Oracle® Solaris 11.1 の管理: ZFS ファイルシステム

Part No: E36656–05
2013 年 9 月

ORACLE

130924@25097

目次

はじめに

『Oracle Solaris 11.1 ZFS 管理ガイド』では、Oracle Solaris ZFS ファイルシステムの設定と管理に関する情報を提供します。

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

注 – この Oracle Solaris のリリースでは、SPARC および x86 系列のプロセッサアーキテクチャーをサポートしています。サポートされるシステムは、http://www.oracle.com/webfolder/technetwork/hcl/index.html の『Oracle Solaris Hardware Compatibility List』に記載されています。このドキュメントでは、プラットフォームにより実装が異なる場合は、それを特記します。

対象読者

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

内容の紹介

次の表で、本書の各章について説明します。

章	説明
第 1 章「Oracle Solaris ZFS ファイルシステム (概要)」	ZFS の概要およびその機能と利点について説明します。また、基本的な概念と用語について説明します。
第 2 章「Oracle Solaris ZFS 入門」	基本的なプールとファイルシステムを使って基本的な ZFS 構成を設定する手順について説明します。この章では、ZFS ファイルシステムの作成に必要なハードウェアとソフトウェアについても説明します。
第 3 章「Oracle Solaris ZFS ストレージプールの管理」	ZFS ストレージプールの作成および管理方法について詳しく説明します。

章	説明
第 4 章「ZFS ルートプールのコンポーネントの管理」	ミラー化ルートプールの構成、ZFS ブート環境のアップグレード、スワップデバイスおよびダンプデバイスのサイズ変更など、ZFS ルートプールのコンポーネントを管理する方法について説明します。
第 5 章「Oracle Solaris ZFS ファイルシステムの管理」	ZFS ファイルシステムの管理について詳しく説明します。たとえば、ファイルシステムの階層レイアウト、プロパティーが継承されること、およびマウントポイント管理および共有が自動的に行われることなどについて、それらの概念を説明します。
第 6 章「Oracle Solaris ZFS のスナップショットとクローンの操作」	ZFS のスナップショットとクローンを作成および管理する方法について説明します。
第 7 章「ACL および属性を使用した Oracle Solaris ZFS ファイルの保護」	アクセス制御リスト (ACL) を使用して UNIX 標準のアクセス権よりアクセス権を詳細に設定することで、ZFS ファイルを保護する方法について説明します。
第 8 章「Oracle Solaris ZFS 委任管理」	ZFS 委任管理を使用して、特権のないユーザーが ZFS 管理タスクを実行できるようにする方法について説明します。
第 9 章「Oracle Solaris ZFS の高度なトピック」	ZFS ボリュームの使用について、ゾーンがインストールされた Oracle Solaris システムでの ZFS の使用について、および代替ルートプールの使用について説明します。
第 10 章「Oracle Solaris ZFS のトラブルシューティングとプールの回復」	ZFS の障害を識別してそこから回復する方法について説明します。また、障害を回避する方法について説明します。
第 11 章「スナップショットのアーカイブとルートプールの回復」	ルートプールスナップショットをアーカイブする方法と、ルートプールの回復を実行する方法について説明します。
第 12 章「推奨の Oracle Solaris ZFS プラクティス」	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。
付録 A 「Oracle Solaris ZFS バージョンの説明」	利用可能な ZFS のバージョン、各バージョンの機能、および Solaris OS の各リリースで提供される ZFS のバージョンと機能について説明します。

関連情報

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

- 『Oracle Solaris 11.1 でのシステム情報、プロセス、およびパフォーマンスの管理』
- 『Oracle Solaris 11.1 のユーザーアカウントとユーザー環境の管理』

- 『Oracle Solaris 11.1 の管理: デバイスとファイルシステム』
- 『Oracle Solaris 11.1 の管理: セキュリティーサービス』

Oracle サポートへのアクセス

Oracle のお客様は、My Oracle Support を通じて電子的なサポートを利用することができます。詳細は、`http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` を参照してください。聴覚に障害をお持ちの場合は、`http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` を参照してください。

表記上の規則

次の表では、このドキュメントで使用される表記上の規則について説明します。

表 P–1 表記上の規則

字体	説明	例
`AaBbCc123`	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、コード例を示します。	`.login` ファイルを編集します。 `ls -a` を使用してすべてのファイルを表示します。 `machine_name% you have mail.`
`AaBbCc123`	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して示します。	`machine_name%` **`su`** `Password:`
aabbcc123	プレースホルダ: 実際に使用する特定の名前または値で置き換えます。	ファイルを削除するには、 `rm` *filename* と入力します。
AaBbCc123	書名、新しい単語、および強調する単語を示します。	『ユーザーズガイド』の第 6 章を参照してください。 キャッシュは、ローカルに格納されるコピーです。 ファイルを保存しないでください。 注: いくつかの強調された項目は、オンラインでは太字で表示されます。

コマンド例のシェルプロンプト

次の表に、Oracle Solaris OS に含まれるシェルの UNIX システムプロンプトおよびスーパーユーザーのプロンプトを示します。コマンド例のシェルプロンプトは、そのコマンドを標準ユーザーで実行すべきか特権ユーザーで実行すべきかを示します。

表 P–2　シェルプロンプト

シェル	プロンプト
Bash シェル、Korn シェル、および Bourne シェル	`$`
Bash シェル、Korn シェル、および Bourne シェルのスーパーユーザー	`#`
C シェル	`machine_name%`
C シェルのスーパーユーザー	`machine_name#`

◆ ◆ ◆ 第 1 章

Oracle Solaris ZFS ファイルシステム (概要)

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

この章は、次のセクションで構成されます。

ZFS の新機能

このセクションでは、ZFS ファイルシステムの新機能について概説します。

改善された ZFS プールデバイスメッセージ

Oracle Solaris 11.1: zpool status コマンドがデバイス障害に関するより詳細な情報を提供するように拡張されました。この例における zpool status の出力は、プールデバイス (c0t5000C500335F907Fd0) が永続的なエラーのために UNAVAIL ステータスになっていることを示しており、それを交換する必要があります。

```
# 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 ファイル共有の改善

Oracle Solaris 11.1: ZFS ファイルシステムの共有が次の主な拡張機能で改善されました。

- 共有の構文は簡素化されています。ファイルシステムを共有するには、新しい `share.nfs` または `share.smb` プロパティーを設定します。
- 子孫ファイルシステムへの共有プロパティーの継承の向上

ファイル共有の改善は、プールバージョン 34 と関連しています。

詳細は、184 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。

共有された var ファイルシステム

Oracle Solaris 11.1: Oracle Solaris 11.1 をインストールすると、`rpool/VARSHARE` ファイルシステムが自動的に作成されて、`/var/share` にマウントされます。このファイルシステムの目的は、ブート環境の間でファイルシステムを共有して、すべての BE で必要となる `/var` ディレクトリ内の容量を減らすことです。例:

```
# ls /var/share
audit  cores  crash  mail
```

互換性のために、上記の `/var` コンポーネントから `/var/share` コンポーネントへのシンボリックリンクが自動的に作成されます。このファイルシステムは通常は管理を必要としませんが、`/var` コンポーネントがルートファイルシステムを使い切ってしまわないようにしてください。

Oracle Solaris 11 システムを Oracle Solaris 11.1 リリースに更新した場合は、元の `/var` ディレクトリから `/var/share` ディレクトリへのデータの移行にいくらか時間がかかる可能性があります。

EFI (GPT) ラベル付きディスクのブートサポート

Oracle Solaris 11.1: このリリースでは、ほとんどの場合、x86 ベースシステムの ZFS ルートプールディスクに EFI (GPT) ディスクラベルがインストールされます。例:

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

    NAME      STATE     READ WRITE CKSUM
```

```
    rpool     ONLINE       0     0     0
      c2t0d0  ONLINE       0     0     0

errors: No known data errors
```

- GPT 対応ファームウェアを搭載した SPARC ベースのシステムまたは x86 ベースのシステムに Oracle Solaris 11.1 をインストールすると、ディスク全体を使用するルートプールディスクで GPT ディスクラベルが適用されます。GPT ラベル付きブートディスクをサポートする SPARC ベースのシステムでは、GPT 対応ファームウェアの適用について Oracle Solaris 11.1 リリースノートを参照してください。それ以外の場合は、SPARC システムに Oracle Solaris 11.1 をインストールすると、単一のスライス 0 を使用してルートプールディスクに SMI (VTOC) ラベルが適用されます。
- ほとんどの場合、x86 ベースシステムに EFI (GPT) ラベル付きディスクがインストールされます
- `zpool` コマンドは、EFI (GPT) ディスクラベルをサポートするように拡張され、システムのインストール後にルートプールを再作成する必要がある場合に、`zpool create -B` コマンドでそれができるようになりました。この新しいコマンドオプションにより、必須パーティションおよびブートに必要なブート情報が作成されます。インストール後にルートプールを作成する方法の詳細は、128 ページの「別のルートプール内で BE を作成する方法 (SPARC または x86/VTOC)」を参照してください。

  EFI (GPT) ラベル付きの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、`zpool replace` 操作後にこのコマンドを実行する必要があります。

  ```
  # bootadm install-bootloader
  ```
- x86 ベースシステムでは、Oracle Solaris のインストールはディスクの最初の 2Ti バイトに制限されなくなりました。

ZFS コマンドの使用法に関する拡張機能

Oracle Solaris 11: `zfs` および `zpool` コマンドには、`zfs` および `zpool` のサブコマンドとそのサポートされているオプションに関する詳細情報を表示するために使用できる `help` サブコマンドがあります。例:

```
# zfs help
The following commands are supported:
allow       clone       create      destroy     diff        get
groupspace  help        hold        holds       inherit     list
mount       promote     receive     release     rename      rollback
send        set         share       snapshot    unallow     unmount
unshare     upgrade     userspace
For more info, run: zfs help <command>
# zfs help create
usage:
        create [-p] [-o property=value] ... <filesystem>
        create [-ps] [-b blocksize] [-o property=value] ... -V <size> <volume>
```

```
# zpool help
The following commands are supported:
add      attach   clear    create   destroy  detach   export   get
help     history  import   iostat   list     offline  online   remove
replace  scrub    set      split    status   upgrade
For more info, run: zpool help <command>
# zpool help attach
usage:
        attach [-f] <pool> <device> <new-device>
```

詳細は、zfs(1M) および zpool(1M) を参照してください。

ZFS スナップショットに関する拡張機能

Oracle Solaris 11: このリリースには、ZFS スナップショットに関する次の拡張機能が含まれています。

- zfs snapshot コマンドには、このコマンドの短縮構文となる snap という別名があります。例:

  ```
  # zfs snap -r users/home@snap1
  ```

- zfs diff コマンドには、2 つのスナップショット間で追加または変更されたすべてのファイルを識別するための列挙型オプション -e があります。生成された出力には追加されたすべてのファイルが示されますが、削除されたものは表示されません。例:

  ```
  # zfs diff -e tank/cindy@yesterday tank/cindy@now
  +       /tank/cindy/
  +       /tank/cindy/file.1
  ```

  -o オプションを使用して、選択したフィールドを表示するように指定することもできます。例:

  ```
  # zfs diff -e -o size -o name tank/cindy@yesterday tank/cindy@now
  +       7       /tank/cindy/
  +       206695  /tank/cindy/file.1
  ```

ZFS スナップショットの作成の詳細は、第 6 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 マニュアルページの変更 (zfs.1m)

Oracle Solaris 11: zfs.1m マニュアルページは、主要な ZFS ファイルシステムの機能が zfs.1m ページに残るように改訂されていますが、委任管理、暗号化、および共有の構文と例については次のページに記載されています。

- zfs_allow(1M)
- zfs_encrypt(1M)
- zfs_share(1M)

改善された aclmode プロパティー

Oracle Solaris 11: aclmode プロパティーは、chmod 操作中にファイルのアクセス制御リスト (ACL) のアクセス権が変更されると必ず ACL 動作を変更します。aclmode プロパティーは次のプロパティー値とともに再導入されました。

- discard – aclmode プロパティーが discard であるファイルシステムでは、ファイルのモードを表さない ACL エントリがすべて削除されます。これがデフォルト値です。
- mask – aclmode プロパティーが mask であるファイルシステムでは、ユーザーまたはグループアクセス権が削減されます。アクセス権は、グループアクセス権ビットと同程度にまで低下します。ただし、アクセス権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きます。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 – aclmode プロパティーが passthrough であるファイルシステムでは、ファイルまたはディレクトリの新規モードを表す必須の ACL エントリを生成する以外、ACL に変更は加えられません。

詳細は、例 7–14 を参照してください。

物理的な場所でプールデバイスを指定する

Oracle Solaris 11: この Solaris リリースでは、zpool status -l コマンドを使用して、/dev/chassis ディレクトリから使用できるプールデバイスの物理ディスクの場所情報を表示します。このディレクトリには、システム上のデバイスのシャーシ、受容装置、および占有装置の値が含まれます。

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-J4400.0912QAJ001 SUN-Storage-J4400.rack22
```

例:

```
% zpool status -l export
  pool: export
 state: ONLINE
  scan: resilvered 492G in 8h22m with 0 errors on Wed Aug  1 17:22:11 2012
config:

 NAME                                                      STATE  READ WRITE CKSUM
   export                                                  ONLINE    0     0     0
     mirror-0                                              ONLINE    0     0     0
       /dev/chassis/SUN-Storage-J4400.rack22/SCSI_Device__2/disk  ONLINE    0     0     0
```

```
      /dev/chassis/SUN-Storage-J4400.rack22/SCSI_Device__3/disk  ONLINE    0     0     0
    mirror-1                                                    ONLINE    0     0     0
      /dev/chassis/SUN-Storage-J4400.rack22/SCSI_Device__4/disk  ONLINE    0     0     0
      /dev/chassis/SUN-Storage-J4400.rack22/SCSI_Device__5/disk  ONLINE    0     0     0
    mirror-2                                                    ONLINE    0     0     0
      /dev/chassis/SUN-Storage-J4400.rack22/SCSI_Device__6/disk  ONLINE    0     0     0
      /dev/chassis/SUN-Storage-J4400.rack22/SCSI_Device__7/disk  ONLINE    0     0     0
    mirror-3                                                    ONLINE    0     0     0
      /dev/chassis/SUN-Storage-J4400.rack22/SCSI_Device__8/disk  ONLINE    0     0     0
      /dev/chassis/SUN-Storage-J4400.rack22/SCSI_Device__9/disk  ONLINE    0     0     0
    mirror-4                                                    ONLINE    0     0     0
      /dev/chassis/SUN-Storage-J4400.rack22/SCSI_Device__10/disk ONLINE    0     0     0
      /dev/chassis/SUN-Storage-J4400.rack22/SCSI_Device__11/disk ONLINE    0     0     0
    spares
      /dev/chassis/SUN-Storage-J4400.rack22/SCSI_Device__0/disk  AVAIL
      /dev/chassis/SUN-Storage-J4400.rack22/SCSI_Device__1/disk  AVAIL

errors: No known data errors
```

`zpool iostat` コマンドはまた、プールのデバイスに関する物理的な場所情報を提供するように更新されています。

さらに、`diskinfo`、`format`、および `prtconf` コマンドもディスクの物理的な場所情報を提供します。詳細は、`diskinfo(1M)` を参照してください。

ZFS シャドウ移行

Oracle Solaris 11: このリリースでは、古いファイルシステムから新しいファイルシステムにデータを移行でき、それと同時に、移行プロセス中に新しいファイルシステムのアクセスおよび変更を行えます。

新しい ZFS ファイルシステムでシャドウプロパティーを設定すると、古いデータの移行が開始されます。シャドウプロパティーは、次のどちらかの値を指定して、データをローカルシステムから移行するか、リモートシステムから移行するかを設定できます。

`file:///`*path*
`nfs://`*host:path*

詳細は、209 ページの「ZFS ファイルシステムを移行する」を参照してください。

ZFS ファイルシステムの暗号化

Oracle Solaris 11: このリリースでは、ZFS ファイルシステムを暗号化できます。

たとえば `tank/cindy` ファイルシステムを、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/cindy
Enter passphrase for 'tank/cindy': xxx
Enter again: xxx
```

暗号化ポリシーは、ZFS ファイルシステムの作成時に設定されます。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。

詳細については、201 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。

送信ストリームに関する ZFS の拡張機能

Oracle Solaris 11: このリリースでは、スナップショットストリーム内で送信および受信されるファイルシステムプロパティーを設定できます。この拡張機能によって、受信側ファイルシステムへの送信ストリーム内でファイルシステムプロパティーを適用したり、受信時に `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を無視するかどうかを判別したりする柔軟性が提供されます。

詳細は、229 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。

ZFS スナップショットの相違点 (`zfs diff`)

Oracle Solaris 11: このリリースでは、`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 /tank/cindy
fileA
$ zfs snapshot tank/cindy@0913
$ ls /tank/cindy
fileA  fileB
$ zfs snapshot tank/cindy@0914
```

たとえば、2 つのスナップショットの相違点を識別するには、次のような構文を使用します。

```
$ zfs diff tank/cindy@0913 tank/cindy@0914
M       /tank/cindy/
+       /tank/cindy/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。

詳細は、220 ページの「ZFS スナップショットの相違点の識別 (`zfs diff`)」を参照してください。

ZFS ストレージプールの回復およびパフォーマンスに関する拡張機能

Oracle Solaris 11: このリリースでは、ZFS ストレージプールに関する次の新機能が提供されています。

- `zpool import -m` コマンドを使用して、ログのないプールをインポートできます。詳細は、108 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。
- 読み取り専用モードでプールをインポートできます。この機能は、主としてプールの回復に使用します。ベースとなるデバイスが損傷を受けているために損傷したプールにアクセスできない場合は、そのプールを読み取り専用でインポートしてデータを回復できます。詳細は、109 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。
- このリリースで作成される RAID-Z (`raidz1`、`raidz2`、または `raidz3`) ストレージプールでは、読み取り入出力のスループットパフォーマンスを改善するために、待ち時間の影響を受けやすいメタデータが自動的にミラー化されています。プールバージョン 29 以上にアップグレードされた既存の RAID-Z プールの場合、一部のメタデータは、新しく書き込まれるすべてのデータについてミラー化されます。

  RAID-Z プール内でミラー化されたメタデータは、ミラー化されたストレージプールによって提供される保護と同様、ハードウェア障害に対する追加の保護を提供しません。ミラー化されたメタデータによって追加容量が消費されますが、RAID-Z 保護は以前のリリースと同じままになります。この拡張機能は、パフォーマンスのみを目的としています。

ZFS 同期動作の調整

Oracle Solaris 11: このリリースでは、`sync` プロパティーを使用することによって、ZFS ファイルシステムの同期動作を決定できます。

デフォルトの同期動作では、データの安定性を確保するために、ファイルシステムのすべて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、すべてのデバイスがフラッシュされます。デフォルトの同期動作を無効にすることはお勧めしません。同期サポートに依存しているアプリケーションが影響を受けることがあり、データ損失が起きる可能性があります。

`sync` プロパティーは、ファイルシステムを作成する前または作成した後に設定できます。いずれの場合でも、プロパティー値はすぐに有効になります。例:

```
# zfs set sync=always tank/neil
```

`sync` プロパティーが追加された Oracle Solaris リリースでは、`zil_disable` パラメータが使用できなくなりました。

詳細は、表 5–1 を参照してください。

改善された ZFS プールメッセージ

Oracle Solaris 11: このリリースでは、-T オプションを使用して zpool list および zpool status コマンドの間隔とカウント値を指定することで、追加情報を表示できるようになりました。

また、zpool status コマンドによって、次のようなプールのスクラブと再同期化に関するより多くの情報が表示されるようになりました。

- 再同期化進捗レポート。例:

```
scan: resilver in progress since Thu Jun  7 14:41:11 2012
    3.83G scanned out of 73.3G at 106M/s, 0h11m to go
    3.80G resilvered, 5.22% done
```

- スクラブ進捗レポート。例:

```
scan: scrub in progress since Thu Jun  7 14:59:25 2012
    1.95G scanned out of 73.3G at 118M/s, 0h10m to go
    0 repaired, 2.66% done
```

- 再同期化完了メッセージ。例:

```
resilvered 73.3G in 0h13m with 0 errors on Thu Jun  7 14:54:16 2012
```

- スクラブ完了メッセージ。例:

```
scan: scrub repaired 512B in 1h2m with 0 errors on Thu Jun  7 15:10:32 2012
```

- 進行中のスクラブの取り消しメッセージ。例:

```
scan: scrub canceled on Thu Jun  7 15:19:20 MDT 2012
```

- スクラブおよび再同期化の完了メッセージはシステムのリブート後も残ります。

次の構文では、進行中のプール再同期化情報を表示するための間隔およびカウントオプションを使用しています。-T d 値を使用すると情報を標準の日付形式で表示でき、-T u を使用すると情報を内部形式で表示できます。

```
# zpool status -T d tank 3 2
Thu Jun 14 14:08:21 MDT 2012

  pool: tank
 state: DEGRADED
status: One or more devices is currently being resilvered.  The pool will
        continue to function in a degraded state.
action: Wait for the resilver to complete.
        Run 'zpool status -v' to see device specific details.
  scan: resilver in progress since Thu Jun 14 14:08:05 2012
    2.96G scanned out of 4.19G at 189M/s, 0h0m to go
    1.48G resilvered, 70.60% done
config:

        NAME                        STATE     READ WRITE CKSUM
```

```
        tank                      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 DEGRADED     0     0     0  (resilvering)

errors: No known data errors
```

ACL 相互運用性に関する ZFS の拡張機能

Oracle Solaris 11: このリリースでは、ACL に関する次の拡張機能が提供されています。

- 例外的なアクセス権を除き、簡易 ACL には deny アクセス制御エントリ (ACE) が必要なくなりました。たとえば、`0644`、`0755`、または `0664` のモードでは deny ACE が不要ですが、`0705` や `0060` などのモードでは deny ACE が必要です。

  以前の動作では、`644` などの簡易 ACL 内に deny ACE を含みます。例:

  ```
  # ls -v file.1
  -rw-r--r--   1 root     root      206663 Jun 14 11:52 file.1
       0:owner@:execute:deny
       1:owner@:read_data/write_data/append_data/write_xattr/write_attributes
           /write_acl/write_owner:allow
       2:group@:write_data/append_data/execute:deny
       3:group@:read_data:allow
       4:everyone@:write_data/append_data/write_xattr/execute/write_attributes
           /write_acl/write_owner:deny
       5:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
           :allow
  ```

  `644` などの簡易 ACL についての新しい動作では、deny ACE を含みません。例:

  ```
  # ls -v file.1
  -rw-r--r--   1 root     root      206663 Jun 22 14:30 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/read_xattr/read_attributes/read_acl/synchronize:allow
       2:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
           :allow
  ```

- 元の変更されていないアクセス権の保持を図るために、継承時に ACL が複数の ACE に分割されなくなりました。代わりに、ファイル作成モードを強制する必要があるときに、アクセス権が変更されます。
- `aclinherit` プロパティーを `restricted` に設定したときのこのプロパティーの動作にアクセス権の削減が追加されました。これにより、継承時に ACL が複数の ACE に分割されなくなります。

- アクセス権モードの新しい計算規則では、ACL に含まれる *user* ACE がファイルの所有者でもある場合、そのアクセス権がアクセス権モードの計算に含まれることが指定されます。*group* ACE がファイルのグループ所有者である場合も同じ規則が適用されます。

詳細は、第 7 章「ACL および属性を使用した Oracle Solaris ZFS ファイルの保護」を参照してください。

ミラー化された ZFS ストレージプールの分割 (zpool split)

Oracle Solaris 11: このリリースでは、`zpool split` コマンドを使用して、ミラー化されたストレージプールを分割できます。これにより、元のミラー化プール内の 1 つまたは複数のディスクが切り離され、別の同一のプールが作成されます。

詳細は、72 ページの「ミラー化 ZFS ストレージプールを分割して新しいプールを作成する」を参照してください。

ZFS iSCSI の変更

Oracle Solaris 11: このリリースでは、Common Multiprotocol SCSI Target (COMSTAR) ターゲットデーモンを使用することにより、iSCSI ターゲットデーモンが置き換えられます。 この変更はまた、iSCSI LUN として ZFS ボリュームを共有するために使用されていた `shareiscsi` プロパティーが、もはや使用されなくなったことを意味します。`stmfadm` コマンドを使用して、iSCSI LUN として ZFS ボリュームを構成し共有します。

詳細は、281 ページの「iSCSI LUN として ZFS ボリュームを使用する」を参照してください。

新しい ZFS システムプロセス

Oracle Solaris 11: このリリースでは、個々の ZFS ストレージプールに `zpool-`*poolname* という名前のプロセスが関連付けられます。このプロセス内のスレッドは、プールと関連付けられた圧縮やチェックサム検証などの入出力タスクを処理するための、プールの入出力処理スレッドです。このプロセスの目的は、各ストレージプールの CPU 使用率を目に見えるようにすることです。

これらの実行中のプロセスに関する情報は、`ps` および `prstat` コマンドを使って確認できます。このプロセスは、大域ゾーンでのみ利用できます。詳細は、`SDC(7)`を参照してください。

ZFS 複製解除プロパティー

Oracle Solaris 11: このリリースでは、複製解除 (dedup) プロパティーを使用して、ZFS ファイルシステムから冗長なデータを削除できます。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。

次のようにしてこのプロパティーを有効にできます。

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

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、次のようにして複製解除比を指定できます。

```
# zpool list tank
NAME  SIZE  ALLOC  FREE  CAP  DEDUP  HEALTH  ALTROOT
tank  556G  4.19G  552G   0%  1.00x  ONLINE  -
```

zpool list 出力は、複製解除プロパティーをサポートするように更新されています。

複製解除プロパティーの設定方法については、167 ページの「dedup プロパティー」を参照してください。

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。

- 複製解除による容量のセクション約によってデータにメリットがあるかどうかを判断します。
- 複製解除をサポートできるだけ十分な物理的メモリーがシステムにあるかどうかを判断します。
- システムパフォーマンスへの影響の可能性

これらの考慮事項の詳細については、167 ページの「dedup プロパティー」を参照してください。

Oracle Solaris ZFS とは

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

プールされた **ZFS** ストレージ

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

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

トランザクションのセマンティクス

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

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

チェックサムと自己修復データ

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

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

優れたスケーラビリティー

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

ZFS スナップショット

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

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

簡素化された管理

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

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

ZFS の用語

このセクションでは、このドキュメントで使用される基本的な用語について説明します。

ブート環境
: ブート環境とは、ブート可能な Oracle Solaris 環境のことであり、ZFS ルートファイルシステムと、必要に応じてその下にマウントされるほかのファイルシステムで構成されます。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。

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

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

  クローンの詳細については、221 ページの「ZFS クローンの概要」を参照してください。

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。データセットは、次の形式を使用して識別されます。

  pool/*path*[@*snapshot*]

  pool
  : データセットを格納するストレージプールの名前

  path
  : データセットコンポーネントのスラッシュ区切りのパス名

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

  データセットの詳細については、第 5 章「Oracle Solaris ZFS ファイルシステムの管理」を参照してください。

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

  ファイルシステムの詳細については、第 5 章「Oracle Solaris ZFS ファイルシステムの管理」を参照してください。

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

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については、第 3 章「Oracle Solaris ZFS ストレージプールの管理」を参照してください。

RAID-Z
: データとパリティーを複数のディスクに格納する仮想デバイス。RAID-Z の詳細については、50 ページの「RAID-Z ストレージプール構成」を参照してください。

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

	ZFS の再同期化の詳細については、316 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。
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については、213 ページの「ZFS スナップショットの概要」を参照してください。
仮想デバイス	プール内の論理デバイス。物理デバイス、ファイル、または一連のデバイスを仮想デバイスに設定できます。 仮想デバイスの詳細については、59 ページの「ストレージプールの仮想デバイスの情報を表示する」を参照してください。
ボリューム	ブロックデバイスを表すデータセット。たとえば、スワップデバイスとして ZFS ボリュームを作成できます。 ZFS ボリュームの詳細については、279 ページの「ZFS ボリューム」を参照してください。

ZFS コンポーネントに名前を付けるときの規則

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

- 各コンポーネントに使用できる文字は、英数字および次の 4 つの特殊文字だけです。
  - 下線 (_)
  - ハイフン (-)
  - コロン (:)
  - ピリオド (.)
- プール名の先頭は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。

また、コンポーネント名を空にすることはできません。

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

ZFS ファイルシステムの構造

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

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

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

ファイルシステム階層の作成方法を示す例については、41 ページの「ZFS ファイルシステム階層を作成する」を参照してください。

ZFS のディスク領域の計上

ZFS は、プールストレージの概念に基づいて構成されます。標準的なファイルシステムでは物理ストレージにマッピングされますが、ZFS ファイルシステムはすべてがプールの中にあって、プール内で使用可能なストレージを共有しています。このため、`df` などのユーティリティーから報告される使用可能なディスク領域は、ファイ

ルシステムがアクティブでないときでも変化する可能性があります。これは、プール内のほかのファイルシステムがディスク領域を消費したり解放したりするためです。

ファイルシステムの最大サイズは、割り当て制限を使用して制限できます。割り当て制限の詳細については、196 ページの「ZFS ファイルシステムに割り当て制限を設定する」を参照してください。予約を使用すれば、指定されたディスク容量をファイルシステムに保証することができます。予約については、199 ページの「ZFS ファイルシステムに予約を設定する」を参照してください。このモデルは、NFS モデルによく似ています。つまり、複数のディレクトリが 1 つのファイルシステム (`/home` など) からマウントされます。

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

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

UFS ファイルシステムでは、`du` コマンドが、ファイル内のデータブロックのサイズを報告します。ZFS ファイルシステムでは、`du` コマンドが、ディスクに格納されているときのファイルの実際のサイズを報告します。このサイズにはメタデータと圧縮データも含まれます。実際、この報告は「このファイルを削除した場合、どれだけの容量を得られるか」という質問に回答するときに役立ちます。したがって、圧縮がオフになっている場合でも、ZFS と UFS では異なる結果が表示されます。

`df` コマンドで報告される使用容量を `zfs list` コマンドの結果と比べる場合、`df` で報告される容量がファイルシステムそのもののサイズではなく、プールサイズであることを考慮してください。さらに `df` は、子孫のファイルシステムを認識せず、スナップショットが存在するかどうかも認識しません。圧縮や割り当て制限などの ZFS プロパティーがファイルシステムで設定されている場合は、`df` によって報告された使用容量を調整しようとしても困難になる可能性があります。

報告される使用容量に影響する可能性もある次の状況を考慮してください。

- recordsize より大きなファイルの場合、ファイルの最後のブロックは通常、約 1/2 になります。デフォルトの recordsize が 128K バイトに設定されている場合、ファイルあたり約 64K バイトが無駄になり、大きな影響になることがあります。RFE 6812608 の統合がこの状況を解決します。圧縮を有効にすることにより、これを回避できます。データがすでに圧縮されている場合でも、最後のブロックの未使用部分は 0 で埋められ、圧縮率が高くなります。
- RAIDZ-2 プールでは、すべてのブロックは、パリティー情報に少なくとも 2 つのセクター (512 バイトのチャンク) を使用します。パリティー情報に使用される容量は報告されません。この容量は変動し、小さなブロックの場合は非常に大きな割合を占める可能性があるので、容量の報告への影響が見られる場合があります。この影響は recordsize を 512 バイトに設定したときに非常に極端になります。この場合、512 バイトの論理ブロックごとに 1.5K バイト (領域の 3 倍) が消費されます。格納されるデータには関係なく、領域効率に主な関心がある場合は、recordsize をデフォルト (128K バイト) のままにして、圧縮を有効にしてください (デフォルトの lzjb に設定)。
- df コマンドは、複製解除したファイルデータを認識しません。

領域が不足した場合の動作

ZFS では、ファイルシステムのスナップショットを負荷をかけずに簡単に作成できます。スナップショットは、ほとんどの ZFS 環境でよく使用されます。ZFS スナップショットについては、第 6 章「Oracle Solaris ZFS のスナップショットとクローンの操作」を参照してください。

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

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

ZFS ファイルシステムをマウントする

ZFS を使えば複雑さが減り、管理が容易になります。たとえば、従来のファイルシステムでは、新しいファイルシステムを追加するたびに /etc/vfstab ファイルを編集す

る必要があります。ZFS では、ファイルシステムのプロパティーに基づいてファイルシステムのマウントとアンマウントを自動的に行うことで、この作業を不要にしました。/etc/vfstab ファイルの ZFS エントリを管理する必要はありません。

ZFS ファイルシステムのマウントおよび共有の詳細については、179 ページの「ZFS ファイルシステムをマウントする」を参照してください。

従来のボリューム管理

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

NFSv4 に基づく Solaris ACL モデル

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

新しい Solaris ACL モデルは、主に次の点が異なっています。

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

ZFS ファイルで ACL を使用するときの詳細については、第 7 章「ACL および属性を使用した Oracle Solaris ZFS ファイルの保護」を参照してください。

◆ ◆ ◆ 第 2 章

Oracle Solaris ZFS 入門

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

この章は、次のセクションで構成されます。

ZFS 権利プロファイル

スーパーユーザー (root) アカウントを使用しないで ZFS 管理タスクを実行する必要がある場合は、次のいずれかのプロファイルが割り当てられた役割引き受けて ZFS 管理タスクを実行できます。

- ZFS ストレージ管理 – ZFS ストレージプール内でデバイスを作成、破棄、および操作できます
- ZFS ファイルシステム管理 – ZFS ファイルシステムを作成、破棄、および変更できます

役割の作成または割り当ての詳細は、『Oracle Solaris 11.1 の管理: セキュリティーサービス』を参照してください。

RBAC の役割を使用して ZFS ファイルシステムを管理するほかに、ZFS 委任管理を使用して ZFS 管理タスクを分散することも検討できます。詳細は、第 8 章「Oracle Solaris ZFS 委任管理」を参照してください。

ZFS のハードウェアとソフトウェアに関する要件および推奨要件

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

- 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 ベースのシステムを使用します。
- ストレージプールに必要な最小ディスク容量は、64M バイトです。最小ディスクサイズは 128M バイトです。
- ZFS のパフォーマンスを高めるには、ワークロードに基づいて必要なメモリーのサイズを設定してください。
- ミラー化プール構成を作成する場合は複数のコントローラを使用します。

基本的な ZFS ファイルシステムを作成する

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

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

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

冗長な ZFS プール構成の詳細については、50 ページの「ZFS ストレージプールの複製機能」を参照してください。

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

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

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

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

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

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

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

通常、ファイルシステムの階層を作成して編成するときには、組織の要件に一致させることをお勧めします。ZFS ファイルシステム階層の作成方法については、41 ページの「ZFS ファイルシステム階層を作成する」を参照してください。

基本的な ZFS ストレージプールを作成する

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

▼ ZFS ストレージプールのストレージ要件を確認する方法

1. ストレージプールに使用可能なデバイスを決定します。

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

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

   ディスクの詳細、および使用方法と名前の付け方については、45 ページの「ZFS ストレージプール内でディスクを使用する」を参照してください。

2. データの複製を選択します。

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

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

ZFS の複製機能の詳細については、50 ページの「ZFS ストレージプールの複製機能」を参照してください。

▼ ZFS ストレージプールを作成する方法

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

ZFS 権利プロファイルの詳細については、37 ページの「ZFS 権利プロファイル」を参照してください。

2 ストレージプールの名前を選択します。

この名前は、`zpool` または `zfs` コマンドの使用時にストレージプールを識別するために使用されます。任意のプール名を選択できますが、32 ページの「ZFS コンポーネントに名前を付けるときの規則」の命名要件を満たしている必要があります。

3 プールを作成します。

たとえば次のコマンドは、`tank` という名前のミラープールを作成します。

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

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

ストレージプールの作成方法については、53 ページの「ZFS ストレージプールを作成する」を参照してください。デバイスの使用状況を確認する方法の詳細については、61 ページの「使用中のデバイスを検出する」を参照してください。

4 結果を確認します。

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

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

プールのステータスを確認する方法の詳細については、90 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 ファイルシステム階層を作成する

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

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

▼ ZFS ファイルシステム階層を決定する方法

1 ファイルシステムの構造を選択します。

ZFS の管理は、ファイルシステムに基づいて行います。このファイルシステムは、軽量で、簡単に作成できます。ファイルシステムは、ユーザーまたはプロジェクトごとに作成することをお勧めします。このモデルを使えば、プロパティー、スナップショット、およびバックアップをユーザーまたはプロジェクト単位で制御することができます。

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

ファイルシステムの管理方法の詳細については、第 5 章「Oracle Solaris ZFS ファイルシステムの管理」を参照してください。

2 属性が似ているファイルシステムをグループにまとめます。

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

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

3 ファイルシステムのプロパティーを選択します。

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

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

プロパティーの詳細については、147 ページの「ZFS のプロパティーの紹介」を参照してください。

▼ ZFS ファイルシステムを作成する方法

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

ZFS 権利プロファイルの詳細については、37 ページの「ZFS 権利プロファイル」を参照してください。

2 必要な階層を作成します。

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

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

3 継承されるプロパティーを設定します。

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

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

ファイルシステムの作成時にファイルシステムのプロパティーを設定できます。例:

```
# zfs create -o mountpoint=/export/zfs -o share.nfs=on -o compression=on tank/home
```

プロパティーおよびプロパティーの継承の詳細は、147 ページの「ZFS のプロパティーの紹介」を参照してください。

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

4 ファイルシステムを個別に作成します。

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

```
# zfs create tank/home/jeff
# zfs create tank/home/bill
```

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

ファイルシステムの作成方法の詳細については、144 ページの「ZFS ファイルシステムを作成する」を参照してください。

ファイルシステムのマウントおよび共有の詳細については、179 ページの「ZFS ファイルシステムをマウントする」を参照してください。

5 ファイルシステム固有のプロパティーを設定します。

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

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

6 結果を確認します。

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

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

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

ファイルシステムのステータスの詳細については、171 ページの「ZFS ファイルシステムの情報のクエリー検索を行う」を参照してください。

ディスク領域がどのように使用および計算されるかの詳細については、33 ページの「ZFS のディスク領域の計上」を参照してください。

◆ ◆ ◆ 第 3 章

Oracle Solaris ZFS ストレージプールの管理

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

この章は、次のセクションで構成されます。

ZFS ストレージプールのコンポーネント

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

ZFS ストレージプール内でディスクを使用する

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

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

```
Current partition table (original):
Total disk sectors available: 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 11.1 をインストールすると、ほとんどの場合、EFI (GPT) ラベルが x86 ベースシステムのルートプールディスクに適用され、次のように表示されます。

```
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
```

上記の出力では、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、それ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。

Oracle Solaris 11.1 とともにインストールされた更新済みのファームウェアを搭載した SPARC システムでは、EFI (GPT) ディスクラベルが適用されます。例:

```
Current partition table (original):
Total disk sectors available: 143358320 + 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
```

ストレージプールのディスク全体を使用する場合は、次の考慮事項を確認してください。

- ディスク全体を使用する場合、通常、`/dev/dsk/cNtNdN` 命名規則を使用してディスクに名前が付けられています。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場合は、ディスクの名前を手動で付けて、ZFS にスライスを渡す必要があります。
- 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。Solaris `fdisk` パーティションの作成または変更の詳細は、『Oracle Solaris 11.1 の管理: デバイスとファイルシステム』の「ZFS ファイルシステム用のディスクの設定 (タスクマップ)」を参照してください。
- ディスク全体を使ってストレージプールを作成するときは、EFI ラベルが適用されます。EFI ラベルの詳細は、『Oracle Solaris 11.1 の管理: デバイスとファイルシステム』の「EFI (GPT) ディスクラベル」を参照してください。
- Oracle Solaris 11.1 インストーラは、ほとんどの場合に、GPT 対応ファームウェアを搭載した SPARC ベースのシステムおよび x86 ベースのシステムでルートプールディスクの EFI (GPT) ラベルを適用します。詳細は、116 ページの「ZFS ルートプールの一般的な要件」を参照してください。

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

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

ZFS ストレージプール内でスライスを使用する

ディスクスライスを使ってストレージプールを作成するときは、ディスクにレガシー Solaris VTOC (SMI) ラベルを付けることができますが、ディスクスライスの管理はより困難であるため、プールにディスクスライスを使用することはお勧めしません。

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

