在 Oracle® Solaris 11.2 中管理 ZFS 文件系统

文件号码 E53918-02
2014 年 12 月

ORACLE

目录

使用本文档

- **概述** – 说明如何设置和管理 ZFS 文件系统。
- **目标读者** – 系统管理员。
- **必备知识** – 具备基本的 Oracle Solaris 或 UNIX 系统管理经验和常规文件系统管理经验。

产品文档库

有关本产品的最新信息和已知问题均包含在文档库中，网址为：http://www.oracle.com/pls/topic/lookup?ctx=solaris11。

获得 Oracle 支持

Oracle 客户可通过 My Oracle Support 获得电子支持。有关信息，请访问 http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info；如果您听力受损，请访问 http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs。

反馈

可以在 http://www.oracle.com/goto/docfeedback 上提供有关此文档的反馈。

♦ ♦ ♦ 第 1 章

1 Oracle Solaris ZFS 文件系统（介绍）

本章概述了 Oracle Solaris ZFS 文件系统及其功能和优点。本章还介绍了在本书所有其余部分中使用的一些基本术语。

本章包含以下各节：

Oracle Solaris 新增的 ZFS 功能

当前的 Oracle Solaris 发行版在 ZFS 文件系统中引入了以下 ZFS 功能。

- 使用临时池名称

  在共享存储或恢复场景中，您可以使用一个临时池名称创建或导入池。有关详细信息，请参见“使用临时名称导入池” [236]。
- 实时监控 ZFS 流传输进度。有关详细信息，请参见“监视 ZFS 发送流的进度” [182]。
- 支持统一归档，因此简化了配置根池恢复设置的过程。有关更多信息，请参见《在 Oracle Solaris 11.2 中使用统一归档文件进行系统恢复和克隆》。

什么是 Oracle Solaris ZFS？

Oracle Solaris ZFS 文件系统是一种从根本上改变文件系统管理方式的文件系统，并具有目前市面上的其他文件系统所没有的功能和优点。ZFS 强健、可伸缩，且易于管理。

ZFS 池存储

ZFS 使用存储池的概念来管理物理存储。以前，文件系统是在单个物理设备的基础上构造的。为了利用多个设备和提供数据冗余性，引入了卷管理器的概念来提供单个设备的表示，以便无需修改文件系统即可利用多个设备。此设计增加了复杂性，最终阻碍了特定文件系统的继续发展，因为这类文件系统无法控制虚拟卷数据的物理位置。

ZFS 可完全避免使用卷管理。ZFS 将设备聚集到存储池中，而不是强制要求创建虚拟卷。存储池描述了存储的物理特征（设备布局、数据冗余等），并充当可以从其创建文件系统的任意数据存储库。文件系统不再受限于单个设备，允许它们与池中的所有文件系统共享磁盘空间。您不再需要预先确定文件系统的大小，因为文件系统会在分配给存储池的磁盘空间内自动增长。添加新存储器后，无需执行其他操作，池中的所有文件系统即可立即使用所增加的磁盘空间。在许多方面，存储池与虚拟内存系统相似：将一个内存 DIMM 加入系统时，操作系统并不强迫您运行命令来配置内存并将其指定给个别进程。系统中的所有进程都会自动使用所增加的内存。

事务性语义

ZFS 是事务性文件系统，这意味着文件系统状态在磁盘上始终是一致的。传统文件系统可就地覆盖数据，这意味着如果系统断电（例如，在分配数据块到将其链接到目录中的时间段内断电），则会使文件系统处于不一致状态。以前，此问题是通过使用 `fsck` 命令解决的。此命令负责检查并验证文件系统状态，并尝试在操作过程中修复任何不一致性。这种文件系统不一致问题曾给管理员造成巨大困扰，`fsck` 命令并不保证能够解决所有可能的问题。最近，文件系统引入了日志记录的概念。日志记录过程在单独的日志中记录操作，在系统发生崩溃时，可以安全地重放该日志。此过程会带来不必要的开销，因为需要两次写入数据，而这通常会导致一组新问题（如日志无法正确重放）。

对于事务性文件系统，数据是使用写复制语义管理的。数据永远不会被覆盖，并且任何操作序列会全部被提交或全部被忽略。因此，文件系统绝对不会因意外断电或系统崩溃而被损坏。尽管最近写入的数据片段可能丢失，但是文件系统本身将始终是一致的。此外，只有在写入同步数据（使用 O_DSYNC 标志写入）后才返回，因此同步数据决不会丢失。

校验和与自我修复数据

对于 ZFS，所有数据和元数据都通过用户可选择的校验和算法进行验证。提供校验和验证的传统文件系统出于卷管理层和传统文件系统设计的必要，会逐块执行此操作。在传统设计中，某些故障可能导致数据不正确但没有校验和错误，如向错误位置写入完整的块等。ZFS 校验和的存储方式可确保检测到这些故障并可以正常地从其中进行恢复。所有校验和验证与数据恢复都是在文件系统层执行的，并且对应用程序是透明的。

此外，ZFS 还会提供自我修复数据。ZFS 支持存储池具有各种级别的数据冗余性。检测到坏的数据块时，ZFS 会从另一个冗余副本中提取正确的数据，而且会用正确的数据替换错误的数据。

独一无二的可伸缩性

ZFS 文件系统的一个关键设计要素是可伸缩性。该文件系统本身是 128 位的，所允许的存储空间是 256 quadrillion zettabyte (256x1015 ZB)。所有元数据都是动态分配的，因此在首次创建时无需预先分配 inode，否则就会限制文件系统的可伸缩性。所有算法在编写时都考虑到了可伸缩性。目录最多可以包含 2^{48}（256 万亿）项，并且对于文件系统数或文件系统中可以包含的文件数不存在限制。

ZFS 快照

快照是文件系统或卷的只读副本。可以快速而轻松地创建快照。最初，快照不会占用池中的任何附加磁盘空间。

活动数据集中的数据更改时，快照通过继续引用旧数据来占用磁盘空间。因此，快照可防止将数据释放回池中。

简化管理

最重要的是，ZFS 提供了一种极度简化的管理模型。通过使用分层次的文件系统布局、属性继承以及自动管理挂载点和 NFS 共享语义，ZFS 可轻松创建和管理文件系统，而无需使用多个命令或编辑配置文件。可以轻松设置配额或预留空间，启用或禁用压缩，或者通过单个命令管理许多文件系统的挂载点。您就可以检查或替换设备，而无需学习另外的一套卷管理命令。您可以发送和接收文件系统快照流。

ZFS 通过分层结构管理文件系统，该分层结构允许对属性（如配额、预留空间、压缩和挂载点）进行这一简化管理。在此模型中，文件系统是中央控制点。文件系统本身的开销非常小（相当于创建一个新目录），因此鼓励您为每个用户、项目、工作区等创建一个文件系统。通过此设计，可定义细分的管理点。

ZFS 术语

本节介绍了在本书中使用的基本术语：

引导环境
: 引导环境是可引导的 Oracle Solaris 环境，由 ZFS 根文件系统和在其下挂载的其他文件系统（可选）组成。同一时间只能有一个引导环境处于活动状态。

校验和
: 文件系统块中的数据的 256 位散列。校验和功能包括简单快捷的 fletcher4（缺省）到 SHA256 这样的加密强散列。

克隆
: 其初始内容与快照内容相同的文件系统。

  有关克隆的信息，请参见“ZFS 克隆概述” [175]。

数据集
: 以下 ZFS 组件的通用名称： 克隆、文件系统、快照和卷。

  每个数据集由 ZFS 名称空间中的唯一名称标识。数据集使用以下格式进行标识：

  pool/*path*[@*snapshot*]

  pool 标识包含数据集的存储池的名称

  path 数据集组件的斜杠分隔路径名

  snapshot 用于标识数据集快照的可选组件

  有关数据集的更多信息，请参见第 5 章 管理 Oracle Solaris ZFS 文件系统。

文件系统
: 挂载在标准系统名称空间中且行为与其他文件系统相似的 `filesystem` 类型的 ZFS 数据集。

  有关文件系统的更多信息，请参见第 5 章 管理 Oracle Solaris ZFS 文件系统。

镜像
: 在两个或多个磁盘上存储数据的相同副本的虚拟设备。如果镜像中的任一磁盘出现故障，则该镜像中的其他任何磁盘可以提供相同的数据。

池
: 设备的逻辑组，用于描述可用存储的布局和物理特征。数据集的磁盘空间是从池中分配的。

  有关存储池的更多信息，请参见第 3 章 管理 Oracle Solaris ZFS 存储池。

RAID-Z
: 在多个磁盘上存储数据和奇偶校验的虚拟设备。有关 RAID-Z 的更多信息，请参见“RAID-Z 存储池配置” [32]。

重新同步
: 将数据从一个设备复制到另一个设备的过程称为重新同步。例如，如果替换了镜像设备或使其脱机，则最新镜像设备中的数据会复制到刚恢复的镜像设备。此过程在传统的卷管理产品中称为镜像重新同步。

  有关 ZFS 重新同步的更多信息，请参见“查看重新同步状态” [255]。

快照
: 文件系统或卷在给定时间点的只读副本。

	有关快照的更多信息，请参见“ZFS 快照概述” [169]。
虚拟设备	池中的逻辑设备，可以是物理设备、文件或设备集合。 有关虚拟设备的更多信息，请参见“显示存储池虚拟设备信息” [39]。
卷	表示块设备的数据集。例如，可以创建 ZFS 卷作为交换设备。 有关 ZFS 卷的更多信息，请参见“ZFS 卷” [227]。

ZFS 组件命名要求

每个 ZFS 组件（例如数据集和池）必须根据以下规则进行命名：

- 每个组件只能包含字母数字字符以及以下四个特殊字符：
  - 下划线 (_)
  - 连字符 (-)
  - 冒号 (:)
  - 句点 (.)
- 池名称必须以字母开头，并且只能包含字母数字字符以及下划线 (_)、短划线 (-) 和句点 (.)。请注意有关池名称的以下限制：
  - 不允许使用起始序列 `c[0-9]`。
  - 名称 `log` 为保留名称。
  - 不允许使用以 `mirror`、`raidz`、`raidz1`、`raidz2`、`raidz3` 或 `spare` 开头的名称，因为这些名称是保留名称。
  - 数据集名称不得包含百分比符号 (%)。
- 数据集名称必须以字母数字字符开头。
- 数据集名称不得包含百分比符号 (%)。

此外，不允许使用空组件。

Oracle Solaris ZFS 与传统文件系统的区别

- “ZFS 文件系统粒度” [18]
- “ZFS 磁盘空间记帐” [18]
- “挂载 ZFS 文件系统” [19]
- “传统卷管理” [20]
- “基于 NFSv4 的 Solaris ACL 模型” [20]

ZFS 文件系统粒度

过去，文件系统局限于单个设备，因而其大小以设备的大小为限。由于存在大小限制，因此创建和重新创建传统文件系统很耗时，有时候还很难。传统的卷管理产品可帮助管理此过程。

由于 ZFS 文件系统不局限于特定设备，因此可以轻松、快捷地创建，其创建方法与目录的创建方法相似。ZFS 文件系统会在分配给其所驻留的存储池的磁盘空间中自动增大。

要管理许多用户子目录，可以为每个用户创建一个文件系统，而不是只创建一个文件系统（如 `/export/home`）。通过应用可被分层结构中包含的后代文件系统继承的属性，可以轻松设置和管理许多文件系统。

有关如何创建文件系统分层结构的示例，请参见“创建 ZFS 文件系统分层结构” [24]。

ZFS 磁盘空间记帐

ZFS 建立在池存储概念的基础上。与典型文件系统映射到物理存储器不同，池中的所有 ZFS 文件系统都共享该池中的可用存储器。因此，即使文件系统处于非活动状态，实用程序（例如 `df`）报告的可用磁盘空间也会发生变化，因为池中的其他文件系统会使用或释放磁盘空间。

注意，使用配额可以限制最大文件系统大小。有关配额的信息，请参见“设置 ZFS 文件系统的配额” [153]。可以利用预留空间保证一个文件系统拥有指定大小的磁盘空间。有关预留的信息，请参见“设置 ZFS 文件系统的预留空间” [156]。此模型与 NFS 模型非常相似，NFS 模型基于同一文件系统（例如 `/home`）挂载多个目录。

ZFS 中的所有元数据都是动态分配的。其他大部分文件系统都会预分配其大量元数据。因此，创建文件系统时，此元数据需要直接占用空间。此行为还意味着文件系统支持的文件总数是预先确定的。由于 ZFS 根据需要分配其元数据，因此不需要初始空间成本，并且文件数只受可用磁盘空间的限制。对于 ZFS 文件系统，对 `df -g` 命令输出的解释必须和其他文件系统不同。报告的 `total files` 只是根据池中可用的存储量得出的估计值。

ZFS 是事务性文件系统。大部分文件系统修改都捆绑到事务组中，并异步提交至磁盘。这些修改在被提交到磁盘之前称为暂挂更改。已用磁盘空间量、可用磁盘空间量以及文件或文件系统引用的磁盘空间量并不考虑暂挂更改。通常，暂挂更改仅占用几秒钟的时间。即使使用 `fsync(3c)` 或 `O_SYNC` 提交对磁盘的更改，也不一定能保证有关磁盘空间使用情况的信息会立即更新。

在 UFS 文件系统上，`du` 命令报告文件中数据块的大小。在 ZFS 文件系统上，`du` 报告在磁盘上存储的文件的实际大小。该大小包括元数据以及压缩。此报告确实有助于解答“删

除此文件可获得多少多余空间？”这一问题。因此，即使压缩已关闭，您仍会在 ZFS 和 UFS 之间看到不同的结果。

对 `df` 命令与 `zfs list` 命令报告的空间占用进行比较时，请注意，`df` 报告的是池大小而不仅仅是文件系统大小。此外，`df` 不了解后代文件系统或快照是否存在。如果在文件系统上设置了任何 ZFS 属性（如压缩和配额），则协调由 `df` 报告的空间占用可能很难。

请考虑也可能影响所报告的空间占用的以下情况：

- 对于大于 `recordsize` 的文件，文件的最后一个数据块通常只填充大约 1/2。当缺省 `recordsize` 设置为 128 KB 时，每个文件大约浪费 64 KB，这可能影响很大。集成 RFE 6812608 可解决此情况。通过启用压缩可解决此问题。即使数据已压缩，最后一个数据块的未使用部分也将以零填充，且压缩非常好。
- 在 RAIDZ-2 池上，每个数据块至少要占用 2 个扇区（512 字节块）来存储奇偶校验信息。奇偶校验信息所占用的空间不会被报告，但是因为它可能是变化的，且在小数据块中所占的百分比更大，因此对空间报告可能有显著影响。对于设置为 512 字节的 `recordsize`，影响更为明显，其中每个 512 字节的逻辑数据块都占用 1.5 KB（该空间的 3 倍）。不管存储什么数据，如果您最关注的是空间效率，则应该将 `recordsize` 保留为缺省值 (128 KB)，并启用压缩（至 `lzjb` 的缺省值）。
- `df` 命令不会识别已由重复数据删除操作剔除的文件数据。

空间不足行为

文件系统的快照开销很小，并且很容易在 ZFS 中创建。在大多数 ZFS 环境中，快照是常见现象。有关 ZFS 快照的信息，请参见第 6 章 使用 Oracle Solaris ZFS 快照和克隆。

尝试释放磁盘空间时，快照的存在会引起某种意外行为。通常，获取适当的权限后，可从整个文件系统中删除一个文件，此操作会使文件系统有更多的可用磁盘空间。但是，如果要删除的文件存在于文件系统的快照中，则删除该文件不会获得任何磁盘空间。快照将继续引用该文件使用的块。

由于需要创建新版本的目录来反映名称空间的新状态，因此删除文件会占用更多的磁盘空间。此行为意味着，尝试删除文件时可能收到意外的 `ENOSPC` 或 `EDQUOT` 错误。

挂载 ZFS 文件系统

ZFS 可降低复杂性并简化管理。例如，使用传统文件系统时，每次添加新文件系统都必须编辑 `/etc/vfstab` 文件。ZFS 可根据文件系统的属性自动挂载和卸载文件系统，从而避免了上述需求。无需管理 `/etc/vfstab` 文件中的 ZFS 项。

有关挂载和共享 ZFS 文件系统的更多信息，请参见“挂载 ZFS 文件系统” [138]。

传统卷管理

如“ZFS 池存储” [14] 中所述，ZFS 不需要单独的卷管理器。ZFS 对原始设备执行操作，因此可能会创建由逻辑卷（软件或硬件）构成的存储池。由于 ZFS 在使用原始物理设备时可获得最佳工作状态，因此建议不使用此配置。使用逻辑卷可能会牺牲性能和/或可靠性，因此应尽量避免。

基于 NFSv4 的 Solaris ACL 模型

Solaris OS 的旧版本支持主要基于 POSIX ACL 草案规范的 ACL 实现。基于 POSIX 草案的 ACL 用来保护 UFS 文件。基于 NFSv4 规范的新 Solaris ACL 模型用来保护 ZFS 文件。

与旧模型相比，新 Solaris ACL 模型的主要变化如下：

- 该模型基于 NFSv4 规范，与 NT 样式的 ACL 类似。
- 此模型提供更为详尽的访问特权集合。
- ACL 分别使用 `chmod` 和 `ls` 命令（而非 `setfacl` 和 `getfacl` 命令）进行设置和显示。
- 提供了更丰富的继承语义，用于指定如何将访问特权从目录应用到子目录等。

有关对 ZFS 文件使用 ACL 的更多信息，请参见第 7 章 使用 ACL 和属性保护 Oracle Solaris ZFS 文件。

♦ ♦ ♦ 第 2 章

2 Oracle Solaris ZFS 入门

本章提供关于 Oracle Solaris ZFS 基本配置的逐步说明。阅读完本章之后，您应基本了解 ZFS 命令的工作原理，并可以创建基本的池和文件系统。本章并非综合概述，有关更多详细信息，请参阅后续章节。

本章包含以下各节：

ZFS 权限配置文件

如果要在不使用超级用户 (root) 帐户的情况下执行 ZFS 管理任务，可承担具有以下任一配置文件的角色来执行 ZFS 管理任务：

- ZFS 存储管理 – 提供了在 ZFS 存储池中创建、销毁和处理设备的特权
- ZFS 文件系统管理 – 提供了创建、销毁和修改 ZFS 文件系统的特权

有关创建或分配角色的更多信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全》。

除了使用 RBAC 角色来管理 ZFS 文件系统之外，还可以考虑使用 ZFS 委托管理来分散 ZFS 管理任务。有关更多信息，请参见第 8 章 Oracle Solaris ZFS 委托管理。

ZFS 硬件和软件要求及建议

尝试使用 ZFS 软件之前，请确保查看了以下硬件和软件要求及建议：

- 使用运行有受支持的 Oracle Solaris 发行版、基于 SPARC® 或 x86 的系统。

- 存储池所需的最小磁盘空间量为 64 MB。最小磁盘空间为 128 MB。
- 为了获得良好的 ZFS 性能，请根据您的工作负荷来确定所需要的内存大小。
- 如果要创建镜像的池配置，应使用多个控制器。

创建基本 ZFS 文件系统

ZFS 管理在设计过程中考虑了简单性。其设计目标之一是减少创建可用文件系统所需的命令数。例如，创建新池的同时会创建一个新 ZFS 文件系统，并自动将其挂载。

以下示例说明如何通过一个命令同时创建名为 `tank` 的基本镜像存储池和名为 `tank` 的 ZFS 文件系统。假定磁盘 `/dev/dsk/c1t0d0` 和 `/dev/dsk/c2t0d0` 全部都可使用。

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

有关冗余 ZFS 池配置的更多信息，请参见“ZFS 存储池的复制功能” [31]。

新 ZFS 文件系统 `tank` 可根据需要使用可用的磁盘空间，并会自动挂载在 `/tank` 中。

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

在池内，可能需要创建其他文件系统。文件系统可提供管理点，用于管理同一池中不同的数据集。

以下示例说明如何在存储池 `tank` 中创建名为 `fs` 的文件系统。

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

新 ZFS 文件系统 `tank/fs` 可根据需要使用可用的磁盘空间，并会自动挂载在 `/tank/fs`。

```
# 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，您需要创建并组织与您公司的需要相符的文件系统分层结构。有关创建 ZFS 文件系统分层结构的信息，请参见“创建 ZFS 文件系统分层结构” [24]。

创建基本的 ZFS 存储池

上一示例说明了 ZFS 的简单性。本章的其余部分将提供一个更复杂的示例，与您的环境中所遇到的情况相似。第一个任务是确定存储要求并创建存储池。该池描述了存储的物理特征，并且必须在创建任何文件系统之前创建。

▼ 如何确定 ZFS 存储池的存储要求

1. **确定存储池可用的设备。**

   创建存储池之前，必须先确定用于存储数据的设备。这些设备必须是大小至少为 128 MB 的磁盘，并且不能由操作系统的其他部分使用。这些设备可以是预先格式化的磁盘上的单独分片，也可以是 ZFS 格式化为单个大分片的整个磁盘。

   在如何创建 ZFS 存储池 [23]的存储示例中，假定磁盘 `/dev/dsk/c1t0d0` 和 `/dev/dsk/c2t0d0` 全部都可供使用。

   有关磁盘及其使用和标记方法的更多信息，请参见“使用 ZFS 存储池中的磁盘” [27]。

2. **选择数据复制。**

   ZFS 支持多种类型的数据复制，这决定了池可以经受的硬件故障的类型。ZFS 支持非冗余（条带化）配置以及镜像和 RAID-Z（RAID-5 的变化形式）。

   如何创建 ZFS 存储池 [23]中的存储示例使用了两个可用磁盘的基本镜像。

   有关 ZFS 复制功能的更多信息，请参见“ZFS 存储池的复制功能” [31]。

▼ 如何创建 ZFS 存储池

1. **成为 root 用户或承担具有适当 ZFS 权限配置文件的等效角色。**

   有关 ZFS 权限配置文件的更多信息，请参见“ZFS 权限配置文件” [21]。

2. **为存储池取名。**

   此名称用于在使用 `zpool` 和 `zfs` 命令时标识存储池。选择您喜欢的任何池名称，但是必须符合“ZFS 组件命名要求” [17]中的命名要求。

3. **创建池。**

   例如，以下命令创建一个名为 `tank` 的镜像池：

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

   如果一个或多个设备包含其他文件系统或正在使用中，则该命令不能创建池。

   有关创建存储池的更多信息，请参见“创建 ZFS 存储池” [34]。有关如何确定设备使用情况的更多信息，请参见“检测使用中的设备” [40]。

4. **查看结果。**

   使用 `zpool list` 命令可以确定是否已成功创建池。

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

```
tank                   80G    137K     80G     0%  ONLINE     -
```

有关查看池状态的更多信息，请参见“查询 ZFS 存储池的状态” [64]。

创建 ZFS 文件系统分层结构

创建用于存储数据的存储池之后，即可创建文件系统分层结构。分层结构是用于组织信息的简单但功能强大的机制。使用过文件系统的用户对分层结构也都很熟悉。

使用 ZFS 可将文件系统组织为分层结构，其中每个文件系统仅有一个父级。分层结构的根始终是池名称。ZFS 通过支持属性继承来利用此分层结构，以便可在整个文件系统树中快速轻松地设置公用属性。

▼ 如何确定 ZFS 文件系统分层结构

1. **选择文件系统粒度。**

   ZFS 文件系统是管理的中心点。它们是轻量型的，很容易创建。适用的模型是为每个用户或项目建立一个文件系统，因为此模型允许按用户或按项目控制属性、快照和备份。

   How to Create ZFS File Systems中创建了两个 ZFS 文件系统：`jeff` 和 如何创建 ZFS 文件系统 [25]。

   有关管理文件系统的更多信息，请参见第 5 章 管理 Oracle Solaris ZFS 文件系统。

2. **对相似的文件系统进行分组。**

   使用 ZFS 可将文件系统组织为分层结构，以便可对相似的文件系统进行分组。此模型提供了一个用于控制属性和管理文件系统的管理中心点。应使用一个公用名称来创建相似的文件系统。

   在如何创建 ZFS 文件系统 [25]的示例中，两个文件系统都放置在名为 `home` 的文件系统下。

3. **选择文件系统属性。**

   大多数文件系统特征都是通过属性进行控制。这些属性可以控制多种行为，包括文件系统的挂载位置、共享方式、是否使用压缩以及是否有任何生效的配额。

   在如何创建 ZFS 文件系统 [25]的示例中，所有起始目录都挂载在 `/export/zfs/`*user* 中，都使用 NFS 来共享并且都已启用压缩。此外，还对用户 `jeff` 强制实施了 10 GB 的配额。

   有关属性的更多信息，请参见“介绍 ZFS 属性” [116]。

▼ 如何创建 ZFS 文件系统

1. **成为 root 用户或承担具有适当 ZFS 权限配置文件的等效角色。**

   有关 ZFS 权限配置文件的更多信息，请参见“ZFS 权限配置文件” [21]。

2. **创建所需的分层结构。**

   在本示例中，创建了一个可充当各文件系统的容器的文件系统。

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

3. **设置继承的属性。**

   建立文件系统分层结构之后，设置需在所有用户之间共享的任何属性：

   ```
   # 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
   ```

   可在创建文件系统时设置文件系统属性。例如：

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

   有关属性和属性继承的更多信息，请参见“介绍 ZFS 属性” [116]。

   然后，在池 `tank` 中的 `home` 文件系统下对各文件系统进行分组。

4. **创建各文件系统。**

   文件系统可能已创建，并可能已在 `home` 级别更改了属性。所有属性均可在使用文件系统的过程中动态进行更改。

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

   这些文件系统从其父级继承属性值，因此会自动挂载在 `/export/zfs/`*user* 中并且通过 NFS 共享。您无需编辑 `/etc/vfstab` 或 `/etc/dfs/dfstab` 文件。

   有关创建文件系统的更多信息，请参见“创建 ZFS 文件系统” [114]。

   有关挂载和共享文件系统的更多信息，请参见“挂载 ZFS 文件系统” [138]。

5. **设置文件系统特定的属性。**

   在本例中，为用户 `jeff` 分配了一个 10 GB 的配额。此属性会对该用户可以使用的空间量施加限制，而不考虑池中的可用空间大小。

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

6. **查看结果。**

使用 `zfs list` 命令查看可用的文件系统信息：

```
# 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
```

请注意，用户 `jeff` 仅有 10 GB 的可用空间，而用户 `bill` 则可使用整个池 (67 GB)。

有关查看文件系统状态的更多信息，请参见“查询 ZFS 文件系统信息” [131]。

有关磁盘空间的使用和计算方法的更多信息，请参见“ZFS 磁盘空间记帐” [18]。

♦ ♦ ♦ 第 3 章

管理 Oracle Solaris ZFS 存储池

本章介绍如何创建和管理 Oracle Solaris ZFS 中的存储池。

本章包含以下各节：

ZFS 存储池的组件

以下各节提供有关以下存储池组件的详细信息：

使用 ZFS 存储池中的磁盘

存储池的最基本元素是物理存储器。物理存储器可以是大小至少为 128 MB 的任何块设备。通常，此设备是 `/dev/dsk` 目录中对系统可见的一个硬盘驱动器。

存储设备可以是整个磁盘 (`c1t0d0`) 或单个分片 (`c0t0d0s7`)。建议的操作模式是使用整个磁盘，在这种情况下，无需对磁盘进行特殊格式化。ZFS 可格式化使用 EFI 标签的磁盘以包含单个大分片。以此方式使用磁盘时，`format` 命令显示的分区表与以下信息类似：

```
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
```

大多数情况下，安装 Oracle Solaris 11.1 时会为基于 x86 的系统上的根池磁盘加上 EFI (GPT) 标签，具体内容类似于以下信息：

```
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
```

在以上输出中，分区 0 (`BIOS boot`) 包含必需的 GPT 引导信息。与分区 8 类似，该分区无需管理，因此不应该修改。根文件系统包含在分区 1 中。

具有 GPT 感知固件的 SPARC 系统加上了 EFI (GPT) 磁盘标签。例如：

```
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
```

在 ZFS 存储池中使用整个的磁盘时，请检查以下注意事项：

- 使用整个磁盘时，通常使用 `/dev/dsk/cNtNdN` 命名约定为该磁盘命名。一些第三方驱动程序使用不同的命名约定，或者将磁盘放置在除 `/dev/dsk` 目录以外的位置中。要使用这些磁盘，必须手动标记磁盘并为 ZFS 提供分片。

- 在基于 x86 的系统上，磁盘必须具有有效的 Solaris `fdisk` 分区。有关创建或更改 Solaris `fdisk` 分区的更多信息，请参见《在 Oracle Solaris 11.2 中管理设备 》中的“为 ZFS 文件系统设置磁盘”。
- 创建包含整个磁盘的存储池时，ZFS 会应用 EFI 标签。有关 EFI 标签的更多信息，请参见《在 Oracle Solaris 11.2 中管理设备 》中的“EFI (GPT) 磁盘标签”。
- 大多数情况下，在具有 GPT 感知固件的基于 SPARC 的系统和基于 x86 的系统上，Oracle Solaris 安装程序会为根池磁盘应用 EFI (GPT) 标签。有关详细信息，请参见“确定 ZFS 根池要求” [88]。
- 要恢复根池，请考虑使用 `archiveadm` 命令创建根池归档文件。拆分根池可能造成错误，因为这需要其他手动步骤，如设置一个新的引导设备，可能要更新 `/etc/vfstab` 文件，以及重置现有的转储设备。

  有关创建根池归档文件的更多信息，请参见《在 Oracle Solaris 11.2 中使用统一归档文件进行系统恢复和克隆 》。

可以通过以下方式指定磁盘：使用 `/dev/dsk/c1t0d0` 这样的完整路径，或表示 `/dev/dsk` 目录中设备名的简写名称，例如 `c1t0d0`。例如，以下是有效的磁盘名称：

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

使用 ZFS 存储池中的分片

当创建包含一个磁盘分片的存储池时，可以为磁盘加上传统 Solaris VTOC (SMI) 标签，但不建议对一个池使用多个磁盘分片，因为管理磁盘分片将更加困难。

在基于 SPARC 的系统上，72 GB 的磁盘在分片 0 上有 68 GB 的可用空间，如下列 `format` 输出所示。

```
# 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
```

在基于 x86C 的系统上，72 GB 的磁盘在分片 0 上有 68 GB 的可用磁盘空间，如下列 `format` 输出所示。分片 8 包含少量引导信息。 分片 8 不需要管理，并且无法对其进行更改。

```
# 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
```

基于 x86 的系统上还存在一个 `fdisk` 分区。`fdisk` 分区由 `/dev/dsk/cN[tN]dNpN` 设备名来表示，并充当磁盘可用分片的容器。请勿对 ZFS 存储池组件使用 `cN[tN]dNpN` 设备，因为该配置既未经过测试，也不受支持。

使用 ZFS 存储池中的文件

ZFS 还允许将文件用作存储池中的虚拟设备。此功能主要用于测试和启用简单的实验，而不是用于生产。

- 如果创建了由 UFS 文件系统中的文件支持的 ZFS 池，即会隐式依赖于 UFS 来保证正确性和同步语义。
- 如果创建的 ZFS 池基于在其他 ZFS 池中创建的文件或卷，则系统可能死锁或崩溃。

但是，如果首次试用 ZFS，或者在没有足够的物理设备时尝试更复杂的配置，则文件会非常有用。所有文件必须以完整路径的形式指定，并且大小至少为 64 MB。

ZFS 存储池的注意事项

创建和管理 ZFS 存储池时，请注意以下事项。

- 创建 ZFS 存储池的最简单方法是使用整个物理磁盘。在从磁盘分片、硬件 RAID 阵列中的 LUN 或基于软件的卷管理器所提供的卷中生成池时，无论从管理、可靠性还是性能的角度而言，ZFS 配置都变得越来越复杂。以下注意事项可能有助于确定如何用其他硬件或软件存储解决方案来配置 ZFS：
  - 如果在硬件 RAID 阵列中的 LUN 上构建 ZFS 配置，则需要了解 ZFS 冗余功能与该阵列所提供的冗余功能之间的关系。有些配置可能会提供足够的冗余和性能，而其他配置可能不会提供足够的冗余和性能。
  - 可以使用由基于软件的卷管理器提供的卷为 ZFS 构造逻辑设备。但是，建议不要使用这些配置。尽管 ZFS 可在这类设备上正常运行，但结果可能是实际性能低于最佳性能。

    有关存储池建议和使用带有硬件 RAID 的 ZFS 时的其他信息，请参见第 11 章 建议的 Oracle Solaris ZFS 做法。
- 有关使用池设备的注意事项的更多信息，请参见“更改池设备” [257]。

ZFS 存储池的复制功能

ZFS 在镜像和 RAID-Z 配置中提供了数据冗余以及自我修复属性。

镜像存储池配置

镜像存储池配置至少需要两个磁盘，而且磁盘最好位于不同的控制器上。可以在一个镜像配置中使用许多磁盘。此外，还可以在每个池中创建多个镜像。从概念上讲，基本镜像配置与以下内容类似：

```
mirror c1t0d0 c2t0d0
```

从概念上讲，更复杂的镜像配置与以下内容类似：

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

有关创建镜像存储池的信息，请参见“创建镜像存储池” [34]。

RAID-Z 存储池配置

除镜像存储池配置外，ZFS 还提供具有单/双/三奇偶校验容错性的 RAID-Z 配置。单奇偶校验 RAID-Z (`raidz` 或 `raidz1`) 与 RAID-5 类似。双奇偶校验 RAID-Z (`raidz2`) 与 RAID-6 类似。

有关 RAIDZ-3 (`raidz3`) 的更多信息，请参见以下博客：

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

所有与 RAID-5 类似的传统算法（例如 RAID-4、RAID-6、RDP 和 EVEN-ODD）都可能存在称为“RAID-5 写入漏洞”的问题。如果仅写入了 RAID-5 条带的一部分，并且在所有块成功写入磁盘之前断电，则奇偶校验将与数据不同步，因此永远无用，除非后续的完全条带化写操作将其覆盖。在 RAID-Z 中，ZFS 使用可变宽度的 RAID 条带，以便所有写操作都是完全条带化写操作。这是唯一可行的设计，因为 ZFS 通过以下方式将文件系统和设备管理集成在一起：文件系统的元数据包含有关底层数据冗余模型的足够信息以处理可变宽度的 RAID 条带。RAID-Z 是世界上针对 RAID-5 写入漏洞的第一个仅使用软件的解决方案。

一个 RAID-Z 配置包含 N 个大小为 X 的磁盘，其中有 P 个奇偶校验磁盘，该配置可以存放大约 (N-P)*X 字节的数据，并且只有在 P 个设备出现故障时才会危及数据完整性。单奇偶校验 RAID-Z 配置至少需要两个磁盘，双奇偶校验 RAID-Z 配置至少需要三个磁盘，以此类推。例如，如果一个单奇偶校验 RAID-Z 配置中有三个磁盘，则奇偶校验数据占用的磁盘空间与其中一个磁盘的空间相等。除此之外，创建 RAID-Z 配置无需任何其他特殊硬件。

从概念上讲，包含三个磁盘的 RAID-Z 配置与以下内容类似：

```
raidz c1t0d0 c2t0d0 c3t0d0
```

从概念上讲，更复杂的 RAID-Z 配置与以下内容类似：

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

如果创建具有许多磁盘的 RAID-Z 配置，请考虑将这些磁盘分为多个组。例如，具有 14 个磁盘的 RAID-Z 配置最好拆分为两个 7 磁盘组。若 RAID-Z 配置包含的分组中的磁盘数目为一位数 (1-9)，则该配置的性能应该更好。

有关创建 RAID-Z 存储池的信息，请参见“创建 RAID-Z 存储池” [35]。

有关基于性能和磁盘空间注意事项在镜像配置或 RAID-Z 配置之间进行选择的更多信息，请参见以下博客：

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

有关 RAID-Z 存储池建议的其他信息，请参见第 11 章 建议的 Oracle Solaris ZFS 做法。

ZFS 混合存储池

Oracle Sun Storage 7000 产品系列所提供的 ZFS 混合存储池是一种组合了 DRAM、SSD 和 HDD 的特殊存储池，可以提高性能、增加容量并降低能耗。通过此产品的管理界面，您可以选择存储池的 ZFS 冗余配置，并轻松管理其他配置选项。

有关此产品的更多信息，请参见《*Sun Storage Unified Storage System* 管理指南》。

冗余配置中的自我修复数据

ZFS 在镜像配置或 RAID-Z 配置中提供了自我修复数据。

检测到坏的数据块时，ZFS 不但从另一个冗余副本中提取正确的数据，而且会用正确的副本替换错误的数据。

存储池中的动态条带化

ZFS 以条带形式将数据动态分布在所有顶层虚拟设备上。由于在写入时确定放置数据的位置，因此在分配时不会创建固定宽度的条带。

向池中添加新虚拟设备时，ZFS 会将数据逐渐分配给新设备，以便维护性能和磁盘空间分配策略。每个虚拟设备也可以是包含其他磁盘设备或文件的镜像或 RAID-Z 设备。使用此配置，可以灵活地控制池的故障特征。例如，可以通过 4 个磁盘创建以下配置：

- 使用动态条带化的四个磁盘
- 一个四向 RAID-Z 配置
- 使用动态条带化的两个双向镜像

虽然 ZFS 支持一个池包含不同类型的虚拟设备，但应避免这种做法。例如，可以创建一个包含一个双向镜像和一个三向 RAID-Z 配置的池。但是，容错能力几乎与最差的虚拟设备（在本示例中为 RAID-Z）相同。最佳做法是使用相同类型的顶层虚拟设备，并且每个设备的冗余级别相同。

创建和销毁 ZFS 存储池

以下各节介绍创建和销毁 ZFS 存储池的不同情况：

- “创建 ZFS 存储池” [34]
- “显示存储池虚拟设备信息” [39]
- “处理 ZFS 存储池创建错误” [40]

- “销毁 ZFS 存储池” [42]

创建和销毁池非常快捷方便。但是，执行这些操作务必谨慎。虽然进行了检查，以防止在新的池中使用现已使用的设备，但是 ZFS 无法始终知道设备何时已在使用中。存储池销毁容易创建难。请谨慎使用 `zpool destroy`。这个简单的命令会有严重后果。

创建 ZFS 存储池

要创建存储池，请使用 `zpool create` 命令。此命令采用池名称和任意数目的虚拟设备作为参数。池名称必须符合“ZFS 组件命名要求” [17]中的命名要求。

创建基本存储池

以下命令创建了一个名为 `tank` 的新池，该池由磁盘 `c1t0d0` 和 `c1t1d0` 组成：

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

代表整个磁盘的设备名称可在 `/dev/dsk` 目录中找到，并通过 ZFS 恰当地标记为包含单个大分片。数据通过这两个磁盘以动态方式进行条带化。

创建镜像存储池

要创建镜像池，请使用 `mirror` 关键字，后跟将组成镜像的任意数目的存储设备。可以通过在命令行中重复使用 `mirror` 关键字指定多个镜像。以下命令创建了一个包含两个双向镜像的池：

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

第二个 `mirror` 关键字指明将指定一个新的顶层虚拟设备。数据在这两个镜像中以动态方式进行条带化，并会相应地在各个磁盘上创建冗余数据。

有关建议的镜像配置的更多信息，请参见第 11 章 建议的 Oracle Solaris ZFS 做法。

目前，ZFS 镜像配置中支持以下操作：

- 向现有镜像配置中添加用于其他顶层虚拟设备 (vdev) 的另一组磁盘。有关更多信息，请参见“向存储池中添加设备” [44]。
- 向现有镜像配置中附加其他磁盘。或者，向非复制配置中附加其他磁盘，以创建镜像配置。有关更多信息，请参见“附加和分离存储池设备” [48]。
- 只要替换磁盘的大小大于或等于要被替换的设备，便可替换现有镜像配置中的一个或多个磁盘。有关更多信息，请参见“替换存储池中的设备” [55]。
- 只要剩余设备可为配置提供足够冗余，便可分离镜像配置中的磁盘。有关更多信息，请参见“附加和分离存储池设备” [48]。

- 通过分离其中一个磁盘来拆分镜像配置，以创建新的相同池。有关更多信息，请参见“通过拆分镜像 ZFS 存储池创建新池” [50]。

不能直接从镜像存储池中移除非备用设备、非日志设备或非缓存设备。

创建 ZFS 根池

请注意以下根池配置要求：

- 在 Oracle Solaris 中，在基于 x86 的系统和具有 GPT 感知固件的受支持 SPARC 系统上用于根池的磁盘会在安装时加上 EFI (GPT) 标签，在没有 GPT 感知固件的基于 SPARC 的系统上用于根池的磁盘会在安装时加上 SMI (VTOC) 标签。安装程序会尽可能地应用 EFI (GPT) 标签，如果需要在安装后重新创建 ZFS 根池，则可以使用以下命令来应用 EFI (GPT) 磁盘标签和正确的引导信息：

  ```
  # zpool create -B rpool2 c1t0d0
  ```
- 根池必须作为镜像配置或单磁盘配置创建。不能使用 `zpool add` 命令添加其他磁盘以创建多个镜像顶层虚拟设备，但可以使用 `zpool attach` 命令扩展镜像虚拟设备。
- 不支持 RAID-Z 或条带化配置。
- 根池不能有单独的日志设备。
- 如果您尝试使用不受支持的根池配置，将会看到类似如下的消息：

  ```
  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
  ```

有关安装和引导 ZFS 根文件系统的更多信息，请参见第 4 章 管理 ZFS 根池组件。

创建 RAID-Z 存储池

创建单奇偶校验 RAID-Z 池与创建镜像池基本相同，不同之处是使用 `raidz` 或 `raidz1` 关键字而不是 `mirror`。以下示例说明如何创建一个包含单个 RAID-Z 设备的池，该设备由 5 个磁盘组成：

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

本例说明可以通过设备的缩写名称或全名指定磁盘。`/dev/dsk/c5t0d0` 和 `c5t0d0` 指代同一个磁盘。

创建池时，通过使用 `raidz2` 或 `raidz3` 关键字，您可以创建双奇偶校验或三奇偶校验 RAID-Z 配置。例如：

```
# 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
```

目前，ZFS RAID-Z 配置中支持以下操作：

- 向现有 RAID-Z 配置中添加用于其他顶层虚拟设备的另一组磁盘。有关更多信息，请参见“向存储池中添加设备” [44]。
- 只要 RAID-Z 磁盘的大小大于或等于要被替换的设备，便可替换现有镜像配置中的一个或多个磁盘。有关更多信息，请参见“替换存储池中的设备” [55]。

目前，RAID-Z 配置中不支持以下操作：

- 向现有 RAID-Z 配置中附加其他磁盘。
- 从 RAID-Z 配置中分离磁盘（分离由备用磁盘替换的磁盘或需要分离备用磁盘时除外）。
- 不能直接从 RAID-Z 配置中移除非日志或缓存设备。对于此功能，已经申请了 RFE（请求提高）。

有关 RAID-Z 配置的更多信息，请参见“RAID-Z 存储池配置” [32]。

创建使用日志设备的 ZFS 存储池

为了满足对同步事务的 POSIX 要求，提供了 ZFS 意图日志 (ZFS intent log, ZIL)。例如，数据库通常要求其事务在从系统调用中返回时应该在稳定的存储设备上。NFS 和其他应用程序也可以使用 `fsync()` 来确保数据的稳定性。

缺省情况下，从主池中的块分配 ZIL。但是，通过使用单独的意图日志设备（如使用 NVRAM 或专用磁盘）可能会获得更佳的性能。

确定设置 ZFS 日志设备是否适合您的环境时，请注意以下几点：

- ZFS 意图日志的日志设备与数据库日志文件无关。
- 通过实施单独的日志设备获得的任何性能改进均取决于设备类型、池的硬件配置，以及应用程序工作负荷。有关初步性能信息，请参见以下博客：

  http://blogs.oracle.com/perrin/entry/slog_blog_or_blogging_on
- 可以取消复制或取消镜像日志设备，但日志设备不支持 RAID-Z。
- 如果未镜像单独的日志设备，且包含日志的设备出现故障，则存储日志块将恢复至存储池。
- 可以将日志设备作为较大存储池的一部分进行添加、替换、移除、附加、分离，以及导入和导出。
- 可以将日志设备附加到现有日志设备，以创建镜像日志设备。此操作等同于在未镜像的存储池中附加设备。
- 日志设备的最小大小与池中每个设备的最小大小 (64 MB) 相同。可能存储在日志设备中的相关的数据量相对较小。提交日志事务（系统调用）时将释放日志块。
- 日志设备的最大大小应大约为物理内存大小的 1/2，因为这是可存储的最大潜在相关的数据量。例如，如果系统的物理内存为 16 GB，请考虑 8 GB 的最大日志设备大小。

创建存储池时或创建存储池以后，您可以设置 ZFS 日志设备。

以下示例显示如何创建使用镜像日志设备的镜像存储池：

```
# 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
```

有关从日志设备故障中进行恢复的信息，请参见例 10-2 “更换出现故障的日志设备”。

创建使用高速缓存设备的 ZFS 存储池

高速缓存设备在主内存和磁盘之间提供了一个进行高速缓存的附加层。使用高速缓存设备，可以最大程度地提高大多数静态内容的随机读取工作的性能。

您可以创建一个使用高速缓存设备来缓存存储池数据的存储池。例如：

```
# 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
```

添加高速缓存设备之后，这些设备中将逐渐填充来自主内存的内容。填充设备可能需要一个小时以上的时间，具体取决于高速缓存设备的大小。可以通过按以下方式使用 `zpool iostat` 命令来监视容量和读取操作：

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

创建池后，可以在池中添加高速缓存设备或从池中删除高速缓存设备。

确定是否创建使用高速缓存设备的 ZFS 存储池时，请注意以下几点：

- 使用高速缓存设备，可以最大程度地提高大多数静态内容的随机读取工作的性能。
- 可以使用 `zpool iostat` 命令监视容量和读取操作。
- 创建池时，可以添加一个或多个高速缓存设备。也可以在创建池后添加或删除高速缓存设备。有关更多信息，请参见例 3-4 “添加和删除高速缓存设备”。

- 高速缓存设备不能镜像或成为 RAID-Z 配置的一部分。
- 如果高速缓存设备发生读取错误，则会向原始存储池设备（它可能是镜像配置或 RAID-Z 配置的一部分）重新发出读取 I/O。高速缓存设备的内容是易失性的，与其他系统高速缓存类似。

创建存储池的注意事项

创建和管理 ZFS 存储池时，请注意以下事项。

- 请勿对属于现有存储池一部分的磁盘重新分区或重新设置标签。如果尝试对根池磁盘重新分区或重新设置标签，可能需要重新安装 OS。
- 创建存储池时，请勿包含来自其他存储池的组件（如文件或卷）。这种配置不受支持，且会发生死锁。
- 使用单个分片或单个磁盘创建的池没有冗余，会面临丢失数据的风险。使用多个分片创建的池如果没有冗余，也会面临丢失数据的风险。使用磁盘中多个分片创建的池比使用整个磁盘创建的池难以管理。
- 创建时未设置 ZFS 冗余（RAID-Z 或镜像）的池只能报告数据不一致性，无法修复数据不一致性。
- 虽然创建池时设置 ZFS 冗余有助于减少因硬件故障而导致的停机时间，但是这样的池仍不可避免地会受硬件故障、电源故障或电缆断开的影响。请确保定期备份您的数据。对非企业级硬件上的池数据定期执行备份很重要。
- 池不能在系统之间共享。ZFS 不是群集文件系统。

显示存储池虚拟设备信息

每个存储池都包含一个或多个虚拟设备。虚拟设备是存储池的内部表示形式，用于描述物理存储器的布局以及存储池的故障特征。因此，虚拟设备表示用于创建存储池的磁盘设备或文件。一个池可以在配置的顶层具有任意数目的虚拟设备，称为顶层 *vdev*。

如果顶层虚拟设备包含两个或更多物理设备，配置将以镜像或 RAID-Z 虚拟设备的形式提供数据冗余。这些虚拟设备由磁盘、磁盘分片或文件构成。备件是一种特殊虚拟设备，用于跟踪一个池的可用热备件。

以下示例说明如何创建一个包含两个顶层虚拟设备（各虚拟设备是由两个磁盘组成的镜像）的池：

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

以下示例说明如何创建包含一个顶层虚拟设备（由 4 个磁盘组成）的池。

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

可以使用 `zpool add` 命令将另一个顶层虚拟设备添加到此池中。例如：

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

非冗余池中使用的磁盘、磁盘分片或文件可用作顶层虚拟设备。存储池通常由多个顶层虚拟设备构成。ZFS 将在池内的所有顶层虚拟设备中以动态方式对数据进行条带化。

可使用 `zpool status` 命令显示 ZFS 存储池中包含的虚拟设备和物理设备。例如：

