作をモニターする

データ上で実行されるバックグラウンドタスク (データの送信、受信、スクラブ、再同期化など) を開始する ZFS コマンドを発行すると、これらのタスクのステータスと進捗状況をリアルタイムでモニターできます。情報が表示される頻度を指定できます。また、モニタリングが実行される期間を指定することもできます。

プールの操作をモニターするには、zpool monitor コマンドを使用します。このコマンドでは、使用するオプションに応じて、タスクに関する次のような情報が提供されます。この情報は、プールごとに個別に提供されます。

- 開始時間。
- 現在のデータ量。
- タイムスタンプ (機能単位で該当する場合)。
- タスク開始時のデータ量 (該当する)。

個々のプール上のタスク、またはシステムにある既存のすべてのプール上のタスクに関する情報を表示できます。

次のように `zpool monitor` コマンドを使用します。

`zpool monitor -t` *provider* `[-T d|u]` [*pool*] [*interval* [*count*]]

-t *provider*
: タスク情報を表示する次のプロバイダのいずれかを指定します。プロバイダには、次のいずれかを指定できます。
  - `send`
  - `receive` または `recv`
  - `scrub`
  - `resilver`

注記 - 最新のプロバイダ一覧については、`zpool help monitor` コマンドを入力してください。

-T d|u
: タイムスタンプとその表示形式を指定します。標準の日付形式で表示するには、`d` を指定します。時間の内部表現の出力表現を表示するには、`u` を指定します。これらの表示形式の詳細については、`date(1)` および `time(2)` のマニュアルページを参照してください。

interval
: 表示する情報が更新される頻度 (秒単位)。

count
: 指定した間隔内に、コマンドがタスクに関する情報を表示する回数。

  count が指定されている場合、このコマンドは情報を `count` で指定された回数更新してから、終了します。*count* が指定されていない場合、Ctrl-C を押すまで、継続的に情報が更新されます。

注記 - また、表示する情報をカスタマイズすることもできます。`-o` オプションを使用すると、コマンドの出力に含まれるように表示フィールドをフィルタ処理できます。`-p` オプションを使用すると、マシンで解析可能な形式で情報を表示できます。詳細については、`zpool(1M)` のマニュアルページを参照してください。

コマンドの出力によって、情報が特定のフィールドに従って整理されます。

DONE　　`zpool monitor` コマンドが発行されてから、これまでに処理されたデータ量。

OTHER　　指定されたプロバイダに応じた追加情報 (現在処理中の項目や現在のタスク状態など)。

PCTDONE　　処理されたデータの割合。

POOL　　情報の取得元であるプール。

PROVIDER　　情報を提供しているタスク。

SPEED　　1 秒当たりの単位 (通常はバイトですが、プロバイダで使用する単位に依存します)。

STRTTIME　　表示されたタスクでプロバイダが起動された時間。

TAG　　操作全体を区別します。TAG 値は常に一意ですが、後続の操作で値を再使用できます。たとえば、両方のタスクが同じデータセット上で動作している場合でも、2 つの同時送信操作は別々の TAG 値を持っています。

TIMELEFT　　特定のタスクが完了する相対時間。

TIMESTMP　　モニター対象のデータのスナップショットが取得される時間。

TOTAL　　処理されるデータの合計量を見積もります。

例 41　　ZFS スナップショットのストリーム表現をモニターする

`zfs send` コマンドは、スナップショットのストリーム表現を作成します。次の例は、このタスクに関する情報を取得する方法を示しています。この情報は、5 秒間隔で 2 回更新されます。

```
# zpool monitor -t send 5 2
POOL       PROVIDER PCTDONE  TOTAL   SPEED   TIMELEFT   OTHER
poolA      send     41.7     10.0G   5.93M   16m49s     poolA/fs:1/team3@all
poolB      send     53.9     10.0G   7.71M   10m13s     poolB/fs1/team3@all
poolC      send     97.9     10.0G   14.4M      14s     poolC/fs1/team1@all
poolA      send     43.5     10.0G   6.17M   15m40s     poolA/fs:1/team3@all
poolB      send     55.8     10.0G   7.95M    9m30s     poolB/fs1/team3@all
poolC      send     99.2     10.0G   14.5M       5s     poolC/fs1/team1@all
```

例 42　　ストリームの受信をモニターする

この例は、受信操作のステータスおよび進捗状況をモニターする方法を示しています。回数が指定されていない場合は、継続的に情報が 5 秒ごとに更新されます。管理者が Ctrl-C を押すと、モニタリングが終了します。

```
# zpool monitor -t recv 5
pool           provider  pctdone  total speed timeleft   other
poolA          receive    34.0    12.0G 6.01M 22m31s     poolA/backup_all/fs2
poolA          receive    68.0    6.01G 6.01M 5m28s      poolA/backup_fs:1
poolB          receive    20.6    10.0G 3.04M 44m39s     poolB/backup_all/fs2
poolB          receive   100.0    6.01G 9.48M 1s         poolB/backup_fs1
poolB          receive    16.7    12.0G 4.16M 41m04s     poolB/pA-bkup/fs2
poolC          receive    26.2    10.0G 3.98M 31m41s     poolC/backup_all/fs2
^C
```

例 43 再同期化操作をモニターする

この例は、システム上にある 3 つすべての ZFS プールでの再同期化操作のステータスをチェックする方法を示しています。

```
# zpool monitor -t resilver 5 4
pool       provider  pctdone  total  speed  timeleft   other
poolA      resilver   10.7    12.0G  37.7M  4m50s      (2/2)
poolB      resilver    7.1    10.0G  31.5M  5m02s      (2/2)
poolC      resilver    1.8    10.0G  42.9M  3m54s      (2/2)
poolA      resilver   13.9    12.0G  38.0M  4m38s      (2/2)
poolB      resilver    9.0    10.0G  30.4M  5m07s      (2/2)
poolC      resilver    5.3    10.0G  41.7M  3m52s      (2/2)
poolA      resilver   14.7    12.0G  36.1M  4m50s      (2/2)
poolB      resilver   10.8    10.0G  29.2M  5m12s      (2/2)
poolC      resilver    7.2    10.0G  41.0M  3m51s      (2/2)
poolA      resilver   14.7    12.0G  32.8M  5m19s      (2/2)
poolB      resilver   10.8    10.0G  29.6M  5m08s      (2/2)
poolC      resilver    7.2    10.0G  40.7M  3m53s      (2/2)
```

例 44 スクラブ操作をモニターする

この例は、poolB でのスクラブ操作の進捗状況をモニターする方法を示しています。

```
# zpool monitor -t scrub 5 poolB
pool        provider  pctdone  total  speed  timeleft
poolB       scrub      16.3    14.0G  35.3M   5m39s
poolB       scrub      16.4    14.0G  33.2M   6m00s
poolB       scrub      16.5    14.0G  31.1M   6m25s
poolB       scrub      16.5    14.0G  29.3M   6m48s
^C
```

ZFS ファイルのコピー

cp -z コマンドを使用して大きいファイルをすばやくコピーできます。-z オプションは、メタデータおよびすべてのデータをコピーする代わりに、各レコードに関連付けられたメタデータをコピーします。レコードサイズが 4K 以上のファイルの場合、この方法は、標準の cp コマンドを使用するよりもずっと速くなる可能性があります。また、OS イメージのように 1 つの大きいファイルの多くのコピーが必要な場合もたいへん便利です。詳細は、reflink(3c) のマニュアルページを参照してください。

♦ ♦ ♦ 第 9 章

9 ACL および属性を使用した 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 UNIX 標準のアクセス権より詳細にアクセス権を制御する方法で、ZFS ファイルを保護する方法について説明します。

この章で扱う内容は、次のとおりです。

Oracle Solaris ACL モデル

Oracle Solaris ACL モデルは、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完全にサポートしています。Windows NT スタイルの ACL に似ており、標準のファイルアクセス権を使用する場合よりも、より詳細なアクセス制御が提供されます。`chmod` コマンドと `ls` コマンドを使用して ACL を設定および表示します。

ACL モデルには、アクセスチェックに影響する `ALLOW` と `DENY` の 2 種類のアクセス制御エントリ (ACE) があります。そのため、一連のアクセス権を定義する 1 つの ACE から、その ACE に定義されていないアクセス権が許可されているか拒否されているかを推論することはできません。

ACL およびバックアップ製品については、179 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 形式

ACL には 2 つの基本的な形式があります。

- **簡易 ACL** – owner@、group@、および everyone@ として表される従来の UNIX ユーザーカテゴリのエントリのみを含みます。

  新しく作成したファイルの場合、デフォルトの ACL には次のエントリがあります。

  ```
  0:owner@:read_data/write_data/append_data/read_xattr/write_xattr
  /read_attributes/write_attributes/read_acl/write_acl/write_owner
  /synchronize:allow
  1:group@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
  2:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
  :allow
  ```

  新しく作成したディレクトリの場合、デフォルトの ACL には次のエントリがあります。

  ```
  0:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
  /append_data/read_xattr/write_xattr/execute/delete_child
  /read_attributes/write_attributes/read_acl/write_acl/write_owner
  /synchronize:allow
  1:group@:list_directory/read_data/read_xattr/execute/read_attributes
  /read_acl/synchronize:allow
  2:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
  /read_acl/synchronize:allow
  ```

- **非簡易 ACL** - 追加したユーザーカテゴリのエントリが含まれます。エントリには継承フラグも含まれる場合があります。または、従来とは異なる方法で順序付けられます。

  非簡易エントリは、次の例のようになります。ここで、アクセス権はユーザー Bob に付与されています。

  ```
  0:user:bob:read_data/write_data:file_inherit:allow
  ```

ACL エントリの説明

参照として次のサンプルエントリを使用して ACL エントリを構成する要素を理解します。これらの要素は簡易および非簡易 ACL の両方に適用されます。

```
0:user:bob:read_data/write_data:file_inherit:allow
```

インデックス　エントリの先頭の番号 (例では数字のゼロ (0))。インデックスは特定のエントリを識別し、ACL 内のその他のエントリと区別します。

ACL エントリタイプ　ユーザーカテゴリ。簡易 ACL では、owner@、group@、および everyone@ のエントリのみが設定されます。非簡易 ACL では、user:*username* および group:*groupname* が追加されます。この例では、エントリのタイプは user:bob です。

アクセス特権　エントリタイプに許可または拒否されるアクセス権。この例では、ユーザー Bob のアクセス権は read_data と write_data です。

継承フラグ　アクセス権がディレクトリ構造で伝播する方法を制御する ACL フラグのオプションのリストです。サンプルエントリでは、file_inherit もユーザー Bob に付与されます。

アクセス権の制御　エントリのアクセス権を許可または拒否するかを決定します。この例では、Bob のアクセス権は許可されています。

次の表は、ACL エントリタイプをそれぞれ示しています。

表 7　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。ユーザー名またはユーザー ID を含む ACL-entry-ID を含める必要があります。値が有効な数値 UID またはユーザー名でない場合、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。グループ名またはグループ ID を含む ACL エントリ ID を含める必要があります。値が有効な数値 GID またはグループ名でない場合、ACL エントリタイプは無効です。

次の表は、ACL アクセス特権を示しています。

表 8　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。

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	ファイルを変更するアクセス権。ただし、ファイルの終わり (EOF) から始まる場合のみ。
delete	d	ファイルを削除するためのアクセス権。delete アクセス権の特定の動作の詳細については、表9を参照してください。
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。delete_child アクセス権の特定の動作の詳細については、表9を参照してください。
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) を読み取るアクセス権。これは stat レベル属性に相当します。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	読み取り操作と書き込み操作が同期された ZFS サーバーでファイルにローカルアクセスするアクセス権。
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時間を任意の値に変更するためのアクセス権。
write_acl	C	ACL を書き込むためのアクセス権。つまり chmod コマンドを使用して ACL を変更することができます。
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。

次の表に、ACL delete および delete_child の動作の追加詳細を示します。

表 9 ACL delete および delete_child アクセス権の動作

親ディレクトリのアクセス権	ターゲットオブジェクトのアクセス権		
" " (空)	ACL は delete を許可	ACL は delete を拒否	未指定のアクセス権を削除
ACL は delete_child を許可	パーミット	パーミット	パーミット
ACL は delete_child を拒否	パーミット	拒否	拒否
ACL は write と execute だけを許可	パーミット	パーミット	パーミット
ACL は write と execute を拒否	パーミット	拒否	拒否

ZFS ACL セット

ACL セットは ACL アクセス権の組み合わせで構成されます。これらの ACL セットは事前に定義されたものであり、変更することはできません。

ACL セット名	含まれる ACL アクセス権
full_set	すべてのアクセス権
modify_set	write_acl と write_owner を除くすべてのアクセス権
read_set	read_data、read_attributes、 read_xattr、および read_acl
write_set	write_data、append_data、write_attributes 、および write_xattr

個々のアクセス権を個別に設定せずに ACL セットを適用できます。たとえば、read_set ACL セットを Bob に付与すると、ACL だけでなく、ファイルコンテンツやそれらの基本属性と拡張された属性を読み取るアクセス権が付与されます。

```
# chmod A+user:bob:read_set:allow file.1
# ls -v file.1
-r--r--r--+  1 root     root      206695 Jul 20 13:43 file.1
0:user:bob:read_data/read_xattr/read_attributes/read_acl:allow
...
```

ACL 継承

ACL 継承では、新しく作成されたファイルまたはディレクトリで、親ディレクトリに対する既存のアクセス権を無視せずに継承することを意図された ACL を継承できます。

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。

注記 - 現在、`successful_access`、`failed_access`、および `inherited` フラグは SMB サーバーにのみ適用されます。

次の表は、オプションの継承フラグを示しています。

表 10 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`	親ディレクトリの ACL をそのディレクトリのファイルにのみ継承します。
`dir_inherit`	`d`	親ディレクトリの ACL をそのディレクトリのサブディレクトリにのみ継承します。
`inherit_only`	`i`	親ディレクトリから ACL を継承します。新しく作成したファイルまたはサブディレクトリだけに適用され、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。
`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。
-	N/A	アクセス権は付与されていません。
`successful_access`	`S`	正常にアクセスしたときに、アラームまたは監査記録を開始するかどうかを指定します。このフラグは監査またはアラームの ACE タイプで使用されます。
`failed_access`	`F`	アクセスに失敗したときに、アラームまたは監査記録を開始するかどうかを指定します。このフラグは監査またはアラームの ACE タイプで使用されます。
`inherited`	`I`	ACE が継承されたことを示します。

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。このプロパティーの詳細は、199 ページの「ACL プロパティー」を参照してください。

ZFS ファイルの ACL 継承の設定の詳細は、205 ページの「ZFS ファイルで ACL 継承を設定する」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 継承の特定の動作と、chmod 操作との ACL の関連を判定する ACL プロパティーが含まれています。これらのプロパティーは次のとおりです。

- aclinherit – ACL 継承の動作を判定します。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。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モードと等価です。
  - 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、ファイルは継承可能な ACE によって決定されるモードで作成されます。モードに影響を与える継承可能な ACE が存在しない場合、モードはアプリケーションから要求されたモードに従って設定されます。
  - passthrough-x – 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モードに影響を与える継承可能な ACE で実行アクセス権が設定されている場合に限りファイルが実行 (x) アクセス権付きで作成されます。

  aclinherit のデフォルトモードは、restricted です。

  aclinherit モードの詳細は、208 ページの「ACL 継承モードを使用した ACL 継承を変更する」を参照してください。
- aclmode – ファイルが最初に作成されたとき、または chmod の操作中に ACL をどのように変更するかを制御するときに ACL の動作を変更します。次の値を使用できます。
  - discard – ファイルのモードを表さない ACL エントリをすべて削除します。これがデフォルト値です。
  - mask – ユーザーまたはグループのアクセス権を減らします。アクセス権は、それが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である場合を除いて、グループアクセス権ビットと同程度にまで低下します。この場合、ACL アクセス権は、所有者のアクセス権ビットと同程度にまで削減されます。また、明示的な ACL セット操作が実行されていない場合、マスク値はモードが変更しても ACL を保持します。
  - passthrough – ファイルまたはディレクトリの新しいモードを表す必要な ACL エントリを生成する以外、ACL が変更されないことを示します。

  aclmode のデフォルトモードは、discard です。

aclmode プロパティーの使用に関する詳細は、例46「ACL プロパティーと変更された ACL アクセス権」を参照してください。

ZFS ファイルに ACL を設定する

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。
- 指定したユーザーがアクセスを要求したユーザーと一致する ACL エントリのみが処理されます。
- いったん付与した許可アクセス権は、その ACL アクセス権セットの後続の ACL 拒否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。

  拒否アクセス権が設定されている場合、またはアクセス権が失われている場合、ファイルの所有者またはスーパーユーザーに付与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。このメカニズムによって、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。

ACL の設定に関するコマンド構文

ACL を設定または変更するには、chmod コマンドを使用します。このコマンドの構文は、演算子 (+、=、または -) を入力する前に *A* を指定する点以外は、ファイルのアクセス権ビットを設定する構文に似ています。

- 簡易 ACL のコマンド構文

```
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 のコマンド構文

```
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
```

chmod コマンドは次の演算子を使用します。

- A+ は ACL エントリを追加します。
- A= は ACL エントリを置き換えます。

  ファイルの ACL 全体を置き換えるには、インデックス ID を指定せずにこの演算子を使用します。次の例では、file.1 の ACL エントリが削除され、everyone@ の単一エントリに置き換えられます。

  ```
  % chmod A=everyone@:read_data:allow file.1
  ```

- A- は ACL エントリを削除します。

  ファイルのすべての非簡易 ACL エントリを例外なく削除するには、この演算子を使用して削除する各エントリをリストせずにファイル名を指定します。