```
# format
.
.
```

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

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

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

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

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

x86 ベースのシステムには `fdisk` パーティションも存在します。`fdisk` パーティションは `/dev/dsk/cN[tN]dNpN` というデバイス名で示され、ディスクの使用可能なスライスのコンテナとして機能します。`cN[tN]dNpN` デバイスを ZFS ストレージプールコンポーネントとして使用する構成は、テストされておらずサポートもされていないため、使用しないでください。

ZFS ストレージプール内のファイルを使用する

ZFS では、ストレージプール内のファイルを仮想デバイスとして使用することもできます。この機能は、本稼働環境で使用するのではなく、主にテストや簡単な実験のために使用します。

- 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。
- 別の ZFS プール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可能性があります。

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

ZFS ストレージプールに関する考慮事項

ZFS ストレージプールを作成して管理する場合は、次の考慮事項を確認してください。

- 物理ディスクの全体を使用するのが、ZFS ストレージプールを作成するためのもっとも簡単な方法です。ディスクスライス、ハードウェア RAID アレイ内の LUN、またはソフトウェアベースのボリュームマネージャーが提供するボリュームからプールを構築する場合、管理、信頼性、およびパフォーマンスの観点から ZFS 構成が次第により複雑になります。次の点を考慮すれば、ほかのハードウェアまたはソフトウェアストレージ解決策を使って ZFS を構成する方法を決定しやすくなる可能性があります。
  - ハードウェア RAID アレイの LUN 上に ZFS 構成を構築する場合、ZFS の冗長機能とアレイが提供する冗長機能との関係を理解する必要があります。ある構成で十分な冗長性やパフォーマンスが得られても、別の構成ではそうならない可能性もあります。
  - 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表現されたボリュームを使用して、ZFS の論理デバイスを構成できます。ただし、そうした構成は推奨されません。ZFS はこのようなデバイス上でも正しく動作しますが、最適なパフォーマンスが得られない場合があります。

    ストレージプールの推奨事項の詳細については、第 12 章「推奨の Oracle Solaris ZFS プラクティス」を参照してください。
- 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。

ZFS ストレージプールの複製機能

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

ミラー化されたストレージプール構成

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

```
mirror c1t0d0 c2t0d0
```

概念的により複雑なミラー化構成は、次のようになります。

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

ミラー化されたストレージプールの作成方法については、53 ページの「ミラー化されたストレージプールを作成する」を参照してください。

RAID-Z ストレージプール構成

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

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

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

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

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

RAID-Z 構成はサイズ X のディスク N 基とパリティーディスク P 基で構成されているので、約 (N-P)*X バイトを保管することができ、P 個のデバイスで障害が発生してもデータの完全性が低下することがありません。シングルパリティーの RAID-Z 構成には 2 基以上のディスク、ダブルパリティーの RAID-Z 構成には 3 基以上のディスク (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。

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

```
raidz c1t0d0 c2t0d0 c3t0d0
```

概念的により複雑な RAID-Z 構成は、次のようになります。

```
raidz c1t0d0 c2t0d0 c3t0d0 c4t0d0 c5t0d0 c6t0d0 c7t0d0
raidz c8t0d0 c9t0d0 c10t0d0 c11t0d0c12t0d0 c13t0d0 c14t0d0
```

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

RAID-Z ストレージプールの作成方法については、55 ページの「RAID-Z ストレージプールを作成する」を参照してください。

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

```
http://blogs.oracle.com/roch/entry/when_to_and_not_to
```

RAID-Z ストレージプールの推奨事項の詳細については、第 12 章「推奨の Oracle Solaris ZFS プラクティス」を参照してください。

ZFS ハイブリッドストレージプール

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

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

冗長構成の自己修復データ

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

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

ストレージプール内の動的なストライプ

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

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

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

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

ZFS ストレージプールを作成および破棄する

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

- 53 ページの「ZFS ストレージプールを作成する」
- 59 ページの「ストレージプールの仮想デバイスの情報を表示する」
- 61 ページの「ZFS ストレージプールの作成エラーに対応する」
- 64 ページの「ZFS ストレージプールを破棄する」

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

ZFS ストレージプールを作成する

ストレージプールを作成するには、`zpool create` コマンドを使用します。このコマンドの引数には、プール名および任意の数の仮想デバイスを指定します。プール名は、32 ページの「ZFS コンポーネントに名前を付けるときの規則」の規則に従って付ける必要があります。

基本的なストレージプールを作成する

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

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

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

ミラー化されたストレージプールを作成する

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

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

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

推奨のミラー化構成の詳細については、第 12 章「推奨の Oracle Solaris ZFS プラクティス」を参照してください。

現時点では、ZFS ミラー化構成では次の操作がサポートされています。

- 最上位レベルの追加の仮想デバイス (vdev) 用の別のディスクセットを既存のミラー化構成に追加する。詳細は、66 ページの「ストレージプールにデバイスを追加する」を参照してください。

- 追加のディスクを既存のミラー化構成に接続する。あるいは、追加のディスクを複製されない構成に接続して、ミラー化構成を作成する。詳細は、71 ページの「ストレージプール内でデバイスを接続する/切り離す」を参照してください。
- 置き換えたあとのディスクのサイズが置き換える前のデバイスと等しいかそれ以上であれば、既存のミラー化構成内の 1 つ以上のディスクを置き換える。詳細は、78 ページの「ストレージプール内のデバイスを置き換える」を参照してください。
- 残りのデバイスがミラー化構成に十分な冗長性を備えているのであれば、その構成に含まれる 1 つのディスクを切り離す。詳細は、71 ページの「ストレージプール内でデバイスを接続する/切り離す」を参照してください。
- ディスクのうちの 1 つを切り離すことによってミラー化構成を分割し、新しい同一のプールを作成する。詳細は、72 ページの「ミラー化 ZFS ストレージプールを分割して新しいプールを作成する」を参照してください。

スペア、ログデバイス、キャッシュデバイス以外のデバイスは、ミラー化されたストレージプールから完全に削除できません。

ZFS ルートプールを作成する

次のルートプール構成要件を考慮してください。

- Oracle Solaris 11.1 では、ルートプールに使用されるディスクは、x86 ベースのシステムまたは GPT 対応ファームウェアを搭載したサポートされる SPARC システムに EFI (GPT) ラベル付きでインストールされます。または、SMI (VTOC) ラベルは、GPT 対応ファームウェアが搭載されていない SPARC ベースのシステムで適用されます。Oracle Solaris 11.1 インストーラは、可能であれば EFI (GPT) ラベルを適するため、インストール後に ZFS ルートプールを再作成する必要がある場合は、次のコマンドを使用して EFI (GPT) ディスクラベルと適切なブート情報を適用できます。

  ```
  # zpool create -B rpool2 c1t0d0
  ```
- ルートプールは、ミラー化構成または単一ディスク構成として作成する必要があります。`zpool add` コマンドを使ってディスクを追加し、複数のミラー化された最上位レベル仮想ディスクを作成することはできませんが、ミラー化された仮想デバイスを `zpool attach` コマンドを使って拡張することは可能です。
- RAID-Z やストライプ化構成はサポートされていません。
- ルートプールに別個のログデバイスを使用することはできません。
- ルートプールにサポートされていない構成を使用しようとすると、次のようなメッセージが表示されます。

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

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

ZFS ルートファイルシステムをインストールしブートする方法については、第 4 章「ZFS ルートプールのコンポーネントの管理」を参照してください。

RAID-Z ストレージプールを作成する

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

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

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

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

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

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

errors: No known data errors

# zpool create tank raidz3 c0t0d0 c1t0d0 c2t0d0 c3t0d0 c4t0d0
c5t0d0 c6t0d0 c7t0d0 c8t0d0
# zpool status -v tank
  pool: tank
 state: ONLINE
 scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        tank        ONLINE       0     0     0
          raidz3-0  ONLINE       0     0     0
            c0t0d0  ONLINE       0     0     0
            c1t0d0  ONLINE       0     0     0
            c2t0d0  ONLINE       0     0     0
            c3t0d0  ONLINE       0     0     0
            c4t0d0  ONLINE       0     0     0
```

```
            c5t0d0  ONLINE       0     0     0
            c6t0d0  ONLINE       0     0     0
            c7t0d0  ONLINE       0     0     0
            c8t0d0  ONLINE       0     0     0
errors: No known data errors
```

現時点では、ZFS RAID-Z 構成では次の操作がサポートされています。

- 最上位レベルの追加の仮想デバイス用の別のディスクセットを既存の RAID-Z 構成に追加する。詳細は、66 ページの「ストレージプールにデバイスを追加する」を参照してください。
- 置き換えたあとのディスクのサイズが置き換える前のデバイスと等しいかそれ以上であれば、既存の RAID-Z 構成内の 1 つ以上のディスクを置き換える。詳細は、78 ページの「ストレージプール内のデバイスを置き換える」を参照してください。

現時点では、RAID-Z 構成では次の操作がサポートされていません。

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

RAID-Z 構成の詳細については、50 ページの「RAID-Z ストレージプール構成」を参照してください。

ログデバイスを持つ ZFS ストレージプールを作成する

同期トランザクションの POSIX 要件を満たすために、ZFS インテントログ (ZIL) が提供されています。たとえば、多くの場合、データベースがシステムコールから戻るときは、そのトランザクションが安定したストレージデバイス上に置かれている必要があります。NFS およびほかのアプリケーションでは、`fsync()` も使用して、データの安定を確保します。

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。しかし、NVRAM や専用ディスクなどで、別個のインテントログデバイスを使用することにより、パフォーマンスを向上できる可能性があります。

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

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

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

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

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

```
# zpool create datap mirror 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
```

ログデバイス障害からの回復の詳細については、例 10–2 を参照してください。

キャッシュデバイスを使用して ZFS ストレージプールを作成する

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

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

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

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

errors: No known data errors
```

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

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

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

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

- キャッシュデバイスを使用すると、ほぼ静的なコンテンツをランダムに読み込む作業負荷のパフォーマンスが大幅に向上します。
- `zpool iostat` コマンドを使用して、容量と読み込みをモニターできます。

- プールの作成時に単一または複数のキャッシュデバイスを追加できます。プールの作成後にキャッシュデバイスを追加または削除することもできます。詳細は、例 3–4 を参照してください。
- キャッシュデバイスは、ミラー化することも、RAID-Z 構成に含めることもできません。
- キャッシュデバイスで読み取りエラーが検出されると、ミラー化構成または RAID-Z 構成に含まれている可能性があるオリジナルのストレージプールデバイスに対して、その読み取り I/O が再発行されます。キャッシュデバイスの内容は、ほかのシステムキャッシュと同様に揮発的とみなされます。

ストレージプールを作成する場合の注意事項

ZFS ストレージプールを作成して管理する場合は、次の注意事項を確認してください。

- 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。ルートプールディスクのパーティションやラベルを変更しようとすると、OS の再インストールが必要になる場合があります。
- 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。このようなサポートされていない構成では、デッドロックが発生することがあります。
- 単一のスライスまたは単一のディスクを使用して作成されたプールには冗長性がなく、データ損失のリスクがあります。複数のスライスを使用してプールを作成しても、冗長性がなければ、やはりデータ損失のリスクがあります。複数のディスクにまたがる複数のスライスを使用して作成したプールは、ディスク全体を使用して作成したプールより管理が難しくなります。
- ZFS 冗長性 (RAIDZ またはミラー) を使用して作成されていないプールでは、データの不一致が報告されるだけです。データの不一致は修正できません。
- ZFS 冗長性を使用して作成されたプールは、ハードウェア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。エンタープライズグレード以外のハードウェアでは、プールデータの定期的なバックアップを実行することが重要です。
- プールはシステム間で共有できません。ZFS はクラスタファイルシステムではありません。

ストレージプールの仮想デバイスの情報を表示する

個々のストレージプールには 1 つ以上の仮想デバイスが含まれます。「仮想デバイス」は、物理ストレージのレイアウトとストレージプールの障害時の特性を定義し

た、ストレージプールの内部表現です。つまり、仮想デバイスは、ストレージプールの作成に使用されるディスクデバイスまたはファイルを表しています。プールでは、構成の最上位に任意の数の仮想デバイス (「最上位レベル *vdev*」と呼ばれる) を含めることができます。

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

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

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

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

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

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

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

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

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

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

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

```
            c1t3d0  ONLINE       0     0     0

errors: No known data errors
```

ZFS ストレージプールの作成エラーに対応する

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

使用中のデバイスを検出する

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

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

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

マウントされたファイルシステム	このディスクまたはそのスライスの 1 つに、現在マウントされているファイルシステムが含まれています。このエラーを訂正するには、`umount` コマンドを使用してください。
/etc/vfstab 内のファイルシステム	このディスクには、`/etc/vfstab` ファイルに指定されているファイルシステムが含まれていますが、そのファイルシステムが現在マウントされていません。このエラーを訂正するには、`/etc/vfstab` ファイルでその行をコメントにしてください。
専用のダンプデバイス	このディスクは、システム専用のダンプデバイスとして使用中です。このエラーを訂正するには、`dumpadm` コマンドを使用してください。
ZFS プールの一部	このディスクまたはファイルは、アクティブな ZFS ストレージプールに含まれています。このエラーを訂正するには、そのプールが不要であれば `zpool destroy` コマンドを使用して破棄してください。または、`zpool detach` コマンドを

	使用して、そのプールからディスクを切り離します。ディスクを切り離すことができるのは、ミラー化ストレージプールの場合のみです。

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

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

次の例は、-f オプションの使用方法を示しています。

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

できるだけ、-f オプションを使用してエラーをオーバーライドする以外の方法でエラーを訂正するようにしてください。

複製レベルが一致しない

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

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

```
invalid vdev specification
use '-f' to override the following errors:
mismatched replication level: 2-way mirror and 3-way mirror vdevs are present
```

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

ストレージプール作成のドライランを行う

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

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

        tank
          mirror
            c1t0d0
            c1t1d0
```

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

ストレージプールのデフォルトマウントポイント

プールが作成されるときに、最上位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。このディレクトリは、存在しないディレクトリか、空のディレクトリにする必要があります。ディレクトリが存在しない場合は、自動的に作成されます。ディレクトリが空の場合は、ルートファイルシステムが既存のディレクトリの最上位にマウントされます。別のデフォルトマウントポイントを使用してプールを作成する場合は、-zpool create コマンドの m オプションを使用します。次に例を示します。

```
# zpool create home c1t0d0
default mountpoint '/home' exists and is not empty
```

```
use '-m' option to provide a different default
# zpool create -m /export/zfs home c1t0d0
```

このコマンドを実行すると、/export/zfs をマウントポイントとして、新しいプール home および home 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、179 ページの「ZFS マウントポイントを管理する」を参照してください。

ZFS ストレージプールを破棄する

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

```
# zpool destroy tank
```

注意 – プールを破棄するときは、十分に注意してください。破棄するプールに間違いがないことを確認し、常にデータをコピーしておいてください。ほかのプールを間違って破棄してしまった場合は、そのプールの回復を試みることができます。詳細は、110 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。

zpool destroy コマンドを使用してプールを破棄した場合でも、110 ページの「破棄された ZFS ストレージプールを回復する」で説明するように、プールをインポートできます。これは、プールの一部だったディスク上の機密データがまだ利用可能であることを意味します。破棄されたプールのディスク上のデータを破棄する場合は、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する必要があります。

システムデータの機密性を維持するもう 1 つの方法は、暗号化された ZFS ファイルシステムを作成することです。暗号化されたファイルシステムを使用するプールが破棄された場合は、破棄されたプールが回復されても、暗号化鍵がなければデータにアクセスできません。詳細は、201 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。

使用できないデバイスが含まれるプールを破棄する

プールを破棄するには、そのプールが有効でなくなったことを示すデータをディスクに書き込む必要があります。この状態情報が書き込まれたデバイスは、インポートを実行するときに、アクティブである可能性のあるプールとして表示されな

くなります。1 つ以上のデバイスが使用できない状態のときでも、そのプールを破棄できます。ただし、これらの使用できないデバイスには、必要な状態情報は書き込まれません。

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

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

プールとデバイスの健全性の詳細については、97 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。

インポートツールの詳細については、106 ページの「ZFS ストレージプールをインポートする」を参照してください。

ZFS ストレージプール内のデバイスを管理する

デバイスに関する基本情報のほとんどは、45 ページの「ZFS ストレージプールのコンポーネント」に記載してあります。プールを作成したあとに、いくつかのタスクを実行してプール内の物理デバイスを管理できます。

- 66 ページの「ストレージプールにデバイスを追加する」
- 71 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」
- 72 ページの「ミラー化 ZFS ストレージプールを分割して新しいプールを作成する」
- 76 ページの「ストレージプール内のデバイスをオンラインまたはオフラインにする」
- 78 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」
- 78 ページの「ストレージプール内のデバイスを置き換える」
- 81 ページの「ストレージプールにホットスペアを指定する」

ストレージプールにデバイスを追加する

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

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

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

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

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

仮想デバイスがどのように検証されるかの詳細については、61 ページの「使用中のデバイスを検出する」を参照してください。

例 3–1　ZFS ミラー化構成にディスクを追加する

次の例では、既存のミラー化 ZFS 構成に別のミラーが追加されます。

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

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

例 3–1 ZFS ミラー化構成にディスクを追加する (続き)

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

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

errors: No known data errors
```

例 3–2 RAID-Z 構成にディスクを追加する

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

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

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

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

        NAME        STATE     READ WRITE CKSUM
        rzpool      ONLINE       0     0     0
          raidz1-0  ONLINE       0     0     0
```

例 3–2 RAID-Z 構成にディスクを追加する (続き)

```
            c1t0d0  ONLINE       0     0     0
            c1t2d0  ONLINE       0     0     0
            c1t3d0  ONLINE       0     0     0
          raidz1-1  ONLINE       0     0     0
            c2t2d0  ONLINE       0     0     0
            c2t3d0  ONLINE       0     0     0
            c2t4d0  ONLINE       0     0     0

errors: No known data errors
```

例 3–3 ミラー化ログデバイスを追加および削除する

次の例では、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 status newpool
  pool: newpool
 state: ONLINE
 scrub: none requested
config:

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

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

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

errors: No known data errors
```

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

例 3–3 ミラー化ログデバイスを追加および削除する (続き)

zpool remove コマンドを使用して、ログデバイスを削除できます。前の例のミラー化ログデバイスは、mirror-1 引数を指定することによって削除できます。例:

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

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

errors: No known data errors
```

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

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

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

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

例 3–4 キャッシュデバイスを追加および削除する

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

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

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

例 3–4 キャッシュデバイスを追加および削除する (続き)

```
config:

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

errors: No known data errors
```

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

`zpool remove` コマンドを使用して、キャッシュデバイスを削除します。例:

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

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

errors: No known data errors
```

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

ZFS ストレージプールでキャッシュデバイスを使用する方法の詳細については、58 ページの「キャッシュデバイスを使用して ZFS ストレージプールを作成する」を参照してください。

ストレージプール内でデバイスを接続する/切り離す

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

ディスクを接続してミラー化ルートプールを作成する場合は、121 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。

ZFS ルートプール内のディスクを交換する場合は、124 ページの「ZFS ルートプールのディスクを交換する方法 (SPARC または x86/VTOC)」を参照してください。

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

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

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

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

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

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

errors: No known data errors
```

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

例 3–6 非冗長な ZFS ストレージプールをミラー化された ZFS ストレージプールに変換する

また、zpool attach コマンドを使用して、非冗長なストレージプールを冗長なストレージプールに変換できます。例:

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

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

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

errors: No known data errors
```

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

```
# zpool detach zeepool c2t1d0
```

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

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

ミラー化 ZFS ストレージプールを分割して新しいプールを作成する

ミラー化 ZFS ストレージプールは、zpool split コマンドを使用することにより、バックアッププールとして簡単に複製できます。この機能を使用して、ミラー化ルートプールを分割できますが、分割されたプールは、いくつかの追加手順を実行するまでブート可能ではありません。

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

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

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

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

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

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

errors: No known data errors

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

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

errors: No known data errors
```

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

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

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

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

分割されるプールの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 /tank2 tank tank2
```

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

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

- RAID-Z 構成または複数のディスクから成る非冗長プールに対しては、この機能を使用できません。
- `zpool split` 操作を試みる前に、データおよびアプリケーションの操作を終了しておいてください。
- 再同期化が進行中の場合、プールを分割できません。
- ミラー化プールの分割は、プールが 2 台か 3 台のディスクを含むときに行うのが最適です。このとき、元のプール内の最後のディスクが新しく作成されるプールで使用されます。その後、`zpool attach` コマンドを使用して元のミラー化ストレージプールを再作成するか、または新しく作成したプールをミラー化ストレージプールに変換することができます。新しい (分割された) プールが非冗長なため、1 回の `zpool split` 操作で既存のミラー化プールから新しいミラー化プールを作成する方法は現時点で存在しません
- 既存のプールが 3 方向ミラーの場合、分割操作後に新しいプールに含まれるディスクは 1 台です。既存のプールが 2 台のディスクから成る 2 方向ミラーの場合の結果は、2 台のディスクから成る 2 つの非冗長プールになります。2 台の追加ディスクを接続して、非冗長プールをミラー化プールに変換する必要があります。
- 分割操作中にデータの冗長性を維持するためのよい方法は、3 台のディスクを含むミラー化ストレージプールを分割し、分割操作後に元のプールが 2 台のミラー化ディスクを含むようにすることです。
- ミラー化プールを分割する前にハードウェアが正しく構成されていることを確認してください。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、333 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。

例 3–7 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 mothership 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 mothership と、1 台のディスクから成る新しいプール luna です。各プールの内容は同じです。

プール luna はバックアップの目的で別のシステムにインポートできます。バックアップの完了後、プール luna を破棄して、ディスクを mothership に再接続できます。その後、このプロセスを繰り返すことができます。

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

        NAME                        STATE     READ WRITE CKSUM
        mothership                  ONLINE       0     0     0
          mirror-0                  ONLINE       0     0     0
            c0t5000C500335F95E3d0   ONLINE       0     0     0
            c0t5000C500335BD117d0   ONLINE       0     0     0
            c0t5000C500335F907Fd0   ONLINE       0     0     0

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

        NAME                      STATE     READ WRITE CKSUM
        luna                      ONLINE       0     0     0
          c0t5000C500335F907Fd0   ONLINE       0     0     0

errors: No known data errors

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

        NAME                        STATE     READ WRITE CKSUM
        mothership                  ONLINE       0     0     0
          mirror-0                  ONLINE       0     0     0
            c0t5000C500335F95E3d0   ONLINE       0     0     0
            c0t5000C500335BD117d0   ONLINE       0     0     0

errors: No known data errors
```

ストレージプール内のデバイスをオンラインまたはオフラインにする

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

注 – デバイスを置き換えるときに、オフラインにする必要はありません。

デバイスをオフラインにする

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

```
# zpool offline tank c0t5000C500335F95E3d0
```

デバイスをオフラインにするときは、次の点を考慮します。

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

  ```
  # zpool offline tank c0t5000C500335F95E3d0
  cannot offline c0t5000C500335F95E3d0: no valid replicas
  ```

- デフォルトでは、`OFFLINE` 状態は持続的です。システムをリブートしても、デバイスはオフラインのままです。

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

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

  システムをリブートすると、このデバイスは自動的に `ONLINE` 状態に戻ります。

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

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

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

元のストレージプールを破棄しないで、デバイスを別のストレージプールから使用する場合は、元のストレージプールにある既存のデバイスを別の類似したデバイスに置き換える方法もあります。デバイスを置き換える方法については、78 ページの「ストレージプール内のデバイスを置き換える」を参照してください。

オフラインのデバイスは、プールステータスのクエリー検索を行うと、OFFLINE ステータスとして表示されます。プールステータスのクエリー検索については、90 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。

デバイスの健全性の詳細については、97 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。

デバイスをオンラインにする

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

```
# zpool online tank c0t5000C500335F95E3d0
```

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

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

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

```
SUNW-MSG-ID: ZFS-8000-LR, TYPE: Fault, VER: 1, SEVERITY: Major
EVENT-TIME: Wed Jun 20 11:35:26 MDT 2012
PLATFORM: ORCL,SPARC-T3-4, CSN: 1120BDRCCD, HOSTNAME: tardis
SOURCE: zfs-diagnosis, REV: 1.0
EVENT-ID: fb6699c8-6bfb-eefa-88bb-81479182e3b7
DESC: ZFS device 'id1,sd@n5000c500335dc60f/a' in pool 'pond' failed to open.
AUTO-RESPONSE: An attempt will be made to activate a hot spare if available.
IMPACT: Fault tolerance of the pool may be compromised.
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-LR for the latest
service procedures and policies regarding this diagnosis.
```

エラー状態のデバイスを置き換える方法については、303 ページの「見つからないデバイスに関する問題を解決する」を参照してください。

zpool online -e コマンドを使用して LUN を拡張できます。デフォルトでは、プールに追加された LUN は、プールの autoexpand プロパティーが有効でない場合はその最大サイズにまで拡張されません。zpool online -e コマンドを使用すると、LUN がすでにオンラインの場合でも、現在オフラインの場合でも、LUN を自動的に拡張できます。例:

```
# zpool online -e tank c0t5000C500335F95E3d0
```

ストレージプールデバイスのエラーをクリアーする

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

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

```
# zpool clear tank
```

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

```
# zpool clear tank c0t5000C500335F95E3d0
```

zpool エラーのクリアーについては、309 ページの「一時的なエラーを解消する」を参照してください。

ストレージプール内のデバイスを置き換える

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

冗長プール内の同じ場所にある別のデバイスでデバイスを物理的に置き換える場合は、置き換えられるデバイスを特定するだけで済むことがあります。一部のハードウェアでは、デバイスは同じ場所にある別のディスクであるということが ZFS によって認識されます。たとえば、障害の発生したディスク (c1t1d0) を置き換える場合、そのディスクを取り除き、同じ場所でそれを置き換えるには、次の構文を使用します。

```
# zpool replace tank c1t1d0
```

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

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

ZFS ルートプール内のディスクを交換する場合は、124 ページの「ZFS ルートプールのディスクを交換する方法 (SPARC または x86/VTOC)」を参照してください。

ディスクを置き換えるための基本的な手順は次のとおりです。

1. 必要に応じて、`zpool offline` コマンドでディスクをオフラインにします。
2. 置き換えるディスクを取り外します。
3. 交換用ディスクを挿入します。
4. `format` の出力を確認して、交換用ディスクが認識できるかどうかを判断します。

   さらに、デバイス ID が変更されているかどうかも確認します。交換用ディスクに WWN が含まれている場合、障害のあるディスクのデバイス ID は変更されます。
5. ディスクが交換されたことを ZFS に知らせます。例:

   ```
   # zpool replace tank c1t1d0
   ```

   上記に示したように、交換用ディスクに異なるデバイス ID が含まれている場合は、新しいデバイス ID を追加します。

   ```
   # zpool replace tank c0t5000C500335FC3E7d0 c0t5000C500335BA8C3d0
   ```
6. 必要に応じて、`zpool online` コマンドでディスクをオンラインに戻します。
7. デバイスが交換されたことを FMA に知らせます。

   `fmadm faulty` の出力から、`Affects:` セクション内の `zfs://pool=name/vdev=guid` 文字列を確認し、その文字列を `fmadm repaired` コマンドの引数として指定します。

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

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で説明したように `zpool replace` コマンドを実行するだけで置き換えを実行できます。

SATA ディスクの交換例については、例 10–1 を参照してください。

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

- プールの autoreplace プロパティーをオンに設定した場合、そのプールに以前属していたデバイスと物理的に同じ位置に新しいデバイスが検出されると、そのデバイスが自動的にフォーマットされて置き換えられます。このプロパティーが有効なときは、zpool replace コマンドを使用する必要はありません。ハードウェアの種類によっては、この機能を使用できない場合があります。
- システムの実行中にデバイスまたはホットスペアが物理的に取り外されると、ストレージプールの状態は REMOVED になります。可能であれば、取り外されたデバイスはホットスペアデバイスで置き換えられます。
- デバイスをいったん取り外してから挿入し直すと、デバイスはオンラインになります。デバイスを挿入し直したときにホットスペアがアクティブになっていた場合は、オンライン処理が完了すると、そのホットスペアが取り外されます。
- デバイスの着脱時の自動検出はハードウェアに依存しているため、すべてのプラットフォームには対応していない可能性があります。たとえば、USB デバイスは挿入時に自動的に構成されます。ただし、cfgadm -c configure コマンドを使用して SATA ドライブを構成する必要がある場合もあります。
- ホットスペアは、オンラインおよび使用可能かどうか定期的に確認されます。
- 交換用デバイスの容量が、ミラー化構成または RAID-Z 構成内でもっとも容量の小さいディスク以上である必要があります。
- 置き換える前のデバイスよりもサイズが大きい交換デバイスをプールに追加しても、プールは自動的にその最大サイズにまで拡張されません。プールの autoexpand プロパティーの値は、ディスクをプールに追加したときに、置き換え後の LUN がその最大サイズにまで拡張されるかどうかを決定します。デフォルトでは、autoexpand プロパティーは無効になっています。容量の大きい LUN をプールに追加する前後どちらでも、このプロパティーを有効にすることで LUN のサイズを拡張できます。

  次の例では、ミラー化プール内の 16G バイトのディスク 2 台を 72G バイトのディスク 2 台で置き換えます。2 番目のデバイスの交換を試行する前に、最初のデバイスが完全に再同期化されていることを確認してください。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 create pool mirror c1t16d0 c1t17d0
# zpool status
  pool: pool
 state: ONLINE
 scrub: none requested
config:

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

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

- 大規模プール内で多数のディスクを置き換える場合は、新しいディスク上にデータを再同期化するために時間がかかります。また、ディスクの置き換えの合間に `zpool scrub` コマンドを実行して、置き換えたあとのデバイスが動作可能なこと、およびデータが正しく書き込まれることを確認することもできます。
- 障害の発生したディスクがホットスペアに自動的に置き換えられた場合は、障害の発生したディスクが置き換えられたあとでスペアの切り離しが必要になることがあります。`zpool detach` コマンドを使用して、ミラー化プールまたは RAID-Z プールのスペアを切り離すことができます。ホットスペアの切り離しについては、83 ページの「ストレージプール内のホットスペアをアクティブにする/非アクティブにする」を参照してください。

デバイスの置き換えの詳細については、303 ページの「見つからないデバイスに関する問題を解決する」および307 ページの「破損したデバイスを交換または修復する」を参照してください。

ストレージプールにホットスペアを指定する

ホットスペア機能を使って、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」として指定したデバイスはプール内ではアクティブデバイスではありませんが、プールのアクティブデバイスで障害が発生した場合には、そのデバイスがホットスペアに自動的に置き換えられます。

次の方法を使って、デバイスをホットスペアとして指定できます。

- プール作成時に `zpool create` コマンドを使用します。
- プール作成後に `zpool add` コマンドを使用します。

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

```
# zpool create zeepool mirror c0t5000C500335F95E3d0 c0t5000C500335F907Fd0
mirror c0t5000C500335BD117d0 c0t5000C500335DC60Fd0 spare c0t5000C500335E106Bd0 c0t5000C500335FC3E7d0
# zpool status zeepool
```

```
  pool: zeepool
 state: ONLINE
  scan: none requested
config:

        NAME                       STATE     READ WRITE CKSUM
        zeepool                    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
        spares
          c0t5000C500335E106Bd0    AVAIL
          c0t5000C500335FC3E7d0    AVAIL

errors: No known data errors
```

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

```
# zpool add zeepool spare c0t5000C500335E106Bd0 c0t5000C500335FC3E7d0
# zpool status zeepool
  pool: zeepool
 state: ONLINE
  scan: none requested
config:

        NAME                       STATE     READ WRITE CKSUM
        zeepool                    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
        spares
          c0t5000C500335E106Bd0    AVAIL
          c0t5000C500335FC3E7d0    AVAIL

errors: No known data errors
```

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

```
# zpool remove zeepool c0t5000C500335FC3E7d0
# zpool status zeepool
  pool: zeepool
 state: ONLINE
  scan: none requested
config:

        NAME                       STATE     READ WRITE CKSUM
        zeepool                    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
        spares
          c0t5000C500335E106Bd0    AVAIL

errors: No known data errors
```

ストレージプールが現在使用しているホットスペアは、削除できません。

ZFS ホットスペアを使用するときは、次の点を考慮してください。

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

  `cannot replace` *`disk3`* `with` *`disk4`*`: device is too small`
- スペアはシステム間で共有できません。
- 同一のシステム上にある 2 つのデータプール間でスペアを共有すると、その 2 つのプール間でのスペアの使用を調整する必要があることを考慮してください。たとえば、プール A には使用中のスペアがあり、プール A はエクスポートされているとします。プール A がエクスポートされている間、プール B が、それと知らずにスペアを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。
- ルートプールとデータプール間でスペアを共有しないでください。

ストレージプール内のホットスペアをアクティブにする/非アクティブにする

ホットスペアをアクティブにするには、次のようにします。

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

  現在使用しているホットスペアで障害が発生した場合、FMA エージェントはそのスペアを切り離し、置き換えた状態を取り消します。続いてエージェントは、別のホットスペアが使用可能であれば、そのスペアを使ってデバイスを置き換えよ

うとします。現時点では、デバイスがシステムから見えなくならないと ZFS 診断エンジンがエラー状態を生成しないので、この機能もその事実に制限されます。

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

ホットスペアが使用可能な場合、`UNAVAIL` のデバイスは自動的に置き換えられます。例:

```
# zpool status -x
  pool: zeepool
 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: resilvered 3.15G in 0h0m with 0 errors on Thu Jun 21 16:46:19 2012
config:

        NAME                         STATE     READ WRITE CKSUM
        zeepool                      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
            spare-1                  DEGRADED   449     0     0
              c0t5000C500335DC60Fd0  UNAVAIL      0     0     0
              c0t5000C500335E106Bd0  ONLINE       0     0     0
        spares
          c0t5000C500335E106Bd0      INUSE

errors: No known data errors
```

現時点では、次の方法でホットスペアを非アクティブにできます。

- ストレージプールからホットスペアを削除する。
- 障害の発生したディスクを物理的に置き換えたあとでホットスペアを切り離す。例 3–8 を参照してください。
- 別のホットスペア内で一時的または永続的に交換を行う。例 3–9 を参照してください。

例 3–8 障害が発生したディスクの置き換え後にホットスペアを切り離す

次の例では、障害が発生したディスク (`c0t5000C500335DC60Fd0`) を物理的に置き換え、`zpool replace` コマンドを使って ZFS に通知します。

例 3–8 障害が発生したディスクの置き換え後にホットスペアを切り離す (続き)

```
# zpool replace zeepool c0t5000C500335DC60Fd0
# zpool status zeepool
  pool: zeepool
 state: ONLINE
  scan: resilvered 3.15G in 0h0m with 0 errors on Thu Jun 21 16:53:43 2012
config:

        NAME                       STATE     READ WRITE CKSUM
        zeepool                    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
        spares
          c0t5000C500335E106Bd0    AVAIL
```

必要に応じて、zpool detach コマンドを使ってホットスペアをスペアプールに戻すことができます。例:

```
# zpool detach zeepool c0t5000C500335E106Bd0
```

例 3–9 障害が発生したディスクを切り離してホットスペアを使用する

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

```
# zpool status zeepool
  pool: zeepool
 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 in progress since Thu Jun 21 17:01:49 2012
    1.07G scanned out of 6.29G at 220M/s, 0h0m to go
    0 repaired, 17.05% done
config:

        NAME                       STATE     READ WRITE CKSUM
        zeepool                    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
```

例 3–9 障害が発生したディスクを切り離してホットスペアを使用する (続き)

```
            c0t5000C500335BD117d0  ONLINE       0     0     0
            c0t5000C500335DC60Fd0  UNAVAIL      0     0     0
        spares
          c0t5000C500335E106Bd0    AVAIL

errors: No known data errors
# zpool detach zeepool c0t5000C500335DC60Fd0
# zpool status zeepool
  pool: zeepool
 state: ONLINE
  scan: resilvered 3.15G in 0h0m with 0 errors on Thu Jun 21 17:02:35 2012
config:

        NAME                       STATE     READ WRITE CKSUM
        zeepool                    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
            c0t5000C500335E106Bd0  ONLINE       0     0     0

errors: No known data errors
(Original failed disk c0t5000C500335DC60Fd0 is physically replaced)
# zpool add zeepool spare c0t5000C500335DC60Fd0
# zpool status zeepool
  pool: zeepool
 state: ONLINE
  scan: resilvered 3.15G in 0h0m with 0 errors on Thu Jun 21 17:02:35 2012
config:

        NAME                       STATE     READ WRITE CKSUM
        zeepool                    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
            c0t5000C500335E106Bd0  ONLINE       0     0     0
        spares
          c0t5000C500335DC60Fd0    AVAIL

errors: No known data errors
```

ディスクが交換され、スペアが切り離されたあと、ディスクが修復されたことを FMA に知らせます。

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

ZFS ストレージプールのプロパティーの管理

zpool get コマンドを使用して、プールのプロパティーの情報を表示できます。例:

```
# zpool get all zeepool
NAME     PROPERTY       VALUE                SOURCE
zeepool  allocated      6.29G                -
zeepool  altroot        -                    default
zeepool  autoexpand     off                  default
zeepool  autoreplace    off                  default
zeepool  bootfs         -                    default
zeepool  cachefile      -                    default
zeepool  capacity       1%                   -
zeepool  dedupditto     0                    default
zeepool  dedupratio     1.00x                -
zeepool  delegation     on                   default
zeepool  failmode       wait                 default
zeepool  free           550G                 -
zeepool  guid           7543986419840620672  -
zeepool  health         ONLINE               -
zeepool  listshares     off                  default
zeepool  listsnapshots  off                  default
zeepool  readonly       off                  -
zeepool  size           556G                 -
zeepool  version        34                   default
```

ストレージプールのプロパティーは zpool set コマンドで設定できます。例:

```
# zpool set autoreplace=on zeepool
# zpool get autoreplace zeepool
NAME     PROPERTY     VALUE    SOURCE
zeepool  autoreplace  on       local
```

使用率が 100% の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 tank
cannot set property for 'tank': out of space
```

プール容量の問題の回避方法については、第 12 章「推奨の Oracle Solaris ZFS プラクティス」を参照してください。

表 3–1 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	なし	読み取り専用の値。物理的に割り当て済みであるプール内のストレージ領域の容量を示します。

表 3–1 ZFS プールのプロパティーの説明 (続き)

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示します。設定されている場合、プール内のすべてのマウントポイントの先頭にこのディレクトリが付加されます。このプロパティーは、不明なプールを調べるときやマウントポイントが信頼できない場合、または通常のパスが有効でない代替ブート環境で使用できます。
autoreplace	ブール型	off	自動デバイス交換を制御します。オフに設定されている場合、zpool replace コマンドを使ってデバイス交換を開始する必要があります。オンに設定されている場合、そのプールに以前属していたデバイスと物理的に同じ位置にある新しいデバイスは、いずれも自動的にフォーマットされ、置き換えられます。このプロパティーの省略名は replace です。
bootfs	ブール型	なし	ルートプールのデフォルトのブート可能ファイルシステムを示します。このプロパティーは通常、インストールプログラムによって設定されます。
cachefile	文字列	なし	プール構成情報がキャッシュされる場所を制御します。システムのブート時に、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 構成では、このプロパティーは使用されません。
capacity	数値	なし	読み取り専用の値。使用されているプール領域の割合を示します。 このプロパティーの省略名は cap です。
dedupditto	文字列	なし	しきい値を設定し、重複除去したブロックの参照数がしきい値を超えた場合に、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	なし	プールに対して達成された読み取り専用の複製解除比。倍率で表されます。
delegation	ブール型	on	ファイルシステムに定義されているアクセス権を非特権ユーザーに付与できるかどうかを制御します。詳細は、第 8 章「Oracle Solaris ZFS 委任管理」を参照してください。

表 3–1 ZFS プールのプロパティーの説明 (続き)

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

表 3–1 ZFS プールのプロパティーの説明 (続き)

プロパティー名	タイプ	デフォルト値	説明
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、読み取り専用モードでプールがインポートされた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	なし	読み取り専用プロパティー。ストレージプールの合計サイズを示します。
version	数値	なし	プールの現在のディスク上バージョンを示します。プールを更新する方法としては zpool upgrade コマンドをお勧めしますが、下位互換性のために特定のバージョンが必要な場合には、このプロパティーを使用できます。このプロパティーには、1 から zpool upgrade -v コマンドで報告される現在のバージョンまでの任意の数値を設定できます。

ZFS ストレージプールの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 種類のストレージプール情報のすべてについて説明します。

- 90 ページの「ZFS ストレージプールについての情報を表示する」
- 95 ページの「ZFS ストレージプールの入出力統計を表示する 」
- 97 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」

ZFS ストレージプールについての情報を表示する

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

すべてのストレージプールまたは特定のプールについての情報を表示する

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

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