```
# 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
```

处理 ZFS 存储池创建错误

出现池创建错误可以有许多原因。其中一些原因是显而易见的（如指定的设备不存在），而其他原因则不太明显。

检测使用中的设备

格式化设备之前，ZFS 会首先确定 ZFS 或操作系统的某个其他部分是否正在使用磁盘。如果磁盘正在使用，则可能会显示类似以下的错误：

```
# 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).
```

通过使用 `-f` 选项可以覆盖部分错误，但是无法覆盖大部分错误。以下情况无法使用 `-f` 选项覆盖，您必须手动更正：

挂载的文件系统 磁盘或其中一分片包含当前挂载的文件系统。要更正此错误，请使用 `umount` 命令。

/etc/vfstab 中的文件系统	磁盘包含 /etc/vfstab 文件中列出的文件系统，但当前未挂载该文件系统。要更正此错误，请删除或注释掉 /etc/vfstab 文件中的相应行。
专用转储设备	正在将磁盘用作系统的专用转储设备。要更正此错误，请使用 dumpadm 命令。
ZFS 池的一部分	磁盘或文件是活动 ZFS 存储池的一部分。要更正此错误，请使用 zpool destroy 命令来销毁其他池（如果不再需要）。或者，使用 zpool detach 命令将磁盘与其他池分离。您只能将磁盘从镜像存储池中分离。

以下使用情况检查用作帮助性警告，并可以使用 -f 选项进行覆盖以创建池：

包含文件系统	磁盘包含已知的文件系统，尽管该系统未挂载并且看起来未被使用。
卷的一部分	磁盘是 Solaris Volume Manager 卷的一部分。
导出的 ZFS 池的一部分	磁盘是已导出的或者从系统中手动删除的存储池的一部分。如果是后一种情况，则会将池的状态报告为可能处于活动状态，因为磁盘可能是也可能不是由其他系统使用的网络连接驱动器。覆盖可能处于活动状态的池时请务必谨慎。

以下示例说明如何使用 -f 选项：

```
# 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
```

理想情况是更正错误，而不是使用 -f 选项覆盖错误。

复制级别不匹配

建议不要创建包含不同复制级别的虚拟设备的池。zpool 命令可尝试防止意外创建冗余级别不匹配的池。如果您尝试创建使用此配置的池，将会看到类似如下的错误：

```
# 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
```

可以使用 -f 选项覆盖这些错误，但应避免这种做法。此命令还会发出警告，指明正使用大小不同的设备创建镜像池或 RAID-Z 池。虽然允许这种配置，但冗余结果的不匹配程度会导致较大设备上出现未使用磁盘空间。要求使用 -f 选项覆盖警告。

在预运行模式下创建存储池

尝试创建池可能会以多种方式意外失败，格式化磁盘是一种潜在有害的操作。因此，zpool create 命令提供了一个额外选项 -n，它模拟创建池，但不真正写入设备。此预运行选项执行设备使用中检查和复制级别验证，并报告该过程中出现的任何错误。如果未找到错误，则会显示类似以下的输出：

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

tank
  mirror
    c1t0d0
    c1t1d0
```

如果不实际创建池，则无法检测到某些错误。最常见的示例是在同一配置中两次指定同一设备。不真正写入数据将无法可靠地检测此错误，因此 zpool create -n 命令可以报告操作成功，但不会创建池。

存储池的缺省挂载点

创建池时，顶层文件系统的缺省挂载点是 */pool-name*。此目录必须不存在或者为空。如果目录不存在，则会自动创建该目录。如果该目录为空，则根文件系统会挂载在现有目录的顶层。要使用其他缺省挂载点创建池，请在 -zpool create 命令中使用 m 选项。例如：

```
# 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
```

此命令会创建新池 home 和挂载点为 /export/zfs 的 home 文件系统。

有关挂载点的更多信息，请参见“管理 ZFS 挂载点” [138]。

销毁 ZFS 存储池

池是通过使用 zpool destroy 命令进行销毁的。此命令会销毁池，即使池中包含挂载的数据集也是如此。

```
# zpool destroy tank
```

注意 - 销毁池时请务必小心。请确保确实要销毁池，并始终保留数据副本。如果意外销毁了不该销毁的池，则可以尝试恢复该池。有关更多信息，请参见“恢复已销毁的 ZFS 存储池” [83]。

如果使用 `zpool destroy` 命令销毁池，该池仍可用于导入，如“恢复已销毁的 ZFS 存储池” [83]中所述。这意味着属于该池的磁盘上的机密数据可能仍可用。如果要将已销毁池的磁盘上的数据销毁，必须对已销毁池中的每个磁盘使用类似于 `format` 实用程序的 `analyze->purge` 选项的功能。

使文件系统数据保密的另一种方法是创建加密的 ZFS 文件系统。采用加密文件系统的池被销毁后，即使恢复了已销毁的池，在没有加密密钥的情况下也无法访问其中的数据。有关更多信息，请参见“加密 ZFS 文件系统” [157]。

销毁包含不可用设备的池

销毁池这一操作要求将数据写入磁盘，以指示池不再有效。此状态信息可防止执行导入操作时这些设备作为潜在的池显示出来。在一个或多个设备不可用的情况下，仍可以销毁池。但是，必需的状态信息将不会写入这些不可用的设备。

经过适当修复后，当您创建新池时，这些设备被报告为潜在活动设备。当您搜索池以便导入时，这些设备显示为有效设备。如果池中包含足够多的 `UNAVAIL` 设备，以致于池本身也变为 `UNAVAIL` 状态（这意味着顶层虚拟设备变为 `UNAVAIL`），则此命令将输出一条警告，并且在不使用 `-f` 选项的情况下无法完成操作。此选项是必需的，因为无法打开池，以致无法知道数据是否存储在池中。例如：

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

有关池和设备的运行状况的更多信息，请参见“确定 ZFS 存储池的运行状况” [71]。

有关导入池的更多信息，请参见“导入 ZFS 存储池” [79]。

管理 ZFS 存储池中的设备

“ZFS 存储池的组件” [27]中介绍了有关设备的大多数基本信息。创建池后，即可执行几项任务来管理池中的物理设备。

- “向存储池中添加设备” [44]
- “附加和分离存储池设备” [48]
- “通过拆分镜像 ZFS 存储池创建新池” [50]
- “使存储池中的设备联机和脱机” [53]

向存储池中添加设备

通过添加新的顶层虚拟设备，可以向池中动态添加磁盘空间。此磁盘空间立即可供池中的所有数据集使用。要向池中添加新虚拟设备，请使用 `zpool add` 命令。例如：

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

用于指定虚拟设备的格式与 `zpool create` 命令中使用的虚拟设备格式相同。将对设备进行检查，确定是否正在使用这些设备，此命令在不使用 `-f` 选项的情况下无法更改冗余级别。此命令还支持 `-n` 选项，以便可以执行预运行。例如：

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

此命令语法会将镜像设备 `c3t1d0` 和 `c3t2d0` 添加到 `zeepool` 池的现有配置中。

有关如何执行虚拟设备验证的更多信息，请参见“检测使用中的设备” [40]。

例 3-1 向镜像 ZFS 配置中添加磁盘

在以下示例中，向现有的镜像 ZFS 配置添加了另一个镜像。

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

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

```
      c0t2d0   ONLINE       0      0      0
      c1t2d0   ONLINE       0      0      0

errors: No known data errors
# zpool add tank mirror c0t3d0 c1t3d0
# zpool status tank
pool: tank
state: ONLINE
scrub: none requested
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 向 RAID-Z 配置中添加磁盘

可按类似方式向 RAID-Z 配置中添加其他磁盘。以下示例说明如何将包含一个 RAID-Z 设备（含 3 个磁盘）的存储池转换为包含两个 RAID-Z 设备（各含 3 个磁盘）的存储池。

```
# 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
      c1t2d0   ONLINE       0      0      0
      c1t3d0   ONLINE       0      0      0
      c1t4d0   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 添加和删除镜像日志设备

以下示例说明如何向镜像存储池中添加镜像日志设备。

```
# 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
```

可以将日志设备附加到现有日志设备，以创建镜像日志设备。此操作等同于在未镜像的存储池中附加设备。

您可以使用 zpool remove 命令移除日志设备。通过指定 mirror-1 参数，可以删除上例中的镜像日志设备。例如：

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

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

errors: No known data errors
```

如果池配置仅包含一个日志设备，应通过指定设备名称的方式移除该日志设备。例如：

```
# 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 添加和删除高速缓存设备

可以向 ZFS 存储池中添加高速缓存设备，不再需要时可以删除。

可使用 zpool add 命令添加高速缓存设备。例如：

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

NAME           STATE     READ  WRITE  CKSUM
```

```
tank           ONLINE       0     0     0
   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
```

高速缓存设备不能镜像或成为 RAID-Z 配置的一部分。

可使用 `zpool remove` 命令删除高速缓存设备。例如：

```
# 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
```

目前，`zpool remove` 命令仅支持删除热备件、日志设备和高速缓存设备。可以使用 `zpool detach` 命令删除属于主镜像池配置的设备。非冗余设备和 RAID-Z 设备无法从池中删除。

有关在 ZFS 存储池中使用高速缓存设备的更多信息，请参见“创建使用高速缓存设备的 ZFS 存储池” [38]。

附加和分离存储池设备

除了 `zpool add` 命令外，还可以使用 `zpool attach` 命令将新设备添加到现有镜像设备或非镜像设备中。

如果要附加磁盘以创建镜像根池，请参见如何配置镜像根池（SPARC 或 x86/VTOC） [92]。

如果要替换 ZFS 根池中的磁盘，请参见如何替换 ZFS 根池中的磁盘（SPARC 或 x86/VTOC） [95]。

例 3-5 将双向镜像存储池转换为三向镜像存储池

在本示例中，zeepool 是现有的双向镜像，通过将新设备 c2t1d0 附加到现有设备 c1t1d0 可将其转换为三向镜像。

```
# 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
```

如果现有设备是三向镜像的一部分，则附加新设备将创建四向镜像，依此类推。在任一情况下，新设备都会立即开始重新同步。

例 3-6 将非冗余 ZFS 存储池转换为镜像 ZFS 存储池

此外，还可以通过使用 zpool attach 命令将非冗余存储池转换为冗余存储池。例如：

```
# 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
```

可以使用 `zpool detach` 命令从镜像存储池中分离设备。例如：

```
# zpool detach zeepool c2t1d0
```

然而，如果不存在数据的其他有效副本，此操作将失败。例如：

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

通过拆分镜像 ZFS 存储池创建新池

通过使用 `zpool split` 命令，可以将镜像 ZFS 存储池快速克隆为备份池。

可以使用 `zpool split` 命令从镜像 ZFS 存储池中分离一个或多个磁盘，以使用分离的磁盘创建新池。新池的内容与原镜像 ZFS 存储池完全相同。

注 - 有关使用 `zpool split` 命令拆分 ZFS 池的更多过程，请在 My Oracle Support (https://support.oracle.com) 登录您的帐户，查看 "*How to Use 'zpool split' to Split an rpool (Doc ID 1637715.1)*"（如何使用 "zpool split" 拆分 rpool（文档 ID 1637715.1））。

缺省情况下，对镜像池执行 `zpool split` 操作将分离最后一个磁盘以用于新创建的池。拆分操作完成后，导入新池。例如：

```
# 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
```

通过在 `zpool split` 命令中指定，您可以标识哪个磁盘应用于新创建的池。例如：

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

发生实际的拆分操作之前，内存中的数据被刷新到镜像磁盘。数据刷新后，将磁盘从池中分离，并将其指定给一个新池 GUID。将生成新池 GUID，以便能在拆分该池的系统上导入该池。

如果要拆分的池具有非缺省的文件系统挂载点，并且已在同一系统上创建了新池，必须使用 `zpool split -R` 选项标识新池的备用根目录，使现有挂载点不会发生冲突。例如：

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

如果不使用 `zpool split -R` 选项，当您试图使用 `-R` 选项导入新池时，挂载点会发生冲突。如果在不同的系统上创建新池，则没必要指定备用根目录，除非发生挂载点冲突。

使用 `zpool split` 功能之前，请检查以下几点：

- 对于 RAID-Z 配置或者由多个磁盘组成的非冗余池，此功能不可用。
- 尝试 `zpool split` 操作之前，数据和应用程序操作应停顿。
- 如果重新同步正在进行中，则无法拆分池。
- 拆分由两到三个磁盘组成的镜像池是最佳的，其中原池中的最后一个磁盘将用于新创建的池。然后，可以使用 `zpool attach` 命令重新创建原镜像存储池，或者将新创建

的池转换为镜像存储池。目前无法通过一个 zpool split 操作从现有镜像池创建新镜像池，因为新（拆分的）池是非冗余的。

- 如果现有池是一个三向池，则在拆分操作之后，新池将包含一个磁盘。如果现有池是一个由两个磁盘组成的双向池，则结果是由两个磁盘组成的两个非冗余池。您必须再附加两个磁盘，以将非冗余池转换为镜像池。
- 在拆分操作期间，保持数据冗余性的一个好方法是拆分由三个磁盘组成的镜像存储池，这样在拆分操作之后，原池由两个镜像磁盘组成。
- 在拆分镜像池之前，请确认硬件配置正确无误。有关确认硬件高速缓存刷新设置的相关信息，请参见“一般系统做法” [267]。

例 3-7 拆分镜像 ZFS 存储池

在以下示例中，对名为 mothership 的镜像存储池（包含三个磁盘）进行拆分。分割产生了两个池，即，镜像池 mothership（包含两个磁盘）和新池 luna（包含一个磁盘）。每个池的内容完全相同。

可以将池 luna 导入另一个系统中以进行备份。备份完成后，可以将池 luna 销毁，并将其磁盘重新附加到 mothership。然后，可以重复执行这一过程。

```
# 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
```

使存储池中的设备联机和脱机

使用 ZFS 可使单个设备脱机或联机。硬件不可靠或无法正常工作时（假定该情况只是暂时的），ZFS 会继续对设备读写数据。如果该情况不是暂时的，您可以指示 ZFS 通过使设备脱机来忽略该设备。ZFS 不会向脱机设备发送任何请求。

注 - 设备无需脱机即可进行替换。

使设备脱机

可以使用 `zpool offline` 命令使设备脱机。如果设备是磁盘，则可以使用路径或短名称指定设备。例如：

```
# zpool offline tank c0t5000C500335F95E3d0
```

使设备脱机时，请注意以下几点：

- 不能将池脱机到变为 `UNAVAIL` 的程度。例如，不能使 `raidz1` 配置中的两个设备脱机，也不能使顶层虚拟设备脱机。

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

- 缺省情况下，`OFFLINE` 状态是持久性的。重新引导系统时，设备会一直处于脱机状态。

  要暂时使设备脱机，请使用 `zpool offline -t` 选项。例如：

  ```
  # zpool offline -t tank c1t0d0
  ```

  重新引导系统时，此设备会自动恢复到 `ONLINE` 状态。

- 当设备脱机时，它不会从存储池中分离出来。如果尝试使用其他池中的脱机设备，那么即使在销毁原始池之后，也会显示类似于以下内容的消息：

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

如果要在销毁原始存储池之后使用其他存储池中的脱机设备，请先使该设备恢复联机，然后销毁原始存储池。

要在保留原存储池的同时使用其他存储池中的设备，还有一种方法是用另一个类似的设备替换原存储池中的现有设备。有关替换设备的信息，请参见“替换存储池中的设备” [55]。

查询池的状态时，已脱机的设备以 OFFLINE 状态显示。有关查询池的状态的信息，请参见“查询 ZFS 存储池的状态” [64]。

有关设备运行状况的更多信息，请参见“确定 ZFS 存储池的运行状况” [71]。

使设备联机

使设备脱机后，可以使用 zpool online 命令使其恢复联机。例如：

```
# zpool online tank c0t5000C500335F95E3d0
```

使设备联机时，已写入池中的任何数据都将与最新可用的设备重新同步。请注意，不能通过使设备联机来替换磁盘。如果使设备脱机，然后替换该设备并尝试使其联机，则设备将一直处于 UNAVAIL 状态。

如果尝试使 UNAVAIL 设备联机，则会显示类似于以下内容的消息：

```
# zpool online tank c0t5000C500335DC60Fd0
warning: device 'c0t5000C500335DC60Fd0' onlined, but remains in faulted state
use 'zpool clear' to restore a faulted device
```

您还可能会看到故障磁盘消息显示在控制台上，或者写入 /var/adm/messages 文件中。例如：

```
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.
```

有关更换故障设备的更多信息，请参见“解决缺少设备或设备被移除的问题” [244]。

您可以使用 zpool online -e 命令扩展 LUN。 缺省情况下，添加至池中的 LUN 不会扩展到其完整大小，除非启用了 autoexpand 池属性。使用 zpool online -e 命令可以自动扩展 LUN，即使 LUN 已经联机或者 LUN 目前脱机。例如：

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

清除存储池设备错误

如果设备发生了在 `zpool status` 输出中列出错误的故障，并因而脱机，您可以使用 `zpool clear` 命令清除错误计数。如果池中的设备失去连接，然后连接恢复，您也需要清除这些错误。

如果不指定任何参数，则此命令将清除池中的所有设备错误。例如：

```
# zpool clear tank
```

如果指定了一个或多个设备，此命令仅清除与指定设备关联的错误。例如：

```
# zpool clear tank c0t5000C500335F95E3d0
```

有关清除 `zpool` 错误的更多信息，请参见“清除瞬态或持久性设备错误” [249]。

替换存储池中的设备

可以使用 `zpool replace` 命令替换存储池中的设备。

如果使用冗余池中同一位置的另一设备以物理方式替换某一设备，则可能只需标识被替换的设备。在某些硬件上，ZFS 会认为该设备是同一位置的不同磁盘。例如，要移除出现故障的磁盘 (`c1t1d0`) 并在同一位置替换该磁盘，请使用以下语法：

```
# zpool replace tank c1t1d0
```

如果要使用位于不同物理位置的磁盘替换存储池中的设备，必须同时指定两个设备。例如：

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

如果要替换 ZFS 根池中的磁盘，请参见如何替换 ZFS 根池中的磁盘（SPARC 或 x86/VTOC） [95]。

下面是替换磁盘的基本步骤：

1. 使用 `zpool offline` 命令使磁盘脱机（如有必要）。
2. 移除要替换的磁盘。
3. 插入替换磁盘。
4. 查看 `format` 输出，确定替换磁盘是否可见。

另外，检查设备 ID 是否已更改。如果替换磁盘具有 WWN，则故障磁盘的设备 ID 已更改。

5. 使 ZFS 分辨出磁盘已替换。例如：

```
# zpool replace tank c1t1d0
```

如果替换磁盘具有不同的设备 ID，如以上所示，请包含该新的设备 ID。

```
# zpool replace tank c0t5000C500335FC3E7d0 c0t5000C500335BA8C3d0
```

6. 如有必要，使用 `zpool online` 命令使磁盘联机。
7. 让 FMA 知道设备已被替换。

在 `fmadm faulty` 输出的 `Affects:` 部分中找到 `zfs://pool=name/vdev=guid` 字符串，并将该字符串作为参数提供给 `fmadm repaired` 命令。

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

在具有 SATA 磁盘的某些系统上，必须先取消配置磁盘才能使其脱机。如果在该系统的同一插槽位置替换磁盘，则可以仅执行 `zpool replace` 命令，如本节的第一个示例所示。

有关替换 SATA 磁盘的示例，请参见例 10-1 “替换 ZFS 存储池中的 SATA 磁盘”。

替换 ZFS 存储池中的设备时，请注意以下几点：

- 如果将池属性 `autoreplace` 设置为 `on`，则会自动对在先前属于该池的设备的同一物理位置处找到的任何新设备进行格式化和替换。启用此属性时，无需使用 `zpool replace` 命令。此功能可能并不是在所有硬件类型上都可用。
- 如果在系统运行期间设备或热备件被物理移除，则会提供存储池状态 `REMOVED`。热备用设备（如果有）会替换移除的设备。
- 如果设备被移除后又重新插入，该设备将联机。如果重新插入设备时热备件处于激活状态，则热备件将在联机操作完成时被移除。
- 在移除或插入设备时自动检测依赖于硬件，而且并非在所有平台上都受支持。例如，USB 设备会在插入时自动进行配置。但是，您可能必须使用 `cfgadm -c configure` 命令来配置 SATA 驱动器。
- 系统会定期检查热备件，以确保它们处于联机状态并可供使用。
- 替换设备的大小必须等于或大于镜像或 RAID-Z 配置中最小磁盘的大小。
- 将大小大于要替换设备的替换设备添加到池中后，它不会自动扩展到完整大小。池属性 `autoexpand` 的值决定当磁盘添加到池中时，是否将替换 LUN 扩展到其完整大小。缺省情况下，`autoexpand` 属性禁用。您可以在将较大的 LUN 添加到池中之前或之后，启用此属性以扩展 LUN 大小。

  在以下示例中，两个 72-GB 磁盘替换镜像池中的两个 16-GB 磁盘。确保第一个设备完全重新同步，然后再尝试替换第二个设备。磁盘替换后，启用 `autoexpand` 属性以扩展到完整的磁盘大小。

```
# 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  -
```

- 替换较大池中的多个磁盘需要较长时间，这是因为需要将数据重新同步到新磁盘。此外，还可以考虑在两次磁盘替换操作之间运行 `zpool scrub` 命令，以确保可供替换的设备可以正常运行，并且正确写入数据。
- 如果已使用热备件自动替换了故障磁盘，则您可能需要在替换故障磁盘后分离该热备件。可以使用 `zpool detach` 命令从镜像池或 RAID-Z 池中分离备件。有关分离热备件的信息，请参见“在存储池中激活和取消激活热备件” [59]。

有关替换设备的更多信息，请参见“解决缺少设备或设备被移除的问题” [244]和“更换或修复损坏的设备” [248]。

在存储池中指定热备件

借助热备件功能，您可以确定哪些磁盘可用来替换存储池中已发生故障或失败的磁盘。指定一个设备作为热备件，意味着在池中该设备不是活动设备，但是如果池中的某一活动设备发生故障，热备件将自动替换该故障设备。

可通过以下方式将设备指定为热备件：

- 使用 `zpool create` 命令创建池时。
- 使用 `zpool add` 命令创建池之后。

以下示例说明创建池时如何将某些设备指定为热备件：

```
# 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
```

以下示例说明创建池之后如何通过向池中添加设备来指定热备件：

```
# 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
```

可使用 `zpool remove` 命令从存储池中删除热备件。例如：

```
# 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
```

如果存储池当前正在使用热备件，则不能将其删除。

使用 ZFS 热备件时，请注意以下几点：

- 目前，`zpool remove` 命令只能用来删除热备件、高速缓存设备和日志设备。
- 要添加磁盘作为热备件，热备件的大小必须等于或大于池中最大磁盘的大小。允许向池中添加更小的磁盘作为备件。但是，当自动激活或使用 `zpool replace` 命令激活较小的备用磁盘时，操作将失败，并显示类似以下内容的错误：

  `cannot replace` *disk3* `with` *disk4*`: device is too small`
- 不能跨系统共享备件。

  不能配置多个系统共享一个备件，即使这些系统能够访问该磁盘也是如此。如果将磁盘配置为在多个池之间共享，则只能有一个系统控制这些池。
- 请记住，如果在同一系统上的两个数据池之间共享了一个备件，则必须协调该备件在两个池之间的使用情况。例如，池 B 正在使用该备件，而池 A 已导出。池 B 可能在池 A 导出的情况下无意中使用了该备件。当池 A 导入时，由于两个池都使用了同一个磁盘，可能会发生数据损坏。即使磁盘对多个池来讲是共享备件，也可能会存在导致池问题的情况，因此，请注意此类极端情况。
- 不要在根池和数据池之间共享备件。

在存储池中激活和取消激活热备件

可通过以下方式激活热备件：

- 手动替换 – 通过 `zpool replace` 命令用热备件替换存储池中的故障设备。
- 自动替换 – 检测到故障后，FMA 代理将检查池中是否有任何可用的热备件。如果有，将使用可用备件替换故障设备。

  如果当前正在使用的热备件发生故障，FMA 代理将分离该备件，从而取消替换。然后，代理将尝试用另一个热备件（如果有）替换该设备。目前，由于 ZFS 诊断引擎仅在设备从系统中消失时才会产生故障信息，因此此功能受到限制。

如果将故障设备物理替换为活动备件，则可以使用 `zpool detach` 命令分离该备件，从而重新激活原设备。如果将 `autoreplace` 池属性设置为 `on`，则在插入新设备并完成联机操作后，该备件会自动分离并回到备件池。

如果热备件可用，将自动替换 `UNAVAIL` 设备。例如：

```
# 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                       ONLINE       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
```

目前，取消激活热备件的方法有以下几种：

- 从存储池中删除热备件。
- 物理替换有故障的磁盘后，分离热备件。请参见例 3-8 “替换故障磁盘后分离热备件”。
- 临时或永久性地换入另一热备件。请参见例 3-9 “分离故障磁盘并使用热备件”。

例 3-8 替换故障磁盘后分离热备件

在本示例中，物理替换了故障磁盘 (`c0t5000C500335DC60Fd0`) 并使用 `zpool replace` 命令通知 ZFS。

```
# 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
```

如有必要，您可以使用 `zpool detach` 命令使热备件回到备件池。例如：

```
# zpool detach zeepool c0t5000C500335E106Bd0
```

例 3-9 分离故障磁盘并使用热备件

如果您想临时或永久性地换入当前正在替换故障磁盘的热备件以替换磁盘，请分离原（故障）磁盘。故障磁盘完成替换后，可以将其再添加到存储池用作备件。例如：

```
# 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                     ONLINE       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
   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                 DEGRADED     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                 DEGRADED     0      0      0
      c0t5000C500335BD117d0 ONLINE       0      0      0
      c0t5000C500335E106Bd0 ONLINE       0      0      0
   spares
      c0t5000C500335DC60Fd0    AVAIL

errors: No known data errors
```

在替换磁盘并分离备件后，通知 FMA 磁盘已修复。

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

管理 ZFS 存储池属性

您可以使用 `zpool get` 命令来显示池属性信息。例如：

```
# 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
```

可以使用 `zpool set` 命令设置存储池属性。例如：

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

如果尝试在 100% 全满的池上设置池属性，则会显示类似于以下内容的消息：

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

有关预防池空间容量问题的信息，请参见第 11 章 建议的 Oracle Solaris ZFS 做法。

表 3-1 ZFS 池属性说明

属性名称	类型	缺省值	说明
`allocated`	字符串	N/A	只读值，表示池中物理分配的存储空间量。
`altroot`	字符串	`off`	标识备用根目录。如果进行了设置，则该目录会被前置到池中的任何挂载点。在检查未知池（如果无法信任其中的挂载点）时，或在典型路径无效的备用引导环境中，可以使用此属性。
`autoreplace`	布尔型	`off`	控制设备的自动替换。如果设置为 `off`，则必须使用 `zpool replace` 命令启动设备替换。如果设置为 `on`，则会自动对在先前属于池的设备所在物理位置处找到的任何新设备进行格式化和替换。该属性缩写为 `replace`。
`bootfs`	布尔型	N/A	标识根池的缺省可引导文件系统。此属性通常由安装程序进行设置。
`cachefile`	字符串	N/A	控制缓存池配置信息的位置。系统引导时会自动导入高速缓存中的所有池。但是，安装和群集环境可能需要将此信息高速缓存到不同的位置，以便不会自动导入池。可设置此属性以将池配置信息高速缓存于不同位置。以后可以使用 `zpool import -c` 命令导入此信息。大多数 ZFS 配置不使用此属性。
`capacity`	数字	N/A	只读值，表示已用池空间的百分比。 此属性的缩写为 `cap`。
`dedupditto`	字符串	N/A	设置一个阈值，如果已进行重复数据删除的块的引用计数超过了该阈值，则将自动存储该块的另一个重复副本。

属性名称	类型	缺省值	说明
dedupratio	字符串	N/A	池实现的重复数据删除比（只读），表示为一个乘数。
delegation	布尔型	on	控制是否可以向非特权用户授予为某个文件系统定义的访问权限。有关更多信息，请参见第 8 章 Oracle Solaris ZFS 委托管理。
failmode	字符串	wait	控制发生灾难性池故障时的系统行为。这种情况通常是由于失去与底层存储设备的连接或池中所有设备出现故障而导致的。这种事件的行为由下列值之一决定： ■ wait – 阻止所有对池的 I/O 请求，直到设备连接恢复且使用 zpool clear 命令清除错误为止。这种状态下，对池的 I/O 操作被阻止，但读操作可能会成功。在设备问题得到解决之前，池一直处于 wait 状态。 ■ continue – 对任何新的写入 I/O 请求返回 EIO 错误，但允许对其余任何运行状况良好的设备执行读取操作。任何未提交到磁盘的写入请求都会被阻止。重新连接或替换设备后，必须使用 zpool clear 命令清除错误。 ■ panic–向控制台打印一则消息并产生系统故障转储。
free	字符串	N/A	只读值，表示池中未分配的块数。
guid	字符串	N/A	只读属性，表示池的唯一标识符。
health	字符串	N/A	只读属性，表示池的当前运行状况（例如 ONLINE、DEGRADED、SUSPENDED、REMOVED 或 UNAVAIL）。
listshares	字符串	off	控制使用 zfs list 命令是否可显示此池中的共享信息。缺省值为 off（关闭）。
listsnapshots	字符串	off	控制使用 zfs list 命令是否可显示与此池有关的快照信息。如果禁用了此属性，可以通过 zfs list -t snapshot 命令显示快照信息。
readonly	布尔型	off	指示某个池是否可以修改。仅当池已在只读模式下导入时才启用此属性。如果已启用，则在以读写模式重新导入池之前，任何仅存在于意图日志中的同步数据将不可访问。
size	数字	N/A	只读属性，表示存储池总大小。
version	数字	N/A	表示池的当前盘上版本。尽管在为了实现向后兼容性而需要一个特定版本时可以使用此属性，但首选的池更新方法是使用 zpool upgrade 命令。可以将此属性设置为 1 与 zpool upgrade -v 命令所报告的当前版本之间的任何数值。

查询 ZFS 存储池的状态

zpool list 命令提供了多种方法来请求有关池状态的信息。可用信息通常分为以下三个类别： 基本使用情况信息、I/O 统计信息和运行状况。本节介绍了所有这三种类型的存储池信息。

- “显示有关 ZFS 存储池的信息” [65]
- “查看 ZFS 存储池的 I/O 统计信息” [69]
- “确定 ZFS 存储池的运行状况” [71]

显示有关 ZFS 存储池的信息

可以使用 `zpool list` 命令显示有关池的基本信息。

显示有关所有存储池或某个特定池的信息

不带任何参数时，`zpool list` 命令显示有关系统上所有池的下列信息。

```
# 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     -
```

此命令输出显示以下信息：

`NAME` 池的名称。

`SIZE` 池的总大小，等于所有顶层虚拟设备大小的总和。

`ALLOC` 分配给所有数据集和内部元数据的物理空间量。请注意，此数量与在文件系统级别报告的磁盘空间量不同。

有关确定可用文件系统空间的更多信息，请参见“ZFS 磁盘空间记帐” [18]。

`FREE` 池中未分配的空间量。

`CAP (CAPACITY)` 已用磁盘空间量，以总磁盘空间的百分比表示。

`HEALTH` 池的当前运行状况。

有关池运行状况的更多信息，请参见“确定 ZFS 存储池的运行状况” [71]。

`ALTROOT` 池的备用根（如有）。

有关备用根池的更多信息，请参见“通过备用根位置使用 ZFS 池” [235]。

通过指定池名称，您还可以为特定池收集统计信息。例如：

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

可以使用 `zpool list` 的时间间隔和计数选项收集一定时期内的统计信息。此外，使用 `-T` 选项可以显示时间戳。例如：

```
# 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  -
```

按物理位置显示池设备

可以使用 `zpool status -l` 选项显示有关池设备的物理位置的信息。当需要以物理方式移除或替换磁盘时，查看物理位置信息很有帮助。

此外，可以使用 `fmadm add-alias` 命令来引入磁盘别名，磁盘别名可帮助您识别磁盘在您的环境中的物理位置。例如：

```
# 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
```

显示特定的存储池统计信息

可以使用 -o 选项请求特定的统计信息。使用此选项可以生成定制报告或快速列出相关信息。例如，要仅列出每个池的名称和大小，可使用以下语法：

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

列名称与“显示有关所有存储池或某个特定池的信息” [65]中列出的属性相对应。

使用脚本处理 ZFS 存储池输出

`zpool list` 命令的缺省输出目的在于提高可读性，因此不能轻易用作 shell 脚本的一部分。为了便于在程序中使用该命令，可以使用 `-H` 选项以便不显示列标题，并使用制表符而不是空格分隔字段。例如，要请求系统中所有池的名称列表，可以使用以下语法：

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

以下是另一个示例：

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

显示 ZFS 存储池命令历史记录

ZFS 会自动记录成功修改池状态信息的 `zfs` 和 `zpool` 命令。使用 `zpool history` 命令可显示此信息。

例如，以下语法显示了根池的命令输出：

```
# 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
```

您可以使用有关系统的类似输出来确定对错误状况进行故障排除时所执行的确切 ZFS 命令。

历史记录日志有如下特点：

- 不能禁用日志。
- 日志持久保存在磁盘上，这意味着系统重新引导后，将保存日志。
- 日志作为环形缓冲区来实现。最小大小为 128 KB。最大大小为 32 MB。
- 对于较小的池，日志最大大小的上限设置为池大小的 1%，而池大小是在创建池时确定的。
- 日志无需任何管理，这意味着不需要调整日志大小或更改日志位置。

要确定特定存储池的命令历史记录，请使用类似以下内容的语法：

```
# 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
```

可使用 `-l` 选项显示长格式（包括用户名、主机名和执行操作的区域）。例如：

```
# 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]
```

可使用 `-i` 选项显示可用于诊断目的的内部事件信息。例如：

```
# 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 ...
```

查看 ZFS 存储池的 I/O 统计信息

要请求池或特定虚拟设备的 I/O 统计信息，请使用 `zpool iostat` 命令。与 `iostat` 命令类似，此命令也可以显示所有 I/O 活动的静态快照，以及每个指定时间间隔的更新统计信息。可以报告以下统计信息：

`alloc capacity`
: 当前存储在池或设备中的数据量。由于具体的内部实现的原因，此数量与可供实际文件系统使用的磁盘空间量有少量差异。

  有关池空间与数据集空间之间的差异的更多信息，请参见“ZFS 磁盘空间记帐” [18]。

`free capacity`
: 池或设备中的可用磁盘空间量。与 `used` 统计信息一样，该空间量与可供数据集使用的磁盘空间量也有少量差异。

`read operations`
: 发送到池或设备的读取 I/O 操作数，包括元数据请求。

`write operations`
: 发送到池或设备的写入 I/O 操作数。

`read bandwidth`
: 所有读取操作（包括元数据）的带宽，以每秒单位数表示。

`write bandwidth`
: 所有写入操作的带宽，以每秒单位数表示。

列出池范围的 I/O 统计信息

如果不使用任何选项，则 `zpool iostat` 命令会显示自引导以来系统中所有池的累积统计信息。例如：

```
# 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
```

```
----------  -----  -----  -----  -----  -----  -----
```

由于这些统计信息是自引导以来累积的，因此，如果池相对空闲，则带宽可能显示为较低。通过指定时间间隔，可以请求查看更准确的当前带宽使用情况。例如：

```
# 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
```

在以上示例中，此命令每隔两秒显示一次池 `tank` 的使用情况统计信息，直到按 Ctrl-C 组合键为止。或者，可以再指定一个 `count` 参数，该参数可使命令在重复执行指定的次数之后终止。

例如，`zpool iostat 2 3` 每隔两秒列显一次摘要信息，重复三次，共六秒。如果仅有一个池，则会在连续的行上显示统计信息。如果存在多个池，则用附加虚线分隔每次重复，以提供直观的分隔效果。

列出虚拟设备 I/O 统计信息

除了池范围的 I/O 统计信息外，`zpool iostat` 命令还可以显示虚拟设备的 I/O 统计信息。此命令可用于识别异常缓慢的设备，或者观察 ZFS 生成的 I/O 的分布情况。要请求完整的虚拟设备布局以及所有 I/O 统计信息，请使用 `zpool iostat -v` 命令。例如：

```
# 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
----------  -----  -----  -----  -----  -----  -----
```

查看虚拟设备的 I/O 统计信息时，必须注意以下两点：

- 首先，磁盘空间使用情况统计信息仅适用于顶层虚拟设备。在镜像和 RAID-Z 虚拟设备中分配磁盘空间的方法是特定于实现的，不能简单地表示为一个数字。
- 其次，这些数字可能不会完全按期望的那样累加。具体来说，通过 RAID-Z 设备和通过镜像设备进行的操作不是完全均等的。这种差异在创建池之后即特别明显，因为在

创建池的过程中直接对磁盘执行了大量 I/O，但在镜像级别并没有考虑这些 I/O。随着时间推移，这些数值会逐渐变得相等。不过，损坏的、无响应的或脱机设备也会影响这种对称性。

检查虚拟设备统计信息时，可以使用相同的一组选项（时间间隔和计次）。

还可以显示有关池的虚拟设备的物理位置信息。例如：

```
# 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
.
.
.
```

确定 ZFS 存储池的运行状况

ZFS 提供了一种检查池和设备运行状况的集成方法。池的运行状况是根据其所有设备的状态确定的。使用 `zpool status` 命令可以显示此状态信息。此外，池和设备的潜在故障由 `fmd` 报告，显示在系统控制台上，并记录于 `/var/adm/messages` 文件中。

本节介绍如何确定池和设备的运行状况。本章不介绍如何修复运行不良的池或从其恢复。有关故障排除和数据恢复的更多信息，请参见第 10 章 Oracle Solaris ZFS 故障排除和池恢复。

池的运行状况通过以下四种状态之一来描述：

`DEGRADED`
: 池有一个或多个设备发生故障，但由于使用了冗余配置，数据仍然可用。

`ONLINE`
: 池中的所有设备都正常运行。

SUSPENDED

池正在等待恢复设备连接。在设备问题得到解决之前，SUSPENDED 池一直处于 wait 状态。

UNAVAIL

池的元数据遭到损坏，或者有一个或多个设备不可用，并且没有足够的副本支持其继续运行。

每个池设备都可以处于以下状态之一：

DEGRADED	虚拟设备出现过故障，但仍能工作。此状态在镜像或 RAID-Z 设备缺少一个或多个组成设备时最为常见。池的容错能力可能会受到损害，因为另一个设备中的后续故障可能无法恢复。
OFFLINE	管理员已将设备显式脱机。
ONLINE	设备或虚拟设备处于正常工作状态。尽管仍然可能会出现一些瞬态错误，但是设备在其他方面处于正常工作状态。
REMOVED	系统正在运行时已物理移除了该设备。设备移除检测依赖于硬件，而且并非在所有平台上都受支持。
UNAVAIL	无法打开设备或虚拟设备。在某些情况下，包含 UNAVAIL 设备的池会以 DEGRADED 模式显示。如果顶层虚拟设备的状态为 UNAVAIL，则无法访问池中的任何设备。

池的运行状况是根据其所有顶层虚拟设备的运行状况确定的。如果所有虚拟设备状态都为 ONLINE，则池的状态也为 ONLINE。如果任何一个虚拟设备状态为 DEGRADED 或 UNAVAIL，则池的状态也为 DEGRADED。如果顶层虚拟设备的状态为 UNAVAIL 或 OFFLINE，则池的状态也为 UNAVAIL 或 SUSPENDED。如果池处于 UNAVAIL 或 SUSPENDED 状态，则完全无法访问该池。附加或修复必需的设备后，才能恢复数据。处于 DEGRADED 状态的池会继续运行，但是，如果池处于联机状态，则可能无法实现相同级别的数据冗余或数据吞吐量。

zpool status 命令还提供有关重新同步和清理操作的详细信息。

- 重新同步进度报告。例如：

```
scan: resilver in progress since Wed Jun 20 14:19:38 2012
7.43G scanned
7.43G resilvered at 26.8M/s, 10.35% done, 0h30m to go
```

- 清理进度报告。例如：

```
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
```

- 重新同步完成消息。例如：

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

- 清理完成消息。例如：

```
scan: scrub repaired 0 in 0h11m with 0 errors on Wed Jun 20 15:08:23 2012
```

- 取消正在进行的清理消息。例如：

```
scan: scrub canceled on Wed Jun 20 16:04:40 2012
```

- 清理和重新同步完成消息在系统重新引导后仍存在

基本的存储池运行状况

使用 `zpool status` 命令可以快速查看池运行状态，如下所示：

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

通过在命令语法中指定池名称，可以检查特定池。如下节所述，应检查不处于 `ONLINE` 状态的所有池是否存在潜在的问题。

详细运行状况

使用 -v 选项可以请求更详细的运行状况汇总信息。例如：

```
# 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
```

此输出显示了池处于其当前状态的原因的完整说明，其中包括问题的易读说明，以及指向知识文章的链接（用于了解更多信息）。每篇知识文章都提供了有关从当前问题恢复的最佳方法的最新信息。使用详细的配置信息，您可以确定哪个设备损坏以及如何修复池。

在以上示例中，UNAVAIL 设备应该被替换。如有必要，替换该设备后，请使用 zpool online 命令使设备联机。例如：

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

以上输出表明该设备一直处于降级状态，直到完成任何重新同步操作为止。

如果启用了 autoreplace 属性，则您可能不必使被替换的设备联机。

如果池包含脱机设备，则命令输出将标识有问题的池。例如：

```
# 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 和 WRITE 列提供了在设备上出现的 I/O 错误的计数，而 CKSUM 列则提供了在设备上出现的无法更正的校验和错误的计数。这两种错误计数指示可能的设备故障，并且需要执行更正操作。如果针对顶层虚拟设备报告了非零错误，则表明部分数据可能无法访问。

errors: 字段标识任何已知的数据错误。

在以上示例输出中，脱机设备不会导致数据错误。

有关诊断和修复 UNAVAIL 池和数据的更多信息，请参见第 10 章 Oracle Solaris ZFS 故障排除和池恢复。

收集 ZFS 存储池状态信息

可以使用 zpool status 的时间间隔和计数选项收集一定时期内的统计信息。此外，使用 -T 选项可以显示时间戳。例如：

```
# 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
```

迁移 ZFS 存储池

有时，可能需要在系统之间移动存储池。为此，必须将存储设备与原始系统断开，然后将其重新连接到目标系统。可以通过以下方法完成此任务：以物理方式重新为设备布线，或者使用多端口设备（如 SAN 中的设备）。使用 ZFS 可将池从一个系统中导出，然后将其导入目标系统，即使这些系统采用不同的字节存储顺序 (endianness)。有关在不同存储池（可能驻留在不同的系统中）之间复制或迁移文件系统的信息，请参见“发送和接收 ZFS 数据” [177]。

准备迁移 ZFS 存储池

应显式导出存储池，以表明可随时将其迁移。此操作会将任何未写入的数据刷新到磁盘，将数据写入磁盘以表明导出已完成，并从系统中删除有关池的所有信息。

如果不显式导出池，而是改为手动删除磁盘，则仍可以在其他系统中导入生成的池。但是，可能会丢失最后几秒的数据事务，并且由于设备不再存在，该池在原始系统中可能会显示为处于 UNAVAIL 状态。缺省情况下，目标系统无法导入未显式导出的池。为防止意外导入包含仍在其他系统中使用的网络连接存储器的活动池，此条件是必要的。

导出 ZFS 存储池

要导出池，请使用 zpool export 命令。例如：

```
# zpool export tank
```

此命令将尝试取消挂载池中任何已挂载的文件系统，然后再继续执行。如果无法取消挂载任何文件系统，则可以使用 -f 选项强制取消挂载这些文件系统。例如：

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

执行此命令后，池 tank 在系统中即不再可见。

如果在导出时设备不可用，则无法将设备标识为正常导出。如果之后将某个这样的设备附加到不包含任何工作设备的系统中，则该设备的状态会显示为可能处于活动状态。

如果池中使用 ZFS 卷，则即使使用 -f 选项也无法导出池。要导出带有 ZFS 卷的池，首先确保该卷的所有使用者不再处于活动状态。

有关 ZFS 卷的更多信息，请参见“ZFS 卷” [227]。

确定要导入的可用存储池

从系统中删除池后（通过显式导出或通过强制删除设备），可以将设备附加到目标系统。ZFS 可以处理仅有其中一些设备可用的情况，但池迁移成功与否取决于设备的整体运行状况。此外，没有必要使用相同的设备名称附加设备。ZFS 可检测任何移动的或重命名的设备，并相应地调整配置。要搜索可用的池，请运行不带任何选项的 zpool import 命令。例如：

```
# 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
```

在本例中，池 tank 可用于在目标系统上导入。每个池都由一个名称以及唯一的数字标识符标识。如果有多个同名池可用于导入，则可以使用数字标识符对其进行区分。

与 zpool status 命令输出类似，zpool import 输出也会包括一个知识文章链接，其中包含有关阻止导入池这一问题的修复过程的最新信息。在此示例中，用户可以强制导入

池。但是，如果导入当前正由其他系统通过存储网络使用的池，则可能导致数据损坏和出现紧急情况，因为这两个系统都尝试写入同一存储器。如果池中的某些设备不可用，但是存在足够的冗余数据可确保池可用，则池会显示 DEGRADED 状态。例如：

```
# 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.
```

在本示例中，第一个磁盘已损坏或缺失，但仍可以导入池，这是因为仍可以访问镜像数据。如果存在过多的不可用设备，则无法导入池。

在本示例中，RAID-Z 虚拟设备中缺少两个磁盘，这意味着没有足够的可用冗余数据来重新构建池。在某些情况下，没有足够的设备就无法确定完整的配置。在这种情况下，虽然 ZFS 会尽可能多地报告有关该情况的信息，但是 ZFS 无法确定池中包含的其他设备。例如：

```
# 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.
```

从替换目录导入 ZFS 存储池

缺省情况下，zpool import 命令仅在 /dev/dsk 目录中搜索设备。如果设备存在于其他目录中，或者使用的是文件支持的池，则必须使用 -d 选项搜索其他目录。例如：

```
# 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
```