  ```
  % chmod A- filename
  ```

  このコマンド構文を使用して、ファイルに簡易 ACL を復元します。コマンドを発行すると、簡易 ACL を構成する owner@、group@、および everyone@ のエントリのみが残ります。

注意 - 既存の ACL の変更には注意が必要です。インデックスなしで演算子を使用するのと、インデックスを使用して演算子を使用するのとでは効果が異なります。たとえば、chmod A= は ACL 全体を置き換えますが、chmod A3= はインデックス番号 3 を持つ既存のエントリのみを置き換えます。

アクセス権と継承のフラグは、表8および表10に掲載されている一意の文字で表現されます。ZFS ACL を設定すると、それらのアクセス権に対応する文字を使用する (コンパクトモード) か、完全なアクセス権を入力する (冗長モード) ことができます。

この例では、両方のコマンドでユーザー Joe に file.1 での読み取りと実行のアクセス権を付与しています。

- chmod A+user:joe:rx:allow file.1
- chmod A+user:joe:read_data/execute:allow file.1

同様に、ユーザー John に新しく作成した dir.2 とそのファイルに対する継承可能な読み取り、書き込み、および実行のアクセス権を付与するには、次のいずれかのコマンドを使用できます。

- chmod A+user:john:rwx:fd:allow dir.2
- chmod A+user:john:read_data/write_data/execute:file_inherit/dir_inherit:allow dir.2

ACL 情報を表示する

ls コマンドでは、2 つの形式のいずれかで ACL 情報を表示できます。-v オプションを使用すると、全体または詳細形式でアクセス権が表示されます。-V オプションは、アクセス権とフラグを表す文字を使用してコンパクトな出力を生成します。

次の例は、同じ ACL 情報が冗長とコンパクトの両方の形式でどのように表示されるかを示しています。

```
ls -v file.1
-rw-r--r--   1 root     root      206695 Jul 20 14:27 file.1
0:owner@:read_data/write_data/append_data/read_attributes
/write_xattr/read_xattr/write_attributes/read_acl/write_acl
/write_owner/synchronize:allow
1:group@:read_data/read_attributes/read_xattr/read_acl
/synchronize:allow
2:everyone@:read_data/append_data/read_xattr/read_acl
/synchronize:allow

# ls -V file.1
-rw-r--r--   1 root     root      206695 Jul 20 14:27 file.1
owner@:rw-p--aARWcCos:-------:allow
group@:r-----a-R-c--s:-------:allow
everyone@:r-----a-R-c--s:-------:allow
```

各ユーザーカテゴリのアクセス権についての説明は、表8を参照してください。

ZFS ファイルの ACL を変更する

このセクションでは、ACL を設定および表示するサンプルコマンドを示します。

次の例では、write_data アクセス権が group@ に付与されます。group@ のインデックスは 1 です。

```
# chmod A1=group@:read_data/write_data:allow file.1
# ls -v file.1
-rw-rw-r--   1 root     root      206695 Jul 20 13:43 file.1
0:owner@:read_data/write_data/append_data/read_xattr/write_xattr
/read_attributes/write_attributes/read_acl/write_acl/write_owner
/synchronize:allow
1:group@:read_data/write_data:allow
2:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
:allow
```

次の例では、test.dir ディレクトリでの read_data/execute アクセス権が、ユーザー Joe に追加されます。

```
# chmod A0+user:joe:read_data/execute:allow test.dir
# ls -dv test.dir
drwxr-xr-x+  2 root     root           2 Jul 20 14:23 test.dir
0:user:joe:list_directory/read_data/execute:allow
1:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
/append_data/read_xattr/write_xattr/execute/delete_child
/read_attributes/write_attributes/read_acl/write_acl/write_owner
```

```
/synchronize:allow
2:group@:list_directory/read_data/read_xattr/execute/read_attributes
/read_acl/synchronize:allow
3:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
/read_acl/synchronize:allow
```

次の例では、ユーザー Joe のアクセス権が削除されます。

```
# chmod A0- test.dir
# ls -dv test.dir
drwxr-xr-x   2 root     root           2 Jul 20 14:23 test.dir
0:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
/append_data/read_xattr/write_xattr/execute/delete_child
/read_attributes/write_attributes/read_acl/write_acl/write_owner
/synchronize:allow
1:group@:list_directory/read_data/read_xattr/execute/read_attributes
/read_acl/synchronize:allow
2:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
/read_acl/synchronize:allow
```

アクセス権ビットとの ACL の関連

ZFS ファイルでは、UNIX アクセス権ビットは ACL エントリに対応しますが、キャッシュに格納されます。ファイルのアクセス権ビットを変更すると、ファイルの ACL がそれに応じて更新されます。同様に、ファイルの ACL を変更すると、アクセス権ビットも変更されます。

アクセス権ビットの詳細は、chmod(1)を参照してください。

次の例は、ファイルまたはディレクトリの ACL とアクセス権ビットの間の関係、および一方のアクセス権が変更されると他方にどのような影響を与えるかを示しています。

例 45 ACL およびアクセス権ビット

最初の例は、`file.2` の次の ACL で始まります。このアクセス権ビットは 644 に設定されています。

```
# ls -v file.2
-rw-r--r--   1 root     root        2693 Jul 20 14:26 file.2     アクセス権ビットは 644。
0:owner@:read_data/write_data/append_data/read_xattr/write_xattr
/read_attributes/write_attributes/read_acl/write_acl/write_owner
/synchronize:allow
1:group@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
2:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
:allow
```

`chmod` コマンドは、`everyone@` の ACL エントリを削除します。その結果、`everyone` の読み取りアクセス権ビットも削除され 640 に変更されます。

```
# chmod A2- file.2     everyone@ のアクセス権が削除される
# ls -v file.2
-rw-r-----   1 root     root        2693 Jul 20 14:26 file.2     アクセス権ビットは 640 になる。
```

```
0:owner@:read_data/write_data/append_data/read_xattr/write_xattr
/read_attributes/write_attributes/read_acl/write_acl/write_owner
/synchronize:allow
1:group@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
```

everyone@ エントリが削除される。

次に、ACL は everyone@ の read_data/write_data アクセス権に置き換えられます。owner と group のアクセス権をオーバーライドする owner@ または group@ の ACL エントリはありません。そのため、アクセス権ビットは 666 になります。

```
# chmod A=everyone@:rw:allow file.2
# ls -v file.2
-rw-rw-rw-   1 root     root        2440 Jul 20 14:28 file.3
```

アクセス権ビットは *666* になる。

```
0:everyone@:read_data/write_data:allow
```

その他すべての *ACL* エントリが削除される。

次に、ACL はユーザー Joe の読み取りアクセス権で置き換えられます。ただし、コマンドは簡易 ACL エントリを残しません。その結果、アクセス権ビットは 000 に設定され、file.2 への Joe のアクセスは拒否されます。ファイルは事実上、アクセス不能になります。

```
# chmod A=user:joe:r:allow file.2
# ls -v file.2
----------+  1 root     root        2440 Jul 20 14:28 file.3
```

アクセス権ビットは *000* になる。

```
0:user:joe:read_data:allow
```

簡易 *ACL* エントリは存在しない。

最後に、アクセス権ビットを設定すると、どのように ACL が更新されるかを示します。file.2 のビットは 655 に設定されます。デフォルトの簡易 ACL アクセス権が自動的に設定されます。

```
# chmod 655 file.2
# ls -v file.3
-rw-r-xr-x   1 root     root        2440 Jul 20 14:28 file.3
```

アクセス権ビットは *655* に設定される。

```
0:user:joe:read_data:allow
1:owner@:execute:deny
```

655 ビットに基づき、*ACL* 実行アクセス権は拒否される。

```
2:owner@:read_data/write_data/append_data/read_xattr/write_xattr
```

デフォルト *ACL* エントリが復元される。

```
/read_attributes/write_attributes/read_acl/write_acl/write_owner
/synchronize:allow
3:group@:read_data/read_xattr/execute/read_attributes/read_acl
/synchronize:allow
4:everyone@:read_data/read_xattr/execute/read_attributes/read_acl
/synchronize:allow
```

例 46 ACL プロパティーと変更された ACL アクセス権

次の例は、aclmode と aclinherit の特定のプロパティー値が ACL 動作にどのような影響を与えるかを示しています。これらのプロパティーが設定されている場合は、ファイルまたはディレクトリの ACL アクセス権は、所有グループに一致して縮小または拡張されます。

aclmode プロパティーは mask に設定され、aclinherit プロパティーはプールの restricted に設定されていると仮定します。また、元のファイルとグループ所有権および ACL 権限は次のようであるとします。

```
# zfs set aclmode=mask system1/data
# zfs set aclinherit=restricted system1/data

# ls -lV file.1
-rwxrwx---+  1 root     root      206695 Aug 30 16:03 file.1
user:amy:r-----a-R-c---:-------:allow
user:rory:r-----a-R-c---:-------:allow
group:sysadmin:rw-p--aARWc---:-------:allow
group:staff:rw-p--aARWc---:-------:allow
owner@:rwxp--aARWcCos:-------:allow
group@:rwxp--aARWc--s:-------:allow
everyone@:------a-R-c--s:-------:allow
```

これら 2 つのプロパティーに設定されている値の意味を理解するには、199 ページの「ACL プロパティー」を参照してください。

chown 操作は、`file.1` の所有権を Amy とグループ Staff に変更します。

```
# chown amy:staff file.1
```

Amy は、`file.1` のアクセス権ビットを 640 に変更します。以前に設定された ACL プロパティーのため、ACL 内のグループに対するアクセス権は所有する Staff のアクセス権を超えないように削減されます。

```
# su - amy
$ chmod 640 file.1
$ ls -lV file.1
-rw-r-----+  1 amy      staff     206695 Aug 30 16:03 file.1
user:amy:r-----a-R-c---:-------:allow
user:rory:r-----a-R-c---:-------:allow
group:sysadmin:r-----a-R-c---:-------:allow
group:staff:r-----a-R-c---:-------:allow
owner@:rw-p--aARWcCos:-------:allow
group@:r-----a-R-c--s:-------:allow
everyone@:------a-R-c--s:-------:allow
```

Amy はアクセス権ビットを 770 に変更します。その結果、ACL 内のグループのアクセス権も所有グループ Staff のアクセス権に一致するように変更されます。

```
$ chmod 770 file.1
$ ls -lV file.1
-rwxrwx---+  1 amy      staff     206695 Aug 30 16:03 file.1
user:amy:r-----a-R-c---:-------:allow
user:rory:r-----a-R-c---:-------:allow
group:sysadmin:rw-p--aARWc---:-------:allow
group:staff:rw-p--aARWc---:-------:allow
owner@:rwxp--aARWcCos:-------:allow
group@:rwxp--aARWc--s:-------:allow
everyone@:------a-R-c--s:-------:allow
```

ZFS ファイルで ACL 継承を設定する

ファイルとディレクトリに ACL をどのように継承するかを決定できます。

aclinherit プロパティーは、ファイルシステム上でグローバルに設定できます。デフォルトでは、aclinherit は restricted に設定されます。

詳細については、197 ページの「ACL 継承」を参照してください。

ファイルによって継承される ACL を付与する

このセクションでは、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 ACE を識別します。

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 joe 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:joe:read_data/write_data:file_inherit:allow test2.dir
# ls -dv test2.dir
drwxr-xr-x+  2 root     root           2 Jul 20 14:55 test2.dir
0:user:joe:read_data/write_data:file_inherit:allow
1:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
/append_data/read_xattr/write_xattr/execute/delete_child
/read_attributes/write_attributes/read_acl/write_acl/write_owner
/synchronize:allow
2:group@:list_directory/read_data/read_xattr/execute/read_attributes
/read_acl/synchronize:allow
3:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
/read_acl/synchronize:allow
```

次の例では、ユーザー joe 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 joe 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。

```
# touch test2.dir/file.2
# ls -v test2.dir/file.2
-rw-r--r--+  1 root     root           0 Jul 20 14:56 test2.dir/file.2
0:user:joe:read_data:inherited:allow
1:owner@:read_data/write_data/append_data/read_xattr/write_xattr
/read_attributes/write_attributes/read_acl/write_acl/write_owner
/synchronize:allow
2:group@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
3:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
:allow
```

このファイルシステムの aclinherit プロパティーがデフォルトモード restricted に設定されているため、ユーザー joe 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 joe 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のみ、everyone@ アクセス権に基づいて行われます。例:

```
# mkdir test2.dir/subdir.2
# ls -dv test2.dir/subdir.2
drwxr-xr-x+  2 root     root           2 Jul 20 14:57 test2.dir/subdir.2
0:user:joe:list_directory/read_data/add_file/write_data:file_inherit
/inherit_only/inherited:allow
1:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
/append_data/read_xattr/write_xattr/execute/delete_child
/read_attributes/write_attributes/read_acl/write_acl/write_owner
/synchronize:allow
2:group@:list_directory/read_data/read_xattr/execute/read_attributes
/read_acl/synchronize:allow
3:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
/read_acl/synchronize:allow
```

ファイルとディレクトリの両方によって継承される ACL を許可する

このセクションでは、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 を識別する例を示します。

次の例では、ユーザー joe に読み取り、書き込み、および実行アクセス権が付与されます。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:joe:read_data/write_data/execute:file_inherit/dir_inherit:allow
test3.dir
# ls -dv test3.dir
drwxr-xr-x+  2 root     root           2 Jul 20 15:00 test3.dir
0:user:joe:list_directory/read_data/add_file/write_data/execute
:file_inherit/dir_inherit:allow
1:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
/append_data/read_xattr/write_xattr/execute/delete_child
/read_attributes/write_attributes/read_acl/write_acl/write_owner
/synchronize:allow
2:group@:list_directory/read_data/read_xattr/execute/read_attributes
/read_acl/synchronize:allow
3:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
/read_acl/synchronize:allow
```

次の出力にある inherited というテキストは、ACE が継承されていることを示す情報メッセージです。

```
# touch test3.dir/file.3
# ls -v test3.dir/file.3
-rw-r--r--+  1 root     root           0 Jul 20 15:01 test3.dir/file.3
0:user:joe:read_data:inherited:allow
1:owner@:read_data/write_data/append_data/read_xattr/write_xattr
/read_attributes/write_attributes/read_acl/write_acl/write_owner
/synchronize:allow
2:group@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
3:everyone@:read_data/read_xattr/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ビットによって、書き込みアクセス権と実行アクセス権が拒否されます。このため、ユーザー joe は書き込みアクセス権と実行アクセス権が拒否されます。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。

次の例では、ユーザー joe に読み取り、書き込み、および実行アクセス権が付与されます。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が、ディレクトリの下位の内容には伝達されません。

```
# chmod A+user:joe:read_data/write_data/execute:file_inherit/no_propagate:allow
test4.dir
# ls -dv test4.dir
drwxr--r--+  2 root     root           2 Mar  1 12:11 test4.dir
0:user:joe:list_directory/read_data/add_file/write_data/execute
:file_inherit/no_propagate:allow
1:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
/append_data/read_xattr/write_xattr/execute/delete_child
/read_attributes/write_attributes/read_acl/write_acl/write_owner
/synchronize:allow
2:group@:list_directory/read_data/read_xattr/read_attributes/read_acl
/synchronize:allow
3:everyone@:list_directory/read_data/read_xattr/read_attributes/read_acl
/synchronize:allow
```

次の例で示すように、joe の read_data/write_data/execute アクセス権は、所有するグループのアクセス権に基づいて減少します。

```
# touch test4.dir/file.4
# ls -v test4.dir/file.4
-rw-r--r--+  1 root     root           0 Jul 20 15:09 test4.dir/file.4
0:user:joe:read_data:inherited:allow
1:owner@:read_data/write_data/append_data/read_xattr/write_xattr
/read_attributes/write_attributes/read_acl/write_acl/write_owner
/synchronize:allow
2:group@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
3:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
:allow
```

ACL 継承モードを使用した ACL 継承を変更する

このセクションでは、aclinherit プロパティーの値について説明します。

例 47 ACL 継承モードが discard に設定された ACL 継承

ファイルシステムの aclinherit 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ビットが変更されたときに、ACL が破棄される可能性があります。例:

```
# zfs set aclinherit=discard system1/cindy
# chmod A+user:joe:read_data/write_data/execute:dir_inherit:allow test5.dir
# ls -dv test5.dir
drwxr-xr-x+  2 root     root           2 Jul 20 14:18 test5.dir
```

```
0:user:joe:list_directory/read_data/add_file/write_data/execute
:dir_inherit:allow
1:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
/append_data/read_xattr/write_xattr/execute/delete_child
/read_attributes/write_attributes/read_acl/write_acl/write_owner
/synchronize:allow
2:group@:list_directory/read_data/read_xattr/execute/read_attributes
/read_acl/synchronize:allow
3: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ビット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。例:

```
# chmod 744 test5.dir
# ls -dv test5.dir
drwxr--r--   2 root     root           2 Jul 20 14:18 test5.dir
0:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
/append_data/read_xattr/write_xattr/execute/delete_child
/read_attributes/write_attributes/read_acl/write_acl/write_owner
/synchronize:allow
1:group@:list_directory/read_data/read_xattr/read_attributes/read_acl
/synchronize:allow
2:everyone@:list_directory/read_data/read_xattr/read_attributes/read_acl
/synchronize:allow
```

例 48 ACL 継承モードが `noallow` に設定された 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 system1/cindy
# chmod A+user:joe: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 Jul 20 14:22 test6.dir
0:user:joe:read_data:file_inherit:deny
1:user:lp:read_data:file_inherit:allow
2:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
/append_data/read_xattr/write_xattr/execute/delete_child
/read_attributes/write_attributes/read_acl/write_acl/write_owner
/synchronize:allow
3:group@:list_directory/read_data/read_xattr/execute/read_attributes
/read_acl/synchronize:allow
4: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
-rw-r--r--+  1 root     root           0 Jul 20 14:23 test6.dir/file.6
0:user:joe:read_data:inherited:deny
1:owner@:read_data/write_data/append_data/read_xattr/write_xattr
/read_attributes/write_attributes/read_acl/write_acl/write_owner
/synchronize:allow
```

```
2:group@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
3:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
:allow
```