```
tank                    80.0G   22.3G   47.7G    28%  ONLINE     -
dozer                    1.2T    384G    816G    32%  ONLINE     -
```

このコマンド出力には、次の情報が表示されます。

NAME　プールの名前。

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

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

使用可能なファイルシステムの容量を確認する方法については、33 ページの「ZFS のディスク領域の計上」を参照してください。

FREE　プール内で割り当てられていない容量。

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

HEALTH　プールの現在の健全性ステータス。

プールの健全性の詳細については、97 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　プールの代替ルート (存在する場合)。

代替ルートプールの詳細については、289 ページの「ZFS 代替ルートプールを使用する」を参照してください。

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

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

zpool list 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 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

zpool status -l オプションを使用して、プールデバイスの物理的な場所に関する情報を表示します。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 Lab10Rack5...

# zpool status -l tank
  pool: tank
 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
        tank                                          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
          mirror-3                                    ONLINE       0     0     0
            /dev/chassis/Lab10Rack5.../DISK_01/disk   ONLINE       0     0     0
            /dev/chassis/Lab10Rack5.../DISK_21/disk   ONLINE       0     0     0
          mirror-4                                    ONLINE       0     0     0
            /dev/chassis/Lab10Rack5.../DISK_23/disk   ONLINE       0     0     0
            /dev/chassis/Lab10Rack5.../DISK_15/disk   ONLINE       0     0     0
          mirror-5                                    ONLINE       0     0     0
            /dev/chassis/Lab10Rack5.../DISK_09/disk   ONLINE       0     0     0
            /dev/chassis/Lab10Rack5.../DISK_04/disk   ONLINE       0     0     0
          mirror-6                                    ONLINE       0     0     0
            /dev/chassis/Lab10Rack5.../DISK_08/disk   ONLINE       0     0     0
            /dev/chassis/Lab10Rack5.../DISK_05/disk   ONLINE       0     0     0
          mirror-7                                    ONLINE       0     0     0
            /dev/chassis/Lab10Rack5.../DISK_07/disk   ONLINE       0     0     0
            /dev/chassis/Lab10Rack5.../DISK_11/disk   ONLINE       0     0     0
          mirror-8                                    ONLINE       0     0     0
            /dev/chassis/Lab10Rack5.../DISK_06/disk   ONLINE       0     0     0
            /dev/chassis/Lab10Rack5.../DISK_19/disk   ONLINE       0     0     0
          mirror-9                                    ONLINE       0     0     0
            /dev/chassis/Lab10Rack5.../DISK_00/disk   ONLINE       0     0     0
            /dev/chassis/Lab10Rack5.../DISK_13/disk   ONLINE       0     0     0
          mirror-10                                   ONLINE       0     0     0
            /dev/chassis/Lab10Rack5.../DISK_03/disk   ONLINE       0     0     0
            /dev/chassis/Lab10Rack5.../DISK_18/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
```

特定のストレージプールの統計を表示する

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

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

列の名前は、90 ページの「すべてのストレージプールまたは特定のプールについての情報を表示する」に示されているプロパティーに対応しています。

ZFS ストレージプールの出力をスクリプトで使えるようにする

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

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

別の例です。

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

ZFS ストレージプールのコマンド履歴を表示する

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

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

```
# zpool history
History for 'rpool':
```

```
2012-04-06.14:02:55 zpool create -f rpool c3t0d0s0
2012-04-06.14:02:56 zfs create -p -o mountpoint=/export rpool/export
2012-04-06.14:02:58 zfs set mountpoint=/export rpool/export
2012-04-06.14:02:58 zfs create -p rpool/export/home
2012-04-06.14:03:03 zfs create -p -V 2048m rpool/swap
2012-04-06.14:03:08 zfs set primarycache=metadata rpool/swap
2012-04-06.14:03:09 zfs create -p -V 4094m rpool/dump
2012-04-06.14:26:47 zpool set bootfs=rpool/ROOT/s11u1 rpool
2012-04-06.14:31:15 zfs set primarycache=metadata rpool/swap
2012-04-06.14:31:46 zfs create -o canmount=noauto -o mountpoint=/var/share rpool/VARSHARE
2012-04-06.15:22:33 zfs set primarycache=metadata rpool/swap
2012-04-06.16:42:48 zfs set primarycache=metadata rpool/swap
2012-04-09.16:17:24 zfs snapshot -r rpool/ROOT@yesterday
2012-04-09.16:17:54 zfs snapshot -r rpool/ROOT@now
```

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

履歴ログの特徴を次に示します。

- ログを無効にすることはできません。
- ログは永続的にディスクに保存されます。つまり、ログはシステムの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。
- ログの管理は不要です。つまり、ログのサイズを調整したり、ログの場所を変更したりする必要はありません。

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

```
# zpool history tank
2012-01-25.16:35:32 zpool create -f tank mirror c3t1d0 c3t2d0 spare c3t3d0
2012-02-17.13:04:10 zfs create tank/test
2012-02-17.13:05:01 zfs snapshot -r tank/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 tank
History for 'tank':
2012-01-25.16:35:32 zpool create -f tank mirror c3t1d0 c3t2d0 spare c3t3d0
[user root on tardis:global]
2012-02-17.13:04:10 zfs create tank/test [user root on tardis:global]
2012-02-17.13:05:01 zfs snapshot -r tank/test@snap1 [user root on tardis: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 tank
History for 'tank':
2012-01-25.16:35:32 zpool create -f tank 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 tardis 5.11 11.1 sun4v
2012-02-17.13:04:10 zfs create tank/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 tank/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。次の統計が報告されます。

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

プール領域とデータセット領域の相違点の詳細については、33 ページの「ZFS のディスク領域の計上」を参照してください。

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

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

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

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

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

プール全体の入出力統計を一覧表示する

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

```
# zpool iostat
               capacity     operations    bandwidth
pool        alloc   free   read  write   read  write
----------  -----  -----  -----  -----  -----  -----
```

```
rpool       6.05G  61.9G      0      0    786    107
tank        31.3G  36.7G      4      1   296K  86.1K
----------  -----  -----  -----  -----  -----  -----
```

これらの統計は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 tank 2
               capacity     operations    bandwidth
pool        alloc   free   read  write   read  write
----------  -----  -----  -----  -----  -----  -----
tank        18.5G  49.5G      0    187      0  23.3M
tank        18.5G  49.5G      0    464      0  57.7M
tank        18.5G  49.5G      0    457      0  56.6M
tank        18.8G  49.2G      0    435      0  51.3M
```

上記の例では、このコマンドによって tank プールの使用状況の統計が 2 秒ごとに表示され、Ctrl - C キーを押すと停止します。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。プールが 1 つだけの場合は、ひと続きの行に統計が表示されます。複数のプールがある場合は、各プールが分かれて見えるように、各プールの間に点線が挿入されます。

仮想デバイスの入出力統計を一覧表示する

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

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

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

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

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

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。例:

```
# 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
  mirror     490G   438G      2      5  8.60K  60.2K
    /dev/chassis/lab10rack15/SCSI_Device__6/disk      -      -      1      0  4.50K  60.2K
    /dev/chassis/lab10rack15/SCSI_Device__7/disk      -      -      1      0  4.49K  60.2K
  mirror     490G   438G      2      5  8.47K  60.1K
    /dev/chassis/lab10rack15/SCSI_Device__8/disk      -      -      1      0  4.42K  60.1K
    /dev/chassis/lab10rack15/SCSI_Device__9/disk      -      -      1      0  4.43K  60.1K
.
.
.
```

ZFS ストレージプールの健全性ステータスを調べる

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

このセクションでは、プールとデバイスの健全性を確認する方法について説明します。この章では、健全でないプールを修復または回復する方法については説明しません。トラブルシューティングおよびデータの回復については、第 10 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、冗長性構成のためにデータを引き続き使用できるプール。

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。プールの耐障害性が損なわれる可能性があります。別のデバイスで続けて障害が発生した場合には、回復できない状態になることがあります。
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	デバイスまたは仮想デバイスは正常に動作しています。一時的なエラーがいくつか発生している可能性はありますが、それらを除けば正常に動作しています。
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の場合と同じレベルのデータ冗長性やデータスループットを実現できない可能性があります。

zpool status コマンドも、再同期およびスクラブ操作に関する詳細を提供します。

- 再同期化進捗レポート。例:

```
scan: resilver in progress since Wed Jun 20 14:19:38 2012
    7.43G scanned out of 71.8G at 36.4M/s, 0h30m to go
    7.43G resilvered, 10.35% done
```

- 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
```

- 再同期化完了メッセージ。例:

```
scan: resilvered 71.8G in 0h14m with 0 errors on Wed Jun 20 14:33:42 2012
```

- 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
```

- スクラブおよび再同期化の完了メッセージはシステムのリブート後も残ります。

ストレージプールの基本的な健全性ステータス

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

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

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

詳細な健全性ステータス

-v オプションを使用すれば、より詳細な健全性のサマリーステータスを要求することができます。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
```

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

前の例では、UNAVAIL のデバイスを交換するようにしてください。デバイスを交換したあとに、必要に応じて zpool online コマンドを使用してデバイスをオンラインにします。例:

```
# zpool online pond c0t5000C500335F907Fd0
warning: device 'c0t5000C500335DC60Fd0' onlined, but remains in degraded state
# zpool status -x
all pools are healthy
```

上記の出力は、再同期化が完了するまで、デバイスが低下した状態のままであることを示しています。

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

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

```
# zpool status -x
 pool: pond
 state: DEGRADED
status: One or more devices has been taken offline by the administrator.
        Sufficient replicas exist for the pool to continue functioning in a
        degraded state.
action: Online the device using 'zpool online' or replace the device with
        'zpool replace'.
config:

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

errors: No known data errors
```

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

errors: フィールドは既知のデータエラーを示します。

前の出力例では、オフラインのデバイスでデータエラーは発生していません。

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 10 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。この作業を行うには、ストレージデバイスを元のシステムから切断して、移動先のシステムに再接続する必要があります。このタスクは、ケーブルをデバイスに物理的に接続し直すか、または複数のポートを持つデバイス (SAN 上のデバイスなど) を使用する方法で、行うことができます。ZFS では、アーキテクチャーエンディアンの異なるシステム間でも、一方のシステムのプールをエクスポートして移行先のシステムにインポートできます。異なるストレージプール間 (異なるシステム上にある場合を含む) でファイルシステムを複製または移行する方法については、223 ページの「ZFS データを送信および受信する」を参照してください。

ZFS ストレージプールの移行を準備する

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

プールを明示的にエクスポートする代わりに、ディスクを手動で取り外した場合でも、そのプールを別のシステムにインポートすることはできます。ただし、最後の数秒間のデータトランザクションが失われる可能性があります。この場合、デバイスが存在しないために、プールが元のシステム上で `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、`zpool export` コマンドを使用します。例:

```
# zpool export tank
```

このコマンドは、プールの中にマウントされたファイルシステムがある場合は、すべてをアンマウントしてから、次の処理を実行しようとします。いずれかのファイルシステムの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。例:

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

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

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

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

ZFS ボリュームの詳細については、279 ページの「ZFS ボリューム」を参照してください。

インポートできるストレージプールを判断する

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

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

        tank        ONLINE
          mirror-0  ONLINE
            c1t0d0  ONLINE
            c1t1d0  ONLINE
```

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

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

```
# zpool import
 pool: tank
    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:

    tank                          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.
```

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

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

```
# 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別のディレクトリからインポートする

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

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

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

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

ZFS ストレージプールをインポートする

インポートできるプールを確認したあとで、zpool import コマンドの引数にプールの名前または数値識別子を指定してインポートできます。例:

```
# zpool import tank
```

インポートできるプールが複数存在し、それらが同じ名前を持っている場合でも、数値識別子を使ってインポートするプールを指定する必要があります。例:

```
# zpool import
  pool: dozer
    id: 2704475622193776801
 state: ONLINE
action: The pool can be imported using its name or numeric identifier.
config:

        dozer       ONLINE
          c1t9d0    ONLINE

  pool: dozer
    id: 6223921996155991199
 state: ONLINE
action: The pool can be imported using its name or numeric identifier.
config:

        dozer       ONLINE
          c1t8d0    ONLINE
# zpool import dozer
cannot import 'dozer': more than one matching pool
import by numeric ID instead
# zpool import 6223921996155991199
```

プール名が既存のプール名と重複する場合は、別の名前でインポートできます。例:

```
# zpool import dozer zeepool
```

このコマンドは、エクスポート済みのプール `dozer` を新しい名前 `zeepool` を使ってインポートします。新しいプール名は永続的な名前です。

プールを明示的にエクスポートしていない場合は、別のシステムでまだ使用されているプールを誤ってインポートすることを防ぐためにインポートできません。`-f` フラグを使用する必要があります。例:

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

注 – あるシステムでアクティブになっているプールを別のシステムにインポートしようとしないでください。ZFS はネイティブのクラスタファイルシステム、分散ファイルシステム、または並列ファイルシステムではないため、異なる複数のホストからの同時アクセスには対応できません。

-R オプションを使用して、プールを代替ルートでインポートすることもできます。代替ルートプールの詳細については、289 ページの「ZFS 代替ルートプールを使用する」を参照してください。

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。例:

```
# zpool import dozer
  pool: dozer
    id: 16216589278751424645
 state: UNAVAIL
status: One or more devices are missing from the system.
action: The pool cannot be imported. Attach the missing
        devices and try again.
   see: http://support.oracle.com/msg/ZFS-8000-6X
config:

        dozer              UNAVAIL  missing device
          mirror-0         ONLINE
            c8t0d0  ONLINE
            c8t1d0  ONLINE

device details:

        missing-1        UNAVAIL          corrupted data
        status: ZFS detected errors on this device.
                The device has bad label or disk contents.


        Additional devices are known to be part of this pool, though their
        exact configuration cannot be determined.
```

ログデバイスがないプールをインポートします。例:

```
# zpool import -m dozer
# zpool status dozer
  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
          mirror-0              ONLINE       0     0     0
```

```
            c8t0d0              ONLINE       0     0     0
            c8t1d0              ONLINE       0     0     0
        logs
          2189413556875979854  UNAVAIL      0     0     0

errors: No known data errors
```

欠落したログデバイスを接続した後、`zpool clear` コマンドを実行してプールエラーをクリアーします。

ミラー化されたログデバイスがない場合も類似の回復を試行することができます。例:

欠落したログデバイスを接続した後、`zpool clear` コマンドを実行してプールエラーをクリアーします。

読み取り専用モードでプールをインポートする

読み取り専用モードでプールをインポートできます。プールが破損してプールにアクセスできない場合、この機能によって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 tank
# zpool scrub tank
cannot scrub tank: 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。このことは、インテントログの保留中の同期書き込みも、プールが読み書きモードでインポートされるまで再生されないことを意味します。
- 読み取り専用のインポート中におけるプールプロパティーの設定の試行は無視されます。

読み取り専用プールは、プールをエクスポートおよびインポートすることによって読み書きモードの設定に戻されることがあります。例:

```
# zpool export tank
# zpool import tank
# zpool scrub tank
```

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

次のコマンドでは、プールの特定のデバイスの 1 つ (この例では `/dev/dsk/c2t3d0`) を識別することによって、プール `dpool` をインポートします。

```
# zpool import -d /dev/dsk/c2t3d0s0 dpool
# zpool status dpool
  pool: dpool
 state: ONLINE
 scan: resilvered 952K in 0h0m with 0 errors on Fri Jun 29 16:22:06 2012
config:

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

このプールがディスク全体から構成されている場合でも、特定のデバイスのスライス識別子をコマンドに含める必要があります。

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。例:

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

        tank        ONLINE
          mirror-0  ONLINE
            c1t0d0  ONLINE
            c1t1d0  ONLINE
```

この zpool import の出力では、次の状態情報により、tank プールが破棄されたプールであることがわかります。

```
state: ONLINE (DESTROYED)
```

破棄されたプールを回復するには、回復するプールに対して zpool import -D コマンドを再度実行します。例:

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

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

```
            c1t0d0  ONLINE
            c1t1d0  ONLINE

errors: No known data errors
```

破棄されたプール内のいずれかのデバイスが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 destroy dozer
# 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 c8t4d0
# 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

以前の Solaris リリースの ZFS ストレージプールがある場合は、`zpool upgrade` コマンドを使ってそのプールをアップグレードすれば、最新リリースのプール機能を利用できます。また、古いバージョンのプールを実行している場合は、`zpool status` コマンドによって知ることができます。例:

```
# zpool status
  pool: tank
 state: ONLINE
status: The pool is formatted using an older on-disk format.  The pool can
        still be used, but some features are unavailable.
action: Upgrade the pool using 'zpool upgrade'.  Once this is done, the
        pool will no longer be accessible on older software versions.
 scrub: none requested
config:
        NAME        STATE     READ WRITE CKSUM
        tank        ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c1t0d0  ONLINE       0     0     0
            c1t1d0  ONLINE       0     0     0
errors: No known data errors
```

次の構文を使って、特定のバージョンやサポートされるリリースに関する追加情報を確認できます。

```
# zpool upgrade -v
This system is currently running ZFS pool version 33.

The following versions are supported:

VER  DESCRIPTION
---  --------------------------------------------------------
 1   Initial ZFS version
 2   Ditto blocks (replicated metadata)
 3   Hot spares and double parity RAID-Z
 4   zpool history
 5   Compression using the gzip algorithm
 6   bootfs pool property
 7   Separate intent log devices
 8   Delegated administration
 9   refquota and refreservation properties
 10  Cache devices
 11  Improved scrub performance
 12  Snapshot properties
 13  snapused property
 14  passthrough-x aclinherit
```

```
 15  user/group space accounting
 16  stmf property support
 17  Triple-parity RAID-Z
 18  Snapshot user holds
 19  Log device removal
 20  Compression using zle (zero-length encoding)
 21  Deduplication
 22  Received properties
 23  Slim ZIL
 24  System attributes
 25  Improved scrub stats
 26  Improved snapshot deletion performance
 27  Improved snapshot creation performance
 28  Multiple vdev replacements
 29  RAID-Z/mirror hybrid allocator
 30  Encryption
 31  Improved 'zfs list' performance
 32  One MB blocksize
 33  Improved share support
 34  Sharing with inheritance

For more information on a particular version, including supported releases,
see the ZFS Administration Guide.
```

これで、`zpool upgrade` コマンドを実行してすべてのプールをアップグレードできます。例:

```
# zpool upgrade -a
```

注 – プールを新しい ZFS バージョンにアップグレードすると、そのプールは古い ZFS バージョンを実行しているシステムではアクセスできなくなります。

◆ ◆ ◆ 第 4 章

ZFS ルートプールのコンポーネントの管理

この章では、ルートプールミラーの接続、ZFS ブート環境の複製、スワップデバイスおよびダンプデバイスのサイズ変更など、Oracle Solaris ZFS ルートプールコンポーネントを管理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。

ルートプールの回復の詳細については、第 11 章「スナップショットのアーカイブとルートプールの回復」を参照してください。

最新の問題については、Oracle Solaris 11.1 のリリースノートを参照してください。

ZFS ルートプールのコンポーネントの管理 (概要)

ZFS は、Oracle Solaris 11 リリースのデフォルトのルートファイルシステムです。Oracle Solaris リリースをインストールするときは、次の考慮事項を確認してください。

- インストール – Oracle Solaris 11 リリースでは、次の方法で ZFS ルートファイルシステムからインストールおよびブートを実行できます。
  - Live CD (x86 のみ) – ZFS ルートプールを単一のディスクにインストールします。インストール中に `fdisk` パーティションメニューを使用して、使用している環境でディスクをパーティションに分割できます。

  - テキストインストール (SPARC および x86) – メディアから、またはネットワークを介して、ZFS ルートプールを単一のディスクにインストールします。インストール中に `fdisk` パーティションメニューを使用して、使用している環境でディスクをパーティションに分割できます。
  - 自動インストーラ (AI) (SPARC および x86) – ZFS ルートプールを自動的にインストールします。AI マニフェストを使用して、ZFS ルートプールに使用するディスクとディスクパーティションを決定できます。
- スワップデバイスおよびダンプデバイス – 上記のすべてのインストール方法によって、ZFS ルートプール内の ZFS ボリュームに自動的に作成されます。ZFS スワップデバイスおよびダンプデバイスの管理方法については、131 ページの「ZFS スワップデバイスおよびダンプデバイスを管理する」を参照してください。
- ミラー化ルートプール構成 – ミラー化ルートプールは自動インストール中に構成できます。インストール後のミラー化ルートプールの構成の詳細は、121 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。
- ルートプールの容量管理 – システムがインストールされたあと、ZFS ルートファイルシステムがいっぱいにならないように、ルートファイルシステムに割り当て制限を設定することを検討してください。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。割り当て制限の設定については、196 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。

ZFS ルートプールの一般的な要件

ZFS ルートプールの容量および構成の要件について説明している以降のセクションを確認してください。

ZFS ルートプールの容量要件

システムがインストールされたとき、スワップボリュームとダンプボリュームのサイズは、物理メモリーの量に依存します。ブート可能な ZFS ルートファイルシステムに最小限必要なプール容量は、物理メモリーの容量、利用可能なディスク容量、および作成するブート環境 (BE) の数によって決まります。

次の ZFS ストレージプール容量要件を確認してください。

- さまざまなインストール方法に応じたメモリー要件については、『Oracle Solaris 11.1 ご使用にあたって』を参照してください。
- 少なくとも 7G - 13G バイトのディスク容量が推奨されます。容量は次のように消費されます。

- スワップ領域とダンプデバイス – Solaris インストールプログラムで作成されるスワップボリュームとダンプボリュームのデフォルトのサイズは、システム上のメモリーの容量とその他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。

  インストール中またはインストール後に、新しいサイズがシステムの動作をサポートしているかぎり、スワップボリュームとダンプボリュームのサイズを新しいサイズに調整できます。詳細は、132 ページの「ZFS スワップデバイスおよびダンプデバイスのサイズを調整する」を参照してください。
- ブート環境 **(BE)** – ZFS BE は、おおむね 4-6G 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 を更新する場合、更新に応じて BE サイズが増加することを考慮してください。 同じルートプール内のすべての 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 OS** コンポーネント – ルートファイルシステムの、OS イメージの一部となっているサブディレクトリのうち、`/var` 以外のものはすべて、ルートファイルシステム内に存在している必要があります。さらに、スワップデバイスとダンプデバイス以外の Solaris OS コンポーネントはすべて、ルートプール内に存在している必要があります。

ZFS ルートプールの構成要件

次の ZFS ストレージプール構成要件を確認してください。

- Oracle Solaris 11.1 では、ルートプールに使用するディスクに、x86 ベースシステムでは EFI (GPT) または SMI (VTOC) ラベルのどちらかを、SPARC システムでは SMI (VTOC) ラベルを付けることができます。
  - 更新済みの GPT 対応ファームウェアを搭載した SPARC システムは、EFI (GPT) ディスクラベルを 1 つまたは複数のルートプールディスクにインストールします。SPARC システムに更新済みのファームウェアがない場合、SMI (VTOC) ディスクラベルは 1 つまたは複数のルートプールディスクにインストールされます。
  - x86 ベースシステムでは、ほとんどの場合、EFI (GPT) ラベルがルートプールディスクまたはディスクにインストールされます。

  x86 ベースシステムで EFI (GPT) ラベルがどのように表示されるかについては、45 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。
- SMI (VTOC) ラベルのディスクが SPARC ベースシステムまたは x86 ベースシステム上にある場合、そのプールはディスクスライスまたはミラー化されたディスクスライスのどちらかに存在している必要があります。あるいは、ルートプールディスクに EFI (GPT) ラベルが付いている場合、そのプールはディスク全体また

はミラー化されたディスク全体のいずれにも存在できます。beadm 操作中に、サポートされていないプール構成を使用しようとすると、次のようなメッセージが表示されます。

```
ERROR: ZFS pool name does not support boot environments
```

サポートされている ZFS ルートプール構成の詳細については、54 ページの「ZFS ルートプールを作成する」を参照してください。

- x86 ベースシステムでは、ディスクに Solaris fdisk パーティションが含まれている必要があります。Solaris fdisk パーティションは、x86 システムのインストール時に自動的に作成されます。Solaris fdisk パーティションの詳細は、『Oracle Solaris 11.1 の管理: デバイスとファイルシステム』の「fdisk パーティションの作成上のガイドライン」を参照してください。
- プールプロパティーまたはファイルシステムプロパティーは、自動インストール中にルートプールで設定できます。ルートプールでは gzip 圧縮アルゴリズムはサポートされていません。
- 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。

ZFS ルートプールを管理する

次のセクションでは、ZFS ルートプールのインストールと更新について、およびミラー化ルートプールの構成について説明します。

ZFS ルートプールをインストールする

Oracle Solaris 11 Live CD によるインストール方法では、デフォルトの ZFS ルートプールが単一ディスク上にインストールされます。Oracle Solaris 11 自動インストール (AI) 方法では、AI マニフェストを作成することで、ZFS ルートプール用のディスクまたはミラー化ディスクを指定できます。

AI インストーラは、ZFS ルートプールをデフォルトのブートディスクにインストールするか、または指定したターゲットディスクにインストールする柔軟性を提供します。c1t0d0 などの論理デバイスや、物理デバイスのパスを指定できます。さらに、MPxIO 識別子や、インストールされるデバイスのデバイス ID も使用できます。

インストールのあとで、ZFS ストレージプールとファイルシステムの情報を確認します。これらの情報は、インストールのタイプとカスタマイズによって異なります。例:

```
# 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 BE 情報を確認します。例:

```
# beadm list
BE      Active Mountpoint Space Policy Created
--      ------ ---------- ----- ------ -------
solaris NR     /          3.75G static 2012-07-20 12:10
```

上記の出力では、Active フィールドは、BE が現在アクティブであるか (N で表現)、リブート時にアクティブになるか (R で表現)、またはその両方であるか (NR で表現) を示します。

▼ ZFS ブート環境を更新する方法

デフォルトの ZFS ブート環境 (BE) には、デフォルトで solaris という名前が付けられます。BE は beadm list コマンドを使用して識別できます。例:

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

上記の出力で、NR は、BE が現在アクティブであり、リブート時にアクティブな BE になることを意味しています。

pkg update コマンドを使用して ZFS ブート環境を更新できます。pkg update コマンドを使用して ZFS BE を更新した場合、既存の BE への更新がきわめてわずかである場合を除き、新しい BE が作成されて自動的にアクティブになります。

1 **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。

更新プロセスの外部でバックアップ BE を作成してアクティブにすることもできます。

```
# beadm create solaris-1
# beadm activate solaris-1
```

2 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 2012-07-20 10:25
solaris-1 NR     /          3.82G  static 2012-07-19 14:45
```

3 新しい **BE** のブート時にエラーが発生した場合、以前の **BE** をアクティブにして戻ります。

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

▼ 代替 BE をマウントする方法

回復するために、別の BE からのファイルをコピーしたりそのファイルにアクセスしたりする必要が生じる場合があります。

1 管理者になります。

2 代替 **BE** をマウントします。

```
# beadm mount solaris-1 /mnt
```

3 **BE** にアクセスします。

```
# ls /mnt
bin         export      media       pkg         rpool       tmp
boot        home        mine        platform    sbin        usr
dev         import      mnt         proc        scde        var
devices     java        net         project     shared
doe         kernel      nfs4        re          src
etc         lib         opt         root        system
```

4 使用し終えたら、代替 **BE** をアンマウントします。

```
# beadm umount solaris-1
```

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

自動インストール中にミラー化ルートプールを構成しない場合は、インストール後にミラー化ルートプールを簡単に構成できます。

ルートプール内のディスクを交換する方法については、124 ページの「ZFS ルートプールのディスクを交換する方法 (SPARC または x86/VTOC)」を参照してください。

1 ルートプールの現在のステータスを表示します。

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

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

errors: No known data errors
```

2 必要に応じて、ルートプールに接続する 2 つ目のディスクを準備します。

- SPARC: ディスクに SMI (VTOC) ディスクラベルとスライス 0 があることを確認してください。ディスクのラベルを変更してスライス 0 を作成する必要がある場合は、『Oracle Solaris 11.1 の管理: デバイスとファイルシステム』の「ZFS ルートファイルシステム用のディスクスライスを作成する方法」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てスライス 0 を作成する必要がある場合は、『Oracle Solaris 11.1 の管理: デバイスとファイルシステム』の「ZFS ルートファイルシステム用のディスクの準備」を参照してください。

3 ミラー化ルートプール構成にするために、2 つ目のディスクを接続します。

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

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。

4 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。

```
# zpool status rpool
# zpool status rpool
  pool: rpool
```

```
 state: DEGRADED
status: One or more devices is currently being resilvered.  The pool will
        continue to function in a degraded state.
action: Wait for the resilver to complete.
        Run 'zpool status -v' to see device specific details.
  scan: resilver in progress since Fri Jul 20 13:39:53 2012
    938M scanned out of 11.7G at 46.9M/s, 0h3m to go
    938M resilvered, 7.86% done
config:

        NAME          STATE     READ WRITE CKSUM
        rpool         DEGRADED     0     0     0
          mirror-0    DEGRADED     0     0     0
            c2t0d0s0  ONLINE       0     0     0
            c2t1d0s0  DEGRADED     0     0     0  (resilvering)
```

上の出力の場合、再同期化処理は完了していません。次のようなメッセージが表示されたら、再同期化が完了しています。

```
resilvered 11.6G in 0h5m with 0 errors on Fri Jul 20 13:57:25 2012
```

5 より大きいディスクを接続する場合は、プールの **autoexpand** プロパティーを設定して、プールのサイズを拡張します。

既存の rpool プールサイズを判別します。

```
# 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
```

拡張した rpool プールサイズを確認します。

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

6 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。

▼ ミラー化ルートプールを構成する方法 (x86/EFI (GPT))

Oracle Solaris 11.1 リリースでは、ほとんどの場合、デフォルトで x86 ベースシステムに EFI (GPT) ラベルがインストールされます。

自動インストール中にミラー化ルートプールを構成しない場合は、インストール後にミラー化ルートプールを簡単に構成できます。

ルートプール内のディスクを交換する方法については、124 ページの「ZFS ルートプールのディスクを交換する方法 (SPARC または x86/VTOC)」を参照してください。

1 ルートプールの現在のステータスを表示します。

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

        NAME      STATE     READ WRITE CKSUM
        rpool     ONLINE       0     0     0
          c2t0d0  ONLINE       0     0     0

errors: No known data errors
```

2 ミラー化ルートプール構成にするために、2 つ目のディスクを接続します。

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

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。

ルートプールディスク上のパーティションをカスタマイズした場合は、次のような構文が必要になることがあります。

```
# zpool attach rpool c2t0d0s0 c2t1d0
```

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。

```
# zpool status rpool
  pool: rpool
 state: DEGRADED
status: One or more devices is currently being resilvered.  The pool will
        continue to function in a degraded state.
action: Wait for the resilver to complete.
        Run 'zpool status -v' to see device specific details.
  scan: resilver in progress since Fri Jul 20 13:52:05 2012
    809M scanned out of 11.6G at 44.9M/s, 0h4m to go
    776M resilvered, 6.82% done
config:

        NAME        STATE     READ WRITE CKSUM
        rpool       DEGRADED     0     0     0
          mirror-0  DEGRADED     0     0     0
            c8t0d0  ONLINE       0     0     0
            c8t1d0  DEGRADED     0     0     0  (resilvering)

errors: No known data errors
```

上の出力の場合、再同期化処理は完了していません。次のようなメッセージが表示されたら、再同期化が完了しています。

```
resilvered 11.6G in 0h5m with 0 errors on Fri Jul 20 13:57:25 2012
```

4 より大きいディスクを接続する場合は、プールの **autoexpand** プロパティーを設定して、プールのサイズを拡張します。

既存の rpool プールサイズを判別します。

```
# 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
```

拡張した rpool プールサイズを確認します。

```
# 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。

▼ ZFS ルートプールのディスクを交換する方法 (SPARC または x86/VTOC)

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。非冗長プールでディスクに障害が発生してシステムがブートしない場合は、CD やネットワークなどの代替メディアからブートしたあとでルートプールのディスクを置き換える必要があります。
- zpool replace コマンドを使用して、ルートプールディスクでディスクを交換する場合は、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、代替メディアからブートしなくてもディスクの置き換えを試行できる場合があります。zpool replace コマンドを使用して、障害が発生しているディスクを置き換えたり、追加ディスクがある場合は zpool attach コマンドを使用したりできます。追加ディスクの接続やルートプールディスクの切り離しの例については、次の手順を参照してください。

SATA ディスクを備えたシステムでは、故障したディスクを交換するための zpool replace 操作を試みる前に、ディスクをオフラインにして構成解除する必要があります。例:

```
# zpool offline rpool c1t0d0s0
# cfgadm -c unconfigure c1::dsk/c1t0d0
<Physically remove failed disk c1t0d0>
<Physically insert replacement disk c1t0d0>
# cfgadm -c configure c1::dsk/c1t0d0
```

```
<Confirm that the new disk has an SMI label and a slice 0>
# zpool online rpool c1t0d0s0
# zpool replace rpool c1t0d0s0
# zpool status rpool
<Let disk resilver before installing the boot blocks>
# bootadm install-bootloader
```

一部のハードウェアでは、交換用ディスクの装着後にそのディスクをオンラインにしたり再構成を行なったりする必要がありません。

1 交換用ディスクを物理的に接続します。

2 必要に応じて、ルートプールに接続する 2 つ目のディスクを準備します。

- SPARC: 交換用 (新しい) ディスクに SMI (VTOC) ラベルが付いていてスライス 0 があることを確認してください。ルートプールに使用するディスクのラベルを変更する方法については、『Oracle Solaris 11.1 の管理: デバイスとファイルシステム』の「ディスクラベルを作成する方法」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てスライス 0 を作成する必要がある場合は、『Oracle Solaris 11.1 の管理: デバイスとファイルシステム』の「ZFS ルートファイルシステム用のディスクを設定する方法」を参照してください。

3 新しいディスクをルートプールに接続します。

例:

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

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。

4 ルートプールのステータスを確認します。

例:

```
# 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
```

5 再同期化が完了したあとで新しいディスクからブートできることを確認します。

SPARC システムの場合、たとえば次のようになります。

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

交換用ディスクからのブートをテストできるように、また、交換用ディスクに障害が発生した場合に必要に応じて既存のディスクから手動でブートできるように、現在のディスクと新しいディスクのブートデバイスのパス名を特定します。次の例では、現在のルートプールのディスク (c2t0d0s0) は次のとおりです。

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

次の例で、交換用ブートディスク (c2t1d0s0) は次のとおりです。

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

6 新しいディスクからシステムがブートした場合は、古いディスクを切り離します。

例:

```
# zpool detach rpool c2t0d0s0
```

7 より大きいディスクを接続する場合は、プールの **autoexpand** プロパティーを設定して、プールのサイズを拡張します。

```
# zpool set autoexpand=on rpool
```

または、デバイスを拡張します。

```
# zpool online -e c2t1d0s0
```

8 新しいディスクから自動的にブートするようシステムをセットアップします。

- SPARC: システムが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。そのためには、eeprom コマンドまたはブート PROM の setenv コマンドを使用します。
- x86: システム BIOS を再構成します。

▼ ZFS ルートプールのディスクを交換する方法 (SPARC または x86/EFI (GPT))

Oracle Solaris 11.1 リリースでは、ほとんどの場合、デフォルトで x86 ベースシステムに EFI (GPT) ラベルがインストールされます。

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。非冗長プールでディスクに障害が発生してシステムがブートしない場合は、CD やネットワークなどの代替メディアからブートしたあとでルートプールのディスクを置き換える必要があります。
- zpool replace コマンドを使用して、ルートプールディスクでディスクを交換する場合は、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、代替メディアからブートしなくてもディスクの置き換えを試行できる場合があります。zpool replace コマンドを使用して、障害が発生しているディスクを置き換えたり、追加ディスクがある場合は zpool attach コマンドを使用したりできます。追加ディスクの接続やルートプールディスクの切り離しの例については、次の手順を参照してください。

SATA ディスクを備えたシステムでは、故障したディスクを交換するための zpool replace 操作を試みる前に、ディスクをオフラインにして構成解除する必要があります。例:

```
# zpool offline rpool c1t0d0
# cfgadm -c unconfigure c1::dsk/c1t0d0
<Physically remove failed disk c1t0d0>
<Physically insert replacement disk c1t0d0>
# cfgadm -c configure c1::dsk/c1t0d0
# zpool online rpool c1t0d0
# zpool replace rpool c1t0d0
# zpool status rpool
<Let disk resilver before installing the boot blocks>
x86# bootadm install-bootloader
```

一部のハードウェアでは、交換用ディスクの装着後にそのディスクをオンラインにしたり再構成を行なったりする必要がありません。

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。

例:

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

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確認します。

例:

```
# zpool status rpool
  pool: rpool
 state: ONLINE
  scan: resilvered 11.6G in 0h5m with 0 errors on Fri Jul 20 12:06:07 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
```

4 再同期化が完了したあとで新しいディスクからブートできることを確認します。

5 新しいディスクからシステムがブートした場合は、古いディスクを切り離します。
例:

```
# zpool detach rpool c2t0d0
```

6 より大きいディスクを接続する場合は、プールの **autoexpand** プロパティーを設定して、プールのサイズを拡張します。

```
# zpool set autoexpand=on rpool
```

または、デバイスを拡張します。

```
# zpool online -e c2t1d0
```

7 新しいディスクから自動的にブートするようシステムをセットアップします。
システム BIOS を再構成します。

▼ 別のルートプール内で BE を作成する方法 (SPARC または x86/VTOC)

別のルートプール内で既存の BE を再作成する場合は、次の手順に従ってください。独立したスワップデバイスおよびダンプデバイスを備えた同様の BE を持つ 2 つのルートプールが必要か、それともスワップデバイスおよびダンプデバイスを共有する別のルートプールに BE を作成するだけかに基づき、手順を変更できます。

2 番目のルートプール内の新しい BE からアクティブ化しブートしたあと、1 番目のルートプール内の以前の BE に関する情報はなくなります。元の BE に戻す場合、元のルートプールのブートディスクから手動でシステムをブートする必要があります。

1 **SMI (VTOC)** ラベルの付いたディスクで **2** 番目のルートプールを作成します。例:

```
# zpool create rpool2 c4t2d0s0
```

2 **2** 番目のルートプール内に新しい **BE** を作成します。例:

```
# beadm create -p rpool2 solaris2
```

3 **2** 番目のルートプールで **bootfs** プロパティーを設定します。例:

```
# zpool set bootfs=rpool2/ROOT/solaris2 rpool2
```

4 新しい **BE** をアクティブにします。例:

```
# beadm activate solaris2
```

5 新しい **BE** からブートしますが、**2** 番目のルートプールのブートデバイスから明示的にブートする必要があります。

```
ok boot disk2
```

システムは新しい BE 下で実行しています。

6 スワップボリュームを再作成します。例:

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

7 新しいスワップデバイスの **/etc/vfstab** エントリを更新します。例:

```
/dev/zvol/dsk/rpool2/swap      -               -               swap -     no      -
```

8 ダンプボリュームを再作成します。例:

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

9 ダンプデバイスをリセットします。例:

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

10 **2** 番目のルートプールのブートディスクからブートするように、デフォルトのブートデバイスをリセットします。

- SPARC – システムが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。そのためには、`eeprom` コマンドまたはブート PROM の `setenv` コマンドを使用します。
- x86 – システム BIOS を再構成します。

11 リブートして、元のルートプールのスワップデバイスとダンプデバイスをクリアします。

```
# init 6
```

▼ 別のルートプール内で BE を作成する方法 (SPARC または x86/EFI (GPT))

Oracle Solaris 11.1 リリースでは、ほとんどの場合、デフォルトで x86 ベースシステムに EFI (GPT) ラベルがインストールされます。

別のルートプール内で既存の BE を再作成する場合は、次の手順に従ってください。独立したスワップデバイスおよびダンプデバイスを備えた同様の BE を持つ 2 つのルートプールが必要か、それともスワップデバイスおよびダンプデバイスを共有する別のルートプールに BE を作成するだけかに基づき、手順を変更できます。

2 番目のルートプール内の新しい BE からアクティブ化しブートしたあと、1 番目のルートプール内の以前の BE に関する情報はなくなります。元の BE に戻す場合、元のルートプールのブートディスクから手動でシステムをブートする必要があります。

1 代替ルートプールを作成します。

```
# zpool create -B rpool2 c2t2d0
```

あるいは、ミラー化された代替ルートプールを作成します。例:

```
# zpool create -B rpool2 mirror c2t2d0 c2t3d0
```

2 **2** 番目のルートプール内に新しい **BE** を作成します。例:

```
# beadm create -p rpool2 solaris2
```

3 **2** 番目のルートプールにブート情報を適用します。例:

```
# bootadm install-bootloader -P rpool2
```

4 **2** 番目のルートプールで **bootfs** プロパティーを設定します。例:

```
# zpool set bootfs=rpool2/ROOT/solaris2 rpool2
```

5 新しい **BE** をアクティブにします。例:

```
# beadm activate solaris2
```

6 新しい **BE** からブートします。

- SPARC – システムが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。そのためには、`eeprom` コマンドまたはブート PROM の `setenv` コマンドを使用します。
- x86 – システム BIOS を再構成します。

システムは新しい BE 下で実行しています。

7 スワップボリュームを再作成します。例:

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

8 新しいスワップデバイスの **/etc/vfstab** エントリを更新します。例:

```
/dev/zvol/dsk/rpool2/swap      -               -               swap -     no      -
```

9 ダンプボリュームを再作成します。例:

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

10 ダンプデバイスをリセットします。例:

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

11 リブートして、元のルートプールのスワップデバイスとダンプデバイスをクリアします。

```
# init 6
```

ZFS スワップデバイスおよびダンプデバイスを管理する

インストールプロセス中に、スワップ領域は ZFS ルートプール内の ZFS ボリュームに作成されます。次に例を示します。

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

インストールプロセス中に、ダンプデバイスは ZFS ルートプール内の ZFS ボリュームに作成されます。ダンプデバイスは一般に、インストール時に自動的に設定されるため、管理の必要はありません。例:

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

ダンプデバイスを無効にして削除した場合、ダンプデバイスを作成し直したあと、`dumpadm` コマンドを使ってデバイスを有効にする必要があります。ほとんどの場合、`zfs` コマンドを使ってダンプデバイスのサイズを調整するだけで済みます。