如果设备存在于多个目录中，则可以指定多个 -d 选项。

导入 ZFS 存储池

确定要导入的池后，即可通过将该池的名称或者其数字标识符指定为 zpool import 命令的参数来将其导入。例如：

```
# zpool import tank
```

如果多个可用池具有相同名称，则必须使用数字标识符指定要导入的池。例如：

```
# 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
```

如果池名称与现有池名称冲突，则可以采用其他名称导入该池。例如：

```
# zpool import dozer zeepool
```

此命令使用新名称 zeepool 导入已导出的池 dozer。新的池名称是持久性的。

注 - 不能直接重命名池。只能在导出和导入池时更改池的名称。

如果池未正常导出，则 ZFS 需要使用 -f 标志，以防止用户意外导入仍在其他系统中使用的池。例如：

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

注 - 请勿尝试将一个系统上处于活动状态的池导入到另一个系统。ZFS 不是本机簇、分布式或平行文件系统，不能从多个不同主机进行并发访问。

通过使用 -R 选项还可以在备用根下导入池。有关备用根池的更多信息，请参见“通过备用根位置使用 ZFS 池” [235]。

导入缺少日志设备的池

缺省情况下，无法导入缺少日志设备的池。但是，可以使用 zpool import -m 命令强制导入缺少日志设备的池。例如：

```
# 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.
```

导入缺少日志设备的池。例如：

```
# 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
```

在附加了缺少的日志设备后，运行 `zpool clear` 命令清除池错误。

缺少镜像日志设备时，也可以尝试进行类似的恢复：例如：

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

cannot import 'dozer': one or more devices is currently unavailable
# zpool import -m dozer
# zpool status dozer
```

```
pool: dozer
state: DEGRADED
status: One or more devices could not be opened.  Sufficient replicas exist for
the pool to continue functioning in a degraded state.
action: Attach the missing device and online it using 'zpool online'.
see: https://support.oracle.com/epmos/faces/KmHome?_adf.ctrl-
state=10oxbvnj5n_4&_afrLoop=1145647522713
scan: scrub repaired 0 in 0h0m with 0 errors on Fri Oct 15 16:51:39 2010
config:

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

在附加了缺少的日志设备后，运行 `zpool clear` 命令清除池错误。

在只读模式下导入池

可以在只读模式下导入池。如果池受损严重而无法访问，此功能也许能使您恢复池中的数据。例如：

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

在只读模式下导入池时，须符合以下条件：

- 所有文件系统和卷均以只读模式挂载。
- 池的事务处理功能被禁用。这也意味着，意图日志 (intent log) 中任何暂停的同步写入操作只有在读写模式下导入池后才启动。
- 只读导入期间，将忽略对池属性的设置尝试。

通过导出再导入池的方法，可以将只读池设置回读写模式。例如：

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

通过特定的设备路径导入池

在本示例中，以下命令通过标识池 `dpool` 的其中一个特定设备 `/dev/dsk/c2t3d0` 来导入该池。

```
# 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
```

即使该池由整个磁盘组成，该命令也必须包括特定设备的分片标识符。

恢复已销毁的 ZFS 存储池

可以使用 `zpool import -D` 命令恢复已销毁的存储池。例如：

```
# 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
```

在此 `zpool import` 输出中，根据以下状态信息，可以确定池 `tank` 为已销毁的池：

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

要恢复已销毁的池，请再次执行 `zpool import -D` 命令，并指定要恢复的池。例如：

```
# 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
```

即使已销毁的池中的某个设备不可用，您仍可以通过加入 -f 选项来恢复已销毁的池。在此情况下，请导入已降级的池，然后尝试修复设备故障。例如：

```
# 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
```

升级 ZFS 存储池

如果您拥有来自先前 Solaris 发行版的 ZFS 存储池，则可使用 `zpool upgrade` 命令升级这些池，以利用当前发行版中的池功能。此外，`zpool status` 命令会在池运行旧版本时通知您。例如：

```
# 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
```

可以使用以下语法来确定有关特殊版本和支持的发行版的其他信息：

```
# 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.
```

然后，可通过运行 `zpool upgrade` 命令来升级所有池。例如：

```
# zpool upgrade -a
```

注 - 如果将池升级到更高的 ZFS 版本，则在运行较早 ZFS 版本的系统中将无法访问该池。

♦ ♦ ♦ 第 4 章

管理 ZFS 根池组件

本章描述如何管理 Oracle Solaris ZFS 根池组件，如附加根池镜像、克隆 ZFS 引导环境以及调整交换和转储设备的大小。

本章包含以下各节：

有关根池恢复的信息，请参见《在 Oracle Solaris 11.2 中使用统一归档文件进行系统恢复和克隆 》。

有关任何最新问题，请参见 Oracle Solaris 11.2 发行说明。

关于管理 ZFS 根池组件

从 Oracle Solaris 11 发行版起，ZFS 成为缺省根文件系统。安装 Oracle Solaris 发行版时请查看以下注意事项：

- **安装** – 您可以通过以下方式安装 ZFS 根文件系统并从中进行引导：
  - Live CD（仅限 x86） – 在单个磁盘上安装 ZFS 根池。在安装期间可以根据您的环境使用 `fdisk` 分区菜单对磁盘分区。
  - 文本安装（SPARC 和 x86） – 在单个磁盘上从介质或通过网络安装 ZFS 根池。在安装期间可以根据您的环境使用 `fdisk` 分区菜单对磁盘分区。
  - 自动化安装程序 (Automated Installer, AI)（SPARC 和 x86） – 自动安装 ZFS 根池。可以使用 AI 清单确定用于 ZFS 根池的磁盘和磁盘分区。
- **交换和转储设备** – 上述所有安装方法会自动在 ZFS 根池的 ZFS 卷中创建交换和转储设备。有关管理 ZFS 交换和转储设备的更多信息，请参见“管理 ZFS 交换和转储设备” [101]。
- **镜像根池配置** – 您可以在自动安装期间配置镜像根池。有关在安装后配置镜像根池的更多信息，请参见如何配置镜像根池（SPARC 或 x86/VTOC） [92]。

- **根池空间管理** – 安装系统后，请考虑对 ZFS 根文件系统设置配额，以防止根文件系统被填满。当前，并未保留一定的 ZFS 根池空间作为整个文件系统的安全网。例如，如果有 68 GB 的磁盘空间用于根池，请考虑对 ZFS 根文件系统 (`rpool/ROOT/solaris`) 设置 67 GB 的配额，从而留出 1 GB 的剩余文件系统空间。有关设置配额的信息，请参见"设置 ZFS 文件系统的配额" [153]。
- **根池迁移或恢复** – 考虑使用 Oracle Solaris 归档实用程序为灾难恢复或迁移创建根池恢复归档文件。有关更多信息，请参阅《在 Oracle Solaris 11.2 中使用统一归档文件进行系统恢复和克隆 》和 `archiveadm(1M)` 手册页。

确定 ZFS 根池要求

请阅读下面介绍 ZFS 根池空间和配置要求的各节。

ZFS 根池空间要求

安装系统时，交换卷和转储卷的大小取决于物理内存量。可引导的 ZFS 根文件系统的最小池空间量取决于物理内存量、可用的磁盘空间以及要创建的引导环境 (boot environment, BE) 的数量。

请查看以下 ZFS 存储池空间要求：

- 有关不同安装方法的内存要求的说明，请参见《《Oracle Solaris 11.2 发行说明》》。
- 建议至少使用 7-13 GB 的磁盘空间。空间的使用情况如下所述：
  - **交换区域和转储设备** – Solaris 安装程序创建的交换和转储卷的缺省大小因系统上的内存量和其他变量而异。转储设备大小约为物理内存大小的一半或更大，具体取决于系统所进行的活动。

    安装期间或安装后，可以将交换卷和转储卷的大小调整为所选择的大小，只要新的大小可支持系统运行。有关更多信息，请参见"调整 ZFS 交换和转储设备的大小" [102]。
  - **引导环境 (boot environment, BE)** – ZFS BE 大约为 4-6 GB。从另一个 ZFS BE 克隆的每个 ZFS BE 不需要额外的磁盘空间。请注意，当 BE 更新时，BE 大小将增大（具体取决于更新）。同一根池中的所有 ZFS BE 都使用相同的交换和转储设备。
  - **Oracle Solaris OS 组件** – 根文件系统中作为 OS 映像的一部分的所有子目录（除 `/var` 之外）必须都位于根文件系统中。此外，除了交换和转储设备之外，所有其他 Solaris OS 组件必须驻留在根池。

ZFS 根池配置要求

请查看以下 ZFS 存储池配置要求：

- 大多数情况下，在具有 GPT 感知固件的基于 SPARC 的系统或基于 x86 的系统上，要用作根池的磁盘可以具有 EFI (GPT) 标签。或者，在不具有 GPT 感知固件的 SPARC 系统上应用 SMI (VTOC) 标签。有关 EFI (GPT) 标签的信息，请参见“使用 ZFS 存储池中的磁盘” [27]。
- 如果存在 SMI (VTOC) 标签磁盘，则池必须存在于某个磁盘分片上或若干被镜像的磁盘分片上。或者，如果根池磁盘带有 EFI (GPT) 标签，则池可以存在于整个磁盘上或若干个镜像的整个磁盘上。如果在 `beadm` 操作期间尝试使用不支持的池配置，将会显示类似于以下内容的消息：

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

  有关支持的 ZFS 根池配置的详细说明，请参见“创建 ZFS 根池” [35]。
- 在基于 x86 的系统上，磁盘必须包含 Solaris `fdisk` 分区。`fdisk` 分区是安装基于 x86 的系统时自动创建的。有关 Solaris `fdisk` 分区的更多信息，请参见《在 Oracle Solaris 11.2 中管理设备 》中的“使用 fdisk 选项”。
- 在自动安装过程中，可以在根池上设置池属性或文件系统属性。根池不支持 `gzip` 压缩算法。
- 通过初始安装创建了根池后，请勿对根池重命名。重命名根池可能会导致系统无法引导。
- 请勿将精简置备的 VMware 设备用作根池设备。

管理 ZFS 根池

以下各节提供了关于安装和更新 ZFS 根池以及配置镜像根池的信息。

安装 ZFS 根池

查看以下安装方法确定如何安装 ZFS 根池：

- Live CD 安装方法在一个磁盘上安装缺省 ZFS 根池。
- AI 安装程序提供了一定的灵活性：可以在缺省引导磁盘上安装 ZFS 根池，也可以在标识的目标磁盘上安装 ZFS 根池。您可以指定逻辑设备（如 `c1t0d0`）或物理设备路径。此外，可以使用 MPxIO 标识符或设备 ID 来表示要用于安装的设备。
- 使用自动化安装 (automated installation, AI) 方法时，可以创建 AI 清单来标识用于 ZFS 根池的磁盘或镜像磁盘。例如，此缺省清单代码段中的以下关键字在两个磁盘上安装镜像根池：

  ```
  <target>
  <disk whole_disk="true" in_zpool="rpool" in_vdev="mirrored">
  <disk_name name="c1t0d0" name_type="ctd"/>
  </disk>
  ```

```
<disk whole_disk="true" in_zpool="rpool" in_vdev="mirrored">
<disk_name name="c2t0d0" name_type="ctd"/>
</disk>
<logical>
<zpool name="rpool" is_root="true">
<vdev name="mirrored" redundancy="mirror"/>
<!--
```

例如，如果要在缺省清单中设置池属性，您应该在文件系统关键字后、be_name 关键字前包含 pool_options 关键字，如下所示：

```
 -->
<filesystem name="export" mountpoint="/export"/>
<filesystem name="export/home"/>
<pool_options>
<option name="listsnaps" value="on"/>
</pool_options>
<be name="solaris"/>
</zpool>
</logical>
</target>
```

在上述语法中，在根池上启用了 listsnaps 池属性。

安装后，可查看 ZFS 存储池和文件系统信息，具体内容会因安装类型和定制选项而有所不同。例如：

```
# 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  -
```

查看 ZFS BE 信息。例如：

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

在上面的输出中，Active 字段指示 BE 当前是否处于活动状态。N 表示当前处于活动状态；R 表示重新引导时处于活动状态；NR 表示在当前以及重新引导时均处于活动状态。

▼ 如何更新 ZFS 引导环境

缺省 ZFS 引导环境 (boot environment, BE) 的缺省名称为 solaris。可以使用 beadm list 命令来标识 BE。例如：

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

在上面的输出中，NR 表示 BE 当前处于活动状态，且在重新引导时也将是活动 BE。

可以使用 pkg update 命令来更新 ZFS 引导环境。如果使用 pkg update 命令更新 ZFS BE，将自动创建并激活一个新 BE（除非对现有 BE 所作的更新极少）。

1. 更新 ZFS BE。

```
# pkg update


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

将自动创建并激活一个新 BE solaris-1。

还可以在更新流程之外创建并激活备份 BE。

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

2. 重新引导系统来完成 BE 激活。然后确认 BE 状态。

```
# 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. 如果在引导新的 BE 时出现错误，请激活并引导回以前的 BE。

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

▼ 如何挂载备用 BE

为实现恢复，可能需要从其他 BE 复制或访问文件。

1. 成为管理员。
2. 挂载备用 BE。

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

3. 访问该 BE。

```
# 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. 在使用完备用 BE 后将其卸载。

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

▼ 如何配置镜像根池（SPARC 或 x86/VTOC）

如果在自动安装期间未配置镜像根池，在安装之后也可以轻松地配置镜像根池。

有关替换根池中磁盘的信息，请参见如何替换 ZFS 根池中的磁盘（SPARC 或 x86/VTOC） [95]。

1. 显示当前根池的状态。

```
# 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. 如有必要，准备另一个要附加到根池的磁盘。
   - SPARC：确认该磁盘具有 SMI (VTOC) 磁盘标签和分片 0。如果您需要重新设置磁盘标签，并创建分片 0，请参见《在 Oracle Solaris 11.2 中管理设备 》中的“如何替换 ZFS 根池 (VTOC)”。
   - x86：确认该磁盘具有 `fdisk` 分区、SMI 磁盘标签和分片 0。如果您需要对磁盘重新分区，并创建分片 0，请参见《在 Oracle Solaris 11.2 中管理设备 》中的“修改分片或分区”。

3. 附加另一个磁盘，以配置镜像根池。

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

   正确的磁盘标签和引导块将被自动应用。

4. 查看根池状态，确认重新同步已完成。

```
# 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
938M resilvered at 46.9M/s, 7.86% done, 0h3m to go
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)
```

   在上面的输出中，重新同步过程未完成。当您看到类似如下的消息时，说明重新同步已完成。

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

5. 如果要附加较大的磁盘，请设置池的 **`autoexpand`** 属性，以扩展池的大小。

   确定当前的 `rpool` 池大小：

```
# 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
```

查看扩展后的 rpool 池大小：

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

6. 验证您是否可以从新磁盘成功引导。

▼ 如何配置镜像根池 (x86/EFI (GPT))

在大多数情况下，在基于 x86 的系统上 Oracle Solaris 11.1 发行版缺省安装 EFI (GPT) 标签。

如果在自动安装期间未配置镜像根池，在安装之后也可以轻松地配置镜像根池。

有关替换根池中磁盘的信息，请参见如何替换 ZFS 根池中的磁盘（SPARC 或 x86/VTOC） [95]。

1. 显示当前根池的状态。

```
# 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. 附加另一个磁盘，以配置镜像根池。

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

正确的磁盘标签和引导块将被自动应用。
如果在根池磁盘上有定制的分区，则可能需要使用类似于以下的语法：

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

3. 查看根池状态，确认重新同步已完成。

```
# 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
776M resilvered at 44.9M/s, 6.82% done, 0h4m to go
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
```

在上面的输出中，重新同步过程未完成。当您看到类似如下的消息时，说明重新同步已完成。

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

4. **如果要附加较大的磁盘，请设置池的 `autoexpand` 属性，以扩展池的大小。**

   确定当前的 `rpool` 池大小：

   ```
   # 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
   ```

   查看扩展后的 `rpool` 池大小：

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

5. **验证您是否可以从新磁盘成功引导。**

▼ 如何替换 ZFS 根池中的磁盘（SPARC 或 x86/VTOC）

由于以下原因，您可能需要替换根池中的磁盘：

- 根池太小，您想使用较大的磁盘替换它
- 根池磁盘发生故障。在非冗余池中，如果磁盘发生故障导致系统无法引导，则需要在替换根池磁盘前从备用介质（如 CD 或网络）进行引导。

- 如果使用 `zpool replace` 命令替换根池磁盘中的磁盘，需要手动应用引导块。

在镜像根池配置下，可以尝试替换磁盘，而不一定要从备用介质引导。可以使用 `zpool replace` 命令替换故障磁盘，如果有额外的磁盘，则可使用 `zpool attach` 命令。有关附加额外磁盘和分离根池磁盘的示例，请参见以下步骤。

具有 SATA 磁盘的系统要求您在尝试通过 `zpool replace` 操作替换故障磁盘之前使磁盘脱机并取消其配置。例如：

```
# 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 replace rpool c1t0d0s0
# zpool online rpool c1t0d0s0
# zpool status rpool
<Let disk resilver before installing the boot blocks>
# bootadm install-bootloader
```

对于一些硬件，插入替换磁盘后不必使其联机并进行配置。

1. **物理连接替换磁盘。**

2. **如有必要，准备另一个要附加到根池的磁盘。**

   - SPARC：确认替换（新）磁盘具有 SMI (VTOC) 标签和分片 0。有关为用作根池的磁盘重新设置标签的信息，请参见《在 Oracle Solaris 11.2 中管理设备 》中的“如何为磁盘设置标签”。
   - x86：确认该磁盘具有 `fdisk` 分区、SMI 磁盘标签和分片 0。如果您需要对磁盘重新分区，并创建分片 0，请参见《在 Oracle Solaris 11.2 中管理设备 》中的“配置磁盘”中有关标签和分区的部分。

3. **将新磁盘连接到根池。**

   例如：

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

   正确的磁盘标签和引导块将被自动应用。

4. **确认根池状态。**

   例如：

   ```
   # 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. **重新同步完成后，验证是否可以从新磁盘引导。**

   例如，在基于 SPARC 的系统上：

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

   标识当前磁盘和新磁盘的引导设备路径名，以便于您从替换磁盘测试引导，而且如有必要在替换磁盘发生故障时也可手动从现有磁盘引导。在以下示例中，当前的根池磁盘 (`c2t0d0s0`) 为：

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

   在以下示例中，替换引导磁盘为 (`c2t1d0s0`)：

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

6. **如果系统从新磁盘引导，则分离旧磁盘。**

   例如：

   ```
   # zpool detach rpool c2t0d0s0
   ```

7. **如果要使用较大的磁盘替换较小的根池磁盘，请设置池的 `autoexpand` 属性，以扩展池的大小。**

   确定当前的 `rpool` 池大小：

   ```
   # 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
   ```

   查看扩展后的 `rpool` 池大小：

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

8. **将系统设置为自动从新磁盘引导。**

   - SPARC：通过从引导 PROM 使用 `eeprom` 命令或 `setenv` 命令，将系统设置为自动从新磁盘引导。
   - x86：重新配置系统 BIOS。

▼ 如何替换 ZFS 根池中的磁盘（SPARC 或 x86/EFI (GPT)）

由于以下原因，您可能需要替换根池中的磁盘：

- 根池太小，您想使用较大的磁盘替换它
- 根池磁盘发生故障。在非冗余池中，如果磁盘发生故障导致系统无法引导，则需要在替换根池磁盘前从备用介质（如 CD 或网络）进行引导。
- 如果使用 `zpool replace` 命令替换根池磁盘中的磁盘，需要手动应用引导块。

在镜像根池配置下，可以尝试替换磁盘，而不一定要从备用介质引导。可以使用 `zpool replace` 命令替换故障磁盘，如果有额外的磁盘，则可使用 `zpool attach` 命令。有关附加额外磁盘和分离根池磁盘的示例，请参见以下步骤。

具有 SATA 磁盘的系统要求您在尝试通过 `zpool replace` 操作替换故障磁盘之前使磁盘脱机并取消其配置。例如：

```
# 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
```

对于一些硬件，插入替换磁盘后不必使其联机并进行配置。

1. **物理连接替换磁盘。**
2. **将新磁盘连接到根池。**

   例如：

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

   正确的磁盘标签和引导块将被自动应用。

3. **确认根池状态。**

   例如：

   ```
   # 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. **重新同步完成后，验证是否可以从新磁盘引导。**

5. **如果系统从新磁盘引导，则分离旧磁盘。**

   例如：

   ```
   # zpool detach rpool c2t0d0
   ```

6. **如果要使用较大的磁盘替换较小的根池磁盘，请设置池的 `autoexpand` 属性，以扩展池的大小。**

   确定当前的 `rpool` 池大小：

   ```
   # 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
   ```

   查看扩展后的 `rpool` 池大小：

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

7. **将系统设置为自动从新磁盘引导。**

   重新配置系统 BIOS。

▼ 如何在另一个根池中创建 BE（SPARC 或 x86/EFI (GPT)）

如果要在另一个根池中重新创建现有的 BE，请按照下面的步骤操作。您可以修改这些步骤，具体取决于您是希望获得具有独立的交换和转储设备的两个类似 BE，还是仅希望在另一个根池中获得共享交换和转储设备的一个 BE。

从第二个根池中的新 BE 激活和引导后，关于第一个根池中以前 BE 的信息将不再存在。如果要引导回原始 BE，将需要从原始根池的引导磁盘手动引导系统。

1. **创建备用根池。**

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

或者，创建一个镜像备用根池。例如：

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

2. 在第二个根池中创建新 BE。例如：

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

3. 将引导信息应用于第二个根池。例如：

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

4. 设置第二个根池的 **bootfs** 属性。例如：

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

5. 激活新 BE。例如：

```
# beadm activate solaris2
```

6. 从新的 BE 引导。

   - SPARC – 通过从引导 PROM 使用 `eeprom` 命令或 `setenv` 命令，将系统设置为自动从新磁盘引导。
   - x86 – 重新配置系统 BIOS。

   系统应该在新 BE 之下运行。

7. 重新创建交换卷。例如：

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

8. 为新交换设备更新 **/etc/vfstab** 项。例如：

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

9. 重新创建转储卷。例如：

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

10. 重置转储设备。例如：

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

11. 重新引导以清除原始根池的交换和转储设备。

```
# init 6
```

管理 ZFS 交换和转储设备

在安装过程中，将在 ZFS 根池的 ZFS 卷中创建交换区域。例如：

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

在安装过程中，将在 ZFS 根池的 ZFS 卷中创建转储设备。一般而言，转储设备不需要管理，因为它是在安装时自动设置的。例如：

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

如果禁用并移除转储设备，则在重新创建转储设备之后，需要使用 `dumpadm` 命令予以启用。大多数情况下，只需要使用 `zfs` 命令调整转储设备的大小。

有关安装程序创建的交换卷和转储卷大小的信息，请参见“确定 ZFS 根池要求” [88]。

在安装后，可以对交换卷的大小和转储卷的大小进行调整。有关更多信息，请参见“调整 ZFS 交换和转储设备的大小” [102]。

使用 ZFS 交换和转储设备时，请考虑以下问题：

- 如果要在非根池中创建交换和转储设备，请勿在 RAID-Z 池中创建交换卷和转储卷。包含交换卷和转储卷的池必须是只有一个磁盘的池或镜像池。否则，您将看到类似以下内容的消息：

  ```
  /dev/zvol/dsk/rzpool/swap: Operation not supported
  ```

- 在非根池中创建交换卷或转储卷。运行 `dumpadm -d` 命令重置转储设备。

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

- 必须将单独的 ZFS 卷用于交换区域和转储设备。
- 稀疏卷不支持用作交换卷。
- 当前，不支持在 ZFS 文件系统上使用交换文件。
- 如果在安装系统后需要更改交换区域或转储设备，请像在以前的 Solaris 发行版中那样使用 `swap` 和 `dumpadm` 命令。有关更多信息，请参见《在 Oracle Solaris 11.2 中管理文件系统》中的第 3 章 “配置附加交换空间”和《在 Oracle Solaris 11.2 中排除系统管理问题》。

调整 ZFS 交换和转储设备的大小

安装后，可能需要调整交换设备和转储设备的大小，或者可能需要重新创建交换卷和转储卷。

- 您可以在安装系统后重置转储设备的 volsize 属性。例如：

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

- 您可以调整交换卷的大小，以便系统立即使用新大小。例如：

```
# swap -l
swapfile                     dev    swaplo    blocks     free
/dev/zvol/dsk/rpool/swap   303,1         8   2097144  2097144
# zfs get volsize rpool/swap
NAME         PROPERTY   VALUE    SOURCE
rpool/swap   volsize       1G     local
# zfs set volsize=2g rpool/swap
# swap -l
swapfile                     dev    swaplo   blocks     free
/dev/zvol/dsk/rpool/swap   303,1         8  2097144  2097144
/dev/zvol/dsk/rpool/swap   303,1   2097160  2097144  2097144
```

另外，您可以使用以下方法调整交换卷的大小。但是如果使用此方法，必须重新引导系统才能看到增大的交换大小。

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

注 - 缺省情况下，为交换大小指定 *n* 个块时，将自动跳过交换文件的第一页。因此，分配的实际大小为 *n*-1 个块。要通过不同方式配置交换文件大小，请使用带有 swaplow 选项的 swap 命令。有关 swap 命令选项的更多信息，请参见 swap(1M) 手册页。

有关在活动系统上移除交换设备的信息，请参见《在 Oracle Solaris 11.2 中管理文件系统 》中的“如何在 Oracle Solaris ZFS 根环境中添加交换空间”。

- 如果在已安装的系统上需要更多的交换空间，且交换设备正忙，只需添加另一个交换卷即可。例如：

```
# zfs create -V 2G rpool/swap2
```

- 激活新的交换卷。例如：

```
# 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
```

- 针对第二个交换卷在 `/etc/vfstab` 文件中添加一项。例如：

```
/dev/zvol/dsk/rpool/swap2     -            -            swap    -    no    -
```

ZFS 转储设备故障排除

有关捕捉系统故障转储或者调整转储设备大小的问题，请查看以下各项：

- 如果没有自动创建故障转储，您可以使用 `savecore` 命令保存故障转储。
- 初始安装 ZFS 根文件系统或者迁移到 ZFS 根文件系统时，会自动创建转储设备。大多数情况下，如果缺省转储设备太小，只需要调整转储设备的大小。例如，在一个大存储器系统中，转储设备大小增大到 40 GB，如下所示：

  ```
  # zfs set volsize=40G rpool/dump
  ```

  调整大转储设备的大小可能是一个耗时的过程。

  由于某种原因，如果您需要在手动创建转储设备后启用转储设备，请使用类似以下的语法：

  ```
  # 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
  ```

- 存储器为 128 GB 或更大的系统所需的转储设备大小大于缺省创建的转储设备大小。如果转储设备太小，无法捕捉现有故障转储，将会显示类似以下内容的消息：

  ```
  # 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
  ```

  有关设置交换和转储设备大小的信息，请参见《在 Oracle Solaris 11.2 中管理文件系统 》中的“规划交换空间”。
- 目前无法将一个转储设备添加到具有多个顶层设备的池中。将显示类似于以下内容的消息：

  ```
  # dumpadm -d /dev/zvol/dsk/datapool/dump
  dump is not supported on device '/dev/zvol/dsk/datapool/dump':
  ```

```
'datapool' has multiple top level vdevs
```

请将转储设备添加到不具有多个顶层设备的根池中。

从 ZFS 根文件系统引导

基于 SPARC 和基于 x86 的系统都通过引导归档文件进行引导，引导归档文件是一个文件系统映像，该映像中包含进行引导时所需的文件。从 ZFS 根文件系统引导时，将会在选择用来进行引导的根文件系统中解析根归档文件和内核文件的路径名。

从 ZFS 文件系统引导不同于从 UFS 文件系统引导，原因是，对于 ZFS，设备说明符标识存储池，而不是单个根文件系统。存储池可能包含多个可引导的 ZFS 根文件系统。从 ZFS 引导时，必须指定引导设备和由该引导设备标识的池中的根文件系统。

缺省情况下，选择用来进行引导的文件系统是由池的 `bootfs` 属性标识的文件系统。可以覆盖此缺省选项，方法如下：在 SPARC 系统上，在 `boot -Z` 命令中指定要包含的备用可引导文件系统；在基于 x86 的系统上，从 BIOS 中选择备用引导设备。

从镜像 ZFS 根池中的备用磁盘引导

从镜像 ZFS 根池磁盘引导时，请注意以下事项：

- 在安装后，可以附加磁盘来创建镜像 ZFS 根池。有关创建镜像根池的更多信息，请参见如何配置镜像根池（SPARC 或 x86/VTOC） [92]。
- 保持根池磁盘联机并处于连接状态，以便必要时可以从其中的任何磁盘引导。
- 您不能直接从已使用 `zpool detach` 命令从系统分离的磁盘引导。您也无法从当前脱机的活动根池磁盘引导。但是，在使用现代 BIOS 的基于 x86 的系统上，如果正确设置了引导顺序且镜像了根池，即使主引导磁盘处于脱机或分离状态，系统也可以自动从辅助磁盘引导。
- SPARC：镜像根池中的主磁盘通常是缺省引导设备。您可以从镜像 ZFS 根池中的其他设备引导，但是您需要明确从该磁盘引导。如果要继续从余下的根池设备引导，或者要从余下的根池磁盘自动引导，您需要更新 PROM 以指定该缺省引导设备。

  例如，您可以从该池中的任一磁盘（`c1t0d0s0` 或 `c1t1d0s0`）引导。

```
# 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
```

在 ok 提示符下输入备用磁盘。

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

重新引导系统后，确认活动引导设备。例如：

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

- x86：在使用现代 BIOS 的基于 x86 的系统上，如果正确设置了引导磁盘顺序，即使主根池磁盘处于分离、脱机状态或因其他原因不可用，系统也可以自动从辅助设备引导。

  确认活动引导设备。例如：

  ```
  x86# prtconf -v|sed -n '/bootpath/,/value/p'
  name='bootpath' type=string items=1
  value='/pci@0,0/pci8086,25f8@4/pci108e,286@0/disk@0,0:a'
  ```

- SPARC 或 x86：如果您使用 `zpool replace` 命令替换根池磁盘，必须使用 `bootadm` 命令在新替换的磁盘上安装引导信息。如果您使用初始安装方法创建镜像 ZFS 根池，或者使用 `zpool attach` 命令向根池附加磁盘，则此步骤不是必需的。`bootadm` 语法如下所示：

  ```
  # bootadm install-bootloader
  ```

  如果要在备用根池上安装引导装载程序，请使用 `-P`（池）选项。

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

  如果要安装 GRUB 传统引导装载程序，请使用传统 `installgrub` 命令。

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

在基于 SPARC 的系统上从 ZFS 根文件系统引导

在具有多个 ZFS BE 的基于 SPARC 的系统上，可以通过使用 `beadm activate` 命令从任何 BE 引导。

在安装和 `beadm` 激活过程中，将会通过 `bootfs` 属性自动指定 ZFS 根文件系统。

一个池中可能存在多个可引导的文件系统。缺省情况下，*/pool-name*`/boot/menu.lst` 文件中的可引导文件系统项由池的 `bootfs` 属性来标识。但是，`menu.lst` 项可以包含 `bootfs` 命令，该命令可指定池中的一个备用文件系统。这样，`menu.lst` 文件就可以包含池中多个根文件系统的项。

系统安装了 ZFS 根文件系统时，将在 menu.lst 文件中添加类似以下内容的项：

```
title Oracle Solaris 11.2 SPARC
bootfs rpool/ROOT/solaris
```

创建新 BE 时，将会自动更新 menu.lst 文件。

```
title Oracle Solaris 11.2 SPARC
bootfs rpool/ROOT/solaris
title solaris
bootfs rpool/ROOT/solaris2
```

在基于 SPARC 的系统上，可以选择要通过其进行引导的 BE，如下所示：

- 激活某个 ZFS BE 后，您可以使用 boot -L 命令显示 ZFS 池中的可引导文件系统的列表。然后，您可以在列表中选择某个可引导文件系统。此时将会显示有关引导该文件系统的详细说明。您可以按照这些说明来引导选定的文件系统。
- 使用 boot -Z *file system* 命令引导特定的 ZFS 文件系统。

这种引导方法不会自动激活 BE。当使用 boot -L 和 -Z 语法引导 BE 之后，必须激活此 BE 以自动通过其继续进行引导。

例 4-1 从特定的 ZFS 引导环境引导

如果系统的引导设备上的 ZFS 存储池中有多个 ZFS BE，您可以使用 beadm activate 命令指定缺省 BE。

例如，有以下 ZFS BE 可用，如 beadm 输出所示：

```
# 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
```

如果基于 SPARC 的系统上有多个 ZFS BE，您可以使用 boot -L 命令。例如：

```
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 11.2 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
```

请记住，使用上述命令引导的 BE 不会被激活供下次重新引导时使用。如果要继续从在 boot -Z 操作期间选择的 BE 自动引导，则将需要激活它。

在基于 x86 的系统上从 ZFS 根文件系统引导

在 Oracle Solaris 11 中，对于安装了传统 GRUB 的 x86 系统，在安装过程或 `beadm activate` 操作过程中，将在 */pool-name* `/boot/grub/menu.lst` 文件中添加以下条目以自动引导 ZFS：

```
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
```

如果由 GRUB 标识为引导设备的设备包含 ZFS 存储池，则 `menu.lst` 文件用于创建 GRUB 菜单。

在具有多个 ZFS BE 的基于 x86 的系统上，您可以从 GRUB 菜单中选择 BE。如果与该菜单项对应的根文件系统是一个 ZFS 文件系统，则会添加以下选项。

```
-B $ZFS-BOOTFS
```

从 Oracle Solaris 11.1 开始，基于 x86 的系统将安装 GRUB2。`menu.lst` 文件会替换为 `/rpool/boot/grub/grub.cfg` 文件，但不应手动编辑该文件。使用 `bootadm` 子命令可添加、更改和删除菜单项。

有关修改 GRUB 菜单项的更多信息，请参见《引导和关闭 Oracle Solaris 11.2 系统》。

例 4-2 x86：引导 ZFS 文件系统

当从 GRUB2 系统上的 ZFS 根文件系统进行引导时，按如下所示指定根设备：

```
# bootadm list-menu
the location of the boot loader configuration files is: /rpool/boot/grub
default 0
console text
timeout 30
0 Oracle Solaris 11.2
```

当从传统 GRUB 系统上的 ZFS 根文件系统进行引导时，根设备由 `-B $ZFS-BOOTFS` 引导参数指定。例如：

```
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：快速重新引导 ZFS 根文件系统

在基于 x86 的系统上使用快速重新引导功能，可以在几秒内重新引导。通过快速重新引导功能，您可以重新引导至新的内核，而不会出现 BIOS 和引导装载程序可能引起的长时间延迟。快速重新引导系统的功能可显著减少停机时间并极大地提高效率。

使用 `beadm activate` 命令在 BE 之间转换时，仍然必须使用 `init 6` 命令。对于适用 `reboot` 命令的其他系统操作，可以使用 `reboot -f` 命令。例如：

```
# reboot -f
```

在 ZFS 根环境中进行引导以恢复系统

如果您需要引导系统，以解决根口令丢失或类似问题，请使用以下过程。

▼ 如何为进行恢复而引导系统

使用以下过程解决引导装载程序损坏问题或 root 口令问题。如果需要替换根池中的磁盘，请参见如何替换 ZFS 根池中的磁盘（SPARC 或 x86/VTOC） [95]。如果需要执行完整系统（裸机）恢复，请参见《在 Oracle Solaris 11.2 中使用统一归档文件进行系统恢复和克隆 》。

1. 选择适当的引导方法：

   - x86：Live Media – 从安装介质进行引导并在恢复过程中使用 GNOME 终端。
   - SPARC：文本安装 – 从安装介质或者从网络进行引导，然后从文本安装屏幕中选择 `3 Shell` 选项。
   - x86：文本安装 – 在 GRUB 菜单中，选择 `Text Installer and command line`（文本安装程序和命令行）引导项，然后从文本安装屏幕中选择 `3 Shell` 选项。
   - SPARC：自动化安装 – 使用以下命令从允许退出到 shell 的安装菜单直接引导。

     ```
     ok boot net:dhcp
     ```

   - x86：自动化安装 – 从网络上的安装服务器进行引导需要 PXE 引导。选择 GRUB 菜单上的 `Text Installer and command line`（文本安装程序和命令行）项。然后，从文本安装屏幕中选择 `3 Shell` 选项。

   例如，引导系统后，选择 `3 Shell` 选项。

   ```
         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. 选择引导恢复问题：

   - 要解决 root shell 错误问题，请将系统引导到单用户模式并更正 /etc/passwd 文件中的 shell 项。

     在 x86 系统上，编辑选定的引导项，然后添加 -s 选项。

     例如，在 SPARC 系统上，关闭系统并引导至单用户模式。作为 root 用户登录后，编辑 /etc/passwd 文件，然后修复 root shell 项。

     ```
     # 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 11.2 64-bit
     Copyright (c) 1983, 2013, 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      11.2    July 2013
     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.
     ```

   - 解决引导装载程序损坏问题。

     首先，您必须使用步骤 1 中列出的引导方法之一从介质或网络引导系统。然后，执行诸如导入根池和修复 GRUB 条目等操作。

     ```
     x86# zpool import -f rpool
     ```

     重新安装引导装载程序。

     ```
     x86# bootadm install-bootloader -f -P rpool
     ```

其中 -f 选项将强制安装引导装载程序并绕过所有版本检查，以便不对系统上的引导装载程序版本进行降级。-P 选项用于指定根池。

您可以使用 bootadm list-menu 命令列出并修改 GRUB2 条目。有关更多信息，请参见 bootadm(1M)。

退出并重新引导系统。

```
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
```

确认系统引导成功。

- 解决 root 口令未知导致无法登录系统的问题。

  首先，您必须使用步骤 1 中列出的引导方法之一从介质或网络引导系统。然后，导入根池 (rpool) 并挂载 BE 以删除 root 口令项。在 SPARC 平台和 x86 平台上，该过程是相同的。

```
# 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
```

  转至下一步设置 root 口令。

3. 通过引导至单用户模式并设置口令来设置 root 口令。

   此步骤假设您在上一步中删除了未知的 root 口令。

在基于 x86 系统上，编辑选定的引导项，然后添加 -s 选项。

在基于 SPARC 系统上，将系统引导至单用户模式，以 root 身份登录，然后设置 root 口令。例如：

```
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 11.2 64-bit
Copyright (c) 1983, 2013, 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      11.2    July 2013
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

管理 Oracle Solaris ZFS 文件系统

本章提供有关管理 Oracle Solaris ZFS 文件系统的详细信息。本章包括分层次的文件系统布局、属性继承和自动挂载点管理以及共享交互等概念。

本章包含以下各节：

管理 ZFS 文件系统

ZFS 文件系统构建于存储池上。文件系统可以动态创建和销毁，而不需要分配或格式化任何底层磁盘空间。因为文件系统非常轻量化，又是 ZFS 中的管理中心，因此您可以大量创建。

使用 `zfs` 命令可以管理 ZFS 文件系统。`zfs` 命令提供了一组用于对文件系统执行特定操作的子命令。本章详细介绍了这些子命令。使用此命令还可以管理快照、卷和克隆，但本章仅对这些功能进行了简短介绍。有关快照和克隆的详细信息，请参见第 6 章 使用 Oracle Solaris ZFS 快照和克隆。有关 ZFS 卷的详细信息，请参见“ZFS 卷” [227]。

注 - 术语数据集在本章中用作通称，表示文件系统、快照、克隆或卷。

创建、销毁和重命名 ZFS 文件系统

可以使用 `zfs create` 和 `zfs destroy` 命令来创建和销毁 ZFS 文件系统。使用 `zfs rename` 命令可重命名 ZFS 文件系统。

创建 ZFS 文件系统

使用 `zfs create` 命令可以创建 ZFS 文件系统。`create` 子命令仅使用一个参数：要创建的文件系统的名称。文件系统名称指定为以池名称开头的路径名，如下所示：

pool-name/[filesystem-name/]filesystem-name

路径中的池名称和初始文件系统名称标识分层结构中要创建新文件系统的位置。路径中的最后一个名称标识要创建的文件系统的名称。文件系统名称必须满足“ZFS 组件命名要求” [17]中所述的命名要求。

在创建文件系统时，必须启用对 ZFS 文件系统进行加密的功能。有关对 ZFS 文件系统进行加密的信息，请参见“加密 ZFS 文件系统” [157]。

在以下示例中，在 `tank/home` 文件系统中创建了一个名为 `jeff` 的文件系统。

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

如果成功创建文件系统，ZFS 会自动挂载新创建的文件系统。缺省情况下，文件系统将使用 `create` 子命令中为文件系统名称提供的路径挂载为 */dataset*。在本示例中，新创建的 `jeff` 文件系统挂载于 `/tank/home/jeff`。有关自动管理的挂载点的更多信息，请参见“管理 ZFS 挂载点” [138]。

有关 `zfs create` 命令的更多信息，请参见 `zfs(1M)`。

可在创建文件系统时设置文件系统属性。

在以下示例中，为 `tank/home` 文件系统创建了挂载点 `/export/zfs`：

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

有关文件系统属性的更多信息，请参见“介绍 ZFS 属性” [116]。

销毁 ZFS 文件系统

要销毁 ZFS 文件系统，请使用 `zfs destroy` 命令。销毁的文件系统将自动取消挂载，并取消共享。有关自动管理的挂载或自动管理的共享的更多信息，请参见“自动挂载点” [139]。

在以下示例中，销毁了 `tank/home/mark` 文件系统：

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

注意 - 使用 `destroy` 子命令时不会出现确认提示。请务必谨慎使用该子命令。

如果要销毁的文件系统处于繁忙状态而无法取消挂载，则 `zfs destroy` 命令将失败。要销毁活动文件系统，请使用 `-f` 选项。由于此选项可取消挂载、取消共享和销毁活动文件系统，从而导致意外的应用程序行为，因此请谨慎使用此选项。

```
# zfs destroy tank/home/matt
cannot unmount 'tank/home/matt': Device busy

# zfs destroy -f tank/home/matt
```

如果文件系统具有后代，则 `zfs destroy` 命令也会失败。要以递归方式销毁文件系统及其所有后代，请使用 `-r` 选项。请注意，递归销毁同时会销毁快照，因此请谨慎使用此选项。

```
# 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
```

如果要销毁的文件系统具有间接依赖项，递归销毁命令也会失败。要强制销毁所有依赖项（包括目标分层结构外的克隆文件系统），必须使用 `-R` 选项。请务必谨慎使用此选项。

```
# 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 - 在 `zfs destroy` 命令中使用 `-f`、`-r` 或 `-R` 选项不会出现确认提示，因此请谨慎使用这些选项。

有关快照和克隆的更多信息，请参见第 6 章 使用 Oracle Solaris ZFS 快照和克隆。

重命名 ZFS 文件系统

使用 `zfs rename` 命令可重命名文件系统。使用 `rename` 子命令可以执行以下操作：

- 更改文件系统的名称。
- 在 ZFS 分层结构内重定位文件系统。
- 更改文件系统的名称并在 ZFS 分层结构内对其重定位。

以下示例使用 `rename` 子命令将一个文件系统从 `eric` 重命名为 `eric_old`：

```
# zfs rename tank/home/eric tank/home/eric_old
```

以下示例说明如何使用 `zfs rename` 重定位文件系统：