ACL passthrough 継承モード

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。ファイルは継承可能な ACE によって決定されるアクセス権モードで作成されます。アクセス権モードに影響を与える継承可能な ACL が存在しない場合、アクセス権モードはアプリケーションから要求されたモードに従って設定されます。

例 49 ACL 継承モードが冗長モードの passthrough に設定された ACL 継承

system1/cindy ファイルシステムの aclinherit プロパティーが passthrough に設定されている場合は、ユーザー joe が新しく作成した file.5 には、test4.dir に適用されている ACL が継承されます。次に例を示します。

```
# zfs set aclinherit=passthrough system1/cindy
# touch test4.dir/file.5
# ls -v test4.dir/file.5
-rw-r--r--+  1 root     root           0 Jul 20 14:16 test4.dir/file.5
0:user:joe:read_data/write_data/execute:inherited:allow
1:owner@:read_data/write_data/append_data/read_xattr/write_xattr
/read_attributes/write_attributes/read_acl/write_acl/write_owner
/synchronize:allow
2:group@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
3:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
:allow
```

例 50 ACL 継承モードがコンパクトモードの passthrough に設定された ACL 継承

次の例では、コンパクト形式の ACL 構文を使用して、aclinherit モードを passthrough に設定することによってアクセス権ビットを継承する方法を示します。

次の例では、ACL を test1.dir 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。

```
# zfs set aclinherit=passthrough system1/cindy
# pwd
/system1/cindy
# 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 Jul 20 14:42 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 Jul 20 14:44 file.1
owner@:rwxpdDaARWcCos:------I:allow
group@:rwxp----------:------I:allow
everyone@:--------------:------I: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 Jul 20 14:45 subdir.1
owner@:rwxpdDaARWcCos:fd----I:allow
group@:rwxp----------:fd----I:allow
everyone@:--------------:fd----I:allow
```

`fd----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 を持つファイルが、継承される ACE が存在しない別のディレクトリに作成されます。

```
# cd /system1/cindy
# mkdir test2.dir
# cd test2.dir
# touch file.2
# ls -V file.2
-rw-r--r--   1 root     root           0 Jul 20 14:48 file.2
owner@:rw-p--aARWcCos:-------:allow
group@:r-----a-R-c--s:-------:allow
everyone@:r-----a-R-c--s:-------:allow
```

ACL 継承 `passthrough-x` モード

`aclinherit=passthrough-x` を有効にすると、ファイル作成モードおよびファイル作成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のみ、`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` モードを `passthrough-x` に設定して実行アクセス権を継承する方法を示します。

```
# zfs set aclinherit=passthrough-x system1/cindy
```

次の ACL は `/system1/cindy/test1.dir` で設定されており、`owner@` 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 Jul 20 14:50 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。

```
# touch test1.dir/file1
# ls -V test1.dir/file1
-rw-rw----+  1 root     root           0 Jul 20 14:52 test1.dir/file1
owner@:rw-pdDaARWcCos:------I:allow
group@:rw-p----------:------I:allow
everyone@:--------------:------I: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 Dec  3 15:19 t
owner@:rwxpdDaARWcCos:------I:allow
group@:rwxp----------:------I:allow
everyone@:--------------:------I: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。

特別な属性を ZFS ファイルに適用する

このセクションでは、ZFS ファイルに特別な属性を適用する方法、およびそれらを表示する方法を示します。特別な属性の表示および適用の詳細は、ls(1)およびchmod(1)のマニュアルページを参照してください。

不変性を ZFS ファイルに適用する

ファイルを変更できないようにするには、次の構文を使用します。

```
# chmod S+ci file.1
# echo this >>file.1
-bash: file.1: Not owner
# rm file.1
rm: cannot remove `file.1': Not owner
```

次の構文を使用することにより、ZFS ファイルに適用されている特別な属性を表示できます。

```
# ls -l/c file.1
```

```
-rw-r--r--+  1 root     root      206695 Jul 20 14:27 file.1
{A-----im----}
```

ファイルの不変性を削除するには、次の構文を使用します。

```
# chmod S-ci file.1
# ls -l/c file.1
-rw-r--r--+  1 root     root      206695 Jul 20 14:27 file.1
{A------m----}
# rm file.1
```

nounlink 属性による誤った削除を防止する

nounlink 属性は、ZFS 内のファイルまたはディレクトリが誤って削除されないように保護することによって、それらの不変性を補完します。ただし、immutable 属性とは異なり、nounlink はファイルが削除されたり、その名前が変更されたりしないようにするだけです。アプリケーションまたはユーザーは引き続き、このファイルを変更できます。

いくつかの例については、次のブログエントリを参照してください。

読み取り専用アクセス権を ZFS ファイルに適用する

次の構文は、読み取り専用アクセス権を ZFS ファイルに適用する方法を示しています。

```
# chmod S+cR file.2
# echo this >>file.2
-bash: file.2: Not owner
```

ZFS ファイル属性を表示し変更する

次の構文を使用して、特別な属性を表示し設定できます。

```
# ls -l/v file.3
-r--r--r--   1 root     root      206695 Jul 20 14:59 file.3
{archive,nohidden,noreadonly,nosystem,noappendonly,nonodump,
noimmutable,av modified,noav_quarantined,nonounlink,nooffline,nosparse}
# chmod S+cR file.3
# ls -l/v file.3
-r--r--r--   1 root     root      206695 Jul 20 14:59 file.3
{archive,nohidden,readonly,nosystem,noappendonly,nonodump,noimmutable,
av_modified,noav_quarantined,nonounlink,nooffline,nosparse}
```

これらの属性の一部は、Oracle Solaris SMB 環境だけで適用されます。

ファイルのすべての属性をクリアできます。例:

```
# chmod S-a file.3
# ls -l/v file.3
-r--r--r--   1 root     root      206695 Jul 20 14:59 file.3
{noarchive,nohidden,noreadonly,nosystem,noappendonly,nonodump,
noimmutable,noav_modified,noav_quarantined,nonounlink,nooffline,nosparse}
```

♦ ♦ ♦ 第 10 章

10 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、次のセクションで構成されます。

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。2 種類の委任アクセス権がサポートされています。

- `create`、`destroy`、`mount`、`snapshot` 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構成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。一般的な Oracle Solaris 管理タスクを管理するための RBAC の使用については、『Oracle Solaris 11.3 でのユーザーとプロセスのセキュリティー保護』 の 第 1 章,「権利を使用したユーザーとプロセスの制御について」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。例:

```
# 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 ファイルシステム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、現在のファイルシステムだけにローカルで委任するか、現在のファイルシステムのすべての子孫に委任できます。

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。
clone	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。
diff	データセット内のパスを識別するアクセス権。	ルート以外のユーザーが zfs diff コマンドを使用するには、このアクセス権が必要です。
hold	スナップショットを保持するアクセス権。	
mount	ファイルシステムのマウントとアンマウント、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。
release	スナップショットの保持を解放するアクセス権で、スナップショットが破棄される場合があります。	
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	ファイルシステムを共有および共有解除するアクセス権。	NFS 共有を作成するには、share と share.nfs の両方を持っていることが必要です。 SMB 共有を作成するには、share と share.smb の両方を持っていることが必要です。
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
- key
- keychange
- 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
- checksum
- compression
- copies
- dedup
- defaultgroupquota
- defaultuserquota
- devices
- encryption
- exec
- keysource
- logbias
- mountpoint
- nbmand
- normalization
- primarycache
- quota
- readonly
- recordsize
- refquota
- refreservation
- reservation
- rstchown
- secondarycache
- setuid
- shadow

- share.nfs
- share.smb
- snapdir
- sync
- 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、zfs(1M)を参照してください。

ZFS アクセス権の委任 (zfs allow)

zfs allow 構文は次のとおりです。

```
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、前のセクション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ファイルシステム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ファイルシステム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、このファイルシステム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`、`snapshot` といったアクセス権を次のように委任したとします。

```
# zfs allow cindy create,destroy,mount,snapshot system1/home/cindy
# zfs allow system1/home/cindy
---- Permissions on system1/home/cindy ---------------------------------
Local+descendant permissions:
user cindy 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 cindy system1/home/cindy
# zfs allow system1/home/cindy
```

ZFS アクセス権の委任の例

例 51 個別のユーザーにアクセス権を委任する

`create` および `mount` アクセス権を個人ユーザーに委任するときは、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 `mark` に `create` アクセス権と `mount` アクセス権を `system1` 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:mark:add_subdirectory:fd:allow /system1/home
```

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。例:

```
# zfs allow mark create,destroy,mount system1/home
```

これで、ユーザー mark は system1/home 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ます。例:

```
# su mark
mark$ zfs create system1/home/mark
mark$ ^D
# su lp
$ zfs create system1/home/lp
cannot create 'system1/home/lp': permission denied
```

例 52 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 system1/home 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 グループメンバーはほかのユーザーのファイルシステムを破棄できません。

```
# zfs allow staff create,mount system1/home
# zfs allow -c create,destroy system1/home
# zfs allow system1/home
---- Permissions on system1/home --------------------------------------
Create time permissions:
create,destroy
Local+descendant permissions:
group staff create,mount
# su cindy
cindy% zfs create system1/home/cindy/files
cindy% exit
# su mark
mark% zfs create system1/home/mark/data
mark% exit
cindy% zfs destroy system1/home/mark/data
cannot destroy 'system1/home/mark/data': permission denied
```

例 53 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 mark 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 mark には system1/home 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 mark snapshot system1/home
# zfs allow system1/home
---- Permissions on system1/home --------------------------------------
Create time permissions:
create,destroy
```

```
Local permissions:
user mark snapshot
Local+descendant permissions:
group staff create,mount
# su mark
mark$ zfs snapshot system1/home@snap1
mark$ zfs snapshot system1/home/mark@snap1
cannot create snapshot 'system1/home/mark@snap1': permission denied
```

ユーザー mark 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。例:

```
# zfs unallow -l mark snapshot system1/home
# zfs allow -d mark snapshot system1/home
# zfs allow system1/home
---- Permissions on system1/home --------------------------------------
Create time permissions:
create,destroy
descendant permissions:
user mark snapshot
Local+descendant permissions:
group staff create,mount
# su mark
$ zfs snapshot system1/home@snap2
cannot create snapshot 'system1/home@snap2': permission denied
$ zfs snapshot system1/home/mark@snappy
```

これで、ユーザー mark は system1/home 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ます。

例 54 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 system1/home ファイルシステムの作成後に委任されます。

```
# zfs allow staff create,mount system1/home
# zfs allow -c destroy,snapshot system1/home
# zfs allow system1/home
---- Permissions on system1/home --------------------------------------
Create time permissions:
create,destroy,snapshot
Local+descendant permissions:
group staff create,mount
```

ユーザー mark は staff グループのメンバーであるため、system1/home 内にファイルシステムを作成できます。また、ユーザー mark は、system1/home/mark2 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。例:

```
# su mark
$ zfs create system1/home/mark2
$ zfs allow system1/home/mark2
---- Permissions on system1/home/mark2 --------------------------------
Local permissions:
user mark create,destroy,snapshot
---- Permissions on system1/home --------------------------------------
```

```
Create time permissions:
create,destroy,snapshot
Local+descendant permissions:
group staff create,mount
```

ただし、ユーザー mark は system1/home/mark 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。例:

```
$ zfs snapshot system1/home/mark@snap1
cannot create snapshot 'system1/home/mark@snap1': permission denied
```

この例では、ユーザー mark 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。

```
$ cd /system1/home/mark2
$ ls
$ cd .zfs
$ ls
shares snapshot
$ cd snapshot
$ ls -l
total 3
drwxr-xr-x   2 mark    staff          2 Sep 27 15:55 snap1
$ pwd
/system1/home/mark2/.zfs/snapshot
$ mkdir snap2
$ zfs list
# zfs list -r system1/home
NAME                      USED  AVAIL  REFER  MOUNTPOINT
system1/home/mark          63K  62.3G    32K  /system1/home/mark
system1/home/mark2         49K  62.3G    31K  /system1/home/mark2
system1/home/mark2@snap1   18K      -    31K  -
system1/home/mark2@snap2     0      -    31K  -
$ ls
snap1  snap2
$ rmdir snap2
$ ls
snap1
```

例 55 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 system1 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 cindy は staff グループのメンバーであり、system1 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 system1 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 system1
# zfs allow system1
---- Permissions on system1 --------------------------------------------
Permission sets:
@myset clone,create,destroy,mount,promote,readonly,snapshot
# zfs allow staff @myset,rename system1
# zfs allow system1
---- Permissions on system1 --------------------------------------------
Permission sets:
@myset clone,create,destroy,mount,promote,readonly,snapshot
```

```
Local+descendant permissions:
group staff @myset,rename
# chmod A+group:staff:add_subdirectory:fd:allow system1
# su cindy
cindy% zfs create system1/data
cindy% zfs allow system1
---- Permissions on system1 --------------------------------------------
Permission sets:
@myset clone,create,destroy,mount,promote,readonly,snapshot
Local+descendant permissions:
group staff @myset,rename
cindy% ls -l /system1
total 15
drwxr-xr-x   2 cindy   staff          2 Jun 24 10:55 data
cindy% exit
# su lp
$ zfs create system1/lp
cannot create 'system1/lp': permission denied
```

ZFS 委任アクセス権の表示の例

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、次のコンポーネントが含まれています。

- 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 56 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 cindy が system1/cindy 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 system1/cindy
-------------------------------------------------------------
Local+descendant permissions on (system1/cindy)
user cindy create,destroy,mount,snapshot
```

例 57 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)
- 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。作成時は、ファイルシステム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ることを意味します。
- 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
---- Permissions on pool/fred --------------------------------------
Permission sets:
@eng create,destroy,snapshot,mount,clone,promote,rename
@simple create,mount
Create time permissions:
@eng,mountpoint
Local permissions:
user tom @eng
user joe create,destroy,mount
Local+descendant permissions:
user fred @basic,share,rename
user barney @basic
group staff @basic
---- Permissions on pool -------------------------------------------
Permission sets:
@simple create,destroy,mount
Local permissions:
group staff @simple
```

委任された ZFS アクセス権の削除の例

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 cindy は system1/cindy ファイルシステムに対して create、destroy、mount、および snapshot のアクセス権を持っています。

```
# zfs allow cindy create,destroy,mount,snapshot system1/home/cindy
# zfs allow system1/home/cindy
---- Permissions on system1/home/cindy --------------------------------
Local+descendant permissions:
user cindy create,destroy,mount,snapshot
```

次の `zfs unallow` 構文では、ユーザー `cindy` の `snapshot` アクセス権が `system1/home/cindy` ファイルシステムから削除されます。

```
# zfs unallow cindy snapshot system1/home/cindy
# zfs allow system1/home/cindy
---- Permissions on system1/home/cindy --------------------------------
Local+descendant permissions:
user cindy create,destroy,mount
cindy% zfs create system1/home/cindy/data
cindy% zfs snapshot system1/home/cindy@today
cannot create snapshot 'system1/home/cindy@today': permission denied
```

別の例として、ユーザー `mark` は `system1/home/mark` 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 system1/home/mark
---- Permissions on system1/home/mark --------------------------------
Local+descendant permissions:
user mark create,destroy,mount
-------------------------------------------------------------
```

次の `zfs unallow` 構文を使用すると、ユーザー `mark` のすべてのアクセス権が `system1/home/mark` ファイルシステムから削除されます。

```
# zfs unallow mark system1/home/mark
```

次の `zfs unallow` 構文では、`system1` 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 system1
---- Permissions on system1 --------------------------------------------
Permission sets:
@myset clone,create,destroy,mount,promote,readonly,snapshot
Create time permissions:
create,destroy,mount
Local+descendant permissions:
group staff create,mount
# zfs unallow -s @myset system1
# zfs allow system1
---- Permissions on system1 --------------------------------------------
Create time permissions:
create,destroy,mount
Local+descendant permissions:
group staff create,mount
```

♦ ♦ ♦ 第 11 章

11 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 Oracle 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。

この章の内容は次のとおりです。

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 `system1/vol` が作成されます。

```
# zfs create -V 5gb system1/vol
```

ボリュームのサイズを変更する場合は注意が必要です。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。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。そのため、ボリュームを作成すると、データの整合性を確保するために予約が自動的にそのボリュームの初期サイズに設定されます。

`zfs get` または `zfs get all` コマンドを使用して、ZFS ボリュームのプロパティー情報を表示できます。例:

```
# zfs get all system1/vol
```

`zfs get` 出力内の `volsize` に表示される疑問符 (?) は、入出力エラーが発生したために不明な値を示しています。例:

```
# zfs get -H volsize system1/vol
system1/vol        volsize ?       local
```

入出力エラーは通常、プールデバイスの問題を示しています。プールデバイスの問題の解決については、242 ページの「ZFS ストレージプールで発生した問題を識別する」を参照してください。

ゾーンがインストールされた Oracle Solaris システムを使用している場合は、ネイティブゾーン内で ZFS ボリュームを作成またはクローニングできません。

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のインストールまたは UFS ルートファイルシステムからの移行中に、ZFS ルートプール内の ZFS ボリューム上にスワップデバイスとダンプデバイスが作成されます。次の例は、スワップデバイスとダンプデバイスに関する情報を表示する方法を示しています。

```
# swap -l
swapfile                     dev    swaplo   blocks     free
/dev/zvol/dsk/rpool/swap   253,3        16  8257520  8257520

# dumpadm
Dump content: kernel pages
Dump device: /dev/zvol/dsk/rpool/dump (dedicated)
Savecore directory: /var/crash/
Savecore enabled: yes
```