インストールプログラムで作成されるスワップボリュームとダンプボリュームのサイズについては、116 ページの「ZFS ルートプールの一般的な要件」を参照してください。

スワップボリュームのサイズとダンプボリュームのサイズはどちらも、インストール後に調整することができます。詳細は、132 ページの「ZFS スワップデバイスおよびダンプデバイスのサイズを調整する」を参照してください。

ZFS のスワップデバイスとダンプデバイスを操作する場合には、次の問題を考慮してください。

- スワップ領域とダンプデバイスには別個の ZFS ボリュームを使用する必要があります。
- 現時点では、ZFS ファイルシステムでスワップファイルを使用することはできません。
- システムのインストール後にスワップ領域やダンプデバイスを変更する必要がある場合は、以前の Solaris リリースと同様に `swap` コマンドと `dumpadm` コマンドを使用します。詳細は、『Oracle Solaris 11.1 の管理: デバイスとファイルシステム』の

第 16 章「追加スワップ空間の構成 (タスク)」および『Oracle Solaris 11.1 での一般的な問題のトラブルシューティング』の第 1 章「システムクラッシュ情報の管理 (タスク)」を参照してください。

ZFS スワップデバイスおよびダンプデバイスのサイズを調整する

インストール後にスワップデバイスとダンプデバイスのサイズを調整したり、場合によってはスワップボリュームとダンプボリュームを再作成することが必要な場合もあります。

- ダンプデバイスの `volsize` プロパティーは、システムのインストール後に再設定することができます。例:

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

- スワップボリュームのサイズは変更できますが、増加したスワップ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。例:

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

  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 11.1 の管理: デバイスとファイルシステム』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。

- すでにインストールされているシステム上でさらに多くのスワップ領域を必要とし、スワップデバイスがビジー状態の場合は、別のスワップボリュームを追加するだけです。例:

```
# zfs create -V 2G rpool/swap2
```

- 新しいスワップボリュームをアクティブにします。例:

```
# swap -a /dev/zvol/dsk/rpool/swap2
# swap -l
swapfile                  dev  swaplo   blocks   free
/dev/zvol/dsk/rpool/swap  256,1      16 1058800 1058800
/dev/zvol/dsk/rpool/swap2 256,3      16 4194288 4194288
```

- 2 つ目のスワップボリュームのエントリを `/etc/vfstab` ファイルに追加します。例:

```
/dev/zvol/dsk/rpool/swap2     -           -           swap    -    no    -
```

ZFS ダンプデバイスの問題のトラブルシューティング

システムクラッシュダンプの取得やダンプデバイスのサイズ変更で問題が発生した場合には、次の項目を確認してください。

- クラッシュダンプが自動的に作成されなかった場合は、savecore コマンドを使ってクラッシュダンプを保存することができます。
- ZFS ルートファイルシステムの初期インストール時や ZFS ルートファイルシステムへの移行時に、ダンプデバイスが自動的に作成されます。ダンプデバイスのデフォルトサイズが小さすぎる場合には、ほとんどの場合、ダンプデバイスのサイズを調整するだけで済みます。たとえば、大量のメモリーが搭載されたシステムでは、次のようにダンプデバイスのサイズを 40G バイトに増やします。

  ```
  # zfs set volsize=40G rpool/dump
  ```

  大きなサイズのダンプデバイスのサイズ変更処理には、長い時間がかかる可能性があります。

  何らかの理由で、ダンプデバイスを手動で作成したあとでそのデバイスを有効化する必要がある場合には、次のような構文を使用します。

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

- 128G バイト以上のメモリーが搭載されたシステムでは、デフォルトで作成されるダンプデバイスよりも大きいダンプデバイスが必要となります。ダンプデバイスが小さすぎて既存のクラッシュダンプを取得できない場合には、次のようなメッセージが表示されます。

  ```
  # dumpadm -d /dev/zvol/dsk/rpool/dump
  dumpadm: dump device /dev/zvol/dsk/rpool/dump is too small to hold a system dump
  dump size 36255432704 bytes, device size 34359738368 bytes
  ```

  スワップデバイスやダンプデバイスのサイジングについては、『Oracle Solaris 11.1 の管理: デバイスとファイルシステム』の「スワップ空間の計画」を参照してください。

- 現在のところ、複数の最上位デバイスを含むプールにダンプデバイスを追加することはできません。次のようなメッセージが表示されます。

  ```
  # dumpadm -d /dev/zvol/dsk/datapool/dump
  dump is not supported on device '/dev/zvol/dsk/datapool/dump':
  'datapool' has multiple top level vdevs
  ```

  ダンプデバイスは、最上位デバイスを複数持つことのできないルートプールに追加してください。

ZFS ルートファイルシステムからのブート

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。ZFS ルートファイルシステムからのブート時には、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、ブート用に選択されたルートファイルシステム内で解決されます。

ZFS では、単一のルートファイルシステムではなくストレージプールがデバイス指定子で指定されるため、ZFS ファイルシステムからのブートは UFS ファイルシステムからのブートとは異なります。ストレージプールには、複数のブート可能 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。ZFS からブートする場合は、ブートデバイスと、ブートデバイスによって指定されたプール内のルートファイルシステムを指定する必要があります。

デフォルトでは、プールの `bootfs` プロパティーで指定されているファイルシステムが、ブート用に選択されます。このデフォルト選択は、SPARC システムで `boot -Z` コマンドに代替のブート可能ファイルシステムを指定するか、x86 システムで BIOS から代替のブートデバイスを選択することによって上書きできます。

ミラー化された ZFS ルートプールの代替ディスクからブートする

インストール後にディスクを追加して、ミラー化された ZFS ルートプールを作成することができます。ミラー化ルートプールの作成の詳細は、121 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。

ミラー化された ZFS ルートプールに関して、次に示す既知の問題を確認してください。

- ミラー化された ZFS ルートプールのさまざまなデバイスからブートすることができます。ハードウェア構成によっては、別のブートデバイスを指定するには、PROM または BIOS の更新が必要になる場合があります。

  たとえば、このプール内のどちらかのディスク (`c1t0d0s0` または `c1t1d0s0`) からブートできます。

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

        NAME          STATE     READ WRITE CKSUM
        rpool         ONLINE       0     0     0
          mirror-0    ONLINE       0     0     0
            c1t0d0s0  ONLINE       0     0     0
            c1t1d0s0  ONLINE       0     0     0
```

SPARC システムでは、ok プロンプトに代替ディスクを入力します。

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

システムがリブートしたら、アクティブなブートデバイスを確認します。例:

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

x86 システムで、次のような構文を使用します。

```
x86# prtconf -v|sed -n '/bootpath/,/value/p'
        name='bootpath' type=string items=1
            value='/pci@0,0/pci8086,25f8@4/pci108e,286@0/disk@0,0:a'
```

- x86 システムでは、ミラー化された ZFS ルートプールの代替ディスクを、適切な BIOS メニューで選択します。
- **SPARC** または **x86**: zpool replace コマンドを使用してルートプールディスクを置き換える場合、bootadm コマンドを使用して、新しく置き換えられるディスクのブート情報をインストールする必要があります。初期インストール方法を使用して、ミラー化された ZFS ルートプールを作成するか、zpool attach コマンドを使用してディスクをルートプールに接続する場合、この手順は不要です。bootadm の構文は次のようになります:

  ```
  # bootadm install-bootloader
  ```

  代替ルートプールにブートローダーをインストールする場合は、-P (プール) オプションを使用します。

  ```
  # bootadm install-bootloader -P rpool2
  ```

  GRUB レガシーブートローダーをインストールする場合は、レガシー installgrub コマンドを使用します。

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

- ミラー化された ZFS ルートプールのさまざまなデバイスからブートすることができます。ハードウェア構成によっては、別のブートデバイスを指定するには、PROM または BIOS の更新が必要になる場合があります。

  たとえば、このプール内のどちらかのディスク (c1t0d0s0 または c1t1d0s0) からブートできます。

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

          NAME          STATE     READ WRITE CKSUM
          rpool         ONLINE       0     0     0
            mirror-0    ONLINE       0     0     0
              c1t0d0s0  ONLINE       0     0     0
              c1t1d0s0  ONLINE       0     0     0
  ```

  SPARC システムでは、ok プロンプトに代替ディスクを入力します。

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

システムがリブートしたら、アクティブなブートデバイスを確認します。例:

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

x86 システムで、次のような構文を使用します。

```
x86# prtconf -v|sed -n '/bootpath/,/value/p'
        name='bootpath' type=string items=1
            value='/pci@0,0/pci8086,25f8@4/pci108e,286@0/disk@0,0:a'
```

- x86 システムでは、ミラー化された ZFS ルートプールの代替ディスクを、適切な BIOS メニューで選択します。

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 SPARC システムでは、`beadm activate` コマンドを使用することによって、任意の BE からブートできます。

インストールおよび `beadm` アクティブ化処理中に、ZFS ルートファイルシステムが `bootfs` プロパティーで自動的に指定されます。

ブート可能なファイルシステムがプール内に複数存在する場合があります。デフォルトでは、*/pool-name*`/boot/menu.lst` ファイルのブート可能ファイルシステムのエントリは、プールの `bootfs` プロパティーで指定されます。ただし、`menu.lst` のエントリに `bootfs` コマンドを含めて、プールの代替ファイルシステムを指定することもできます。このように、`menu.lst` ファイルには、プール内の複数のルートファイルシステムに対応するエントリが含まれている場合があります。

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

```
title Oracle Solaris 11.1 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 Oracle Solaris 11.1 SPARC
bootfs rpool/ROOT/solaris
title solaris
bootfs rpool/ROOT/solaris2
```

SPARC ベースシステムでは、次のようにしてブート元の BE を選択できます。

- ZFS BE がアクティブになったあと、boot -L コマンドを使用して ZFS プール内のブート可能なファイルシステムのリストを表示できます。その後、ブート可能なファイルシステムの 1 つをリストで選択できます。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。手順に従って、選択したファイルシステムをブートできます。
- 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。

このブート方法では、BE は自動的にアクティブ化されません。boot -L および -Z 構文で BE をブートしたあと、この BE をアクティブ化して、そこから自動的にブートし続けるようにする必要があります。

例 4–1 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上の ZFS ストレージプールに複数の ZFS BE が存在する場合は、beadm activate コマンドを使用してデフォルトの BE を指定できます。

たとえば、次の ZFS BE が beadm の出力のとおり使用可能であるとします。

```
# 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
```

SPARC システムに複数の ZFS BE が存在する場合は、boot -L コマンドを使用できます。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 Oracle Solaris 11.1 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
```

上記のコマンドでブートした BE は、次回のリブートにはアクティブになっていないことに留意してください。boot -Z 操作中に選択した BE から自動的にブートし続けるようにする場合は、この BE をアクティブにする必要があります。

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 では、x86 システムにレガシー GRUB がインストールされ、インストールプロセス中または `beadm activate` 操作中に、ZFS を自動的にブートするための次のようなエントリが */pool-name*`/boot/grub/menu.lst` ファイルに追加されます。

```
title solaris
bootfs rpool/ROOT/solaris
kernel$ /platform/i86pc/kernel/amd64/unix -B $ZFS-BOOTFS
module$ /platform/i86pc/amd64/boot_archive
title solaris-1
bootfs rpool/ROOT/solaris-1
kernel$ /platform/i86pc/kernel/amd64/unix -B $ZFS-BOOTFS
module$ /platform/i86pc/amd64/boot_archive
```

GRUB によってブートデバイスとして識別されたデバイスに ZFS ストレージプールが含まれている場合、`menu.lst` ファイルを使用して GRUB メニューが作成されます。

複数の ZFS BE が存在する x86 システムでは、BE を GRUB メニューから選択できます。このメニューエントリに対応するルートファイルシステムが ZFS ファイルシステムである場合は、次のオプションが追加されます。

```
-B $ZFS-BOOTFS
```

Oracle Solaris 11.1 では、x86 ベースシステムに GRUB2 が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ますが、このファイルを手動で編集しないようにしてください。`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。

GRUB メニュー項目の変更の詳細は、『Oracle Solaris 11.1 システムのブートおよびシャットダウン』を参照してください。

例 4–2　x86: ZFS ファイルシステムをブートする

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 Oracle Solaris 11.1
```

レガシー GRUB 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、ルートデバイスはブート用の `-B $ZFS-BOOTFS` パラメータによって指定されます。例:

例 4–2 x86: ZFS ファイルシステムをブートする (続き)

```
title solaris
bootfs rpool/ROOT/solaris
kernel$ /platform/i86pc/kernel/amd64/unix -B $ZFS-BOOTFS
module$ /platform/i86pc/amd64/boot_archive
title solaris-1
bootfs rpool/ROOT/solaris-1
kernel$ /platform/i86pc/kernel/amd64/unix -B $ZFS-BOOTFS
module$ /platform/i86pc/amd64/boot_archive
```

例 4–3 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。システムを高速にリブートするこの機能により、大幅にダウンタイムが短縮され、効率が向上します。

`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。`reboot` コマンドが適切となるほかのシステム操作については、`reboot -f` コマンドを使用できます。例:

```
# reboot -f
```

ZFS ルート環境での回復のためのブート

失われたルートパスワードやそれに似た問題から回復するためにシステムをブートする必要がある場合は、次の手順を使用します。

▼ 復旧のためにシステムをブートする方法

`menu.lst` の問題やルートパスワードの問題を伴う問題を解決するには、次の手順を使用します。ルートプールのディスクを交換する必要がある場合は、124 ページの「ZFS ルートプールのディスクを交換する方法 (SPARC または x86/VTOC)」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、第 11 章「スナップショットのアーカイブとルートプールの回復」を参照してください。

1 適切なブート方法を選択します。

- x86: Live Media - インストールメディアからブートし、回復手順のために GNOME 端末を使用します。
- SPARC: テキストインストール – インストールメディアまたはネットワークからブートし、テキストインストール画面からオプション「`3 Shell`」を選択します。

- x86: テキストインストール – GRUB メニューから「Text Installer and command line」ブートエントリを選択し、次にテキストインストール画面からオプション「3 Shell」を選択します。
- SPARC: 自動インストール – 次のコマンドを使用して、シェルに出られるインストールメニューから直接ブートします。

  ```
  ok boot net:dhcp
  ```
- x86: 自動インストール – ネットワーク上のインストールサーバーからのブートには PXE ブートが必要です。GRUB メニューから「Text Installer and command line」エントリを選択します。次に、テキストインストール画面からオプション「3 Shell」を選択します。

たとえば、システムがブートしたあとで、オプション「3 Shell」を選択します。

```
        1  Install Oracle Solaris
        2  Install Additional Drivers
        3  Shell
        4  Terminal type (currently xterm)
        5  Reboot

Please enter a number [1]: 3
To return to the main menu, exit the shell
#
```

2 ブート回復の問題を選択します。

- システムをシングルユーザーモードでブートし、/etc/passwd ファイルのシェルエントリを修正して、不正なルートシェルを解決します。

  x86 システムでは、選択したブートエントリを編集して -s オプションを追加します。

  たとえば、SPARC システムでは、システムをシャットダウンしてシングルモードでブートします。root としてログインしたら、/etc/passwd ファイルを編集してルートシェルエントリを修正します。

  ```
  # init 0
  ok boot -s
  Boot device: /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/disk@0,0:a ...
  SunOS Release 5.11 Version 11.1 64-bit
  Copyright (c) 1983, 2012, Oracle and/or its affiliates. All rights reserved.
  Booting to milestone "milestone/single-user:default".
  Hostname: tardis.central
  Requesting System Maintenance Mode
  SINGLE USER MODE

  Enter user name for system maintenance (control-d to bypass): root
  Enter root password (control-d to bypass): xxxx
  single-user privilege assigned to root on /dev/console.
  Entering System Maintenance Mode

  Aug  3 15:46:21 su: 'su root' succeeded for root on /dev/console
  Oracle Corporation      SunOS 5.11      11.1    October 2012
  su: No shell /usr/bin/mybash.  Trying fallback shell /sbin/sh.
  ```

```
root@tardis.central:~# TERM =vt100; export TERM
root@tardis.central:~# vi /etc/passwd
root@tardis.central:~# <Press control-d>
logout
svc.startd: Returning to milestone all.
```

- x86 ベースシステムのブートを妨げている問題を解決します。

  最初に、手順 1 に示したいずれかのブート方法を使用してメディアまたはネットワークからブートする必要があります。次に、たとえば、ルートプールをインポートして GRUB エントリを修正します。

  bootadm list-menu コマンドを使用すると、GRUB2 エントリを一覧表示して変更できます。また、set-menu サブコマンドを使用すると、ブートエントリを変更できます。詳細は、bootadm(1M) を参照してください。

```
x86# zpool import -f rpool
x86# bootadm list-menu
x86# bootadm set-menu default=1
x86# exit
       1  Install Oracle Solaris
       2  Install Additional Drivers
       3  Shell
       4  Terminal type (currently sun-color)
       5  Reboot

Please enter a number [1]: 5
```

  システムが正常にブートすることを確認します。

- システムにログインできない原因となっている不明なルートパスワードを解決します。

  最初に、手順 1 に示したいずれかのブート方法を使用してメディアまたはネットワークからブートする必要があります。次に、ルートプール (rpool) をインポートし、BE をマウントしてルートパスワードエントリを削除します。この処理は、SPARC プラットフォームと x86 プラットフォームで同一です。

```
# zpool import -f rpool
# beadm list
be_find_current_be: failed to find current BE name
be_find_current_be: failed to find current BE name
BE        Active Mountpoint Space  Policy Created
--        ------ ---------- -----  ------ -------
solaris   -      -          46.95M static 2012-07-20 10:25
solaris-2 R      -          3.81G  static 2012-07-19 13:44
# mkdir /a
# beadm mount solaris-2 /a
# TERM=vt100
# export TERM
# cd /a/etc
# vi shadow
<Carefully remove the unknown password>
# cd /
# beadm umount solaris-2
# halt
```

  次の手順に進んでルートパスワードを設定します。

3 シングルユーザーモードでブートしてパスワードを設定することで、ルートパスワードを設定します。

この手順は、前の手順で不明なルートパスワードが削除されていることを前提としています。

x86 ベースシステムでは、選択したブートエントリを編集して -s オプションを追加します。

SPARC ベースシステムでは、システムをシングルユーザーモードでブートし、root としてログインして、root パスワードを設定します。例:

```
ok boot -s
Boot device: /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/disk@0,0:a ...
SunOS Release 5.11 Version 11.1 64-bit
Copyright (c) 1983, 2012, Oracle and/or its affiliates. All rights reserved
Booting to milestone "milestone/single-user:default".

Enter user name for system maintenance (control-d to bypass): root
Enter root password (control-d to bypass): <Press return>
single-user privilege assigned to root on /dev/console.
Entering System Maintenance Mode

Jul 20 14:09:59 su: 'su root' succeeded for root on /dev/console
Oracle Corporation      SunOS 5.11      11.1    October 2012
root@tardis.central:~# passwd -r files root
New Password: xxxxxx
Re-enter new Password: xxxxxx
passwd: password successfully changed for root
root@tardis.central:~# <Press control-d>
logout
svc.startd: Returning to milestone all.
```

◆ ◆ ◆ 第 5 章

Oracle Solaris ZFS ファイルシステムの管理

この章では、Oracle Solaris ZFS ファイルシステムの管理について詳しく説明します。ファイルシステムの階層レイアウト、プロパティーが継承されること、およびマウントポイント管理および共有が自動的に行われることなどについて、それらの概念を説明しています。

この章は、次のセクションで構成されます。

ZFS ファイルシステムの管理 (概要)

ZFS ファイルシステムは、ストレージプールの最上位に構築されます。ファイルシステムは動的に作成および破棄することができ、基礎となるディスク領域を割り当てたりフォーマットしたりする必要はありません。ファイルシステムが非常に軽量であることと、ZFS はファイルシステムに基づいて管理することから、作成されるファイルシステムの数は多くなる傾向があります。

ZFS ファイルシステムの管理には、`zfs` コマンドを使用します。`zfs` コマンドには、ファイルシステムに特定の操作を実行するために一連のサブコマンドが用意されています。この章では、これらのサブコマンドについて詳細に説明します。スナップショット、ボリューム、およびクローンもこのコマンドを使って管理します

が、これらの機能についてはこの章では簡単に取り上げるだけにとどめます。スナップショットおよびクローンの詳細については、第 6 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、279 ページの「ZFS ボリューム」を参照してください。

注 – 「データセット」という用語は、この章ではファイルシステム、スナップショット、クローン、またはボリュームの総称として使用します。

ZFS ファイルシステムの作成、破棄、および名前変更を行う

ZFS ファイルシステムは、`zfs create` および `zfs destroy` コマンドを使って作成および破棄できます。`zfs rename` コマンドを使用して、ZFS ファイルシステムの名前を変更できます。

ZFS ファイルシステムを作成する

ZFS ファイルシステムの作成には、`zfs create` コマンドを使用します。`create` サブコマンドの引数は 1 つだけです。作成するファイルシステムの名前です。ファイルシステム名は、次のようにプール名から始まるパス名として指定します。

pool-name/[filesystem-name/]filesystem-name

パスのプール名と最初のファイルシステム名は、新規ファイルシステムを階層内のどこに作成するかを示しています。パスの最後にある名前は、作成するファイルシステムの名前です。ファイルシステム名は、32 ページの「ZFS コンポーネントに名前を付けるときの規則」の命名要件に従って付ける必要があります。

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、201 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。

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
```

新しく作成するファイルシステムが正常に作成されると、自動的にマウントされます。ファイルシステムは、デフォルトでは `create` サブコマンドのファイルシステム

名に指定したパスを使って、*/dataset* としてマウントされます。この例では、新しく作成した 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、179 ページの「ZFS マウントポイントを管理する」を参照してください。

zfs create コマンドの詳細については、zfs(1M) を参照してください。

ファイルシステムの作成時にファイルシステムのプロパティーを設定できます。

次の例では、tank/home ファイルシステム用に /export/zfs というマウントポイントが作成されます。

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

ファイルシステムのプロパティーの詳細については、147 ページの「ZFS のプロパティーの紹介」を参照してください。

ZFS ファイルシステムを破棄する

ZFS ファイルシステムを破棄するには、zfs destroy コマンドを使用します。破棄されたファイルシステムは、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および共有の詳細については、180 ページの「自動マウントポイント」を参照してください。

次の例では、tank/home/mark ファイルシステムが破棄されます。

```
# 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。

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 tank/home/matt
cannot unmount 'tank/home/matt': Device busy

# zfs destroy -f tank/home/matt
```

zfs destroy コマンドは、ファイルシステムに子孫が存在する場合にも失敗します。ファイルシステムとそのすべての子孫を再帰的に破棄するときは、-r オプションを使用します。再帰的な破棄を実行すると、スナップショットも破棄されるので、このオプションは慎重に使用してください。

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

破棄するファイルシステムに間接的な依存関係が存在する場合は、再帰的な破棄コマンドでも失敗します。破棄する階層の外部に複製されたファイルシステムなど、すべての依存関係を強制的に破棄する場合は、-R オプションを使用する必要があります。このオプションは、慎重に使用してください。

```
# zfs destroy -r tank/home/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
```

注意 – zfs destroy コマンドの -f、-r、または -R オプションでは、確認を求めるプロンプトは表示されないので、これらのオプションは慎重に使用してください。

スナップショットおよびクローンの詳細については、第 6 章「Oracle Solaris ZFS のスナップショットとクローンの操作」を参照してください。

ZFS ファイルシステムの名前を変更する

zfs rename コマンドを使用して、ファイルシステムの名前を変更できます。rename サブコマンドを使用すれば、次の操作を実行できます。

- ファイルシステムの名前を変更します。
- 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。

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。

```
# zfs rename tank/home/eric tank/home/eric_old
```

次の例では、zfs rename を使用してファイルシステムの場所を移動する方法を示しています。

```
# zfs rename tank/home/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、195 ページの「ZFS の割り当て制限と予約を設定する」を参照してください。

rename を実行すると、ファイルシステムおよび子孫のファイルシステム (存在する場合) を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。

スナップショットの名前を変更する方法については、217 ページの「ZFS スナップショットの名前を変更する」を参照してください。

ZFS のプロパティーの紹介

ファイルシステム、ボリューム、スナップショット、およびクローンの動作を制御するときには、主にプロパティーという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。

- 162 ページの「ZFS の読み取り専用のネイティブプロパティー」
- 163 ページの「設定可能な ZFS ネイティブプロパティー」
- 170 ページの「ZFS ユーザープロパティー」

プロパティーは、ネイティブプロパティーとユーザー定義プロパティーの 2 種類に分けられます。ネイティブプロパティーは、内部の統計情報を提供するか、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、170 ページの「ZFS ユーザープロパティー」を参照してください。

設定可能なプロパティーのほとんどは、継承可能なプロパティーでもあります。継承可能なプロパティーとは、親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。

継承可能なプロパティーには必ず、どのようにしてプロパティーが取得されたかを示すソースが関連付けられています。プロパティーのソースには、次の値が記述される可能性があります。

local	そのプロパティーが zfs set コマンドを使用して明示的にデータセットに設定されたことを示しています。174 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、170 ページの「ZFS ユーザープロパティー」を参照してください。

表 5–1　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`	ファイルとディレクトリが作成されるときに ACL エントリをどのように継承するかを制御します。値は、`discard`、`noallow`、`secure`、および `passthrough` です。これらの値については、242 ページの「ACL プロパティー」を参照してください。
`aclmode`	文字列	`groupmask`	`chmod` を実行するときに ACL エントリをどのように変更するかを制御します。値は、`discard`、`groupmask`、および `passthrough` です。これらの値については、242 ページの「ACL プロパティー」を参照してください。
`atime`	ブール型	`on`	ファイルを読み取るときにファイルのアクセス時間を更新するかどうかを制御します。このプロパティーをオフに設定すると、ファイルを読み取るときに書き込みトラフィックが生成されなくなるため、パフォーマンスが大幅に向上する可能性があります。ただし、メールプログラムなどのユーティリティーが予期しない動作をすることがあります。

表 5–1 ZFS のネイティブプロパティーの説明 (続き)

プロパティー名	タイプ	デフォルト値	説明
available	数値	なし	読み取り専用プロパティー。ファイルシステムおよびそのすべての子が利用できるディスク容量を調べます。プール内でほかのアクティビティーが実行されていないことを前提とします。ディスク容量はプール内で共有されるため、プールの物理サイズ、割り当て制限、予約、プール内のほかのデータセットなどのさまざまな要因によって、利用できる容量が制限されることがあります。 このプロパティーの省略名は avail です。 ディスク領域の計上の詳細については、33 ページの「ZFS のディスク領域の計上」を参照してください。
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。 詳細については、165 ページの「canmount プロパティー」を参照してください。

表 5–1 ZFS のネイティブプロパティーの説明 (続き)

プロパティー名	タイプ	デフォルト値	説明
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、166 ページの「casesensitivity プロパティー」を参照してください。 casesensitivity プロパティーの設定とは関係なく、ファイルシステムは、ファイルの作成に指定された名前の大文字と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	データの完全性を検証するために使用するチェックサムを制御します。デフォルト値は on で、適切なアルゴリズム (現在は fletcher4) が自動的に選択されます。値は、on、off、fletcher2、fletcher4、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。値は on、 off、lzjb、gzip、および gzip-*N* です。現時点では、このプロパティーを lzjb、gzip、または gzip-*N* に設定することは、このプロパティーを on に設定することと同じ効果を持ちます。既存のデータを持つファイルシステムで compression を有効にした場合は、新しいデータのみが圧縮されます。既存のデータは圧縮されないまま残されます。 このプロパティーの省略名は compress です。

表 5–1 ZFS のネイティブプロパティーの説明 (続き)

プロパティー名	タイプ	デフォルト値	説明
compressratio	数値	なし	読み取り専用プロパティー。データセットに適用された圧縮率を調べます。乗数で表現されます。zfs set compression=on *dataset* コマンドを使用すると、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、新たに書き出されるデータが影響を受けるだけなので、ファイルシステムの作成時にこのプロパティーの設定を検討してください。
creation	文字列	なし	読み取り専用プロパティー。このデータセットが作成された日時を調べます。
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、167 ページの「dedup プロパティー」を参照してください。
devices	ブール型	on	ファイルシステム内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	ファイルシステムを暗号化す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	ファイルシステムに含まれるプログラムの実行を許可するかどうかを制御します。off に設定した場合は、PROT_EXEC による mmap(2) 呼び出しも許可されません。

表 5–1 ZFS のネイティブプロパティーの説明 (続き)

プロパティー名	タイプ	デフォルト値	説明
keychangedate	文字列	なし	指定されたファイルシステムに対する zfs key -c 操作からラッピング鍵が最後に変更された日付を特定します。鍵の変更操作が行われなかった場合、この読み取り専用プロパティーの値はファイルシステムの作成日と同じになります。
keysource	文字列	なし	ファイルシステムの鍵をラップする鍵の形式と場所を指定します。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、または読み込むときに、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のステータスを識別する読み取り専用のプロパティー。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	このファイルシステムに対する同期要求を ZFS で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。

表 5–1 ZFS のネイティブプロパティーの説明 (続き)

プロパティー名	タイプ	デフォルト値	説明
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。 信頼できる拡張機能ゾーンにファイルシステムを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、信頼できる拡張機能が有効になり、適切な特権だけが与えられている場合にのみ変更できます。
mounted	ブール型	なし	読み取り専用のプロパティー。ファイルシステム、クローン、またはスナップショットが現在マウントされているかどうかを調べます。このプロパティーは、ボリュームには適用されません。値には yes または no を指定できます。
mountpoint	文字列	なし	このファイルシステムで使用されるマウントポイントを制御します。ファイルシステムの mountpoint プロパティーを変更すると、そのマウントポイントを継承するファイルシステムおよびそのすべての子孫がアンマウントされます。新しい値が legacy の場合は、アンマウント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、179 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。

表 5–1 ZFS のネイティブプロパティーの説明 (続き)

プロパティー名	タイプ	デフォルト値	説明
multilevel	ブール型	off	このプロパティーは、Trusted Extensions が有効になっているシステムでのみ使用できます。デフォルトはオフです。 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。 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 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。

表 5–1 ZFS のネイティブプロパティーの説明 (続き)

プロパティー名	タイプ	デフォルト値	説明
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。使用している環境にとってキャッシュプロパティーの設定が適切かどうかを判断する必要があります。
nbmand	ブール型	off	ファイルシステムを nbmand (非ブロッキング強制) ロックでマウントするべき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、ファイルシステムで 2 つのファイル名を比較するとき常にファイル名の 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を使用するかを指定します。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	なし	複製されたファイルシステムまたはボリュームのための読み取り専用プロパティー。どのスナップショットからクローンが作成されたかを調べます。クローンが存在する場合には、-r や -f オプションを使用しても、作成元は破棄できません。 複製されていないファイルシステムの origin は none になります。

表 5–1 ZFS のネイティブプロパティーの説明 (続き)

プロパティー名	タイプ	デフォルト値	説明
quota	数値 (または none)	none	ファイルシステムおよびその子孫が消費できるディスク容量を制限します。このプロパティーは、使用されるディスク容量に強い制限値を適用します。容量には、子孫 (ファイルシステムやスナップショットを含む) が使用するすべての容量も含まれます。割り当て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、196 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。
rekeydate	文字列	なし	このファイルシステムに対する zfs key -K または zfs clone - K 操作により、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。
readonly	ブール型	off	データセットを変更できるかどうかを制御します。on に設定すると、変更できなくなります。 このプロパティーの省略名は rdonly です。
recordsize	数値	128K	ファイルシステム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、169 ページの「recordsize プロパティー」を参照してください。
referenced	数値	なし	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。

表 5–1 ZFS のネイティブプロパティーの説明 (続き)

プロパティー名	タイプ	デフォルト値	説明
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、親データセットが使用するディスク容量に計上されるので、親データセットの割り当て制限と予約を減らすことになります。 refreservation を設定すると、スナップショットを作成できるのは、データセットの *referenced* の現在のバイト数を格納できる十分な空きプール領域が、この予約容量のほかに存在する場合だけになります。 このプロパティーの省略名は refreserv です。
reservation	数値 (または none)	none	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、199 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。

表 5–1 ZFS のネイティブプロパティーの説明 (続き)

プロパティー名	タイプ	デフォルト値	説明
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。
setuid	ブール型	on	ファイルシステムで setuid ビットを考慮するかどうかを制御します。
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、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については、184 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、184 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については、184 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。

表 5–1 ZFS のネイティブプロパティーの説明 (続き)

プロパティー名	タイプ	デフォルト値	説明
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、213 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。設定可能な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。 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	なし	読み取り専用プロパティー。データセットの種類を調べます。filesystem (ファイルシステムまたはクローン)、volume、または snapshot のいずれかになります。

表 5–1 ZFS のネイティブプロパティーの説明 (続き)

プロパティー名	タイプ	デフォルト値	説明
used	数値	なし	読み取り専用プロパティー。データセットおよびそのすべての子孫が消費するディスク容量を調べます。 詳細については、163 ページの「used プロパティー」を参照してください。
usedbychildren	数値	off	このデータセットの子によって使用されるディスク領域の量を特定する読み取り専用プロパティー。この領域は、データセットのすべての子が破棄されると、解放されます。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領域の量を特定する読み取り専用プロパティー。この領域は、最初にあらゆるスナップショットが破棄されてから refreservation 予約がすべて削除された後に、データセットが破棄されると、解放されます。プロパティーの省略名は usedds です。
usedbyrefreservation	数値	off	データセットに設定されている refreservation によって使用されるディスク領域の量を特定する読み取り専用プロパティー。この領域は、refreservation が削除されると、解放されます。プロパティーの省略名は usedrefreserv です。
usedbysnapshots	数値	off	データセットのスナップショットによって消費されるディスク領域の量を特定する読み取り専用プロパティー。特に、このディスク領域は、このデータセットのすべてのスナップショットが破棄されると、解放されます。この値はスナップショットの used プロパティーの値を単純に合計した結果ではないことに注意してください。複数のスナップショットで共有されている容量も存在するためです。プロパティーの省略名は usedsnap です。
version	数値	なし	ファイルシステムのディスク上バージョンを識別します。プールのバージョンとは無関係です。このプロパティーは、サポートされるソフトウェアリリースから入手可能な最近のバージョンにしか設定できません。詳細については、zfs upgrade コマンドを参照してください。

表 5–1 ZFS のネイティブプロパティーの説明 (続き)

プロパティー名	タイプ	デフォルト値	説明
utf8only	ブール型	Off	このプロパティーは、UTF-8 文字コードセットに存在しない文字が含まれているファイル名をファイルシステムで拒否するかどうかを指定します。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。
volsize	数値	なし	ボリュームの場合に、ボリュームの論理サイズを指定します。 詳細については、169 ページの「volsize プロパティー」を参照してください。
volblocksize	数値	8K バイト	ボリュームの場合に、ボリュームのブロックサイズを指定します。ボリュームが書き込まれたあとに、ブロックサイズを変更することはできません。ブロックサイズはボリュームを作成するときに設定してください。ボリュームのデフォルトブロックサイズは、8K バイトです。512 バイトから 128K バイトの範囲で、任意の 2 の累乗を指定できます。 このプロパティーの省略名は volblock です。
vscan	ブール型	Off	通常ファイルを開くときや閉じるときに、ファイルのウイルススキャンを実行す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。

表 5–1 ZFS のネイティブプロパティーの説明 (続き)

プロパティー名	タイプ	デフォルト値	説明
zoned	ブール型	なし	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指定します。このプロパティーが設定されている場合、そのマウントポイントは大域ゾーンで考慮されません。ZFS では、このようなファイルシステムを要求されても、マウントすることはできません。ゾーンを最初にインストールしたときには、追加されたすべてのファイルシステムにこのプロパティーが設定されます。 ゾーンがインストールされている環境で ZFS を使用する方法の詳細については、282 ページの「ゾーンがインストールされている Solaris システムで ZFS を使用する」を参照してください。
xattr	ブール型	on	このファイルシステムの拡張属性を有効 (on)、無効 (off) のいずれにするかを示します。

ZFS の読み取り専用のネイティブプロパティー

読み取り専用のネイティブプロパティーは、取得はできますが設定はできません。読み取り専用のネイティブプロパティーは継承されません。一部のネイティブプロパティーは、特定の種類のデータセットに固有です。このような場合は、データセットの種類について、表 5–1 の説明の中で記載しています。

読み取り専用のネイティブプロパティーをここに示します。説明は、表 5–1 を参照してください。

- available
- compressratio
- creation
- keystatus
- mounted
- origin
- referenced
- rekeydate
- type
- used

  詳細については、163 ページの「used プロパティー」を参照してください。
- usedbychildren

- usedbydataset
- usedbyrefreservation
- usedbysnapshots

used 、 referenced、available プロパティーなど、ディスク領域の計上の詳細については、33 ページの「ZFS のディスク領域の計上」を参照してください。

used プロパティー

used プロパティーは読み取り専用のプロパティーであり、このデータセットとそのすべての子孫が消費するディスク容量を特定します。この値は、データの割り当て制限および予約を対象にして確認されます。使用されるディスク領域にデータセットの予約は含まれませんが、子孫のデータセットがある場合はそれらの予約も考慮されます。データセットがその親から継承して消費するディスク容量、およびデータセットが再帰的に破棄されるときに解放されるディスク容量は、使用済み領域および予約の中で大きな割合を占めます。

スナップショットを作成したときは、それらのディスク領域は最初はスナップショットとファイルシステムの間で共有されます。それまでに作成したスナップショットと領域が共有されることもあります。ファイルシステムが変化していくにつれて、それまで共有されていたディスク領域がスナップショット固有になり、スナップショットが使用する領域に計上されます。スナップショットが使用するディスク領域には、その固有データが計上されます。また、スナップショットを削除すると、ほかのスナップショットに固有の (および使用される) ディスク容量を増やすことができます。スナップショットと領域の詳細については、35 ページの「領域が不足した場合の動作」を参照してください。

使用済み、使用可能、参照済みの各ディスク容量には、保留状態の変更は含まれません。保留状態の変更は通常、数秒以内に計上されます。fsync(3c) や O_SYNC 関数を使用してディスクへの変更をコミットしても、ディスク領域の使用状況の情報がすぐに更新されることが保証されているわけではありません。

usedbychildren、usedbydataset、usedbyrefreservation、および usedbysnapshots プロパティーの情報は、zfs list - o space コマンドを使用して表示することができます。これらのプロパティーを使用して、used プロパティーを、子孫によって消費されるディスク領域に分解することができます。詳細は、表 5–1 を参照してください。

設定可能な ZFS ネイティブプロパティー

設定可能なネイティブプロパティーとは、値の取得および設定ができるプロパティーのことです。設定可能なネイティブプロパティーは、zfs set コマンド (説明は 174 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 144 ページの「ZFS ファイルシステムを作成する」を参照) を使って設定しま

す。設定可能なネイティブプロパティーは、割り当て制限と予約を除いて継承されます。割り当て制限と予約の詳細については、195 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 5–1 の説明の中で記載しています。特に記載している場合を除いて、プロパティーはすべての種類のデータセットに適用されます。つまり、ファイルシステム、ボリューム、クローン、およびスナップショットに適用されます。

次のプロパティーは設定可能です。説明は、表 5–1 を参照してください。

- `aclinherit`

  詳細については、242 ページの「ACL プロパティー」を参照してください。
- `atime`
- `canmount`
- `casesensitivity`
- `checksum`
- `compression`
- `copies`
- `devices`
- `dedup`
- `encryption`
- `exec`
- `keysource`
- `logbias`
- `mlslabel`
- `mountpoint`
- `nbmand`
- `normalization`
- `primarycache`
- `quota`
- `readonly`
- `recordsize`

  詳細については、169 ページの「`recordsize` プロパティー」を参照してください。
- `refquota`

- refreservation
- reservation
- rstchown
- secondarycache
- share.smb
- share.nfs
- setuid
- snapdir
- version
- vscan
- utf8only
- volsize

  詳細については、169 ページの「volsize プロパティー」を参照してください。
- volblocksize
- zoned
- xattr

canmount プロパティー

canmount プロパティーを off に設定した場合は、zfs mount または zfs mount -a コマンドを使ってファイルシステムをマウントすることはできません。このプロパティーを off に設定する場合は、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、親のファイルシステムは「コンテナ」の役目を果たしているため、そのコンテナにプロパティーを設定することはできますが、コンテナ自体にはアクセスできません。

次の例では、userpool が作成され、その canmount プロパティーが off に設定されます。子孫のユーザーファイルシステムのマウントポイントは、1 つの共通したマウントポイント /export/home に設定されます。親のファイルシステムに設定されたプロパティーは子孫のファイルシステムに継承されますが、親のファイルシステム自体がマウントされることはありません。

```
# zpool create userpool mirror c0t5d0 c1t6d0
# zfs set canmount=off userpool
# zfs set mountpoint=/export/home userpool
# zfs set compression=on userpool
# zfs create userpool/user1
```

```
# zfs create userpool/user2
# zfs mount
userpool/user1                  /export/home/user1
userpool/user2                  /export/home/user2
```

canmount プロパティーを noauto に設定することは、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、266 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 – 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セクション約がデータに有益であるかどうかを判断します。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 によって領域がセクション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、201 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、ファイルシステムに格納するファイルの推奨ブロックサイズを指定します。