```
# zfs rename tank/home/mark tank/ws/mark
```

在本示例中，`mark` 文件系统从 `tank/home` 重定位到 `tank/ws`。通过重命名来重定位文件系统时，新位置必须位于同一池中，并且必须具有足够的磁盘空间来存放这一新文件系统。如果新位置没有足够的磁盘空间（可能是因为已达到配额），则 `rename` 操作将失败。

有关配额的更多信息，请参见“设置 ZFS 配额和预留空间” [152]。

`rename` 操作会尝试对文件系统以及任何后代文件系统按顺序执行取消挂载/重新挂载操作。如果该操作无法取消挂载活动文件系统，则 `rename` 命令将失败。发生这种问题时，必须强行取消挂载该文件系统。

有关重命名快照的信息，请参见“重命名 ZFS 快照” [172]。

介绍 ZFS 属性

属性是用来对文件系统、卷、快照和克隆的行为进行控制的主要机制。除非另有说明，否则本节中阐述的属性适用于所有数据集类型。

- “ZFS 只读本机属性” [124]
- “可设置的 ZFS 本机属性” [125]
- “ZFS 用户属性” [130]

属性分为两种类型：本机属性和用户定义的属性。本机属性用于提供内部统计信息或控制 ZFS 文件系统行为。此外，本机属性是可设置的或只读的。用户属性对 ZFS 文件系统行为没有影响，但可通过用户环境中有意义的方式来注释数据集。有关用户属性的更多信息，请参见“ZFS 用户属性” [130]。

大多数可设置的属性也是可继承的。可继承属性是这样的属性：如果为父文件系统设置了该属性，则该属性会向下传播给其所有后代。

所有可继承属性都有一个关联的源，此源指示获得属性的方式。属性的源可具有以下值：

`local` 表示属性是使用 `zfs set` 命令对数据集进行显式设置的，如“设置 ZFS 属性” [134]中所述。

`inherited from` *dataset-name* 表示属性是从指定的祖先继承而来。

`default` 表示属性值不是继承而来或在本地设置。如果没有祖先具有属性源 `local`，则会使用此源。

下表介绍了只读的和可设置的本机 ZFS 文件系统属性。只读本机属性在表中注明为“只读属性”。此表中列出的所有其他本机属性均为可设置的属性。有关用户属性的信息，请参见“ZFS 用户属性” [130]。

表 5-1 ZFS 本机属性说明

属性名称	类型	缺省值	说明
`aclinherit`	字符串	`secure`	控制创建文件和目录时继承 ACL 项的方式。该属性的值包括 `discard`、`noallow`、`secure` 和 `passthrough`。有关这些值的说明，请参见“ACL 属性” [194]。
`aclmode`	字符串	`groupmask`	控制 `chmod` 操作过程中修改 ACL 项的方式。该属性的值包括 `discard`、`groupmask` 和 `passthrough`。有关这些值的说明，请参见“ACL 属性” [194]。
`atime`	布尔型	`on`	控制在读取文件时是否更新文件的访问时间。禁用该属性可避免在读取文件时产生写入流量，因此可显著提高性能，但可能会使邮件程序与相似的实用程序感到困惑。
`available`	数字	N/A	只读属性，用于指明可供某个文件系统及其所有子级使用的磁盘空间量，假定池中没有其他活动。由于池中会共享磁盘空间，因此可用空间会受到许多因素的限制，包括物理池大小、配额、预留空间和池中的其他数据集。 此属性的缩写为 `avail`。 有关磁盘空间记帐的更多信息，请参见“ZFS 磁盘空间记帐” [18]。
`canmount`	布尔型	`on`	控制是否可以使用 `zfs mount` 命令挂载文件系统。在任意文件系统中均可设置该属性，该属性本身不可继承。不过，当此属性设置为 `off` 时，后代文件系统可以继承挂载点，但永远不会挂载文件系统本身。 当设置了 `noauto` 选项时，只能显式挂载和卸载文件系统。文件系统不会在创建或导入时自动挂载，也不能通过 `zfs mount-a` 命令挂载或通过 `zfs unmount-a` 命令卸载。 有关详细信息，请参见“canmount 属性” [126]。

属性名称	类型	缺省值	说明
casesensitivity	字符串	mixed	此属性指示文件系统使用的文件名匹配算法应当是 casesensitive、caseinsensitive，还是允许这两种匹配方式的组合 (mixed)。传统上，UNIX 和 POSIX 文件系统的文件名区分大小写。 此属性的值为 mixed 时表示文件系统对区分大小写和不区分大小写的匹配行为要求均可支持。当前，在支持混合行为的文件系统上，不区分大小写的匹配行为仅限于 Oracle Solaris SMB 服务器产品。有关使用 mixed 值的更多信息，请参见“casesensitivity 属性” [127]。 无论 casesensitivity 属性的设置是什么，文件系统都会保留创建文件时指定的名称的大小写。在创建文件系统后无法更改此属性。
checksum	字符串	on	控制用于验证数据完整性的校验和。缺省值为 on，这将自动选择合适的算法，当前算法为 fletcher4。值包括 on, off, fletcher2, fletcher4, sha256 和 sha256+mac。值为 off 将禁用对用户数据的完整性检查。建议不要使用值 off。
compression	字符串	off	启用或禁用数据集压缩。该属性的值包括 on、off、lzjb、gzip 和 gzip- *N*。目前，将此属性设置为 lzjb、gzip 或 gzip-*N* 与将此属性设置为 on 具有相同的效果。在包含现有数据的文件系统中启用压缩将只压缩新数据。现有数据不会被压缩。 此属性的缩写为 compress。
compressratio	数字	N/A	只读属性，用于指明数据集实现的压缩率，表示为一个乘数。可通过 zfs set compression=on *dataset* 命令启用压缩。 根据所有文件的逻辑大小和引用的物理数据量计算此值。它包括通过使用 compression 属性实现的节省量。
copies	数字	1	设置每个文件系统的用户数据副本数。可用的值为 1、2 或 3。这些副本是对任何池级别冗余的补充。用户数据多个副本所使用的磁盘空间将计入相应的文件和数据集，并根据配额和预留空间进行计数。此外，启用多个副本时还会更新 used 属性。由于在现有文件系统中更改此属性仅影响新写入的数据，因此请考虑在创建文件系统时设置此属性。
creation	字符串	N/A	只读属性，用于指明创建数据集的日期和时间。
dedup	字符串	off	控制在 ZFS 文件系统中删除重复数据的功能。可能的值包括 on、off、verify 和 sha256[,verify]。针对重复数据删除的缺省校验和是 sha256。 有关详细信息，请参见“dedup 属性” [128]。
devices	布尔型	on	控制是否可以打开某个文件系统中的设备文件。
encryption	布尔型	off	控制是否对文件系统进行加密。加密的文件系统意味着，数据已编码，文件系统所有者需要有密钥才能访问数据。

属性名称	类型	缺省值	说明
exec	布尔型	on	控制是否允许执行某个文件系统中的程序。如果设置为 off，将禁止带 PROT_EXEC 的 mmap(2) 调用。
keychangedate	字符串	none	标识在针对指定文件系统的 zfs key -c 操作中上次更改包装密钥的日期。如果未发生任何密钥更改操作，此只读属性的值与文件系统的创建日期相同。
keysource	字符串	none	标识对文件系统密钥进行封装的密钥的格式和位置。有效的属性值包括 raw、hex、passphrase, prompt 或 *file*。使用 zfs key -l 命令创建、挂载或装入文件系统时，必须提供该密钥。如果为新文件系统启用加密，则缺省的 keysource 为 passphrase, prompt。
keystatus	字符串	none	标识文件系统的加密密钥状态的只读属性。文件系统密钥的可用性由 available 或 unavailable 予以指示。对于没有启用加密的文件系统，则显示 none。
logbias	字符串	latency	控制 ZFS 优化该文件系统的同步请求的方式。如果 logbias 设置为 latency，ZFS 将使用池的不同日志设备（如有）低延迟地处理请求。如果 logbias 设置为 throughput，ZFS 将不使用池的不同日志设备。相反，ZFS 将优化同步操作，以提高池的全局吞吐量并有效使用资源。缺省值为 latency。
mlslabel	字符串	None	有关多级别文件系统中 mlslabel 属性行为的描述，请参见 multilevel 属性。下面的 mlslabel 描述适用于非多级别文件系统。 提供敏感标签，确定文件系统是否可以在 Trusted Extensions 区域中挂载。如果有标签的文件系统与有标签的区域相符，则可以从有标签的区域挂载和访问该文件系统。缺省值为 none。仅当启用了 Trusted Extensions 且具有相应的特权时才能修改此属性。
mounted	布尔型	N/A	只读属性，用于指明当前是否挂载了文件系统、克隆或快照。该属性不适用于卷。值可以是 yes 或 no。
mountpoint	字符串	N/A	控制用于此文件系统的挂载点。当文件系统的 mountpoint 属性发生更改时，将取消挂载该文件系统以及继承该挂载点的任何后代。如果新值为 legacy，则该文件系统和子级将保持卸载状态。否则，如果属性以前为 legacy 或 none，或者该文件系统和子级在属性发生更改之前处于挂载状态，则会自动在新位置重新挂载它们。此外，任何共享文件系统都将取消共享，并在新位置进行共享。 有关使用该属性的更多信息，请参见“管理 ZFS 挂载点” [138]。
multilevel	布尔型	off	此属性只能在启用了 Trusted Extensions 的系统上使用。缺省值为 off。 多级别文件系统中的对象带有各自的标签，这些标签使用自动生成的显式敏感标签属性。可以就地重新标记对象，方法是使用 setlabel 或 setflabel 接口更改此标签属性。

属性名称	类型	缺省值	说明
			根文件系统、Oracle Solaris Zone 文件系统或包含已打包的 Solaris 代码的文件系统不应为多级别文件系统。 多级别文件系统中的 mlslabel 属性有所不同。mlslabel 值为文件系统中的对象定义最高可能标签。不允许将文件的标签创建为或重新标记为高于 mlslabel 值的标签。基于 mlslabel 值的挂载策略不适用于多级别文件系统。 对于多级别文件系统，在创建文件系统时可显式设置 mlslabel 属性。否则，会自动创建缺省 mlslabel 属性 ADMIN_HIGH。创建多级别文件之后，可以更改 mlslabel 属性，但不能将其设置为较低级别标签，也不能设置为 none 或将其删除。
primarycache	字符串	all	控制主高速缓存 (ARC) 中缓存的内容。可能的值包括 all、none 和 metadata。如果设置为 all，则用户数据和元数据都会被缓存。如果设置为 none，则用户数据和元数据都不会被缓存。如果设置为 metadata，则只有元数据会被缓存。如果在现有文件系统上设置这些属性，则根据这些属性的值仅缓存新 I/O。对某些数据集环境而言，不高速缓存用户数据可能会带来一些好处。您必须确定您的环境是否适合设置高速缓存属性。
nbmand	布尔型	off	控制在挂载文件系统时是否应使用 nbmand（非阻塞强制性）锁。此属性仅适用于 SMB 客户机。对此属性所做的更改只有在卸载文件系统并重新挂载后才有效。
normalization	字符串	None	此属性指示每次对两个文件名进行比较时，文件系统是否应对文件名执行 unicode 标准化，以及应使用哪种标准化算法。文件名在存储时始终保持未修改状态，并将其标准化作为任何比较进程的一部分。如果将此属性设置为 none 以外的合法值，并且不指定 utf8only 属性，则 utf8only 属性自动设置为 on。normalization 属性的缺省值为 none。在创建文件系统后无法更改此属性。
origin	字符串	N/A	克隆的文件系统或卷的只读属性，用于标识创建克隆所在的快照。只要克隆存在，便不能销毁克隆源（即使使用 -r 或 -f 选项也是如此）。 非克隆文件系统的 origin 为 none。
quota	数字（或 none）	none	限制文件系统及其后代可以占用的磁盘空间量。该属性可对已使用的磁盘空间量强制实施硬限制，包括后代（含文件系统和快照）占用的所有空间。对已有配额的文件系统的后代设置配额不会覆盖祖先的配额，但会施加额外的限制。不能对卷设置配额，因为 volsize 属性可用作隐式配额。 有关设置配额的信息，请参见“设置 ZFS 文件系统的配额” [153]。
rekeydate	字符串	N/A	只读属性，指示上次因对此文件系统执行 zfs key -K 或 zfs clone - K 操作而导致数据加密密钥发生更改

属性名称	类型	缺省值	说明
			的日期。如果未执行 rekey 操作，则此属性的值与 creation 日期相同。
readonly	布尔型	off	控制某个数据集是否可以修改。如果设置为 on，则不能进行任何修改。 此属性的缩写为 rdonly。
recordsize	数字	128K	为文件系统中的文件指定建议的块大小。 此属性的缩写为 recsize。有关详细说明，请参见“recordsize 属性” [129]。
referenced	数字	N/A	只读属性，用于指明数据集可访问的数据量，这些数据可能会也可能不会与池中的其他数据集共享。 创建快照或克隆时，首先会引用与创建该属性时所在的文件系统或快照相同的磁盘空间量，因为其内容相同。 此属性的缩写为 refer。
refquota	数字（或 none）	none	设置数据集可以占用的磁盘空间量。此属性对使用的空间量强制实施硬限制。此硬限制不包括后代（如快照和克隆）所使用的磁盘空间。
refreservation	数字（或 none）	none	设置为数据集（不包括快照和克隆等后代）保留的最小磁盘空间量。如果使用的磁盘空间量低于该值，则认为数据集正在使用 refreservation 指定的空间量。refreservation 预留空间计算在父数据集的已用磁盘空间内，并会针对父数据集的配额和预留空间进行计数。 如果设置了 refreservation，则仅当在此预留空间之外有足够的可用池空间来容纳数据集中的当前引用字节数时，才允许使用快照。 此属性的缩写为 refreserv。
reservation	数字（或 none）	none	设置为文件系统及其后代所保留的最小磁盘空间量。如果使用的磁盘空间量低于该值，则认为文件系统正在使用其预留空间指定的空间量。预留空间计入父文件系统的已用磁盘空间内，并将计入父文件系统的配额和预留空间。 此属性的缩写为 reserv。 有关更多信息，请参见“设置 ZFS 文件系统的预留空间” [156]。
rstchown	布尔型	on	指示文件系统所有者是否可以准许文件所有权更改。缺省情况下禁止 chown 操作。当 rstchown 设置为 off 时，用户具有 PRIV_FILE_CHOWN_SELF 特权，可执行 chown 操作。
secondarycache	字符串	all	控制辅助高速缓存 (L2ARC) 中缓存的内容。可能的值包括 all、none 和 metadata。如果设置为 all，则用户数据和元数据都会被缓存。如果设置为 none，则用户数据和元数据都不会被缓存。如果设置为 metadata，则只有元数据会被缓存。

属性名称	类型	缺省值	说明
setuid	布尔型	on	控制文件系统中是否考虑 setuid 位。
shadow	字符串	None	将某个 ZFS 文件系统标识为 *URI* 描述的文件系统的影子。数据将从 URI 标识的文件系统迁移到设置了该属性的影子文件系统。要实现完整迁移，要迁移的文件系统必须为只读的。
share.nfs	字符串	off	控制是否创建和发布 ZFS 文件系统的 NFS 共享并控制使用的选项。您也可以使用 zfs share 命令发布 NFS 共享，使用 zfs unshare 命令取消发布 NFS 共享。使用 zfs share 命令发布 NFS 共享时还需要设置 NFS 共享属性。有关设置 NFS 共享属性的信息，请参见“共享和取消共享 ZFS 文件系统” [142]。 有关共享 ZFS 文件系统的更多信息，请参见“共享和取消共享 ZFS 文件系统” [142]。
share.smb	字符串	off	控制是否创建和发布 ZFS 文件系统的 SMB 共享并控制使用的选项。您也可以使用 zfs share 命令发布 SMB 共享，使用 zfs unshare 命令取消发布 SMB 共享。使用 zfs share 命令发布 SMB 共享时还需要设置 SMB 共享属性。有关设置 SMB 共享属性的信息，请参见“共享和取消共享 ZFS 文件系统” [142]。
snapdir	字符串	hidden	控制 .zfs 目录在文件系统的根目录中是隐藏还是可见。有关使用快照的更多信息，请参见“ZFS 快照概述” [169]。
sync	字符串	standard	确定文件系统事务的同步行为。可能的值包括： ■ standard（缺省值），表示将同步文件系统事务（如 fsync、O_DSYNC、O_SYNC 等）写入到意图日志 (intent log)。 ■ always，确保写入每个文件系统事务并通过一个返回系统调用将其刷新到稳定的存储器。此值会导致显著的性能损失。 ■ disabled，表示禁用同步请求。仅当下一次提交事务组时，才将文件系统事务提交到稳定的存储器，这可能会延迟好几秒钟。此值可提供最佳的性能，且没有损坏池的风险。 注意 - 此 disabled 值非常危险，因为 ZFS 会忽略应用程序的同步事务需求，例如数据库操作或 NFS 操作。在当前活动的根文件系统或 /var 文件系统上设置此值可能会导致意外行为、应用程序数据丢失或重放攻击的漏洞加重。只有完全了解所有相关风险时才能使用此值。
type	字符串	N/A	只读属性，用于指明数据集类型是 filesystem（文件系统或克隆）、volume 还是 snapshot。
used	数字	N/A	只读属性，用于指明数据集及其所有后代占用的磁盘空间量。 有关详细说明，请参见“used 属性” [124]。

属性名称	类型	缺省值	说明
usedbychildren	数字	off	只读属性，用于指明此数据集子代占用的磁盘空间量，如果该数据集的所有子代被销毁，则将释放此空间量。此属性的缩写为 usedchild。
usedbydataset	数字	off	只读属性，用于指明此数据集本身占用的空间量，在首先销毁任何快照并删除任何 refreservation 预留空间后销毁此数据集，将释放该空间量。该属性缩写为 usedds。
usedbyrefreservation	数字	off	只读属性，用于指明在数据集上设置的 refreservation 占用的磁盘空间量，如果删除 refreservation，则将释放该空间量。此属性的缩写为 usedrefreserv。
usedbysnapshots	数字	off	只读属性，用于指明数据集的快照占用的磁盘空间量。特别是，如果此数据集的所有快照都被销毁，将释放该磁盘空间。请注意，此值不是简单的快照 used 属性总和，因为多个快照可以共享空间。此属性的缩写为 usedsnap。
version	数字	N/A	指明文件系统的盘上版本，与池的版本无关。此属性只能设置为比支持的软件发行版所提供的版本更高的版本。有关更多信息，请参见 zfs upgrade 命令。
utf8only	布尔型	Off	此属性指示当文件名含有 UTF-8 字符代码集中不存在的字符时，文件系统是否应拒绝此类文件名。如果已将此属性显式设置为 off，则不允许显式设置 normalization 属性，也不允许将该属性设置为 none。utf8only 属性的缺省值为 off。在创建文件系统后无法更改此属性。
volsize	数字	N/A	为卷指定卷的逻辑大小。 有关详细说明，请参见“volsize 属性” [130]。
volblocksize	数字	8 KB	为卷指定卷的块大小。一旦写入卷后，块大小便不能更改，因此应在创建卷时设置块大小。卷的缺省块大小为 8 KB。位于 512 字节到 128 KB 之间的 2 的任意次幂都有效。 此属性的缩写为 volblock。
vscan	布尔型	Off	控制打开和关闭常规文件时是否应为其扫描病毒。如果具有第三方病毒扫描软件，则除了启用此属性外，还必须启用病毒扫描服务才会执行病毒扫描。缺省值为 off（关闭）。
zoned	布尔型	N/A	指明是否将已文件系统添加到非全局区域。如果设置该属性，全局区域中将不会标记挂载点，因此 ZFS 在收到请求时不能挂载此类文件系统。首次安装区域时，会为添加的所有文件系统设置该属性。 有关将 ZFS 用于已安装的区域的更多信息，请参见“在安装了区域的 Solaris 系统中使用 ZFS” [229]。
xattr	布尔型	on	指明此文件系统是启用 (on) 还是禁用了 (off) 扩展属性。

ZFS 只读本机属性

可以检索但无法设置只读本机属性。只读本机属性不可继承。有些本机属性特定于特殊类型的数据集。在这种情况下，表 5-1 “ZFS 本机属性说明” 的说明部分会注明数据集类型。

下面列出了只读本机属性，表 5-1 “ZFS 本机属性说明” 对其进行了描述。

- available
- compressratio
- creation
- keystatus
- mounted
- origin
- referenced
- rekeydate
- type
- used

  有关详细信息，请参见 “used 属性” [124]。
- usedbychildren
- usedbydataset
- usedbyrefreservation
- usedbysnapshots

有关磁盘空间记帐（包括 used、referenced 和 available 属性）的更多信息，请参见“ZFS 磁盘空间记帐” [18]。

used 属性

used 属性是一个只读属性，表明此数据集及其所有后代占用的磁盘空间量。可根据此数据集的配额和预留空间来检查该值。使用的磁盘空间不包括数据集的预留空间，但会考虑任何后代数据集的预留空间。数据集占用其父级的磁盘空间量以及以递归方式销毁该数据集时所释放的磁盘空间量应为其使用空间和预留空间的较大者。

创建快照时，其磁盘空间最初在快照与文件系统之间进行共享，还可能是与以前的快照进行共享。随着文件系统的变化，以前共享的磁盘空间将供快照专用，并会计算在快照的使用空间内。快照使用的磁盘空间会将其专用数据所占空间计算在内。此外，删除快照可增加其他快照专用（和使用）的磁盘空间量。有关快照和空间问题的更多信息，请参见“空间不足行为” [19]。

已用磁盘空间量、可用磁盘空间量以及引用磁盘空间量并不包括暂挂更改。通常，暂挂更改仅占用几秒钟的时间。使用 `fsync(3c)` 或 `O_SYNC` 功能提交对磁盘的更改，不一定可以保证磁盘空间使用情况信息会立即更新。

使用 `zfs list -o space` 命令，可以显示 `usedbychildren`、`usedbydataset`、`usedbyrefreservation` 和 `usedbysnapshots` 属性信息。这些属性将 `used` 属性细分为后代占用的磁盘空间。有关更多信息，请参见表 5-1 “ZFS 本机属性说明”。

可设置的 ZFS 本机属性

可设置的本机属性是其值可同时进行检索和设置的属性。可设置的本机属性可以使用 `zfs set` 命令或 “设置 ZFS 属性” [134] 命令进行设置，请分别参见`Setting ZFS Properties`和“创建 ZFS 文件系统” [114]中的描述。除了配额和预留空间外，可设置的本机属性均可继承。有关配额的更多信息，请参见“设置 ZFS 配额和预留空间” [152]。

有些可设置的本机属性特定于特殊类型的数据集。在这种情况下，表 5-1 “ZFS 本机属性说明” 的说明部分会注明数据集类型。如果未明确注明，则表明属性适用于所有数据集类型： 文件系统、卷、克隆和快照。

下面列出了可设置的属性，表 5-1 “ZFS 本机属性说明” 对其进行了描述。

- `aclinherit`

  有关详细说明，请参见“ACL 属性” [194]。
- `aclmode`

  有关详细说明，请参见“ACL 属性” [194]。
- `atime`
- `canmount`
- `casesensitivity`
- `checksum`
- `compression`
- `copies`
- `devices`
- `dedup`
- `encryption`
- `exec`
- `keysource`
- `logbias`
- `mlslabel`
- `mountpoint`

- nbmand
- normalization
- primarycache
- quota
- readonly
- recordsize

  有关详细说明，请参见“recordsize 属性” [129]。
- refquota
- refreservation
- reservation
- rstchown
- secondarycache
- share.smb
- share.nfs
- setuid
- snapdir
- version
- vscan
- utf8only
- volsize

  有关详细说明，请参见“volsize 属性” [130]。
- volblocksize
- zoned
- xattr

canmount 属性

如果 canmount 属性设置为 off，则不能使用 zfs mount 或 zfs mount -a 命令挂载文件系统。将此属性设置为 off 与将 mountpoint 属性设置为 none 的效果相似，区别在于文件系统仍有一个可以继承的普通 mountpoint 属性。例如，可将该属性设置为 off，为后代文件系统建立可继承属性，但父文件系统本身永远不会挂载，也无法供用户访问。在这种情况下，父文件系统将充当一个容器，这样便可以在容器中设置属性，但容器本身永远不可访问。

在以下示例中，创建了 userpool 并将其 canmount 属性设置为 off。将后代用户文件系统的挂载点设置为一个公共挂载点 /export/home。在父文件系统中设置的属性可由后代文件系统继承，但永远不会挂载父文件系统本身。

```
# 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
```

将 canmount 属性设置为 noauto 意味着文件系统只能显式挂载，而不能自动挂载。

casesensitivity 属性

此属性指示文件系统使用的文件名匹配算法应当是 casesensitive、caseinsensitive，还是允许这两种匹配方式的组合 (mixed)。

对混合敏感性文件系统发出不区分大小写的匹配请求时，行为通常与纯正的不区分大小写的文件系统的预期行为相同。区别在于，在混合敏感性文件系统中可能存在以下情况：目录中的多个名称从区分大小写的角度来看是唯一的，但从不区分大小写的角度来看则不是唯一的。

例如，某个目录中可能包含文件 foo、Foo 和 FOO。如果请求对 foo 的任何可能形式（例如 foo、FOO、FoO、fOo 等）进行不区分大小写的匹配，则该匹配算法会选择三个现有文件之一作为匹配项。无法保证该算法到底选择哪个文件作为匹配项，但可以保证会选择同一文件作为 foo 的任何形式的匹配项。只要目录保持不变，被选作 foo、FOO、foO、Foo 等的不区分大小写匹配项的文件就始终相同。

utf8only、normalization 和 casesensitivity 属性还提供了可以通过使用 ZFS 委托管理指定给非特权用户的新权限。有关更多信息，请参见“授予 ZFS 权限” [216]。

copies 属性

作为一项可靠性功能，如果可能，ZFS 文件系统元数据会在不同的磁盘中自动存储多次。此功能称为重复块 (ditto block)。

在此发行版中，还可以使用 zfs set copies 命令存储用户数据的多个副本，这些副本也按文件系统进行存储。例如：

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

可用的值为 1、2 或 3。缺省值为 1。除了任何池级别的冗余以外，这些副本还用于诸如镜像或 RAID-Z 之类的配置中。

存储 ZFS 用户数据的多个副本的优点如下：

- 通过支持所有 ZFS 配置从不可恢复的块读取故障（例如介质故障（一般称为位损坏））恢复来提高数据保留能力。
- 提供数据保护，即使只有一个磁盘可用。
- 允许您在存储池功能之外以每个文件系统为基础选择数据保护策略。

注 - 根据存储池中重复块 (ditto block) 的分配，可能会将多个副本置于单个磁盘上。某个后续的满载磁盘故障可能会导致所有重复块 (ditto block) 都不可用。

无意中创建了非冗余池时以及需要设置数据保留策略时，可能会考虑使用重复块 (ditto block)。

dedup 属性

dedup 属性控制是否从文件系统中删除重复数据。如果文件系统启用了 dedup 属性，则会以同步方式删除重复的数据块。结果是仅存储唯一的数据，在文件之间共享通用组件。

在检查以下注意事项之前，不要在驻留于生产系统上的文件系统上启用 dedup 属性：

1. 确定数据是否将受益于重复数据删除产生的空间节省。您可以运行 zdb -S 命令模拟在池上启用重复数据删除可能会节省的空间。此命令必须在静默池上运行。如果您的数据不是可进行重复数据删除的，则启用 dedup 没有意义。例如：

```
# 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
```

如果估计的重复数据删除比大于 2，则重复数据删除可能会带来空间节省。

在上述示例中，重复数据删除比小于 2，因此建议不要启用 dedup。

2. 请确保系统具有足够的内存来支持重复数据删除。
   - 每个核心中重复数据删除表项约为 320 字节
   - 用分配的块数乘以 320。例如：

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

3. 当重复数据删除表可以完全装入内存时，重复数据删除的性能最佳。如果不得不将重复数据删除表写入磁盘，则性能将降低。例如，如果系统不满足上述内存要求，则删除启用了 dedup 的大文件系统将大大降低系统性能。

在启用 dedup 时，dedup 校验和算法会覆盖 checksum 属性。将属性值设置为 verify 等效于指定 sha256,verify。如果将属性设置为 verify，且两个块具有相同的签名，则 ZFS 会与现有块进行逐字节比较，以确保内容完全相同。

可以按文件系统启用此属性。例如：

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

可以使用 zfs get 命令确定是否设置了 dedup 属性。

虽然重复数据删除是作为文件系统属性设置的，但是它在池范围内起作用。例如，您可以确定重复数据删除比。例如：

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

DEDUP 列指示已发生了多少重复数据删除。如果在任何文件系统上都没有启用 dedup 属性，或者如果刚刚在文件系统上启用了 dedup 属性，则 DEDUP 比是 1.00x。

可以使用 zpool get 命令确定 dedupratio 属性的值。例如：

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

此池属性说明此池已实现了多少重复数据删除。

encryption 属性

可以使用 encryption 属性来加密 ZFS 文件系统。有关更多信息，请参见“加密 ZFS 文件系统” [157]。

recordsize 属性

recordsize 属性为文件系统中的文件指定建议的块大小。

该属性专门设计用于对大小固定的记录中的文件进行访问的数据库工作负荷。ZFS 会根据为典型的访问模式优化的内部算法来自动调整块大小。对于创建很大的文件但访问较小的随机块中的文件的数据库而言，这些算法可能不是最优的。将 `recordsize` 值指定为大于或等于数据库的记录大小的值可以显著提高性能。强烈建议不要将该属性用于一般用途的文件系统，否则可能会对性能产生不利影响。指定的大小必须是 2 的若干次幂，并且必须大于或等于 512 字节同时小于或等于 1 MB。更改文件系统的 `recordsize` 值仅影响之后创建的文件。现有文件不会受到影响。

此属性的缩写为 `recsize`。

`share.smb` 属性

此属性通过 Oracle Solaris SMB 服务启用 ZFS 文件系统的共享，并标识要使用的选项。

当属性从 off 更改为 on 时，任何继承该属性的共享将使用其当前选项重新共享。此属性设置为 off 时，继承此属性的共享将会取消共享。有关使用 `share.smb` 属性的示例，请参见“共享和取消共享 ZFS 文件系统” [142]。

`volsize` 属性

`volsize` 属性指定卷的逻辑大小。缺省情况下，创建卷会产生相同大小的预留空间。对 `volsize` 的任何更改都会反映为对预留空间的等效更改。这些检查用来防止用户产生的意外行为。如果卷包含的空间比其声明可用的空间少，则会导致未定义的行为或数据损坏，具体取决于卷的使用方法。如果在卷的使用过程中更改卷大小，特别是在收缩大小时，也会出现上述影响。调整卷大小时，应该格外小心。

有关使用卷的更多信息，请参见“ZFS 卷” [227]。

ZFS 用户属性

除了本机属性外，ZFS 还支持任意用户属性。用户属性对 ZFS 行为没有影响，但可通过用户环境中有意义的信息来注释数据集。

用户属性名称必须符合以下约定：

- 必须包含冒号字符 (':')，以与本机属性相区分。
- 必须包含小写字母、数字或以下标点符号：':'、'+'、'.'、'_'。
- 用户属性名称的最大长度为 256 个字符。

预期约定是属性名分为以下两个部分，但 ZFS 不强制使用此名称空间：

module:property

在程序中使用用户属性时，请对属性名的 *module* 部分使用反向 DNS 域名，以尽量避免两个独立开发的软件包将同一属性名用于不同用途。以 `com.oracle.` 开头的属性名保留供 Oracle Corporation 使用。

用户属性的值必须符合以下约定：

- 必须由始终继承且从不验证的任意字符串组成。
- 用户属性值的最大长度为 1024 个字符。

例如：

```
# zfs set dept:users=finance userpool/user1
# zfs set dept:users=general userpool/user2
# zfs set dept:users=itops userpool/user3
```

对属性执行操作的所有命令（如 `zfs list`、`zfs get`、`zfs set` 等）都可用来处理本机属性和用户属性。

例如：

```
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
```

要清除某一用户属性，请使用 `zfs inherit` 命令。例如：

```
# zfs inherit -r dept:users userpool
```

如果任意父数据集中均未定义该属性，则会将其完全删除。

查询 ZFS 文件系统信息

`zfs list` 命令提供了一种用于查看和查询数据集信息的可扩展机制。本节中对基本查询和复杂查询都进行了说明。

列出基本 ZFS 信息

通过使用不带任何选项的 `zfs list` 命令可以列出基本数据集信息。此命令可显示系统中所有数据集的名称，以及其 `used`、`available`、`referenced` 和 `mountpoint` 属性的值。有关这些属性的更多信息，请参见“介绍 ZFS 属性” [116]。

例如：

```
# 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
```

另外，还可使用此命令通过在命令行中提供数据集名称来显示特定数据集。此外，使用 -r 选项将以递归方式显示该数据集的所有后代。例如：

```
# 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  -
```

您可以结合文件系统的挂载点使用 zfs list 命令。例如：

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

以下示例说明了如何显示关于 tank/home/gina 及其所有后代文件系统的基本信息：

```
# 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
```

有关 zfs list 命令的其他信息，请参见 zfs(1M)。

创建复杂的 ZFS 查询

使用 o、-t 和 -H 选项可对 -zfs list 输出进行定制。

通过使用 -o 选项以及所需属性的逗号分隔列表可以定制属性值输出。可以将任何数据集属性作为有效参数提供。有关所有受支持的数据集属性的列表，请参见“介绍 ZFS 属性” [116]。除了定义的属性外，-o 选项列表还可以包含字符 name，以指明输出应包括数据集的名称。

以下示例使用 zfs list 来显示数据集名称以及 share.nfs 和 mountpoint 属性值。

```
# 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
```

您可以使用 -t 选项指定要显示的数据集类型。下表中介绍了有效的类型。

表 5-2 ZFS 对象的类型

类型	说明
filesystem	文件系统和克隆
volume	卷
share	文件系统共享
snapshot	快照

-t 选项接受要显示的数据集类型的逗号分隔列表。以下示例同时使用 -t 和 -o 选项来显示所有文件系统的名称和 used 属性：

```
# 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
```

您可以使用 -H 选项在生成的输出中省略 zfs list 标题。使用 -H 选项时，所有空格都被 Tab 字符取代。当需要可解析的输出（例如编写脚本时），此选项可能很有用。以下示例显示了使用带有 H 选项的 -zfs list 命令所生成的输出：

```
# 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
```

管理 ZFS 属性

数据集属性通过 zfs 命令的 set、inherit 和 get 子命令来管理。

设置 ZFS 属性

可以使用 `zfs set` 命令修改任何可设置的数据集属性。或者，也可以使用 `zfs create` 命令在创建数据集时设置属性。有关可设置的数据集属性的列表，请参见“可设置的 ZFS 本机属性” [125]。

`zfs set` 命令采用 *property=value* 格式的属性/值序列，然后是数据集名称。每次调用 `zfs set` 时只能设置或修改一个属性。

以下示例将 `tank/home` 的 `atime` 属性设置为 `off`。

```
# zfs set atime=off tank/home
```

此外，任何文件系统属性均可在创建文件系统时设置。例如：

```
# zfs create -o atime=off tank/home
```

可以使用以下易于理解的后缀（按大小递增顺序）指定数字属性值：`BKMGTPEZ`。其中任一后缀都可后跟可选的 `b`，用于表示字节，但 `B` 后缀除外，因为它已表示了字节。以下四个 `zfs set` 调用是等效的数字表达式，在 `users/home/mark` 文件系统中将 `quota` 属性设置为值 20 GB：

```
# 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
```

如果尝试在 100% 全满的文件系统上设置属性，则会显示类似于以下内容的消息：

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

非数字属性的值区分大小写，并且必须为小写字母，但 `mountpoint` 除外。该属性的值既可以包含大写字母，也可以包含小写字母。

有关 `zfs set` 命令的更多信息，请参见 `zfs(1M)`。

继承 ZFS 属性

除非已对后代文件系统显式设置了配额或预留空间，否则除了配额和预留空间外，所有可设置的属性都从父文件系统继承各自的值。如果没有祖先为继承的属性设置显式值，

则使用该属性的缺省值。可以使用 zfs inherit 命令清除某个属性值，从而促使从父文件系统继承该值。

以下示例使用 zfs set 命令为 tank/home/jeff 文件系统启用压缩。然后，使用 zfs inherit 清除 compression 属性，从而使该属性继承缺省值 off。由于 home 和 tank 都未本地设置 compression 属性，因此会使用缺省值。如果两者都启用了压缩，则使用最直接的祖先中设置的值（在本示例中为 home）。

```
# 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
```

如果指定了 -r 选项，则会以递归方式应用 inherit 子命令。在以下示例中，该命令将使 tank/home 以及它可能具有的所有后代都继承 compression 属性的值：

```
# zfs inherit -r compression tank/home
```

注 - 请注意，使用 -r 选项会清除所有后代文件系统的当前属性设置。

有关 zfs inherit 命令的更多信息，请参见 zfs(1M)。

查询 ZFS 属性

查询属性值的最简单方法是使用 zfs list 命令。有关更多信息，请参见“列出基本 ZFS 信息” [131]。但是，对于复杂查询和脚本编写，请使用 zfs get 命令以定制格式提供更详细的信息。

可以使用 zfs get 命令检索任何数据集属性。以下示例说明如何在数据集中检索单个属性值：

```
# zfs get checksum tank/ws
NAME            PROPERTY       VALUE                    SOURCE
tank/ws         checksum       on                       default
```

第四栏 SOURCE 表示此属性值的来源。下表定义可能的源值。

表 5-3 可能的 SOURCE 值 (zfs get 命令)

Source 值	说明
default	从来不为数据集或其任何祖先显式设置此属性值。使用的是该属性的缺省值。
inherited from *dataset-name*	该属性值继承自 *dataset-name* 所指定的父数据集。
local	使用 zfs set 可为此数据集显式设置该属性值。
temporary	该属性值是使用 zfs mount -o 选项设置的，并且仅在挂载期间有效。有关临时挂载点属性的更多信息，请参见"使用临时挂载属性" [141]。
- (none)	此属性为只读。其值由 ZFS 生成。

可以使用特殊关键字 all 检索所有数据集属性值。以下示例使用 all 关键字:

```
# zfs get all tank/home
NAME       PROPERTY              VALUE                  SOURCE
tank/home  type                  filesystem             -
tank/home  creation              Mon Dec  3 13:10 2012  -
tank/home  used                  291K                   -
tank/home  available             58.7G                  -
tank/home  referenced            291K                   -
tank/home  compressratio         1.00x                  -
tank/home  mounted               yes                    -
tank/home  quota                 none                   default
tank/home  reservation           none                   default
tank/home  recordsize            128K                   default
tank/home  mountpoint            /tank/home             default
tank/home  sharenfs              off                    default
tank/home  checksum              on                     default
tank/home  compression           off                    default
tank/home  atime                 on                     default
tank/home  devices               on                     default
tank/home  exec                  on                     default
tank/home  setuid                on                     default
tank/home  readonly              off                    default
tank/home  zoned                 off                    default
tank/home  snapdir               hidden                 default
tank/home  aclmode               discard                default
tank/home  aclinherit            restricted             default
tank/home  canmount              on                     default
tank/home  shareiscsi            off                    default
tank/home  xattr                 on                     default
tank/home  copies                1                      default
tank/home  version               5                      -
tank/home  utf8only              off                    -
tank/home  normalization         none                   -
tank/home  casesensitivity       mixed                  -
tank/home  vscan                 off                    default
tank/home  nbmand                off                    default
tank/home  sharesmb              off                    default
tank/home  refquota              none                   default
tank/home  refreservation        none                   default
```

```
tank/home  primarycache          all                  default
tank/home  secondarycache        all                  default
tank/home  usedbysnapshots       0                    -
tank/home  usedbydataset         291K                 -
tank/home  usedbychildren        0                    -
tank/home  usedbyrefreservation  0                    -
tank/home  logbias               latency              default
tank/home  sync                  standard             default
tank/home  rekeydate             -                    default
tank/home  rstchown              on                   default
```

`zfs get` 的 `-s` 选项用于按照源类型指定要显示的属性。通过此选项可获取一个逗号分隔列表，用于指明所需的源类型。仅会显示具有指定源类型的属性。有效的源类型包括 `local`、`default`、`inherited`、`temporary` 和 `none`。以下示例显示了在 `tank/ws` 上本地设置的所有属性。

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

以上任何选项均可与 `-r` 选项结合使用，以便以递归方式显示指定文件系统的所有子级的指定属性。在以下示例中，以递归方式显示了 `tank/home` 中所有文件系统的所有临时属性：

```
# 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
```

可以在不指定目标文件系统的情况下使用 `zfs get` 命令查询属性值，这意味着该命令对所有池或文件系统有效。例如：

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

有关 `zfs get` 命令的更多信息，请参见 `zfs(1M)`。

查询用于编写脚本的 ZFS 属性

`zfs get` 命令支持为编写脚本而设计的 `-H` 和 `-o` 选项。可以使用 `-H` 选项省去标题信息并用 Tab 字符替换空格。使用一致的空格可使数据便于解析。可以使用 `-o` 选项以如下方式定制输出：

- 字符 `name` 可以与逗号分隔的属性列表一起使用，如 “介绍 ZFS 属性” [116] 部分所述。
- 输出逗号分隔的字面字段 `name`、`value`、`property` 和 `source` 的列表，后面跟随空格和参数，这就是逗号分隔的属性列表。

以下示例说明如何使用 -zfs get 的 -H 和 o 选项来检索单个值：

```
# zfs get -H -o value compression tank/home
on
```

-p 选项会将数字值报告为精确值。例如，1MB 将报告为 1000000。此选项可按如下方式使用：

```
# zfs get -H -o value -p used tank/home
182983742
```

可以结合使用 -r 选项与前述任何选项，以递归方式为所有后代检索请求值。以下示例使用 -H、-o 和 -r 选项检索文件系统名称和 export/home 及其后代的 used 属性值，同时省略标题输出：

```
# zfs get -H -o name,value -r used export/home
```

挂载 ZFS 文件系统

本节介绍了 ZFS 如何挂载文件系统。

管理 ZFS 挂载点

缺省情况下，ZFS 文件系统在创建时自动挂载。可以确定文件系统的特定挂载点行为，如本节所述。

另外，也可以在创建时使用 zpool create 的 -m 选项为池文件系统设置缺省挂载点。有关创建池的更多信息，请参见“创建 ZFS 存储池” [34]。

所有 ZFS 文件系统都由 ZFS 通过使用服务管理工具 (Service Management Facility, SMF) 的 svc://system/filesystem/local 服务在引导时挂载。文件系统挂载在 */path* 下，其中 *path* 是文件系统的名称。

可以使用 zfs set 命令将 mountpoint 属性设置为特定路径，以覆盖缺省挂载点。ZFS 自动创建指定的挂载点（如果需要），并自动挂载关联的文件系统。

ZFS 文件系统无需您编辑 /etc/vfstab 文件即可在引导时自动挂载。

mountpoint 属性是继承的。例如，如果 pool/home 的 mountpoint 属性设置为 /export/stuff，则 pool/home/user 将继承 /export/stuff/user 的 mountpoint 属性值。

要防止挂载文件系统，请将 mountpoint 属性设置为 none。此外，canmount 属性可以用来控制是否能挂载文件系统。有关 canmount 属性的更多信息，请参见“canmount 属性” [126]。

也可以使用 zfs set 将 mountpoint 属性设置为 legacy，从而通过传统挂载接口显式管理文件系统。这样做可以防止 ZFS 自动挂载和管理文件系统。不过必须改用包括 mount 和 umount 命令在内的传统工具以及 /etc/vfstab 文件。有关传统挂载的更多信息，请参见“传统挂载点” [139]。

自动挂载点

- 将 mountpoint 属性从 legacy 或 none 更改为特定路径时，ZFS 会自动挂载文件系统。
- 如果 ZFS 正在管理文件系统，但该文件系统当前已取消挂载，并且 mountpoint 属性已更改，则文件系统将保持取消挂载状态。

mountpoint 属性不是 legacy 的所有文件系统都由 ZFS 来管理。在以下示例中，创建了一个挂载点由 ZFS 自动管理的文件系统：

```
# 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                       -
```

另外，也可按以下示例所示，显式设置 mountpoint 属性：

```
# 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                       -
```

如果更改了 mountpoint 属性，文件系统将自动从旧挂载点取消挂载，并重新挂载到新挂载点。挂载点目录根据需要进行创建。如果 ZFS 由于文件系统正处于活动状态而无法将其取消挂载，则会报告错误，并需要强制进行手动取消挂载。

传统挂载点

通过将 mountpoint 属性设置为 legacy，可以使用传统工具来管理 ZFS 文件系统。传统文件系统必须通过 mount 和 umount 命令以及 /etc/vfstab 文件来管理。ZFS 在引导时不

会自动挂载传统文件系统，并且 ZFS `mount` 和 `umount` 命令不会对此类型的文件系统执行操作。以下示例展示了如何在传统模式下设置和管理 ZFS 文件系统：

```
# zfs set mountpoint=legacy tank/home/eric
# mount -F zfs tank/home/eschrock /mnt
```

要在引导时自动挂载传统文件系统，必须向 `/etc/vfstab` 文件中添加一项。`/etc/vfstab` 文件中的项类似于下例：

```
#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` 和 `fsck pass` 项设置为 `-`，因为 `fsck` 命令不适用于 ZFS 文件系统。有关 ZFS 数据完整性的更多信息，请参见 “事务性语义” [14]。

挂载 ZFS 文件系统

创建文件系统或系统引导时，ZFS 会自动挂载文件系统。仅当需要更改挂载选项，或者显式挂载或取消挂载文件系统时，才有必要使用 `zfs mount` 命令。

不带任何参数的 `zfs mount` 命令可以显示 ZFS 管理的当前已挂载的所有文件系统。传统管理的挂载点不会显示。例如：

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

可以使用 -a 选项挂载 ZFS 管理的所有文件系统。传统管理的文件系统不会挂载。例如：

```
# zfs mount -a
```

在缺省情况下，ZFS 不允许挂载到非空目录上。例如：

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

传统挂载点必须通过传统工具进行管理。尝试使用 ZFS 工具将产生错误。例如：

```
# 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
```

当挂载文件系统时，它根据与文件系统关联的属性值使用一组挂载选项。属性与挂载选项之间的相互关系如下：

表 5-4 ZFS 挂载相关的属性和挂载选项

属性	挂载选项
atime	atime/noatime
devices	devices/nodevices
exec	exec/noexec
nbmand	nbmand/nonbmand
readonly	ro/rw
setuid	setuid/nosetuid
xattr	xattr/noaxttr

挂载选项 nosuid 是 nodevices,nosetuid 的别名。

可以使用 NFSv4 镜像挂载功能来帮助您更好地管理已挂载 NFS 的 ZFS 起始目录。

在 NFS 服务器上创建文件系统后，NFS 客户机可以在这些新创建的文件系统的现有父文件系统挂载内自动对其进行搜索。

例如，如果服务器 neo 已共享了 tank 文件系统并且客户机 zee 已将其挂载，则在该服务器上创建 /tank/baz 后，它会在该客户机上自动可见。

```
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
```

使用临时挂载属性

如果使用带有 -o 选项的 zfs mount 命令显式设置了前一部分所述的任何挂载选项，则会临时覆盖关联的属性值。zfs get 命令将这些属性值报告为 temporary，并在文件系统取消挂载时恢复为其初始值。如果在挂载文件系统时更改了某个属性值，更改将立即生效，并覆盖所有临时设置。

在以下示例中，对 tank/home/neil 文件系统临时设置了只读挂载选项。假设要取消挂载文件系统。

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

要临时更改当前已挂载的文件系统的属性值，必须使用特殊的 remount 选项。在以下示例中，对于当前挂载的文件系统，atime 属性暂时更改为 off：

```
# zfs mount -o remount,noatime users/home/neil
NAME            PROPERTY  VALUE  SOURCE
users/home/neil  atime     off    temporary
# zfs get atime users/home/perrin
```

有关 `zfs mount` 命令的更多信息，请参见 `zfs(1M)`。

取消挂载 ZFS 文件系统

通过使用 `zfs unmount` 子命令可以取消挂载 ZFS 文件系统。`unmount` 命令可以接受挂载点或文件系统名称作为参数。

在以下示例中，按文件系统名称取消挂载一个文件系统：

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

在以下示例中，按挂载点取消挂载一个文件系统：

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

如果文件系统处于繁忙状态，则 `unmount` 命令将失败。要强行取消挂载文件系统，可以使用 `-f` 选项。如果文件系统内容正处于使用状态，强行取消挂载该文件系统时请务必小心。否则，会产生不可预测的应用程序行为。

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

要提供向后兼容性，可以使用传统的 `umount` 命令来取消挂载 ZFS 文件系统。例如：

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

有关 `zfs umount` 命令的更多信息，请参见 `zfs(1M)`。

共享和取消共享 ZFS 文件系统

Oracle Solaris 11.1 发行版通过利用 ZFS 属性继承来简化 ZFS 共享管理。在运行池版本 34 的池中启用了新的共享语法。

以下为 NFS 和 SMB 的文件系统软件包：

- NFS 客户机和服务器软件包
  - `service/file-system/nfs`（服务器）
  - `service/file-system/nfs`（客户机）

  有关其他 NFS 配置信息，请参见《在 Oracle Solaris 11.2 中管理网络文件系统》。

- SMB 客户机和服务器软件包
  - `service/file-system/smb`（服务器）
  - `service/file-system/smb`（客户机）

  有关其他 SMB 配置信息（包括 SMB 口令管理），请参见《Managing SMB File Sharing and Windows Interoperability in Oracle Solaris 11.2 》中的“Managing SMB Mounts in Your Local Environment”。

每个文件系统可定义多个共享。一个共享名可唯一标识一个共享。您可以定义用来共享文件系统中的特定路径的属性。缺省情况下，所有文件系统都不共享。通常，在创建共享之前，NFS 服务器服务不会启动。如果创建了有效的共享，NFS 服务将自动启动。如果 ZFS 文件系统的 `mountpoint` 属性设置为 legacy，则只能通过使用传统的 `share` 命令共享该文件系统。

- `share.nfs` 属性替换以前发行版中用于定义和发布 NFS 共享的 `sharenfs` 属性。
- `share.smb` 属性替换以前发行版中用于定义和发布 SMB 共享的 `sharesmb` 属性。
- `sharenfs` 属性和 `sharesmb` 属性分别是 `share.nfs` 属性和 `sharenfs` 属性的别名。
- `/etc/dfs/dfstab` 文件不再用于在引导时共享文件系统。设置这些属性以自动共享文件系统。SMF 管理 ZFS 或 UFS 共享信息，以便在重新引导系统时自动共享文件系统。此功能意味着，其 `sharenfs` 或 `sharesmb` 属性未设置为 off 的所有文件系统在引导时均处于共享状态。
- `sharemgr` 接口不再可用。传统的 `share` 命令仍可用于创建传统的共享。请参见下面的示例。
- `share-a` 命令与以前的 `share -ap` 命令类似，因此，共享文件系统将是持久性的。`share -p` 选项不再可用。