システムがインストールされたあとにスワップ領域またはダンプデバイスを変更する必要がある場合は、以前の Oracle Solaris リリースと同様に `swap` および `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、新しいスワップデバイスのエントリを `/etc/vfstab` ファイルに追加します。例:

```
# 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、93 ページの「ZFS スワップおよびダンプデバイスのサイズ調整」を参照してください。

iSCSI LUN として ZFS ボリュームを使用する

iSCSI ターゲットとしての ZFS ボリュームは、ほかの ZFS データセットと同様に管理されます。ただし、ZFS ボリュームが iSCSI LUN として共有されている間、データセットの名前を変更したり、ボリュームスナップショットをロールバックしたり、プールをエクスポートすることはできません。これらの操作を実行しようとすると、次のようなメッセージが表示されます。

```
# zfs rename system1/volumes/v2 system1/volumes/v1
cannot rename 'system1/volumes/v2': dataset is busy
# zpool export system1
cannot export 'system1': pool is busy
```

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。

Common Multiprotocol SCSI Target (COMSTAR) ソフトウェアフレームワークを使用すると、あらゆる Oracle Solaris システムを、ストレージネットワークを介してイニシエータホストからアクセスできる SCSI ターゲットデバイスに変換できます。ZFS ボリュームを作成し、iSCSI 論理ユニット (LUN) として共有するように構成できます。

▼ ZFS ボリュームを iSCSI LUN として使用する方法

1. **まず、COMSTAR パッケージをインストールします。**

   ```
   # pkg install group/feature/storage-server
   ```

2. **iSCSI ターゲットとして使用される ZFS ボリュームを作成します。**
   例:

   ```
   # zfs create -V 2g system1/volumes/v2
   ```

3. **SCSI ブロックデバイスベースの LUN を作成します。**
   例:

   ```
   # sbdadm create-lu /dev/zvol/rdsk/system1/volumes/v2
   Created the following LU:

   GUID                    DATA SIZE           SOURCE
   --------------------------------  -------------------  ----------------
   600144f000144f1dafaa4c0faff20001  2147483648           /dev/zvol/rdsk/system1/volumes/v2
   # sbdadm list-lu
   Found 1 LU(s)

   GUID                    DATA SIZE           SOURCE
   --------------------------------  -------------------  ----------------
   600144f000144f1dafaa4c0faff20001  2147483648           /dev/zvol/rdsk/system1/volumes/v2
   ```

4. **すべての ZFS クライアントまたは選択された ZFS クライアントに LUN ビューを共有します。**

すべての ZFS クライアントまたは選択された ZFS クライアントのリストに LUN ビューを公開できます。次の例では、LUN ビューがすべての ZFS クライアントに共有されます。

a. LUN GUID を識別します。

```
# stmfadm list-lu
LU Name: 600144F000144F1DAFAA4C0FAFF20001
```

b. LUN ビューを共有します。

```
# stmfadm add-view 600144F000144F1DAFAA4C0FAFF20001
# stmfadm list-view -l 600144F000144F1DAFAA4C0FAFF20001
View Entry: 0
Host group   : All
Target group : All
LUN          : 0
```

5. iSCSI ターゲットを作成します。

iSCSI ターゲットの作成方法については、『Oracle Solaris 11.3 でのデバイスの管理』の 第 8 章, 「COMSTAR を使用したストレージデバイスの構成」を参照してください。

ゾーンがインストールされている Oracle Solaris システムで ZFS を使用する

Oracle Solaris オペレーティングシステムの Oracle™ Solaris ゾーン機能では、システム上にアプリケーションを実行するための隔離された環境が提供されます。以降のセクションでは、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。

- 231 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」。
- 232 ページの「データセットを非大域ゾーンに委任する」。
- 233 ページの「ZFS ボリュームを非大域ゾーンに追加する」。
- 233 ページの「ZFS ストレージプールをゾーンで使用する」。
- 234 ページの「ZFS プロパティーをゾーンで管理する」。
- 234 ページの「zoned プロパティーについて」。

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 クローンをネイティブ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ネイティブ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。

注記 - Oracle Solaris カーネルゾーンは、ネイティブな Oracle Solaris ゾーンとは異なる方法でストレージを使用します。カーネルゾーンでのストレージ使用の詳細は、`solaris-kz(5)` のマニュアルページのストレージアクセスのセクションを参照してください。

共有ストレージに対するストレージ使用については、『Oracle Solaris ゾーンの作成と使用』 の 第 13 章,「共有ストレージでの Oracle Solaris ゾーンの使用開始」を参照してください。

`fs` リソースを使用して ZFS ファイルシステムを追加すると、ネイティブゾーンが大域ゾーンまたはカーネルゾーンとディスク領域を共有できるようになります。ただし、ゾーン管理者はプロパティーを制御したり、ベースとなるファイルシステム階層内に新しいファイルシステムを作成したりできません。この操作は、ゾーンにほかの任意のタイプのファイルシステムを追加することと同じです。ネイティブゾーンへのファイルシステムの追加は、共通のディスク領域を共有するためにのみ行うようにしてください。

また、ZFS データセットをネイティブゾーンに委任することもでき、これによってゾーン管理者に対してデータセットおよびそのすべての子に完全な制御が付与されます。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ている最上位レベルの割り当て制限の超過を含め、ゾーンに追加されていないデータセットを操作することはできません。

ソース `zonepath` とターゲット `zonepath` の両方が 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は、`zfs clone` ではなく、`zoneadm clone` コマンドがゾーンをクローニングするためのコマンドになります。`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。詳細は、『Oracle Solaris ゾーンの作成と使用』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追加する

ネイティブゾーンに追加する 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。たとえば、`system1/zone/zion` ファイルシステムの場合は、大域ゾーンまたはカーネルゾーンで次のコマンドを入力します。

```
global# zfs set mountpoint=legacy system1/zone/zion
```

次に、`zonecfg` コマンドの `add fs` サブコマンドを使用して、そのファイルシステムをネイティブゾーンに追加します。

注記 - ファイルシステムを追加するには、それが以前に別の場所にマウントされていないことを確認してください。

```
global# zonecfg -z zion
zonecfg:zion> add fs
zonecfg:zion:fs> set type=zfs
zonecfg:zion:fs> set special=system1/zone/zion
zonecfg:zion:fs> set dir=/opt/data
zonecfg:zion:fs> end
```

この構文では、ZFS ファイルシステム system1/zone/zion がすでに構成済みの zion ゾーンに追加され、/opt/data にマウントされます。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。この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。同様に、ゾーン管理者は、このファイルシステムのプロパティー (atime、readonly、compression など) を変更できません。

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 コマンドおよび zonecfg でのリソースタイプの構成の詳細については、『Oracle Solaris ゾーンの作成と使用』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

ストレージ管理をゾーンに委任するというプライマリ目標を満たすために、ZFS は、zonecfg add dataset コマンドの使用を通じてネイティブゾーンへのデータセットの追加をサポートします。

次の例では、大域ゾーンまたはカーネルゾーンの大域ゾーン管理者によって、ZFS ファイルシステムがネイティブゾーンに委任されています。

```
global# zonecfg -z zion
zonecfg:zion> add dataset
zonecfg:zion:dataset> set name=system1/zone/zion
zonecfg:zion:dataset> set alias=system1
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 system1/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。zion ゾーン内では、このファイルシステムは system1/zone/zion としてアクセスできませんが、system1 という名前の仮想プールとしてアクセスできます。委任されるファイルシステムの別名は、仮想プールとして、元のプールのビューをゾーンに提供します。別名プロパティーは、仮想プールの名前を指定します。別名が指定されていない場合、ファイルシステム名の最後のコンポーネントに一致するデフォルトの別名が使用されます。 この例では、デフォルトの別名は zion になります。

委任されたデータセット内で、ゾーン管理者はファイルシステム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。委任されたファイルシステム内に ZFS ボリュームが作成された場合、

これらのボリュームは、デバイスリソースとして追加される ZFS ボリュームと競合する可能性があります。

ZFS ボリュームを非大域ゾーンに追加する

次の方法で、ネイティブゾーンで ZFS ボリュームを追加または作成したり、またはネイティブゾーン内のボリュームのデータへのアクセスを追加したりできます。

- ネイティブゾーンでは、特権ゾーン管理者は、以前に委任されたファイルシステムの子孫として ZFS ボリュームを作成できます。たとえば、前の例で委任されたファイルシステム `system1/zone/zion` の場合は、次のコマンドを入力できます。

  ```
  # zfs create -V 2g system1/zone/zion/vol1
  ```

  このボリュームが作成されたあと、ゾーン管理者はネイティブゾーンでそのボリュームのプロパティーやデータを管理したり、スナップショットを作成したりできます。
- 大域ゾーンまたはカーネルゾーンで `zonecfg add device` コマンドを使用して、ネイティブゾーンでアクセスできるデータを含んだ ZFS ボリュームを指定します。例:

  ```
  global# zonecfg -z zion
  zonecfg:zion> add device
  zonecfg:zion:device> set match=/dev/zvol/dsk/system1/volumes/vol2
  zonecfg:zion:device> end
  ```

  この例では、ネイティブゾーンでボリュームデータにのみアクセスできます。

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ネイティブ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、大域ゾーンまたはカーネル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ネイティブ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ネイティブ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ネイティブゾーンの内部から実行することはできません。`zonecfg add device` コマンドを使用して物理デバイスをネイティブ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ネイティブ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。

データストレージ管理の点から言えば、カーネルゾーンの方が強力で高い柔軟性を備えています。大域ゾーンのように、デバイスおよびボリュームをカーネルゾーンに委任できます。また、ZFS ストレージプールをカーネルゾーンの内部で作成できます。

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。データセットがゾーンに委任されると、そ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
system1
system1/home
system1/data
system1/data/matrix
system1/data/zion
system1/data/zion/home
```

system1/data/zion がデフォルト zion エイリアスでゾーンに追加された場合には、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
system1	いいえ	-	-
system1/home	いいえ	-	-
system1/data	いいえ	-	-
system1/data/zion	はい	はい	zoned、quota、reservation
system1/data/zion/home	はい	はい	zoned

system1/zone/zion のすべての親は表示できず、すべての子孫は書き込み可能です。zoned プロパティーを変更すると、次のセクションで説明するセキュリティーリスクにさらされるため、ゾーン管理者はこのプロパティーを変更できません。

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、ネイティブ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、データセットをネイティブゾーンに委任したあとに、大域ゾーン管理者が share.nfs 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

データセットがネイティブゾーンに委任されると、特定のプロパティーが大域ゾーンまたはカーネルゾーンのコンテキスト内で解釈されないようにするために、データセットを特別にマーク付けする必要があります。データセットがネイティブゾーン

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。すべてのファイルシステムと同様に、大域ゾーンまたはカーネルゾーンのセキュリティーに悪影響を与える可能性のある setuid バイナリ、シンボリックリンク、またはそれ以外の疑わしい内容が存在する場合があります。また、mountpoint プロパティーは、大域ゾーンまたはカーネルゾーンのコンテキストでは解釈できません。それ以外に、ゾーン管理者が大域ゾーンまたはカーネルゾーンの名前空間に影響を及ぼす可能性もあります。後者に対応するために、ZFS は、zoned プロパティーを使用して、データセットがある時点でネイティブゾーンに委任されたことを示します。

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ブート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またはカーネル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、system1/zone/zion はゾーンに委任されていますが、system1/zone/global は追加されていません。

```
# zfs list -o name,zoned,mountpoint -r system1/zone
NAME                  ZONED   MOUNTPOINT              MOUNTED
system1/zone/global     off   /system1/zone/global        yes
system1/zone/zion        on   /system1/zone/zion          yes
# zfs mount
system1/zone/global           /system1/zone/global
system1/zone/zion             /export/zone/zion/root/system1/zone/zion


root@kzx-05:~# zonecfg -z sol info dataset
dataset:
    name: rpool/foo
    alias: foo
root@kzx-05:~# zfs list -o name,zoned,mountpoint,mounted -r rpool/foo
NAME       ZONED  MOUNTPOINT                  MOUNTED
rpool/foo     on  /system/zones/sol/root/foo      yes
root@kzx-05:~# zfs mount | grep /foo
rpool/foo                       /system/zones/sol/root/foo
```

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クリアされることはありません。この動作により、これらのタスクに関連する固有のセキュリティーリスクが回避されます。信頼されないユーザーがデータセットやその子孫への完全なアクセス権を持っているため、mountpoint プロパティーが不正な値に設定されたり、ファイルシステム上に 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クリア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ことは、データセットがネイティブ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。

ほかのシステムにゾーンをコピーする

1 つ以上のゾーンを別のシステムに移行する必要がある場合は、オペレーティングシステム内のすべてのクローニングおよび回復操作を管理し、大域ゾーン、ネイティブゾーン、およびカーネルゾーンで動作する Oracle Solaris 統合アーカイブを使用します。統合アーカイブの詳細は、『Oracle Solaris 11.3 でのシステム復旧とクローン』を参照してください。ゾーンの移行 (ほかのシステムへのゾーンのコピーを含む) に関する手順については、『Oracle Solaris ゾーンの作成と使用』 の 第 7 章,「Oracle Solaris ゾーンの移行および変換」を参照してください。

あるシステム上のすべてのゾーンを別のシステム上の別の ZFS プールに移動する必要がある場合は、スナップショットやクローンが保持されるという理由から、レプリケーションストリームの使用を検討してください。スナップショットとクローンは、pkg update、beadm create、および zoneadm clone コマンドで幅広く使用されます。

次の例では、sysA のゾーンが rpool/zones ファイルシステムにインストールされており、それらを sysB 上の newpool/zones ファイルシステムにコピーする必要があります。次のコマンドは、スナップショットを作成し、複製ストリームを使用してデータを sysB にコピーします。

```
sysA# zfs snapshot -r rpool/zones@send-to-sysB
sysA# zfs send -R rpool/zones@send-to-sysB | ssh sysB zfs receive -d newpool
```

注記 - これらのコマンドは、操作の ZFS の側面のみを示しています。このタスクを完了するには、ゾーン関連のほかのコマンドを実行する必要があります。詳細は、『Oracle Solaris ゾーンの作成と使用』 の 第 7 章,「Oracle Solaris ゾーンの移行および変換」を参照してください。

代替ルート場所で ZFS プールを使用する

プールは、本質的にホストシステムに結び付けられています。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、代替メディアからブート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

ルート場所」プール機能が用意されています。代替ルートプール場所はシステムのリブート後は維持されません。すべてのマウントポイントはプールのルートに相対的であるように変更されます。

代替のルート場所で ZFS プールを作成する

代替場所で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プールが作成されると、ルートファイルシステムのマウントポイントは、代替ルートの値に相当する `/` 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場所としての `/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場所 `/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。そのあと、このファイルシステムをエクスポートし、`-R` *alternate-root-value* 構文を使用して別のシステム上の任意の代替ルート場所の下にインポートできます。

```
# 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
```

代替ルート場所でプールをインポートする

代替ルート場所を使って、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マウントポイン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。前のセクションで説明したとおり、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マウントポイントとしての /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
```

一時的な名前でプールをインポートする

代替ルート場所でのプールのインポートに加えて、一時的な名前でプールをインポートできます。特定の共有ストレージまたは回復状況では、この機能により、同じ永続的な名前を持つ 2 つのプールを同時にインポートできます。これらのプールのいずれかを一時的な名前でインポートする必要があります。

次の例で、rpool プールは、代替ルート場所で一時的な名前を使用してインポートされます。永続的なプール名がすでにインポートされているプールと競合するため、プール ID によって、またはデバイスを指定してインポートする必要があります。

```
# zpool import
pool: rpool
id: 16760479674052375628
state: ONLINE
action: The pool can be imported using its name or numeric identifier.
config:

rpool     ONLINE
c8d1s0  ONLINE
# zpool import -R /a -t altrpool 16760479674052375628
# zpool list
NAME      SIZE  ALLOC  FREE  CAP  DEDUP  HEALTH  ALTROOT
altrpool   97G  22.4G   74G  23%  1.00x  ONLINE  /a
rpool     465G  75.1G  390G  16%  1.00x  ONLINE  -
```

zpool create -t オプションを使用して一時的な名前を持つプールを作成することもできます。

♦ ♦ ♦ 第 12 章

12 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、次のセクションで構成されます。

ルートプールの完全な回復については、『Oracle Solaris 11.3 でのシステム復旧とクローン』を参照してください。

ZFS の問題の識別

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、一般的なハードウェア障害を診断する方法、およびプールデバイスとファイルシステムの問題を解決する方法について概要を説明します。次の種類の問題が発生する可能性があります。

- **一般的なハードウェアの問題** – ハードウェアの問題は、プールのパフォーマンスやプールデータの可用性に影響を与える可能性があります。プールやファイルシステムなどの上位レベルの問題を特定する前に、障害のあるコンポーネントやメモリーなどの一般的なハードウェアの問題を排除してください。
- **ZFS ストレージプールの問題**
  - 242 ページの「ZFS ストレージプールで発生した問題を識別する」。

    - 247 ページの「ZFS ストレージデバイスの問題を解決する」。
- **データが破損している** – 262 ページの「ZFS ストレージプール内のデータの問題を解決する」。
- **構成が破損している** – 274 ページの「損傷した ZFS 構成を修復する」。
- **システムがブートしない** – 274 ページの「ブートできないシステムを修復する」。

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。

一般的なハードウェアの問題を解決する

プールの問題やファイルシステムの使用不能がハードウェアの問題 (障害のあるシステムボード、メモリー、デバイス、HBA、または構成ミスなど) に関連しているかどうかを判定するには、次のセクションを確認してください。