このプロパティーは、レコードサイズが固定されているファイルにアクセスするデータベースワークロードだけで使用するように設計されています。ZFS では、標準的なアクセスパターンに最適化された内部アルゴリズムに従って、ブロックサイズが自動的に調整されます。作成されるファイルのサイズが大きく、それらのファイルにさまざまなパターンの小さなブロック単位でアクセスするデータベースの場合には、このようなアルゴリズムが最適でないことがあります。recordsize にデータベースのレコードサイズ以上の値を設定すると、パフォーマンスが大きく向上することがあります。このプロパティーを汎用目的のファイルシステムに使用することは、パフォーマンスが低下する可能性があるため、できるだけ避けてください。指定するサイズは、512 バイト - 128K バイトの 2 の累乗にしてください。ファイルシステムの recordsize 値を変更した場合、そのあとに作成されたファイルだけに適用されます。既存のファイルには適用されません。

このプロパティーの省略名は recsize です。

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。このプロパティーを off に設定すると、そのプロパティーを継承する共有が共有解除されます。share.smb プロパティーの使用例については、184 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、特に注意してください。

推奨される方法ではありませんが、`zfs create -V` に `-s` フラグを指定するか、またはボリュームの作成後に予約を変更すると、疎ボリュームを作成できます。「疎ボリューム」とは、予約がボリュームサイズと等しくないボリュームのことです。疎ボリュームの場合、`volsize` を変更しても予約には反映されません。

ボリュームの使用方法の詳細については、279 ページの「ZFS ボリューム」を参照してください。

ZFS ユーザープロパティー

ZFS は、ネイティブプロパティーに加えて、任意のユーザープロパティーもサポートします。ユーザープロパティーは ZFS の動作には影響しませんが、これらを使用すると、使用環境内で意味のある情報をデータセットに注釈として付けることができます。

ユーザープロパティーの名前は、次の規則に適合している必要があります。

- ネイティブプロパティーと区別するためのコロン(':') を含んでいる必要がある。
- 小文字の英字、数字、または次の句読文字を含んでいる必要がある。「:」、「+」、「.」、「_」。
- ユーザープロパティー名の最大長は、256 文字である。

想定されている規則では、プロパティー名は次の 2 つの部分に分割しますが、この名前空間は ZFS によって強制されているものではありません。

module:property

ユーザープロパティーをプログラムで使用する場合、プロパティー名の *module* 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、147 ページの「ZFS のプロパティーの紹介」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の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 コマンドの追加情報については、zfs(1M) を参照してください。

複雑な ZFS クエリーを作成する

-o、-t、および -H オプションを使用して、zfs list の出力をカスタマイズすることができます。

-o オプションと必要なプロパティーのコンマ区切りのリストを使用すれば、プロパティー値の出力をカスタマイズできます。任意のデータセットプロパティーを有効な引数として指定できます。サポートされているすべてのデータセットプロパティーのリストは、147 ページの「ZFS のプロパティーの紹介」を参照してください。また、定義されているプロパティー以外に、-o オプションのリストにリテラル name を指定すれば、出力にデータセットの名前が表示されるはずです。

次の例では、zfs list と一緒に 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 オプションを使用すれば、表示するデータセットの種類を指定できます。次の表は、有効な種類について説明しています。

表 5–2 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
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、表示するデータセットの種類をコンマ区切りのリストとして指定します。次の例では、-t オプションと - o オプションを同時に使用して、すべてのファイルシステムの名前と used プロパティーを表示しています。

```
# zfs list -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。次の例では、zfs list コマンドと -H オプションを使用して生成される出力を示しています。

```
# zfs list -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
```

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。

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、163 ページの「設定可能な ZFS ネイティブプロパティー」を参照してください。

zfs set コマンドには、*property=value* の形式のプロパティー/値のシーケンスを指定したあと、続けてデータセット名を指定します。各 zfs set 呼び出しで設定または変更できるプロパティーは、1 つだけです。

次の例では、tank/home の atime プロパティーを off に設定しています。

```
# zfs set atime=off tank/home
```

また、どのファイルシステムプロパティーもファイルシステムの作成時に設定できます。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
```

数値でないプロパティーの値では、大文字と小文字が区別されるので、小文字を使用する必要があります。ただし、mountpoint は例外です。このプロパティーの値には、大文字と小文字を混在させることができます。

zfs set コマンドの詳細については、zfs(1M) を参照してください。

ZFS プロパティーを継承する

割り当て制限と予約を除いて、すべての設定可能なプロパティーは、親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できます。その場合は、親ファイルシステムの値を継承することになります。

次の例では、zfs set コマンドを使用して tank/home/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 – -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 コマンドの詳細については、zfs(1M) を参照してください。

ZFS プロパティーのクエリー検索を行う

プロパティー値のクエリー検索を行うもっとも簡単な方法は、`zfs list` コマンドを使用することです。詳細については、171 ページの「基本的な ZFS 情報を表示する」を参照してください。ただし、複雑なクエリーを使用する場合およびスクリプトで使用する場合は、より詳細な情報をカスタマイズした書式で渡すために `zfs get` コマンドを使用します。

`zfs get` コマンドを使用して、任意のデータセットプロパティーを取得できます。次の例は、データセット上の 1 つのプロパティー値を取得する方法を示しています。

```
# zfs get checksum tank/ws
NAME            PROPERTY       VALUE                     SOURCE
tank/ws         checksum       on                        default
```

4 番目の列 `SOURCE` は、このプロパティー値の起点を示します。次の表は、表示される可能性のあるソース値を定義したものです。

表 5–3　出力される可能性のある SOURCE 値 (`zfs get` コマンド)

ソース値	説明
`default`	このプロパティー値は、このデータセットまたはその祖先 (存在する場合) で明示的に設定されたことが一度もありません。このプロパティーのデフォルト値が使用されています。
`inherited from` *dataset-name*	このプロパティー値は、*dataset-name* に指定されている親データセットから継承されます。
`local`	このプロパティー値は、`zfs set` を使って、このデータセットに明示的に設定されました。
`temporary`	このプロパティー値は、`zfs mount -o` オプションを使って設定され、マウントの有効期間だけ有効です。一時的なマウントプロパティーの詳細については、183 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             66.9G                  -
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              Fri May 11 10:58 2012  -
tank/home  dedup                 off                    default
tank/home  devices               on                     default
tank/home  encryption            off                    -
tank/home  exec                  on                     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            43K                    -
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.*               ...                    local
tank/home  snapdir               hidden                 default
tank/home  sync                  standard               default
tank/home  type                  filesystem             -
tank/home  used                  8.54M                  -
tank/home  usedbychildren        8.49M                  -
tank/home  usedbydataset         43K                    -
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` に `-s` オプションを使用すると、表示するプロパティーをソースの種類ごとに指定できます。このオプションには、必要なソースの種類をコンマ区切りのリストとして指定します。指定したソースの種類のプロパティーだけが表示されます。有効なソースの種類は、`local`、`default`、`inherited`、`temporary`、および `none` です。次の例では、`tank/ws` 上にローカルに設定されているすべてのプロパティーを表示しています。

```
# zfs get -s local all 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 コマンドの詳細については、zfs(1M) を参照してください。

スクリプトで使用できるように ZFS プロパティーのクエリー検索を行う

zfs get コマンドでは、スクリプトで使用できるように設計された -H および - o オプションを利用できます。-H オプションを使用すると、ヘッダー情報を省略し、空白をタブ文字で置き換えることができます。空白が揃うことで、データが見やすくなります。-o オプションを使用すると、次の方法で出力をカスタマイズできます。

- リテラル name は、147 ページの「ZFS のプロパティーの紹介」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。

- 179 ページの「ZFS マウントポイントを管理する」
- 181 ページの「ZFS ファイルシステムをマウントする」
- 183 ページの「一時的なマウントプロパティーを使用する」
- 183 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、53 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* にマウントされます。*path* はファイルシステムの名前です。

デフォルトのマウントポイントを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、165 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、181 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 プロパティーを変更すると、ファイルシステムが古いマウントポイントから自動的にアンマウントされて、新しいマウントポイントに再マウントされます。マウントポイントのディレクトリは必要に応じて作成されます。ファイルシステムがアクティブであるためにアンマウントできない場合は、エラーが報告され、手動で強制的にアンマウントする必要があります。

レガシーマウントポイント

mountpoint プロパティーを legacy に設定することで、ZFS ファイルシステムをレガシーツールを使って管理することができます。レガシーファイルシステムは、mount と umount コマンド、および /etc/vfstab ファイルを使用して管理する必要があります。レガシーファイルシステムは、ZFS が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。次に、/etc/vfstab ファイル内のエントリの例を示します。

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

tank/home/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 データの整合性の詳細については、28 ページの「トランザクションのセマンティクス」を参照してください。

ZFS ファイルシステムをマウントする

ZFS では、ファイルシステムが作成されるときまたはシステムがブートするときに、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、空でないディレクトリの最上位にマウントすることは許可されません。次に例を示します。

```
# 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(1M) to mount this filesystem
# mount -F zfs tank/home/billm
```

ファイルシステムがマウントされるとき、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 5–4 ZFS のマウント関連プロパティーとマウントオプション

プロパティー	マウントオプション
atime	atime/noatime
devices	devices/nodevices
exec	exec/noexec
nbmand	nbmand/nonbmand
readonly	ro/rw
setuid	setuid/nosetuid
xattr	xattr/noaxttr

マウントオプション nosuid は、nodevices,nosetuid の別名です。

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 コマンドの詳細については、zfs(1M) を参照してください。

ZFS ファイルシステムをアンマウントする

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 コマンドの引数として、マウントポイントまたはファイルシステム名のいずれかを指定できます。

次の例では、ファイルシステム名を使ってファイルシステムを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 コマンドの詳細については、zfs(1M) を参照してください。

ZFS ファイルシステムを共有および共有解除する

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。

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解除されます。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。

- NFS 共有を定義および公開するための以前のリリースの sharenfs プロパティーは share.nfs プロパティーに置き換えられました。
- SMB 共有を定義および公開するための以前のリリースの sharesmb プロパティーは share.smb プロパティーに置き換えられました。
- 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、次のような構文を使用します。

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

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
```

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
```

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 つの手順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 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については、share_nfs(1M) および share_smb(1M) 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の変更

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  on     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、Oracle Solaris の非大域ゾーン内で 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

このセクションでは移行に関する問題を説明します。

- 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 11 Express** からのアップグレード - 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 Express 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 でブートする予定がある場合は、ZFS データセットで共有構成を復元できるように、pkg update 操作の前に、最初に既存の共有構成のコピーを保存するようにしてください。

  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

共有動作に関する次のシナリオと考慮事項を確認してください。

- 共有プロパティーと .zfs/shares ファイルは、zfs clone および zfs send 操作での処理が異なります。.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。名前付き共有などの共有プロパティーは、スナップショットに含まれていません。zfs send および zfs receive 操作中のプロパティーの動作については、229 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。
- 特定のレガシー共有操作によって、自動的に自動共有が off になり、既存の自動共有が同等の名前付き共有に変換されます。ファイルシステムが予想どおりに共有されていない場合は、その share.auto 値が off に設定されているかどうかを確認してください。

- 名前付き共有を作成するリクエストが、その共有が自動共有と競合するために失敗する場合、処理を続けるためには自動共有を off にする必要があることがあります。
- プールが読み取り専用でインポートされると、そのプロパティーやそのファイルは変更できません。この状況では新しい共有を取り入れることもできない可能性があります。プールがエクスポートされる前に共有が確立されていた場合、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。
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。両方のプロパティーは、設定した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 Solaris リリースでは、特定のユーザーまたはグループが所有するファイルによって消費されるディスク領域の量に割り当て制限を設定することができます。ファイルシステムバージョン 4 より古いファイルシステム上のボリューム、またはバージョン 15 より古いプール上のボリュームには、ユーザーおよびグループの割り当て制限プロパティーを設定できません。

ファイルシステムを管理するために、割り当て制限と予約の機能としてどれがもっとも役立つかを判断するには、次の点を考慮してください。

- quota プロパティーと reservation プロパティーは、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。

割り当て制限と予約の設定方法の詳細については、196 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199 ページの「ZFS ファイルシステムに予約を設定する」を参照してください。

ZFS ファイルシステムに割り当て制限を設定する

ZFS ファイルシステムの割り当て制限は、zfs set および zfs get コマンドを使用して設定および表示できます。次の例では、10G バイトの割り当て制限が tank/home/jeff に設定されます。

```
# 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。

割り当て制限には、ファイルシステムが現在使用している容量より少ない容量を設定することはできません。例:

```
# zfs set quota=10K tank/home/jeff
cannot set property for 'tank/home/jeff':
size is less than current used or reserved space
```

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。この強い制限値には、子孫が消費するディスク容量は含まれません。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
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、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
```

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、割り当て制限を超えているのでこれ以上は書き込みが許可されないことが `EDQUOT` エラーメッセージによって通知される前にユーザーが自身の割り当て制限を超えている可能性があります。

従来の quota コマンドを使用して、NFS 環境 (例えば、ZFS ファイルシステムがマウントされているものなど) におけるユーザーの割り当て制限を確認することができます。ユーザーが割り当て制限を超えてる場合は、何もオプションを指定しなくても、quota コマンドだけで、出力情報が表示されます。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           -      -      -      -           -
```

ZFS ファイルシステムに予約を設定する

ZFS の「予約」とは、データセットが使用できることを保証された、プールから割り当てられたディスク領域のことです。つまり、プールで現在使用できないディスク容量をデータセットのディスク容量として予約することはできません。未処理の使用されていない予約の合計容量が、プールで消費されていないディスク容量を超えることはできません。ZFS の予約は、zfs set および zfs get コマンドを使用して設定および表示できます。次に例を示します。

```
# zfs set reservation=5G tank/home/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。
- ZFS 暗号化は子孫のファイルシステムに継承できます。鍵管理は、ZFS 委任管理を通じて委任できます。
- 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。

暗号化ポリシーは ZFS ファイルシステムの作成時に設定できますが、ポリシーの変更はできません。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、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。詳細は、例 5–4 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

Solaris 11 システムを Solaris 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、212 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 5–1 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 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 5–2 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 5–3 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 5–4 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

シャドウ移行機能を使用すると、次のようにファイルシステムを移行できます:

- ローカルまたはリモート ZFS ファイルシステムからターゲット ZFS ファイルシステムへ
- ローカルまたはリモート UFS ファイルシステムからターゲット ZFS ファイルシステムへ

シャドウマイグレーションは、移行するデータを取得するプロセスです。

- 空の ZFS ファイルシステムを作成します。
- ターゲット (またはシャドウ) ファイルシステムである空の ZFS ファイルシステム上で、移行するファイルシステムを示すように `shadow` プロパティーを設定します。
- 移行するファイルシステムからのデータは、シャドウファイルシステムにコピーされます。

shadow プロパティー URI を使用して、次の 2 つの方法で移行するファイルシステムを指定できます。

- `shadow=file:///`*path* – ローカルのファイルシステムを移行するにはこの構文を使用する
- `shadow=nfs://`*host/path* – NFS ファイルシステムを移行するにはこの構文を使用する

ファイルシステムを移行する場合は、次の考慮事項を確認してください。

- 移行するファイルシステムは読み取り専用に設定する必要があります。ファイルシステムが読み取り専用に設定されていない場合、進行中の変更が移行されない可能性があります。
- ターゲットファイルシステムは、完全に空である必要があります。
- 移行中にシステムをリブートした場合、システムがブートしたあと、移行は継続します。
- 完全に移行されていないディレクトリコンテンツへのアクセス、または完全に移行されていないファイルコンテンツへのアクセスは、コンテンツ全体が移行されるまでブロックされます。
- NFS での移行時に、UID、GID、および ACL 情報をシャドウファイルシステムに移行する場合は、ネームサービス情報がローカルおよびリモートシステムの間でアクセス可能であることを確認してください。NFS で大規模な移行をすべて行う前に、移行するファイルシステムデータの一部を移行テスト用にコピーして、すべての情報が適切に移行されるかどうかを確認してみることもできます。
- 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。しばらく待ってください。

- shadowstat コマンドを使用して、ファイルシステムの移行をモニターできます。次のデータが得られます。
  - BYTES XFRD 列には、シャドウファイルシステムに転送されたバイト数が示されます。
  - BYTES LEFT 列は、移行がほとんど完了するまで、継続的に増減します。ZFS は、移行の開始時に、移行に必要なデータ量を特定しません。このプロセスには非常に時間がかかるためです。
  - BYTES XFRD と ELAPSED TIME の情報を使用して移行プロセスの時間を見積もることを検討してください。

▼ 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する場合は、両方のシステム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、データが移行される予定のシステムにシャドウ移行パッケージをインストールし、**shadowd** サービスを有効にして移行プロセスに役立てます。

   ```
   # pkg install shadow-migration

   # svcadm enable shadowd
   ```

   shadowd プロセスを有効にしない場合、移行プロセスが完了したときに、shadow プロパティーを none にリセットする必要があります。

3. 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 readonly=on tank/home/data
   ```

   リモートのファイルシステムを移行している場合は、これを読み取り専用で共有します。たとえば、次のように指定します。

   ```
   # share -F nfs -o ro /export/home/ufsdata
   # share
   -               /export/home/ufsdata   ro   ""
   ```

4 シャドウプロパティーを移行するファイルシステムに設定して、新しい **ZFS** ファイルシステムを作成します。

たとえば、ローカルの ZFS ファイルシステム、rpool/old を新しい ZFS ファイルシステム、users/home/shadow に移行している場合、users/home/shadow ファイルシステムが作成されるときに、shadow プロパティーを rpool/old に設定します。

```
# zfs create -o shadow=file:///rpool/old users/home/shadow
```

たとえば、リモートサーバーから /export/home/ufsdata を移行するには、ZFS ファイルシステムが作成されるときに、shadow プロパティーを設定します。

```
# zfs create -o shadow=nfs://neo/export/home/ufsdata users/home/shadow2
```

5 移行の進捗を確認します。

例:

```
# shadowstat
                                  EST
                          BYTES   BYTES           ELAPSED
DATASET                   XFRD    LEFT    ERRORS  TIME
users/home/shadow         45.5M   2.75M   -       00:02:31
users/home/shadow         55.8M   -       -       00:02:41
users/home/shadow         69.7M   -       -       00:02:51
No migrations in progress
```

移行が完了したら、shadow プロパティーは none に設定されます。

```
# zfs get -r shadow users/home/shadow*
NAME                PROPERTY  VALUE   SOURCE
users/home/shadow   shadow    none    -
users/home/shadow2  shadow    none    -
```

ZFS ファイルシステムの移行のトラブルシューティング

ZFS の移行の問題をトラブルシューティングするときは、次の点を確認してください:

- 移行するファイルシステムが読み取り専用に設定されていない場合、一部のデータは移行されません。
- shadow プロパティーが設定されているときにターゲットファイルシステムが空でない場合、データの移行は開始しません。
- 移行の進行中に、移行するファイルシステムに対してデータを追加または削除した場合、これらの変更は移行されないことがあります。
- 移行の進行中に、シャドウファイルシステムのマウントを変更しようとした場合、次のメッセージが表示されます。

```
# zfs set mountpoint=/users/home/data users/home/shadow3
cannot unmount '/users/home/shadow3': Device busy
```

ZFS ファイルシステムをアップグレードする

以前の 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することができます。またこのコマンドは、ファイルシステムが古いバージョンを実行中である場合に通知します。

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

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、206 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 6 章

Oracle Solaris ZFS のスナップショットとクローンの操作

この章では、Oracle Solaris ZFS のスナップショットとクローンを作成して管理する方法について説明します。スナップショットの保存についての情報も示します。

この章は、次のセクションで構成されます。

ZFS スナップショットの概要

「スナップショット」とは、ファイルシステムまたはボリュームの読み取り専用コピーのことです。スナップショットはほとんど瞬間的に作成することができ、最初はプール内で追加のディスク領域を消費しません。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。

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、223 ページの「ZFS データを送信および受信する」を参照してください。

- 214 ページの「ZFS スナップショットを作成および破棄する」
- 217 ページの「ZFS スナップショットを表示してアクセスする」
- 219 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は次のように指定します。

filesystem@snapname
volume@snapname

スナップショット名は、32 ページの「ZFS コンポーネントに名前を付けるときの規則」の命名要件に従って付ける必要があります。

次の例では、`tank/home/cindy` のスナップショットが `friday` という名前で作成されます。

```
# zfs snapshot tank/home/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 tank/home@snap1
# zfs list -t snapshot -r tank/home
NAME                   USED  AVAIL  REFER  MOUNTPOINT
tank/home@snap1           0      -  2.11G  -
tank/home/cindy@snap1     0      -   115M  -
tank/home/lori@snap1      0      -  2.00G  -
tank/home/mark@snap1      0      -  2.00G  -
tank/home/tim@snap1       0      -  57.3M  -
```

スナップショットには、変更できるプロパティーはありません。また、データセットのプロパティーをスナップショットに適用することもできません。例:

```
# zfs set compression=on tank/home/cindy@friday
cannot set property for 'tank/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 tank/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、そのデータセットを破棄することはできません。例:

```
# zfs destroy tank/home/cindy
cannot destroy 'tank/home/cindy': filesystem has children
use '-r' to destroy the following datasets:
tank/home/cindy@tuesday
tank/home/cindy@wednesday
tank/home/cindy@thursday
```

また、スナップショットからクローンが作成されている場合は、スナップショットを破棄する前にクローンを破棄する必要があります。

`destroy` サブコマンドの詳細については、145 ページの「ZFS ファイルシステムを破棄する」を参照してください。

ZFS スナップショットの保持

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

スナップショットを「保持」すると、そのスナップショットは破棄されなくなります。また、この機能と `zfs destroy -d` コマンドを使用することにより、最後のクローンの消去を保留しながら、クローンが存在するスナップショットを削除できます。個々のスナップショットには、初期値が 0 のユーザー参照カウントが関連付けられます。このカウントは、スナップショットの保持を設定するたびに 1 増加し、保持を解除するたびに 1 減少します。

以前の Oracle Solaris リリースでは、スナップショットを破棄するには、スナップショットにクローンが存在しない状態で `zfs destroy` コマンドを使用する必要がありました。この 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、保持タグ `keep` を `tank/home/cindy/snap@1` に付与します。

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

`-r` オプションを使用すると、すべての子孫ファイルシステムのスナップショットを再帰的に保持できます。例:

```
# zfs snapshot -r tank/home@now
# zfs hold -r keep tank/home@now
```

この構文は、単一の参照 keep を特定のスナップショットまたはスナップショットの集合に追加します。個々のスナップショットには独自のタグ名前空間があり、その空間内で保持タグが一意である必要があります。スナップショットに保持が設定されている場合、保持されたそのスナップショットを zfs destroy コマンドを使って破棄しようとしても失敗します。例:

```
# zfs destroy tank/home/cindy@snap1
cannot destroy 'tank/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 tank/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 tank/home@now
NAME           TAG   TIMESTAMP
tank/home@now  keep  Fri Aug  3 15:15:53 2012

# zfs holds -r tank/home@now
NAME                 TAG   TIMESTAMP
tank/home/cindy@now  keep  Fri Aug  3 15:15:53 2012
tank/home/lori@now   keep  Fri Aug  3 15:15:53 2012
tank/home/mark@now   keep  Fri Aug  3 15:15:53 2012
tank/home/tim@now    keep  Fri Aug  3 15:15:53 2012
tank/home@now        keep  Fri Aug  3 15:15:53 2012
```

zfs release コマンドを使用すると、保持されたスナップショットまたはスナップショットの集合を解放することができます。例:

```
# zfs release -r keep tank/home@now
```

スナップショットが解放されたら、zfs destroy コマンドを使用してスナップショットを破棄できます。例:

```
# zfs destroy -r tank/home@now
```

スナップショットの保持情報を示す 2 つの新しいプロパティーがあります。

- zfs destroy -d コマンドを使ってスナップショットの遅延破棄が予約されている場合、defer_destroy プロパティーがオンになります。それ以外の場合、このプロパティーはオフです。
- userrefs プロパティーの値は、このスナップショットに設定されている保持の数に設定されます。この数のことをユーザー参照カウントとも呼びます。

ZFS スナップショットの名前を変更する

スナップショットの名前を変更することはできますが、名前を変更するときはそれらが作成された同じプールとデータセットの中で行う必要があります。次に例を示します。

```
# zfs rename tank/home/cindy@snap1 tank/home/cindy@today
```

また、次のショートカット構文は前の構文と同等です。

```
# zfs rename tank/home/cindy@snap1 today
```

次のようなスナップショットの名前変更操作はサポートされていません。ターゲットのプールとファイルシステムの名前が、スナップショットの作成されたプールとファイルシステムと異なるためです。

```
# zfs rename tank/home/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。あるいは、listsnapshots プールプロパティーを有効にします。例:

```
# zpool get listsnapshots tank
NAME  PROPERTY       VALUE      SOURCE
tank  listsnapshots  off        default
# zpool set listsnapshots=on tank
# zpool get listsnapshots tank
NAME  PROPERTY       VALUE      SOURCE
tank  listsnapshots  on         local
```

ファイルシステムのスナップショットには、ルートの .zfs/snapshot ディレクトリからアクセスできます。たとえば、tank/home/cindy が /home/cindy にマウントされている場合は、tank/home/cindy@thursday スナップショットのデータには、/home/cindy/.zfs/snapshot/thursday ディレクトリからアクセスできます。

```
# ls /tank/home/cindy/.zfs/snapshot
thursday   tuesday    wednesday
```

スナップショットの一覧は次の方法で表示できます。

```
# zfs list -t snapshot -r tank/home
NAME                        USED  AVAIL  REFER  MOUNTPOINT
tank/home/cindy@tuesday      45K      -  2.11G  -
tank/home/cindy@wednesday    45K      -  2.11G  -
tank/home/cindy@thursday       0      -  2.17G  -
```

特定のファイルシステムのために作成したスナップショットの一覧は、次の方法で表示できます。

```
# zfs list -r -t snapshot -o name,creation tank/home
NAME                       CREATION
tank/home/cindy@tuesday    Fri Aug  3 15:18 2012
tank/home/cindy@wednesday  Fri Aug  3 15:19 2012
tank/home/cindy@thursday   Fri Aug  3 15:19 2012
tank/home/lori@today       Fri Aug  3 15:24 2012
tank/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したときは、そのディスク領域は最初はスナップショットとファイルシステムの間で共有されます。それまでに作成したスナップショットと領域が共有されることもあります。ファイルシステムが変化していくにつれて、それまで共有されていたディスク領域がスナップショット固有になり、スナップショットの used プロパティーに計上されます。また、スナップショットを削除すると、ほかのスナップショットに固有の (および使用される) ディスク容量を増やすことができます。

スナップショット領域の referenced プロパティーの値は、スナップショットを作成したときのファイルシステムのプロパティーと同じです。

used プロパティーの値がどのように消費されているかについて、さらに詳細な情報を確認することができます。新しい読み取り専用ファイルシステムプロパティーは、クローン、ファイルシステム、およびボリュームに関するディスク領域使用状況を示します。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、表 5–1 を参照してください。

ZFS スナップショットにロールバックする

`zfs rollback` コマンドを使用すると、特定のスナップショットが作成された時点よりもあとにファイルシステムに対して行われたすべての変更を破棄できます。ファイルシステムは、そのスナップショットが作成されたときの状態に戻ります。デフォルトでは、このコマンドを使って、最新のスナップショット以外のスナップショットにロールバックすることはできません。

それより前のスナップショットにロールバックするには、中間にあるスナップショットをすべて破棄する必要があります。`-r` オプションを指定すれば、古いスナップショットを破棄できます。

中間にあるスナップショットのクローンが存在する場合は、`-R` オプションを指定してクローンも破棄する必要があります。

注 – ロールバックするファイルシステムが現在マウントされている場合は、アンマウントされてから再度マウントされます。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` オプションを指定すると、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、`tank/home/cindy` ファイルシステムが `tuesday` スナップショットにロールバックされます:

```
# zfs rollback tank/home/cindy@tuesday
cannot rollback to 'tank/home/cindy@tuesday': more recent snapshots exist
use '-r' to force deletion of the following snapshots:
tank/home/cindy@wednesday
tank/home/cindy@thursday
# zfs rollback -r tank/home/cindy@tuesday
```

この例では、スナップショット `wednesday` および `thursday` が破棄されます。これらよりも古いスナップショット `tuesday` にロールバックされるためです。

```
# zfs list -r -t snapshot -o name,creation tank/home/cindy
NAME                      CREATION
tank/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 /tank/home/tim
fileA
$ zfs snapshot tank/home/tim@snap1
$ ls /tank/home/tim
fileA  fileB
$ zfs snapshot tank/home/tim@snap2
```

たとえば、2 つのスナップショットの相違点を識別するには、次のような構文を使用します。

```
$ zfs diff tank/home/tim@snap1 tank/home/tim@snap2
M       /tank/home/tim/
+       /tank/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 M 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 /tank/cindy/fileB /tank/cindy/fileC
$ zfs snapshot tank/cindy@snap2
$ zfs diff tank/cindy@snap1 tank/cindy@snap2
M       /tank/cindy/
R       /tank/cindy/fileB -> /tank/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

詳細は、zfs(1M) を参照してください。

ZFS クローンの概要

「クローン」とは、書き込み可能なボリュームまたはファイルシステムのことで、最初の内容は作成元のデータセットと同じです。スナップショットの場合と同様に、クローンは瞬間的に作成され、最初は追加のディスク領域を消費しません。また、クローンのスナップショットを作成することもできます。

クローンは、スナップショットだけから作成できます。スナップショットが複製されるときに、クローンとスナップショットの間に暗黙の依存関係が作成されます。クローンは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。

クローンには、作成元のデータセットのプロパティーは継承されません。`zfs get` および `zfs set` コマンドを使用して、複製したデータセットのプロパティーを表示して変更することができます。ZFS データセットのプロパティーの設定方法の詳細については、174 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。

ZFS クローンを作成する

クローンを作成するには、`zfs clone` コマンドを使用します。クローンをどのスナップショットから作成するかを指定し、新しいファイルシステムまたはボリュームの名前を指定します。新しいファイルシステムまたはボリュームは、ZFS 階層内の任意の場所に配置できます。新しいデータセットは、クローンの作成元になったスナップショットと同じ種類 (ファイルシステムやボリュームなど) です。クローンを作成するためのファイルシステムは、基にするファイルシステムスナップショットがあるプールに存在している必要があります。

次の例では、`tank/home/matt/bug123` という名前の新しいクローンが作成されます。最初の内容は、スナップショット `tank/ws/gate@yesterday` と同じです。

```
# zfs snapshot tank/ws/gate@yesterday
# zfs clone tank/ws/gate@yesterday tank/home/matt/bug123
```

次の例では、スナップショット `projects/newproject@today` から複製されたワークスペースが、一時的なユーザーのために `projects/teamA/tempuser` という名前で作成されます。次に、複製されたワークスペースにプロパティーが設定されます。

```
# zfs snapshot projects/newproject@today
# zfs clone projects/newproject@today projects/teamA/tempuser
# zfs set 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。次に例を示します。

```
# zfs destroy tank/home/matt/bug123
```

親のスナップショットを破棄するときは、その前にクローンを破棄する必要があります。

ZFS ファイルシステムを ZFS クローンで置き換える

`zfs promote` コマンドを使えば、アクティブな ZFS ファイルシステムをそのファイルシステムのクローンで置き換えることができます。この機能を使ってファイルシステムの複製と置換を実行でき、「作成元」のファイルシステムが、指定されたファイルシステムのクローンになります。さらに、この機能を使えば、クローンの作成元となるファイルシステムを破棄することもできます。クローンの移行促進を行わない限り、アクティブクローンの元のファイルシステムを破棄することはできません。クローンの破棄に関する詳細については、222 ページの「ZFS クローンを破棄する」を参照してください。

次の例では、`tank/test/productA` ファイルシステムがクローンされたあと、クローンファイルシステム `tank/test/productAbeta` が元の `tank/test/productA` ファイルシステムになっています。

```
# zfs create tank/test
# zfs create tank/test/productA
# zfs snapshot tank/test/productA@today
# zfs clone tank/test/productA@today tank/test/productAbeta
# zfs list -r tank/test
NAME                        USED  AVAIL  REFER  MOUNTPOINT
tank/test                   104M  66.2G    23K  /tank/test
tank/test/productA          104M  66.2G   104M  /tank/test/productA
tank/test/productA@today       0      -   104M  -
tank/test/productAbeta         0  66.2G   104M  /tank/test/productAbeta
# zfs promote tank/test/productAbeta
# zfs list -r tank/test
NAME                           USED  AVAIL  REFER  MOUNTPOINT
tank/test                      104M  66.2G    24K  /tank/test
tank/test/productA                0  66.2G   104M  /tank/test/productA
tank/test/productAbeta         104M  66.2G   104M  /tank/test/productAbeta
tank/test/productAbeta@today      0      -   104M  -
```

この zfs list の出力では、元の productA ファイルシステムのディスク領域計上情報が、productAbeta ファイルシステムのものに置き換わっています。

ファイルシステムの名前を変更することで、クローンの置換処理を完了することができます。例:

```
# zfs rename tank/test/productA tank/test/productAlegacy
# zfs rename tank/test/productAbeta tank/test/productA
# zfs list -r tank/test
```

また、旧バージョンのファイルシステムを削除することもできます。例:

```
# zfs destroy tank/test/productAlegacy
```

ZFS データを送信および受信する

zfs send コマンドを実行すると、スナップショットのストリーム表現が作成され、標準出力に書き込まれます。デフォルトでは、完全なストリームが生成されます。この出力は、ファイルまたは別のシステムにリダイレクトできます。zfs receive コマンドを実行すると、ストリームに内容が指定されているスナップショットが作成され、標準入力に渡されます。ストリーム全体を受信する場合、新しいファイルシステムも作成されます。これらのコマンドを使えば、ZFS スナップショットデータを送信したり、ZFS スナップショットデータやファイルシステムを受信したりできます。次のセクションの例を参照してください。

- 224 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」
- 226 ページの「ZFS スナップショットを送信する」
- 228 ページの「ZFS スナップショットを受信する」
- 229 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」
- 231 ページの「複雑な ZFS スナップショットストリームを送信および受信する」
- 234 ページの「ZFS データのリモート複製」

ZFS データを保存するために、次のバックアップ方法が用意されています。

- 企業向けバックアップ製品 – 次の機能が必要な場合は、企業向けバックアップソリューションを検討してください。
  - ファイルごとの復元
  - バックアップメディアの検証
  - メディアの管理
- ファイルシステムのスナップショットとスナップショットのロールバック – ファイルシステムのコピーを作成して、必要に応じて以前のバージョンのファイルシステムに戻す作業を簡単に実行するには、zfs snapshot および zfs rollback コマンドを使用します。たとえば、以前のバージョンのファイルシステムからファイルを復元するために、この方法を使用できます。

スナップショットの作成およびロールバックの詳細については、213 ページの「ZFS スナップショットの概要」を参照してください。

- スナップショットの保存 – `zfs send` および `zfs receive` コマンドを使用して、ZFS スナップショットの送信と受信を行います。スナップショットから次のスナップショットまでの増分変更を保存することができますが、ファイルを個別に復元することはできません。ファイルシステムのスナップショット全体を復元する必要があります。これらのコマンドでは、ZFS データを保存するための完全なバックアップソリューションは提供されません。
- リモート複製 – あるシステムのファイルシステムを別のシステムにコピーするには、`zfs send` および `zfs receive` コマンドを使用します。この処理は、WAN 経由でデバイスをミラー化する従来のボリューム管理製品とは異なります。特殊な構成やハードウェアは必要ありません。ZFS ファイルシステムを複製する利点は、ファイルシステムを別のシステムのストレージプール上に再作成し、その新しく作成したプールに同じファイルシステムデータを格納しながら RAID-Z などの別の構成レベルを指定できることです。
- アーカイブユーティリティー – `tar`、`cpio`、`pax`、サードパーティーバックアップ製品などのアーカイブユーティリティーを使って ZFS データを保存します。現時点では、`tar` と `cpio` では NFSv4 方式の ACL を正しく変換できますが、`pax` では変換できません。

ほかのバックアップ製品を使用して ZFS データを保存する

`zfs send` および `zfs receive` コマンド以外に、`tar` や `cpio` コマンドなどのアーカイブユーティリティーを使用して、ZFS ファイルを保存することもできます。これらのユーティリティーは、ZFS ファイル属性と ACL を保存して復元します。`tar` コマンドと `cpio` コマンドの適切なオプションを確認してください。

ZFS とサードパーティー製バックアップ製品の問題に関する最新情報については、Oracle Solaris 11 リリースノートを参照してください。

ZFS スナップショットストリームを特定する

ZFS ファイルシステムまたはボリュームのスナップショットは、`zfs send` コマンドを使用してスナップショットストリームに変換されます。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の作成で使用された `zfs send` オプションに応じて、さまざま種類のストリーム形式が生成されます。

- 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、コマンド行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んだストリームタイプです。次の 3 つの種類のストリームパッケージが存在します。

- 複製ストリームパッケージ – 指定されたデータセットおよびその子孫から構成されます。すべての中間スナップショットを含みます。クローンが作成されたデータセットの複製元が、コマンド行で指定されたスナップショットの子孫ではない場合、この複製元のデータセットはストリームパッケージには含まれません。ストリームを受信するには、複製元のデータセットが送信先のストレージプールに存在している必要があります。

  次のデータセットとその複製元の一覧を考慮してください。これらのデータセットが、下に示された順番で作成されたとします。

```
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           -
```

  次の構文で作成された複製ストリームパッケージは、

```
# zfs send -R pool/b@send ....
```

  次の完全および増分ストリームから構成されます。

```
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
```

  前の出力で、pool/a/1@clone スナップショットは、複製ストリームパッケージに含まれていません。したがって、この複製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 – 指定されたデータセットおよびその子孫から構成されます。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、データセットの複製元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、動作は複製ストリームと同じようになります。ただし、下で説明する自己完結型の再帰的ストリームは、外的な依存関係がないようにして作成されます。

  次の構文で作成された再帰的ストリームパッケージは、

  ```
  # zfs send -r pool/b@send ...
  ```

  次の完全および増分ストリームから構成されます。

  ```
  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
  ```

  前の出力で、pool/a/1@clone スナップショットは、再帰的ストリームパッケージに含まれていません。したがって、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 - ストリームパッケージに含まれないデータセットに依存しません。この再帰的ストリームパッケージは次の構文で作成されます。

  ```
  # zfs send -rc pool/b@send ...
  ```

  次の完全および増分ストリームから構成されます。

  ```
  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
  ```

  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、別のプール上のスナップショットストリームを同じシステムに送信するには、次のような構文を使用します。

```
# zfs send tank/dana@snap1 | zfs recv spool/ds01
```

zfs receive コマンドの別名として、zfs recv を 使用できます。

スナップショットストリームを別のシステムに送信する場合は、zfs send の出力を ssh コマンドにパイプします。例:

```
sys1# zfs send tank/dana@snap1 | ssh sys2 zfs recv newtank/dana
```

完全なストリームを送信するときは、対象のファイルシステムが存在してはいけません。

zfs send -i オプションを使用すれば、増分データを送信できます。例:

```
sys1# zfs send -i tank/dana@snap1 tank/dana@snap2 | ssh sys2 zfs recv newtank/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 newtank/dana ファイルシステムがあらかじめ存在している必要があります。

注 – 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こ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 tank/dana@snap2 into newtank/dana:
most recent snapshot of tank/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。

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だけで指定できます。このショートカットは、*snap1* には @ 記号のあとの名前を指定するだけでよいことを意味し、この場合 snap1 は *snap2* と同じファイルシステムから作成されたものと見なされます。例:

```
sys1# zfs send -i snap1 tank/dana@snap2 | ssh sys2 zfs recv newtank/dana
```

このショートカット構文は、前の例の増分構文と同等です。

異なるファイルシステム snapshot1 から増分ストリームを生成しようとすると、次のメッセージが表示されます。

```
cannot send 'pool/fs@name': not an earlier snapshot from the same fs
```

多数のコピーを保管する必要がある場合は、gzip コマンドを使って ZFS スナップショットのストリーム表現を圧縮することを検討してください。例:

```
# zfs send pool/fs@snap | gzip > backupfile.gz
```

ZFS スナップショットを受信する

ファイルシステムのスナップショットを受信するときは、次の重要な点に留意してください。

- スナップショットとファイルシステムの両方が受信されます。
- ファイルシステムとその子孫のすべてのファイルシステムが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。
- 受信される元のファイルシステムは、その転送中に存在してはいけません。
- ファイルシステム名がすでに存在する場合は、`zfs rename` コマンドを使ってファイルシステムの名前を変更できます。

例:

```
# zfs send tank/gozer@0830 > /bkups/gozer.083006
# zfs receive tank/gozer2@today < /bkups/gozer.083006
# zfs rename tank/gozer tank/gozer.old
# zfs rename tank/gozer2 tank/gozer
```