例如，如果要共享 `tank/home` 文件系统，请使用如下语法：

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

在上一示例中，对 `tank/home` 文件系统设置了 `share.nfs` 属性，`share.nfs` 属性值将继承到任何后代文件系统。例如：

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

您还可以在现有文件系统共享上指定其他属性值，也可以修改现有属性值。例如：

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

传统的 ZFS 共享语法

Oracle Solaris 11 语法仍受支持，因此，您可以分两步共享文件系统。此语法在所有池版本中均受支持。

- 首先使用 `zfs set share` 命令创建 ZFS 文件系统的 NFS 或 SMB 共享。

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

- 然后将 `sharenfs` 或 `sharesmb` 属性设置为 on 以发布共享。例如：

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

可以使用传统的 `zfs get share` 命令显示文件系统共享。

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

此外，用于共享文件系统的 `share` 命令仍受支持，以共享文件系统中的任何目录，其语法与 Oracle Solaris 10 发行版中的语法类似。例如，要共享某个 ZFS 文件系统：

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

上面的语法与共享 UFS 文件系统的语法完全一致：

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

新的 ZFS 共享语法

`zfs set` 命令用于通过 NFS 或 SMB 协议共享和发布 ZFS 文件系统。或者，您可以在创建文件系统时设置 `share.nfs` 或 `share.smb` 属性。

例如，创建和共享 `tank/sales` 文件系统。对于每个用户，缺省共享权限均为读写权限。后代 `tank/sales/logs` 文件也将自动共享，因为 `share.nfs` 属性会继承到后代文件系统且 `tank/sales/log` 文件系统会设置为只读访问权限。

```
# 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
```

您可以按照以下方式为共享文件系统提供特定系统的 root 访问权限：

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

包含每属性继承的 ZFS 共享

在已升级到最新池版本 34 的池中，提供了新的共享语法，可利用 ZFS 属性继承，从而更轻松地进行共享维护。每个共享特征将成为单独的 share 属性。share 属性由以 share. 前缀开头的名称标识。share 属性示例包括 share.desc、share.nfs.nosuid 和 share.smb.guestok。

share.nfs 属性控制是否启用 NFS 共享。share.smb 属性控制是否启用 SMB 共享。传统的 sharenfs 和 sharesmb 属性名称仍可用，因为在新池中，sharenfs 是 share.nfs 的别名，sharesmb 是 share.smb 的别名。如果要共享 tank/home 文件系统，请使用类似如下的语法：

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

在此示例中，share.nfs 属性值会继承到任何后代文件系统。例如：

```
# 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
```

旧池中的 ZFS 共享继承

在旧池中，只有 sharenfs 和 sharesmb 属性由后代文件系统继承。其他共享特征均存储在每个共享的 .zfs/shares 文件中，不会被继承。

一个特殊规则是，只要创建了从其父项继承 sharenfs 或 sharesmb 的新文件系统，就会基于 sharenfs 或 sharesmb 值为此文件系统创建缺省共享。请注意，如果直接将 sharenfs 设置为 on，则在后代文件系统中创建的缺省共享只具有缺省 NFS 特征。例如：

```
# 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 命名共享

您可以创建命名共享，它可以在 SMB 环境中设置权限和属性时提供更多的灵活性。例如：

```
# zfs share -o share.smb=on tank/workspace%myshare
```

在上一示例中，`zfs share` 命令为 `tank/workspace` 文件系统创建了名为 `myshare` 的 SMB 共享。您可以通过此文件系统的 `.zfs/shares` 目录访问 SMB 共享以及显示或设置特定权限或 ACL。每个 SMB 共享均由单独的 `.zfs/shares` 文件表示。例如：

```
# 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
```

命名共享从父文件系统继承共享属性。如果在上一示例中将 `share.smb.guestok` 属性添加到父文件系统，此属性将继承到命名共享。例如：

```
# 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
```

在为文件系统的子目录定义共享时，命名服务在 NFS 环境中可能会很有帮助。例如：

```
# 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
```

上一示例还说明了，在保持所有其他属性继承不变的情况下，将文件系统的 `share.auto` 设置为 off 将禁用该文件系统的自动共享。与大多数其他共享属性不同，`share.auto` 属性不可继承。

创建公共 NFS 共享时还可以使用命名共享。只能在命名 NFS 共享上创建公共共享。例如：

```
# 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
```

有关 NFS 和 SMB 共享属性的详细说明，请参见 share_nfs(1M) 和 share_smb(1M)。

ZFS 自动共享

创建自动共享时，会从文件系统名称中构建唯一资源名称。构建的名称是文件系统名称的副本，但是，对于文件系统名称中的字符，如果在资源名称中不合法，将被替换为下划线字符 (_)。例如，data/home/john 的资源名称是 data_home_john。

通过设置 share.autoname 属性名称，可以在创建自动共享时将文件系统名称替换为特定名称。在继承时还会使用该特定名称替换前缀文件系统名称。例如：

```
# 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
```

如果在尚未处于共享状态的文件系统上使用传统的 share 命令或 zfs set share 命令，其 share.auto 值将自动设置为 off。传统的命令始终会创建命名共享。此特殊规则可防止自动共享干扰要创建的命名共享。

显示 ZFS 共享信息

通过使用 zfs get 命令显示文件共享属性的值。以下示例显示了如何显示一个文件系统的 share.nfs 属性：

```
# zfs get share.nfs tank/sales
NAME        PROPERTY   VALUE  SOURCE
tank/sales  share.nfs  on     local
```

以下示例显示了如何显示后代文件系统的 share.nfs 属性：

```
# 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 命令语法不能获取扩展共享属性信息。

可以通过使用以下语法显示有关 NFS 或 SMB 共享信息的特定详细信息：

```
# 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
```

因为存在多个共享属性，请考虑使用非缺省值显示属性。例如：

```
# 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
```

更改 ZFS 共享属性值

可以通过在文件系统共享上指定新属性或修改的属性来更改共享属性值。例如，如果在创建文件系统时设置只读属性，则可将此属性设置为 off。

```
# zfs create -o share.nfs.ro=\* tank/data
# zfs get share.nfs.ro tank/data
NAME       PROPERTY             VALUE  SOURCE
tank/data  share.nfs.sec.sys.ro  *     local
# zfs set share.nfs.ro=none tank/data
# zfs get share.nfs.ro tank/data
NAME       PROPERTY             VALUE  SOURCE
tank/data  share.nfs.sec.sys.ro  off    local
```

如果创建 SMB 共享，还可以添加 NFS 共享协议。例如：

```
# 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     -
```

删除 SMB 协议：

```
# zfs set share.smb=off tank/multifs
# grep multifs /etc/dfs/sharetab
/tank/multifs   tank_multifs    nfs     sec=sys,rw
```

您可以重命名已命名的共享。例如：

```
# 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     -
```

发布和取消发布 ZFS 共享

您可以通过使用 `zfs unshare` 命令临时取消共享命名共享而不将其销毁。例如：

```
# 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     -
```

发出 `zfs unshare` 命令时，会取消共享所有文件系统共享。在对文件系统发出 `zfs share` 命令之前或为文件系统设置 `share.nfs` 或 `share.smb` 属性之前，这些共享将保持独享状态。

在发出 `zfs unshare` 命令时不会删除已定义的共享，在下次对文件系统发出 `zfs share` 命令时或为文件系统设置 `share.nfs` 或 `share.smb` 属性时，它们会重新处于共享状态。

删除 ZFS 共享

您可以通过将 `share.nfs` 或 `share.smb` 属性设置为 off 来取消文件系统共享。例如：

```
# zfs set share.nfs=off tank/multifs
# grep multifs /etc/dfs/sharetab
#
```

您可以使用 `zfs destroy` 命令永久删除命名共享。例如：

```
# zfs destroy tank/home/abc%a1share
```

非全局区域中的 ZFS 文件共享

从 Oracle Solaris 11 开始，您可以在 Oracle Solaris 非全局区域创建和发布 NFS 共享。

- 如果挂载了某个 ZFS 文件系统且它在非全局区域中可用，则可以在该区域中共享它。

- 如果文件系统未委托给非全局区域或者未在非全局区域中挂载，则可以在全局区域中共享该文件系统。如果文件系统已添加到非全局区域，则只能使用传统的 `share` 命令来共享该文件系统。

例如，`/export/home/data` 和 `/export/home/data1` 文件系统在 `zfszone` 中可用。

```
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 共享迁移/转换问题

检查以下转换问题：

- **导入带有旧共享属性的文件系统** – 导入池或接收文件系统流（它们均在 Oracle Solaris 11 之前创建）时，`sharenfs` 和 `sharesmb` 属性会在属性值中直接包括所有共享属性。在大多数情况下，一旦共享每个文件系统，这些传统的共享属性就会转换为一组等效的命名共享。因为在大多数情况下，导入操作会触发挂载和共享，所以，在导入过程中会直接发生转换为命名共享的操作。
- **从 Oracle Solaris 11 升级** – 在池升级到版本 34 之后，第一个文件系统共享需要的时间可能会很长，因为命名服务需要转换为新格式。升级过程中创建的命名服务是正确的，但不能利用共享属性继承。
  - 显示共享属性值：

    ```
    # zfs get share.nfs filesystem
    # zfs get share.smb filesystem
    ```

  - 如果引导回较旧的 BE，请将 `sharenfs` 和 `sharesmb` 属性重置为其原始值。
- **从 Oracle Solaris 11 升级** – 在 Oracle Solaris 11 和 11.1 中，`sharenfs` 和 `sharesmb` 属性只能有 `off` 和 `on` 两个值。这些属性不再用于定义共享特征。

  `/etc/dfs/dfstab` 文件不再用于在引导时共享文件系统。引导时，将自动共享包括已启用文件系统共享的所有已挂载 ZFS 文件系统。将 `sharenfs` 或 `sharesmb` 设置为 `on` 时，将启用共享。

  `sharemgr` 接口不再可用。传统的 `share` 命令仍可用于创建传统的共享。`share-a` 命令与以前的 `share -ap` 命令类似，因此，共享文件系统将是持久性的。`share -p` 选项不再可用。
- **升级系统** – 如果因该发行版中的属性更改而引导回 Oracle Solaris 11 BE，则 ZFS 共享将是不正确的。非 ZFS 共享不受影响。如果打算引导回较旧的 BE，则应在执行 `pkg update` 操作之前首先保存现有共享配置的副本，以便能够恢复 ZFS 共享配置。

  在较旧的 BE 中，使用 `sharemgr show -vp` 命令可列出所有共享及其配置。

  使用以下命令显示共享属性值：

  ```
  # zfs get sharenfs filesystem
  ```

```
# zfs get sharesmb filesystem
```

如果返回到较旧的 BE，请将 sharenfs 和 sharesmb 属性以及使用 sharemgr 定义的所有共享重置为其原始值。

- **传统的取消共享行为** – 使用 unshare -a 命令或 unshareall 命令可取消共享文件系统，但是不会更新 SMF 共享系统信息库。如果尝试重新共享现有的共享，则会检查到共享系统信息库中的冲突，并显示一个错误。

排除 ZFS 文件系统共享问题

检查以下共享错误情况：

- **新共享或之前的共享未共享**
  - **确认池和文件系统版本为新版本** – 如果通过设置 share.nfs 或 share.smb 属性未共享新共享，则确认池版本是否为 34，文件系统版本是否为 6。
  - **在 NFS 服务启动前共享必须已存在** – 在共享文件系统后 NFS 服务器服务才会运行。首先创建 NFS 共享，然后尝试远程访问共享。
  - **升级了存在共享的系统，但共享不可用** – 升级了存在共享的系统，但是尝试重新共享这些共享时失败。这些共享无法共享，因为禁用了 share.auto 属性。如果 share.auto 设置为 off，则只能使用命名共享，以强制与早期的共享语法兼容。现有共享可能类似于：

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

    1. 确保启用了 share.auto 属性。如果未启用，启用该属性。

       ```
       # zfs get -r share.auto tank/data
       # zfs set share.auto=on tank/data
       ```

    2. 重新共享文件系统。

       ```
       # zfs set -r share.nfs=on tank/data
       ```

    3. 您可能还需要删除命名共享，然后重新创建，这样前面的命令才能成功运行。

       ```
       # zfs list -t share -Ho name -r tank/data | xargs -n1 zfs destroy
       ```

    4. 如果必要，重新创建命名共享。

       ```
       # zfs create -o share.nfs=on tank/data%share
       ```

- **快照中未包含包括命名共享的共享属性** – 在 zfs clone 和 zfs send 操作中，共享属性和 .zfs/shares 文件的处理方式不同。.zfs/shares 文件包含在快照中并保留在 zfs clone 和 zfs send 操作。有关 zfs send 和 zfs receive 操作期间的属性行为的说明，请参见“向 ZFS 快照流应用不同的属性值” [183]。完成克隆操作之后，所有文件均来自上一克隆快照，而属性将继承自克隆在 ZFS 文件系统分层结构中的新位置。

- **命名共享请求失败** – 如果创建命名共享的请求由于此共享与自动共享冲突而失败，则可能需要禁用 `auto.share` 属性。
- **存在共享的池之前已导出** – 如果以只读方式导入池，则该池的属性和文件都无法修改，因此创建新共享会失败。如果在导出池之前存在共享，则可以使用现有共享特征（如果可能）。

下表列出了已知共享状态以及解决它们的方式（如果需要）。

共享状态	说明	解决方法
`INVALID`	共享无效，因为它内部不一致或者与其他共享冲突。	尝试使用以下命令重新共享无效共享： **`# zfs share FS%share`** 使用以下命令会显示有关共享的哪个部分未通过验证的错误消息。先更正此错误，然后重试此共享。
`SHARED`	共享处于共享状态。	无需执行任何操作。
`UNSHARED`	共享有效但已取消共享。	使用 `zfs share` 命令重新共享单个共享或父文件系统。
`UNVALIDATED`	共享尚无效。包含此共享的文件系统可能未处于可共享状态。例如，它未挂载或已委托给当前区域之外的某个区域。或者，表示所需共享的 ZFS 属性已创建，但尚未证实为合法共享。	使用 `zfs share` 命令重新共享单个共享或父文件系统。如果文件系统本身可共享，则尝试重新共享要么成功共享（并将状态转换为 "SHARED"（已共享）），要么不能共享（并将状态转换为 "INVALID"（无效））。或者，您可以使用 `share -A` 命令列出所有已挂载文件系统的所有共享。这将导致将已挂载文件系统中的所有共享解析为 "UNSHARED"（独享）（有效但尚未共享）或 "INVALID"（无效）。

设置 ZFS 配额和预留空间

可以使用 `quota` 属性对文件系统可以使用的磁盘空间量设置限制。此外，还可以使用 `reservation` 属性来保证预留一定的磁盘空间量供文件系统使用。这两个属性将应用于设置了它们的文件系统以及该文件系统的所有后代。

也就是说，如果对 `tank/home` 文件系统设置了配额，则 `tank/home` 及其所有后代使用的总磁盘空间量不能超过该配额。同样，如果为 `tank/home` 指定了预留空间，则 `tank/home` 及其所有后代都会使用该预留空间。文件系统及其所有后代使用的磁盘空间量由 `used` 属性进行报告。

`refquota` 和 `refreservation` 属性用于管理文件系统空间，但不会将后代（如快照和克隆）占用的磁盘空间计算在内。

在此 Solaris 发行版中，您可以根据属于特定用户或组的文件所占用的磁盘空间量来设置 *user* 或 *group* 配额。不能基于卷、早于文件系统版本 4 的文件系统或早于池版本 15 的池设置用户和组配额属性。

确定哪个配额和预留空间功能更有利于管理您的文件系统时，请注意以下几点：

- 管理文件系统及其后代使用的磁盘空间时，使用 quota 和 reservation 属性会很方便。
- refquota 和 refreservation 属性适合于管理文件系统占用的磁盘空间。
- 将 refquota 或 refreservation 属性设置为高于 quota 或 reservation 属性无效。如果设置了 quota 或 refquota 属性，则尝试超出任一值的操作都将失败。可能会超出大于 refquota 的 quota。例如，如果有些快照块被修改，则可能在超出 refquota 之前实际已超出 quota。
- 用户和组配额提供了一种方法，可以在具有很多用户帐户的情况下更轻松地管理磁盘空间，例如在大学环境里。

有关设置配额和预留空间的更多信息，请参见“设置 ZFS 文件系统的配额” [153]和“设置 ZFS 文件系统的预留空间” [156]。

设置 ZFS 文件系统的配额

使用 zfs set 和 zfs get 命令可以设置和显示 ZFS 文件系统的配额。在以下示例中，在 tank/home/jeff 上设置了 10 GB 的配额：

```
# zfs set quota=10G tank/home/jeff
# zfs get quota tank/home/jeff
NAME            PROPERTY  VALUE  SOURCE
tank/home/jeff  quota     10G    local
```

配额还会影响 zfs list 和 df 命令的输出。例如：

```
# 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
```

请注意，虽然 tank/home 具有 66.9 GB 的可用磁盘空间，但由于 tank/home/jeff 存在配额，tank/home/jeff 和 tank/home/jeff/ws 各自仅有 10 GB 的可用磁盘空间。

可对文件系统设置 refquota，以限制该文件系统可以使用的磁盘空间量。硬限制不包括后代所占用的磁盘空间。例如，快照占用的空间不会影响 studentA 的 10 GB 配额。

```
# 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  -
```

为了更加方便，可对文件系统设置其他配额，以帮助管理快照使用的磁盘空间。例如：

```
# 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  -
```

在此情况下，studentA 可能会达到 refquota (10 GB) 硬限制，但studentA可以删除文件进行恢复，即使存在快照也是如此。

在上例中，zfs list 输出显示两个配额中的较小者（10 GB 与 20 GB 相比较小）。要查看两个配额的值，请使用 zfs get 命令。例如：

```
# zfs get refquota,quota students/studentA
NAME              PROPERTY  VALUE              SOURCE
students/studentA  refquota  10G                local
students/studentA  quota     20G                local
```

在 ZFS 文件系统中设置用户和组配额

可以使用 zfs userquota 或 zfs groupquota 命令分别设置用户配额或组配额：例如：

```
# zfs create students/compsci
# zfs set userquota@student1=10G students/compsci
# zfs create students/labstaff
# zfs set groupquota@labstaff=20GB students/labstaff
```

按以下方式显示当前用户配额或组配额：

```
# 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
```

可以通过查询以下属性来显示一般用户或组的磁盘空间使用情况：

```
# 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
```

要确定个别用户或组的磁盘空间使用情况，可以查询以下属性：

```
# 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
```

使用 `zfs get all` *dataset* 命令不会显示用户和组配额属性，而是显示所有其他文件系统属性的列表。

可以按以下方式删除用户配额或组配额：

```
# zfs set userquota@student1=none students/compsci
# zfs set groupquota@labstaff=none students/labstaff
```

ZFS 文件系统的用户和组配额提供以下功能：

- 在父文件系统上设置的用户配额或组配额不会被后代文件系统自动继承。
- 但是，基于具有用户或组配额的文件系统创建克隆或快照时，将应用用户或组配额。同样，使用 `zfs send` 命令（即使不带 `-R` 选项）创建流时，文件系统将具有用户或组配额。
- 非特权用户只能访问自己的磁盘空间使用情况。root 用户或被授予 `userused` 或 `groupused` 特权的用户可以访问所有人的用户或组磁盘空间记帐信息。
- 不能基于 ZFS 卷、早于文件系统版本 4 的文件系统或早于池版本 15 的池设置 `userquota` 和 `groupquota` 属性。

用户和组配额的实施可能会延迟几秒钟。这种延迟意味着，在系统发现已超出配额并拒绝其他写入操作（同时显示 `EDQUOT` 错误消息）之前，用户可能已超出其配额。

您可以使用传统 `quota` 命令查看 NFS 环境（例如，挂载了 ZFS 文件系统）中的用户配额。不带任何选项的 `quota` 命令仅显示是否超出用户配额的输出信息。例如：

```
# 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
```

如果重置用户配额，而且不再超出配额限制，则可以使用 `quota -v` 命令查看用户的配额。例如：

```
# 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           -      -      -      -           -
```

设置 ZFS 文件系统的预留空间

ZFS 预留空间是从池中分配的保证可供数据集使用的磁盘空间。因此，如果磁盘空间当前在池中不可用，则不能为数据集预留该空间。所有未占用的预留空间的总量不能超出池中未使用的磁盘空间量。通过使用 `zfs set` 和 `zfs get` 命令可以设置和显示 ZFS 预留空间。例如：

```
# zfs set reservation=5G tank/home/bill
# zfs get reservation tank/home/bill
NAME            PROPERTY     VALUE   SOURCE
tank/home/bill  reservation  5G      local
```

预留空间可能会影响 `zfs list` 命令的输出。例如：

```
# 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
```

请注意，`tank/home` 使用的磁盘空间为 5 GB，但 `tank/home` 及其后代引脚的总空间量远远小于 5 GB。已用空间反映了为 `tank/home/bill` 预留的空间。预留空间计入父文件系统的已用磁盘空间内，并将计入父文件系统的配额或预留空间，或同时计入这两者中。

```
# 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
```

只要池中有未预留的空间可用，并且数据集的当前使用率低于其配额，数据集便能使用比其预留空间更多的磁盘空间。数据集不能占用为其他数据集预留的磁盘空间。

预留空间无法累积。也就是说，第二次调用 `zfs set` 来设置预留空间时，不会将该数据集的预留空间添加到现有预留空间中，而是使用第二个预留空间替换第一个预留空间。例如：

```
# 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
```

可通过设置 refreservation 预留空间来保证用于数据集的磁盘空间，该空间不包括快照和克隆使用的磁盘空间。此预留空间计算在父数据集的使用空间内，并会针对父数据集的配额和预留空间进行计数。例如：

```
# 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
```

还可以对同一数据集设置预留空间，以保证数据集空间和快照空间。例如：

```
# 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
```

常规的预留空间计算在父级的使用空间内。

在上例中，zfs list 输出显示两个配额中的较小者（10 GB 与 20 GB 相比较小）。要查看两个配额的值，请使用 zfs get 命令。例如：

```
# zfs get reservation,refreserv profs/prof1
NAME         PROPERTY        VALUE        SOURCE
profs/prof1  reservation     20G          local
profs/prof1  refreservation  10G          local
```

如果设置了 refreservation，则仅当在此预留空间之外有足够的未预留池空间来容纳数据集中的当前引用字节数时，才允许使用快照。

加密 ZFS 文件系统

加密是对数据进行编码以实现保密性的过程，数据所有者需要使用密钥才能访问已编码的数据。使用 ZFS 加密的优点如下所述：

- ZFS 加密与 ZFS 命令集相集成。与其他 ZFS 操作一样，加密操作（例如密钥更改和重建密钥）是联机执行的。
- 您可以使用现有的存储池，只要对它们进行了升级。可以灵活地加密特定的文件系统。
- 在 CCM 和 GCM 操作模式下，使用密钥长度为 128、192 和 256 的 AES（Advanced Encryption Standard，高级加密标准）对数据进行加密。

- ZFS 加密使用 Oracle Solaris 加密框架，该框架自动允许它访问加密算法的任何可用硬件加速或优化的软件实现。
- 当前，您无法对 ZFS 根文件系统或其他 OS 组件（例如 /var 目录）进行加密，即使它是单独的文件系统也是如此。
- ZFS 加密可由后代文件系统继承。
- 如果向一般用户分配了 create、mount、keysource、checksum 以及 encryption 权限，则该用户可以创建加密的文件以及管理密钥操作。

可以在创建 ZFS 文件系统时设置加密策略，但是无法更改该策略。例如，在启用了加密属性的情况下创建了 tank/home/darren 文件系统。缺省的加密策略是提示您输入口令短语，口令短语的长度必须至少为 8 个字符。

```
# zfs create -o encryption=on tank/home/darren
Enter passphrase for 'tank/home/darren': xxxxxxx
Enter again: xxxxxxxx
```

确认文件系统已启用加密。例如：

```
# zfs get encryption tank/home/darren
NAME              PROPERTY    VALUE        SOURCE
tank/home/darren  encryption  on           local
```

当文件系统的加密值为 on 时，缺省的加密算法为 aes-128-ccm。

包装密钥用于加密实际的数据加密密钥。如上例所示，在创建加密的文件系统时，会将包装密钥从 zfs 命令传递到内核。包装密钥位于一个文件（采用 raw 或 hex 格式）中，或者派生自口令短语。

包装密钥的格式和位置是在 keysource 属性中指定的，如下所示：

```
keysource=format,location
```

- 格式为下列值之一：
  - raw – 原始的密钥字节
  - hex – 十六进制密钥字符串
  - passphrase – 用以生成密钥的字符串
- 位置为下列值之一：
  - prompt – 创建或挂载文件系统时提示您输入密钥或口令短语
  - file:///*filename* – 密钥文件或口令短语文件在文件系统中的位置
  - pkcs11 – 描述 PKCS#11 令牌中密钥或口令短语位置的 URI
  - https://*location* – 密钥文件或口令短语文件在安全服务器上的位置。不建议使用此方法以明文形式传输密钥信息。根据在 keysource 属性的格式部分中请求的内容，对 URL 执行 GET 可仅返回密钥值或口令短语。

    对 keysource 使用 https:// 定位符时，服务器提供的证书必须是 libcurl 和 OpenSSL 信任的证书。将自己的信任锚点证书或自签名证书添加到 /etc/

openssl/certs 下的证书库中。将 PEM 格式证书放在 /etc/certs/CA 目录下并运行以下命令：

```
# svcadm refresh ca-certificates
```

如果 keysource 格式为 *passphrase*，则包装密钥派生自口令短语。否则，keysource 属性值指向实际的包装密钥，为原始字节或十六进制格式。您可以指定口令短语存储在文件中，或者存储在提示输入的原始字节流中，这可能仅适合于编写脚本。

如果文件系统的 keysource 属性值标识 passphrase，则包装密钥派生自口令短语（使用 PKCS#5 PBKD2 和按文件系统随机生成的 salt）。这意味着，相同的口令短语在后代文件系统上使用时可生成不同的包装密钥。

文件系统的加密策略由后代文件系统继承，且不可删除。例如：

```
# 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
```

如果需要复制或迁移加密或未加密的 ZFS 文件系统，请注意以下要点：

- 当前，不能将以未加密形式发送的数据集流接收为加密的流，即使接收池的数据集已启用加密。
- 可以使用以下命令将未加密的数据迁移到已启用加密的池/文件系统：
  - cp -r
  - find | cpio
  - tar
  - rsync
- 可以将复制的加密文件系统流接收到加密文件系统中，且数据保持加密状态。有关更多信息，请参见例 5-4 “发送和接收加密的 ZFS 文件系统”。

更改加密 ZFS 文件系统的密钥

通过使用 zfs key -c 命令，可以更改加密文件系统的包装密钥。必须首先已通过以下方式装入了现有的包装密钥：在引导时、通过显式装入文件系统密钥 (zfs key -l) 或者通过挂载文件系统 (zfs mount *filesystem*)。例如：

```
# zfs key -c tank/home/darren
Enter new passphrase for 'tank/home/darren': xxxxxxxx
Enter again: xxxxxxxx
```

在以下示例中，更改了包装密钥，并更改了 keysource 属性值以指定包装密钥来自一个文件。

```
# zfs key -c -o keysource=raw,file:///media/stick/key tank/home/darren
```

通过使用 `zfs key -K` 命令，可以更改加密文件系统的数据加密密钥，但是新的加密密钥仅用于新写入的数据。此功能可用来满足针对数据加密密钥的时限的 NIST 800-57 准则。例如：

```
# zfs key -K tank/home/darren
```

在上面的示例中，数据加密密钥不可见，也不是由您直接管理的。此外，您需要具有 `keychange` 委托才能执行密钥更改操作。

有以下加密算法可用：

- `aes-128-ccm`、`aes-192-ccm`、`aes-256-ccm`
- `aes-128-gcm`、`aes-192-gcm`、`aes-256-gcm`

ZFS `keysource` 属性标识用来包装文件系统数据加密密钥的密钥的格式和位置。例如：

```
# zfs get keysource tank/home/darren
NAME              PROPERTY   VALUE              SOURCE
tank/home/darren  keysource  passphrase,prompt  local
```

ZFS `rekeydate` 属性标识上次执行 `zfs key -K` 操作的日期。例如：

```
# zfs get rekeydate tank/home/darren
NAME              PROPERTY   VALUE                  SOURCE
tank/home/darren  rekeydate  Wed Jul 25 16:54 2012  local
```

如果加密文件系统的 `creation` 和 `rekeydate` 属性具有相同的值，则表明从未通过 `zfs key -K` 操作为该文件系统重建密钥。

管理 ZFS 加密密钥

您可以采用不同方式管理 ZFS 加密密钥，具体取决于您的需求，您既可以在本地系统上进行管理，也可以远程管理（如果需要一个中央位置）。

- **本地** – 上述示例说明了包装密钥既可以是口令短语提示符，也可以是本地系统上文件中的存储的原始密钥。
- **远程** – 可以使用密钥集中管理系统（如 Oracle 密钥管理程序）或者使用支持对 http 或 https URI 的简单 GET 请求的 Web 服务远程存储密钥信息。通过使用 PKCS#11 令牌，Oracle Solaris 系统可以访问 Oracle 密钥管理程序的密钥信息。

有关管理 ZFS 加密密钥的更多信息，请参见： http://www.oracle.com/technetwork/articles/servers-storage-admin/manage-zfs-encryption-1715034.html

有关使用 Oracle 密钥管理程序管理密钥信息的信息，请参见：

http://docs.oracle.com/cd/E24472_02/

委托 ZFS 密钥操作权限

查看关于委托密钥操作的以下权限描述：

- 通过使用 `zfs key -l` 和 `zfs key -u` 命令装入或卸载文件系统密钥需要 `key` 权限。大多数情况下，还将需要 `mount` 权限。
- 通过使用 `zfs key -c` 和 `zfs key -K` 命令更改文件系统密钥需要 `keychange` 权限。

请考虑针对密钥使用（装入或卸载）和密钥更改授予单独的权限，这样您可以采用“两人”密钥操作模型。例如，确定哪些用户可以使用密钥，哪些用户可以更改这些密钥。或者，两种用户需要同时在场才能进行密钥更改。此模型还允许您构建密钥契约系统。

挂载加密的 ZFS 文件系统

在尝试挂载加密的 ZFS 文件系统时，请检查以下注意事项：

- 如果加密文件系统的密钥在引导时不可用，则不会自动挂载该文件系统。例如，加密策略设置为 `passphrase,prompt` 的文件系统在引导时不会挂载，因为引导过程不会中断以提示您输入口令短语。
- 如果要在引导时挂载加密策略设置为 `passphrase,prompt` 的文件系统，将需要使用 `zfs mount` 命令显式挂载它并指定口令短语，或者使用 `zfs key -l` 命令在引导系统之后提示您输入密钥。

  例如：

  ```
  # zfs mount -a
  Enter passphrase for 'tank/home/darren': xxxxxxxx
  Enter passphrase for 'tank/home/ws': xxxxxxxx
  Enter passphrase for 'tank/home/mark': xxxxxxxx
  ```

- 如果加密文件系统的 `keysource` 属性指向另一个文件系统中的文件，则文件系统的挂载顺序可能影响在引导时是否挂载加密的文件系统，尤其是文件在可移除介质上时。

升级加密的 ZFS 文件系统

将 Solaris 11 系统升级到 Solaris 11.1 之前，请确保加密的文件系统已挂载。挂载加密的文件系统并提供口令短语（如果提示）。

```
# 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
```

然后，升级加密的文件系统。

```
# zfs upgrade -a
```

如果尝试升级的加密 ZFS 文件系统未挂载，将显示类似于以下内容的消息：

```
# zfs upgrade -a
cannot set property for 'pond/amy': key not present
```

此外，`zpool status` 输出可能显示损坏的数据。

```
# zpool status -v pond
.
.
.
pond/amy:<0x1>
pond/rory:<0x1>
```

如果出现上述错误，请按上文所述重新挂载加密的文件系统。然后，清理并清除池错误。

```
# zpool scrub pond
# zpool clear pond
```

有关升级文件系统的更多信息，请参见“升级 ZFS 文件系统” [166]。

ZFS 压缩、重复数据删除和加密属性之间的交互

在使用 ZFS 压缩、重复数据删除和加密属性时，请检查以下注意事项：

- 在写入文件时，会对数据进行压缩和加密，并验证校验和。然后，会尽可能对数据进行重复删除。
- 在读取文件时，会验证校验和，并对数据进行解密。然后，根据需要对数据进行解压缩。
- 如果在克隆的加密文件系统上启用了 `dedup` 属性，且未对克隆使用 `zfs key -K` 或 `zfs clone -K` 命令，则会尽可能对来自所有克隆的数据进行重复数据删除。

加密 ZFS 文件系统的示例

例 5-1 通过使用原始密钥加密 ZFS 文件系统

在以下示例中，通过使用 `pktool` 命令生成了一个 `aes-256-ccm` 加密密钥，并将其写入了文件 `/cindykey.file` 中

```
# pktool genkey keystore=file outkey=/cindykey.file keytype=aes keylen=256
```

然后，在创建 `tank/home/cindy` 文件系统时指定了 `/cindykey.file`。

```
# zfs create -o encryption=aes-256-ccm -o keysource=raw,file:///cindykey.file
tank/home/cindy
```

例 5-2 使用不同的加密算法加密 ZFS 文件系统

您可以创建一个 ZFS 存储池，并让该存储池中的所有文件系统都继承加密算法。在此示例中，创建了 users 池，创建了 users/home 文件系统，并通过使用口令短语对该文件系统进行了加密。缺省的加密算法是 aes-128-ccm。

然后，创建了 users/home/mark 文件系统，并通过使用 aes-256-ccm 加密算法对其进行了加密。

```
# 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 克隆加密的 ZFS 文件系统

如果克隆文件系统从与其源快照相同的文件系统继承了 keysource 属性，则新的 keysource 是不必要的，并且如果 keysource=passphrase,prompt，系统也不会提示您输入新的口令短语。将为克隆使用相同的 keysource。例如：

缺省情况下，在克隆加密文件系统的后代时，系统不会提示您输入密钥。

```
# 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
```

如果要为克隆文件系统创建新密钥，请使用 zfs clone -K 命令。

如果是克隆加密的文件系统而不是加密的后代文件系统，则系统将提示您提供新密钥。例如：

```
# 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 发送和接收加密的 ZFS 文件系统

在下面的示例中，根据加密的 /tank/home/darren 文件系统创建了 tank/home/darren@snap1 快照。然后，将快照发送到 bpool/snaps，由于已启用了加密属性，因此会加密生成的接收数据。但是，在发送过程中，tank/home/darren@snap1 流未加密。

```
# 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
```

在这种情况下，将为接收的加密文件系统自动生成新密钥。

迁移 ZFS 文件系统

可以使用影子迁移功能迁移文件系统，如下所述：

- 本地或远程 ZFS 文件系统到目标 ZFS 文件系统
- 本地或远程 UFS 文件系统到目标 ZFS 文件系统

影子迁移是一个推送要迁移的数据的过程：

- 创建一个空的 ZFS 文件系统。
- 在要用作目标（或影子）文件系统的空 ZFS 文件系统上设置 shadow 属性，以指向要迁移的文件系统。
- 数据从要迁移的文件系统复制到影子文件系统。

可以使用 shadow 属性 URI 按以下两种方式标识要迁移的文件系统：

- shadow=file:///*path* – 使用此语法迁移本地文件系统
- shadow=nfs://*host/path* – 使用此语法迁移 NFS 文件系统

在迁移文件系统时，请检查以下注意事项：

- 要迁移的文件系统必须设置为只读的。如果文件系统未设置为只读的，则可能不会迁移正在进行的更改。
- 目标文件系统必须完全为空。

- 如果在迁移过程中重新引导了系统，则在引导系统后迁移将继续执行。
- 在迁移完整个内容之前，会阻止访问未完全迁移的目录内容或未完全迁移的文件内容。
- 如果希望在 NFS 迁移过程中将 UID、GID 和 ACL 信息迁移到影子文件系统，请确保名称服务信息在本地系统和远程系统之间是可访问的。在通过 NFS 来完成大规模的数据迁移之前，可以考虑复制一部分要迁移的文件系统数据进行测试迁移，以便查看是否正确迁移了所有 ACL 信息。
- 通过 NFS 迁移文件系统数据，具体取决于您的网络带宽。请耐心等待。
- 可以使用 `shadowstat` 命令监视文件系统迁移，它提供以下数据：
  - `BYTES XFRD` 列标识已传输到影子文件系统的字节数。
  - `BYTES LEFT` 列不断变化，直到迁移几乎完成时为止。ZFS 在迁移开始时不标识需要迁移的数据量，因为此过程所用的时间可能太长。
  - 可考虑使用 `BYTES XFRD` 和 `ELAPSED TIME` 信息来估算迁移过程所用时间。

▼ 如何将文件系统迁移到 ZFS 文件系统

1. **如果要从远程 NFS 服务器迁移数据，请确认名称服务信息在两个系统上都是可访问的。**
   对于使用 NFS 的大型迁移，可以考虑执行数据子集的测试迁移，以确保 UID、GUID 和 ACL 信息正确地迁移。

2. **如有必要，请在要从中迁移数据的系统上安装影子迁移软件包，并启用 `shadowd` 服务以帮助执行迁移过程。**

   ```
   # pkg install shadow-migration

   # svcadm enable shadowd
   ```

   如果不启用 `shadowd` 进程，则在迁移过程完成时必须将 `shadow` 属性重置为 `none`。

3. **将要迁移的本地或远程文件系统设置为只读的。**
   如果要迁移本地 ZFS 文件系统，请将它设置为只读的。例如：

   ```
   # zfs set readonly=on tank/home/data
   ```

   如果要迁移远程文件系统，请将其共享为只读的。例如：

   ```
   # share -F nfs -o ro /export/home/ufsdata
   # share
   -               /export/home/ufsdata   ro   ""
   ```

4. **创建一个新的 ZFS 文件系统，并将 shadow 属性设置为要迁移的文件系统。**
   例如，如果要将本地 ZFS 文件系统 `rpool/old` 迁移到新的 ZFS 文件系统 `users/home/shadow`，请在创建 `users/home/shadow` 文件系统时将 `shadow` 属性设置为 `rpool/old`。

```
# zfs create -o shadow=file:///rpool/old users/home/shadow
```

例如，要从远程服务器迁移 /export/home/ufsdata，请在创建 ZFS 文件系统时设置 shadow 属性。

```
# zfs create -o shadow=nfs://neo/export/home/ufsdata users/home/shadow2
```

5. 检查迁移进度。
例如：

```
# 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
```

在迁移完成时，shadow 属性设置为 none。

```
#zfs get -r shadow users/home/shadow*
NAME                PROPERTY  VALUE   SOURCE
users/home/shadow   shadow    none    -
users/home/shadow2  shadow    none    -
```

ZFS 文件系统迁移故障排除

在解决 ZFS 迁移问题时，请检查以下几点：

- 如果要迁移的文件系统未设置为只读的，则不会迁移所有数据。
- 当设置了 shadow 属性时，如果目标文件系统不为空，则数据迁移将不会开始。
- 如果当迁移正在进行时在要迁移的文件系统中添加或删除了数据，则可能不会迁移这些更改。
- 如果在迁移正在进行时尝试更改影子文件系统的挂载，则将看到以下消息：

```
# zfs set mountpoint=/users/home/data users/home/shadow3
cannot unmount '/users/home/shadow3': Device busy
```

升级 ZFS 文件系统

如果您具有先前的 Solaris 发行版的 ZFS 文件系统，可以使用 zfs upgrade 命令升级该文件系统，以利用当前发行版中的文件系统功能。此外，此命令在文件系统运行旧版本时会通知您。

例如，此文件系统当前运行的版本是 5。

```
# zfs upgrade
This system is currently running ZFS filesystem version 5.

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

使用此命令可以确定每个文件系统版本的可用功能。

```
# 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.
```

有关升级加密的文件系统的信息，请参见“升级加密的 ZFS 文件系统” [161]。

♦ ♦ ♦ 第 6 章

使用 Oracle Solaris ZFS 快照和克隆

本章介绍如何创建和管理 Oracle Solaris ZFS 快照和克隆。此外还提供了有关保存快照的信息。

本章包含以下各节：

ZFS 快照概述

快照是文件系统或卷的只读副本。快照几乎可以即时创建，而且最初不占用池中的其他磁盘空间。但是，活动数据集中的数据更改时，快照仍将继续引用旧数据，这会占用磁盘空间，从而阻止释放磁盘空间。

ZFS 快照具有以下特征：

- 在系统重新引导后会继续存在。
- 理论最大快照数是 2^{64}。
- 快照不使用单独的后备存储。快照直接占用存储池（从中创建这些快照的文件系统或卷所在的存储池）中的磁盘空间。
- 递归快照可作为一个原子操作快速创建。要么一起创建快照（一次创建所有快照），要么不创建任何快照。原子快照操作的优点是始终在一个一致的时间捕获快照数据，即使跨后代文件系统也是如此。

无法直接访问卷的快照，但是可以对它们执行克隆、备份、回滚等操作。有关备份 ZFS 快照的信息，请参见“发送和接收 ZFS 数据” [177]。

创建和销毁 ZFS 快照

快照是使用 `zfs snapshot` 或 `zfs snap` 命令创建的，该命令采用要创建的快照的名称作为其唯一参数。快照名称按如下方式指定：

filesystem@snapname
volume@snapname

快照名称必须满足“ZFS 组件命名要求” [17]中所述的命名要求。

在以下示例中，创建了一个 `tank/home/cindy` 的快照，名为 `friday`。

```
# zfs snapshot tank/home/cindy@friday
```

通过使用 `-r` 选项可为所有后代文件系统创建快照。例如：

```
# 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  -
```

快照没有可修改的属性。也不能将数据集属性应用于快照。例如：

```
# zfs set compression=on tank/home/cindy@friday
cannot set property for 'tank/home/cindy@friday':
this property can not be modified for snapshots
```

使用 `zfs destroy` 命令可以销毁快照。例如：

```
# zfs destroy tank/home/cindy@friday
```

如果数据集存在快照，则不能销毁该数据集。例如：

```
# 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
```

此外，如果已从快照创建克隆，则必须先销毁克隆，才能销毁快照。

有关 destroy 子命令的更多信息，请参见“销毁 ZFS 文件系统” [115]。

保持 ZFS 快照

不同的自动快照或数据保留策略可能意味着会无意中破坏之前的快照。如果删除的快照是正在进行的 zfs 发送和接收操作的一部分，则该操作可能会失败。要避免这种情况，请考虑为快照设置保持。

保持快照可以防止它被销毁。此外，该功能支持在删除一个带有克隆的快照时暂挂最后一个克隆的删除操作（使用 zfs destroy - d 命令）。每个快照都有一个关联的用户引用计数，其初始值为 0。在一个快照上设置一个保持标志时，此计数递增 1；释放一个保持标志时，此计数递减 1。

在先前的 Oracle Solaris 发行版中，只有在快照无克隆时，才能使用 zfs destroy 命令销毁快照。在此 Oracle Solaris 发行版中，快照的用户引用计数也必须为零。

可以保持一个快照或一组快照。例如，以下语法在 tank/home/cindy/snap@1 上设置一个保持标志 keep。

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

可以使用 -r 选项递归保持所有后代文件系统的快照。例如：

```
# zfs snapshot -r tank/home@now
# zfs hold -r keep tank/home@now
```

此语法向给定的快照或快照集添加一个引用 keep。每个快照都有其自己的标志名称空间，保持标志在该空间内必须是唯一的。如果一个快照上存在一个保持，尝试使用 zfs destroy 命令销毁受保持的快照将失败。例如：

```
# zfs destroy tank/home/cindy@snap1
cannot destroy 'tank/home/cindy@snap1': dataset is busy
```

要销毁保持的快照，须使用 -d 选项。例如：

```
# zfs destroy -d tank/home/cindy@snap1
```

使用 zfs holds 命令显示受保持的快照列表。例如：

```
# 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
```

可以使用 `zfs release` 命令释放对一个快照或一组快照的保持。例如：

```
# zfs release -r keep tank/home@now
```

释放快照后，可以使用 `zfs destroy` 命令销毁快照。例如：

```
# zfs destroy -r tank/home@now
```

有两个新属性用来表示快照保持信息：

- `defer_destroy` 属性在下述情况下为 `on`：已使用 `zfs destroy -d` 命令将快照标记为延期销毁。否则，此属性为 `off`。
- `userrefs` 属性设置为此快照上的保持数，也称为用户引用计数。

重命名 ZFS 快照

可以重命名快照，但是必须在从中创建它们的池和数据集中对它们进行重命名。例如：

```
# zfs rename tank/home/cindy@snap1 tank/home/cindy@today
```

此外，以下快捷方式语法等效于以上的语法：

```
# zfs rename tank/home/cindy@snap1 today
```

不支持以下快照 `rename` 操作，因为目标池和文件系统名称与从中创建快照的池和文件系统不同：

```
# zfs rename tank/home/cindy@today pool/home/cindy@saturday
cannot rename to 'pool/home/cindy@today': snapshots must be part of same
dataset
```

可以使用 `zfs rename -r` 命令以递归方式重命名快照。例如：

```
# 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  -
```

显示和访问 ZFS 快照

缺省情况下，zfs list 输出中不再显示快照。您必须使用 zfs list -t snapshot 命令显示快照信息。或者，启用 listsnapshots 池属性。例如：

```
# 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
```

在文件系统的根的 .zfs/snapshot 目录中，可以访问文件系统的快照。例如，如果 tank/home/cindy 挂载在 /home/cindy 下，则可以在 /home/cindy/.zfs/snapshot/thursday 目录中访问 tank/home/cindy@thursday 快照数据。

```
# ls /tank/home/cindy/.zfs/snapshot
thursday   tuesday    wednesday
```

可以列出快照，如下所示：

```
# 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  -
```

可以列出为特定文件系统创建的快照，如下所示：

```
# 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 快照的磁盘空间记帐

创建快照时，最初在快照和文件系统之间共享其磁盘空间，还可能与以前的快照共享其空间。在文件系统发生更改时，以前共享的磁盘空间将变为该快照专用的空间，因此会将该空间算入快照的 used 属性。此外，删除快照可增加其他快照专用（使用）的磁盘空间量。

创建快照时，快照的 referenced 空间属性值与文件系统的相同。

可以找到有关 used 属性值如何使用的附加信息。新的只读文件系统属性说明克隆、文件系统和卷的磁盘空间使用情况。例如：

```
$ 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
```

有关这些属性的说明，请参见表 5-1 “ZFS 本机属性说明”。

回滚 ZFS 快照

可以使用 zfs rollback 命令放弃自特定快照创建以来对文件系统所做的全部更改。文件系统恢复到创建快照时的状态。缺省情况下，该命令无法回滚到除最新快照以外的快照。

要回滚到早期快照，必须销毁所有的中间快照。可以通过指定 -r 选项销毁早期的快照。

如果存在任何中间快照的克隆，则还必须指定 -R 选项以销毁克隆。

注 - 如果要回滚的文件系统当前为挂载状态，则会取消挂载并重新挂载。如果无法取消挂载该文件系统，则回滚将失败。-f 选项可强制取消挂载文件系统（如有必要）。

在以下示例中，将 tank/home/cindy 文件系统回滚到 tuesday 快照：

```
# 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
```

在本示例中，因为已回滚到以前的 tuesday 快照，所以销毁了 wednesday 和 thursday 快照。

```
# zfs list -r -t snapshot -o name,creation tank/home/cindy
NAME                      CREATION
tank/home/cindy@tuesday  Fri Aug  3 15:18 2012
```

确定 ZFS 快照的差异 (zfs diff)

可以使用 zfs diff 命令确定 ZFS 快照的差异。

例如，假定创建了以下两个快照：