たとえば、ビジー状態の ZFS プール上にエラーや障害の発生したディスクがあると、システム全体のパフォーマンスが低下します。

最初に簡単に検出できるハードウェアの問題を診断して特定し、すべてのハードウェアを確認すれば、この章で説明するプールおよびファイルシステムの問題の診断に進むことができます。ハードウェア、プール、およびファイルシステムの構成に問題がない場合は、一般により複雑で解明しにくく、このガイドでは取り上げていないアプリケーションの問題を診断することを検討してください。

ハードウェアおよびデバイスの障害を識別する

Oracle Solaris Fault Manager は、エラーログ内の具体的な兆候を示すエラー遠隔監視情報を特定し、エラーの兆候が実際の障害になったときに実際の障害診断を報告することにより、ソフトウェア、ハードウェア、および特定のデバイスの問題を追跡します。

次のコマンドは、ソフトウェアまたはハードウェア関連の障害を特定します。

```
# fmadm faulty
```

障害が発生したサービスまたはデバイスを特定するには、上記のコマンドを定期的に使用します。

ハードウェアまたはデバイス関連のエラーを特定するには、次のコマンドを定期的に使用します。

```
# fmdump -eV | more
```

このログファイルのエラーメッセージは、vdev.open_failed、checksum、または io_failure の問題を示しており、fmadm 障害コマンドで表示される実際の障害に発展する可能性があるため、注意が必要です。

上記によってデバイスに障害が発生していることが示された場合は、交換用デバイスが用意されているかどうかを確認することをお勧めします。

iostat コマンドを使用して、その他のデバイスエラーを追跡することもできます。エラー統計のサマリーを確認するには、次の構文を使用します。

```
# iostat -en
---- errors ---
s/w h/w trn tot device
0   0   0   0 c0t5000C500335F95E3d0
0   0   0   0 c0t5000C500335FC3E7d0
0   0   0   0 c0t5000C500335BA8C3d0
0  12   0  12 c2t0d0
0   0   0   0 c0t5000C500335E106Bd0
0   0   0   0 c0t50015179594B6F11d0
0   0   0   0 c0t5000C500335DC60Fd0
0   0   0   0 c0t5000C500335F907Fd0
0   0   0   0 c0t5000C500335BD117d0
```

上記の出力では、内部ディスク c2t0d0 のエラーが報告されています。より詳細なデバイスエラーを表示するには、次の構文を使用します。

永続的または一時的なトランスポートエラーを解決する

再試行またはリセットについて言及する永続的な SCSI トランスポートエラーは、ダウン改訂ファームウェア、不良ディスク、不良ケーブル、または障害が発生したるハードウェア接続によって引き起こされる可能性があります。一部の一時的なトランスポートエラーは、HBA またはデバイスファームウェアをアップグレードすることで解決することがあります。トランスポートエラーがファームウェアの更新後も持続し、すべてのデバイスが作動していると思われる場合は、ハードウェアコンポーネント間に不良ケーブルまたはほかの障害が発生した接続がないか調べます。

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。次のシナリオは通知イベントを生成します。

- **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。

ZFS ストレージプールで発生した問題を識別する

次のセクションでは、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。

- 243 ページの「ZFS ストレージプールに問題があるかどうかを確認する」。
- 244 ページの「ZFS ストレージプールのステータス情報を確認する」。
- 241 ページの「ZFS エラーメッセージのシステムレポート」。

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。
- 間違って破棄された ZFS ストレージプールは、`zpool import -D` コマンドを使用して回復できますが、デバイスが再利用または間違って上書きされないよう、プールが迅速に回復されることが重要です。詳細は、75 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 ファイルシステムまたはデータを回復するための同様の機能は存在しません。常に適切なバックアップを用意しておいてください。

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。ほとんどの場合、fmd メッセージは、回復に関するさらなる指示について zpool status コマンドを案内します。

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。例:

  ```
  # zpool history system1
  History for 'system1':
  2012-11-12.13:01:31 zpool create system1 mirror c0t1d0 c0t2d0 c0t3d0
  2012-11-12.13:28:10 zfs create system1/eric
  2012-11-12.13:37:48 zfs set checksum=off system1/eric
  ```

  この出力では、system1/eric 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。
  - 使用できないデバイスまたは見つからないデバイスを交換して、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)。

このセクションでは、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。以降のセクションでは、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 フラグがないと、コマンドは、プールがほかの点では健全な場合でも、すべてのプール (または、コマンド行で指定された場合は、要求されたプール) の完全なステータスを表示します。

`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、60 ページの「ZFS ストレージプールのステータスのクエリー検索を行う」を参照してください。

ZFS ストレージプールのステータス情報を確認する

`zpool status` コマンドを使用すると、ZFS ストレージプールのステータス情報が表示されます。例:

```
# zpool status pond
pool: pond
state: DEGRADED
status: One or more devices are unavailable in response to persistent errors.
Sufficient replicas exist for the pool to continue functioning in a
degraded state.
action: Determine if the device needs to be replaced, and clear the errors
using 'zpool clear' or 'fmadm repaired', or replace the device
with 'zpool replace'.
Run 'zpool status -v' to see device specific details.
scan: scrub repaired 0 in 0h0m with 0 errors on Wed Jun 20 13:16:09 2012
config:

NAME                     STATE     READ WRITE CKSUM
pond                     DEGRADED     0     0     0
mirror-0                 ONLINE       0     0     0
c0t5000C500335F95E3d0  ONLINE       0     0     0
c0t5000C500335F907Fd0  ONLINE       0     0     0
mirror-1                 DEGRADED     0     0     0
c0t5000C500335BD117d0  ONLINE       0     0     0
c0t5000C500335DC60Fd0  UNAVAIL      0     0     0

errors: No known data errors
```

この出力については、次のセクションで説明します。

プールの全般的なステータス情報

`zpool status` 出力のこのセクションには次のフィールドが含まれます。その一部は、問題が発生しているプールに対してのみ表示されます。

`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ステータスが出力されます。前回のスクラブが完了した日付と時間、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。

ZFS ストレージプールの構成情報

`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、デバイスのステータス、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`、または `SUSPENDED`。`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、251 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 UNAVAIL の場合、このフィールド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を示しています。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モニターする方法の詳細については、260 ページの「再同期化のステータスを表示する」を参照してください。

ZFS ストレージプールのスクラブのステータス

zpool status 出力のスクラブセクションには、すべてのスクラブ操作の現在のステータスが説明されます。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。

次の zpool status スクラブステータスメッセージが表示されます。

- スクラブ進捗レポート。例:

```
scan: scrub in progress since Wed Jun 20 14:56:52 2012
529M scanned out of 71.8G at 48.1M/s, 0h25m to go
0 repaired, 0.72% done
```

- スクラブ完了メッセージ。例:

```
scan: scrub repaired 0 in 0h11m with 0 errors on Wed Jun 20 15:08:23 2012
```

- 進行中のスクラブの取り消しメッセージ。例:

```
scan: scrub canceled on Wed Jun 20 16:04:40 2012
```

スクラブ完了メッセージはシステムのリブート後も残ります。

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、266 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。

ZFS 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。例:

```
# zpool status -v system1
pool: system1
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://support.oracle.com/msg/ZFS-8000-8A
scan: scrub repaired 0 in 0h0m with 2 errors on Fri Jun 29 16:58:58 2012
config:

NAME           STATE     READ WRITE CKSUM
system1        ONLINE       2     0     0
   c8t0d0      ONLINE       0     0     0
   c8t1d0      ONLINE       2     0     0

errors: Permanent errors have been detected in the following files:

/system1/file.1
```

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、270 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 ストレージデバイスの問題を解決する

見つからないデバイス、削除されたデバイス、または障害が発生したデバイスを解決するには、以降のセクションを確認してください。

見つからないデバイスまたは削除されたデバイスを解決する

デバイスを開けない場合には、`zpool status` の出力に `UNAVAIL` ステータスが表示されます。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。失敗したために UNAVAIL であるデバイスを置き換える必要がある場合は、254 ページの「ZFS 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。

デバイスがルートプールまたはミラー化ルートプール内で UNAVAIL の場合は、次を参照してください。

- **ミラー化ルートプールディスクで障害が発生した** – 95 ページの「代替ルートプールディスクからのブート」
- **ルートプール内でディスクを交換する**
  - 87 ページの「ZFS ルートプールのディスクを交換する方法」
  - 87 ページの「ZFS ルートプールのディスクを交換する方法」
- **完全なルートプール障害回復** – 『Oracle Solaris 11.3 でのシステム復旧とクローン』

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-QJ, TYPE: Fault, VER: 1, SEVERITY: Minor
EVENT-TIME: Wed Jun 20 13:09:55 MDT 2012
...
SOURCE: zfs-diagnosis, REV: 1.0
EVENT-ID: e13312e0-be0a-439b-d7d3-cddaefe717b0
DESC: Outstanding dtls on ZFS device 'id1,sd@n5000c500335dc60f/a' in pool 'pond'.
AUTO-RESPONSE: No automated response will occur.
IMPACT: None at this time.
REC-ACTION: Use 'fmadm faulty' to provide a more detailed view of this event.
Run 'zpool status -lx' for more information. Please refer to the associated
reference document at http://support.oracle.com/msg/ZFS-8000-QJ for the latest
service procedures and policies regarding this diagnosis.
```

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 -v コマンドを使用します。例:

```
# zpool status -v
pool: pond
state: DEGRADED
status: One or more devices are unavailable in response to persistent errors.
Sufficient replicas exist for the pool to continue functioning in a
degraded state.
action: Determine if the device needs to be replaced, and clear the errors
using 'zpool clear' or 'fmadm repaired', or replace the device
with 'zpool replace'.
scan: scrub repaired 0 in 0h0m with 0 errors on Wed Jun 20 13:16:09 2012
config:

NAME                    STATE     READ WRITE CKSUM
pond                    DEGRADED     0     0     0
mirror-0                ONLINE       0     0     0
c0t5000C500335F95E3d0   ONLINE       0     0     0
c0t5000C500335F907Fd0   ONLINE       0     0     0
mirror-1                DEGRADED     0     0     0
c0t5000C500335BD117d0   ONLINE       0     0     0
```

```
c0t5000C500335DC60Fd0  UNAVAIL      0     0     0

device details:

c0t5000C500335DC60Fd0    UNAVAIL          cannot open
status: ZFS detected errors on this device.
The device was missing.
see: http://support.oracle.com/msg/ZFS-8000-LR for recovery
```

この出力から、c0t5000C500335DC60Fd0 デバイスが機能していないことがわかります。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。

必要に応じて、zpool online コマンドを使用して、交換したデバイスをオンラインにします。例:

```
# zpool online pond c0t5000C500335DC60Fd0
```

fmadm faulty の出力でデバイスエラーが特定された場合に、デバイスが交換されていることを FMA に知らせます。例:

```
# fmadm faulty
--------------- ------------------------------------  -------------- ---------
TIME            EVENT-ID                              MSG-ID         SEVERITY
--------------- ------------------------------------  -------------- ---------
Jun 20 13:15:41 3745f745-371c-c2d3-d940-93acbb881bd8  ZFS-8000-LR    Major

Problem Status    : solved
Diag Engine       : zfs-diagnosis / 1.0
System
Manufacturer  : unknown
Name          : ORCL,SPARC-T3-4
Part_Number   : unknown
Serial_Number : 1120BDRCCD
Host_ID       : 84a02d28

----------------------------------------
Suspect 1 of 1 :
Fault class : fault.fs.zfs.open_failed
Certainty   : 100%
Affects     : zfs://pool=86124fa573cad84e/
  vdev=25d36cd46e0a7f49/pool_name=pond/
  vdev_name=id1,sd@n5000c500335dc60f/a
Status      : faulted and taken out of service

FRU
Name             : "zfs://pool=86124fa573cad84e/
vdev=25d36cd46e0a7f49/pool_name=pond/
vdev_name=id1,sd@n5000c500335dc60f/a"
Status        : faulty

Description : ZFS device 'id1,sd@n5000c500335dc60f/a'
in pool 'pond' failed to open.

Response    : An attempt will be made to activate a hot spare if available.

Impact      : Fault tolerance of the pool may be compromised.

Action      : Use 'fmadm faulty' to provide a more detailed view of this event.
Run 'zpool status -lx' for more information. Please refer to the
associated reference document at
http://support.oracle.com/msg/ZFS-8000-LR for the latest service
procedures and policies regarding this diagnosis.
```

fmadm faulty の出力から Affects: セクション内の文字列を取り出し、次のコマンドでそれを含めてデバイスが交換されたことを FMA に知らせます。

```
# fmadm repaired zfs://pool=86124fa573cad84e/ \
   vdev=25d36cd46e0a7f49/pool_name=pond/ \
   vdev_name=id1,sd@n5000c500335dc60f/a
fmadm: recorded repair to of zfs://pool=86124fa573cad84e/
   vdev=25d36cd46e0a7f49/pool_name=pond/vdev_
name=id1,sd@n5000c500335dc60f/a
```

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。例:

```
# zpool status -x system1
pool 'system1' is healthy
```

削除されたデバイスを解決する

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。次の状況では、プールは UNAVAIL になる場合があります。つまり、デバイスが再接続されるまでデータにはアクセスできません。

冗長ストレージプールデバイスが間違って取り外されて挿入し直された場合は、ほとんどの場合、デバイスエラーを単にクリアできます。例:

```
# zpool clear system1 c1t1d0
```

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。251 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。

デバイスの接続が危殆化している場合、プールは SUSPENDED になる可能性があります。デバイスの問題が解決されるまで、SUSPENDED プールの状態は wait のままです。例:

```
# zpool status cybermen
```

```
pool: cybermen
state: SUSPENDED
status: One or more devices are unavailable in response to IO failures.
The pool is suspended.
action: Make sure the affected devices are connected, then run 'zpool clear' or
'fmadm repaired'.
Run 'zpool status -v' to see device specific details.
see: http://support.oracle.com/msg/ZFS-8000-HC
scan: none requested
config:

NAME        STATE     READ WRITE CKSUM
cybermen    UNAVAIL      0    16     0
c8t3d0      UNAVAIL      0     0     0
c8t1d0      UNAVAIL      0     0     0
```

デバイスの接続が復元されたあとで、プールまたはデバイスのエラーを解消します。

```
# zpool clear cybermen
# fmadm repaired zfs://pool=name/vdev=guid
```

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。プールが以前 UNAVAIL または SUSPENDED だった場合、または attach 手順の一環としてシステムがリブートされた場合、ZFS は、プールを開くときにすべてのデバイスを自動的に再スキャンします。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。例:

```
# zpool online system1 c0t1d0
```

デバイスをオンラインする方法の詳細については、48 ページの「ストレージプールのデバイスをオフラインにするまたはオンラインに戻す」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

このセクションでは、デバイスの障害の種類を確認し、一時的なエラーをクリアし、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

破損したデバイスという用語は、どちらかといえばあいまいで、多数の可能性のある状況を表す場合があります。

- **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。例:

```
# zpool status -v system1
pool: system1
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://support.oracle.com/msg/ZFS-8000-8A
config:

NAME           STATE     READ WRITE CKSUM
system1        ONLINE       2     0     0
   c8t0d0      ONLINE       0     0     0
   c8t0d0      ONLINE       2     0     0

errors: Permanent errors have been detected in the following files:

/system1/file.1
```

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ます。これらの修復手順については、次のセクションで説明します。

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、269 ページの「破損した ZFS データを修復する」を参照してください。

一時的または永続的なデバイスエラーをクリアする

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クリアするには、zpool clear コマンドを使用します。例:

```
# zpool clear system1 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クリアされます。

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クリアし、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クリアするには、次の構文を使用します。

```
# zpool clear system1
```

プールエラーのクリアの詳細については、49 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。

一時的なデバイスエラーは、ほとんどの場合、zpool clear コマンドを使用してクリアされます。デバイスで障害が発生した場合は、デバイスの置換に関する次のセクションを参照してください。冗長デバイスが間違って上書きされた、または長時間 UNAVAIL であった場合は、zpool status 出力で指示されているとおりに fmadm repaired コマンドを使用してこのエラーを解決する必要がある場合があります。例:

```
# zpool status -v pond
pool: pond
state: DEGRADED
status: One or more devices are unavailable in response to persistent errors.
Sufficient replicas exist for the pool to continue functioning in a
degraded state.
action: Determine if the device needs to be replaced, and clear the errors
using 'zpool clear' or 'fmadm repaired', or replace the device
with 'zpool replace'.
scan: scrub repaired 0 in 0h0m with 0 errors on Wed Jun 20 15:38:08 2012
config:

NAME                    STATE     READ WRITE CKSUM
pond                    DEGRADED     0     0     0
mirror-0                DEGRADED     0     0     0
c0t5000C500335F95E3d0   ONLINE       0     0     0
c0t5000C500335F907Fd0   UNAVAIL      0     0     0
mirror-1                ONLINE       0     0     0
c0t5000C500335BD117d0   ONLINE       0     0     0
c0t5000C500335DC60Fd0   ONLINE       0     0     0

device details:

c0t5000C500335F907Fd0    UNAVAIL          cannot open
status: ZFS detected errors on this device.
The device was missing.
see: http://support.oracle.com/msg/ZFS-8000-LR for recovery


errors: No known data errors
```

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。

- 254 ページの「デバイスを置き換えられるかどうかを確認する」。
- 255 ページの「置き換えることができないデバイス」。
- 255 ページの「ZFS ストレージプール内のデバイスを置き換える」。
- 260 ページの「再同期化のステータスを表示する」。

デバイスを置き換えられるかどうかを確認する

置き換える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。たとえば、4 方向ミラーの 2 台のディスクが UNAVAIL の場合は、健全な複製を入手できるので、どちらのディスクも交換できます。ただし、4 方向 RAID-Z (raidz1) 仮想デバイス内の 2 台のディスクが UNAVAIL の場合は、データを入手するために必要な複製が不足しているため、どちらのディスクも交換できません。デバイスが損傷していてもオンラインである場合は、プールの状態が UNAVAIL でないかぎり、それを交換できます。ただし、デバイス上の破損した