対象のファイルシステムに変更を加え、新たに増分スナップショットを送信する場合は、まず受信側のファイルシステムをロールバックする必要があります。

次のような例を考えます。まず、次のようにファイルシステムに変更を加えます。

```
sys2# rm newtank/dana/file.1
```

次に、`tank/dana@snap3` の増分を送信します。ただし、新しい増分スナップショットを受信するには、まず受信側のファイルシステムをロールバックする必要があります。または、`-F` オプションを使用すれば、ロールバック手順を実行する必要がなくなります。例:

```
sys1# zfs send -i tank/dana@snap2 tank/dana@snap3 | ssh sys2 zfs recv -F newtank/dana
```

増分スナップショットを受信するときは、対象のファイルシステムが存在している必要があります。

ファイルシステムに変更を加えたあとで、新しい増分スナップショットを受信するために受信側のファイルシステムのロールバックを行わない場合、または `-F` オプションを使用しない場合は、次のようなメッセージが表示されます。

```
sys1# zfs send -i tank/dana@snap4 tank/dana@snap5 | ssh sys2 zfs recv newtank/dana
cannot receive: destination has been modified since most recent snapshot
```

`-F` オプションが正常に実行される前に、次の検査が行われます。

- 最新のスナップショットが増分ソースと一致しない場合は、ロールバックも受信も完了せず、エラーメッセージが返される。
- zfs receive コマンドで指定された増分ソースと一致しない異なるファイルシステムの名前を間違って指定した場合は、ロールバックも受信も完了せず、次のエラーメッセージが返される。

```
cannot send 'pool/fs@name': not an earlier snapshot from the same fs
```

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

ZFS スナップショットストリームを特定のファイルシステムプロパティー値で送信することができますが、スナップショットストリームを受信したときに異なるローカルプロパティー値を指定することができます。または、元のファイルシステムを再作成するために、スナップショットストリームを受信したときに元のプロパティー値を使用するように指定することもできます。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値 (存在する場合) に戻すには、zfs inherit -S を使用します。プロパティーに受信値が存在しない場合、zfs inherit - S コマンドの動作は、-S オプションを指定しない zfs inherit コマンドと同じです。プロパティーに受信値が存在する場合、zfs inherit コマンドは、zfs inherit -S コマンドの発行によって継承値が受信値に戻されるまでの間、受信値を継承値でマスクします。
- zfs get -o を使用すると、新しい非デフォルトの RECEIVED 列を含めることができます。または、zfs get -o all コマンドを使用すると、RECEIVED を含むすべての列を含めることができます。
- zfs send -p オプションを使用すると、-R オプションを使用せずにプロパティーを送信ストリームに含めることができます。
- zfs receive -e オプションを使用すると、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決定できます。次の例では、poola/bee/cee@1 スナップショットを poold/eee ファイルシステムに送信し、スナップショット名の最後の要素 (cee@1) のみを使用して、受信側のファイルシステムおよびスナップショットを作成します。

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

```
poold              134K   134G    23K  /poold
poold/eee           44K   134G    23K  /poold/eee
poold/eee/cee       21K   134G    21K  /poold/eee/cee
poold/eee/cee@1      0      -    21K  -
```

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、tank/data というファイルシステムの compression プロパティーが無効になっているとします。tank/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 tank/data
NAME       PROPERTY     VALUE     SOURCE
tank/data  compression  off       default
# zfs snapshot tank/data@snap1
# zfs send -p tank/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 コマンドを使用します。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、これは元の tank/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。

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って、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 コマンドを使用します。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 tank/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 tank/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、zfs send -I および -R オプションを使用して、より複雑なスナップショットストリームを送受信する方法について説明します。

複雑な ZFS スナップショットストリームを送受信するときは、次の点に留意してください。

- 1 つのスナップショットのすべての増分ストリームを累積スナップショットに送信する場合は、zfs send -I オプションを使用します。または、元のスナップショットからの増分ストリームを送信してクローンを作成する場合にも、このオプションを使用します。増分ストリームを受け入れるには、元のスナップショットが受信側にすでに存在している必要があります。
- すべての子孫ファイルシステムの複製ストリームを送信する場合は、zfs send -R オプションを使用します。複製ストリームの受信時には、すべてのプロパティー、スナップショット、下位ファイルシステム、およびクローンが維持されます。
- -c オプションを付けずに zfs send -r オプションを使用した場合や、zfs send -R オプションを使用した場合、ストリームパッケージは一部の状況でクローンの origin を省略します。詳細については、224 ページの「ZFS スナップショットストリームを特定する」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。

- 複製ストリームの受信時に zfs recv -F が指定されていない場合、データセットの destroy 操作は無視されます。この場合の zfs recv -F 構文は、「必要に応じてロールバックする」という意味も持っています。
- (zfs send -R ではない) ほかの -i または -I の場合と同様に、-I を使用すると、snapA から snapD までのすべてのスナップショットが送信されます。-i を使用すると、(すべての子孫の) snapD だけが送信されます。

- このような新しい種類の zfs send ストリームを受信するには、そのストリームを送信できるソフトウェアバージョンが受信側のシステムで稼働している必要があります。ストリームのバージョンは 1 増やされています。

  ただし、新しいソフトウェアバージョンを使用して古いプールバージョンのストリームにアクセスすることはできます。たとえば、新しいオプションで作成されたストリームを、バージョン 3 プールに対して送受信することができます。ただし、新しいオプションで送信されたストリームを受信するには、最近のソフトウェアが稼働している必要があります。

例 6–1 複雑な ZFS スナップショットストリームを送信および受信する

zfs send -I オプションを使用すると、一連の増分スナップショットを結合して 1 つのスナップショットを作成できます。例:

```
# zfs send -I pool/fs@snapA pool/fs@snapD > /snaps/fs@all-I
```

次に、snapB、snapC、および snapD を削除します。

```
# zfs destroy pool/fs@snapB
# zfs destroy pool/fs@snapC
# zfs destroy pool/fs@snapD
```

結合されたスナップショットを受信するには、次のコマンドを使用します。

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

zfs send -I コマンドを使用すると、スナップショットとクローンスナップショットを結合して、結合されたデータセットを作成することもできます。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
```

例 6–1 複雑な ZFS スナップショットストリームを送信および受信する (続き)

```
# zfs destroy pool/clone
# zfs receive -F pool/clone < /snaps/fsclonesnap-I
```

zfs send -R コマンドを使用すると、ZFS ファイルシステムおよび指定されたスナップショットまでのすべての子孫ファイルシステムを複製できます。このストリームの受信時には、すべてのプロパティー、スナップショット、子孫ファイルシステム、およびクローンが維持されます。

次の例では、ユーザーのファイルシステムのスナップショットが作成されます。すべてのユーザースナップショットから 1 つの複製ストリームが作成されます。次に、元のファイルシステムおよびスナップショットが破棄されてから回復されます。

```
# zfs snapshot -r users@today
# zfs list
NAME                USED  AVAIL  REFER  MOUNTPOINT
users               187K  33.2G    22K  /users
users@today            0      -    22K  -
users/user1          18K  33.2G    18K  /users/user1
users/user1@today      0      -    18K  -
users/user2          18K  33.2G    18K  /users/user2
users/user2@today      0      -    18K  -
users/user3          18K  33.2G    18K  /users/user3
users/user3@today      0      -    18K  -
# zfs send -R users@today > /snaps/users-R
# zfs destroy -r users
# zfs receive -F -d users < /snaps/users-R
# zfs list
NAME                USED  AVAIL  REFER  MOUNTPOINT
users               196K  33.2G    22K  /users
users@today            0      -    22K  -
users/user1          18K  33.2G    18K  /users/user1
users/user1@today      0      -    18K  -
users/user2          18K  33.2G    18K  /users/user2
users/user2@today      0      -    18K  -
users/user3          18K  33.2G    18K  /users/user3
users/user3@today      0      -    18K  -
```

次の例では、zfs send -R コマンドを使用して、users ファイルシステムとその子孫を複製し、複製したストリームを別のプール users2 に送信します。

```
# zfs create users2 mirror c0t1d0 c1t1d0
# zfs receive -F -d users2 < /snaps/users-R
# zfs list
NAME                 USED  AVAIL  REFER  MOUNTPOINT
users                224K  33.2G    22K  /users
users@today             0      -    22K  -
users/user1           33K  33.2G    18K  /users/user1
users/user1@today     15K      -    18K  -
users/user2           18K  33.2G    18K  /users/user2
users/user2@today       0      -    18K  -
users/user3           18K  33.2G    18K  /users/user3
```

例 6–1 複雑な ZFS スナップショットストリームを送信および受信する (続き)

```
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 tank/cindy@today | ssh newsys zfs recv sandbox/restfs@today
```

このコマンドは、tank/cindy@today スナップショットのデータを送信し、そのデータを sandbox/restfs ファイルシステムに受信します。このコマンドは、restfs@today スナップショットを newsys システム上にも作成します。この例のユーザーは、リモートシステム上で ssh を使用するように構成されています。

◆ ◆ ◆ 第 7 章

7 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。

この章は、次のセクションで構成されます。

Solaris ACL モデル

以前のバージョンの Solaris では、主に POSIX ドラフト ACL 仕様に基づく ACL 実装がサポートされていました。POSIX ドラフトに基づく ACL は、UFS ファイルを保護するために使用され、NFSv4 より前のバージョンの NFS によって変換されます。

NFSv4 を導入したことにより、NFSv4 が提供する UNIX クライアントと UNIX 以外のクライアントとの間の相互運用性を、新しい ACL モデルを使って完全にサポートできるようになりました。NFSv4 仕様で定義されている新しい ACL 実装は、NT 方式の ACL を使用して、より豊かなセマンティクスを実現します。

新しい ACL モデルは、主に次の点が異なっています。

- NFSv4 仕様に基づいており、NT 方式の ACL に似ています。
- アクセス特権をより詳細に設定できます。詳細は、表 7–2 を参照してください。
- `setfacl` や `getfacl` コマンドではなく、`chmod` および `ls` コマンドを使用して設定および表示します。
- ディレクトリのアクセス特権をどのようにサブディレクトリに適用するかを指定するために、より多くの継承セマンティクスを利用できます。詳細については、241 ページの「ACL 継承」を参照してください。

どちらの ACL モデルを使った場合でも、標準のファイルアクセス権の場合より詳細にアクセス権を制御できます。新しい ACL は、POSIX ドラフト ACL と同様に、複数のアクセス制御エントリ (ACE) で構成されています。

POSIX ドラフト方式の ACL では、1 つのエントリを使用して、許可するアクセス権と拒否するアクセス権を定義します。新しい ACL モデルでは、アクセス権を検査するために、2 種類の ACE が利用されます。`ALLOW` と `DENY` です。つまり、どちらの ACE にもアクセス権が定義されていますが、その ACE に定義されていないアクセス権については、その ACE だけを使ってアクセス権を許可または拒否するかを推論することはできません。

NFSv4 方式の ACL と POSIX ドラフト ACLとの 間の変換は、次のように行われます。

- ACL に対応するユーティリティー (`cp`、`mv`、`tar`、`cpio`、`rcp` コマンドなど) のいずれかを使用している場合は、ACL が含まれる UFS ファイルを ZFS ファイルシステムに転送するときに、POSIX ドラフト ACL が同等の NFSv4 方式の ACL に変換されます。
- 一部の NFSv4 方式の ACL は、POSIX ドラフト ACL に変換されます。NFSv4 方式の ACL が POSIX ドラフト ACL に変換されない場合は、次のようなメッセージが表示されます。

  ```
  # cp -p filea /var/tmp
  cp: failed to set acl entries on /var/tmp/filea
  ```
- 最新の Solaris リリースを実行するシステム上で ACL の保持オプション (`tar -p` または `cpio -P`) を使って UFS `tar` または `cpio` アーカイブを作成した場合でも、以前の Solaris リリースを実行するシステム上でそのアーカイブを展開したときは、ACL が失われます。

  すべてのファイルが正しいファイルモードで展開されますが、ACL エントリは無視されます。
- `ufsrestore` コマンドを使って ZFS ファイルシステムにデータを復元することができます。元のデータに POSIX 方式の ACL が含まれている場合、それらは NFSv4 方式の ACL に変換されます。
- UFS ファイルに NFSv4 方式の ACL を設定しようとすると、次のようなメッセージが表示されます。

  ```
  chmod: ERROR: ACL type's are different
  ```
- ZFS ファイルに POSIX 方式の ACL を設定しようとすると、次のようなメッセージが表示されます。

  ```
  # getfacl filea
  File system doesn't support aclent_t style ACL's.
  See acl(5) for more information on Solaris ACL support.
  ```

ACL およびバックアップ製品に関するその他の制限については、224 ページの「ほかのバックアップ製品を使用して ZFS データを保存する」を参照してください。

ACL を設定する構文の説明

基本的な ACL の形式として、次の 2 つの形式が用意されています。

- 簡易 **ACL** – 従来の UNIX `user`、`group`、および `owner` エントリのみが含まれます。
- 非簡易 **ACL** – 所有者、グループ、および全員だけでなくそれ以外のエントリが含まれるか、継承フラグセットが含まれるか、またはエントリが従来とは異なる方法で順序付けられます。

簡易 **ACL** を設定する構文

```
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** を設定する構文

```
chmod [options] A[index]{+|=}user|group:name:access-permissions
/...[:inheritance-flags]:deny | allow file

chmod [options] A-user|group:name:access-permissions /...[:inheritance-flags]:deny |
allow file ...

chmod [options] A[index]- file
```

owner@, group@, everyone@
: 簡易 ACL 構文の *ACL-entry-type* を指定します。*ACL-entry-types* については、表 7–1 を参照してください。

user または group:*ACL-entry-ID=username* または *groupname*
: 明示的な ACL 構文の *ACL-entry-type* を指定します。ユーザーとグループの *ACL-entry-type* には、*ACL-entry-ID* と、*username* または *groupname* も含める必要があります。*ACL-entry-types* については、表 7–1 を参照してください。

access-permissions/.../
: 許可または拒否するアクセス権を指定します。ACL アクセス特権については、表 7–2 を参照してください。

inheritance-flags
: ACL 継承フラグのオプションリストを指定します。ACL 継承フラグについては、表 7–4 を参照してください。

`deny` | `allow`
: そのアクセス権を許可するかまたは拒否するかを指定します。

次の例では、owner@、group@、または everyone@ についての *ACL-entry-ID* の値は存在しません。

```
group@:write_data/append_data/execute:deny
```

次の例では、*ACL-entry-ID* が含まれています。特定のユーザー (*ACL-entry-type*) を ACL に含めるためです。

```
0:user:gozer:list_directory/read_data/execute:allow
```

ACL エントリが表示されるときは、次のようになります。

```
2:group@:write_data/append_data/execute:deny
```

この例では **2** つまり *index-ID* が指定されていますが、これは、より大きな ACL の ACL エントリであることを示しています。所有者、特定の UID、グループ、および全員のための複数のエントリで構成される可能性があります。`chmod` コマンドと一緒に *index-ID* を指定すれば、ACL のどの部分を変更するかを指定できます。たとえば次のように、`chmod` コマンドに `A3` と指定して、インデックス ID 3 を特定することができます。

```
chmod A3=user:venkman:read_acl:allow filename
```

ACL エントリタイプは、所有者やグループなどの ACL 表現です。次の表の説明を参照してください。

表 7–1　ACL エントリタイプ

ACL エントリタイプ	説明
`owner@`	オブジェクトの所有者に許可するアクセス権を指定します。
`group@`	オブジェクトを所有するグループに許可するアクセス権を指定します。
`everyone@`	ほかのどの ACL エントリにも一致しないすべてのユーザーまたはグループに許可するアクセス権を指定します。
`user`	ユーザー名を使って、オブジェクトに追加するユーザーに許可するアクセス権を指定します。*ACL-entry-ID* を含める必要があります。*username* または *userID* を指定します。値が有効な数値 UID または *username* でない場合、その ACL エントリタイプは無効です。
`group`	グループ名を使って、オブジェクトに追加するグループに許可するアクセス権を指定します。*ACL-entry-ID* を含める必要があります。*groupname* または *groupID* を指定します。値が有効な数値 GID または *groupname* でない場合、その ACL エントリタイプは無効です。

ACL アクセス特権について、次の表で説明します。

表 7–2 ACL アクセス特権

アクセス特権	アクセス特権のコンパクト表現	説明
add_file	w	ディレクトリに新しいファイルを追加するためのアクセス権。
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	現時点では実装されていません。
delete	d	ファイルを削除するためのアクセス権。delete アクセス権の特定の動作の詳細については、表 7–3 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 7–3 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) を読み取るためのアクセス権。基本属性は、stat レベルの属性と考えてください。このアクセスマスクビットを許可したエンティティーは、ls(1) および stat(2) を実行できる状態になります。
read_data	r	ファイルの内容を読み取るためのアクセス権。
read_xattr	R	ファイルの拡張属性を読み取るためのアクセス権。または、ファイルの拡張属性ディレクトリの検索を実行するためのアクセス権。
synchronize	s	現時点では実装されていません。
write_xattr	W	拡張属性を作成するためのアクセス権。または、拡張属性ディレクトリに書き込み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。

表 7–2 ACL アクセス特権 (続き)

アクセス特権	アクセス特権のコンパクト表現	説明
write_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 7–3 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		
	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

次の ACL の組み合わせを、個々のアクセス権を個別に設定するのではなく *ACL* セットで適用できます。次のような ACL セットが利用可能です。

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

ACL セット名	含まれる ACL アクセス権
write_set	write_data、append_data、write_attributes 、および write_xattr

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 継承

ACL 継承を使用する目的は、親ディレクトリの既存のアクセス権ビットを考慮しながら、意図した ACL を新しく作成するファイルまたはディレクトリが継承できるようにすることです。

デフォルトでは、ACL は伝達されません。ディレクトリに非簡易 ACL を設定した場合でも、その ACL はそれ以降に作成されるディレクトリには継承されません。ACL を継承する場合は、ファイルまたはディレクトリにそのことを指定する必要があります。

オプションの継承フラグについて、次の表で説明します。

表 7–4 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 を継承しますが、新しく作成したファイルまたはサブディレクトリにのみ適用され、そのディレクトリ自体には適用されません。このフラグを使用する場合は、何を継承するかを指定するために、file_inherit フラグまたは dir_inherit フラグ、あるいはその両方を指定する必要があります。
no_propagate	n	親ディレクトリの ACL をそのディレクトリの第 1 レベルの内容にのみ継承します。第 2 レベル以降の内容には継承しません。このフラグを使用する場合は、何を継承するかを指定するために、file_inherit フラグまたは dir_inherit フラグ、あるいはその両方を指定する必要があります。
-	なし	アクセス権は付与されていません。

現在、次のフラグは、SMB クライアントまたはサーバーにのみ適用できます。

表 7–4 ACL 継承フラグ (続き)

継承フラグ	継承フラグのコンパクト表現	説明
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。詳細については、次のセクション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、ACL と chmod 操作との関連を判定する、次のプロパティーが含まれています。

- aclinherit – ACL 継承の動作を判定します。値には次のものが含まれます。
  - 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、作成されるファイルのモードは継承可能な ACE によって決定されます。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。
- 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 – aclmode プロパティーが discard であるファイルシステムでは、ファイルのモードを表さない ACL エントリがすべて削除されます。これがデフォルト値です。
- mask – aclmode プロパティーが mask であるファイルシステムでは、ユーザーまたはグループアクセス権が削減されます。アクセス権は、グループアクセス権ビットと同程度にまで低下します。ただし、アクセス権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きます。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 – aclmode プロパティーが passthrough であるファイルシステムでは、ファイルまたはディレクトリの新規モードを表す必須の ACL エントリを生成する以外、ACL に変更は加えられません。

aclmode のデフォルトモードは、discard です。

aclmode プロパティーの使用方法については、例 7–14 を参照してください。

ZFS ファイルに ACL を設定する

ZFS と一緒に実装される ACL は、ACL エントリの配列で構成されます。ZFS の ACL モデルは「純粋」です。つまり、すべてのファイルに ACL が含まれます。この ACL は、従来の UNIX の owner/group/other エントリを表現しているだけという点で、全体的に見れば簡易な ACL です。

ZFS ファイルにもアクセス権ビットとモードが含まれます。ただし、より正確に表現すれば、これらの値は ACL が表現するアクセス権のキャッシュです。つまり、ファイルのアクセス権を変更した場合には、それに応じてファイルの ACL が更新されます。また、ファイルまたはディレクトリへのアクセスをユーザーに許可するための非簡易 ACL を削除しても、グループまたは全員にアクセスを許可するファイルまたはディレクトリのアクセス権ビットが設定されている場合には、そのユーザーはそのファイルまたはディレクトリに引き続きアクセスできます。アクセス制御に関するすべての決定は、ファイルまたはディレクトリの ACL に表現されているアクセス権によって制御されます。

ZFS ファイルの ACL アクセス権に関する主な規則は、次のとおりです。

- ZFS では、ACL に指定されている順序に従って、上から順番に ACL エントリが処理されます。
- ACL エントリが処理されるのは、アクセスを要求したユーザーが ACL エントリに設定されているユーザーと一致した場合だけです。
- いったん付与した許可アクセス権は、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許可アクセス権が失われている場合でも、ファイルの所有者またはスーパーユーザーに許可されるアクセス要求は、特権サブシステムによって決められます。この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。

ディレクトリに非簡易 ACL を設定しても、その ACL はそのディレクトリの子に自動的に継承されることはありません。非簡易 ACL を設定し、それがそのディレクトリの子に継承されるようにする場合は、ACL 継承フラグを使用する必要があります。詳細については、表 7–4 および 251 ページの「ZFS ファイルの ACL 継承を冗長形式で設定する」を参照してください。

新しいファイルを作成すると、umask の値に応じて、次のようなデフォルトの簡易 ACL が適用されます。

```
$ ls -v file.1
-rw-r--r--   1 root     root      206663 Jun 23 15:06 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/read_xattr/read_attributes/read_acl/synchronize:allow
     2:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
         :allow
```

この例では、ユーザーカテゴリ (owner@、group@、everyone@) ごとに ACL エントリが存在します。

このファイル ACL について説明します。

0:owner@	この所有者は、このファイルの内容を読み取って変更することができます (read_data/write_data/append_data/read_xattr)。この所有者は、タイムスタンプ、拡張属性、ACL などのファイル属性を変更することもできます (write_xattr/read_attributes/write_attributes/read_acl/write_acl)。さらに、この所有者はファイルの所有権を変更できます (write_owner:allow)。 synchronize の許可アクセス権は、現在のところ実装されていません。
1:group@	グループには、ファイルおよびファイルの属性への読み取りアクセス権が付与されます (read_data/read_xattr/read_attributes/read_acl:allow)。
2:everyone@	ユーザーまたはグループ以外の全員には、ファイルおよびファイルの属性を読み取るアクセス権が付与されます

(read_data/read_xattr/read_attributes/read_acl/
synchronize:allow)。synchronize の許可アクセス権は、現在のところ実装されていません。

新しいディレクトリを作成すると、umask の値に応じて、デフォルトのディレクトリ ACL は次のようになります。

```
$ ls -dv dir.1
drwxr-xr-x   2 root     root           2 Jul 20 13:44 dir.1
     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
```

このディレクトリ ACL について説明します。

0:owner@ この所有者は、ディレクトリの内容を読み取って変更すること (list_directory/read_data/add_file/write_data/add_subdirectory /append_data)、およびタイムスタンプ、拡張属性、ACL などのファイル属性を読み取って変更すること (/read_xattr/write_xattr/read_attributes/write_attributes/read_acl/ write_acl) ができます。さらに、所有者は、内容を検索すること (execute)、ファイルまたはディレクトリを削除すること (delete_child)、およびディレクトリの所有権を変更すること (write_owner:allow) ができます。

synchronize の許可アクセス権は、現在のところ実装されていません。

1:group@ グループは、ディレクトリ内容とディレクトリの属性を表示して読み取ることができます。またこのグループは、ディレクトリ内容を検索する実行権を持ちます (list_directory/read_data/read_xattr/execute/read_attributes /read_acl/synchronize:allow)。

2:everyone@ ユーザーまたはグループ以外の全員に、ディレクトリの内容および属性を読み取って実行するアクセス権が付与されます (list_directory/read_data/read_xattr/execute/read_ attributes/read_acl/synchronize:allow)。synchronize の許可アクセス権は、現在のところ実装されていません。

ZFS ファイルの ACL を冗長形式で設定および表示する

chmod コマンドを使用して、ZFS ファイルの ACL を変更できます。次の chmod 構文では、ACL を変更するために *acl-specification* を使って ACL の形式を指定しています。*acl-specification* については、237 ページの「ACL を設定する構文の説明」を参照してください。

- ACL エントリを追加する
  - ユーザーの ACL エントリを追加する

    ```
    % chmod A+acl-specification filename
    ```
  - *index-ID* を使用して ACL エントリを追加する

    ```
    % chmod Aindex-ID+acl-specification filename
    ```

    この構文では、指定した *index-ID* の位置に新しい ACL エントリが挿入されます。
- ACL エントリを置き換える

  ```
  % chmod A=acl-specification filename
  ```

  ```
  % chmod Aindex-ID=acl-specification filename
  ```
- ACL エントリを削除する
  - *index-ID* を使用して ACL エントリを削除する

    ```
    % chmod Aindex-ID- filename
    ```
  - ユーザーを使用して ACL エントリを削除する

    ```
    % chmod A-acl-specification filename
    ```
  - 非簡易 ACL をファイルからすべて削除する

    ```
    % chmod A- filename
    ```

ls - v コマンドを使用することで、詳細な ACL 情報が表示されます。例:

```
# ls -v file.1
-rw-r--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/read_xattr/read_attributes/read_acl/synchronize:allow
     2:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
         :allow
```

コンパクト形式の ACL の使用方法については、256 ページの「ZFS ファイルの ACL をコンパクト形式で設定および表示する」を参照してください。

例 7–1 ZFS ファイルの簡易 ACL を変更する

このセクションでは、簡易 ACL を設定して表示する例を示します。これは、従来の UNIX エントリ (ユーザー、グループ、およびその他) のみが ACL に含まれていることを意味します。

次の例では、簡易 ACL が file.1 にあります。

```
# ls -v file.1
-rw-r--r--   1 root     root      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/read_xattr/read_attributes/read_acl/synchronize:allow
     2:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
         :allow
```

次の例では、write_data アクセス権が group@ に付与されます。

```
# chmod 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、permissions on file.1 へのアクセス権の設定が 644 に戻されます。

```
# chmod 644 file.1
# ls -v file.1
-rw-r--r--   1 root     root      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/read_xattr/read_attributes/read_acl/synchronize:allow
     2:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
         :allow
```

例 7–2 ZFS ファイルに非簡易 ACL を設定する

このセクションでは、非簡易 ACL を設定して表示する例を紹介します。

次の例では、read_data/execute アクセス権が、test.dir ディレクトリのユーザー gozer に追加されます。

```
# chmod A+user:gozer:read_data/execute:allow test.dir
# ls -dv test.dir
drwxr-xr-x+  2 root     root           2 Jul 20 14:23 test.dir
     0:user:gozer:list_directory/read_data/execute:allow
     1:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
```

例 7–2 ZFS ファイルに非簡易 ACL を設定する (続き)

```
         /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、read_data/execute アクセス権がユーザー gozer から削除されます。

```
# chmod A0- test.dir
# ls -dv test.dir
drwxr-xr-x   2 root     root           2 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
```

例 7–3 ACL を使用して ZFS ファイルのアクセス権を操作する

次の ACL の例では、ACL を設定してから、ファイルまたはディレクトリのアクセス権ビットを変更するまでの操作を説明します。

次の例では、簡易 ACL が file.2 にあります。

```
# ls -v file.2
-rw-r--r--   1 root     root        2693 Jul 20 14:26 file.2
     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
```

次の例では、ACL allow アクセス権が everyone@ から削除されます。

```
# chmod A2- file.2
# ls -v file.2
-rw-r-----   1 root     root        2693 Jul 20 14:26 file.2
     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
```

この出力では、ファイルのアクセス権ビットが 644 から 640 に再設定されています。everyone@ の読み取りアクセス権は、everyone@ の ACL 許可アクセス権が削除されるときに、ファイルのアクセス権ビットから事実上削除されています。

例 7–3 ACL を使用して ZFS ファイルのアクセス権を操作する (続き)

次の例では、既存の ACL が everyone@ の read_data/write_data アクセス権に置き換わります。

```
# chmod A=everyone@:read_data/write_data:allow file.3
# ls -v file.3
-rw-rw-rw-   1 root     root        2440 Jul 20 14:28 file.3
     0:everyone@:read_data/write_data:allow
```

この出力では、chmod 構文を使って、owner@、group@、および everyone@ が読み取りまたは書き込みできるように、既存の ACL を read_data/write_data:allow アクセス権に事実上置き換えています。このモデルでは、everyone@ を使って、すべてのユーザーまたはグループへのアクセス権を指定しています。所有者とグループのアクセス権をオーバーライドする owner@ と group@ の ACL エントリがないので、アクセス権ビットは 666 に設定されます。

次の例では、既存の ACL がユーザー gozer の読み取りアクセス権に置き換わります。

```
# chmod A=user:gozer:read_data:allow file.3
# ls -v file.3
----------+  1 root     root        2440 Jul 20 14:28 file.3
     0:user:gozer:read_data:allow
```

この出力では、従来のファイルアクセス権コンポーネントを表す owner@、group@、または everyone@ の ACL エントリがないので、ファイルアクセス権は 000 になります。ファイルの所有者は、次のようにアクセス権 (および ACL) を再設定することで、この問題を解決できます。

```
# chmod 655 file.3
# ls -v file.3
-rw-r-xr-x   1 root     root        2440 Jul 20 14:28 file.3
     0:owner@:execute: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/execute/read_attributes/read_acl
         /synchronize:allow
     3:everyone@:read_data/read_xattr/execute/read_attributes/read_acl
         /synchronize:allow
```

例 7–4 ZFS ファイルの簡易 ACL を復元する

chmod コマンドを使用して、ファイルまたはディレクトリの非簡易 ACL をすべて削除できます。

次の例では、2 つの非簡易 ACL が test5.dir にあります。

例 7–4 ZFS ファイルの簡易 ACL を復元する (続き)

```
# ls -dv test5.dir
drwxr-xr-x+  2 root     root           2 Jul 20 14:32 test5.dir
     0:user:lp:read_data:file_inherit:deny
     1:user:gozer:read_data:file_inherit:deny
     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
```

次の例では、ユーザー gozer と lp の非簡易 ACL が削除されます。残りの ACL には、owner@、group@、および everyone@ のデフォルト値が含まれています。

```
# chmod A- test5.dir
# ls -dv test5.dir
drwxr-xr-x   2 root     root           2 Jul 20 14:32 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/execute/read_attributes
         /read_acl/synchronize:allow
     2:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
         /read_acl/synchronize:allow
```

例 7–5 ACL セットを ZFS ファイルに適用する

ACL セットを利用すれば、ACL アクセス権を個別に適用する必要がなくなります。セットの説明については、240 ページの「ZFS ACL セット」を参照してください。

たとえば、次のように read_set を適用できます。

```
# chmod A+user:otto:read_set:allow file.1
# ls -v file.1
-r--r--r--+  1 root     root      206695 Jul 20 13:43 file.1
     0:user:otto:read_data/read_xattr/read_attributes/read_acl:allow
     1:owner@:read_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
```

write_set と read_set は次のように適用できます。

```
# chmod A+user:otto:read_set/write_set:allow file.2
# ls -v file.2
-rw-r--r--+  1 root     root        2693 Jul 20 14:26 file.2
```

例 7–5 ACL セットを ZFS ファイルに適用する (続き)

```
     0:user:otto:read_data/write_data/append_data/read_xattr/write_xattr
         /read_attributes/write_attributes/read_acl: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
```

ZFS ファイルの ACL 継承を冗長形式で設定する

ファイルとディレクトリに ACL をどのように継承するかまたは継承しないかを決定できます。デフォルトでは、ACL は伝達されません。ディレクトリに非簡易 ACL を設定した場合でも、その ACL はそれ以降に作成されるディレクトリには継承されません。ACL を継承する場合は、ファイルまたはディレクトリにそのことを 指定する必要があります。

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、241 ページの「ACL 継承」を参照してください。

例 7–6 デフォルトの ACL 継承を許可する

デフォルトでは、ACL はディレクトリ階層に伝達されません。

次の例では、read_data/write_data/execute の非簡易 ACL が、test.dir のユーザー gozer に適用されます。

```
# chmod A+user:gozer:read_data/write_data/execute:allow test.dir
# ls -dv test.dir
drwxr-xr-x+  2 root     root           2 Jul 20 14:53 test.dir
     0:user:gozer:list_directory/read_data/add_file/write_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
```

test.dir サブディレクトリが作成されても、ユーザー gozer の ACE は伝達されません。sub.dir 上でユーザー gozer に許可されているアクセス権がファイル所有者、グループメンバー、または everyone@ としてのアクセス権の場合には、このユーザーは sub.dir にしかアクセスできません。

例 7–6 デフォルトの ACL 継承を許可する (続き)

```
# mkdir test.dir/sub.dir
# ls -dv test.dir/sub.dir
drwxr-xr-x   2 root     root           2 Jul 20 14:54 test.dir/sub.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
```

例 7–7 ファイルとディレクトリの ACL 継承を許可する

ここで示す一連の例では、file_inherit フラグが設定されているときに適用されるファイルとディレクトリの ACE を示しています。

次の例では、test2.dir ディレクトリ上のファイルへの read_data/write_data アクセス権がユーザー gozer に追加されます。このユーザーは、新しく作成されたすべてのファイルに読み取りアクセスできるようになります。

```
# chmod A+user:gozer:read_data/write_data:file_inherit:allow test2.dir
# ls -dv test2.dir
drwxr-xr-x+  2 root     root           2 Jul 20 14:55 test2.dir
     0:user:gozer: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 gozer のアクセス権が、新しく作成されたファイル test2.dir/file.2 に適用されます。ACL 継承が許可されているので (read_data:file_inherit:allow)、ユーザー gozer は新しく作成されたすべてのファイルの内容を読み取ることができます。

```
# 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:gozer: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
```

例 7–7 ファイルとディレクトリの ACL 継承を許可する (続き)

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 gozer には file.2 への write_data アクセス権は割り当てられません。これは、ファイルのグループアクセス権が許可していないためです。

inherit_only アクセス権は、file_inherit または dir_inherit フラグが設定されているときに適用されます。このアクセス権は、ディレクトリ階層に ACL を伝達するために使用します。この場合、ユーザー gozer のアクセス権の許可または拒否は、ファイル所有者またはファイルのグループ所有者のメンバーである場合を除いて、everyone@ アクセス権に基づいてのみ行われます。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:gozer: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
```

次に示す一連の例では、file_inherit と dir_inherit フラグが両方設定されているときに適用される、ファイルとディレクトリの ACL を示しています。

次の例では、ユーザー gozer に読み取り、書き込み、および実行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:gozer: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:gozer: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
```

下記の出力内にある 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。

例 7–7 ファイルとディレクトリの ACL 継承を許可する (続き)

```
# 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:gozer: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
```

上記の例では、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るため、ユーザー gozer 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 gozer 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:gozer: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:gozer: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、gozer 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:gozer: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
```

例 7–8　ACL 継承モードが passthrough に設定された ACL 継承

tank/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 gozer 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 tank/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:gozer: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 7–9　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 tank/cindy
# chmod A+user:gozer:read_data/write_data/execute:dir_inherit:allow test5.dir
# ls -dv test5.dir
drwxr-xr-x+  2 root     root           2 Jul 20 14:18 test5.dir
     0:user:gozer: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 7–10　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 tank/cindy
# chmod A+user:gozer:read_data:file_inherit:deny,user:lp:read_data:file_inherit:allow
test6.dir
# ls -dv test6.dir
drwxr-xr-x+  2 root     root           2 Jul 20 14:22 test6.dir
     0:user:gozer: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:gozer: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
```

ZFS ファイルの ACL をコンパクト形式で設定および表示する

ZFS ファイルのアクセス権をコンパクト形式で設定および表示できます。コンパクト形式では、一意の 14 個の文字を使ってアクセス権を表現します。コンパクトなアクセス権を表現する文字の一覧は、表 7–2 および表 7–4 に記載されています。

ファイルとディレクトリのコンパクト形式の ACL リストは、ls -V コマンドを使用して表示できます。例:

```
# 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
```

コンパクト形式の ACL 出力について説明します。

`owner@` この所有者は、このファイルの内容を読み取って変更することができます (`rw=read_data/write_data`)、 (`p=append_data`)。この所有者は、タイムスタンプ、拡張属性、ACL などのファイル属性を変更することもできます (`a=read_attributes`、 `W=write_xattr`、 `R=read_xattr`、 `A=write_attributes`、 `c=read_acl`、 `C=write_acl`)。さらに、この所有者はファイルの所有権を変更することもできます (`o=write_owner`)。

`synchronize` (`s`) アクセス権は、現在のところ実装されていません。

`group@` グループには、ファイルへの読み取りアクセス権 (`r= read_data`) およびファイルの属性への読み取りアクセス権 (`a=read_attributes`、 `R=read_xattr`、 `c= read_acl`) が付与されます。

`synchronize` (`s`) アクセス権は、現在のところ実装されていません。

`everyone@` ユーザーやグループ以外の全員は、このファイルおよびこのファイルの属性を読み取るアクセス権が付与されます (`r=read_data`、 `a=append_data`、 `R=read_xattr`、 `c=read_acl`、 `s=synchronize`)。

`synchronize` (`s`) アクセス権は、現在のところ実装されていません。

コンパクト形式の ACL には、冗長形式の ACL と比べて次の利点があります。

- アクセス権を `chmod` コマンドに指定するときに、順対応引数として指定できます。
- アクセス権がないことを表すハイフン (-) 文字は、省略してもかまいません。必要な文字だけを指定する必要があります。
- アクセス権と継承フラグは、同じ方法で設定します。

冗長形式の ACL の使用方法については、246 ページの「ZFS ファイルの ACL を冗長形式で設定および表示する」を参照してください。

例 7–11 コンパクト形式で ACL を設定および表示する

次の例では、簡易 ACL が `file.1` にあります。

```
# ls -V file.1
-rw-r--r--   1 root     root      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
```

例 7–11 コンパクト形式で ACL を設定および表示する (続き)

次の例では、ユーザー gozer に file.1 の read_data/execute アクセス権が追加されます。

```
# chmod A+user:gozer:rx:allow file.1
# ls -V file.1
-rw-r--r--+  1 root     root      206695 Jul 20 14:27 file.1
             user:gozer:r-x-----------:-------:allow
                 owner@:rw-p--aARWcCos:-------:allow
                 group@:r-----a-R-c--s:-------:allow
              everyone@:r-----a-R-c--s:-------:allow
```

次の例では、コンパクト形式を使ってユーザー gozer に読み取り、書き込み、および実行アクセス権を付与します。これらのアクセス権は、新しく作成されたファイルとディレクトリに継承されます。

```
# chmod A+user:gozer:rwx:fd:allow dir.2
# ls -dV dir.2
drwxr-xr-x+  2 root     root           2 Jul 20 14:33 dir.2
             user:gozer:rwx-----------:fd-----:allow
                 owner@:rwxp-DaARWcCos:-------:allow
                 group@:r-x---a-R-c--s:-------:allow
              everyone@:r-x---a-R-c--s:-------:allow
```

ls -V の出力にあるアクセス権と継承フラグをコンパクト形式の chmod にカット&ペーストすることもできます。たとえば、ユーザー gozer についての dir.2 のアクセス権と継承フラグを dir.2 上のユーザー cindy に複製するには、アクセス権と継承フラグ (rwx-----------:fd-----:allow) を chmod コマンドにコピー&ペーストします。例:

```
# chmod A+user:cindy:rwx-----------:fd-----:allow dir.2
# ls -dV dir.2
drwxr-xr-x+  2 root     root           2 Jul 20 14:33 dir.2
             user:cindy:rwx-----------:fd-----:allow
             user:gozer:rwx-----------:fd-----:allow
                 owner@:rwxp-DaARWcCos:-------:allow
                 group@:r-x---a-R-c--s:-------:allow
              everyone@:r-x---a-R-c--s:-------:allow
```

例 7–12 ACL 継承モードが passthrough に設定された ACL 継承

aclinherit プロパティーが passthrough に設定されているファイルシステムは、継承時に ACL エントリに加えられた変更を除く、継承可能なすべての ACL エントリを継承します。このプロパティーが passthrough に設定されている場合、作成されるファイルのアクセス権モードは継承可能な ACL によって決定されます。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 7–12　ACL 継承モードが passthrough に設定された ACL 継承　(続き)

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 tank/cindy
# pwd
/tank/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。

例 7–12　ACL 継承モードが passthrough に設定された ACL 継承　　(続き)

```
# cd /tank/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
```

例 7–13　ACL 継承モードが passthrough-x に設定された ACL 継承

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 に設定して実行権を継承する方法を示します。

```
# zfs set aclinherit=passthrough-x tank/cindy
```

次の ACL は /tank/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
```