```
$ ls /tank/home/tim
fileA
$ zfs snapshot tank/home/tim@snap1
$ ls /tank/home/tim
fileA  fileB
$ zfs snapshot tank/home/tim@snap2
```

例如，要确定两个快照之间的差异，请使用类似以下的语法：

```
$ zfs diff tank/home/tim@snap1 tank/home/tim@snap2
M       /tank/home/tim/
+       /tank/home/tim/fileB
```

在输出中，M 表示该目录已经过修改。+ 表示 fileB 存在于较新的快照中。

以下输出中的 R 表示快照中的某个文件已经重命名。

```
$ 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
```

下表概括了 zfs diff 命令确定的文件或目录更改。

文件或目录更改	标识符
文件或目录已被修改，或文件或目录链接已更改	M
文件或目录出现在较旧的快照中，但未出现在较新的快照中	–
文件或目录出现在较新的快照中，但未出现在较旧的快照中	+
文件或目录已重命名	R

有关更多信息，请参见 zfs(1M)。

ZFS 克隆概述

克隆是可写入的卷或文件系统，其初始内容与从中创建它的数据集的内容相同。与快照一样，创建克隆几乎是即时的，而且最初不占用其他磁盘空间。此外，还可以创建克隆的快照。

克隆只能从快照创建。克隆快照时，会在克隆和快照之间建立隐式相关性。即使克隆是在文件系统分层结构中的其他位置创建的，但只要克隆存在，就无法销毁原始快照。`origin` 属性显示此相关项，而 `zfs destroy` 命令会列出任何此类相关项（如果存在）。

克隆不继承从其中创建它的数据集的属性。使用 `zfs get` 和 `zfs set` 命令，可以查看和更改克隆数据集的属性。有关设置 ZFS 数据集属性的更多信息，请参见“设置 ZFS 属性” [134]。

由于克隆最初与原始快照共享其所有磁盘空间，因此其 `used` 属性值最初为零。随着不断对克隆进行更改，它使用的磁盘空间将越来越多。原始快照的 `used` 属性不包括克隆所占用的磁盘空间。

创建 ZFS 克隆

要创建克隆，请使用 `zfs clone` 命令，指定从中创建克隆的快照以及新文件系统或卷的名称。新文件系统或卷可以位于 ZFS 分层结构中的任意位置。新数据集与从其中创建克隆的快照属同一类型（例如文件系统或卷）。不能在原始文件系统快照所在池以外的池中创建该文件系统的克隆。

在以下示例中，将创建一个名为 `tank/home/matt/bug123` 的新克隆，其初始内容与快照 `tank/ws/gate@yesterday` 的内容相同：

```
# zfs snapshot tank/ws/gate@yesterday
# zfs clone tank/ws/gate@yesterday tank/home/matt/bug123
```

在以下示例中，从 `projects/newproject@today` 快照为临时用户创建克隆工作区 `projects/teamA/tempuser`。然后，在克隆工作区上设置属性。

```
# 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
```

销毁 ZFS 克隆

使用 `zfs destroy` 命令可以销毁 ZFS 克隆。例如：

```
# zfs destroy tank/home/matt/bug123
```

必须先销毁克隆，才能销毁父快照。

使用 ZFS 克隆替换 ZFS 文件系统

借助 `zfs promote` 命令可以用活动的 ZFS 文件系统的克隆来替换该文件系统。利用此功能可以克隆并替换文件系统，使源文件系统变为指定文件系统的克隆。此外，通过此功能还可以销毁最初创建克隆所基于的文件系统。如果没有克隆提升 (clone promotion) 功能，就无法销毁活动克隆的源文件系统。有关销毁克隆的更多信息，请参见“销毁 ZFS 克隆” [176]。

在以下示例中，对 `tank/test/productA` 文件系统进行了克隆，然后克隆文件系统 `tank/test/productAbeta` 成为原始 `tank/test/productA` 文件系统。

```
# 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  -
```

在此 `zfs list` 输出中，注意源 `productA` 文件系统的磁盘空间记帐信息已被 `productAbeta` 文件系统取代。

可以通过重命名文件系统完成克隆替换过程。例如：

```
# zfs rename tank/test/productA tank/test/productAlegacy
# zfs rename tank/test/productAbeta tank/test/productA
# zfs list -r tank/test
```

或者，也可以删除传统的文件系统。例如：

```
# zfs destroy tank/test/productAlegacy
```

发送和接收 ZFS 数据

`zfs send` 命令创建写入标准输出的快照流表示。缺省情况下，生成完整的流。可以将输出重定向到文件或其他系统。`zfs receive` 命令创建其内容在标准输入提供的流中指定

的快照。如果接收了完整的流，那么同时会创建一个新文件系统。可通过这些命令来发送 ZFS 快照数据并接收 ZFS 快照数据和文件系统。请参见下一节中的示例。

以下是用于保存 ZFS 数据的备份解决方案：

- **企业备份产品** – 如果需要以下功能，则应考虑企业备份解决方案：
  - 按文件恢复
  - 备份介质验证
  - 介质管理
- **文件系统快照和回滚快照** – 如果要轻松创建文件系统的副本并恢复到以前的文件系统版本（如有必要），请使用 `zfs snapshot` 和 `zfs rollback` 命令。例如，要从文件系统的早期版本恢复一个或多个文件，可以使用此解决方案。

  有关创建快照和回滚到快照的更多信息，请参见“ZFS 快照概述” [169]。
- **保存快照** – 使用 `zfs send` 和 `zfs receive` 命令可发送和接收 ZFS 快照。可以保存快照之间的增量更改，但不能逐个恢复文件。必须恢复整个文件系统快照。这些命令不提供用于保存 ZFS 数据的完整备份解决方案。
- **远程复制** – 要将文件系统从一个系统复制到另一个系统，请使用 `zfs send` 和 `zfs receive` 命令。此过程与可能跨 WAN 镜像设备的传统卷管理产品有所不同。不需要特殊的配置或硬件。复制 ZFS 文件系统的优点是，可以在其他系统的存储池上重新创建文件系统，并为新创建的池指定不同的配置级别（如 RAID-Z），但是新创建的池使用相同的文件系统数据。
- **归档实用程序** – 使用归档实用程序（如 `tar`、`cpio` 和 `pax`）或第三方备份产品保存 ZFS 数据。目前，`tar` 和 `cpio` 均能正确转换 NFSv4 样式的 ACL，但是 `pax` 还不行。

使用其他备份产品保存 ZFS 数据

除 `zfs send` 和 `zfs receive` 命令外，还可以使用归档实用程序（如 `tar` 和 `cpio` 命令）保存 ZFS 文件。这些实用程序可以保存和恢复 ZFS 文件属性和 ACL。请选中 `tar` 和 `cpio` 命令的适当选项。

有关 ZFS 和第三方备份产品的问题的最新信息，请参见 Oracle Solaris 11.2 发行说明。

识别 ZFS 快照流

通过使用 `zfs send` 命令，可以将 ZFS 文件系统或卷的快照转换为快照流。然后，可以使用快照流重新创建 ZFS 文件系统或卷（通过 `zfs receive` 命令）。

根据用于创建快照流的 `zfs send` 选项，将生成不同类型的流格式。

- 完整流 – 包含从创建数据集时开始到指定的快照为止的所有数据集内容。

  `zfs send` 命令生成的缺省流是完整流。它包含一个文件系统或卷，直到并包括指定的快照。流不会包含在命令行上指定的快照之外的快照。
- 增量流 – 包含一个快照与另一个快照之间的差异。

流数据包是包含一个或多个完整流或增量流的流类型。存在以下三种类型的流数据包：

- 复制流数据包 – 包含指定的数据集及其后代。它包括所有的中间快照。如果克隆数据集的源不是在命令行上指定的快照的后代，则该源数据集将不包括在流数据包中。要接收流，源数据集必须存在于目标存储池中。

  请考虑以下列表中的数据集及其源。假定它们是按如下所示顺序创建的。

```
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           -
```

  使用以下语法创建的复制流数据包：

```
# zfs send -R pool/b@send ....
```

  包含以下完整流和增量流：

```
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
```

在上面的输出中，pool/a/1@clone 快照未包括在复制流数据包中。因此，只能在已具有 pool/a/1@clone 快照的池中接收此复制流数据包。

- 递归流数据包 – 包含指定的数据集及其后代。与复制流数据包不同，除非中间快照是流中包括的克隆数据集的源，否则将不包括它们。缺省情况下，如果数据集的源不是在命令行上指定的快照的后代，则行为类似于复制流。但是，下面所述的自包含递归流在创建过程中不存在外部依赖项。

  使用以下语法创建的递归流数据包：

  ```
  # zfs send -r pool/b@send ...
  ```

  包含以下完整流和增量流：

  ```
  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
  ```

  在上面的输出中，pool/a/1@clone 快照未包括在递归流数据包中。因此，只能在已具有 pool/a/1@clone 快照的池中接收此递归流数据包。此行为与上面所述的复制流数据包情况类似。
- 自包含递归流数据包 – 不依赖于流数据包中未包括的任何数据集。此递归流数据包是使用以下语法创建的：

  ```
  # zfs send -rc pool/b@send ...
  ```

  包含以下完整流和增量流：

  ```
  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
  ```

  可以看到，该自包含递归流具有 pool/b/1@clone2 快照的完整流，这使得接收没有外部相关项的 pool/b/1 快照成为可能。

发送 ZFS 快照

可以使用 zfs send 命令来发送某个快照流的副本，并在同一系统的另一个池中或用于存储备份数据的不同系统上的另一个池中接收快照流。例如，要将不同池的快照流发送到同一系统，请使用类似如下的语法：

```
# zfs send tank/dana@snap1 | zfs recv spool/ds01
```

可以将 `zfs recv` 用作 `zfs receive` 命令的别名。

如果要将快照流发送到不同的系统，请通过 `ssh` 命令传输 `zfs send` 输出。例如：

```
sys1# zfs send tank/dana@snap1 | ssh sys2 zfs recv newtank/dana
```

发送完整的流时，目标文件系统必须不能存在。

使用 `zfs send -i` 选项可以发送增量数据。例如：

```
sys1# zfs send -i tank/dana@snap1 tank/dana@snap2 | ssh sys2 zfs recv newtank/dana
```

第一个参数 (`snap1`) 是较早的快照，第二个参数 (`snap2`) 是较晚的快照。这种情况下，`newtank/dana` 文件系统必须已经存在，增量接收才能成功。

注 - 访问原始接收文件系统中的文件信息可能导致增量快照接收操作失败，并显示类似于以下的消息：

```
cannot receive incremental stream of tank/dana@snap2 into newtank/dana:
most recent snapshot of tank/dana@snap2 does not match incremental source
```

如果需要访问原始接收文件系统中的文件信息，同时还需要将增量快照接收到该接收文件系统中，请考虑将 `atime` 属性设置为 off。

可将增量 *snap1* 源指定为快照名称的最后一个组成部分。此快捷方式意味着只需在 `@` 符号后指定 *snap1* 的名称，假定它与 *snap2* 都来自同一文件系统。例如：

```
sys1# zfs send -i snap1 tank/dana@snap2 | ssh sys2 zfs recv newtank/dana
```

这一快捷方式语法等效于上例中的增量语法。

尝试从其他文件系统 `snapshot1` 生成增量流时，将显示以下消息：

```
cannot send 'pool/fs@name': not an earlier snapshot from the same fs
```

如果需要存储许多副本，可以考虑使用 `gzip` 命令压缩 ZFS 快照流表示。例如：

```
# zfs send pool/fs@snap | gzip > backupfile.gz
```

接收 ZFS 快照

接收文件系统快照时，请注意以下几点：

- 将接收快照和文件系统。
- 将取消挂载文件系统和所有后代文件系统。

- 文件系统在接收期间不可访问。
- 要接收的原始文件系统在传输期间必须不存在。
- 如果文件系统名称已经存在，可以使用 `zfs rename` 命令重命名文件系统。

例如：

```
# 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
```

如果对目标文件系统进行更改并且要再次以增量方式发送快照，则必须先回滚接收文件系统。

请参考以下示例。首先更改文件系统，如下所示：

```
sys2# rm newtank/dana/file.1
```

然后以增量方式发送 `tank/dana@snap3`。但是，要接收新的增量快照，首先必须回滚接收文件系统。或者，使用 `-F` 选项可以取消回滚步骤。例如：

```
sys1# zfs send -i tank/dana@snap2 tank/dana@snap3 | ssh sys2 zfs recv -F newtank/dana
```

接收增量快照时，目标文件系统必须已存在。

如果对文件系统进行更改，但不回滚接收文件系统以接收新的增量快照，或者不使用 `-F` 选项，则会显示类似于以下内容的消息：

```
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
```

在 `-F` 选项成功之前，会执行以下检查：

- 如果最新快照与增量源不匹配，则回滚和接收都无法完成，并且会返回一条错误消息。
- 如果意外地提供了与 `zfs receive` 命令所指定的增量源不匹配的其他文件系统名称，则回滚和接收都无法完成，并且会返回以下错误消息。

  ```
  cannot send 'pool/fs@name': not an earlier snapshot from the same fs
  ```

监视 ZFS 发送流的进度

以下预运行模式的 `zfs snapshot` 语法可提供 ZFS 快照流的估计大小。您可以使用大小估计值度量发送流本身的进度。

使用以下预运行语法估计快照流的大小，而不发送快照流。

```
# zfs snapshot -r tank/source@snap1
```

```
# zfs send -rnv tank/source@snap1
sending full stream to tank/source@snap1
sending full stream to tank/source/data@snap1
estimated stream size: 10.0G
```

通过在 zfs send 和 zfs receive 命令之间插入 pv 命令，您可以监视发送流的进度。在以下示例中，第一个命令估计流的大小。随后通过第二个命令发送快照同时进行监视。

```
# zfs send tank/source@snap1 | pv | zfs recv pond/target
882MB 0:00:10 [95.4MB/s] [       <=>   ]
```

在快照流接收完毕后，进度监视程序将标识接收的总大小。例如：

```
# zfs send tank/source@snap1 | pv |zfs recv pond/target
10GB 0:01:55 [88.5MG/s] [       <=>   ]
```

向 ZFS 快照流应用不同的属性值

您可以发送带有特定文件系统属性值的 ZFS 快照流，但在接收快照流时，可以指定不同的本地属性值。或者，您可以指定收到快照流时使用原始属性值以重新创建原始文件系统。此外，收到快照流时，还可以禁用文件系统属性。

- 使用 zfs inherit -S 将本地属性值恢复为接收值（如有）。如果属性没有接收值，则 zfs inherit - S 命令的行为与不带 -S 选项的 zfs inherit 命令相同。如果属性有接收值，则 zfs inherit 命令会用继承的值覆盖接收的值，直到发出 zfs inherit -S 命令将其恢复为接收的值。
- 可以使用 zfs get -o 以包括新的非缺省栏 RECEIVED。或者，可以使用 zfs get -o all 命令以包括所有栏，其中包括 RECEIVED。
- 您可以使用 zfs send -p 选项以包括发送流中的属性，而无需使用 -R 选项。
- 可以使用 zfs receive -e 选项来利用所发送的快照名的最后一个元素确定新的快照名。以下示例将 poola/bee/cee@1 快照发送给 poold/eee 文件系统，并仅利用快照名的最后一个元素 (cee@1) 创建接收的文件系统和快照。

```
# 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  -
```

在某些情况下，发送流中的文件系统属性可能不适用于接收方文件系统，或者本地文件系统属性（如 mountpoint 属性值）可能会干扰恢复。

例如，tank/data 文件系统禁用了 compression 属性。tank/data 文件系统的一个快照在发送到备份池时带有属性（-p 选项），在接收该快照时启用了 compression 属性。

```
# 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
```

在该示例中，bpool 池接收快照时启用了 compression 属性。因此 bpool/data 的 compression 值为 on。

如果将此快照流发送到新池 restorepool 以用于恢复，您可能要保留所有原始的快照属性。在这种情况下，可使用 zfs send -b 命令恢复原始的快照属性。例如：

```
# 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
```

在该示例中，"compression" 的值是 off，代表来自原始 tank/data 文件系统的快照的 "compression" 值。

如果快照流中有本地文件系统属性值，而您希望在接收快照流时禁用该属性，可使用 zfs receive -x 命令。例如，以下命令发送一个起始目录文件系统的递归快照流，并将所有文件系统属性保留到备份池，但没有配额属性值：

```
# 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     -      -
```

如果未使用 -x 选项接收该递归快照，将在接收方文件系统设置配额属性。

```
# 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     -      -
```

发送和接收复杂的 ZFS 快照流

本节介绍如何使用 `zfs send -I` 和 `-R` 选项来发送和接收更复杂的快照流。

发送和接收复杂的 ZFS 快照流时，请牢记以下要点：

- 使用 `zfs send -I` 选项可将所有增量流从一个快照发送到某个累积快照。或者，使用此选项可从源快照发送增量流，以创建一个克隆。原始快照必须已存在于接收方之上才能接受增量流。
- 使用 `zfs send -R` 选项可发送所有后代文件系统的复制流。接收复制流时，所有属性、快照、后代文件系统和克隆都将被保留。
- 在没有 `-c` 选项的情况下使用 `zfs send -r` 选项时以及使用 `zfs send -R` 选项时，流数据包在某些情况下将忽略克隆的源。有关更多信息，请参见“识别 ZFS 快照流” [179]。
- 使用这两个选项发送增量复制流。
  - 对属性所做的更改得到保留，就像快照和文件系统的 `rename` 和 `destroy` 操作得到保留一样。
  - 如果在接收复制流时未指定 `zfs recv -F`，则将忽略数据集 `destroy` 操作。本例中的 `zfs recv -F` 语法还将保留其“如有必要则回滚”的含义。
  - 与其他（非 `zfs send -R`）`-i` 或 `-I` 情况一样，如果使用 `-I`，则将发送 `snapA` 与 `snapD` 之间的所有快照。如果使用 `-i`，则只发送 `snapD`（对于所有后代）。
- 要接收任何这些新类型的 `zfs send` 流，接收系统必须正在运行能够发送这些流的软件版本。流版本将递增。

  但是，您可以使用较新的软件版本来访问较旧池版本中的流。例如，您可以将使用较新的选项创建的流发送到版本 3 池，并可从版本 3 池中接收这些流。但是，要接收使用较新的选项发送的流，必须运行最新软件。

例 6-1 发送和接收复杂的 ZFS 快照流

可以使用 `zfs send -I` 选项将一组增量快照合并为一个快照。例如：

```
# zfs send -I pool/fs@snapA pool/fs@snapD > /snaps/fs@all-I
```

然后，您可以删除 `snapB`、`snapC` 和 `snapD`。

```
# zfs destroy pool/fs@snapB
# zfs destroy pool/fs@snapC
# zfs destroy pool/fs@snapD
```

要接收组合快照，可以使用以下命令。

```
# 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  -
```

您还可以使用 `zfs send -I` 命令来合并快照和克隆快照，以创建一个合并数据集。例如：

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

可以使用 `zfs send -R` 命令将 ZFS 文件系统和所有后代文件系统复制到一个已命名的快照中。接收此流时，所有属性、快照、后代文件系统和克隆都将被保留。

在以下示例中，将创建用户文件系统的快照。为所有用户快照创建一个复制流。然后，原始文件系统和快照将被销毁并恢复。

```
# 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  -
```

以下示例使用 `zfs send -R` 命令来复制 `users` 文件系统及其后代，并将复制的流发送到另一个池 `users2`。

```
# zfs create users2 mirror c0t1d0 c1t1d0
# zfs receive -F -d users2 < /snaps/users-R
# zfs list
NAME                 USED  AVAIL  REFER  MOUNTPOINT
users                224K  33.2G    22K  /users
users@today             0      -    22K  -
users/user1           33K  33.2G    18K  /users/user1
users/user1@today     15K      -    18K  -
users/user2           18K  33.2G    18K  /users/user2
users/user2@today       0      -    18K  -
users/user3           18K  33.2G    18K  /users/user3
users/user3@today       0      -    18K  -
users2               188K  16.5G    22K  /users2
users2@today            0      -    22K  -
users2/user1          18K  16.5G    18K  /users2/user1
users2/user1@today      0      -    18K  -
users2/user2          18K  16.5G    18K  /users2/user2
users2/user2@today      0      -    18K  -
users2/user3          18K  16.5G    18K  /users2/user3
users2/user3@today      0      -    18K  -
```

远程复制 ZFS 数据

可以使用 `zfs send` 和 `zfs recv` 命令，将快照流表示从一个系统远程复制到另一个系统。例如：

```
# zfs send tank/cindy@today | ssh newsys zfs recv sandbox/restfs@today
```

此命令发送 `tank/cindy@today` 快照数据，并在 `sandbox/restfs` 文件系统中予以接收。该命令还会在 `newsys` 系统上创建 `restfs@today` 快照。在此示例中，已将用户配置为在远程系统上使用 `ssh`。

♦ ♦ ♦ 第 7 章

7 使用 ACL 和属性保护 Oracle Solaris ZFS 文件

本章介绍有关使用访问控制列表 (access control list, ACL) 通过提供比标准 UNIX 权限更详尽的权限来保护 ZFS 文件的信息。

本章包含以下各节：

Solaris ACL 模型

Solaris 的旧版本支持的 ACL 实现主要基于 POSIX ACL 草案规范。基于 POSIX 草案的 ACL 用于保护 UFS 文件，由 NFSv4 之前的 NFS 版本转换。

引入 NFSv4 后，新 ACL 模型完全支持 NFSv4 在 UNIX 和非 UNIX 客户机之间提供的互操作性。如 NFSv4 规范中所定义，这一新的 ACL 实现提供了更丰富的基于 NT 样式 ACL 的语义。

与旧模型相比，新 ACL 模型的主要变化如下：

- 基于 NFSv4 规范并与 NT 样式的 ACL 相似。
- 提供了更详尽的访问特权集合。有关更多信息，请参见表 7-2 “ACL 访问特权”。
- 分别使用 `chmod` 和 `ls` 命令（而非 `setfacl` 和 `getfacl` 命令）进行设置和显示。
- 提供了更丰富的继承语义，用于指定如何将访问特权从目录应用到子目录等。有关更多信息，请参见“ACL 继承” [193]。

两种 ACL 模型均可比标准文件权限提供更精细的访问控制。与 POSIX 式 ACL 非常相似，新 ACL 也由多个访问控制条目 (Access Control Entry, ACE) 构成。

POSIX 式 ACL 使用单个条目定义允许哪些权限，拒绝哪些权限。而新 ACL 模型包含两种类型的 ACE，用于进行访问检查： `ALLOW` 和 `DENY`。因此，不能根据定义了一组权限的单个 ACE 来推断是允许还是拒绝该 ACE 中未定义的权限。

NFSv4 样式的 ACL 与 POSIX 式 ACL 之间的转换如下：

- 如果使用任何可识别 ACL 的实用程序（如 cp、mv、tar、cpio 或 rcp 命令）将具有 ACL 的 UFS 文件传送到 ZFS 文件系统，则 POSIX 式 ACL 会转换为等效的 NFSv4 样式的 ACL。
- 一些 NFSv4 样式的 ACL 会转换为 POSIX 式 ACL。如果 NFSv4 样式的 ACL 未转换为 POSIX 式 ACL，则会显示类似于以下内容的消息：

```
# cp -p filea /var/tmp
cp: failed to set acl entries on /var/tmp/filea
```

- 如果在运行当前 Solaris 发行版的系统上使用保留的 ACL 选项（tar -p 或 cpio -P）创建 UFS tar 或 cpio 归档文件，则在运行以前的 Solaris 发行版的系统中提取该归档文件时将丢失 ACL。

  所有文件都以正确的文件模式提取，但会忽略 ACL 项。
- 可以使用 ufsrestore 命令将数据恢复至 ZFS 文件系统中。如果原始数据包括 POSIX 样式的 ACL，则这些 ACL 会被转换为 NFSv4 样式的 ACL。
- 如果尝试对 UFS 文件设置 NFSv4 样式的 ACL，则会显示类似于以下内容的消息：

```
chmod: ERROR: ACL type's are different
```

- 如果尝试对 ZFS 文件设置 POSIX 样式的 ACL，则会显示以下类似信息：

```
# getfacl filea
File system doesn't support aclent_t style ACL's.
See acl(5) for more information on Solaris ACL support.
```

有关对 ACL 和备份产品的其他限制信息，请参见“使用其他备份产品保存 ZFS 数据” [178]。

ACL 设置语法的说明

提供以下两种基本的 ACL 格式：

- **普通 ACL** – 只包含传统的 UNIX user、group 和 owner 条目。

  使用以下命令语法设置普通 ACL。

```
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
```

- **非普通 ACL** – 不仅包含 owner、group 和 everyone 条目，还包含其他条目，或者包含一组继承标志，或者是其中的条目以非传统顺序排列。

使用以下命令语法设置非普通 ACL。

```
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
```

以下列表说明了设置普通和非普通 ACL 的命令所使用的选项。

owner@, group@, everyone@
: 标识用于普通 ACL 语法的 *ACL-entry-type*。有关 *ACL-entry-type* 的说明，请参见表 7-1 “ACL 项类型”。

user|group:*ACL-entry-ID*（*username* 或 *groupname*）
: 标识用于显式 ACL 语法的 *ACL-entry-type*。用户和组的 *ACL-entry-type* 还必须包含 *ACL-entry-ID*、*username* 或 *groupname*。有关 *ACL-entry-type* 的说明，请参见表 7-1 “ACL 项类型”。

access-permissions/.../
: 标识授予或拒绝的访问权限。有关 ACL 访问特权的说明，请参见表 7-2 “ACL 访问特权”。

inheritance-flags
: 标识一组可选的 ACL 继承标志。有关 ACL 继承标志的说明，请参见表 7-4 “ACL 继承标志”。

deny |allow
: 标识授予还是拒绝访问权限。

在以下示例中，owner@、group@ 或 everyone@ 不存在 ACL-entry-ID 值。

```
group@:write_data/append_data/execute:deny
```

由于 ACL 中包括特定用户 (ACL-entry-type)，因此以下示例中包括 ACL-entry-ID。

```
0:user:joe:list_directory/read_data/execute:allow
```

显示的 ACL 项类似于以下内容：

```
2:group@:write_data/append_data/execute:deny
```

本示例中指定的2 或索引 ID 用于标识较大 ACL 中的 ACL 项，较大的 ACL 中可能包含对应于所有者、特定 UID、组和各用户的多个项。可以使用 chmod 命令指定索引 *ID*，以标识 ACL 要修改的部分。例如，可将索引 ID 3 标识为 chmod 命令中的 A3，与以下内容类似：

```
chmod A3=user:venkman:read_acl:allow filename
```

下表说明了 ACL 项类型，即所有者、组和其他用户的 ACL 表示形式。

表 7-1 ACL 项类型

ACL 项类型	说明
owner@	指定授予对象所有者的访问权限。
group@	指定授予对象所属组的访问权限。
everyone@	指定向不与其他任何 ACL 项匹配的任何用户或组授予的访问权限。
user	通过用户名指定向对象的其他用户授予的访问权限。必须包括 *ACL-entry-ID*，其中包含 *username* 或 *userID*。如果该值不是有效的数字 UID 或 *username*，则该 ACL 项的类型无效。
group	通过组名指定向对象的其他组授予的访问权限。必须包括 *ACL-entry-ID*，其中包含 *groupname* 或 *groupID*。如果该值不是有效的数字 UID 或 *groupname*，则该 ACL 项的类型无效。

下表介绍了 ACL 访问特权。

表 7-2 ACL 访问特权

访问特权	缩写访问特权	说明
add_file	w	向目录中添加新文件的权限。
add_subdirectory	p	在目录中创建子目录的权限。
append_data	p	当前未实现。
delete	d	删除文件的权限。有关特定的 delete 权限行为的更多信息，请参见表 7-3 “ACL delete 和 delete_child 权限行为”。
delete_child	D	删除目录中的文件或目录的权限。有关特定的 delete_child 权限行为的更多信息，请参见表 7-3 “ACL delete 和 delete_child 权限行为”。
execute	x	执行文件或搜索目录内容的权限。
list_directory	r	列出目录内容的权限。
read_acl	c	读取 ACL 的权限 (ls)。
read_attributes	a	读取文件的基本属性（非 ACL）的权限。将基本属性视为 stat 级别属性。允许此访问掩码位意味着该实体可以执行 ls(1) 和 stat(2)。
read_data	r	读取文件内容的权限。
read_xattr	R	读取文件的扩展属性或在文件的扩展属性目录中执行查找的权限。
synchronize	s	当前未实现。
write_xattr	W	创建扩展属性或向扩展属性目录进行写入的权限。 向用户授予此权限意味着用户可为文件创建扩展属性目录。属性文件的权限可以控制用户对属性的访问。
write_data	w	修改或替换文件内容的权限。
write_attributes	A	将与文件或目录关联的时间更改为任意值的权限。
write_acl	C	编写 ACL 的权限或使用 chmod 命令修改 ACL 的能力。
write_owner	o	更改文件的所有者或组的权限，或者对文件执行 chown 或 chgrp 命令的能力。

访问特权	缩写访问特权	说明
		获取文件所有权的权限或将文件的组所有权更改为由用户所属组的权限。如果要将文件或组的所有权更改为任意用户或组，则需要 PRIV_FILE_CHOWN 特权。

下表提供了有关 ACL delete 和 delete_child 行为的其他详细信息。

表 7-3 ACL delete 和 delete_child 权限行为

父目录权限	目标对象权限		
	ACL 允许 delete	ACL 拒绝 delete	未指定 delete 权限
ACL 允许 delete_child	允许	允许	允许
ACL 拒绝 delete_child	允许	拒绝	拒绝
ACL 仅允许 write 和 execute	允许	允许	允许
ACL 拒绝 write 和 execute	允许	拒绝	拒绝

ZFS ACL 集合

可以在 *ACL* 集合中应用以下 ACL 组合，而不需要分别设置各个权限。有以下 ACL 集合可用。

ACL 集合名称	包括的 ACL 权限
full_set	所有权限
modify_set	除 write_acl 和 write_owner 外的所有权限
read_set	read_data、read_attributes、read_xattr 和 read_acl
write_set	write_data、append_data、write_attributes 和 write_xattr

这些 ACL 集合是预定义的，不能修改。

ACL 继承

使用 ACL 继承的目的是使新创建的文件或目录可以继承需要继承的 ACL，同时考虑父目录的现有权限位。

缺省情况下，不会传播 ACL。如果在某个目录上设置了非普通 ACL，则任何后续目录都不会继承该 ACL。必须对文件或目录指定 ACL 的继承。

下表中介绍了可选继承标志。

注 - 当前，successful_access、failed_access 和 inherited 标志仅适用于 SMB 服务器。

表 7-4 ACL 继承标志

继承标志	缩写继承标志	说明
file_inherit	f	仅将 ACL 从父目录继承到该目录中的文件。
dir_inherit	d	仅将 ACL 从父目录继承到该目录的子目录。
inherit_only	i	从父目录继承 ACL，但仅适用于新创建的文件或子目录，而不适用于该目录自身。该标志要求使用 file_inherit 标志或 dir_inherit 标志，或同时使用两者来表示要继承的内容。
no_propagate	n	仅将 ACL 从父目录继承到该目录的第一级内容，而不是第二级或后续内容。该标志要求使用 file_inherit 标志或 dir_inherit 标志，或同时使用两者来表示要继承的内容。
-	N/A	不授予权限。
successful_access	S	指示在成功访问时是否应该启动报警或审计记录。此标志随审计或报警 ACE 类型一起使用。
failed_access	F	指示在访问失败时是否应该启动报警或审计记录。此标志随审计或报警 ACE 类型一起使用。
inherited	I	表示某个 ACE 是继承的。

此外，还可以使用 aclinherit 文件系统属性对文件系统设置更为严格或更为宽松的缺省 ACL 继承策略。有关更多信息，请参见下一节。

ACL 属性

ZFS 文件系统包括了以下 ACL 属性来确定 ACL 继承的特定行为和 ACL 与 chmod 操作的交互。

- aclinherit – 确定 ACL 继承的行为。包括以下属性值：
  - discard – 对于新对象，创建文件或目录时不会继承任何 ACL 项。文件或目录的 ACL 等效于该文件或目录的权限模式。
  - noallow – 对于新对象，仅继承访问类型为 deny 的可继承 ACL 项。
  - restricted – 对于新对象，继承 ACL 项时将删除 write_owner 和 write_acl 权限。
  - passthrough – 当属性值设置为 passthrough 时，会使用由可继承 ACE 确定的模式来创建文件。如果不存在影响模式的可继承 ACE，则会根据应用程序要求的模式设置模式。
  - passthrough-x – 与 passthrough 语义相同，只不过如果启用 passthrough-x，将使用执行 (x) 权限创建文件，但前提是必须在文件创建模式和影响模式的可继承 ACE 中设置执行权限。

aclinherit 的缺省模式为 restricted。

- aclmode – 在最初创建文件时修改 ACL 行为，或者在 chmod 操作期间控制如何修改 ACL。包括以下属性值：
  - discard – aclmode 属性为 discard 的文件系统将删除不表示文件模式的所有 ACL 项。这是缺省值。
  - mask – aclmode 属性为 mask 的文件系统将减少用户或组的权限。除非用户项与文件或目录的所有者具有相同的 UID，否则将减少权限，以使其不会大于组权限位。在这种情况下，减少 ACL 权限，以使其不会大于所有者权限位。如果未执行显式 ACL 集合操作，则 mask 值还会在模式更改之后保留 ACL。
  - passthrough – aclmode 属性为 passthrough 的文件系统指示除了生成必要的 ACL 项来表示文件或目录的新模式外，不对 ACL 进行其他更改。

  aclmode 的缺省模式为 discard。

  有关使用 aclmode 属性的更多信息，请参见例 7-14 “ACL 与对 ZFS 文件的 chmod 操作的交互”。

设置 ZFS 文件的 ACL

正如 ZFS 所实现的那样，ACL 由 ACL 项的数组构成。ZFS 提供了一个纯 ACL 模型，其中所有文件都包括 ACL。通常，ACL 很普通，因为它仅表示传统的 UNIX owner/group/other 项。

ZFS 文件仍然具有权限位和模式，但这些值大部分是 ACL 所表示内容的高速缓存。因此，如果更改文件的权限，该文件的 ACL 也会相应地更新。此外，如果删除授予某一用户对文件或目录访问权限的非普通 ACL，而该文件或目录的权限位将访问权限授予组或所有用户，则该用户仍可访问这一文件或目录。所有访问控制决策都由文件或目录的 ACL 中表示的权限来管理。

对于 ZFS 文件，ACL 访问权限的主要规则如下：

- ZFS 按照 ACL 项在 ACL 中的排列顺序从上至下对其进行处理。
- 仅处理具有与访问权限的请求者匹配的“对象”的 ACL 项。
- 一旦授予允许权限，同一 ACL 权限集当中的后续 ACL 拒绝项即不能拒绝此权限。
- 无条件地授予文件所有者 write_acl 权限，即使显式拒绝此权限时也是如此。否则，将拒绝仍未指定的所有权限。

  如果拒绝权限或缺少访问权限，特权子系统将确定为文件所有者或超级用户授予的访问请求。此机制可以防止文件所有者无法访问其文件，并允许超级用户修改文件以进行恢复。

如果在某个目录上设置了非普通 ACL，则该目录的子项不会自动继承该 ACL。如果设置了非普通 ACL 并希望目录的子项继承该 ACL，则必须使用 ACL 继承标志。有关更多信息，请参见表 7-4 “ACL 继承标志” 和“以详细格式对 ZFS 文件设置 ACL 继承” [202]。

创建新文件时，根据 umask 值将应用类似如下的缺省的普通 ACL：

```
$ 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
```

在此示例中每个用户类别（owner@、group@、everyone@）都具有一个 ACL 项。

此文件 ACL 的说明如下：

0:owner@	所有者可以读取和修改文件的内容 (read_data/write_data/append_data/read_xattr)。所有者还可以修改文件的属性，如时间戳、扩展属性和 ACL (write_xattr/read_attributes/write_attributes/ read_acl/write_acl)。此外，所有者还可以修改文件的所有权 (write_owner:allow)。 synchronize 访问权限当前未实现。
1:group@	向组授予对文件和文件属性的读取权限 (read_data/read_xattr/read_attributes/read_acl:allow)。
2:everyone@	向用户或组之外的所有用户授予对文件和文件属性的读取权限 (read_data/read_xattr/read_attributes/read_acl/synchronize:allow)。synchronize 访问权限当前未实现。

创建新目录时，根据 umask 值，缺省目录 ACL 将类似如下：

```
$ 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
```

此目录 ACL 的说明如下：

0:owner@	所有者可以读取和修改目录内容 (list_directory/read_data/add_file/write_data/add_subdirectory /append_data)，以及读取和修改文件的属性，如时间戳、扩展属性和 ACL (/read_xattr/write_xattr/read_attributes/write_attributes/read_acl/

write_acl)。此外，所有者可以搜索内容 (execute)，删除文件或目录 (delete_child)，以及修改目录的所有权 (write_owner:allow)。synchronize 访问权限当前未实现。

1:group@	组可以列出和读取目录内容和目录属性。此外，组还具有搜索目录内容的执行权限 (list_directory/read_data/read_xattr/execute/read_attributes /read_acl/synchronize:allow)。
2:everyone@	向用户或组之外的所有人员授予对目录内容和目录属性的读取和执行权限 (list_directory/read_data/read_xattr/execute/read_attributes/read_acl/synchronize:allow)。synchronize 访问权限当前未实现。

以详细格式设置和显示 ZFS 文件的 ACL

可以使用 chmod 命令修改 ZFS 文件的 ACL。以下用于修改 ACL 的 chmod 语法使用 *acl-specification* 来确定 ACL 的格式。有关 *acl-specification* 的说明，请参见“ACL 设置语法的说明” [190]。

- 添加 ACL 项
  - 为用户添加 ACL 项

    ```
    % chmod A+acl-specification filename
    ```
  - 按 *index-ID* 添加 ACL 项

    ```
    % chmod Aindex-ID+acl-specification filename
    ```

    此语法用于在指定的 *index-ID* 位置插入新的 ACL 项。
- 替换 ACL 项

  ```
  % chmod A=acl-specification filename
  ```

  ```
  % chmod Aindex-ID=acl-specification filename
  ```
- 删除 ACL 项
  - 按 *index-ID* 删除 ACL 项

    ```
    % chmod Aindex-ID- filename
    ```
  - 由用户删除 ACL 项

    ```
    % chmod A-acl-specification filename
    ```
  - 从文件中删除所有非普通 ACE

    ```
    % chmod A- filename
    ```

详细 ACL 信息是通过使用 `ls -v` 命令来显示的。例如：

```
# 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
```

有关使用缩写 ACL 格式的信息，请参见“以缩写格式设置和显示 ZFS 文件的 ACL” [207]。

例 7-1 修改 ZFS 文件的普通 ACL

本节提供了有关设置和显示普通 ACL 的示例，普通 ACL 是指 ACL 中只包含传统的 UNIX 条目（用户、组以及其他人）。

在以下示例中，普通 ACL 存在于 `file.1` 中：

```
# 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
```

在以下示例中，为 `group@` 授予了 `write_data` 权限。

```
# 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
```

在以下示例中，对 `file.1` 的权限重新设置为 644。

```
# 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 设置 ZFS 文件的非普通 ACL

本节提供了设置和显示非普通 ACL 的示例。

在以下示例中，为用户 joe 添加了对 test.dir 目录的 read_data/execute 权限。

```
# chmod A+user:joe:read_data/execute:allow test.dir
# ls -dv test.dir
drwxr-xr-x+  2 root     root           2 Jul 20 14:23 test.dir
0:user:joe:list_directory/read_data/execute:allow
1:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
/append_data/read_xattr/write_xattr/execute/delete_child
/read_attributes/write_attributes/read_acl/write_acl/write_owner
/synchronize:allow
2:group@:list_directory/read_data/read_xattr/execute/read_attributes
/read_acl/synchronize:allow
3:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
/read_acl/synchronize:allow
```

在以下示例中，为用户 joe 删除了 read_data/execute 权限。

```
# 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 与 ZFS 文件权限的 ACL 交互

以下 ACL 示例展示了在设置 ACL 与随后更改文件或目录的权限位之间发生的交互。

在以下示例中，普通 ACL 存在于 file.2 中：

```
# 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
```

在以下示例中，从 everyone@ 中删除了 ACL allow 权限。

```
# 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
```

在此输出中，文件的权限位从 644 重置为 640。删除 everyone@ 的 ACL 允许权限时，已有效地从文件的权限位中删除了 everyone@ 的读取权限。

在以下示例中，现有 ACL 将替换为 everyone@ 的 read_data/write_data 权限。

```
# 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
```

在此输出中，chmod 语法有效地将现有 ACL 中的 read_data/write_data:allow 权限替换为对所有者、组和 everyone@ 的读取/写入权限。在此模型中，everyone@ 用于指定对任何用户或组的访问权限。由于不存在用以覆盖所有者和组的权限的 owner@ 或 group@ ACL 项，因此权限位会设置为 666。

在以下示例中，现有 ACL 将替换为用户 joe 的读取权限。

```
# chmod A=user:joe:read_data:allow file.3
# ls -v file.3
----------+  1 root     root        2440 Jul 20 14:28 file.3
0:user:joe:read_data:allow
```

在此输出中，文件权限计算结果为 000，这是因为不存在对应 owner@、group@ 或 everyone@ 的 ACL 项，这些项用于表示文件的传统权限组成部分。文件所有者可通过重置权限（和 ACL）来解决此问题，如下所示：

```
# 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 恢复 ZFS 文件的普通 ACL

可以使用 chmod 命令来删除文件或目录的所有非普通 ACL。

在以下示例中，test5.dir 中存在两个非普通 ACE。

```
# 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:joe: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
```

在以下示例中，删除了用户 `joe` 和 `lp` 的非普通 ACL。剩余的 ACL 包含 `owner@`、`group@` 和 `everyone@` 的缺省值。

```
# 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 将 ACL 集合应用于 ZFS 文件

因为有 ACL 集合可用，因此您不必分别应用各个 ACL 权限。有关 ACL 集合的说明，请参见“ZFS ACL 集合” [193]。

例如，您可以如下所示应用 `read_set`：

```
# chmod A+user:bob:read_set:allow file.1
# ls -v file.1
-r--r--r--+  1 root     root      206695 Jul 20 13:43 file.1
0:user:bob:read_data/read_xattr/read_attributes/read_acl:allow
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
```

您可以如下所示应用 `write_set` 和 `read_set`：

```
# chmod A+user:bob:read_set/write_set:allow file.2
# ls -v file.2
-rw-r--r--+  1 root     root        2693 Jul 20 14:26 file.2
0:user:bob: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
```

以详细格式对 ZFS 文件设置 ACL 继承

可以确定如何在文件和目录中继承或不继承 ACL。缺省情况下，不会传播 ACL。如果在某个目录上设置了非普通 ACL，则任何后续目录都不会继承该 ACL。必须对文件或目录指定 ACL 的继承。

可以在文件系统中全局设置 `aclinherit` 属性。缺省情况下，`aclinherit` 设置为 `restricted`。

有关更多信息，请参见“ACL 继承” [193]。

例 7-6 授予缺省 ACL 继承

缺省情况下，ACL 不通过目录结构传播。

在以下示例中，为用户 `joe` 应用了针对 `test.dir` 的非普通 ACE `read_data/write_data/execute`。

```
# chmod A+user:joe: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:joe: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
```

如果创建了 `test.dir` 子目录，则不会传播用户 `joe` 的 ACE。如果对 `sub.dir` 的权限授予用户 `joe` 作为文件所有者、组成员或 `everyone@` 进行访问的权限，则该用户只能访问 `sub.dir`。

```
# 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 对文件和目录授予 ACL 继承

以下一系列示例标识了设置 file_inherit 标志时应用的文件和目录的 ACE。

在以下示例中，为用户 joe 添加了对 test2.dir 目录中的文件的 read_data/write_data 权限，以便该用户对任何新创建的文件都具有读取访问权限。

```
# chmod A+user:joe:read_data/write_data:file_inherit:allow test2.dir
# ls -dv test2.dir
drwxr-xr-x+  2 root     root           2 Jul 20 14:55 test2.dir
0:user:joe:read_data/write_data:file_inherit:allow
1:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
/append_data/read_xattr/write_xattr/execute/delete_child
/read_attributes/write_attributes/read_acl/write_acl/write_owner
/synchronize:allow
2:group@:list_directory/read_data/read_xattr/execute/read_attributes
/read_acl/synchronize:allow
3:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
/read_acl/synchronize:allow
```

在以下示例中，用户 joe 的权限应用于新创建的 test2.dir/file.2 文件。授予 ACL 继承 read_data:file_inherit:allow 意味着用户 joe 可以读取任何新创建的文件的内容。

```
# touch test2.dir/file.2
# ls -v test2.dir/file.2
-rw-r--r--+  1 root     root           0 Jul 20 14:56 test2.dir/file.2
0:user:joe:read_data:inherited:allow
1:owner@:read_data/write_data/append_data/read_xattr/write_xattr
/read_attributes/write_attributes/read_acl/write_acl/write_owner
/synchronize:allow
2:group@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
3:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
:allow
```

由于此文件系统的 aclinherit 属性设置为缺省模式 restricted，因此用户 joe 对 file.2 不具有 write_data 权限，这是因为该文件的组权限不允许使用此权限。

请注意，设置 file_inherit 或 dir_inherit 标志时所应用的 inherit_only 权限用来通过目录结构传播 ACL。因此，除非用户 joe 是文件的所有者或文件所属组的成员，否则将仅授予或拒绝该用户 everyone@ 权限中的权限。例如：

```
# mkdir test2.dir/subdir.2
# ls -dv test2.dir/subdir.2
drwxr-xr-x+  2 root     root           2 Jul 20 14:57 test2.dir/subdir.2
0:user:joe:list_directory/read_data/add_file/write_data:file_inherit
/inherit_only/inherited:allow
1:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
```

```
/append_data/read_xattr/write_xattr/execute/delete_child
/read_attributes/write_attributes/read_acl/write_acl/write_owner
/synchronize:allow
2:group@:list_directory/read_data/read_xattr/execute/read_attributes
/read_acl/synchronize:allow
3:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
/read_acl/synchronize:allow
```

以下一系列示例标识了同时设置 `file_inherit` 和 `dir_inherit` 标志时所应用的文件和目录的 ACL。

在以下示例中，向用户 `joe` 授予了继承用于新创建的文件和目录的读取、写入和执行权限。

```
# chmod A+user:joe:read_data/write_data/execute:file_inherit/dir_inherit:allow
test3.dir
# ls -dv test3.dir
drwxr-xr-x+  2 root     root           2 Jul 20 15:00 test3.dir
0:user:joe:list_directory/read_data/add_file/write_data/execute
:file_inherit/dir_inherit:allow
1:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
/append_data/read_xattr/write_xattr/execute/delete_child
/read_attributes/write_attributes/read_acl/write_acl/write_owner
/synchronize:allow
2:group@:list_directory/read_data/read_xattr/execute/read_attributes
/read_acl/synchronize:allow
3:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
/read_acl/synchronize:allow
```