データは、適切なデータを含む十分なレプリカが存在しないかぎり、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              UNAVAIL
```

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、UNAVAIL のディスクはどれも交換できません。プール自体が UNAVAIL のため、ONLINE のディスクも交換できません。

```
    raidz1              UNAVAIL
c1t0d0              ONLINE
c2t0d0              UNAVAIL
c3t0d0              UNAVAIL
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が UNAVAIL の場合は、プール自体が UNAVAIL のため、交換を行うことはできません。

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 UNAVAIL 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交換す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、272 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。破損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 system1 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。例:

```
# zpool replace system1 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、50 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 58 ZFS ストレージプール内の SATA ディスクを置き換える

次の例は、システムのミラー化ストレージプール `system1` のデバイス (`c1t3d0`) を SATA デバイスに置き換える方法を示しています。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。置き換えられるディスクが SATA ディスクでない場合は、50 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。

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 SATA 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 SATA ディスク (`c1t3d0`) を識別し、それを構成解除します。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。可能であれば、`UNAVAIL` の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。
- SATA 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。

注記 - あらかじめプールの `autoreplace` プロパティーを `on` 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 system1 c2t4d0
```

- FMA が障害の発生したデバイスを報告している場合、デバイスの障害を解決してください。

```
# fmadm faulty
# fmadm repaired zfs://pool=name/vdev=guid
```

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 system1 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 system1 c1t3d0
# zpool replace system1 c1t3d0
# zpool status system1
pool: system1
state: ONLINE
scrub: resilver completed after 0h0m with 0 errors on Tue Feb  2 13:17:32 2010
config:

NAME           STATE     READ WRITE CKSUM
system1        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
```

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。例:

```
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。例:

```
# zpool replace system1 c1t3d0 c4t3d0
# zpool status
pool: system1
state: DEGRADED
scrub: resilver completed after 0h0m with 0 errors on Tue Feb  2 13:35:41 2010
config:

NAME           STATE     READ WRITE CKSUM
system1        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 system1
pool: system1
state: ONLINE
scrub: resilver completed after 0h0m with 0 errors on Tue Feb  2 13:35:41 2010
config:

NAME          STATE     READ WRITE CKSUM
system1       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 59 障害が発生したログデバイスを交換する

`zpool status` コマンド出力でインテントログ障害が ZFS によって特定されています。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。

次の例は、ストレージプール `storpool` 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ています。基本的な手順は次のとおりです。

- `zpool status -x` の出力および https://support.oracle.com/ にある *ZFS* インテントログ読み取りエラー *(*ドキュメント *ID 1021625.1)* の項目に記載されている FMA 診断メッセージを確認してください。
- 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。
- FMA エラーを解決します。

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: storpool
state: FAULTED
status: One or more of the intent logs could not be read.
Waiting for administrator 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
storpool       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 storpool c0t5d0
# zpool clear storpool
# fmadm faulty
# fmadm repair zfs://pool=name/vdev=guid
```

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` です。
- ログデバイスをオンラインに戻します。

  ```
  # zpool online storpool 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 storpool
  ```

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。

再同期化のステータスを表示する

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` コマンドを使ってモニターできます。

次の `zpool status` 再同期化ステータスメッセージが表示されます。

- 再同期化進捗レポート。例:

```
scan: resilver in progress since Mon Jun  7 09:17:27 2010
13.3G scanned
13.3G resilvered at 18.5M/s, 82.34% done, 0h2m to go
```

- 再同期化完了メッセージ。例:

```
resilvered 16.2G in 0h16m with 0 errors on Mon Jun  7 09:34:21 2010
```

再同期化完了メッセージはシステムのリブート後も残ります。

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。プールの使用済みディスク領域が数ギガバイトのみの場合は、500G バイトディスクの置換には数秒かかることがあります。
- システムの電源が切れるか、またはシステムがリブートした場合には、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。例:

```
# zpool status system1
pool: system1
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.
scan: resilver in progress since Mon Jun  7 10:49:20 2010
54.6M scanned54.5M resilvered at 5.46M/s, 24.64% done, 0h0m to go

config:

NAME             STATE     READ WRITE CKSUM
system1          ONLINE       0     0     0
   mirror-0      ONLINE       0     0     0
   replacing-0   ONLINE       0     0     0
```

```
      c1t0d0     ONLINE       0     0     0
      c2t0d0     ONLINE       0     0     0  (resilvering)
      c1t1d0     ONLINE       0     0     0
```

この例では、ディスク c1t0d0 が c2t0d0 に置き換わります。ステータスが replacing の仮想デバイスが構成に存在しているので、このイベントはステータス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。例:

```
# zpool status system1
pool: system1
state: ONLINE
scrub: resilver completed after 0h1m with 0 errors on Tue Feb  2 13:54:30 2010
config:

NAME          STATE     READ WRITE CKSUM
system1       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。

プールデバイスを変更する

アクティブなプールの下にあるプールデバイスを変更しようとしないでください。

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。

そのほかの問題は、プールの下にあるデバイスを変更しようとし、続いて zpool status コマンドを非ルートユーザーとしてを使用した場合、以前のデバイス名が表示される可能性があることです。

ZFS ストレージプール内のデータの問題を解決する

データに関する問題の例には次のものがあります。

- プールまたはファイルシステム領域が見つからない
- 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、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、あとのセクションで説明します。

ZFS の領域の問題を解決する

ZFS がファイルシステム領域とプール領域の計上をどのように報告するかわからない場合は、次のセクションを確認してください。

ZFS ファイルシステム領域の報告

利用可能なプールおよびファイルシステムの領域を判別する場合、`zpool list` および `zfs list` コマンドは、以前の `df` および `du` コマンドより優れています。旧バージョンのコマンドでは、プールおよびファイルシステムの領域を簡単に識別できず、下位のファイルシステムまたはスナップショットによって消費される領域の詳細を表示できません。

たとえば、次のルートプール (`rpool`) は、5.46GB が割り当て済みで、68.5GB は空き領域です。

```
# zpool list rpool
NAME   SIZE  ALLOC   FREE  CAP  DEDUP  HEALTH  ALTROOT
rpool   74G  5.46G  68.5G   7%  1.00x  ONLINE  -
```

個々のファイルシステムの `USED` 列を確認することでプール領域の数値とファイルシステム領域の数値を比較すれば、`ALLOC` で報告されるプール領域はファイルシステムの `USED` の合計であることがわかります。例:

```
# zfs list -r rpool
NAME                       USED  AVAIL  REFER  MOUNTPOINT
rpool                     5.41G  67.4G  74.5K  /rpool
rpool/ROOT                3.37G  67.4G    31K  legacy
rpool/ROOT/solaris        3.37G  67.4G  3.07G  /
rpool/ROOT/solaris/var     302M  67.4G   214M  /var
rpool/dump                1.01G  67.5G  1000M  -
rpool/export              97.5K  67.4G    32K  /rpool/export
rpool/export/home         65.5K  67.4G    32K  /rpool/export/home
rpool/export/home/admin   33.5K  67.4G  33.5K  /rpool/export/home/admin
rpool/swap                1.03G  67.5G  1.00G  -
```

ZFS ストレージプール領域の報告

`zpool list` コマンドによって報告される SIZE 値は、通常、プール内の物理ディスク領域の大きさですが、プールの冗長性レベルに応じて異なります。次の例を参照してください。`zfs list` コマンドは、使用可能な領域のうち、ファイルシステムで利用できる領域を示します。これは、ディスク領域から ZFS プール冗長性メタデータオーバーヘッド (ある場合) を差し引いたものです。

次の ZFS データセット構成は、`zfs list` コマンドでは割り当てられた領域として追跡されますが、`zpool list` の出力では割り当てられた領域として追跡されません。

- ZFS ファイルシステム割り当て制限
- ZFS ファイルシステム予約
- ZFS 論理ボリュームサイズ

次の各項目では、さまざまなプール構成、ZFS ボリューム、および ZFS 予約の使用が、消費されたディスク容量や使用可能なディスク容量にどのように影響する場合があるかについて説明します。構成に応じて、下に示されている手順を使用してプール領域のモニタリングを追跡するようにしてください。

- **非冗長性ストレージプール** – 136G バイトのディスク 1 つでプールを作成すると、`zpool list` コマンドによって `SIZE` および初期 `FREE` 値が 136G バイトとして報告されます。`zfs list` コマンドによって報告された初期 `AVAIL` 領域は、プールメタデータオーバーヘッドが少量あるため 134G バイトです。例:

  ```
  # zpool create system1 c0t6d0
  # zpool list system1
  NAME      SIZE  ALLOC   FREE    CAP  DEDUP  HEALTH  ALTROOT
  system1   136G  95.5K   136G     0%  1.00x  ONLINE  -
  # zfs list system1
  NAME      USED  AVAIL  REFER  MOUNTPOINT
  system1    72K   134G    21K  /system1
  ```

- **ミラー化ストレージプール** – 136G バイトのディスク 2 つでプールを作成すると、`zpool list` コマンドによって `SIZE` が 136G バイト、初期 `FREE` 値が 136G バイトとして報告されます。この報告は、デフレートされた領域値と呼ばれます。`zfs list` コマンドによって報告された初期 `AVAIL` 領域は、プールメタデータオーバーヘッドが少量あるため 134G バイトです。例:

```
# zpool create system1 mirror c0t6d0 c0t7d0
# zpool list system1
NAME      SIZE  ALLOC   FREE    CAP  DEDUP  HEALTH  ALTROOT
system1   136G  95.5K   136G     0%  1.00x  ONLINE  -
# zfs list system1
NAME      USED  AVAIL  REFER  MOUNTPOINT
system1    72K   134G    21K  /system1
```

- **RAID-Z ストレージプール** – 136G バイトのディスク 3 つで `raidz2` プールを作成すると、`zpool list` コマンドによって `SIZE` および初期 `FREE` 値が 408G バイトとして報告されます。この報告は、インフレートされたディスク領域値と呼ばれます。パリティー情報などの冗長性オーバーヘッドが含まれています。`zfs list` コマンドによって報告される初期 `AVAIL` 領域は、プール冗長性オーバーヘッドのため 133G バイトです。RAID-Z プールに関する `zpool list` および `zfs list` の出力間で領域に違いがあるのは、`zpool list` によってインフレートされたプール領域が報告されたためです。

```
# zpool create system1 raidz2 c0t6d0 c0t7d0 c0t8d0
# zpool list system1
NAME      SIZE  ALLOC   FREE    CAP  DEDUP  HEALTH  ALTROOT
system1   408G   286K   408G     0%  1.00x  ONLINE  -
# zfs list system1
NAME      USED  AVAIL  REFER  MOUNTPOINT
system1  73.2K   133G  20.9K  /system1
```

- **NFS マウントされたファイルシステム領域** – `zpool list` も `zfs list` も、NFS マウントされたファイルシステム領域を考慮しません。ただし、ローカルデータファイルは、マウントされた NFS ファイルシステムの下で非表示になっている可能性があります。ファイルシステム領域が見つからない場合は、NFS ファイルシステムの下でデータファイルが非表示になっていないか確認してください。
- **ZFS ボリュームの使用** – ZFS ファイルシステムが作成され、プール領域が消費された場合は、zpool list コマンドを使用してファイルシステムの容量消費を表示できます。例:

```
# zpool create nova mirror c1t1d0 c2t1d0
# zfs create nova/fs1
# mkfile 10g /nova/fs1/file1_10g
# zpool list nova
NAME  SIZE  ALLOC   FREE  CAP  DEDUP  HEALTH  ALTROOT
nova   68G  10.0G  58.0G  14%  1.00x  ONLINE  -
# zfs list -r nova
NAME       USED  AVAIL  REFER  MOUNTPOINT
nova      10.0G  56.9G    32K  /nova
nova/fs1  10.0G  56.9G  10.0G  /nova/fs1
```

10G バイトの ZFS ボリュームを作成した場合、この容量は zpool list コマンドの出力に含まれません。この容量は、zfs list コマンドの出力に含まれます。ストレー

ジプール内の ZFS ボリュームを使用している場合は、zfs list コマンドを使用して ZFS ボリュームの容量消費をモニターします。例:

```
# zfs create -V 10g nova/vol1
# zpool list nova
NAME  SIZE  ALLOC   FREE  CAP  DEDUP  HEALTH  ALTROOT
nova   68G  10.0G  58.0G  14%  1.00x  ONLINE  -
# zfs list -r nova
NAME        USED  AVAIL  REFER  MOUNTPOINT
nova       20.3G  46.6G    32K  /nova
nova/fs1   10.0G  46.6G  10.0G  /nova/fs1
nova/vol1  10.3G  56.9G    16K  -
```

上の出力で、ZFS ボリュームの容量は zpool list の出力では追跡されません。そのため、ZFS ボリュームによって消費された容量を識別するには zfs list または zfs list -o space コマンドを使用します。

さらに、ZFS ボリュームは raw デバイスのように機能するため、メタデータのための一定の容量が refreservation プロパティーによって自動的に予約されます。これにより、ボリュームでは、そのボリュームが作成されたときに指定された量より少し多い容量が消費されます。ZFS ボリューム上の refreservation を削除しないでください。削除すると、ボリューム容量が不足するおそれがあります。

- **ZFS 予約の使用** – 予約を使用してファイルシステムを作成するか、または既存のファイルシステムに予約を追加した場合、予約または refreservation は zpool list コマンドによって追跡されません。

  zfs list -r コマンドを使用して、増加した USED 容量を識別することにより、ファイルシステムの予約によって消費された容量を識別します。例:

```
# zfs create -o reservation=10g nova/fs2
# zpool list nova
NAME  SIZE  ALLOC   FREE  CAP  DEDUP  HEALTH  ALTROOT
nova   68G  10.0G  58.0G  14%  1.00x  ONLINE  -
# zfs list -r nova
NAME        USED  AVAIL  REFER  MOUNTPOINT
nova       30.3G  36.6G    33K  /nova
nova/fs1   10.0G  36.6G  10.0G  /nova/fs1
nova/fs2     31K  46.6G    31K  /nova/fs2
nova/vol1  10.3G  46.9G    16K  -
```

  refreservation を使用してファイルシステムを作成した場合、その refreservation は zfs list -r コマンドを使用して識別できます。例:

```
# zfs create -o refreservation=10g nova/fs3
# zfs list -r nova
NAME        USED  AVAIL  REFER  MOUNTPOINT
nova       40.3G  26.6G    35K  /nova
nova/fs1   10.0G  26.6G  10.0G  /nova/fs1
```

```
nova/fs2     31K  36.6G    31K  /nova/fs2
nova/fs3     10G  36.6G    31K  /nova/fs3
nova/vol1  10.3G  36.9G    16K  -
```

USED 容量の合計に含まれる既存のすべての予約を識別するには、次のコマンドを使用します。

```
# zfs get -r reserv,refreserv nova
NAME        PROPERTY        VALUE  SOURCE
nova        reservation     none   default
nova        refreservation  none   default
nova/fs1    reservation     none   default
nova/fs1    refreservation  none   default
nova/fs2    reservation     10G    local
nova/fs2    refreservation  none   default
nova/fs3    reservation     none   default
nova/fs3    refreservation  10G    local
nova/vol1   reservation     none   default
nova/vol1   refreservation  10.3G  local
```

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。整合性のないデータが ZFS 構成内のディスク上に存在するようになるのは、ハードウェアに障害がある場合 (この場合、プールは冗長であったはずです) または ZFS ソフトウェアにバグが存在する場合のみです。

`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、272 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。このタスクでは従来から、ファイルシステムをアンマウントし、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メカニズムが用意されています。スクラブとして知られているこの機能は、ハードウェアまたはソフトウェアの障害に進む前にエラーを検出および防止する方法として、一般にメモリーおよびほかのシステム内で使用されます。

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。例:

```
# zpool scrub system1
```

現在のスクラブ操作のステータスは、zpool status コマンドを使用して表示できます。例:

```
# zpool status -v system1
```

```
pool: system1
state: ONLINE
scan: scrub in progress since Mon Jun  7 12:07:52 2010
201M scanned out of 222M at 9.55M/s, 0h0m to go
0 repaired, 90.44% done
config:

NAME          STATE     READ WRITE CKSUM
system1       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。例:

```
# zpool scrub -s system1
```

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。システムの大部分がアイドル状態で、ディスクへの電力を節約する場合は、バックグラウンドスクラブではなく、cron スケジュールされた明示的なスクラブを使用することを検討する必要があります。この場合もデータの完全なスクラブが実行されます。ただし、大量の I/O が生成されるのはスクラブが終了するまでで、終了時点でディスクを通常どおり電源管理できるようになります。(I/O の増加以外の) 欠点は、スクラブがまったく行われていない時間が大きくなり、その時間内に破損のリスクが増える可能性があることです。

`zpool status` の出力の解釈の詳細については、60 ページの「ZFS ストレージプールのステータス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。スクラブ操作の実行中に再同期化を実行すると、現在のスクラブは中断され、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、260 ページの「再同期化のステータスを表示する」を参照してください。

破損した ZFS データを修復する

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。次のセクションで説明するように、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。

次のセクションでは、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。

- 270 ページの「データ破壊の種類を確認する」。
- 271 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」。
- 272 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、266 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。例:

```
# zpool status system1
pool: system1
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://support.oracle.com/msg/ZFS-8000-8A
config:

NAME                          STATE     READ WRITE CKSUM
   system1                    ONLINE       4     0     0
      c0t5000C500335E106Bd0   ONLINE       0     0     0
      c0t5000C500335FC3E7d0   ONLINE       4     0     0


errors: 2 data errors, use '-v' for a list
```