例 7–13　ACL 継承モードが passthrough-x に設定された ACL 継承　*(続き)*

```
-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。

例 7–14　ZFS ファイルでの ACL と chmod 操作との相互作用

次の例では、特定の aclmode および aclinherit プロパティー値が、既存の ACL と chmod 操作との関連にどのように影響するかについて説明します。この操作は、ファイルまたはディレクトリアクセス権を変更して、所有グループに一致するように既存の ACL アクセス権を縮小または拡張させるものです。

この例では、aclmode プロパティーは mask に設定され、aclinherit プロパティーは restricted に設定されます。この例の ACL アクセス権は、変更中のアクセス権をより示しやすくするコンパクトモードで表示されます。

元のファイルおよびグループ所有権と ACL アクセス権は次のとおりです。

```
# zfs set aclmode=mask pond/whoville
# zfs set aclinherit=restricted pond/whoville

# 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
```

chown 操作によって file.1 のファイル所有権が変更され、所有しているユーザー amy によって出力が表示されます。例:

```
# chown amy:staff file.1
# su - amy
$ 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
```

例 7–14 ZFS ファイルでの ACL と chmod 操作との相互作用 (続き)

次の chmod 操作では、アクセス権がより制限の厳しいモードに変更されます。この例では、変更された sysadmin グループと staff グループの ACL アクセス権が、所有しているグループのアクセス権を超えることはありません。

```
$ 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
```

次の chmod 操作では、アクセス権がより制限の緩やかなモードに変更されます。この例では、変更された sysadmin グループと staff グループの ACL アクセス権が、所有しているグループと同じアクセス権を許可するように復元されます。

```
$ 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
```

特別な属性を ZFS ファイルに適用する

次の例は、不変性や読み取り専用アクセス権などの特別な属性を ZFS ファイルに適用し表示する方法について示します。

特別な属性の表示および適用の詳細については、ls(1) および chmod(1) を参照してください。

例 7–15 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
```

例 7–15 不変性を ZFS ファイルに適用する (続き)

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
```

例 7–16 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
```

例 7–17 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 8 章

Oracle Solaris ZFS 委任管理

この章では、ZFS 委任管理を使用して、特権のないユーザーが ZFS 管理タスクを実行できるようにする方法について説明します。

この章は、次の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。次の 2 種類の委任アクセス権がサポートされています。

- 作成、破棄、マウント、スナップショットといった個別のアクセス権を明示的に委任できます。
- 「アクセス権セット」と呼ばれるアクセス権の集まりを定義できます。アクセス権セットはあとで更新することができ、そのセットの使用者は自動的に変更内容を取得します。アクセス権セットは @ 記号で始まり、64 文字以下の長さに制限されています。@ 記号に続くセット名の残り部分の文字には、通常の ZFS ファイルシステム名と同じ制限事項が適用されます。

ZFS 委任管理では、RBAC セキュリティーモデルに似た機能が提供されます。ZFS 委任を使用すると、ZFS ストレージプールおよびファイルシステムの管理に次のような利点が得られます。

- ZFS ストレージプールの移行時には常にアクセス権も移行されます。

- 動的継承により、ファイルシステム間でアクセス権をどのように伝達するかを制御できます。
- ファイルシステムの作成者だけがそのファイルシステムを破棄できるように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。RBAC を使用して一般的な Oracle Solaris 管理タスクを管理する方法については、『Oracle Solaris 11.1 の管理: セキュリティーサービス』のパート III「役割、権利プロファイル、特権」を参照してください。

ZFS 委任アクセス権を無効にする

委任管理機能を制御するには、プールの `delegation` プロパティーを使用します。次に例を示します。

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

デフォルトでは、`delegation` プロパティーは有効になっています。

ZFS アクセス権の委任

`zfs allow` コマンドを使用して、ZFS 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。	

アクセス権 (サブコマンド)	説明	依存関係
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
- 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、147 ページの「ZFS のプロパティーの紹介」を参照してください。

ZFS アクセス権の委任 (zfs allow)

zfs allow の構文を次に示します。

zfs allow -[ldugecs] everyone|*user*|*group*[,...] *perm*|@*setname*,...] *filesystem*| *volume*

次の zfs allow 構文 (太字) は、アクセス権の委任先を示しています。

zfs allow [-uge]|user|group|everyone [,...] *filesystem* | *volume*

複数のエンティティーをコンマ区切りのリストとして指定できます。-uge オプションが指定されていない場合、引数はキーワード everyone、ユーザー名、最後にグループ名という優先順位で解釈されます。「everyone」という名前のユーザーまたはグループを指定するには、-u オプションまたは -g オプションを使用します。ユーザーと同じ名前のグループを指定するには、-g オプションを使用します。-c オプションは作成時のアクセス権を委任します。

次の zfs allow 構文 (太字) は、アクセス権およびアクセス権セットの指定方法を示しています。

zfs allow [-s] ... perm|@setname [,...] *filesystem* | *volume*

複数のアクセス権をコンマ区切りのリストとして指定できます。アクセス権の名前は、ZFS のサブコマンドおよびプロパティーと同じです。詳細は、前の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、および snapshot アクセス権を次のように委任したとします。

```
# zfs allow cindy create,destroy,mount,snapshot tank/home/cindy
# zfs allow tank/home/cindy
---- Permissions on tank/home/cindy --------------------------------
Local+Descendent 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 tank/home/cindy
# zfs allow tank/home/cindy
```

ZFS アクセス権を委任する (例)

例 8–1 個別のユーザーにアクセス権を委任する

create アクセス権と mount アクセス権を個別のユーザーに委任する場合は、そのユーザーが配下のマウントポイントに対するアクセス権を持っていることを確認する必要があります。

たとえば、ユーザー mark に create アクセス権と mount アクセス権を tank ファイルシステムに関して委任するには、まず次のようにアクセス権を設定します。

```
# chmod A+user:mark:add_subdirectory:fd:allow /tank/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 tank/home
```

これで、ユーザー mark は tank/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るようになります。例:

```
# su mark
mark$ zfs create tank/home/mark
mark$ ^D
# su lp
$ zfs create tank/home/lp
cannot create 'tank/home/lp': permission denied
```

例 8–2 グループに create および destroy アクセス権を委任する

次の例では、ファイルシステムを設定して、staff グループの任意のメンバーが tank/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 グループのメンバーであっても、ほかのメンバーのファイルシステムを破棄することはできません。

```
# zfs allow staff create,mount tank/home
# zfs allow -c create,destroy tank/home
# zfs allow tank/home
---- Permissions on tank/home --------------------------------------
Create time permissions:
        create,destroy
Local+Descendent permissions:
        group staff create,mount
# su cindy
cindy% zfs create tank/home/cindy/files
cindy% exit
# su mark
mark% zfs create tank/home/mark/data
mark% exit
cindy% zfs destroy tank/home/mark/data
```

例 8–2 グループに create および destroy アクセス権を委任する (続き)

```
cannot destroy 'tank/home/mark/data': permission denied
```

例 8–3 正しいファイルシステムレベルでアクセス権を委任する

ユーザーにアクセス権を委任する場合は、必ず正しいファイルシステムレベルで委任してください。たとえば、ユーザー 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 tank/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 tank/home
# zfs allow tank/home
---- Permissions on tank/home --------------------------------------
Create time permissions:
        create,destroy
Local permissions:
        user mark snapshot
Local+Descendent permissions:
        group staff create,mount
# su mark
mark$ zfs snapshot tank/home@snap1
mark$ zfs snapshot tank/home/mark@snap1
cannot create snapshot 'tank/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 tank/home
# zfs allow -d mark snapshot tank/home
# zfs allow tank/home
---- Permissions on tank/home --------------------------------------
Create time permissions:
        create,destroy
Descendent permissions:
        user mark snapshot
Local+Descendent permissions:
        group staff create,mount
# su mark
$ zfs snapshot tank/home@snap2
cannot create snapshot 'tank/home@snap2': permission denied
$ zfs snapshot tank/home/mark@snappy
```

これで、ユーザー mark は tank/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るようになります。

例 8–4 複雑な委任アクセス権を定義して使用する

特定のアクセス権をユーザーやグループに委任できます。たとえば、次の zfs allow コマンドでは、特定のアクセス権が staff グループに委任されます。また、destroy アクセス権と snapshot アクセス権が tank/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 tank/home
# zfs allow -c destroy,snapshot tank/home
# zfs allow tank/home
---- Permissions on tank/home --------------------------------------
Create time permissions:
        create,destroy,snapshot
Local+Descendent permissions:
        group staff create,mount
```

ユーザー mark は staff グループのメンバーなので、tank/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、tank/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 tank/home/mark2
$ zfs allow tank/home/mark2
---- Permissions on tank/home/mark2 --------------------------------
Local permissions:
        user mark create,destroy,snapshot
---- Permissions on tank/home --------------------------------------
Create time permissions:
        create,destroy,snapshot
Local+Descendent permissions:
        group staff create,mount
```

ただし、ユーザー mark は tank/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 tank/home/mark@snap1
cannot create snapshot 'tank/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 /tank/home/mark2
$ ls
$ cd .zfs
$ ls
shares snapshot
$ cd snapshot
$ ls -l
total 3
```

例 8–4 複雑な委任アクセス権を定義して使用する (続き)

```
drwxr-xr-x   2 mark    staff          2 Sep 27 15:55 snap1
$ pwd
/tank/home/mark2/.zfs/snapshot
$ mkdir snap2
$ zfs list
# zfs list -r tank/home
NAME                   USED  AVAIL  REFER  MOUNTPOINT
tank/home/mark          63K  62.3G    32K  /tank/home/mark
tank/home/mark2         49K  62.3G    31K  /tank/home/mark2
tank/home/mark2@snap1   18K      -    31K  -
tank/home/mark2@snap2     0      -    31K  -
$ ls
snap1  snap2
$ rmdir snap2
$ ls
snap1
```

例 8–5 ZFS 委任アクセス権セットを定義して使用する

次の例では、アクセス権セット @myset を作成し、グループ staff にこのアクセス権セットと rename アクセス権を tank ファイルシステムに関して委任する方法を示します。ユーザー cindy は staff グループのメンバーであり、tank にファイルシステムを作成するアクセス権を持っています。ただし、ユーザー lp は tank にファイルシステムを作成するアクセス権を持っていません。

```
# zfs allow -s @myset create,destroy,mount,snapshot,promote,clone,readonly tank
# zfs allow tank
---- Permissions on tank --------------------------------------------
Permission sets:
        @myset clone,create,destroy,mount,promote,readonly,snapshot
# zfs allow staff @myset,rename tank
# zfs allow tank
---- Permissions on tank --------------------------------------------
Permission sets:
        @myset clone,create,destroy,mount,promote,readonly,snapshot
Local+Descendent permissions:
        group staff @myset,rename
# chmod A+group:staff:add_subdirectory:fd:allow tank
# su cindy
cindy% zfs create tank/data
cindy% zfs allow tank
---- Permissions on tank --------------------------------------------
Permission sets:
        @myset clone,create,destroy,mount,promote,readonly,snapshot
Local+Descendent permissions:
        group staff @myset,rename
cindy% ls -l /tank
total 15
drwxr-xr-x   2 cindy   staff          2 Jun 24 10:55 data
cindy% exit
# su lp
$ zfs create tank/lp
cannot create 'tank/lp': permission denied
```

ZFS 委任アクセス権を表示する (例)

次のコマンドを使用して、アクセス権を表示できます。

 # zfs allow *dataset*

このコマンドでは、指定されたデータセットに設定または許可されているアクセス権が表示されます。出力には、次の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 8–6 基本的な委任管理アクセス権を表示する

次の出力は、ユーザー cindy が tank/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 tank/cindy
       -------------------------------------------------------------
       Local+Descendent permissions on (tank/cindy)
               user cindy create,destroy,mount,snapshot
```

例 8–7 複雑な委任管理アクセス権を表示する

次の例の出力は、pool/fred ファイルシステムと pool ファイルシステムに対する次のようなアクセス権を示しています。

pool/fred ファイルシステムに対しては次のとおりです。

- 次の 2 つのアクセス権セットが定義されています。
  - @eng (create, destroy , snapshot, mount, clone , promote, rename)
  - @simple (create, mount)
- 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。

例 8–7 複雑な委任管理アクセス権を表示する (続き)

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+Descendent 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 アクセス権を削除する (例)

zfs unallow コマンドを使用して、委任したアクセス権を削除できます。たとえば、ユーザー cindy は tank/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 tank/home/cindy
# zfs allow tank/home/cindy
---- Permissions on tank/home/cindy --------------------------------
Local+Descendent 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 tank/home/cindy ファイルシステムから削除されます。

```
# zfs unallow cindy snapshot tank/home/cindy
# zfs allow tank/home/cindy
---- Permissions on tank/home/cindy --------------------------------
Local+Descendent permissions:
        user cindy create,destroy,mount
cindy% zfs create tank/home/cindy/data
```

```
cindy% zfs snapshot tank/home/cindy@today
cannot create snapshot 'tank/home/cindy@today': permission denied
```

別の例として、ユーザー mark は tank/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 tank/home/mark
---- Permissions on tank/home/mark --------------------------------
Local+Descendent 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 tank/home/mark ファイルシステムから削除されます。

```
# zfs unallow mark tank/home/mark
```

次の zfs unallow 構文では、tank ファイルシステムのアクセス権セットが削除されます。

```
# zfs allow tank
---- Permissions on tank ------------------------------------------
Permission sets:
        @myset clone,create,destroy,mount,promote,readonly,snapshot
Create time permissions:
        create,destroy,mount
Local+Descendent permissions:
        group staff create,mount
# zfs unallow -s @myset tank
# zfs allow tank
---- Permissions on tank ------------------------------------------
Create time permissions:
        create,destroy,mount
Local+Descendent permissions:
        group staff create,mount
```

◆ ◆ ◆ 第 9 章

9 Oracle Solaris ZFS の高度なトピック

この章では、ZFS ボリューム、ゾーンがインストールされた Solaris システムで ZFS を使用する方法、ZFS 代替ルートプール、および ZFS 権利プロファイルについて説明します。

この章は、次のセクションで構成されます。

ZFS ボリューム

ZFS ボリュームとは、ブロックデバイスを表すデータセットです。ZFS ボリュームは、`/dev/zvol/{dsk,rdsk}/pool` ディレクトリのデバイスとして識別されます。

次の例では、5G バイトの ZFS ボリューム `tank/vol` が作成されます。

```
# zfs create -V 5gb tank/vol
```

ボリュームの作成時には、予期しない動作が発生しないよう、予約が自動的にボリュームの初期サイズに設定されます。たとえば、ボリュームのサイズを縮小すると、データが破壊される可能性があります。ボリュームのサイズを変更するときは、注意深く行う必要があります。

また、サイズが変化するボリュームのスナップショットを作成する場合は、スナップショットをロールバックしたり、スナップショットからのクローンを作成しようとすると、不一致が発生する可能性があります。

ボリュームに適用可能なファイルシステムプロパティーについては、表 5–1 を参照してください。

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 tank/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 tank/vol
tank/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、297 ページの「ZFS の問題を解決する」を参照してください。

ゾーンがインストールされた Solaris システムを使用している場合は、非大域ゾーンの中で ZFS ボリュームを作成または複製することはできません。そうしようとしても失敗します。ZFS ボリュームを大域ゾーンで使用する方法については、285 ページの「ZFS ボリュームを非大域ゾーンに追加する」を参照してください。

ZFS ボリュームをスワップデバイスまたはダンプデバイスとして使用する

ZFS ルートファイルシステムをインストールするとき、または UFS ルートファイルシステムから移行するときに、ZFS ルートプールの ZFS ボリュームにスワップデバイスが作成されます。例:

```
# swap -l
swapfile                  dev    swaplo   blocks     free
/dev/zvol/dsk/rpool/swap 253,3        16  8257520  8257520
```

ZFS ルートファイルシステムをインストールするとき、または UFS ルートファイルシステムから移行するときに、ZFS ルートプールの ZFS ボリュームにダンプデバイスが作成されます。ダンプデバイスを設定したあとは、ダンプデバイスの管理は不要です。例:

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

システムのインストール後にスワップ領域やダンプデバイスを変更する必要がある場合は、以前の Solaris リリースと同様に swap コマンドと dumpadm コマンドを使用します。追加のスワップボリュームを作成する必要がある場合は、特定のサイズの ZFS ボリュームを作成してから、そのデバイスでスワップを有効にします。次に、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、132 ページの「ZFS スワップデバイスおよびダンプデバイスのサイズを調整する」を参照してください。

iSCSI LUN として ZFS ボリュームを使用する

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 ホストを、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。

まず、COMSTAR パッケージをインストールします。

```
# pkg install group/feature/storage-server
```

次に、iSCSI ターゲットとして使用する ZFS ボリュームを作成し、続いて SCSI ブロック型デバイスベースの LUN を作成します。例:

```
# zfs create -V 2g tank/volumes/v2
# sbdadm create-lu /dev/zvol/rdsk/tank/volumes/v2
Created the following LU:

              GUID                    DATA SIZE           SOURCE
--------------------------------  -------------------  ----------------
600144f000144f1dafaa4c0faff20001  2147483648           /dev/zvol/rdsk/tank/volumes/v2
# sbdadm list-lu
Found 1 LU(s)

              GUID                    DATA SIZE           SOURCE
--------------------------------  -------------------  ----------------
600144f000144f1dafaa4c0faff20001  2147483648           /dev/zvol/rdsk/tank/volumes/v2
```

すべてのクライアントまたは選択したクライアントに LUN ビューを公開できます。LUN GUID を特定し、続いて LUN ビューを共有します。次の例では、LUN ビューがすべてのクライアントに共有されています。

```
# stmfadm list-lu
LU Name: 600144F000144F1DAFAA4C0FAFF20001
# stmfadm add-view 600144F000144F1DAFAA4C0FAFF20001
# stmfadm list-view -l 600144F000144F1DAFAA4C0FAFF20001
```

```
View Entry: 0
    Host group   : All
    Target group : All
    LUN          : 0
```

次の手順は、iSCSI ターゲットの作成です。iSCSI ターゲットの作成については、『Oracle Solaris 11.1 の管理: デバイスとファイルシステム』の第 11 章「COMSTAR を使用したストレージデバイスの構成 (タスク)」を参照してください。

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。次のようなメッセージが表示されます。

```
# zfs rename tank/volumes/v2 tank/volumes/v1
cannot rename 'tank/volumes/v2': dataset is busy
# zpool export tank
cannot export 'tank': 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。

ゾーンがインストールされている Solaris システムで ZFS を使用する

以降のセクションでは、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。

- 283 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」
- 284 ページの「データセットを非大域ゾーンに委任する」
- 285 ページの「ZFS ボリュームを非大域ゾーンに追加する」
- 285 ページの「ZFS ストレージプールをゾーンで使用する」
- 286 ページの「ZFS プロパティーをゾーンで管理する」
- 286 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 クローンを非大域ゾーンに追加できますが、管理者制御を委任しても委任しなくてもかまいません。
- ZFS ボリュームをデバイスとして非大域ゾーンに追加できます。
- この時点で、ZFS スナップショットをゾーンに関連付けることはできません。

以降のセクションでは、ZFS データセットはファイルシステムまたはクローンを指します。

データセットを追加すると、非大域ゾーンは大域ゾーンとディスク領域を共有できます。ただし、ゾーン管理者は、配下のファイルシステム階層でプロパティーを制御したり、新しいファイルシステムを作成したりすることはできません。この動作は、ほかの種類のファイルシステムをゾーンに追加する場合と同じであり、共通のディスク領域を共有することが目的の場合にのみ使用してください。

ZFS では、データセットを非大域ゾーンに委任して、データセットとそのすべての子を完全に制御する権限をゾーン管理者に渡すこともできます。ゾーン管理者は、そのデータセット内でファイルシステムやクローンを作成および破棄したり、データセットのプロパティーを変更したりできます。ゾーン管理者は、委任されたデータセットに設定された最上位の割り当て制限を超過するなど、ゾーンに追加されていないデータセットに影響を与えることはできません。

Oracle Solaris ゾーンがインストールされたシステム上で ZFS を操作する場合には、次の点を考慮してください。

- 非大域ゾーンに追加する ZFS ファイルシステムでは、`mountpoint` プロパティーを `legacy` に設定する必要があります。
- ソース `zonepath` とターゲット `zonepath` がどちらも ZFS ファイルシステム上に存在し、同じプール内にある場合、`zoneadm clone` は自動的に ZFS クローンを使ってゾーンを複製するようになりました。`zoneadm clone` コマンドは、ソース `zonepath` の ZFS スナップショットを作成し、ターゲット `zonepath` を設定します。`zfs clone` コマンドを使用してゾーンを複製することはできません。詳細は、『Oracle Solaris のシステム管理 (Oracle Solaris ゾーン、Oracle Solaris 10 ゾーン、およびリソース管理)』のパート II「Oracle Solaris ゾーン」を参照してください。

ZFS ファイルシステムを非大域ゾーンに追加する

大域ゾーンと領域を共有する必要がある場合は、ZFS ファイルシステムを汎用のファイルシステムとして追加して、その目的のためだけに使用できます。 非大域ゾーンに追加する ZFS ファイルシステムでは、`mountpoint` プロパティーを `legacy` に設定する必要があります。たとえば、`tank/zone/zion` ファイルシステムを非大域ゾーンに追加する場合、大域ゾーンの `mountpoint` プロパティーを次のように設定します。

```
# zfs set mountpoint=legacy tank/zone/zion
```

`zonecfg` コマンドの `add fs` サブコマンドを使用することで、ZFS ファイルシステムを非大域ゾーンに追加できます。

次の例では、大域ゾーンの大域ゾーン管理者が、ZFS ファイルシステムを非大域ゾーンに追加しています。

```
# zonecfg -z zion
zonecfg:zion> add fs
zonecfg:zion:fs> set type=zfs
```

```
zonecfg:zion:fs> set special=tank/zone/zion
zonecfg:zion:fs> set dir=/opt/data
zonecfg:zion:fs> end
```

この構文では、ZFS ファイルシステム tank/zone/zion がすでに構成済みの zion ゾーンに追加され、/opt/data にマウントされます。ファイルシステムの mountpoint プロパティーは、legacy に設定する必要があります。別の場所にすでにマウントされているファイルシステムは追加できません。ゾーン管理者は、ファイルシステム内でファイルを作成および破棄することができます。ファイルシステムを別の場所に再マウントすることはできません。また、ゾーン管理者がファイルシステムのプロパティー (atime、readonly、compression など) を変更することもできません。大域ゾーン管理者は、ファイルシステムのプロパティーの設定および制御を担当します。

zonecfg コマンドの詳細と、zonecfg を使用したリソースタイプの構成方法については、『Oracle Solaris のシステム管理 (Oracle Solaris ゾーン、Oracle Solaris 10 ゾーン、およびリソース管理)』のパート II「Oracle Solaris ゾーン」を参照してください。

データセットを非大域ゾーンに委任する

ストレージの管理をゾーンに委任するという主要目的を果たせるよう、ZFS では、zonecfg コマンドの add dataset サブコマンドを使用してデータセットを非大域ゾーンに追加することができます。

次の例では、大域ゾーンの大域ゾーン管理者が、ZFS ファイルシステムを非大域ゾーンに委任しています。

```
# zonecfg -z zion
zonecfg:zion> add dataset
zonecfg:zion:dataset> set name=tank/zone/zion
zonecfg:zion:dataset> set alias=tank
zonecfg:zion:dataset> end
```

ファイルシステムを追加する場合と異なり、この構文を実行すると、ZFS ファイルシステム tank/zone/zion がすでに構成済みの zion ゾーンから見えるようになります。zion ゾーン内では、このファイルシステムは tank/zone/zion としてアクセスできませんが、tank 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。 特定の別名が指定されていなかったとすると、上記の例では、デフォルトの別名が zion になっていたところでした。

委任されたデータセット内で、ゾーン管理者は、ファイルシステムのプロパティーを設定したり、子孫ファイルシステムを作成したりできます。また、ゾーン管理者は、スナップショットやクローンを作成し、およびファイルシステム階層全体を制御することができます。委任されたファイルシステムで ZFS ボリュームが作

成されると、デバイスリソースとして追加される ZFS ボリュームとの競合が可能になります。詳細については、次のセクションと dev(7FS) を参照してください。

ZFS ボリュームを非大域ゾーンに追加する

次のようにして、非大域ゾーンで ZFS ボリュームを追加または作成したり、非大域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。

- 非大域ゾーンでは、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。例:

  ```
  # zfs create -V 2g tank/zone/zion/vol1
  ```

  上記の構文は、ゾーン管理者が非大域ゾーンでボリュームのプロパティーおよびデータを管理できることを示しています。

- 大域ゾーンで zonecfg add dataset サブコマンドを使用し、非大域ゾーンに追加する ZFS ボリュームを指定します。例:

  ```
  # zonecfg -z zion
  zonecfg:zion> add dataset
  zonecfg:zion:dataset> set name=tank/volumes/vol1
  zonecfg:zion:dataset> end
  ```

  上記の構文は、ゾーン管理者が非大域ゾーンでボリュームのプロパティーおよびデータを管理できることを示しています。

- 大域ゾーンで zonecfg add device サブコマンドを使用して、非大域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:

  ```
  # zonecfg -z zion
  zonecfg:zion> add device
  zonecfg:zion:device> set match=/dev/zvol/dsk/tank/volumes/vol2
  zonecfg:zion:device> end
  ```

  上記の構文は、非大域ゾーンでボリュームデータだけにアクセスできることを示しています。

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ゾーンの内部で作成または変更することはできません。委任管理モデルを使用することで、大域ゾーン内の物理ストレージデバイスの制御と仮想ストレージの制御をすべて非大域ゾーンで行うことができます。プールレベルのデータセットをゾーンに追加することはできますが、デバイスを作成したり、追加したり、削除したりするなど、プールの物理特性を変更するコマンドはゾーンの内部から実行することはできません。zonecfg コマンドの add device サブコマンドを使用して物理デバイスをゾーンに追加する場合でも、ファイルを使用する場合でも、zpool コマンドを使用してゾーンの内部に新しいプールを作成することはできません。

ZFS プロパティーをゾーンで管理する

データセットをゾーンに委任したあとで、ゾーン管理者は特定のデータセットプロパティーを制御できます。ゾーンに委任したデータセットのすべての祖先は、読み取り専用データセットとして表示されます。ただし、データセット自体およびそのすべての子孫は書き込み可能です。たとえば、次のような構成を考えてみます。

```
global# zfs list -Ho name
tank
tank/home
tank/data
tank/data/matrix
tank/data/zion
tank/data/zion/home
```

tank/data/zion 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
tank	いいえ	-	-
tank/home	いいえ	-	-
tank/data	いいえ	-	-
tank/data/zion	はい	はい	zoned、 quota、 reservation
tank/data/zion/home	はい	はい	zoned

tank/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、非大域ゾーンで使用されるすべてのデータセットが使用するディスク容量を制御できます。

また、データセットを非大域ゾーンに委任したあとに、大域ゾーン管理者が 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

データセットを非大域ゾーンに委任するときに、特定のプロパティーが大域ゾーンのコンテキストで解釈されないように、データセットに特別な設定を行う必要があります。データセットが非大域ゾーンに委任され、ゾーン管理者の制御下に入る

と、その内容は信頼できる状態ではなくなります。どのファイルシステムにも該当することですが、setuid バイナリやシンボリックリンクなどの安全性に問題のある内容が含まれていることがあります。これらは、大域ゾーンのセキュリティーに悪影響を及ぼす可能性があります。また、mountpoint プロパティーは、大域ゾーンのコンテキストでは解釈できません。それ以外に、ゾーン管理者が大域ゾーンの名前空間に影響を及ぼす可能性もあります。後者の問題に対処するために、ZFS では zoned プロパティーを使って、データセットがある時点で非大域ゾーンに委任されていることを示しています。

zoned プロパティーはブール値で、ZFS データセットを含むゾーンが最初に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オンに設定する必要はありません。zoned プロパティーを設定した場合、そのデータセットを大域ゾーンでマウントしたり共有したりすることはできません。次の例では、tank/zone/zion はゾーンに委任されていますが、tank/zone/global は追加されていません。

```
# zfs list -o name,zoned,mountpoint -r tank/zone
NAME                  ZONED  MOUNTPOINT
tank/zone/global        off  /tank/zone/global
tank/zone/zion           on  /tank/zone/zion
# zfs mount
tank/zone/global           /tank/zone/global
tank/zone/zion             /export/zone/zion/root/tank/zone/zion
```

mountpoint プロパティーと、tank/zone/zion データセットが現在マウントされているディレクトリとが異なっていることに注意してください。mountpoint プロパティーには、データセットがシステム上で現在マウントされている場所ではなく、ディスクに格納されているプロパティーが反映されます。

データセットがゾーンから削除されたり、ゾーンが破棄されたりした場合でも、zoned プロパティーが自動的に消去されることはありません。これらのタスクに関連するセキュリティー上の危険が潜在的に存在するために、このような動作になっています。信頼されないユーザーがデータセットとその子孫へのアクセスを完了してしまっているので、mountpoint プロパティーが不正な値に設定されたり、ファイルシステムに setuid バイナリが存在したりする可能性があります。

意図しないセキュリティー上の危険を防ぐために、データセットをなんらかの方法で再利用する場合には、大域ゾーン管理者が zoned プロパティーを手動で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のは、データセットが非大域ゾーンで使用されていないことを確認した場合にのみ行なってください。

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、`zfs send` コマンドと `zfs receive` コマンドを使用することを検討してください。状況によっては、複製ストリームや再帰的ストリームを使用する方法が最適な場合もあります。

このセクションの例では、ゾーンデータをシステム間でコピーする方法について説明します。ゾーンの構成を転送し、各ゾーンを新しいシステムに接続するには、追加手順が必要になります。詳細は、『Oracle Solaris 11.1 の管理: Oracle Solaris ゾーン、Oracle Solaris 10 ゾーン、およびリソース管理』のパート II「Oracle Solaris ゾーン」を参照してください。

あるシステムのすべてのゾーンを別のシステムに移動する必要がある場合、複製ストリームを使用することを検討してください。複製ストリームはスナップショットとクローンを保持するからです。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、`sys` 上の `tank/zones` ファイルシステムにこれらのゾーンを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 tank
```

次の例では、複数のゾーンのうち 1 つが `sysC` から `sysD` にコピーされます。`ssh` コマンドは使用できませんが、NFS サーバーインスタンスは使用できるものとします。次のコマンドを使用すると、ゾーンが別のゾーンのクローンであるかどうかを気にせずに、`zfs send` 再帰的ストリームを生成できます。

```
sysC# zfs snapshot -r rpool/zones/zone1@send-to-nfs
sysC# zfs send -rc rpool/zones/zone1@send-to-nfs > /net/nfssrv/export/scratch/zone1.zfs
sysD# zfs create tank/zones
sysD# zfs receive -d tank/zones < /net/nfssrv/export/scratch/zone1.zfs
```

ZFS 代替ルートプールを使用する

プールが作成されると、そのプールはデフォルトでホストシステムに関連付けられます。ホストシステムでは、プールに関する情報を管理しているので、プールが使用できなくなったときにそのことを自動的に検出することができます。この情報は、通常の操作では有効な情報ですが、代替メディアから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」プール機能が用意されています。代替ルートプールは、システムのリブート後には有効でなくなり、すべてのマウントポイントはプールのルートへの相対パスに変更されます。

ZFS 代替ルートプールを作成する

代替ルートプールを作成する理由としてもっとも一般的なのは、リムーバブルメディアでの使用です。このような場合には、必要なファイルシステムは通常 1 つだけなので、ターゲットシステムでユーザーが選択した場所にマウントする必要があります。`zpool create -R` オプションを使用して代替ルートプールを作成すると、ルートファイルシステムのマウントポイントは代替ルート値と同じ `/` に自動的に設定されます。

次の例では、`morpheus` という名前のプールが 代替ルートパスとしての `/mnt` に作成されます。

```
# zpool create -R /mnt morpheus c0t0d0
# zfs list morpheus
NAME                    USED  AVAIL  REFER  MOUNTPOINT
morpheus               32.5K  33.5G     8K  /mnt
```

ファイルシステムが 1 つだけで (`morpheus`)、そのマウントポイントがプールの代替ルート `/mnt` であることに注意してください。ディスクに格納されているマウントポイントは、実際に `/` になっています。`/mnt` のフルパスは、プール作成のこの初期コンテキストでのみ解釈されます。その後、このファイルシステムをエクスポートし、それを別のシステム上の任意の代替ルートプールの下で、`-R` *alternate root value* 構文を使ってインポートすることができます。

```
# zpool export morpheus
# zpool import morpheus
cannot mount '/': directory is not empty
# zpool export morpheus
# zpool import -R /mnt morpheus
# zfs list morpheus
NAME                    USED  AVAIL  REFER  MOUNTPOINT
morpheus               32.5K  33.5G     8K  /mnt
```

代替ルートプールをインポートする

代替ルートを使って、プールをインポートすることもできます。この機能は、回復を行う状況で利用できます。つまり、マウントポイントを現在のルートのコンテキストではなく、修復を実行できるように一時的なディレクトリとして解釈するような状況で利用できます。前セクションで説明したように、この機能はリムーバブルメディアをマウントするときにも使用できます。

次の例では、morpheus という名前のプールが代替ルートパスとしての /mnt にインポートされます。この例では、morpheus がすでにエクスポート済みであることを前提としています。

```
# zpool import -R /a pool
# zpool list morpheus
NAME   SIZE   ALLOC  FREE    CAP  HEALTH  ALTROOT
pool  44.8G    78K  44.7G     0%  ONLINE  /a
# zfs list pool
NAME   USED  AVAIL  REFER  MOUNTPOINT
pool  73.5K  44.1G    21K  /a/pool
```

◆ ◆ ◆ 第 10 章

Oracle Solaris ZFS のトラブルシューティングとプールの回復

この章では、ZFS の障害をどのように識別し、そこから回復するかについて説明します。また、障害の発生を防ぐ方法についても説明します。

この章は、次のセクションで構成されます。

ルートプールの完全な復旧については、第 11 章「スナップショットのアーカイブとルートプールの回復」を参照してください。

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。33 ページの「ZFS のディスク領域の計上」も確認してください。

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。

- 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 tank c0t6d0
  # zpool list tank
  NAME   SIZE  ALLOC   FREE    CAP  DEDUP  HEALTH  ALTROOT
  tank   136G  95.5K   136G     0%  1.00x  ONLINE  -
  # zfs list tank
  NAME   USED  AVAIL  REFER  MOUNTPOINT
  tank    72K   134G    21K  /tank
  ```

- 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 および初期 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 tank mirror c0t6d0 c0t7d0
# zpool list tank
NAME   SIZE  ALLOC   FREE    CAP  DEDUP  HEALTH  ALTROOT
tank   136G  95.5K   136G     0%  1.00x  ONLINE  -
# zfs list tank
NAME   USED  AVAIL  REFER  MOUNTPOINT
tank    72K   134G    21K  /tank
```

- **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 tank raidz2 c0t6d0 c0t7d0 c0t8d0
# zpool list tank
NAME   SIZE  ALLOC   FREE    CAP  DEDUP  HEALTH  ALTROOT
tank   408G   286K   408G     0%  1.00x  ONLINE  -
# zfs list tank
NAME   USED  AVAIL  REFER  MOUNTPOINT
tank  73.2K   133G  20.9K  /tank
```

ZFS の障害を識別する

ZFS では、ファイルシステムとボリュームマネージャーが統合されているために、多くの異なる障害が存在します。この章では、さまざまな障害の概要を説明してから、実行しているシステムでそれらをどのように識別するかについて説明します。この章の最後では、問題を修復する方法について説明します。ZFS で発生する可能性がある基本的なエラーには、次の 3 種類があります。

- 294 ページの「ZFS ストレージプール内でデバイスが見つからない」
- 294 ページの「ZFS ストレージプール内のデバイスが損傷している」
- 294 ページの「ZFS データが破壊している」

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。

ZFS ストレージプール内でデバイスが見つからない

デバイスがシステムから完全に削除されると、ZFS はそのデバイスを開けないことを検出し、REMOVED 状態にします。この削除が原因でプール全体が使用できない状態になるかどうかは、そのプールのデータ複製レベルによって決まります。ミラー化されたデバイスまたは RAID-Z デバイスにあるディスクが取り外されても、そのプールには引き続きアクセスできます。プールは UNAVAIL になる可能性があります。これは、次の条件のもとでは、デバイスが再接続されるまでどのデータにもアクセスできないことを意味します:

- ミラーのすべてのコンポーネントが削除される場合
- RAID-Z (raidz1) デバイス内の複数のデバイスが削除される場合
- 単一ディスク構成で最上位レベルのデバイスが削除される場合

ZFS ストレージプール内のデバイスが損傷している

「損傷している」という用語には、発生する可能性のあるさまざまなエラーが含まれます。たとえば、次のようなものがあります。

- ディスクまたはコントローラが不良であるために、一時的な入出力エラーが発生する
- 宇宙線が原因で、ディスク上のデータが破壊される
- ドライバのバグが原因で、間違った場所からデータが転送されたり、間違った場所にデータが転送されたりする
- ユーザーが誤って物理デバイスの一部を上書きしてしまう

これらのエラーは、ある場合には一時的に発生します。たとえば、コントローラに問題があるときは、入出力が無作為にエラーになります。また、ディスク上の破壊のように、損傷が永続することもあります。ただし、損傷が永続的だからといって、そのエラーが再度発生する可能性が高いことには必ずしもなりません。たとえば、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 データが破壊している

データの破壊が発生するのは、1 つ以上のデバイスエラー (1 つ以上のデバイスが見つからないか、損傷している) が最上位レベルの仮想デバイスに影響するときで

す。たとえば、データは破壊されていないけれども、一方のミラーに大量のデバイスエラーが発生する場合があります。もう一方のミラーの正確に同じ場所にエラーが発生した場合は、データが破壊されたことになります。

データの破壊は常に永続的であり、修復時は特に注意する必要があります。配下のデバイスを修復または置き換えても、元のデータは永久に失われています。このような状況では、ほとんどの場合、バックアップからデータを復元する必要があります。データエラーは発生するたびに記録されます。次の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。

ZFS ファイルシステムの整合性をチェックする

fsck に相当するユーティリティーは、ZFS には存在しません。このユーティリティーは従来から、ファイルシステムの修復と検証という 2 つの目的に利用されてきました。

ファイルシステムの修復

従来のファイルシステムのデータ書き込み方法は、本質的に予期しない障害によってファイルシステムの不一致が発生しやすい性質を持っています。従来のファイルシステムはトランザクション方式ではないので、参照されないブロックや不正なリンクカウントなど、ファイルシステム構造の矛盾が発生する可能性があります。ジャーナリングを導入することでこれらの問題のいくつかは解決されますが、ログをロールバックできないときには別の問題が発生する可能性があります。データの不一致が ZFS 構成内のディスク上で発生するとすれば、それはハードウェア障害が発生した場合か、ZFS ソフトウェアにバグが存在する場合だけです。ただし、ハードウェア障害の場合は、プールに冗長性があるはずです。

fsck ユーティリティーは、UFS ファイルシステムに固有の既知の問題を修復します。ZFS ストレージプールの問題の大半は一般に、ハードウェアまたは電源の障害に関連しています。冗長プールを利用することで、多くの問題を回避できます。ハードウェアの障害または電源の停止が原因でプールが損傷している場合は、321 ページの「ZFS ストレージプール全体の損傷を修復する」を参照してください。

プールに冗長性がない場合は、ファイルシステムの破壊によってデータの一部またはすべてにアクセスできなくなるリスクが常に存在します。

ファイルシステムの検証

fsck ユーティリティーには、ファイルシステムの修復を実行する以外に、ディスク上のデータに問題がないことを検証する機能があります。この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。この機能は「スクラブ」と呼ばれ、メモリーやほかのシステム内で、ハードウェアまたはソフトウェア障害が発生する前にエラーを検出および回避する手段として一般的に使用されます。

ZFS データのスクラブを制御する

スクラブを行なっているときまたは必要なファイルにアクセスしているときにエラーが発生した場合には、そのエラーが内部でログに記録されるので、そのプールで認識されているすべてのエラーの概要をすぐに確認できます。

ZFS データの明示的なスクラブ

データの完全性をもっとも簡単にチェックする方法は、プールに含まれるすべてのデータのスクラブを明示的に開始することです。この処理では、プールに含まれるすべてのデータを 1 回たどってみて、すべてのブロックが読み取り可能であることを確認します。スクラブは、デバイスが実現できる最大速度で進行します。ただし、入出力が発生する場合には、その優先順位は通常の操作よりも低くなります。この操作によって、パフォーマンスが低下することがあります。ただし、スクラブの実行中でも、プールのデータはそのまま使用することができ、応答時間もほとんど変わらないはずです。明示的なスクラブを開始するには、zpool scrub コマンドを使用します。次に例を示します。

```
# zpool scrub tank
```

現在のスクラブ操作のステータスは、zpool status コマンドを使用して表示できます。例:

```
# zpool status -v tank
  pool: tank
 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
        tank        ONLINE       0     0     0
```