下面输出中的 `inherited` 文本是信息性消息，指示该 ACE 是继承的。

```
# touch test3.dir/file.3
# ls -v test3.dir/file.3
-rw-r--r--+  1 root     root           0 Jul 20 15:01 test3.dir/file.3
0:user:joe:read_data:inherited:allow
1:owner@:read_data/write_data/append_data/read_xattr/write_xattr
/read_attributes/write_attributes/read_acl/write_acl/write_owner
/synchronize:allow
2:group@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
3:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
:allow

# touch test3.dir/file.3
# ls -v test3.dir/file.3
-rw-r--r--+  1 root     root           0 Jun 23 15:25 test3.dir/file.3
0:user:joe:read_data: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

# mkdir test3.dir/subdir.1
```

```
# ls -dv test3.dir/subdir.1
drwxr-xr-x+  2 root     root           2 Jun 23 15:26 test3.dir/subdir.1
0:user:joe:list_directory/read_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/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
```

在上面的示例中，由于 group@ 和 everyone@ 的父目录的权限位拒绝写入和执行权限，因此拒绝了用户 joe 的写入和执行权限。缺省的 aclinherit 属性为 restricted，这意味着未继承 write_data 和 execute 权限。

在以下示例中，向用户 joe 授予了继承用于新创建的文件的读取、写入和执行权限，但未将这些权限传播给该目录的后续内容。

```
# chmod A+user:joe:read_data/write_data/execute:file_inherit/no_propagate:allow
test4.dir
# ls -dv test4.dir
drwxr--r--+  2 root     root           2 Mar  1 12:11 test4.dir
0:user:joe:list_directory/read_data/add_file/write_data/execute
:file_inherit/no_propagate:allow
1:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
/append_data/read_xattr/write_xattr/execute/delete_child
/read_attributes/write_attributes/read_acl/write_acl/write_owner
/synchronize:allow
2:group@:list_directory/read_data/read_xattr/read_attributes/read_acl
/synchronize:allow
3:everyone@:list_directory/read_data/read_xattr/read_attributes/read_acl
/synchronize:allow
```

如以下示例所示，基于所属组的权限降低了用户 joe 的 read_data/write_data/execute 权限。

```
# touch test4.dir/file.4
# ls -v test4.dir/file.4
-rw-r--r--+  1 root     root           0 Jul 20 15:09 test4.dir/file.4
0:user:joe:read_data:inherited:allow
1:owner@:read_data/write_data/append_data/read_xattr/write_xattr
/read_attributes/write_attributes/read_acl/write_acl/write_owner
/synchronize:allow
2:group@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
3:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
:allow
```

例 7-8 ACL 继承模式设置为 Passthrough 时的 ACL 继承

如果 tank/cindy 文件系统的 aclinherit 属性设置为 passthrough，则对于新创建的 file.5，用户 joe 将继承应用于 test4.dir 的 ACL，如下所示：

```
# 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:joe:read_data/write_data/execute:inherited:allow
1:owner@:read_data/write_data/append_data/read_xattr/write_xattr
/read_attributes/write_attributes/read_acl/write_acl/write_owner
/synchronize:allow
2:group@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
3:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
:allow
```

例 7-9 ACL 继承模式设置为 Discard 时的 ACL 继承

如果将文件系统的 `aclinherit` 属性设置为 `discard`，则目录的权限位更改时，可能会废弃 ACL。例如：

```
# zfs set aclinherit=discard tank/cindy
# chmod A+user:joe:read_data/write_data/execute:dir_inherit:allow test5.dir
# ls -dv test5.dir
drwxr-xr-x+  2 root     root           2 Jul 20 14:18 test5.dir
0:user:joe:list_directory/read_data/add_file/write_data/execute
:dir_inherit:allow
1:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
/append_data/read_xattr/write_xattr/execute/delete_child
/read_attributes/write_attributes/read_acl/write_acl/write_owner
/synchronize:allow
2:group@:list_directory/read_data/read_xattr/execute/read_attributes
/read_acl/synchronize:allow
3:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
/read_acl/synchronize:allow
```

如果以后决定要加强目录的权限位，则会废弃非普通 ACL。例如：

```
# 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 继承模式设置为 Noallow 时的 ACL 继承

在以下示例中，设置了两个包含文件继承的非普通 ACL。一个 ACL 允许 `read_data` 权限，一个 ACL 拒绝 `read_data` 权限。此示例还说明了如何可在同一 `chmod` 命令中指定两个 ACE。

```
# zfs set aclinherit=noallow tank/cindy
# chmod A+user:joe:read_data:file_inherit:deny,user:lp:read_data:file_inherit:allow
test6.dir
# ls -dv test6.dir
drwxr-xr-x+  2 root     root           2 Jul 20 14:22 test6.dir
0:user:joe:read_data:file_inherit:deny
1:user:lp:read_data:file_inherit:allow
2:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
/append_data/read_xattr/write_xattr/execute/delete_child
/read_attributes/write_attributes/read_acl/write_acl/write_owner
/synchronize:allow
3:group@:list_directory/read_data/read_xattr/execute/read_attributes
/read_acl/synchronize:allow
4:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
/read_acl/synchronize:allow
```

如以下示例所示，创建新文件时，将废弃允许 `read_data` 权限的 ACL。

```
# touch test6.dir/file.6
# ls -v test6.dir/file.6
-rw-r--r--+  1 root     root           0 Jul 20 14:23 test6.dir/file.6
0:user:joe:read_data:inherited:deny
1:owner@:read_data/write_data/append_data/read_xattr/write_xattr
/read_attributes/write_attributes/read_acl/write_acl/write_owner
/synchronize:allow
2:group@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
3:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
:allow
```

以缩写格式设置和显示 ZFS 文件的 ACL

可通过使用 14 个唯一字母表示权限的缩写格式来设置和显示 ZFS 文件的权限。表 7-2 “ACL 访问特权” 和表 7-4 “ACL 继承标志” 列出了表示简写权限的字母。

可以使用 `ls -V` 命令显示用于文件和目录的缩写 ACL 列表。例如：

```
# 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
```

以下介绍了缩写的 ACL 输出：

`owner@` 所有者可以读取和修改文件的内容（`rw=read_data/write_data`、`p=append_data`）。所有者还可以修改文件的属性，如时间戳、扩展属性和 ACL（`a=read_attributes` 、`W=write_xattr`、`R=`

read_xattr、A=write_attributes、c=read_acl、C=write_acl）。此外，所有者还可以修改文件的所有权 (o=write_owner)。

synchronize (s) 访问权限当前未实现。

group@ 向组授予对文件的读取权限 (r= read_data) 和对文件属性的读取权限（a=read_attributes 、R=read_xattr、c= read_acl）。

synchronize (s) 访问权限当前未实现。

everyone@ 向用户或组之外的所有人员授予对文件以及文件属性的读取权限（r=read_data、a=append_data、R=read_xattr、c=read_acl 和 s=synchronize）。

synchronize (s) 访问权限当前未实现。

与详细 ACL 格式相比，缩写 ACL 格式具有以下优点：

- 可将权限指定为 chmod 命令的位置参数。
- 可以删除用于标识无权限的连字符 (-) 字符，并且只需指定必需的字母。
- 可以同一方式设置权限和继承标志。

有关使用详细 ACL 格式的信息，请参见“以详细格式设置和显示 ZFS 文件的 ACL” [197]。

例 7-11 以缩写格式设置和显示 ACL

在以下示例中，普通 ACL 存在于 file.1 中：

```
# 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
```

在本示例中，为用户 joe 添加了对 file.1 的 read_data/execute 权限。

```
# chmod A+user:joe:rx:allow file.1
# ls -V file.1
-rw-r--r--+  1 root     root      206695 Jul 20 14:27 file.1
user:joe:r-x-----------:-------:allow
owner@:rw-p--aARWcCos:-------:allow
group@:r-----a-R-c--s:-------:allow
everyone@:r-----a-R-c--s:-------:allow
```

在以下示例中，通过使用缩写 ACL 格式向用户 joe 授予了在新创建文件和目录时会继承的读取、写入和执行权限。

```
# chmod A+user:joe:rwx:fd:allow dir.2
# ls -dV dir.2
```

```
drwxr-xr-x+  2 root     root           2 Jul 20 14:33 dir.2
user:joe:rwx-----------:fd-----:allow
owner@:rwxp-DaARWcCos:-------:allow
group@:r-x---a-R-c--s:-------:allow
everyone@:r-x---a-R-c--s:-------:allow
```

另外，还可以剪切 `ls -V` 输出中的权限和继承标志并将其粘贴到缩写的 `chmod` 格式中。例如，要将用户 `joe` 对 `dir.2` 的权限和继承标志复制给 `dir.2` 的用户 `cindy`，可将权限和继承标志 (`rwx-----------:fd-----:allow`) 复制并粘贴到您的 `chmod` 命令中。例如：

```
# 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:joe: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 继承模式设置为 Passthrough 时的 ACL 继承

将 `aclinherit` 属性设置为 `passthrough` 的文件系统会继承所有可继承 ACL 项，并且继承 ACL 项时不会对其进行任何修改。当此属性设置为 `passthrough` 时，会使用由可继承 ACE 确定的权限模式来创建文件。如果不存在影响权限模式的可继承 ACE，则会根据应用程序要求的模式设置权限模式。

以下示例使用缩写 ACL 语法来说明如何通过将 `aclinherit` 模式设置为 `passthrough` 来继承权限位。

在本示例中，对 `test1.dir` 设置了 ACL 以强制继承。该语法会为新创建的文件创建 `owner@`、`group@` 和 `everyone@` ACL 项。新创建的目录会继承 `@owner`、`group@` 和 `everyone@` ACL 项。

```
# 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
```

在此示例中，新创建的文件会继承指定继承到新创建的文件的 ACL。

```
# 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
```

在本示例中，新创建的目录会继承用于控制对此目录访问权限的 ACE 以及用于将来传播到新创建目录的子级的 ACE。

```
# 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` 项用于传播继承，在访问控制期间不会被考虑。

在以下示例中，在不存在继承 ACE 的另一目录中使用普通 ACL 创建了一个文件。

```
# 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 继承模式设置为 Passthrough-X 时的 ACL 继承

如果启用 `aclinherit=passthrough-x`，对于 `owner@`、`group@` 或 `everyone@` 设置，将使用执行 (x) 权限创建文件，但是只有在文件创建模式以及影响该模式的可继承 ACE 中设置执行权限才行。

以下示例说明了如何通过将 `aclinherit` 模式设置为 `passthrough-x` 来继承执行权限。

```
# zfs set aclinherit=passthrough-x tank/cindy
```

在 `/tank/cindy/test1.dir` 上设置了以下 ACL，以便为 `owner@` 的文件提供可执行 ACL 继承。

```
# 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
```

使用请求的权限 `0666` 创建文件 (`file1`)，但生成的权限为 `0660`。没有继承执行权限的原因是，创建模式未请求该权限。

```
# 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
```

接下来，在 testdir 目录下使用 cc 编译器来生成名为 t 的可执行文件。

```
# cc -o t t.c
# ls -V t
-rwxrwx---+  1 root     root        7396 Dec  3 15:19 t
owner@:rwxpdDaARWcCos:------I:allow
group@:rwxp----------:------I:allow
everyone@:--------------:------I:allow
```

生成的权限为 0770，这是因为 cc 请求了权限 0777，这导致从 owner@、group@ 和 everyone@ 条目继承了执行权限。

例 7-14 ACL 与对 ZFS 文件的 chmod 操作的交互

以下示例展示了特定的 aclmode 和 aclinherit 属性值如何影响现有的 ACL 与 chmod 操作（更改文件或目录权限来减少或扩展现有的任何 ACL 权限以便与所属组一致）的交互。

在此示例中，aclmode 属性设置为 mask，aclinherit 属性设置为 restricted。此示例中的 ACL 权限以简写模式显示，这样可以更方便地展示权限更改。

原始的文件和组所有权以及 ACL 权限如下所示：

```
# 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
```

一个 chown 操作更改了 file.1 的文件所有权，现在所属用户 amy 在查看输出。例如：

```
# 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
```

下面的 chmod 操作将权限更改为限制性更强的模式。在此示例中，sysadmin 组和 staff 组的修改后 ACL 权限未超出所属组的权限。

```
$ 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
```

下面的 chmod 操作将权限更改为限制性更弱的模式。在此示例中，sysadmin 组和 staff 组的修改后 ACL 权限恢复为允许与所属组相同的权限。

```
$ 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
```

向 ZFS 文件应用特殊属性

以下示例展示了如何为 ZFS 文件应用和显示特殊属性，如不变性或只读访问权限。

有关显示和应用特殊属性的更多信息，请参见 ls(1) 和 chmod(1)。

例 7-15 向 ZFS 文件应用不变性

使用以下语法可使文件不变：

```
# chmod S+ci file.1
# echo this >>file.1
-bash: file.1: Not owner
# rm file.1
rm: cannot remove `file.1': Not owner
```

可以使用以下语法显示 ZFS 文件的特殊属性：

```
# ls -l/c file.1
-rw-r--r--+  1 root     root      206695 Jul 20 14:27 file.1
{A-----im----}
```

使用以下语法可删除文件不变性：

```
# 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 向 ZFS 文件应用只读访问权限

以下示例展示了如何向 ZFS 文件应用只读访问权限。

```
# chmod S+cR file.2
# echo this >>file.2
-bash: file.2: Not owner
```

例 7-17 显示和更改 ZFS 文件属性

可以使用以下语法显示和设置特殊属性：

```
# 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}
```

其中一些属性仅适用于 Oracle Solaris SMB 环境。

您可以清除文件的所有属性。例如：

```
# 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 委托管理

本章介绍如何使用委托管理来允许非特权用户执行 ZFS 管理任务。

本章包含以下各节：

ZFS 委托管理概述

通过 ZFS 委托管理，可向特定用户、组或每个人分配精确的权限。支持两种类型的委托权限：

- 可以显式授予单个权限，如 `create`、`destroy`、`mount`、`snapshot` 等。
- 可以定义称为权限集的权限组。以后可以更新权限集，并且权限集的所有使用者都会自动获得更改。权限集以 `@` 符号开头，长度不能超出 64 个字符。在 `@` 符号之后，集名称中的其余字符与标准的 ZFS 文件系统名称具有同样的限制。

ZFS 委托管理可提供与 RBAC 安全模型类似的功能。ZFS 委托方式在管理 ZFS 存储池和文件系统方面具有以下优点：

- 迁移 ZFS 存储池时，权限跟随存储池。
- 提供动态继承性，以便您可以控制权限通过文件系统传播的方式。
- 可进行配置，以便只有文件系统的创建者可以销毁该文件系统。
- 可授予对特定文件系统的权限。新创建的文件系统可以自动获得权限。
- 提供简单的 NFS 管理。例如，具有显式权限的用户可以在适当的 `.zfs/snapshot` 目录中通过 NFS 创建快照。

可考虑使用委托管理来分配 ZFS 任务。有关使用 RBAC 来管理常规 Oracle Solaris 管理任务的信息，请参见《在 Oracle Solaris 11.2 中确保用户和进程的安全 》中的第 1 章 “使用权限控制用户和进程”。

禁用 ZFS 委托权限

使用池的 delegation 属性控制委托管理功能。例如：

```
# 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
```

缺省情况下，delegation 属性处于启用状态。

授予 ZFS 权限

可以使用 zfs allow 命令通过以下方式将对 ZFS 文件系统的权限委托给非 root 用户：

- 可将单个权限授予用户、组或每个人。
- 可将多个权限构成的组作为权限集授予用户、组或每个人。
- 可以仅局部地委托对当前文件系统的权限，也可以委托对当前文件系统的所有后代的权限。

下表介绍了可以委托的操作以及执行委托操作所需要的所有相关权限。

权限（子命令）	说明	相关性
allow	将您拥有的权限授予其他用户的权限。	还必须具有正在允许的权限。
clone	克隆数据集的任何快照的权限。	还必须在原始文件系统中具有 create 权限和 mount 权限。
create	创建后代数据集的权限。	还必须具有 mount 权限。
destroy	销毁数据集的权限。	还必须具有 mount 权限。
diff	在数据集内标识路径的权限。	非 root 用户需要有此权限才能使用 zfs diff 命令。
hold	保持快照的权限。	
mount	挂载和卸载文件系统以及创建和销毁卷设备链路的权限。	
promote	将克隆提升为数据集的权限。	还必须在原始文件系统中具有 mount 权限和 promote 权限。
receive	使用 zfs receive 命令创建后代文件系统的权限。	还必须具有 mount 权限和 create 权限。
release	释放快照保持标志（这样可能会销毁快照）的权限。	

权限（子命令）	说明	相关性
rename	重命名数据集的权限。	还必须在新父级中具有 create 权限和 mount 权限。
rollback	回滚快照的权限。	
send	发送快照流的权限。	
share	共享和取消共享文件系统的权限。	必须同时具有 share 和 share.nfs 才能创建 NFS 共享。 必须同时具有 share 和 share.smb 才能创建 SMB 共享。
snapshot	创建数据集快照的权限。	

您可以授予下面的一组权限，但权限可能只限于访问、读取或更改权限：

- groupquota
- groupused
- key
- keychange
- userprop
- userquota
- userused

此外，还可向非 root 用户委托以下 ZFS 属性的管理：

- aclinherit
- aclmode
- atime
- canmount
- casesensitivity
- checksum
- compression
- copies
- dedup
- defaultgroupquota
- defaultuserquota
- devices
- encryption
- exec
- keysource
- logbias
- mountpoint
- nbmand

- normalization
- primarycache
- quota
- readonly
- recordsize
- refquota
- refreservation
- reservation
- rstchown
- secondarycache
- setuid
- shadow
- share.nfs
- share.smb
- snapdir
- sync
- utf8only
- version
- volblocksize
- volsize
- vscan
- xattr
- zoned

其中有些属性只能在创建数据集时设置。有关这些属性的说明，请参见“介绍 ZFS 属性” [116]。

授予 ZFS 权限 (`zfs allow`)

`zfs allow` 语法如下：

`zfs allow -[ldugecs] everyone|`*user|group[,...] perm|@setname,...] filesystem| volume*

下面的 `zfs allow` 语法（以粗体显示）标识将权限委托给的对象：

`zfs allow [-uge]|user|group|everyone` `[,...]` *filesystem* | *volume*

可以按逗号分隔的列表形式指定多个实体。如果未指定 -uge 选项，则优先将该参数解释为关键字 everyone，然后解释为用户名，最后解释为组名。要指定名为 "everyone" 的用户或组，请使用 -u 或 -g 选项。要将同名的组指定为用户，请使用 -g 选项。-c 选项可授予创建时权限。

以下 zfs allow 语法（以粗体显示）标识权限和权限集的指定方式：

zfs allow [-s] ... perm|@setname [,...] *filesystem* | *volume*

可以按逗号分隔的列表形式指定多个权限。权限名与 ZFS 子命令和属性相同。有关更多信息，请参见上一节。

可以将权限聚合到权限集中，并由 -s 选项来标识。其他 zfs allow 命令可将权限集用于指定的文件系统及其后代。可以对权限集进行动态评估，因此对权限集的更改可立即更新。权限集与 ZFS 文件系统遵循相同的命名要求，但名称必须以 at 符号 (@) 开头，且长度不能超过 64 个字符。

下面的 zfs allow 语法（以粗体显示）标识权限的授予方式：

zfs allow [-ld] ... ... *filesystem* | *volume*

-l 选项指示权限允许用于指定的文件系统，但不允许用于该文件系统的后代，除非同时指定了 -d 选项。-d 选项指示权限允许用于后代文件系统，但不允许用于该文件系统，除非同时指定了 -l 选项。如果两个选项均未指定，则权限允许用于文件系统或卷及其所有后代。

删除 ZFS 委托权限 (zfs unallow)

使用 zfs unallow 命令可以删除以前授予的权限。

例如，假设您授予了 create、destroy、mount 和 snapshot 权限，如下所示：

```
# 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
```

要删除这些权限，请使用以下语法：

```
# zfs unallow cindy tank/home/cindy
# zfs allow tank/home/cindy
```

授予 ZFS 权限示例

例 8-1 向单个用户授予权限

如果您向单个用户授予 create 和 mount 权限，必须确保该用户拥有底层挂载点的权限。

例如，要向用户 mark 授予对 tank 文件系统的 create 和 mount 权限，需要先设置权限：

```
# chmod A+user:mark:add_subdirectory:fd:allow /tank/home
```

然后，使用 zfs allow 命令授予 create、destroy 和 mount 权限。例如：

```
# zfs allow mark create,destroy,mount tank/home
```

现在，用户 mark 可以在 tank/home 文件系统中创建自己的文件系统。例如：

```
# 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 向组授予 create 和 destroy 权限

以下示例展示了如何设置文件系统，以使 staff 组中的每个人都可以在 tank/home 文件系统中创建和挂载文件系统，以及销毁其自己的文件系统。但是，staff 组成员不能销毁其他人的文件系统。

```
# 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
cannot destroy 'tank/home/mark/data': permission denied
```

例 8-3 在正确的文件系统级别授予权限

确保在正确的文件系统级别授予用户权限。例如，向用户 mark 授予了对本地和后代文件系统的 create、destroy 和 mount 权限。向用户 mark 授予了对 tank/home 文件系统创建快照的本地权限，但不允许该用户对自己的文件系统创建快照。因此，未在正确的文件系统级别为他授予 snapshot 权限。

```
# 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
```

要在后代文件系统级别向用户 mark 授予权限，请使用 zfs allow -d 选项。例如：

```
# 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
```

现在，用户 mark 只能在 tank/home 文件系统级别之下创建快照。

例 8-4 定义和使用复杂委托权限

可向用户或组授予特定权限。例如，以下 zfs allow 命令将向 staff 组授予特定权限。此外，还会在创建 tank/home 文件系统后授予 destroy 和 snapshot 权限。

```
# 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
```

由于用户 mark 是 staff 组的成员，因此他可以在 tank/home 中创建文件系统。此外，用户 mark 可以创建 tank/home/mark2 的快照，因为他具有执行此操作的特定权限。例如：

```
# 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
```

但是，用户 mark 不能在 tank/home/mark 中创建快照，因为他不具有执行此操作的特定权限。例如：

```
$ zfs snapshot tank/home/mark@snap1
cannot create snapshot 'tank/home/mark@snap1': permission denied
```

在此示例中，用户 mark 在其起始目录中具有 create 权限，这意味着他可以创建快照。当文件系统通过 NFS 挂载时，此方案很有用。

```
$ cd /tank/home/mark2
$ ls
$ cd .zfs
$ ls
shares snapshot
$ cd snapshot
$ ls -l
total 3
drwxr-xr-x   2 mark    staff          2 Sep 27 15:55 snap1
$ pwd
/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 定义和使用 ZFS 委托权限集

以下示例说明如何创建权限集 @myset 并针对 tank 文件系统向组 staff 授予该权限集和 rename 权限。用户 cindy 是 staff 组成员，他具有在 tank 中创建文件系统的权限。但是，用户 lp 没有在 tank 中创建文件系统的权限。

```
# 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
```

显示 ZFS 授予的权限示例

可以使用以下命令来显示权限：

```
# zfs allow dataset
```

此命令显示针对指定数据集设置或授予的权限。输出包含以下组成部分：

- 权限集
- 各个权限或创建时权限
- 本地数据集
- 本地和后代数据集
- 仅后代数据集

例 8-6 显示基本委托管理权限

以下输出表示用户 `cindy` 在 `tank/cindy` 文件系统上具有 `create`、`destroy`、`mount` 和 snapshot 权限。

```
# zfs allow tank/cindy
```

```
-------------------------------------------------------------
Local+Descendent permissions on (tank/cindy)
user cindy create,destroy,mount,snapshot
```

例 8-7 显示复杂委托管理权限

此示例中的输出表示针对 pool/fred 和 pool 文件系统的以下权限。

对于 pool/fred 文件系统：

- 定义了两个权限集：
  - @eng（create、destroy、snapshot、mount、clone、promote 和 rename）
  - @simple（create 和 mount）
- 为 @eng 权限集和 mountpoint 属性设置了创建时权限。“创建时”意味着在文件系统创建后，便授予 @eng 权限集和设置 mountpoint 属性的权限。
- 向用户 tom 授予了对本地文件系统的 @eng 权限集，向用户 joe 授予了对本地文件系统的 create、destroy 和 mount 权限。
- 向用户 fred 授予了对本地和后代文件系统的 @basic 权限集，以及 share 和 rename 权限。
- 向用户 barney 和 staff 组授予了仅针对后代文件系统的 @basic 权限集。

对于 pool 文件系统：

- 定义了权限集 @simple（创建、销毁、挂载）。
- 向组 staff 授予了对本地文件系统的 @simple 权限集。

下面是此示例的输出：

```
$ 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
```

删除 ZFS 授予的权限示例

可以使用 zfs unallow 命令删除已授予的委托权限。例如，用户 cindy 在 tank/cindy 文件系统上具有 create、destroy、mount 和 snapshot 权限。

```
# 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
```

以下 zfs unallow 语法将从 tank/home/cindy 文件系统中删除用户 cindy 的 snapshot 权限：

```
# 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
```

作为另一个示例，用户 mark 在 tank/home/mark 文件系统上具有以下权限：

```
# zfs allow tank/home/mark
---- Permissions on tank/home/mark --------------------------------
Local+Descendent permissions:
user mark create,destroy,mount
-------------------------------------------------------------
```

以下 zfs unallow 语法将从 tank/home/mark 文件系统中删除用户 mark 的所有权限：

```
# zfs unallow mark tank/home/mark
```

以下 zfs unallow 语法将删除对 tank 文件系统的权限集。

```
# 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 高级主题

本章介绍仿真卷、在安装了区域的 Solaris 系统中使用 ZFS、ZFS 备用根池以及 ZFS 权限配置文件。

本章包含以下各节：

ZFS 卷

ZFS 卷是表示块设备的数据集。ZFS 卷被标识为 `/dev/zvol/{dsk,rdsk}/pool` 目录中的设备。

以下示例将创建 5 GB 的 ZFS 卷 `tank/vol`：

```
# zfs create -V 5gb tank/vol
```

创建卷时，会自动设置卷初始大小的预留空间，以防发生意外行为。例如，如果卷大小减小，则可能导致数据受损。更改卷大小时请务必小心。

此外，如果对大小发生更改的卷创建快照，并且尝试回滚该快照或从该快照中创建克隆，则可能会引入不一致性。

有关可应用于卷的文件系统属性的信息，请参见表 5-1 “ZFS 本机属性说明”。

可以使用 `zfs get` 或 `zfs get all` 命令显示 ZFS 卷的属性信息。例如：

```
# zfs get all tank/vol
```

`zfs get` 输出中针对 `volsize` 显示的问号 (?) 表示值未知，这是因为发生了 I/O 错误。例如：

```
# zfs get -H volsize tank/vol
tank/vol        volsize ?       local
```

I/O 错误通常表示池设备有问题。有关解决池设备问题的信息，请参见“确定 ZFS 存储池的问题” [239]。

如果使用安装了区域的 Solaris 系统，则不能在非全局区域中创建或克隆 ZFS 卷。试图这样做必定会失败。有关在全局区域中使用 ZFS 卷的信息，请参见“向非全局区域中添加 ZFS 卷” [232]。

使用 ZFS 卷作为交换设备或转储设备

安装 ZFS 根文件系统或从 UFS 根文件系统迁移期间，会在 ZFS 根池中的 ZFS 卷上创建交换设备。例如：

```
# swap -l
swapfile                  dev    swaplo   blocks     free
/dev/zvol/dsk/rpool/swap 253,3        16  8257520  8257520
```

安装 ZFS 根文件系统或从 UFS 根文件系统迁移期间，会在 ZFS 根池中的 ZFS 卷上创建转储设备。转储设备在设置后便无需管理。例如：

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

如果在安装系统后需要更改交换区域或转储设备，请像在以前的 Solaris 发行版中那样使用 `swap` 和 `dumpadm` 命令。如果需要创建其他交换卷，请创建一个特定大小的 ZFS 卷，然后在该设备中启用交换。然后，在 `/etc/vfstab` 文件中为新交换设备添加一个条目。例如：

```
# 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
```

在 ZFS 文件系统中，不要交换到文件。不支持 ZFS 交换文件配置。

有关调整交换和转储卷大小的信息，请参见“调整 ZFS 交换和转储设备的大小” [102]。

将 ZFS 卷用作 iSCSI LUN

通过通用多协议 SCSI 目标 (Common Multiprotocol SCSI Target, COMSTAR) 软件框架，可以将任何 Oracle Solaris 主机转换为启动器主机可以通过存储网络访问的 SCSI 目标设备。可以创建和配置要作为 iSCSI 逻辑单元 (LUN) 共享的 ZFS 卷。

首先，安装 COMSTAR 软件包。

```
# pkg install group/feature/storage-server
```

接下来，创建要用作 iSCSI 目标的 ZFS 卷，然后创建基于 SCSI 块设备的 LUN。例如：

```
# 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
```

您可以向所有客户机或选定的客户机公开 LUN 视图。确定 LUN GUID，然后共享 LUN 视图。在以下示例中，LUN 视图由所有客户机共享。

```
# 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
```

下一步是创建 iSCSI 目标。有关创建 iSCSI 目标的信息，请参见《在 Oracle Solaris 11.2 中管理设备 》中的第 8 章 “使用 COMSTAR 配置存储设备”。

对于用作 iSCSI 目标的 ZFS 卷，其管理方式与任何其他 ZFS 数据集一样，不过，在 ZFS 卷被作为 iSCSI LUN 共享时，您无法重命名数据集、回滚卷快照或者导出池。将显示以下类似消息：

```
# 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
```

所有 iSCSI 目标配置信息都存储在数据集内。与 NFS 共享文件系统相似，在其他系统中导入的 iSCSI 目标也会相应进行共享。

在安装了区域的 Solaris 系统中使用 ZFS

以下各节介绍如何在具有 Oracle Solaris 区域的系统中使用 ZFS。

将 ZFS 数据集与区域关联时，请牢记以下要点：

- 可将 ZFS 文件系统或克隆添加至非全局区域（可以授予或不授予管理控制权）。
- 可将 ZFS 卷作为设备添加至非全局区域。
- 此时不能将 ZFS 快照与区域关联。

在以下各节中，ZFS 数据集是指文件系统或克隆。

通过添加数据集，非全局区域可与全局区域共享磁盘空间，但区域管理员不能在底层文件系统分层结构中控制属性或创建新文件系统。该操作与向区域中添加其他任何类型的文件系统相同，应该在主要目的只是为了共享公用磁盘空间时才这样做。

使用 ZFS，还可将数据集委托给非全局区域，从而授予区域管理员对数据集及其所有子项的完全控制。区域管理员可以在此数据集中创建和销毁文件系统或克隆，并可修改数据集的属性。区域管理员无法影响尚未添加到区域的数据集，也无法超出在委托数据集上设置的任何顶层配额。

在安装了 Oracle Solaris 区域的系统中使用 ZFS 时，请注意以下事项：

- 添加到非全局区域的 ZFS 文件系统必须将其 `mountpoint` 属性设置为 `legacy`。
- 当源 `zonepath` 和目标 `zonepath` 都驻留在 ZFS 文件系统上并且位于同一个池中时，`zoneadm clone` 现在将自动使用 ZFS 克隆来克隆区域。`zoneadm clone` 命令将创建源 `zonepath` 的 ZFS 快照，并设置目标 `zonepath`。不能使用 `zfs clone` 命令来克隆区域。有关更多信息，请参见《创建和使用 Oracle Solaris 区域》。

向非全局区域中添加 ZFS 文件系统

如果目标只是与全局区域共享空间，则可添加 ZFS 文件系统作为通用文件系统。添加到非全局区域的 ZFS 文件系统必须将其 `mountpoint` 属性设置为 `legacy`。例如，如果 `tank/zone/zion` 文件系统将添加到非全局区域，请按如下所示在全局区域中设置 `mountpoint` 属性：

```
# zfs set mountpoint=legacy tank/zone/zion
```

可以使用 `zonecfg` 命令的 `add fs` 子命令将 ZFS 文件系统添加到非全局区域中。

在以下示例中，全局区域中的全局区域管理员会向非全局区域中添加一个 ZFS 文件系统：

```
# 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
```

此语法将 ZFS 文件系统 tank/zone/zion 添加到已配置的 zion 区域（挂载在 /opt/data）。文件系统的 mountpoint 属性必须设置为 legacy，并且该文件系统不能已在其他位置挂载。区域管理员可在文件系统中创建和销毁文件。不能在其他位置重新挂载文件系统，区域管理员也不能更改该文件系统的属性，如 atime、readonly、compression 等。

全局区域管理员负责设置和控制文件系统的属性。

有关 zonecfg 命令以及有关使用 zonecfg 配置资源类型的更多信息，请参见《创建和使用 Oracle Solaris 区域》。

将数据集委托给非全局区域

为实现将存储管理委托给区域的主要目标，ZFS 支持通过使用 zonecfg 命令的 add dataset 子命令将数据集添加到非全局区域。

在以下示例中，全局区域中的全局区域管理员会将一个 ZFS 文件系统委托给非全局区域。

```
# 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
```

与添加文件系统不同，此语法会使 ZFS 文件系统 tank/zone/zion 在已配置的 zion 区域中可见。在 zion 区域内，此文件系统不可被作为 tank/zone/zion 进行访问，但是可被作为名为 tank 的虚拟池进行访问。委托的文件系统别名以虚拟池的形式向区域提供原始池的视图。别名属性指定虚拟池的名称。如果未指定别名，则使用与文件系统名称的最后一个组件匹配的缺省别名。如果未提供具体的别名，则上述示例中的缺省别名为 zion。

在委托的数据集内，区域管理员可以设置文件系统属性，还可以创建后代文件系统。此外，区域管理员还可以创建快照和克隆，或者控制整个文件系统分层结构。如果在委托的文件系统内创建 ZFS 卷，则它们可能会与作为设备资源添加的 ZFS 卷相冲突。有关更多信息，请参见下一节。

向非全局区域中添加 ZFS 卷

可以通过以下方式，在非全局区域中添加或创建 ZFS 卷，或者在非全局区域中添加对卷数据的访问权限：

- 在非全局区域中，特权区域管理员可以创建 ZFS 卷，将其作为以前委托的文件系统的后代。例如：

  ```
  # zfs create -V 2g tank/zone/zion/vol1
  ```

  以上语法意味着，区域管理员可以在非全局区域中管理卷的属性和数据。
- 在全局区域中，使用 `zonecfg add dataset` 子命令并指定要添加到非全局区域的 ZFS 卷。例如：

  ```
  # zonecfg -z zion
  zonecfg:zion> add dataset
  zonecfg:zion:dataset> set name=tank/volumes/vol1
  zonecfg:zion:dataset> end
  ```

  以上语法意味着，区域管理员可以在非全局区域中管理卷的属性和数据。
- 在全局区域中，使用 `zonecfg add device` 子命令并指定可以在非全局区域中访问其数据的 ZFS 卷。例如：

  ```
  # zonecfg -z zion
  zonecfg:zion> add device
  zonecfg:zion:device> set match=/dev/zvol/dsk/tank/volumes/vol2
  zonecfg:zion:device> end
  ```

  以上语法意味着，在非全局区域中只能访问卷数据。

在区域中使用 ZFS 存储池

不能在区域中创建或修改 ZFS 存储池。委托的管理模型可将对全局区域内的物理存储设备的控制以及对虚拟存储的控制集中到非全局区域。尽管可向区域中添加池级别数据集，但区域内不允许使用用于修改该池的物理特征的任何命令，如创建、添加或删除设备。即使使用 `zonecfg` 命令的 `add device` 子命令向区域中添加物理设备或是已使用了文件，`zpool` 命令也不允许在该区域内创建任何池。

在区域内管理 ZFS 属性

将数据集委托给区域后，区域管理员便可控制特定的数据集属性。将数据集委托给区域后，该数据集的所有祖先都显示为只读数据集，而该数据集本身则与其所有后代一样是可写的。例如，考虑以下配置：

```
global# zfs list -Ho name
tank
tank/home
tank/data
tank/data/matrix
tank/data/zion
tank/data/zion/home
```

如果将 tank/data/zion 添加到具有缺省 zion 别名的区域中，则每个数据集都将具有以下属性。

数据集	可见	可写	不变属性
tank	否	-	-
tank/home	否	-	-
tank/data	否	-	-
tank/data/zion	是	是	zoned、quota、reservation
tank/data/zion/home	是	是	zoned

请注意，tank/zone/zion 的每个父项均不可见，所有后代均可写。区域管理员不能更改 zoned 属性，否则将产生下节介绍的安全风险。

区域中的特权用户可以更改除 quota 和 reservation 之外的任何属性。全局区域管理员借助此行为可以控制非全局区域所使用的所有数据集的磁盘空间占用情况。

此外，将数据集委托给非全局区域后，全局区域管理员便无法更改 share.nfs 和 mountpoint 属性。

了解 zoned 属性

将数据集委托给非全局区域时，必须对该数据集进行特殊标记，以便特定属性不使用全局区域上下文进行解释。将数据集委托给受区域管理员控制的非全局区域之后，便不能再信任其内容。与任何文件系统一样，可能存在 setuid 二进制命令、符号链接或可能对全局区域的安全性造成不利影响的可疑内容。此外，不能在全局区域的上下文中解释 mountpoint 属性。否则，区域管理员可能会影响全局区域的名称空间。为解决后一个问题，ZFS 使用 zoned 属性来指示已在某一时刻将数据集委托给非全局区域。

zoned 属性是在首次引导包含 ZFS 数据集的区域时自动启用的布尔值。区域管理员将无需手动启用此属性。如果设置了 zoned 属性，则无法在全局区域中挂载或共享数据集。以下示例将 tank/zone/zion 委托给一个区域，tank/zone/global 则没有委托：

```
# 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
```

请注意 mountpoint 属性与当前挂载 tank/zone/zion 数据集的目录之间的差异。mountpoint 属性反映的是磁盘上存储的属性，而不是数据集当前在系统中的挂载位置。

从区域中删除数据集或销毁区域时，不会自动清除 zoned 属性。此行为是由与这些任务关联的固有安全风险引起的。由于不受信任的用户已取得对数据集及其后代的完全访问权限，因此 mountpoint 属性可能会设置为错误值，或者文件系统中可能存在 setuid 二进制命令。

为了防止意外的安全风险，要通过任何方式重用数据集时，必须由全局区域管理员手动清除 zoned 属性。将 zoned 属性设置为 off 之前，请确保数据集及其所有后代的 mountpoint 属性已设置为合理值并且不存在 setuid 二进制命令，或禁用 setuid 属性。

确定没有任何安全漏洞后，即可使用 zfs set 或 zfs inherit 命令禁用 zoned 属性。如果在区域正在使用数据集时禁用 zoned 属性，则系统的行为方式可能无法预测。仅当确定非全局区域不再使用数据集时，才能更改该属性。

将区域复制到其他系统

需要将一个或多个区域迁移到其他系统时，可考虑使用 zfs send 和 zfs receive 命令。根据具体情况，可能使用复制流最好，也可能使用递归流最好。

本节中的示例介绍了如何在系统之间复制区域数据。需要执行其他步骤来传输每个区域的配置并将每个区域附加到新系统。有关更多信息，请参见《创建和使用 Oracle Solaris 区域》。

如果一个系统上的所有区域都需要移动到另一系统，请考虑使用复制流，因为它可保留快照和克隆。快照和克隆由 pkg update、beadm create 和 zoneadm clone 命令广泛使用。

在以下示例中，sysA 的区域安装在 rpool/zones 文件系统中，需要将它们复制到 sys 上的 tank/zones 文件系统。以下命令创建一个快照并通过使用复制流将数据复制到 sysB：

```
sysA# zfs snapshot -r rpool/zones@send-to-sysB
sysA# zfs send -R rpool/zones@send-to-sysB | ssh sysB zfs receive -d tank
```

在以下示例中，若干个区域中的一个从 sysC 复制到 sysD。假定 ssh 命令不可用，但 NFS 服务器实例可用。以下命令可以用于生成递归的 zfs send 流，而不必担心区域是否是另一个区域的克隆。

```
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
```

通过备用根位置使用 ZFS 池

创建池时，该池将固定绑定到主机系统。主机系统一直掌握着池的状况信息，以便可以检测到池何时不可用。此信息虽然对于正常操作很有用，但在从备用介质引导或在可移除介质上创建池时则会成为障碍。为解决此问题，ZFS 提供了备用根位置池功能。系统重新引导之后备用根池位置不会保留，并且所有挂载点都会修改为相对于该池的根。

使用备用根位置创建 ZFS 池

在备用位置创建池的最常见目的是为了与可移除介质结合使用。在这些情况下，用户通常需要一个单独的文件系统，并且希望在目标系统中选择的任意位置挂载该系统。如果使用 `zpool create -R` 选项创建一个池，根文件系统的挂载点自动设为 `/`，这相当于备用根值。

在以下示例中，使用 `/mnt` 备用根位置创建了名为 `morpheus` 的池：

```
# zpool create -R /mnt morpheus c0t0d0
# zfs list morpheus
NAME                  USED  AVAIL  REFER  MOUNTPOINT
morpheus             32.5K  33.5G     8K  /mnt
```

请注意单个文件系统 `morpheus`，其挂载点是池的备用根位置 `/mnt`。存储在磁盘上的挂载点是 `/`，`/mnt` 的全路径仅在池创建这一初始上下文中进行解释。然后可以使用 `-R` 备用根值语法在不同系统的任意备用根位置下导出和导入该文件系统。

```
# 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
```

使用备用根位置导入池

也可以使用备用根位置来导入池。此功能可用于恢复，在这种情况下挂载点不是在当前根挂载点上下文中解释，而是在可以执行修复的某个临时目录下解释。挂载可移除介质时也可以使用此功能，如上一节所述。

在以下示例中，使用 /mnt 备用根挂载点导入了名为 morpheus 的池：本示例假定之前已导出了 morpheus。

```
# 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
```

使用临时名称导入池

除了在备用根位置导入池外，您可以使用临时名称导入池。在某些共享存储或恢复场景中，此功能允许同时导入具有相同持久名称的两个池。其中一个池必须使用临时名称导入。

在下例中，在备用根位置使用临时名称导入 rpool 池。因为持久池名称与已经导入的一个池冲突，必须按照池 ID 或通过指定设备来导入该池。

```
# zpool import
pool: rpool
id: 16760479674052375628
state: ONLINE
action: The pool can be imported using its name or numeric identifier.
config:

rpool     ONLINE
c8d1s0  ONLINE
# zpool import -R /a -t altrpool 16760479674052375628
# zpool list
NAME      SIZE  ALLOC  FREE  CAP  DEDUP  HEALTH  ALTROOT
altrpool   97G  22.4G   74G  23%  1.00x  ONLINE  /a
rpool     465G  75.1G  390G  16%  1.00x  ONLINE  -
```

通过使用 zpool create -t 选项，也可以使用临时名称创建一个池。

♦ ♦ ♦ 第 10 章

10 Oracle Solaris ZFS 故障排除和池恢复

本章介绍如何确定 ZFS 故障以及如何从相应故障中恢复。还提供了有关预防故障的信息。

本章包含以下各节：

有关完整根池恢复的信息，请参见《在 Oracle Solaris 11.2 中使用统一归档文件进行系统恢复和克隆 》。

确定 ZFS 问题

作为组合的文件系统和卷管理器，ZFS 可以呈现许多不同的故障。本章概述了如何诊断一般的硬件故障以及如何解决池设备和文件系统问题。您可能会遇到以下类型的问题：

- **一般硬件问题** – 硬件问题可能会影响池的性能和池数据的可用性。在确定更高层（例如池和文件系统）的问题之前，请排除一般的硬件问题，例如有故障的组件和内存。
- **ZFS 存储池问题**
  - “确定 ZFS 存储池的问题” [239]
  - “解决 ZFS 存储设备问题” [244]
- **数据损坏** – “解决 ZFS 存储池中的数据问题” [257]
- **配置损坏** – “修复损坏的 ZFS 配置” [266]
- **系统无法引导** – “修复无法引导的系统” [266]

请注意，单个池可能会遇到所有这三种错误，因此完整的修复过程依次查找和更正各个错误。

解决一般的硬件问题

请查看以下各节来确定池问题或文件系统不可用是否与硬件问题（例如有故障的系统板、内存、设备、HBA 或错误配置）相关。

例如，在一个繁忙的 ZFS 池上，将要发生故障或已经发生故障的磁盘可能会大大降低总体系统性能。

如果先从诊断和确定硬件问题开始，这些问题比较容易检测，在检查完所有硬件后，您可以按本章剩余部分中所述继续对池和文件系统问题进行诊断。如果硬件、池和文件系统配置都正常，请考虑诊断应用程序问题，这类问题的解决通常比较复杂，本指南中未涵盖这方面的内容。

确定硬件和设备故障

Solaris Fault Manager 通过以下方式来跟踪软件、硬件和特定的设备问题：在错误日志中标识指明特定症状的错误遥测信息，然后在错误症状导致了实际故障时报告实际的故障诊断信息。

以下命令用于确定任何与软件或硬件相关的故障。

```
# fmadm faulty
```

可例行使用以上命令来确定发生故障的服务或设备。

可例行使用以下命令来确定与硬件或设备相关的错误。

```
# fmdump -eV | more
```

需要注意此日志文件中描述 `vdev.open_failed`、`checksum` 或 `io_failure` 问题的错误消息，否则它们可能会演变为实际错误（可通过 `fmadm` 故障命令显示）。

如果以上信息指明某个设备将要发生故障，则正好趁此时确保有可替换的设备。

还可以通过使用 `iostat` 命令来跟踪额外的设备错误。使用以下语法可标识错误统计信息摘要。

```
# iostat -en
---- errors ---
s/w h/w trn tot device
0   0   0   0 c0t5000C500335F95E3d0
0   0   0   0 c0t5000C500335FC3E7d0
0   0   0   0 c0t5000C500335BA8C3d0
0  12   0  12 c2t0d0
0   0   0   0 c0t5000C500335E106Bd0
0   0   0   0 c0t50015179594B6F11d0
0   0   0   0 c0t5000C500335DC60Fd0
0   0   0   0 c0t5000C500335F907Fd0
0   0   0   0 c0t5000C500335BD117d0
```

在上面的输出中，报告了内部磁盘 c2t0d0 的错误。使用以下语法可显示更详细的设备错误。

解决持久性或瞬态传输错误

固件版本低、磁盘损坏、电缆损坏或硬件连接故障可能导致需要重试或重置的持久性 SCSI 传输错误。可以通过升级您的 HBA 或设备固件解决部分瞬态传输错误。升级固件并确认所有设备都运行正常后，如果传输错误仍然存在，则应检查硬件组件是否存在电缆损坏或其他连接故障。

ZFS 错误消息的系统报告

除了持久跟踪池中的错误外，ZFS 还在发生相关事件时显示 syslog 消息。以下情况将生成通知事件：

- **设备状态转换** – 如果设备变为 FAULTED 状态，则 ZFS 将记录一条消息，指出池的容错能力可能已受到危害。如果稍后将设备联机，将池恢复正常，则将发送类似的消息。
- **数据损坏** – 如果检测到任何数据损坏，则 ZFS 将记录一条消息，描述检测到数据损坏的时间和位置。仅在首次检测到数据损坏时才记录此消息。后续访问不生成消息。
- **池故障和设备故障** – 如果出现池故障或设备故障，则 Fault Manager 守护进程将通过 syslog 消息以及 fmdump 命令报告这些错误。

如果 ZFS 检测到设备错误并自动从其恢复，则不进行通知。这样的错误不会造成池冗余或数据完整性方面的故障。并且，这样的错误通常是由伴随有自己的一组错误消息的驱动程序问题导致的。

确定 ZFS 存储池的问题

以下各节介绍如何确定并解决 ZFS 文件系统或存储池中的问题：

- “确定 ZFS 存储池中是否存在问题” [240]
- “查看 ZFS 存储池状态信息” [241]
- “ZFS 错误消息的系统报告” [239]

可以使用以下功能来确定 ZFS 配置所存在的问题：

- 使用 zpool status 命令可以显示 ZFS 存储池的详细信息。
- 通过 ZFS/FMA 诊断消息报告池和设备故障。
- 使用 zpool history 命令可以显示以前修改了池状态信息的 ZFS 命令。

- 使用 `zpool import -D` 命令可以恢复意外销毁的 ZFS 存储池，但是必须快速恢复该池，否则可能会重用或意外覆盖该池。有关更多信息，请参见“恢复已销毁的 ZFS 存储池” [83]。没有类似功能可以恢复 ZFS 文件系统或数据。务必保留高质量备份。

大多数 ZFS 故障排除工作都会涉及到 `zpool status` 命令。此命令对系统中的各种故障进行分析并确定最严重的问题，同时为您提供建议的操作和指向知识文章（用于获取更多信息）的链接。请注意，虽然池可能存在多个问题，但是此命令仅确定其中的一个问题。例如，数据损坏错误一般意味着一台设备发生故障，但更换故障设备可能无法解决所有数据损坏问题。

此外，ZFS 诊断引擎也会诊断和报告池故障和设备故障。另外还会报告与这些故障关联的校验和、I/O、设备和池错误。`fmd` 报告的 ZFS 故障在控制台上以及系统消息文件中显示。在大多数情况下，`fmd` 消息会指示您查看 `zpool status` 命令的输出，以便获得进一步的恢复说明。

基本的恢复过程如下所示：

- 如果合适，请使用 `zpool history` 命令错误情况出现以前的 ZFS 命令。例如：