特定の時間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。例:

```
# zpool status -v morpheus
pool: morpheus
id: 13289416187275223932
state: UNAVAIL
status: The pool metadata is corrupted.
action: The pool cannot be imported due to damaged devices or data.
see: http://support.oracle.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。例:

```
# zpool status system1 -v
pool: system1
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://support.oracle.com/msg/ZFS-8000-8A
config:

NAME                          STATE     READ WRITE CKSUM
   system1                    ONLINE       4     0     0
      c0t5000C500335E106Bd0   ONLINE       0     0     0
      c0t5000C500335FC3E7d0   ONLINE       4     0     0

errors: Permanent errors have been detected in the following files:
/system1/file.1
/system1/file.2
```

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。例:

  ```
  /path1/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。例:

  ```
  path1/documents/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。例:

  ```
  path1/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。

プールをスクラブし、複数回繰り返すプールエラーをクリアすることによって、よりマイナーなデータ破損を解決しようとすることができます。最初にスクラブし、繰り返しをクリアしても破損ファイルが解決されない場合は、再度実行してください。例:

```
# zpool scrub system1
# zpool clear system1
```

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。

複数のブロック参照を持つ、破壊されたデータを修復する

損傷を受けたファイルシステムにある破壊されたデータが、スナップショットなどの複数のブロック参照を持つ場合は、`zpool status -v` コマンドを使用しても、破壊された**すべての**データパスが表示されるわけではありません。現在の `zpool status` による破壊されたデータの報告は、破壊されたメタデータの量や、`zpool status` コマンドの実行後にブロックが再利用されたかどうかによって制限されます。複製解除されたブロックがあると、破壊されたすべてのデータの報告がさらに複雑になります。

破壊されたデータがあり、スナップショットデータが影響を受けることが `zpool status -v` コマンドによって示された場合は、破壊されたほかのパスを特定するために次のコマンドを実行することを検討してください。

```
# find mount-point -inum $inode -print
# find mount-point/.zfs/snapshot -inum $inode -print
```

最初のコマンドは、指定されたファイルシステムとそのすべてのスナップショット内で、破壊が報告されたデータの inode 番号を検索します。2 番目のコマンドは、同じ inode 番号を持つスナップショットを検索します。

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ることができます。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。例:

  ```
  # zpool status
  pool: storpool
  ```

```
state: UNAVAIL
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 Fri Jun 29 17:22:49 2012
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://support.oracle.com/msg/ZFS-8000-72
scrub: none requested
config:

NAME         STATE     READ WRITE CKSUM
storpool     UNAVAIL      0     0     1  corrupted data
   c1t1d0    ONLINE       0     0     2
   c1t3d0    ONLINE       0     0     4
```

前の出力で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 storpool
```

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 storpool
cannot import 'storpool': I/O error
Recovery is possible, but will result in some data loss.
Returning the pool to its state as of Fri Jun 29 17:22:49 2012
should correct the problem.  Approximately 5 seconds of data
must be discarded, irreversibly.  Recovery can be attempted
by executing 'zpool import -F storpool'.  A scrub of the pool
is strongly recommended after recovery.
```

前の出力で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 storpool
Pool storpool returned to its state as of Fri Jun 29 17:22:49 2012.
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、システムのブート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。

- 損傷したプールを読み取り専用モードでインポートできます。この方法によってプールをインポートでき、データにアクセスできます。例:

```
# zpool import -o readonly=on storpool
```

プールを読み取り専用でインポートすることの詳細については、73 ページの「読み取り専用モードでプールをインポートする」を参照してください。

- `zpool import -m` コマンドを使用して、ログデバイスのないプールをインポートできます。詳細は、73 ページの「ログデバイスがないプールをインポートする」を参照してください。
- いずれのプール回復方法によっても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` コマンドによっ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、プールにアクセスできなくなると、この情報にもアクセスできなくなるため、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を記述するファイルをどこか安全な場所に保存します。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。その後、好きなバックアップまたは復元方法を使用してデータを投入できます。

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。いずれの場合も、プール (表示されている場合) をエクスポートし、再インポートして、構成を回復できます。

プールのインポートとエクスポートについては、69 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。

ブート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。各プールはブート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リブート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、起動時にプールを検索しないよう ZFS に通知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。システムをブートできなくなった場合は、`-m milestone=none` ブートオプションを使用して、マイルストーン `none` でブートします。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。代替ルートからブート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、ブート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ブート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。

♦ ♦ ♦ 第 13 章

13 Oracle Solaris ZFS の推奨されるプラクティス

この章では、ZFS ストレージプールおよびファイルシステムを作成、モニタリング、および保守するための推奨のプラクティスについて説明します。

この章は、次のセクションで構成されます。

Oracle データベースの調整を含む ZFS の一般的な調整情報については、『Oracle Solaris 11.3 カーネルのチューンアップ・リファレンスマニュアル』 の 第 3 章, 「Oracle Solaris ZFS チューニング可能パラメータ」を参照してください。

推奨のストレージプールのプラクティス

以降のセクションでは、ZFS ストレージプールを作成およびモニターするための推奨のプラクティスを紹介します。ストレージプールの問題をトラブルシューティングする方法については、第12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。

一般的なシステムプラクティス

- 最新の Oracle Solaris 更新およびリリースでシステムを最新の状態に保ちます
- データが必ず安全に書き込まれるように、コントローラがキャッシュフラッシュコマンドを受け付けることを確認してください。これは、プールのデバイスを変更する前、またはミラー化ストレージプールを分割する前に重要になります。これは通常 Oracle/Sun ハードウェアの問題ではありませんが、ハードウェアのキャッシュフラッシュ設定が有効であることを確認するのをお勧めします。
- 実際のシステム作業負荷に必要なメモリーのサイズを特定します
  - データベースアプリケーションなどの既知のアプリケーションのメモリーフットプリントでは、アプリケーションが ZFS キャッシュから必要なメモリーを繰

り返し要求する必要がないように、ARC サイズに上限を設定してもかまいません。

- 複製解除のメモリー要件を考慮します
- 次のコマンドで ZFS のメモリー使用量を特定します。

```
# mdb -k
> ::memstat
Page Summary                Pages                MB  %Tot
------------     ----------------  ----------------  ----
Kernel                     388117              1516   19%
ZFS File Data               81321               317    4%
Anon                        29928               116    1%
Exec and libs                1359                 5    0%
Page cache                   4890                19    0%
Free (cachelist)             6030                23    0%
Free (freelist)           1581183              6176   76%

Total                     2092828              8175
Physical                  2092827              8175
> $q
```

- ZFS ARC キャッシュをチューニングするヒントについては、My Oracle Support (MOS) のドキュメント 1663862.1、『*Memory Management Between ZFS and Applications in Oracle Solaris 11.x*』を参照してください。このドキュメントには、`user_reserver_hint_pct` メモリー管理パラメーターを変更するために使用できるスクリプトが含まれています。
- メモリーの破損を防止するため、ECC メモリーの使用を検討してください。メモリーが暗黙のうちに破損すると、データも破損する可能性があります。

- 定期的にバックアップを実行する – ZFS の冗長性を伴って作成されたプールは、ハードウェアの障害によるダウンタイムの短縮に役立ちますが、ハードウェア障害、電源障害、またはケーブルの切断の影響を受けないわけではありません。必ず定期的にデータをバックアップしてください。データが重要である場合、バックアップする必要があります。データのコピーを取得するには次のようなさまざまな方法があります。
  - 定期的または日単位の ZFS スナップショット
  - ZFS プールデータの週単位のバックアップ。`zpool split` コマンドを使用して、ZFS ミラー化ストレージプールの正確な複製を作成できます。
  - エンタープライズレベルのバックアップ製品を使用した月単位のバックアップ
- ハードウェア RAID
  - ZFS がストレージと冗長性を管理できるように、ストレージアレイにハードウェア RAID でなく JBOD モードを使用することを検討してください。
  - ハードウェア RAID または ZFS 冗長性、あるいは両方を使用します
  - ZFS 冗長性を使用することで多くの利点があります。本稼働環境では、ZFS がデータ不整合を修復できるように構成します。ベースとなるストレージデバ

イスに実装されている RAID レベルに関係なく、RAID-Z、RAID-Z-2、RAID-Z-3、ミラーなどの ZFS 冗長性を使用します。こうした冗長性を使用することで、配下のストレージデバイスまたはそこからホストへの接続での障害を ZFS で検出して修復できます。

- ハードウェア RAID ソリューションの冗長性が確実な場合は、ZFS 冗長性なしの ZFS をハードウェア RAID アレイとともに使用することを検討します。ただし、データ整合性を確保するために、次の推奨事項に従ってください。
  - ハードウェア RAID アレイで障害が発生した場合は ZFS がデータの不整合を解決できないことを考慮して、安心感に基づいて LUN および ZFS ストレージプールのサイズを割り当てます。
  - RAID5 LUN とグローバルホットスペアを作成します。
  - `zpool status` を使用して ZFS ストレージプールを、ハードウェア RAID モニタリングツールを使用して配下の LUN をモニターします。
  - 障害が発生したデバイスはすみやかに置換します。
  - データセンター品質のサービスを使用している場合は、毎月など、定期的に ZFS ストレージプールをスクラブします。
  - 重要なデータの正常な最新のバックアップを常に用意します。

282 ページの「ローカルまたはネットワーク接続のストレージアレイでのプール作成のプラクティス」も参照してください。

- クラッシュダンプは多くのディスク容量を消費し、通常、物理メモリーの 1/2 - 3/4 の範囲のサイズになります。

ZFS ストレージプール作成のプラクティス

以降のセクションでは、プールの全般的なプラクティスとより具体的なプールのプラクティスについて説明します。

全般的なストレージプールのプラクティス

- ディスク全体を使用して、ディスク書き込みキャッシュを有効にし、保守をより簡単にします。スライス上にプールを作成すると、ディスクの管理および回復がより複雑になります。
- ZFS がデータ不整合を修復できるように、ZFS 冗長性を使用します。
  - 冗長でないプールが作成されると、次のメッセージが表示されます。

```
# zpool create system1 c4t1d0 c4t3d0
'system1' successfully created, but with no redundancy; failure
of one device will cause loss of the pool
```

  - ミラー化プールの場合は、ミラー化ディスクペアを使用します

    - RAID-Z プールの場合は、VDEV ごとに 3 - 9 個のディスクをグループ化します
    - 同じプール内に RAID-Z とミラー化コンポーネントを混在させないでください。これらのプールは管理が難しく、パフォーマンスが低下する可能性があります。
- ホットスペアを使用してハードウェアの障害によるダウンタイムを短縮します
- デバイス間で I/O が均衡するように、サイズが同程度のディスクを使用します
    - 小さな LUN は大きな LUN に拡張できます
    - metaslab を適切なサイズに保つために、LUN のサイズを極端に異なるもの (128M バイトから 2T バイトへなど) に拡張しないでください
- より高速なシステム回復をサポートするために小さなルートプールと大きなデータプールを作成することを検討してください
- 推奨される最小のプールサイズは 8G バイトです。最小のプールサイズは 64M バイトですが、8G バイト未満では空きプール領域の割り当てや再利用が難しくなります。
- 推奨される最大のプールサイズは、実際の作業負荷やデータサイズが十分収まるサイズのはずです。定期的にバックアップできる量を超えるデータを格納しようとしないでください。そうしないと、データに予期しない問題が発生する可能性があります。

282 ページの「ローカルまたはネットワーク接続のストレージアレイでのプール作成のプラクティス」も参照してください。

ルートプール作成のプラクティス

- **SPARC (SMI (VTOC))**: `s*` 識別子を使用して、スライスでルートプールを作成します。`p*` 識別子を使用しないでください。通常、システムの ZFS ルートプールは、システムがインストールされるときに作成されます。2 つ目のルートプールを作成するか、またはルートプールを再作成する場合は、SPARC システムで次のような構文を使用します。

  ```
  # zpool create rpool c0t1d0s0
  ```

  あるいは、ミラー化ルートプールを作成します。例:

  ```
  # zpool create rpool mirror c0t1d0s0 c0t2d0s0
  ```

- **Solaris 11.1 x86 (EFI (GPT))**: `d*` 識別子を使用して、ディスク全体でルートプールを作成します。`p*` 識別子を使用しないでください。通常、システムの ZFS ルートプールは、システムがインストールされるときに作成されます。2 つめのルートプールを作成するか、またはルートプールを再作成する場合は、次のような構文を使用します。

  ```
  # zpool create rpool c0t1d0
  ```

  あるいは、ミラー化ルートプールを作成します。例:

```
# zpool create rpool mirror c0t1d0 c0t2d0
```

- 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。RAID-Z もストライプ化構成もサポートされていません。`zpool add` コマンドを使って、追加ディスクを追加して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。
- 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。
- プールプロパティーは、AI インストール中に設定できますが、`gzip` 圧縮アルゴリズムはルートプールでサポートされていません。
- ルートプールを初期インストールによって作成したあとは、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、システムがブートできなくなる可能性があります。
- ルートプールディスクは連続的な操作に重要であるため (特にエンタープライズ環境で)、本番システムではルートプールを USB スティック上に作成しないでください。ルートプールにシステムの内蔵ディスクを使用することを検討するか、あるいは、少なくとも非ルートデータに使用するのと同品質のディスクを使用してください。また、USB スティックは、物理メモリーの少なくとも 1/2 のサイズに等しいダンプボリュームサイズをサポートするのに十分な大きさではない可能性があります。
- ルートプールにホットスペアを追加するのではなく、2 方向または 3 方向のミラールートプールを作成することを検討してください。さらに、ルートプールとデータプール間でホットスペアを共有しないでください。
- VMware のシンプロビジョニングされたデバイスをルートプールデバイスに使用しないでください。

非ルートプール作成のプラクティス

- `d*` 識別子を使用して、ディスク全体で非ルートプールを作成します。`p*` 識別子を使用しないでください。
  - ZFS は、追加のボリューム管理ソフトウェアを一切使わないで最適に機能します。
  - パフォーマンスを向上させるために、個々のディスクを使用するか、または少数のディスクで構成される LUN のみを使用します。ZFS での LUN セットアップに対する視認性を高めることにより、ZFS は入出力のスケジューリングをより適切に決定できます。
- 複数のコントローラにまたがる冗長なプール構成を作成して、コントローラの障害によるダウンタイムを短縮します。
  - **ミラー化ストレージプール** – 多くのディスクを消費しますが、一般に、小さなランダム読み取りでパフォーマンスが向上します。

    ```
    # zpool create system1 mirror c1d0 c2d0 mirror c3d0 c4d0
    ```

- **RAID-Z ストレージプール** – 3 つのパリティー方式を使って作成できます。この場合、パリティーは 1 (`raidz`)、2 (`raidz2`)、または 3 (`raidz3`) に等しくなります。RAID-Z 構成は、ディスク容量を最大化し、通常、データが大きなチャンク (128K 以上) で読み取りおよび書き込みされるときに、パフォーマンスが高くなります。
  - それぞれ 3 つのディスク (2+1) の 2 つの VDEV を持つシングルパリティー RAID-Z (`raidz`) 構成を検討してください。

    ```
    # zpool create rzpool raidz1 c1t0d0 c2t0d0 c3t0d0 raidz1 c1t1d0 c2t1d0 c3t1d0
    ```

  - RAIDZ-2 構成では、データの可用性が向上し、RAID-Z と同様の性能が提供されます。RAIDZ-2 は、RAID-Z または双方向ミラーよりもデータ損失までの平均時間 (MTTDL) がかなり短縮されます。6 台のディスク (4+2) でダブルパリティーの RAID-Z (`raidz2`) 構成を作成します。

    ```
    # zpool create rzpool raidz2 c0t1d0 c1t1d0 c4t1d0 c5t1d0 c6t1d0 c7t1d0
    raidz2 c0t2d0 c1t2d0 c4t2d0 c5t2d0 c6t2d0 c7t2d
    ```

  - RAIDZ-3 構成では、ディスク容量が最大となり、3 台のディスク障害に耐えられるため、優れた可用性が提供されます。9 つのディスク (6+3) では、トリプルパリティー RAID-Z (`raidz3`) 構成を作成します。

    ```
    # zpool create rzpool raidz3 c0t0d0 c1t0d0 c2t0d0 c3t0d0 c4t0d0
    c5t0d0 c6t0d0 c7t0d0 c8t0d0
    ```

ローカルまたはネットワーク接続のストレージアレイでのプール作成のプラクティス

ローカルまたはリモートで接続されているストレージアレイに ZFS ストレージプールを作成するときには、次のストレージプールのプラクティスを考慮してください。

- SAN デバイスにプールを作成し、ネットワーク接続の速度が低下した場合は、プールのデバイスが一定期間 `UNAVAIL` になる可能性があります。ネットワーク接続が連続的な方法でデータを提供するのに適しているかどうかを評価する必要があります。また、ルートプールに SAN デバイスを使用する場合は、システムがブートするとすぐにそれらが使用できなくなる可能性があり、ルートプールのデバイスも `UNAVAIL` になる可能性があることを考慮してください。
- フラッシュ書き込みキャッシュリクエストが ZFS から発行されたあとにディスクアレイがそのキャッシュをフラッシュしていないことを、アレイベンダーに確認してください。
- Oracle Solaris ZFS がローカルの小さなディスクキャッシュをアクティブ化できるように、ディスクスライスではなくディスク全体をストレージプールデバイスとして使用します。これにより、適切な時期にフラッシュされます。
- 最良のパフォーマンスを得るために、アレイ内の物理ディスクごとに 1 つの LUN を作成します。大きな LUN を 1 つだけ使用すると、ZFS がキューに入れる入出力読み取り操作の数が少なすぎて実際にはストレージを最適なパフォーマンスにする