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

errors: No known data errors
```

一度に実行できるスクラブ操作は、各プールで 1 つだけです。

`-s` オプションを使用すれば、進行中のスクラブ操作を中止できます。例:

```
# zpool scrub -s tank
```

ほとんどの場合、データの完全性を保証するスクラブ操作は、完了するまで続けるようにしてください。操作によってシステム性能に影響が出る場合は、ユーザー自身の判断でスクラブ操作を中止してください。

定期的にスクラブを実行すると、システム上のすべてのディスクへの継続的な入出力が保証されます。定期的なスクラブには、電源管理がアイドル状態のディスクを低電力モードにすることができなくなるという副作用があります。システムによる入出力がほとんど常に実行されている場合や、電力消費を気にする必要がない場合には、この問題は無視しても問題ありません。

`zpool status` の出力の解釈の詳細については、90 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、316 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 の問題を解決する

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。

次の機能を使用して、ZFS 構成で発生した問題を識別することができます。

- `zpool status` コマンドを使用すると、ZFS ストレージプールについての詳細な情報を表示できます。

- プールおよびデバイスの障害が ZFS/FMA の診断メッセージで報告されます。
- zpool history コマンドを使用すると、プール状態の情報を変更した以前の ZFS コマンドを表示できます。

ZFS のほとんどのトラブルシューティングで、zpool status コマンドを使用します。このコマンドを実行すると、システム上のさまざまな障害が分析され、もっとも重大な問題が識別されます。さらに、推奨する処置と、詳細情報が掲載されたナレッジ記事へのリンクが提示されます。プールで複数の問題が発生している可能性がある場合でも、このコマンドで識別できる問題は 1 つだけです。たとえば、データ破壊のエラーは一般に、いずれかのデバイスで障害が発生したことを示唆しますが、障害が発生したデバイスを置き換えても、データ破壊の問題がすべて解決するとは限りません。

また、ZFS 診断エンジンはプールの障害とデバイスの障害を診断し、報告します。これらの障害に関連するチェックサム、入出力、デバイス、およびプールのエラーも報告されます。fmd で報告される ZFS 障害は、コンソールとシステムメッセージファイルに表示されます。ほとんどの fmd メッセージで、zpool status コマンドを実行して詳細な回復の手順を確認することを求めています。

基本的な回復方法は次のとおりです。

- 該当する場合、zpool history コマンドを使って、エラーシナリオに至る前に実行された ZFS コマンドを特定します。例:

  ```
  # zpool history tank
  History for 'tank':
  2010-07-15.12:06:50 zpool create tank mirror c0t1d0 c0t2d0 c0t3d0
  2010-07-15.12:06:58 zfs create tank/eric
  2010-07-15.12:07:01 zfs set checksum=off tank/eric
  ```

  この出力では、tank/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 フラグを指定しないでこのコマンドを実行した場合は、すべてのプールが健全である場合でも、すべてのプール (コマンド行で特定のプールを指定した場合は、要求したプール) のすべてのステータスが表示されます。

zpool status コマンドのコマンド行オプションの詳細については、90 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。

zpool status の出力を確認する

zpool status の完全な出力は次のようになります。

```
# zpool status 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は、次のフィールドで構成されます。一部の項目は、プールに問題がある場合にのみ表示されます。

pool　プールの名前を示します。

state　プールの現在の健全性を示します。この情報は、プールが必要な複製レベルを提供できるかどうかだけを示しています。

status　プールで発生している問題の説明です。エラーが検出されない場合は、このフィールドは省略されます。

action　エラーを修復するために推奨される処置。エラーが検出されない場合は、このフィールドは省略されます。

see　詳細な修復情報が掲載されているナレッジ記事を紹介します。オンラインの記事はこのガイドよりも頻繁に更新されます。そのため、最新の修復手順については常にオンラインの記事を参照してください。エラーが検出されない場合は、このフィールドは省略されます。

scrub　スクラブ操作の現在の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。

プール構成情報

zpool status 出力の config フィールドには、プール内のデバイスの構成、デバイスのステータス、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、307 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、316 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。

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、すべての明示的な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、295 ページの「ZFS ファイルシステムの整合性をチェックする」を参照してください。

データ破壊エラー

zpool status コマンドでは、既知のエラーが発生している場合に、それらがプールに関連するものであるかどうかも出力されます。これらのエラーは、データのスクラブ中または通常の操作中に検出されている可能性があります。ZFS では、プールに関連するすべてのデータエラーの持続的なログを管理しています。システムの完全なスクラブが終了するたびに、このログのローテーションが行われます。

データ破壊エラーは、常に致命的です。このエラーが発生している場合は、プールのデータが破壊されたために、1 つ以上のアプリケーションで入出力エラーが発生したことになります。冗長なプール内でデバイスエラーが発生してもデータは破壊されないので、このログの一部として記録されません。デフォルトでは、検出されたエラーの数だけが表示されます。エラーおよびその詳細の完全なリストは、zpool status -v オプションを使用すれば表示できます。次に例を示します。

```
# zpool status -v tank
  pool: tank
 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
        tank             ONLINE       2     0     0
          c8t0d0         ONLINE       0     0     0
          c8t1d0         ONLINE       2     0     0

errors: Permanent errors have been detected in the following files:

        /tank/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、318 ページの「データ破壊の種類を確認する」を参照してください。

ZFS エラーメッセージのシステムレポート

ZFS では、プール内のエラーを継続的に追跡するだけでなく、そのようなイベントが発生したときに syslog メッセージを表示することもできます。次のような場合に、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。

損傷した ZFS 構成を修復する

ZFS では、アクティブなプールとその構成のキャッシュをルートファイルシステム上で管理しています。このキャッシュファイルが破壊された場合、またはこのファイルがなんらかの形でディスクに保管されている構成情報と同期しなくなった場合には、そのプールを開くことができなくなります。ZFS ではこのような状況を回避しようとしますが、配下のストレージの特性から、なんらかの破壊は常に発生する可能性があります。こうした状況になると、ほかの方法で使用できるとしても、通常はプールがシステムに表示されなくなります。また、この状況から構成が不完全であること、つまり、最上位レベルの仮想デバイスが見つからない (その数は不明) ことがわかる場合もあります。どちらの場合でも、なんらかの方法でプールを見ることができる場合には、プールをエクスポートして再度インポートする方法で、構成を回復することができます。

プールのインポートとエクスポートについては、103 ページの「ZFS ストレージプールを移行する」を参照してください。

見つからないデバイスに関する問題を解決する

デバイスを開けない場合には、zpool status の出力に UNAVAIL 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。UNAVAIL のデバイスに障害が発生しているため、それを交換する必要がある場合は、310 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 – 134 ページの「ミラー化された ZFS ルートプールの代替ディスクからブートする」
- ルートプール内でディスクを交換する
  - 124 ページの「ZFS ルートプールのディスクを交換する方法 (SPARC または x86/VTOC)」
  - 126 ページの「ZFS ルートプールのディスクを交換する方法 (SPARC または x86/EFI (GPT))」
- ルートプール障害の完全な復旧 – 第 11 章「スナップショットのアーカイブとルートプールの回復」

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
PLATFORM: ORCL,SPARC-T3-4, CSN: 1120BDRCCD, HOSTNAME: tardis
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:

`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 tank
pool 'tank' is healthy
```

デバイスを物理的に再接続する

見つからないデバイスを再接続するための正確な手順は、そのデバイスごとに異なります。デバイスがネットワークに接続されているドライブの場合は、ネットワークへの接続を復元するべきです。デバイスが USB デバイスなどのリムーバブルメディアである場合は、システムに再接続するべきです。デバイスがローカルディスクである場合は、コントローラに障害が発生していたために、デバイスがシステムから見えない状態になっていた可能性があります。この場合は、コントローラを置き換えれば、ディスクが再び使用できる状態になるはずです。ハードウェアの種類と構成によっては、ほかの問題が存在する可能性もあります。ドライブに障害が発生してシステムから認識されなくなった場合には、デバイスが損傷していると見なすべきです。307 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。次に例を示します。

```
# zpool online tank c0t1d0
```

デバイスをオンラインする方法の詳細については、77 ページの「デバイスをオンラインにする」を参照してください。

破損したデバイスを交換または修復する

このセクションでは、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

「損傷したデバイス」という用語は定義があいまいですが、発生する可能性のあるいくつかの状況はこの用語で説明できます。

- ビットの腐敗 – 時間の経過とともに、磁力の影響や宇宙線などのさまざまなことが原因で、ディスクに格納されているビットが反転してしまうことがあります。このようなことはあまり発生しませんが、発生した場合には、大規模なまたは長期間稼働するシステムでデータが破壊する可能性は十分にあります。

- 間違った方向への読み取りまたは書き込み – ファームウェアのバグまたはハードウェア障害のために、ブロック全体の読み取りまたは書き込みで、ディスク上の不正な場所を参照してしまうことがあります。これらのエラーは通常、一時的です。ただし、エラーの数が多い場合には、ドライブの障害が発生している可能性があります。
- 管理者エラー – 管理者が意図せずにディスクの一部を不正なデータで上書きする (ディスクの一部に `/dev/zero` をコピーするなど) ことで、ディスクが永続的に破壊されてしまう場合があります。これらのエラーは常に一時的です。
- 一時的な機能停止– ディスクが一定期間使用できなくなり、入出力に失敗することがあります。この状況は通常、ネットワークに接続されたデバイスに発生しますが、ローカルディスクでも一時的に機能が停止することがあります。これらのエラーは、一時的な場合と、そうでない場合があります。
- 不良または信頼性の低いハードウェア – この状況は、ハードウェアの障害によって引き起こされるさまざまな問題の総称です。問題の例としては、断続的な入出力エラー、不規則な破壊を引き起こす転送エラー、その他のさまざまな障害があります。これらのエラーは通常永続的です。
- オフラインのデバイス – デバイスがオフラインである場合は、そのデバイスに障害が発生していると判断した管理者がデバイスをこの状態にしたと推定されます。管理者は、デバイスをこの状態にしたうえで、この推定が正しいかどうかを判断できます。

デバイスのどこに問題があるかを正確に判断することは、難しい作業です。最初に行うことは、`zpool status` 出力のエラー数を調べることです。例:

```
# zpool status -v tank
  pool: tank
 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
        tank             ONLINE       2     0     0
          c8t0d0         ONLINE       0     0     0
          c8t0d0         ONLINE       2     0     0

errors: Permanent errors have been detected in the following files:

        /tank/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、317 ページの「損傷したデータを修復する」を参照してください。

一時的なエラーを解消する

デバイスエラーが一時的と考えられる場合、つまりデバイスの今後の健全性に影響しないと考えられる場合は、デバイスエラーを安全に解消することで、致命的なエラーが発生していないと示すことができます。RAID-Z デバイスまたはミラーデバイスのエラー数をクリアするには、zpool clear コマンドを使用します。次に例を示します。

```
# zpool clear tank c1t1d0
```

この構文を実行すると、すべてのデバイスエラーと、デバイスに関連付けられたすべてのデータエラー数がクリアされます。

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 tank
```

プールエラーのクリアの詳細については、78 ページの「ストレージプールデバイスのエラーをクリアーする」を参照してください。

ZFS ストレージプール内のデバイスを置き換える

デバイスの損傷が永続的である場合、または永続的な損傷が今後発生する可能性がある場合には、そのデバイスを置き換える必要があります。デバイスを置き換えられるかどうかは、構成によって異なります。

デバイスを置き換えられるかどうかを確認する

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、損傷を受けたデバイス上の壊れたデータは、正常なデータが格納されている複製が必要な数だけ存在しない場合には、新しいデバイスにコピーされます。

次の構成で、c1t1d0 ディスクは置き換えることができます。プール内のすべてのデータは正常な複製 c1t0d0 からコピーされます。

```
    mirror            DEGRADED
    c1t0d0             ONLINE
    c1t1d0             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、321 ページの「ZFS ストレージプール全体の損傷を修復する」を参照してください。

ZFS ストレージプール内のデバイスを置き換える

置き換えられるデバイスであることを確認したあとで、zpool replace コマンドを使ってデバイスを置き換えます。損傷したデバイスを別のデバイスに置き換える場合は、次のような構文を使用します。

```
# zpool replace tank c1t1d0 c2t0d0
```

このコマンドを実行すると、損傷したデバイスまたはプール内のほかのデバイス (冗長な構成の場合) から新しいデバイスにデータが移行されます。コマンドが完了すると、損傷したデバイスが構成から切り離され、そのデバイスをシステムから取り外せる状態になります。1 つの場所ですでにデバイスを取り外して新しいデバイスに置き換えている場合には、1 つのデバイス形式のコマンドを使用します。例:

```
# zpool replace tank c1t1d0
```

このコマンドにフォーマットされていないディスクを指定すると、そのディスクが適切な状態にフォーマットされたのち、残りの構成からデータが再同期化されます。

zpool replace コマンドの詳細については、78 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 10–1　ZFS ストレージプール内の SATA ディスクを置き換える

次の例では、システムのミラー化ストレージプール tank のデバイス (c1t3d0) を SATA デバイスに置き換える方法を示します。ディスク c1t3d0 を同じ位置 (c1t3d0) で新しいディスクに置き換えるには、ディスクを置き換える前に構成解除する必要があります。置き換えられるディスクが SATA ディスクでない場合は、78 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 – あらかじめプールの autoreplace プロパティーをオンに設定してあった場合、そのプールに以前属していたデバイスと物理的に同じ位置に新しいデバイスが検出されると、そのデバイスは自動的にフォーマットされ、zpool replace コマンドを使用せずに置き換えられます。ハードウェアによっては、この機能はサポートされない場合があります。

- 障害の発生したディスクがホットスペアに自動的に置き換えられる場合は、障害の発生したディスクが置き換えられたあとでホットスペアの切り離しが必要になることがあります。たとえば、障害の発生したディスクが置き換えられたあとも c2t4d0 がアクティブなホットスペアになっている場合は、切り離してください。

```
# zpool detach tank c2t4d0
```

- 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 tank c1t3d0
# cfgadm | grep c1t3d0
```

例 10–1 ZFS ストレージプール内の SATA ディスクを置き換える *(続き)*

```
sata1/3::dsk/c1t3d0            disk         connected    configured   ok
# cfgadm -c unconfigure sata1/3
Unconfigure the device at: /devices/pci@0,0/pci1022,7458@2/pci11ab,11ab@1:3
This operation will suspend activity on the SATA device
Continue (yes/no)? yes
# cfgadm | grep sata1/3
sata1/3                        disk         connected    unconfigured ok
<Physically replace the failed disk c1t3d0>
# cfgadm -c configure sata1/3
# cfgadm | grep sata1/3
sata1/3::dsk/c1t3d0            disk         connected    configured   ok
# zpool online tank c1t3d0
# zpool replace tank c1t3d0
# zpool status tank
  pool: tank
 state: ONLINE
 scrub: resilver completed after 0h0m with 0 errors on Tue Feb  2 13:17:32 2010
config:

        NAME        STATE     READ WRITE CKSUM
        tank        ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c0t1d0  ONLINE       0     0     0
            c1t1d0  ONLINE       0     0     0
          mirror-1  ONLINE       0     0     0
            c0t2d0  ONLINE       0     0     0
            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 tank c1t3d0 c4t3d0
# zpool status
  pool: tank
 state: DEGRADED
 scrub: resilver completed after 0h0m with 0 errors on Tue Feb  2 13:35:41 2010
config:
```

例 10–1 ZFS ストレージプール内の SATA ディスクを置き換える (続き)

```
        NAME             STATE     READ WRITE CKSUM
        tank             DEGRADED     0     0     0
          mirror-0       ONLINE       0     0     0
            c0t1d0       ONLINE       0     0     0
            c1t1d0       ONLINE       0     0     0
          mirror-1       ONLINE       0     0     0
            c0t2d0       ONLINE       0     0     0
            c1t2d0       ONLINE       0     0     0
          mirror-2       DEGRADED     0     0     0
            c0t3d0       ONLINE       0     0     0
            replacing    DEGRADED     0     0     0
              c1t3d0     OFFLINE      0     0     0
              c4t3d0     ONLINE       0     0     0

errors: No known data errors
```

ディスクの置き換えが完了するまでに zpool status コマンドを数回実行する必要がある場合があります。

```
# zpool status tank
  pool: tank
 state: ONLINE
 scrub: resilver completed after 0h0m with 0 errors on Tue Feb  2 13:35:41 2010
config:

        NAME           STATE     READ WRITE CKSUM
        tank           ONLINE       0     0     0
          mirror-0     ONLINE       0     0     0
            c0t1d0     ONLINE       0     0     0
            c1t1d0     ONLINE       0     0     0
          mirror-1     ONLINE       0     0     0
            c0t2d0     ONLINE       0     0     0
            c1t2d0     ONLINE       0     0     0
          mirror-2     ONLINE       0     0     0
            c0t3d0     ONLINE       0     0     0
            c4t3d0     ONLINE       0     0     0
```

例 10–2 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。

次の例では、ストレージプール pool で障害が発生したログデバイス c0t5d0 を回復する方法を示します。基本的な手順は次のとおりです。

- zpool status -x の出力と FMA 診断メッセージを確認します (次のサイトの説明を参照)。

  https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&doctype=REFERENCE&alias=EVENT:ZFS-8000-K4

例 10–2 障害が発生したログデバイスを交換する (続き)

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

        NAME          STATE     READ WRITE CKSUM
        pool          FAULTED      0     0     0 bad intent log
          mirror-0    ONLINE       0     0     0
            c0t1d0    ONLINE       0     0     0
            c0t4d0    ONLINE       0     0     0
        logs          FAULTED      0     0     0 bad intent log
          c0t5d0      UNAVAIL      0     0     0 cannot open
<Physically replace the failed log device>
# zpool online pool c0t5d0
# zpool clear pool
# 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 pool c0t5d0
  ```
- 障害が発生したログデバイスのエラー状況がリセットされます。

  ```
  # zpool clear pool
  ```

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

例 10–2 障害が発生したログデバイスを交換する (続き)

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 out of 16.2G at 18.5M/s, 0h2m to go
   13.3G resilvered, 82.34% done
```

- 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。500G バイトのディスクを置き換えるときでも、プールで使用されているディスク容量が数 G バイトであれば、数秒で完了できます。
- 再同期化は、割り込み可能で安全です。システムの電源が切れるか、またはシステムが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 tank
  pool: tank
 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 scanned out of 222M at 5.46M/s, 0h0m to go
    54.5M resilvered, 24.64% done
config:

        NAME             STATE     READ WRITE CKSUM
        tank             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 tank
  pool: tank
 state: ONLINE
 scrub: resilver completed after 0h1m with 0 errors on Tue Feb  2 13:54:30 2010
config:

        NAME        STATE     READ WRITE CKSUM
        tank        ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c2t0d0  ONLINE       0     0     0  377M resilvered
            c1t1d0  ONLINE       0     0     0

errors: No known data errors
```

プールは `ONLINE` に戻り、元の障害が発生したディスク (`c1t0d0`) は構成から削除されています。

損傷したデータを修復する

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。

- 318 ページの「データ破壊の種類を確認する」
- 319 ページの「破壊されたファイルまたはディレクトリを修復する」

- 321 ページの「ZFS ストレージプール全体の損傷を修復する」

ZFS では、データ破壊のリスクを最小限に抑えるために、チェックサム、冗長性、および自己修復データが使用されます。それでも、プールが冗長でない場合、プールの機能が低下しているときに破壊が発生した場合、または予期しないことが一度に起こってデータの複数のコピーが破壊された場合は、データの破壊が発生することがあります。どのような原因であったとしても、結果は同じです。 データが破壊され、その結果アクセスできなくなっています。対処方法は、破壊されたデータの種類とその相対的な価値により異なります。破壊されるデータは、大きく 2 つの種類に分けられます。

- プールメタデータ – ZFS では、プールを開いてデータセットにアクセスするために、一定量のデータを解析する必要があります。これらのデータが破壊された場合には、プール全体またはデータセット階層の一部が使用できなくなります。
- オブジェクトデータ – この場合、破壊は特定のファイルまたはディレクトリに限定されます。この問題が発生すると、そのファイルまたはディレクトリの一部がアクセスできなくなる可能性があります。この問題が原因で、オブジェクトも一緒に破壊されることがあります。

データの検証は、通常の操作中およびスクラブ時に行われます。プールデータの完全性を検証する方法については、295 ページの「ZFS ファイルシステムの整合性をチェックする」を参照してください。

データ破壊の種類を確認する

デフォルトでは、`zpool status` コマンドでは、破壊が発生したことだけが報告され、破壊が発生した場所は報告されません。次に例を示します。

```
# zpool status tank
  pool: tank
 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
        tank                        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 tank -v
  pool: tank
 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
        tank                        ONLINE       4     0     0
          c0t5000C500335E106Bd0     ONLINE       0     0     0
          c0t5000C500335FC3E7d0     ONLINE       4     0     0

errors: Permanent errors have been detected in the following files:
        /tank/file.1
        /tank/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:

  ```
  /monkey/a.txt
  ```

- ファイルへの完全なパスは見つかったが、データセットがマウントされていない場合は、前にスラッシュ (/) が付かず、後ろにファイルへのデータセット内のパスが付いたデータセット名が表示されます。例:

  ```
  monkey/ghost/e.txt
  ```

- エラーにより、または dnode_t の場合のようにオブジェクトに実際のファイルパスが関連付けられていないことにより、ファイルパスに対するオブジェクト番号を正常に変換できない場合は、後ろにオブジェクト番号の付いたデータセット名が表示されます。例:

  ```
  monkey/dnode:<0x0>
  ```

- メタオブジェクトセット (MOS) のオブジェクトが破壊された場合は、後ろにオブジェクト番号の付いた <metadata> という特別なタグが表示されます。

ディレクトリまたはファイルのメタデータの中で破壊は発生している場合には、そのファイルを別の場所に移動するしかありません。任意のファイルまたはディレクトリを不便な場所に安全に移動することができ、そこで元のオブジェクトを復元することができます。

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から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。ZFS スクラブアルゴリズムはプールを横断して、データの各ブロックに一度のみアクセスします。これは、最初に検出した破壊のみを報告できます。そのため、影響を受けたファイルへの単一のパスのみが生成されます。これは、複製解除された、破壊されたブロックにも適用されます。

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、破壊された追加のパスを検索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: tpool
   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
           tpool      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 tpool
  ```

  損傷したストレージプールをインポートしようとすると、次のようなメッセージが表示されます。

  ```
  # zpool import tpool
  cannot import 'tpool': I/O error
          Recovery is possible, but will result in some data loss.
  ```

```
        Returning the pool to its state as of 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 tpool'.  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 tpool
Pool tpool 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 tpool
  ```

  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、109 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、108 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、データセットのレイアウトやローカルに設定されたさまざまなプロパティーが記述されているファイルを別の安全な場所に保存します。これは、プールがアクセスできない状態になった場合に、これらの情報にアクセスできなくなるためです。プールを破棄したあとに、プールの構成とデータセットのレイアウトを使用して、完全な構成を再構築できます。次に、なんらかのバックアップまたは採用している復元方法を使って、データを生成することができます。

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、起動時にどのプールも探さないように ZFS を設定する必要があります。

ZFS では、利用できるプールとその構成の内部キャッシュを `/etc/zfs/zpool.cache` で管理しています。このファイルの場所と内容は非公開で、変更される可能性があります。システムを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 11 章

11 スナップショットのアーカイブとルートプールの回復

この章では、システムの障害時に Oracle Solaris 11 を移行または回復するために使用できるスナップショットをアーカイブする方法について説明します。これらの手順を使用して、基本的な障害回復計画の中核を作成できます。また、システムの構成を新しいブートデバイスに移行するために使用することもできます。

この章は、次のセクションで構成されます。

ZFS 回復プロセスの概要

少なくとも、システム障害によるダウンタイムを短縮するために、すべてのファイルシステムデータを定期的にバックアップする必要があります。破滅的なシステム障害が起きた場合、OS を再インストールしてシステム構成を再作成するのではなく、ZFS ルートプールスナップショットを回復できます。続いて、非ルートプールデータを回復します。

Oracle Solaris 11 を稼働しているシステムが、バックアップおよびアーカイブの対象候補になります。全体のプロセスには次の手順が含まれます。

- ルートプールファイルシステムの ZFS スナップショットアーカイブと、移行または回復する必要のある非ルートプールの ZFS スナップショットアーカイブを作成します。

  OS を更新したあと、ルートプールのスナップショットを再アーカイブしてください。
- USB ドライブなどのローカルリムーバブルメディアにスナップショットアーカイブを保存するか、リモートシステムにスナップショットを送信して、回復する場合に備えます。

- 障害の発生したディスクやほかのシステムコンポーネントを取り替えます。
- ターゲットシステムが Oracle Solaris 11 インストールメディアからブートし、新しいストレージプールが作成され、ファイルシステムが回復されます。
- 最小のブート構成を実行すると、システムが使用可能になり、アーカイブ時に実行していたすべてのサービスを提供します。

ZFS プールの回復要件

- アーカイブされたシステムと回復システムは、同じアーキテクチャーを使用している必要があり、Oracle Solaris 11 のサポート対象プラットフォームの最低要件を満たしている必要があります。
- 新しい ZFS ストレージプールを含む交換用ディスクは、少なくとも、アーカイブされたプールで使用していたデータと容量が同じである必要があります。

  Oracle Solaris 11 では、ルートプールディスクに SMI (VTOC) ラベルが付いている必要があります。Oracle Solaris 11.1 では、x86 ベースシステムのルートプールディスクに SMI (VTOC) と EFI (GPT) ディスクラベルのどちらかを付けることができます。EFI (GPT) ラベル付きディスクのブートサポートについては、17 ページの「EFI (GPT) ラベル付きディスクのブートサポート」を参照してください。
- ルートアクセスは、アーカイブされたスナップショットを含んだシステムと回復システムの両方で必要になります。ssh を使用してリモートシステムにアクセスしている場合、特権アクセスを許可するようにこのコマンドを構成する必要があります。

回復用の ZFS スナップショットアーカイブを作成する

ZFS ルートプールスナップショットを作成する前に、次の情報を保存することを検討してください。

- ルートプールのプロパティーを取得します。

```
sysA# zpool get all rpool
```

- ルートプールディスクのサイズおよび現在の容量を特定します。

```
sysA# zpool list
NAME   SIZE  ALLOC   FREE  CAP  DEDUP  HEALTH  ALTROOT
rpool   74G  5.42G  68.6G   7%  1.00x  ONLINE  -
```

- ルートプールのコンポーネントを特定します。

```
sysA# zfs list -r rpool
NAME                      USED  AVAIL  REFER  MOUNTPOINT
rpool                    13.8G  53.1G  73.5K  /rpool
rpool/ROOT               3.54G  53.1G    31K  legacy
rpool/ROOT/solaris       3.54G  53.1G  3.37G  /
rpool/ROOT/solaris/var    165M  53.1G   163M  /var
```

```
rpool/VARSHARE           37.5K  53.1G  37.5K  /var/share
rpool/dump               8.19G  53.4G  7.94G  -
rpool/export               63K  53.1G    32K  /export
rpool/export/home          31K  53.1G    31K  /export/home
rpool/swap               2.06G  53.2G  2.00G  -
```

▼ ZFS スナップショットアーカイブを作成する方法

次の手順では、ルートプールのすべてのファイルシステムを含むルートプールの再帰的なスナップショットを作成する方法について説明します。その他の非ルートプールも同じ方法でアーカイブできます。

次の点を考慮してください。

- 完全なシステム回復の場合、スナップショットをリモートシステム上のプールに送信します。
- リモートシステムから NFS 共有を作成し、また必要に応じて、特権アクセスを許可するように ssh を構成します。
- 再帰的なルートプールスナップショットは、1 つの大きなスナップショットファイルとしてリモートシステムに送信されますが、再帰的なスナップショットを送信して、リモートシステムに個々のスナップショットとして格納することもできます。

以降の手順では、再帰的なスナップショットには rpool@snap1 という名前が付けられています。 回復対象のローカルシステムは sysA であり、リモートシステムは sysB です。rpool はデフォルトのルートプール名であり、使用しているシステムでは異なる場合があります。

1 管理者になります。

2 ルートプールの再帰的なスナップショットを作成します。

```
sysA# zfs snapshot -r rpool@rpool.snap1
```

3 必要に応じて、スワップスナップショットおよびダンプスナップショットを削除することにより、スナップショットアーカイブを削減します。

```
sysA# zfs destroy rpool/dump@rpool.snap1
sysA# zfs destroy rpool/swap@rpool.snap1
```

スワップボリュームには、システムの移行または回復に関連していないデータは含まれません。クラッシュダンプを保持する場合は、ダンプボリュームのスナップショットは削除しないでください。

4 ルートプールの再帰的なスナップショットを、別のシステム上の別のプールに送信します。

a. 1つ以上のスナップショットを受信するためにリモートシステム上のファイルシステムを共有します。

次の手順では、/tank/snaps ファイルシステムが、ルートの再帰的なスナップショットを格納するために共有されています。

```
sysB# zfs set share.nfs=on tank/snaps
sysB# zfs set share.nfs.sec.default.root=sysA tank/snaps
```

b. リモートシステムにルートプールの再帰的なスナップショットを送信します。

前の手順で共有したリモートファイルシステムに、再帰的なスナップショットを送信します。

```
sysA# zfs send -Rv rpool@rpool.snap1 | gzip > /net/sysB/tank/snaps/
rpool.snap1.gz
sending from @ to rpool@rpool.snap1
sending from @ to rpool/VARSHARE@rpool.snap1
sending from @ to rpool/export@rpool.snap1
sending from @ to rpool/export/home@rpool.snap1
sending from @ to rpool/ROOT@rpool.snap1
sending from @ to rpool/ROOT/solaris@install
sending from @install to rpool/ROOT/solaris@rpool.snap1
sending from @ to rpool/ROOT/solaris/var@install
sending from @install to rpool/ROOT/solaris/var@rpool.snap1
```

ルートプールの再作成とルートプールのスナップショットの回復

ルートプールを再作成し、ルートプールのスナップショットを回復する必要がある場合、一般的な手順は次のようになります。

- 1つ以上の交換用のルートプールディスクを用意し、ルートプールを再作成します。
- ルートプールファイルシステムスナップショットを復元します。
- 必要なブート環境を選択しアクティブにします。
- システムをブートします。

▼ 回復システムでのルートプールの再作成方法

ルートプールを回復するときには、次の考慮事項を確認してください。

- 冗長でないルートプールディスクに障害が発生した場合、インストールメディアまたはインストールサーバーからシステムをブートして、OS を再インストールするか、以前にアーカイブしたルートプールのスナップショットを復元する必要があります。

システムのディスクを交換する方法については、ハードウェアのドキュメントを参照してください。

- ミラー化ルートプールディスクに障害が発生した場合、まだシステムが作動している間に、障害の発生したディスクを交換できます。ミラー化ルートプールで障害の発生したディスクを交換する方法については、124 ページの「ZFS ルートプールのディスクを交換する方法 (SPARC または x86/VTOC)」を参照してください。

1 障害の発生したルートプールディスクまたはシステムコンポーネントを特定して交換します。

このディスクは通常、デフォルトのブートデバイスです。または、別のディスクを選択して、デフォルトのブートデバイスをリセットすることもできます。

2 次のいずれかの方法を選択して、**Oracle Solaris 11** インストールメディアからシステムをブートします。

- DVD または USB インストールメディア (SPARC または x86) – メディアを挿入し、ブートデバイスとして該当するデバイスを選択します。

  テキストベースのメディアを使用する場合、インストーラメニューからシェルオプションを選択します。
- Live Media (x86 のみ) - 回復手順中に GNOME デスクトップセッションを使用できます。
- 自動インストーラまたは AI メディアのローカルコピー (SPARC または x86) – テキストインストーラメニューから、シェルオプションを選択します。SPARC システムで、AI メディアをブートし (ローカルまたはネットワークを通じて)、シェルオプションを選択します。

```
ok boot net:dhcp
.
.
.Welcome to the Oracle Solaris 11 installation menu

        1  Install Oracle Solaris
        2  Install Additional Drivers
        3  Shell
        4  Terminal type (currently xterm)
        5  Reboot

Please enter a number [1]: 3
```

3 **SPARC** または **x86 (VTOC):** ルートプールディスクを用意します。

a. 交換用のルートプールディスクが **format** ユーティリティーから認識されていることを確認します。

```
# format
Searching for disks...done
AVAILABLE DISK SELECTIONS:
```

```
       0. c2t0d0 <FUJITSU-MAY2073RCSUN72G-0401 cyl 14087 alt 2 hd 24 sec 424>
          /pci@780/pci@0/pci@9/scsi@0/sd@0,0
       1. c2t1d0 <FUJITSU-MAY2073RCSUN72G-0401 cyl 14087 alt 2 hd 24 sec 424>
          /pci@780/pci@0/pci@9/scsi@0/sd@1,0
       2. c2t2d0 <SEAGATE-ST973402SSUN72G-0400-68.37GB>
          /pci@780/pci@0/pci@9/scsi@0/sd@2,0
       3. c2t3d0 <SEAGATE-ST973401LSUN72G-0556-68.37GB>
          /pci@780/pci@0/pci@9/scsi@0/sd@3,0
Specify disk (enter its number): 0
```

b. SPARC または x86 (VTOC): ルートプールディスクに SMI (VTOC) ラベルとディスク容量の大部分を占めるスライス 0 があることを確認します。

パーティションテーブルを調べ、ルートプールディスクに SMI ラベルとスライス 0 があることを確認します。

```
selecting c2t0d0
[disk formatted]
format> partition
partition> print
```

c. SPARC または x86 (VTOC): 必要に応じて、SMI (VTOC) ラベルを持つディスクのラベルを変更します。

次のショートカットコマンドを使用してディスクのラベルを変更します。これらのコマンドはエラーチェックを備えていないため、正しいディスクのラベルが変更されていることを確認してください。

- SPARC:

  ```
  sysA# format -L vtoc -d c2t0d0
  ```

  スライス 0 にディスク容量が適切に割り当てられていることを確認してください。上記のコマンドではデフォルトのパーティションが適用されますが、これは、ルートプールスライス 0 には小さ過ぎる可能性があります。デフォルトのパーティションテーブルの変更については、『Oracle Solaris 11.1 の管理: デバイスとファイルシステム』の「ZFS ルートプールディスク (EFI (GPT)) の交換方法」を参照してください。

- x86:

  ```
  sysA# fdisk -B /dev/rdsk/c2t0d0p0
  sysA# format -L vtoc -d c2t0d0
  ```

  スライス 0 にディスク容量が適切に割り当てられていることを確認してください。上記のコマンドではデフォルトのパーティションが適用されますが、これは、ルートプールスライス 0 には小さ過ぎる可能性があります。デフォルトのパーティションテーブルの変更については、『Oracle Solaris 11.1 の管理: デバイスとファイルシステム』の「ZFS ルートプールディスク (EFI (GPT)) の交換方法」を参照してください。

4 ルートプールを再作成します。

SPARC または x86 (VTOC) システム:

```
sysA# zpool create rpool c2t0d0s0
```

EFI (GPT) ラベル付きルートプールディスクを備えた x86 ベースシステムでは、次のような構文を使用します。

```
sysA# zpool create -B rpool c2t0d0
```

5 リモートシステムのスナップショットを含むファイルシステムをマウントします。

```
sysA# mount -F nfs sysB:/tank/snaps /mnt
```

6 ルートプールのスナップショットを復元します。

```
sysA# gzcat /mnt/rpool.snap1.qz | zfs receive -Fv rpool
receiving full stream of rpool@rpool.snap1 into rpool@rpool.snap1
received 92.7KB stream in 1 seconds (92.7KB/sec)
receiving full stream of rpool/export@rpool.snap1 into rpool/export@rpool.snap1
received 47.9KB stream in 1 seconds (47.9KB/sec)
.
.
.
```

7 必要に応じて、スワップデバイスとダンプデバイスを再作成します。

例:

```
sysA# zfs create -V 4G rpool/swap
sysA# zfs create -V 4G rpool/dump
```

スワップボリュームやダンプボリュームのサイジングについては、『Oracle Solaris 11.1 の管理: デバイスとファイルシステム』の「スワップ空間の計画」を参照してください。

8 BE をマウントします。

次の手順では、ブートブロックをインストールできるように、BE をマウントすることが必要になります。

```
sysA# beadm mount solaris /tmp/mnt
```

9 新しいディスクにブートブロックをインストールします。

SPARC または x86 ベースシステムで次のコマンドを使用します。

```
sysA# bootadm install-bootloader -P rpool
```

10 同じデバイスを使用しない場合や、元のシステムではデバイスが異なる方法で構成されている場合は、既存のデバイス情報をクリアします。次に、新しいデバイス情報を再構成するようにシステムに指示します。

```
# devfsadm -Cn -r /tmp/mnt
# touch /tmp/mnt/reconfigure
```

11 BE をアンマウントします。

```
#beadm unmount solaris
```

12 必要に応じて、ブート環境をアクティブにします。

例:

```
sysA# beadm list
BE        Active Mountpoint Space  Policy Created
--        ------ ---------- -----  ------ -------
solaris-1 -      -          46.95M static 2012-07-20 10:25
solaris   -      -           3.83G static 2012-07-19 13:44
# beadm activate solaris
```

13 交換用ルートプールディスクから正常にブートできることを確認します。

必要に応じて、デフォルトのブートデバイスをリセットします。

- SPARC: システムが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。そのためには、`eeprom` コマンドまたはブート PROM の `setenv` コマンドを使用します。
- x86: システム BIOS を再構成します。

◆ ◆ ◆ 第 12 章

推奨の Oracle Solaris ZFS 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。

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 10 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 Solaris リリースおよびパッチ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
```

- 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。

  337 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 tank c4t1d0 c4t3d0
    'tank' 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

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 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
  ```

- **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)、本稼働システムのルートプールを 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。

非ルート (データ) 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* 識別子は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 tank 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。

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

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> は仮想マシンの名前です
  - <type> はコントローラの種類で、piix3ide (通常の IDE 仮想コントローラを使用している場合) と ahci (SATA コントローラを使用している場合) のどちらかになります
  - <n> はディスク番号です

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

- 最適なパフォーマンスには、プール容量を 90% 以下に保ちます
- 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
  - 冗長プールの場合は、`zpool status` および `fmdump` を使用して、週単位でプールをモニターします
  - 冗長でないプールの場合は、`zpool status` および `fmdump` を使用して、2 週間に 1 度プールをモニターします
- `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、2 週間に 1 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.space.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。

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、多数の子孫の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、『Oracle Solaris 11.1 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。

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、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領域をモニターします。

  詳細は、291 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 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

Solaris の各リリースで利用可能な特定の ZFS バージョンを使用することにより、プールやファイルシステムに関する新しい ZFS の機能が導入され、利用できるようになります。`zpool upgrade` または `zfs upgrade` を使用すると、プールまたはファイルシステムのバージョンが、現在実行中の Solaris リリースで提供されるバージョンよりも古いかどうかを識別できます。これらのコマンドを使用して、プールまたはファイルシステムのバージョンをアップグレードすることもできます。

`zpool upgrade` および `zfs upgrade` コマンドの使用方法については、212 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112 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。

バージョン	Oracle Solaris 11	説明
1	snv_36	初期バージョンの ZFS
2	snv_38	Ditto ブロック (複製されたメタデータ)
3	snv_42	ホットスペアおよびダブルパリティー RAID-Z
4	snv_62	`zpool history`
5	snv_62	`gzip` 圧縮アルゴリズム
6	snv_62	`bootfs` プールプロパティー
7	snv_68	別のインテントログデバイス
8	snv_69	委任管理
9	snv_77	`refquota` および `refreservation` プロパティー
10	snv_78	キャッシュデバイス
11	snv_94	スクラブパフォーマンスの向上
12	snv_96	スナップショットプロパティー
13	snv_98	`snapused` プロパティー
14	snv_103	`aclinherit passthrough-x` プロパティー
15	snv_114	ユーザーおよびグループの領域の計上
16	snv_116	`stmf` プロパティー
17	snv_120	トリプルパリティー RAID-Z
18	snv_121	ユーザーによるスナップショットの保持
19	snv_125	ログデバイスの削除
20	snv_128	`zle` (長さゼロのエンコード) 圧縮アルゴリズム
21	snv_128	複製解除
22	snv_128	受信プロパティー
23	snv_135	スリム ZIL
24	snv_137	システム属性

バージョン	Oracle Solaris 11	説明
25	snv_140	スクラブ統計の向上
26	snv_141	スナップショット削除パフォーマンスの向上
27	snv_145	スナップショット作成パフォーマンスの向上
28	snv_147	複数 vdev の交換
29	snv_148	RAID-Z/ミラーハイブリッドアロケータ
30	snv_149	暗号化
31	snv_150	「zfs list」パフォーマンスの向上
32	snv_151	1M バイトのブロックサイズ
33	snv_163	共有サポートの向上
34	S11.1	継承による共有

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、特定のプールバージョンが必要です。

バージョン	Oracle Solaris 11	説明
1	snv_36	初期バージョンの ZFS ファイルシステム
2	snv_69	拡張されたディレクトリエントリ
3	snv_77	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	snv_114	`userquota` および `groupquota` プロパティー
5	snv_137	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

索引

S

T

U

V

X

Z

り

ろ

わ