ことができない可能性があります。反対に、小さな LUN を多数使用すると、ストレージが多数の保留中の入出力読み取り操作であふれてしまう可能性があります。

- 動的 (シン) プロビジョニングソフトウェアを使用して仮想領域割り当てを実装するストレージアレイは、Oracle Solaris ZFS にはお勧めしません。Oracle Solaris ZFS が変更されたデータを空き領域に書き込むと、それは LUN 全体に書き込まれます。Oracle Solaris ZFS の書き込みプロセスでは、すべての仮想領域をストレージアレイの視点から割り当てますが、これは動的プロビジョニングの利点を打ち消すものです。

  ZFS の使用時は、動的プロビジョニングソフトウェアが不要になる可能性があることを考慮してください。

  - 既存の ZFS ストレージプールで LUN を拡張できるため、新しい領域が使用されます。
  - 小さな LUN が大きな LUN に置き換えられるときも同様の動作が行われます。
  - プールのストレージニーズを評価し、必要なストレージニーズに等しい小さな LUN でプールを作成した場合、より多くの領域が必要であれば、いつでもそれらの LUN を大きなサイズに拡張できます。
- アレイが個々のデバイスを提供できる場合 (JBOD モード) は、このタイプのアレイに冗長な ZFS ストレージプール (ミラーまたは RAID-Z) を作成して、ZFS がデータの矛盾を報告および訂正できるようにすることを考慮してください。

Oracle データベース用のプール作成のプラクティス

Oracle データベースを作成するときには、次のストレージプールのプラクティスを考慮してください。

- ミラー化プールまたはハードウェア RAID を使用します。
- ランダム読み取り作業負荷には、一般的に RAID-Z プールは推奨されていません。
- データベース redo ログ用の個別のログデバイスで小さな個別のプールを作成します。
- アーカイブログ用の小さな個別のプールを作成します。

Oracle データベースに対する ZFS の調整の詳細については、『Oracle Solaris 11.3 カーネルのチューンアップ・リファレンスマニュアル』 の 「Oracle データベース用の ZFS のチューニング」を参照してください。

VirtualBox での ZFS ストレージプールの使用

- VirtualBox は、デフォルトでベースとなるストレージからキャッシュフラッシュコマンドを無視するように構成されています。これは、システムクラッシュやハードウェア障害が発生した場合にデータが失われる可能性があることを意味します。

- 次のコマンドを発行して、VirtualBox でのキャッシュフラッシュを有効にします。

  ```
  VBoxManage setextradata vm-name "VBoxInternal/Devices/type/0/LUN#n/Config/IgnoreFlush" 0
  ```

  - *vm-name* – 仮想マシンの名前
  - *type* – `piix3ide` (IDE 仮想コントローラーを使用している場合) または `ahci` (SATA コントローラーを使用している場合) のいずれかのコントローラータイプ。
  - *n* – ディスク番号

パフォーマンスを高めるためのストレージプールのプラクティス

- 一般的に、パフォーマンスを最適にするためには、プール容量が 90% を下回るようにします。パフォーマンスが影響を受けるパーセントは、ワークロードによって大きく左右されます。
  - ほとんどがデータの追加の場合 (1 回書き込み、削除なし)、ZFS が新しいブロックを見つけることは非常に簡単です。この場合、パーセントは通常よりも高くてもかまいません。最大で 95% に設定できます。
  - データが大きいファイルまたは大きいブロック (128K ファイルまたは 1M バイトブロック) で作成されており、データが一括操作で削除される場合、パーセントは通常よりも高くてもかまいません。最大で 95% に設定できます。
  - プールの大きな部分 (50% 以上) が 8k のチャンク (DB ファイル、iSCSI LUN、または多くの小さいファイル) で構成され、常に書き換えがある場合、90% ルールに厳しく従うようにします。
  - すべてのデータが小さいブロックで、常に書き換えが行われている場合、容量が 80% を超えたらプールをしっかりモニターします。監視する兆候は、同じレベルのクライアント IOPS を達成するためにディスク IOPS が増加していることです
- ランダムな読み取り/書き込み作業負荷の場合、RAID-Z プールにわたるミラー化プールをお勧めします
- 個別のログデバイス
  - 同期書き込みパフォーマンスを高めるために推奨されています
  - 同期書き込み負荷が高い場合でも、メインプール内の多数のログブロックに書き込むことでの断片化を防ぎます
- 読み取りパフォーマンスを高めるには、個別のキャッシュデバイスをお勧めします
- スクラブ/再同期化 - 多数のデバイスで構成される非常に大きな RAID-Z プールは、スクラブや再同期化の時間が長くなります
- プールパフォーマンスが低い – `zpool status` コマンドを使用して、プールのパフォーマンス問題の原因となっているハードウェアの問題を排除します。`zpool status` コマンドで問題が現れない場合は、`fmdump` コマンドを使用して、ハード

ウェアの障害を表示するか、fmdump -eV コマンドを使用して、報告された障害にはまだなっていないハードウェアエラーを確認します。

ZFS ストレージプールの保守およびモニタリングのプラクティス

- パフォーマンスを最適にするために、必ずプール容量が 90% を下回るようにします。

  ビジー状態のメールサーバー上など、プールがいっぱいでファイルシステムが頻繁に更新されるときは、プールパフォーマンスが低下する可能性があります。プールがいっぱいになると、パフォーマンスペナルティーが発生することがありますが、それ以外の問題は発生しません。主要な作業負荷が不変のファイルの場合は、プール使用率の範囲を 95 - 96% に維持してください。95 - 96% の範囲のほとんど静的なコンテンツでも、書き込み、読み取り、および再同期のパフォーマンスが低下することがあります。

  - プールとファイルシステムの容量をモニターして、それらがいっぱいにならないようにします。
  - ZFS の割り当て制限と予約を使用して、ファイルシステムの容量がプール容量の 90% を超えないようにすることを検討します。
- プールの健全性をモニターします
  - 冗長プールを少なくとも週に一度、zpool status および fmdump でモニターします
  - 冗長でないプールを少なくとも週に二度、zpool status および fmdump でモニターします
- zpool scrub を定期的に実行して、データ整合性の問題を特定します。
  - 消費者品質のドライブがある場合は、スクラブを週に 1 度行うスケジュールを考えます。
  - データセンター品質のドライブがある場合は、スクラブを月に 1 度行うスケジュールを考えます。
  - デバイスを交換する前やプールの冗長性を一時的に下げる前にもスクラブを実行して、すべてのデバイスが現在運用可能であることを確認するようにしてください。
- プールまたはデバイス障害のモニタリング - 下記のように zpool status を使用します。また、fmdump または fmdump -eV を使用して、デバイスの障害またはエラーが発生しているかどうかを調べます。
  - 冗長プールの場合は、zpool status および fmdump を使用して、週単位でプールの健全性をモニターします
  - 冗長でないプールの場合は、zpool status および fmdump を使用して、週に二度プールの健全性をモニターします

- プールデバイスが UNAVAIL または OFFLINE である – プールデバイスが使用できない場合、そのデバイスが format コマンド出力に一覧表示されているかどうかを確認します。デバイスが format 出力に一覧表示されていない場合、デバイスは ZFS に認識されていません。

  プールデバイスが UNAVAIL または OFFLINE である場合、これは通常、デバイスに障害があったりケーブルが切断されていたりすること、または、不良ケーブルや不良コントローラなど、ほかのハードウェアの問題が原因でデバイスにアクセスできないことを示します。

- ハードウェアコンポーネントが欠陥があると診断されたときに通知するように、smtp-notify サービスを構成することを検討してください。詳細は、smf(5) および smtp-notify(1M) の通知パラメータセクションを参照してください。

  デフォルトでは、いくつかの通知は root ユーザーに送信されるように自動的に設定されます。/etc/aliases ファイルでユーザーアカウントの別名を root として追加した場合は、次のような電子メール通知を受け取ります。

```
From noaccess@tardis.space.com Fri Jun 29 16:58:59 2012
Date: Fri, 29 Jun 2012 16:58:58 -0600 (MDT)
From: No Access User <noaccess@tardis.space.com>
Message-Id: <201206292258.q5TMwwFL002753@tardis.space.com>
Subject: Fault Management Event: tardis:ZFS-8000-8A
To: root@tardis.central.com
Content-Length: 771

SUNW-MSG-ID: ZFS-8000-8A, TYPE: Fault, VER: 1, SEVERITY: Critical
EVENT-TIME: Fri Jun 29 16:58:58 MDT 2012
PLATFORM: ORCL,SPARC-T3-4, CSN: 1120BDRCCD, HOSTNAME: tardis
SOURCE: zfs-diagnosis, REV: 1.0
EVENT-ID: 76c2d1d1-4631-4220-dbbc-a3574b1ee807
DESC: A file or directory in pool 'pond' could not be read due to corrupt data.
AUTO-RESPONSE: No automated response will occur.
IMPACT: The file or directory is unavailable.
REC-ACTION: Use 'fmadm faulty' to provide a more detailed view of this event.
Run 'zpool status -xv' and examine the list of damaged files to determine what
has been affected. Please refer to the associated reference document at
http://support.oracle.com/msg/ZFS-8000-8A for the latest service procedures
and policies regarding this diagnosis.
```

- ストレージプール容量をモニターする – zpool list コマンドと zfs list コマンドを使用して、ファイルシステムデータによってどれだけのディスクが消費されたかを特定します。ZFS スナップショットがディスク容量を消費する可能性があります。zfs list コマンドで ZFS スナップショットが一覧表示されない場合、認識されずにディスクを消費していることがあります。zfs list - t スナップショットコマンドを使用して、スナップショットが消費するディスク容量を特定します。

推奨のファイルシステムのプラクティス

以降のセクションでは、推奨のファイルシステムのプラクティスについて説明します。

ルートファイルシステムのプラクティス

- ルートプールの回復が高速になるよう、ルートファイルシステムを小さく保ち、ほかのルートに関連しないデータから分離することを検討します。
- rpool/ROOT にファイルシステムを含めないようにします。これは、管理を必要としない特別なコンテナであり、追加のコンポーネントを含めないようにする必要があります。

ファイルシステム作成のプラクティス

以降のセクションでは、ZFS ファイルシステム作成のプラクティスについて説明します。

- ホームディレクトリ用にユーザーごとに 1 つのファイルシステムを作成します。
- ファイルシステムの割り当て制限と予約を使用して、重要なファイルシステムのディスク容量を管理し確保することを検討してください。
- 多数のユーザーがいる環境では、ユーザーおよびグループの割り当て制限を使用して、ディスク容量を管理することを検討してください。
- ZFS プロパティー継承を使用して、多数の子孫ファイルシステムにプロパティーを適用します。

Oracle データベース用のファイルシステム作成のプラクティス

Oracle データベースを作成する場合、次のファイルシステムのプラクティスを考慮してください。

- ZFS `recordsize` プロパティーを Oracle `db_block_size` に一致させます。
- 8K バイトの `recordsize` とデフォルトの `primarycache` 値を使用して、メインデータベースプールに、データベーステーブルおよびインデックスファイルシステムを作成します。
- デフォルトの `recordsize` および `primarycache` 値を使用して、メインデータベースプールに、temp データおよび undo テーブル領域ファイルシステムを作成します。

- 圧縮を有効にし、デフォルトの `recordsize` 値を使用し、`primarycache` を `metadata` に設定して、アーカイブプールにアーカイブログファイルシステムを作成します。

詳細は、次のホワイトペーパーを参照してください。

http://blogs.oracle.com/storage/entry/new_white_paper_configuring_oracle

ZFS ファイルシステムのプラクティスをモニターする

使用可能であることを確認するため、および容量消費の問題を特定するために、ZFS ファイルシステムをモニターする必要があります。

- レガシーコマンドでは、子孫ファイルシステムまたはスナップショットによって消費される容量が明らかにならないため、`du` および `df` コマンドではなく、`zpool list` および `zfs list` コマンドを使用して週単位でファイルシステムの空き容量をモニターします。

  詳細は、262 ページの「ZFS の領域の問題を解決する」を参照してください。
- `zfs list -o space` コマンドを使用して、ファイルシステムの容量消費を表示します。
- ファイルシステムの容量は、知らないうちにスナップショットによって消費されている場合があります。次の構文を使用して、すべてのデータセット情報を表示できます。

```
# zfs list -t all
```

- システムのインストール時に自動的に個別の `/var` ファイルシステムが作成されますが、このファイルシステムに割り当て制限と予約を設定して、ルートプールの容量が知らないうちに消費されないようにする必要があります。
- さらに、`fsstat` コマンドを使用して、ZFS ファイルシステムのファイル操作アクティビティーを表示できます。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 /
new  name   name  attr  attr lookup rddir  read read  write write
file remov  chng   get   set    ops   ops   ops bytes   ops bytes
832   589   286  837K 3.23K  2.62M 20.8K 1.15M 1.75G 62.5K  348M /
```

- バックアップ
  - ファイルシステムスナップショットを残します
  - 週単位または月単位のバックアップのためにエンタープライズレベルソフトウェアを検討します

- ベアメタル回復のために、リモートシステムにルートプールスナップショットを保存します

♦ ♦ ♦ 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 Oracle 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、次のセクションで構成されます。

ZFS バージョンの概要

Oracle 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 Oracle 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、プールおよびファイルシステム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、165 ページの「ZFS ファイルシステムをアップグレードする」および76 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、この Oracle Solaris 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。このリストは `zpool upgrade -v` コマンドを使用して作成できます。

バージョン	Oracle Solaris リリース	説明
1	S11 11/11	初期バージョンの ZFS
2	S11 11/11	Ditto ブロック (複製されたメタデータ)

バージョン	Oracle Solaris リリース	説明
3	S11 11/11	ホットスペアおよびダブルパリティー RAID-Z
4	S11 11/11	`zpool history`
5	S11 11/11	`gzip` 圧縮アルゴリズム
6	S11 11/11	`bootfs` プールプロパティー
7	S11 11/11	別のインテントログデバイス
8	S11 11/11	委任管理
9	S11 11/11	`refquota` および `refreservation` プロパティー
10	S11 11/11	キャッシュデバイス
11	S11 11/11	スクラブパフォーマンスの向上
12	S11 11/11	スナップショットプロパティー
13	S11 11/11	`snapused` プロパティー
14	S11 11/11	`aclinherit passthrough-x` プロパティー
15	S11 11/11	ユーザーおよびグループの領域の計上
16	S11 11/11	`stmf` プロパティー
17	S11 11/11	トリプルパリティー RAID-Z
18	S11 11/11	ユーザーによるスナップショットの保持
19	S11 11/11	ログデバイスの削除
20	S11 11/11	`zle` (長さゼロのエンコード) 圧縮アルゴリズム
21	S11 11/11	複製解除
22	S11 11/11	受信プロパティー
23	S11 11/11	スリム ZIL
24	S11 11/11	システム属性
25	S11 11/11	スクラブ統計の向上
26	S11 11/11	スナップショット削除パフォーマンスの向上
27	S11 11/11	スナップショット作成パフォーマンスの向上
28	S11 11/11	複数 vdev の交換
29	S11 11/11	RAID-Z/ミラーハイブリッドアロケータ
30	S11 11/11	暗号化
31	S11 11/11	「zfs list」パフォーマンスの向上
32	S11 11/11	1M バイトのブロックサイズ
33	S11 11/11	共有サポートの向上
34	S11.1	継承による共有
35	S11.2	順次再同期化
36	S11.3	効率的なログブロック割り当て
37	S11.3	LZ4 圧縮

ZFS ファイルシステムのバージョン

次の表に、この Oracle Solaris 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。特定のファイルシステムバージョンで使用可能な機能には、特定のプールが必要であることに留意してください。この一覧は、`zfs upgrade -v` コマンドを使用して作成できます。

バージョン	Oracle Solaris リリース	説明
1	S11 11/11	初期バージョンの ZFS ファイルシステム
2	S11 11/11	拡張されたディレクトリエントリ
3	S11 11/11	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	S11 11/11	`userquota` および `groupquota` プロパティー
5	S11 11/11	システム属性
6	S11.1	マルチレベルファイルシステムのサポート

用語集

B

ブート環境 ZFS ルートファイルシステム、およびオプションでその下にマウントされているその他のファイルシステムから成るブート可能な Oracle Solaris 環境。一度にアクティブにできるのは、1 つのブート環境だけです。

C

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。

クローンについては、175 ページの「ZFS クローンの概要」を参照してください。

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。チェックサム機能は、簡素で高速の fletcher4 (デフォルト) から SHA256 などの暗号面で強力なハッシュまで多岐にわたります。

D

データセット 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。

データセットの詳細は、第7章「Oracle Solaris ZFS ファイルシステムの管理」を参照してください。

複製解除 ZFS ファイルシステム内の重複したデータのブロックを削除するプロセス。重複したブロックを削除したあと、一意のブロックが複製解除テーブル内に格納されます。

F

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は、第7章「Oracle Solaris ZFS ファイルシステムの管理」を参照してください。

M

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。ミラー内のいずれかのディスクで障害が発生した場合、そのミラー内のほかのディスクが同じデータを提供できます。

P

プール　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は、第5章「Oracle Solaris ZFS ストレージプールの管理」を参照してください。

R

再同期化　データをあるデバイスから別のデバイスにコピーするプロセス。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。従来のボリューム管理製品では、このプロセスはミラー再同期化と呼ばれます。

ZFS 再同期化の詳細は、260 ページの「再同期化のステータスを表示する」を参照してください。

ルートプール　ブートファイルシステムを含む ZFS プール。

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 の詳細は、19 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。

S

スナップショット　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は、167 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。

V

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は、60 ページの「ZFS ストレージプールのステータスのクエリー検索を行う」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、227 ページの「ZFS ボリューム」を参照してください。

索引

さ

た

な

は

ま

ら

わ

A

B

C

D

E