```
# zpool history tank
History for 'tank':
2012-11-12.13:01:31 zpool create tank mirror c0t1d0 c0t2d0 c0t3d0
2012-11-12.13:28:10 zfs create tank/eric
2012-11-12.13:37:48 zfs set checksum=off tank/eric
```

  在此输出中可以看到，对 `tank/eric` 文件系统禁用了校验和。建议不要使用此配置。
- 通过在系统控制台上或 `/var/adm/messages` 文件中显示的 `fmd` 消息来确定错误。
- 使用 `zpool status -x` 命令查找进一步的修复说明。
- 排除故障涉及到以下步骤：
  - 更换不可用设备或缺少的设备，并使其联机。
  - 从备份恢复故障配置或损坏的数据。
  - 使用 `zpool status -x` 命令验证恢复情况。
  - 备份所恢复的配置（如果适用）。

本节介绍如何解读 `zpool status` 输出，以便诊断可能出现的故障类型。尽管大多数工作是由命令自动执行的，但是准确了解所确定的问题以便诊断故障是很重要的。后续部分将介绍如何解决可能遇到的各种问题。

确定 ZFS 存储池中是否存在问题

确定系统上是否存在任何已知问题的最简单的方法是使用 `zpool status -x` 命令。此命令仅对出现问题的池进行说明。如果系统中不存在运行状态不佳的池，该命令将显示以下信息：

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

如果不带 -x 标志，该命令显示所有池（如果在命令行中指定，则为所请求的池）的完成状态，即使这些池运行状态良好。

有关 zpool status 命令的命令行选项的更多信息，请参见“查询 ZFS 存储池的状态” [64]。

查看 ZFS 存储池状态信息

使用 zpool status 命令显示 ZFS 存储池状态信息。例如：

```
# 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
```

该输出将在下一节中进行介绍。

总体池状态信息

zpool status 输出中的这一部分包含以下字段（其中一些字段仅出现问题的池才会显示）：

pool　　标识池的名称。

state　　指示池的当前运行状况。此信息仅指池提供必要复制级别的能力。

`status`	描述池存在的问题。如果未发现错误，则省略此字段。
`action`	建议用于修复错误的操作。如果未发现错误，则省略此字段。
`see`	对包含详细修复信息的知识文章的引用。在线文章的更新频率高于本指南的更新频率。因此，务必参考这些文章以了解最新的修复程序。如果未发现错误，则省略此字段。
`scrub`	确定清理操作的当前状态，它可能包括完成上一清理的日期和时间、正在进行的清理或者是否未请求清理。
`errors`	确定是否存在已知的数据错误。

ZFS 存储池配置信息

`zpool status` 输出中的 `config` 字段描述池中的设备的配置，以及设备的状态和设备产成的任何错误。其状态可以是以下状态之一：`ONLINE`、`FAULTED`、`DEGRADED` 或 `SUSPENDED`。如果状态是除 `ONLINE` 之外的任何状态，则说明池的容错能力已受到损害。

配置输出的第二部分显示错误统计信息。这些错误分为以下三类：

- `READ` – 发出读取请求时出现 I/O 错误
- `WRITE` – 发出写入请求时出现 I/O 错误
- `CKSUM` – 校验和错误，意味着设备对读取请求返回损坏的数据

这些错误可用于确定损坏是否是永久性的。小量 I/O 错误数可能指示临时故障，而大量 I/O 错误则可能指示设备出现了永久性问题。这些错误不一定对应于应用程序所解释的数据损坏。如果设备处于冗余配置中，则设备可能显示无法更正的错误，而镜像或 RAID-Z 设备级别上不显示错误。这种情况下，ZFS 成功检索到良好的数据并试图利用现有副本修复受损数据。

有关解释这些错误的更多信息，请参见“确定设备故障的类型” [248]。

最后，在 `zpool status` 输出的最后一列中显示其他辅助信息。此信息是对 `state` 字段的详述，以帮助诊断故障。如果设备处于 `UNAVAIL` 状态，则此字段指示是否无法访问设备或者设备上的数据是否已损坏。如果设备正在进行重新同步，则此字段显示当前的进度。

有关监视重新同步进度的信息，请参见“查看重新同步状态” [255]。

ZFS 存储池清理状态

`zpool status` 输出的 scrub 部分描述任何清理操作的当前状态。此信息不是用于指示系统上是否检测到任何错误，但是可以利用此信息来判定数据损坏错误报告的准确性。如果上一清理是最近结束的，则很可能已发现任何已知的数据损坏。

提供了以下 zpool status 清理状态消息：

- 清理进度报告。例如：

```
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
```

- 清理完成消息。例如：

```
scan: scrub repaired 0 in 0h11m with 0 errors on Wed Jun 20 15:08:23 2012
```

- 取消正在进行的清理消息。例如：

```
scan: scrub canceled on Wed Jun 20 16:04:40 2012
```

清理完成消息可在系统重新引导后存留下来。

有关数据清理以及如何解释此信息的更多信息，请参见“检查 ZFS 文件系统完整性” [259]。

ZFS 数据损坏错误

zpool status 命令还显示是否有已知错误与池关联。在数据清理或常规操作期间，可能已发现这些错误。ZFS 将与池关联的所有数据错误记录在持久性日志中。每当系统的完整清理完成时，都会轮转此日志。

数据损坏错误始终是致命的。出现这种错误表明至少一个应用程序因池中的数据损坏而遇到 I/O 错误。冗余池中的设备错误不会导致数据损坏，而且不会被记录在此日志中。缺省情况下，仅显示发现的错误数。使用 zpool status -v 选项可以列出带有详细说明的完整错误列表。例如：

```
# 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
```

也可使用 fmd 在系统控制台上和 /var/adm/messages 文件中显示类似的消息。还可以使用 fmdump 命令跟踪这些消息。

有关解释数据损坏错误的更多信息，请参见“确定数据损坏的类型” [262]。

解决 ZFS 存储设备问题

请查看以下各节来解决缺少设备、设备被移除或发生故障等问题。

解决缺少设备或设备被移除的问题

如果设备无法打开，则它在 zpool status 输出中显示为 UNAVAIL状态。此状态表示在首次访问池时 ZFS 无法打开设备，或者设备自那时以来已变得不可用。如果设备导致顶层虚拟设备不可用，则无法访问池中的任何内容。此外，池的容错能力可能已受到损害。无论哪种情况，只需要将设备重新附加到系统即可恢复正常操作。如果需要替换因发生故障而处于 UNAVAIL 状态的设备，请参见“替换 ZFS 存储池中的设备” [250]。

如果根池或镜像的根池中的某个设备状态为 UNAVAIL，请参见以下参考资料：

- **镜像根池磁盘发生故障** – “从镜像 ZFS 根池中的备用磁盘引导” [104]
- **替换根池中的磁盘**
  - 如何替换 ZFS 根池中的磁盘（SPARC 或 x86/EFI (GPT)） [98]
  - 如何替换 ZFS 根池中的磁盘（SPARC 或 x86/EFI (GPT)） [98]
- **完整的根池灾难恢复** – 《在 Oracle Solaris 11.2 中使用统一归档文件进行系统恢复和克隆 》。

例如，设备出现故障后，可能会在 fmd 的输出中看到与以下内容类似的消息：

```
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.
```

要查看有关设备问题和解决方法的更多详细信息，请使用 zpool status -v 命令。例如：

```
# 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
```

从此输出中可以看到，设备 `c0t5000C500335DC60Fd0` 不起作用。如果您确定该设备有问题，请予以替换。

如有必要，可使用 `zpool online` 命令使替换的设备联机。例如：

```
# zpool online pond c0t5000C500335DC60Fd0
```

如果 `fmadm faulty` 的输出标识出设备错误，请让 FMA 知道该设备已被替换。例如：

```
# 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.
```

摘取 `fmadm faulty` 命令输出中 `Affects:` 部分的字符串，并将其包含在以下命令中，以便通知 FMA 该设备已替换：

```
# 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
```

最后一步是确认设备更换后的池正常运行。例如：

```
# zpool status -x tank
pool 'tank' is healthy
```

解决设备被移除的问题

如果某个设备已从系统中彻底删除，则 ZFS 会检测到该设备无法打开，并将其置于 `REMOVED` 状态。这一删除可能会导致整个池变得不可用，但也可能不会，具体取决于池的数据复制级别。如果镜像设备或 RAID-Z 设备中的一个磁盘被删除，仍可以继续访问池。在下列情况下，池可能会变为 `UNAVAIL`（即无法访问数据，除非重新附加设备）：

如果意外移除并重新插入了冗余存储池设备，那么大多数情况下您只需清除设备错误即可。例如：

```
# zpool clear tank c1t1d0
```

以物理方式重新附加设备

重新附加缺少的设备的具体方式取决于相关设备。如果设备是网络连接驱动器，则应该恢复与网络的连接。如果设备是 USB 设备或其他可移除介质，则应该将它重新附加到系统。如果设备是本地磁盘，则控制器可能已出现故障，以致设备对于系统不再可见。在这种情况下，应该替换控制器，以使磁盘重新可用。可能存在其他问题，具体取决于硬件的类型及其配置。如果驱动器出现故障，且对系统不再可见，则应该将该设备视为损坏的设备。按照“更换或修复损坏的设备” [248]中概述的过程进行操作。

如果设备连接受到损害，池可能变为 SUSPENDED 状态。在设备问题得到解决之前，SUSPENDED 池一直处于 wait 状态。例如：

```
# 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
```

当设备连接恢复后，请清除池或设备错误。

```
# zpool clear cybermen
# fmadm repaired zfs://pool=name/vdev=guid
```

将设备可用性通知 ZFS

将设备重新附加到系统后，ZFS 可能会也可能不会自动检测其可用性。如果池先前为 UNAVAIL 或 SUSPENDED 状态，或者在执行 attach 的过程中系统进行了重新引导，则 ZFS 在尝试打开该池时，会自动重新扫描所有设备。如果在系统运行时池的性能降低且设备已替换，则必须通知 ZFS 设备现在是可用的并可以使用 zpool online 命令重新打开。例如：

```
# zpool online tank c0t1d0
```

有关使设备联机的更多信息，请参见“使设备联机” [54]。

更换或修复损坏的设备

本节介绍如何确定设备故障类型、清除瞬态错误和替换设备。

确定设备故障的类型

术语损坏的设备相当含糊，它可以用来描述许多可能的情况：

- **位损坏** – 随着时间的推移，随机事件（如电磁感应和宇宙射线）可能会导致存储在磁盘上的位发生翻转。这些事件相对少见，但是通常足以导致大系统或长时间运行的系统出现潜在的数据损坏。
- **误导的读取或写入** – 固件已知问题或硬件故障可以导致整个块的读取或写入引用磁盘上的不正确位置。这些错误通常是瞬态的，尽管大量此类错误可能指示驱动器有故障。
- **管理员错误** – 管理员可能无意中用错误的数据覆盖了部分磁盘（如在部分磁盘上复制 `/dev/zero`），从而导致磁盘上出现永久性损坏。这些错误始终是瞬态的。
- **临时故障** – 磁盘可能在某段时间内变得不可用，从而导致 I/O 失败。此情况通常与网络连接设备相关联，尽管本地磁盘也可能遇到临时故障。这些错误可能是也可能不是瞬态的。
- **劣质或不可靠的硬件** – 这种情况涵盖故障硬件表现出来的所有各种问题，包括一致的 I/O 错误、故障传输导致随机损坏或任何数量的故障。这些错误通常是永久性的。
- **脱机的设备** – 如果设备处于脱机状态，则假定是管理员因该设备有故障而将它置于此状态。将设备置于此状态的管理员可以确定此假定是否正确。

准确确定设备的问题可能是一个很困难的过程。第一步是检查 `zpool status` 输出中的错误计数。例如：

```
# 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
```

错误分为 I/O 错误与校验和错误，这两种错误都指示可能的故障类型。典型操作可预知的错误数非常少（在很长一段时间内只能预知几个错误）。如果看到大量的错误，则此情况可能指示即将出现或已出现设备故障。然而，管理员错误也可能导致错误计数较大。另一信息源是 `syslog` 系统日志。如果日志显示大量的 SCSI 或光纤通道驱动程序消息，则此情况可能指示出现了严重的硬件问题。如果未生成 `syslog` 消息，则损坏很可能是瞬态的。

目的是回答以下问题：

此设备上是否可能出现另一错误？

仅出现一次的错误被认为是瞬态的，不指示存在潜在的故障。其持久性或严重性足以指明潜在硬件故障的错误被认为是致命的。由于确定错误类型的行为已超出当前可用于 ZFS 的任何自动化软件的功能范围，因此如此多的操作必须由您（即管理员）手动执行。在确定后，可以执行相应的操作。清除瞬态错误，或者替换出现致命错误的设备。以下几节将介绍这些修复过程。

即使设备错误被认为是瞬态的，仍然可能导致池中出现了无法更正的数据错误。这些错误需要特殊的修复过程，即使认为底层设备运行状况良好或已进行修复也是如此。有关修复数据错误的更多信息，请参见“修复损坏的 ZFS 数据” [261]。

清除瞬态或持久性设备错误

如果认为设备错误是瞬态的（因为它们不大可能影响设备将来的运行状况），则可以安全地清除设备错误，以指示未出现致命错误。要将 RAID-Z 或镜像设备的错误计数器清零，请使用 `zpool clear` 命令。例如：

```
# zpool clear tank c1t1d0
```

此语法清除所有设备错误，并清除与设备关联的任何数据错误计数。

要清除与池中虚拟设备关联的所有错误，并清除与池关联的任何数据错误计数，请使用以下语法：

```
# zpool clear tank
```

有关清除池错误的更多信息，请参见“清除存储池设备错误” [55]。

使用 `zpool clear` 命令就可以清除瞬态设备错误。如果设备发生故障，请参见下一节了解如何替换设备。如果某个冗余设备被意外覆盖或长时间处于 UNAVAIL 状态，则可能需要按照 `zpool status` 输出结果的指导使用 `fmadm repaired` 命令解决此错误。例如：

```
# 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
```

替换 ZFS 存储池中的设备

如果设备损坏是永久性的，或者将来很可能出现永久性损坏，则必须替换该设备。是否可以替换设备取决于配置。

确定是否可以替换设备

如果要替换的设备是冗余配置的一部分，则必须存在可以从其中检索正确数据的足够副本。例如，如果在一个四向镜像中有两个磁盘处于 UNAVAIL 状态，则可以替换其中任何一个磁盘（因为有运行状况良好的副本可用）。但是，如果四向 RAID-Z (raidz1) 虚拟设备中有两个磁盘为 UNAVAIL 状态，则这两个磁盘都不能替换，因为不存在可从其中检索数据的足够副本。如果设备已损坏但处于联机状态，则只要池不处于 UNAVAIL 状态就可以替换它。但是，除非存在足够多的包含正确数据的副本，否则会将设备上的损坏数据复制到新设备。

在以下配置中，可以替换磁盘 c1t1d0，而且池中的数据将从完好的副本 c1t0d0 复制得到：

```
    mirror            DEGRADED
c1t0d0              ONLINE
c1t1d0              UNAVAIL
```

虽然因没有可用的正确副本而无法对数据进行自我修复，但还是可以替换磁盘 c1t0d0。

在以下配置中，无法替换任一 UNAVAIL 磁盘。也无法替换 ONLINE 磁盘，因为池本身为 UNAVAIL 状态。

```
    raidz1              UNAVAIL
c1t0d0              ONLINE
c2t0d0              UNAVAIL
c3t0d0              UNAVAIL
c4t0d0              ONLINE
```

在以下配置中，尽管已将磁盘上存在的错误数据复制到新磁盘，但是任一顶层磁盘都可替换。

```
c1t0d0         ONLINE
c1t1d0         ONLINE
```

如果任一个磁盘为 UNAVAIL 状态，则无法执行任何替换操作，因为池本身为 UNAVAIL 状态。

无法替换的设备

如果设备缺失导致池变为 UNAVAIL 状态，或者设备在非冗余配置中包含太多的数据错误，则无法安全地替换设备。如果没有足够的冗余，则不存在可用来恢复损坏设备的正确数据。这种情况下，唯一的选择是销毁池并重新创建配置，然后从备份副本恢复数据。

有关恢复整个池的更多信息，请参见“修复 ZFS 存储池范围内的损坏” [264]。

替换 ZFS 存储池中的设备

确定可以替换设备后，可以使用 zpool replace 命令替换设备。如果要用不同的设备替换损坏的设备，请使用类似以下的语法：

```
# zpool replace tank c1t1d0 c2t0d0
```

此命令将数据从损坏的设备或从池中的其他设备（如果处于冗余配置中）迁移到新设备。此命令完成后，将从配置中拆离损坏的设备，此时可以将该设备从系统中移除。如果已移除设备并在同一位置中将它替换为新设备，请使用命令的单设备形式。例如：

```
# zpool replace tank c1t1d0
```

此命令接受未格式化的磁盘，适当地将它格式化，然后重新同步其余配置中的数据。

有关 zpool replace 命令的更多信息，请参见“替换存储池中的设备” [55]。

例 10-1 替换 ZFS 存储池中的 SATA 磁盘

以下示例展示了如何将系统上的镜像存储池 tank 中的设备 (c1t3d0) 替换为 SATA 设备。要在同一位置将磁盘 c1t3d0 替换为新磁盘 (c1t3d0)，尝试替换磁盘之前必须取消磁盘配置。如果要替换的磁盘不是 SATA 磁盘，则请参见“替换存储池中的设备” [55]。

基本步骤如下：

- 使要替换的磁盘 (c1t3d0) 脱机。您不能取消配置当前正在使用的 SATA 磁盘。
- 使用 cfgadm 命令确定要取消配置的 SATA 磁盘 (c1t3d0) 并取消其配置。如果磁盘在此镜像配置中脱机，该池将降级，但该池将继续可用。
- 物理替换磁盘 (c1t3d0)。在物理移除 UNAVAIL 状态的驱动器（如果有的话）之前，请确保蓝色的可以移除 LED 指示灯亮起。
- 重新配置 SATA 磁盘 (c1t3d0)。
- 使新磁盘 (c1t3d0) 联机。
- 运行 zpool replace 命令以替换磁盘 (c1t3d0)。

注 - 如果先前将池属性 autoreplace 设置为 on，则会自动对在先前属于池的设备所在物理位置处找到的任何新设备进行格式化和替换，而无需使用 zpool replace 命令。此功能可能并不是在所有硬件上都受支持。

- 如果已使用热备件自动替换了故障磁盘，则您可能需要在替换故障磁盘后分离该热备件。例如，如果替换故障磁盘后，c2t4d0 仍为活动热备件，则对其进行分离。

  ```
  # zpool detach tank c2t4d0
  ```

- 如果 FMA 报告了有故障的设备，您应当清除设备故障。

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

以下示例分步显示了替换 ZFS 存储池中的磁盘的过程。

```
# zpool offline tank c1t3d0
# cfgadm | grep c1t3d0
sata1/3::dsk/c1t3d0            disk         connected    configured   ok
# cfgadm -c unconfigure sata1/3
Unconfigure the device at: /devices/pci@0,0/pci1022,7458@2/pci11ab,11ab@1:3
This operation will suspend activity on the SATA device
Continue (yes/no)? yes
# cfgadm | grep sata1/3
sata1/3                        disk         connected    unconfigured ok
<Physically replace the failed disk c1t3d0>
# cfgadm -c configure sata1/3
# cfgadm | grep sata1/3
sata1/3::dsk/c1t3d0            disk         connected    configured   ok
# zpool online tank c1t3d0
```

```
# zpool replace tank c1t3d0
# zpool status tank
pool: tank
state: ONLINE
scrub: resilver completed after 0h0m with 0 errors on Tue Feb  2 13:17:32 2010
config:

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

请注意，上述 `zpool output` 可能会在 *replacing* 标题下显示新磁盘和旧磁盘。例如：

```
replacing     DEGRADED     0     0    0
c1t3d0s0/o  FAULTED      0     0    0
c1t3d0      ONLINE       0     0    0
```

此文本表示替换过程正在进行，且新磁盘正在重新同步。

如果您打算用一个磁盘 (`c4t3d0`) 替换另一个磁盘 (`c1t3d0`)，则只需运行 `zpool replace` 命令。例如：

```
# zpool replace tank c1t3d0 c4t3d0
# zpool status
pool: tank
state: DEGRADED
scrub: resilver completed after 0h0m with 0 errors on Tue Feb  2 13:35:41 2010
config:

NAME       STATE     READ WRITE CKSUM
tank       DEGRADED     0     0     0
mirror-0   ONLINE       0     0     0
c0t1d0     ONLINE       0     0     0
c1t1d0     ONLINE       0     0     0
mirror-1   ONLINE       0     0     0
c0t2d0     ONLINE       0     0     0
c1t2d0     ONLINE       0     0     0
mirror-2   DEGRADED     0     0     0
c0t3d0     ONLINE       0     0     0
replacing  DEGRADED     0     0     0
c1t3d0     OFFLINE      0     0     0
c4t3d0     ONLINE       0     0     0

errors: No known data errors
```

磁盘替换完成之前，您可能需要多次运行 zpool status 命令。

```
# 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 更换出现故障的日志设备

ZFS 在 zpool status 命令输出中标识意图日志 (intent log) 故障。故障管理架构 (Fault Management Architecture, FMA) 也会报告这些错误。ZFS 和 FMA 都介绍如何从意图日志 (intent log) 故障中恢复。

以下示例说明如何从存储池 (pool) 中出现故障的日志设备 (c0t5d0) 进行恢复。基本步骤如下：

- 按照 https://support.oracle.com/中的 "*ZFS intent log read failure (Doc ID 1021625.1)*"（ZFS 意图日志读取失败（文档 ID 1021625.1））所述，查看 zpool status -x 输出和 FMA 诊断消息。
- 物理更换出现故障的日志设备。
- 使新日志设备联机。
- 清除池的错误状态。
- 清除 FMA 错误。

例如，如果系统在将同步写操作提交给具有单独日志设备的池之前突然关闭，您将会看到类似以下内容的信息：

```
# 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
```

您可以通过以下方式解决日志设备故障：

- 更换或恢复日志设备。在此示例中，日志设备是 `c0t5d0`。
- 将日志设备重新联机。

  ```
  # zpool online pool c0t5d0
  ```
- 重置故障日志设备的错误状态。

  ```
  # zpool clear pool
  ```

要从此错误中恢复而不更换故障日志设备，可以使用 `zpool clear` 命令清除该错误。在这种情况下，池将在降级模式下运行，并且日志记录将被写入到主池，直到更换单独的日志设备。

请考虑使用镜像日志设备来避免日志设备故障情形。

查看重新同步状态

替换设备这一过程可能需要很长一段时间，具体取决于设备的大小和池中的数据量。将数据从一个设备移动到另一个设备的过程称为重新同步，可以使用 `zpool status` 命令监视此过程。

提供了以下 `zpool status` 重新同步状态消息：

- 重新同步进度报告。例如：

  ```
  scan: resilver in progress since Mon Jun  7 09:17:27 2010
  13.3G scanned
  13.3G resilvered at 18.5M/s, 82.34% done, 0h2m to go
  ```
- 重新同步完成消息。例如：

  ```
  resilvered 16.2G in 0h16m with 0 errors on Mon Jun  7 09:34:21 2010
  ```

重新同步完成消息在系统重新引导后仍会保留。

传统的文件系统在块级别上重新同步数据。由于 ZFS 消除了卷管理器的人为分层，因此它能够以更强大的受控方式执行重新同步。此功能的两个主要优点如下：

- ZFS 仅重新同步最少量的必要数据。如果是短暂的断电（而不是设备替换），整个磁盘可以在几分钟或几秒内重新同步。替换整个磁盘时，重新同步过程所用的时间与磁盘上所用的数据量成比例。如果只使用了池中几 GB 的磁盘空间，则替换 500 GB 的磁盘可能只需要几秒的时间。
- 如果系统断电或者进行重新引导，则重新同步过程会准确地从它停止的位置继续，而无需手动干预。

要查看重新同步过程，请使用 `zpool status` 命令。例如：

```
# 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 scanned54.5M resilvered at 5.46M/s, 24.64% done, 0h0m to go

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
```

在本示例中，磁盘 `c1t0d0` 被替换为 `c2t0d0`。通过查看状态输出的配置部分中是否显示有 `replacing` ，可观察到此替换虚拟设备的事件。此设备不是真正的设备，不可能使用它创建池。此设备的用途仅仅是显示重新同步进度，以及确定被替换的设备。

请注意，当前正进行重新同步的任何池都处于 `ONLINE` 或 `DEGRADED` 状态，这是因为在重新同步过程完成之前，池无法提供所需的冗余级别。虽然 I/O 始终是按照比用户请求的 I/O 更低的优先级调度的（以最大限度地减少对系统的影响），但是重新同步会尽可能快地进行。重新同步完成后，该配置将恢复为新的完整配置。例如：

```
# 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
```

池再次处于 `ONLINE` 状态，而且原故障磁盘 (`c1t0d0`) 已从配置中删除。

更改池设备

请勿尝试更改活动池的池设备。

磁盘由其路径及其设备 ID（如果可用）标识。在设备 ID 信息可用的系统上，这种标识方法允许重新配置设备而无需更新 ZFS。由于设备 ID 生成和管理可能因系统而异，因此应在移动设备之前导出池，例如在将磁盘从一个控制器移动到另一个控制器之前。诸如固件更新或其他硬件变化之类的系统事件可能会更改 ZFS 存储池中的设备 ID，导致设备不可用。

另一个问题是如果您尝试更改池下的设备，然后作为非 root 用户使用 `zpool status` 命令，则可能会显示以前的设备名。

解决 ZFS 存储池中的数据问题

以下举例说明了部分数据问题：

- 池或文件系统空间缺失
- 由于损坏的磁盘或控制器而导致的瞬态 I/O 错误
- 磁盘上的数据因宇宙射线而损坏
- 导致数据传输至错误目标或从错误源位置传输的驱动程序已知问题
- 用户意外地覆写了物理设备的某些部分

在一些情况下，这些错误是瞬态的，如控制器出现问题时的随机 I/O 错误。在另外一些情况下，损坏是永久性的，如磁盘损坏。但是，若损坏是永久性的，则并不一定表明该错误很可能会再次出现。例如，如果您意外覆盖了某个磁盘的一部分，且未出现任何类型的硬件故障，则不需要替换该设备。准确确定设备的问题不是一项轻松的任务，在稍后的一节中将对此进行更详细的介绍。

解决 ZFS 空间问题

如果不确定 ZFS 如何报告文件系统和池空间记帐信息，请查看以下各节。另请查看“ZFS 磁盘空间记帐” [18]。

ZFS 文件系统空间报告

在确定可用的池和文件系统空间方面，`zpool list` 和 `zfs list` 命令要比以前的 `df` 和 `du` 命令出色。使用传统命令，既不能轻易分辨池和文件系统空间，也不能对后代文件系统或快照使用的空间做出解释。

例如，以下根池 (rpool) 有 5.46 GB 的已分配空间和 68.5 GB 的空闲空间。

```
# zpool list rpool
NAME   SIZE  ALLOC   FREE  CAP  DEDUP  HEALTH  ALTROOT
rpool   74G  5.46G  68.5G   7%  1.00x  ONLINE  -
```

如果通过查看各个文件系统的 USED 列来比较池空间记帐和文件系统空间记帐，则会看到 ALLOC 中报告的池空间可以用文件系统的 USED 总和来计算。例如：

```
# 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 存储池空间报告

由 zpool list 命令报告的 SIZE 值通常为池中的物理磁盘空间量，具体大小视池的冗余级别而异。请参见下面的示例。zfs list 命令列出了可供文件系统使用的可用空间，该空间等于磁盘空间减去 ZFS 池冗余元数据开销（若有）。

zfs list 命令将以下 ZFS 数据集配置视为已分配空间，但是在 zpool list 输出中没有将这些配置视为已分配空间：

- ZFS 文件系统配额
- ZFS 文件系统预留空间
- ZFS 逻辑卷大小

以下项说明了不同池配置对 zpool list 和 zfs list 输出的影响：

- **非冗余存储池** – 当池是使用一个 136 GB 的磁盘创建时，zpool list 命令会将 SIZE 值和初始的 FREE 值报告为 136 GB。由于存在少量的池元数据开销，因此 zfs list 命令报告的初始 AVAIL 空间为 134 GB。例如：

  ```
  # 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
  ```

- **镜像的存储池** – 当使用两个 136 GB 的磁盘创建一个池时，`zpool list` 命令会报告 `SIZE` 为 136 GB，初始 `FREE` 值为 136 GB。此处报告的是已压缩空间值。由于存在少量的池元数据开销，因此 `zfs list` 命令报告的初始 `AVAIL` 空间为 134 GB。例如：

```
# 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 存储池** – 当 `raidz2` 池是使用三个 136 GB 的磁盘创建时，`zpool list` 命令会将 `SIZE` 值和初始 `FREE` 值均报告为 408 GB。此处报告的是已解压磁盘空间值，其中包括冗余开销，如奇偶校验信息。由于存在池冗余开销，因此 `zfs list` 命令报告的初始 `AVAIL` 空间为 133 GB。`zpool list` 和 `zfs list` 输出的 RAID-Z 池空间存在差异的原因在于，`zpool list` 报告的是“已解压的”池空间。

```
# 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
```

  有关 `recordsize` 更改对 RAIDZ 空间记帐的影响，请参见“ZFS 磁盘空间记帐” [18]。

- **NFS 挂载文件系统空间** – `zpool list` 或 `zfs list` 都不考虑 NFS 挂载文件系统空间。但是，本地数据文件可能会隐藏在挂载的 NFS 文件系统下。如果您的文件系统空间缺失，请确保没有在 NFS 文件系统下隐藏数据文件。

检查 ZFS 文件系统完整性

对于 ZFS，不存在与 `fsck` 等效的实用程序。此实用程序传统上有两个作用：文件系统修复和文件系统验证。

文件系统修复

对于传统的文件系统，写入数据的方法本身容易出现导致文件系统不一致的意外故障。由于传统的文件系统不是事务性的，因此可能会出现未引用的块、错误的链接计数或其他不一致的文件系统结构。添加日志记录确实解决了其中的一些问题，但是在无法回滚日志时可能会带来其他问题。采用 ZFS 配置的磁盘存在数据不一致的唯一原因是硬件故障（在这种情况下该池应该已经设置冗余）或者 ZFS 软件存在错误。

fsck 实用程序可以解决 UFS 文件系统特有的已知问题。大多数 ZFS 存储池问题一般都与硬件故障或电源故障有关。使用冗余池可以避免许多问题。如果硬件故障或断电导致池损坏，请参见“修复 ZFS 存储池范围内的损坏” [264]。

如果没有冗余池，则始终存在因文件系统损坏而造成无法访问某些或所有数据的风险。

文件系统验证

除了文件系统修复外，fsck 实用程序还能验证磁盘上的数据是否没有问题。过去，此任务要求取消挂载文件系统并运行 fsck 实用程序，在该过程中可能会使系统进入单用户模式。此情况导致的停机时间的长短与所检查文件系统的大小成比例。ZFS 提供了一种对所有不一致性执行例程检查的机制，而不是要求显式实用程序执行必要的检查。此功能（称为清理）通常在内存或其他系统中使用，用于在错误导致硬件或软件故障前检测及预防这些错误。

控制 ZFS 数据清理

每当 ZFS 遇到错误时（不管是在清理时还是按需访问文件时），都会在内部记录该错误，以便您可以快速查看池中所有已知错误的概述。

显式 ZFS 数据清理

检查数据完整性的最简单方式是对该池中的数据进行显式清理。此操作对池中的所有数据遍历一次，并验证是否可以读取所有块。尽管任何 I/O 的优先级一直低于常规操作的优先级，但是清理以设备所允许的最快速度进行。虽然进行清理时池数据应该保持可用而且几乎都做出响应，但是此操作可能会对性能产生负面影响。要启动显式清理，请使用 zpool scrub 命令。例如：

```
# zpool scrub tank
```

使用 zpool status 命令可以显示当前清理操作的状态。例如：

```
# 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
```

每个池一次只能发生一个活动的清理操作。

可通过使用 -s 选项来停止正在进行的清理操作。例如：

```
# zpool scrub -s tank
```

在大多数情况下，目的在于确保数据完整性的清理操作应该一直执行到完成。如果清理操作影响了系统性能，您可以自行决定停止清理操作。

执行例程清理可以保证对系统上所有磁盘执行连续的 I/O。例程清理具有副作用，即阻止电源管理将空闲磁盘置于低功耗模式。如果系统基本上一直在执行 I/O，或功耗不是重要的考虑因素，则可以安全地忽略此问题。如果系统基本处于空闲状态，您希望节省磁盘功耗，应考虑使用 cron(1M) 安排显式清理，而不使用后台清理。这仍然会执行数据的全面清理，尽管在清理完成前会产生大量 I/O，完成时可以将磁盘作为正常状态进行电源管理。缺点（除了增加 I/O 外）是有大量时间没有进行任何清理操作，在这些时间段内数据损坏风险可能会增加。

有关解释 `zpool status` 输出的更多信息，请参见“查询 ZFS 存储池的状态” [64]。

ZFS 数据清理和重新同步

替换设备时，将启动重新同步操作，以便将正确副本中的数据移动到新设备。此操作是一种形式的磁盘清理。因此，在给定的时间，池中只能发生一个这样的操作。如果清理操作正在进行，则重新同步操作会暂停当前清理，并在重新同步完成后重新启动清理操作。

有关重新同步的更多信息，请参见“查看重新同步状态” [255]。

修复损坏的 ZFS 数据

一个或多个设备错误（指示一个或多个设备缺少或已损坏）影响顶层虚拟设备时，将出现数据损坏。例如，镜像的一半可能会遇到数千个绝不会导致数据损坏的设备错误。如果在镜像另一面的完全相同位置中遇到错误，则会导致数据损坏。

数据损坏始终是永久性的，因此在修复期间需要特别注意。即使修复或替换底层设备，也将永远丢失原始数据。这种情况通常要求从备份恢复数据。在遇到数据错误时会记录错误，并可以通过例程池清理对错误进行控制，如下一节所述。删除损坏的块后，下一遍清理会识别出数据损坏已不再存在，并从系统中删除该错误的任何记录。

以下各节介绍如何确定数据损坏的类型以及如何修复数据（如有可能）。

- “确定数据损坏的类型” [262]

- “修复损坏的文件或目录” [263]
- “修复 ZFS 存储池范围内的损坏” [264]

ZFS 使用校验和、冗余和自我修复数据来最大限度地减少出现数据损坏的风险。但是，如果没有冗余池，如果将池降级时出现损坏，或者不大可能发生的一系列事件协同损坏数据段的多个副本，则可能会出现数据损坏。不管是什么原因，结果都是相同的： 数据被损坏，因此无法再进行访问。所执行的操作取决于被损坏数据的类型及其相对值。可能损坏以下两种基本类型的数据：

- 池元数据 – ZFS 需要解析一定量的数据才能打开池和访问数据集。如果此数据被损坏，则整个池或部分数据集分层结构将变得不可用。
- 对象数据 – 在这种情况下，损坏发生在特定的文件或目录中。此问题可能会导致无法访问该文件或目录的一部分，或者此问题可能导致对象完全损坏。

数据是在常规操作期间和清理过程中验证的。有关如何验证池数据完整性的信息，请参见“检查 ZFS 文件系统完整性” [259]。

确定数据损坏的类型

缺省情况下，`zpool status` 命令仅说明已出现损坏，而不说明出现此损坏的位置。例如：

```
# 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
```

每个错误仅指示在给定时间点出现了错误。每个错误不一定仍存在于系统上。一般情况下是如此。某些临时故障可能会导致数据损坏，故障结束后会自动修复。完整的池清理可保证检查池中的每个活动块，因此每当清理完成后都会重置错误日志。如果确定错误不再存在，并且不希望等待清理完成，则使用 `zpool online` 命令重置池中的所有错误。

如果数据损坏位于池范围内的元数据中，则输出稍有不同。例如：

```
# 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
```

如果出现池范围的损坏，池将被置于 FAULTED 状态，这是因为池无法提供所需的冗余级别。

修复损坏的文件或目录

如果文件或目录被损坏，则系统可能仍然正常工作，具体取决于损坏的类型。如果系统中存在完好的数据副本，则任何损坏实际上都是不可恢复的。如果数据具有价值，必须从备份中恢复受影响的数据。尽管这样，您也许能够从此损坏恢复而不必恢复整个池。

如果损坏出现在文件数据块中，则可以安全地删除该文件，从而清除系统中的错误。使用 zpool status -v 命令可以显示包含持久性错误的文件名列表。例如：

```
# 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
```

包含持久性错误的文件名列表可能描述如下：

- 如果找到文件的全路径并且已挂载数据集，则会显示该文件的全路径。例如：

  ```
  /monkey/a.txt
  ```

- 如果找到文件的全路径但未挂载数据集，则会显示不带前导斜杠 (/) 的数据集名称，后面是数据集中文件的路径。例如：

```
monkey/ghost/e.txt
```

- 如果由于错误或由于对象没有与之关联的实际文件路径而导致文件路径的对象编号无法成功转换（dnode_t 便是这种情况），则会显示数据集名称，后跟该对象的编号。例如：

```
monkey/dnode:<0x0>
```

- 如果元对象集 (Metaobject Set, MOS) 中的对象损坏，则会显示特殊标签 <metadata>，后跟该对象的编号。

您可以尝试通过多次清理池和清除池错误来解决不太严重的数据损坏。如果在首次清理和清除中没有解决损坏的文件，请重新运行。例如：

```
# zpool scrub tank
# zpool clear tank
```

如果损坏发生在目录或文件的元数据中，则唯一的选择是将文件移动到别处。可以安全地将任何文件或目录移动到不太方便的位置，以允许恢复原始对象。

修复具有多块引用的损坏数据

如果受损的文件系统包含具有多块引用的损坏数据（如快照），则 zpool status -v 命令无法显示所有损坏数据的路径。当前 zpool status 对损坏数据的报告受以下项的限制：元数据的损坏量以及执行 zpool status 命令后是否重用了任何块。删除了重复数据的块使得对所有损坏数据的报告更加复杂。

如果有数据损坏，并且 zpool status -v 命令确定有快照数据受影响，请考虑运行以下命令来确定进一步的损坏数据路径：

修复 ZFS 存储池范围内的损坏

如果池元数据发生损坏，并且该损坏导致池无法打开或导入，则可以使用以下选项：

- 可以尝试使用 zpool clear -F 命令或 zpool import - F 命令恢复池。这些命令尝试回滚最后几次池事务，使其回到运行状态。可以使用 zpool status 命令查看损坏的池和建议的恢复步骤。例如：

```
# 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
```

前面的输出中描述的恢复过程要使用以下命令：

```
# zpool clear -F tpool
```

如果您尝试导入损坏的存储池，将会看到类似如下的消息：

```
# 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.
```

前面的输出中描述的恢复过程要使用以下命令：

```
# 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
```

如果损坏的池位于 zpool.cache 文件中，则系统引导时会发现该问题，并通过 zpool status 命令报告损坏的池。如果该池不在 zpool.cache 文件中，它将无法成功导入或打开，当您尝试导入该池时，会看到池受损消息。

- 您可以在只读模式下导入受损的池。此方法使您可以导入该池，从而可以访问数据。例如：

  ```
  # zpool import -o readonly=on tpool
  ```

  有关在只读模式下导入池的更多信息，请参见“在只读模式下导入池” [82]。
- 您可以使用 zpool import -m 命令导入缺少日志设备的池。有关更多信息，请参见“导入缺少日志设备的池” [80]。
- 如果无法使用上述池恢复方法恢复池，则必须从备份副本中恢复池及其所有数据。所用的机制通常随池配置和备份策略的不同而有很大差别。首先，保存 zpool status 命令所显示的配置，以便在销毁池后可以重新创建它。然后，使用 zpool destroy -f 命令销毁池。

此外，将描述数据集布局以及各个本地设置的属性的文件保存在某个安全位置，否则如果池出现无法访问的情况，将无法访问这些信息。使用池配置和数据集布局，可以在销毁池后重新构造完整的配置。然后可以使用任何备份或恢复策略填充数据。

修复损坏的 ZFS 配置

ZFS 在根文件系统中维护活动池及其配置的高速缓存。如果此高速缓存文件损坏或者不知何故变得与磁盘上所存储的配置信息不同步，则无法再打开池。虽然底层存储设备的质量始终可能会带来任意的损坏，但是 ZFS 会尽量避免出现此情况。此情况通常会导致池从系统中消失（它原本应该是可用的）。此情况还可能表现为不是一个完整的配置，缺少数量不明的顶层虚拟设备。在这两种情况下，都可以通过导出池（如果它确实可见）再重新导入池来恢复配置。

有关导入和导出池的信息，请参见“迁移 ZFS 存储池” [76]。

修复无法引导的系统

根据设计，即使在出错时 ZFS 也是强健而稳定的。尽管这样，在访问池时，软件已知问题或某些意外问题可能导致系统发出警告音。在引导过程中，必须打开每个池，这意味着这样的故障将导致系统进入应急重新引导循环。要从此情况中恢复，必须通知 ZFS 不要在启动时查找任何池。

ZFS 在 `/etc/zfs/zpool.cache` 中维护可用池及其配置的内部高速缓存。此文件的位置和内容是专用的，有可能更改。如果系统变得无法引导，则使用 `-m milestone=none` 引导选项引导到 `none` 里程碑。系统启动后，将根文件系统重新挂载为可写入，然后重命名 `/etc/zfs/zpool.cache` 文件或将其移动到其他位置。这些操作使 ZFS 忘记系统上存在池，从而阻止它尝试访问导致问题的有问题池。然后可以通过发出 `svcadm milestone all` 命令进入正常系统状态。从备用根引导时，可以使用类似的过程执行修复。

系统启动后，可以尝试使用 `zpool import` 命令导入池。但是，这样做很可能会导致在引导期间出现的相同错误，因为该命令使用相同机制访问池。如果系统中存在多个池，请执行以下操作：

- 重命名 `zpool.cache` 文件或将其移动到其他位置，如上文所述。
- 使用 `fmdump -eV` 命令显示报告有致命错误的池，确定哪个池可能有问题。
- 逐个导入池，跳过有问题的池，如 `fmdump` 输出中所示。

♦ ♦ ♦ 第 11 章

11 建议的 Oracle Solaris ZFS 做法

本章介绍了用于创建、监视和维护 ZFS 存储池和文件系统的建议做法。

本章包含以下各节：

有关包括 Oracle 数据库调优在内的一般 ZFS 调优信息，请参见《Oracle Solaris 11.2 可调参数参考手册》中的第 3 章 “Oracle Solaris ZFS 可调参数”。

建议的存储池做法

以下各节提供了用于创建和监视 ZFS 存储池的建议做法。有关解决存储池问题的信息，请参见第 10 章 Oracle Solaris ZFS 故障排除和池恢复。

一般系统做法

- 通过最新的 Solaris 更新和发行版使系统保持最新
- 在更改池设备或分隔镜像存储池之前确认控制器支持高速缓存刷新命令是至关重要的，这有助于您了解数据能否安全写入。这通常不是 Oracle/Sun 硬件的问题，但最好还是确认是否已启用硬件的高速缓存刷新设置。
- 根据实际的系统工作负荷确定所需的内存大小
  - 通过已知应用程序内存资源占用（例如，用于数据库应用程序），可以限定 ARC 的大小，这样，应用程序将不需要从 ZFS 高速缓存回收其所需的内存。
  - 考虑重复数据删除内存要求
  - 使用以下命令确定 ZFS 内存使用情况：

```
# 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
```

  - 请考虑使用 ECC 内存以避免内存损坏。如果内存损坏而无提示，则可能会损坏数据。
- 执行定期备份 – 虽然所创建的具有 ZFS 冗余的池有助于减少因硬件故障而导致的停机时间，但是它不可避免地会受硬件故障、电源故障或断开的电缆的影响。请确保定期备份您的数据。如果数据很重要，则应该对其进行备份。提供数据副本的不同方法如下：
  - 定期或每天创建 ZFS 快照
  - 每周备份 ZFS 池数据。可以使用 `zpool split` 命令创建 ZFS 镜像存储池的精确副本。
  - 使用企业级备份产品每月进行备份
- 硬件 RAID
  - 考虑将 JBOD 模式用于存储阵列而不是硬件 RAID，以便 ZFS 可以管理存储和冗余。
  - 使用硬件 RAID 和/或 ZFS 冗余
  - 使用 ZFS 冗余有很多好处 – 对于生产环境，配置 ZFS 以便它可以修复数据不一致性。无论在底层存储设备上实施的 RAID 级别如何，均可使用 ZFS 冗余，如 RAID-Z、RAID-Z-2、RAID-Z-3、镜像。通过此类冗余，ZFS 可以搜索和修复底层存储设备或它到主机的连接上的故障。
  - 如果您对硬件 RAID 解决方案的冗余性有信心，那么可以考虑对 ZFS 使用硬件 RAID 阵列而不使用 ZFS 冗余功能。但是，请按照以下建议来确保数据完整性。
    - 考虑到硬件 RAID 阵列发生故障时 ZFS 无法解决数据不一致性，根据您的方便程度分配 LUN 和 ZFS 存储池的大小。
    - 使用全局热备件创建 RAID5 LUN。
    - 通过使用 `zpool status` 监视 ZFS 存储池，通过使用硬件 RAID 监视工具监视底层 LUN。
    - 迅速更换任何故障设备。
    - 如果您使用的是数据中心品质的服务，请定期（例如每月）清理您的 ZFS 存储池。
    - 必须始终为重要数据保留最新的高质量备份。

  另请参见“在本地或网络连接存储阵列上创建池的做法” [271]。
- 故障转储会占用更多磁盘空间，范围通常为物理内存大小的 1/2- 3/4。

ZFS 存储池创建做法

以下各节提供了一般的和较具体的池做法。

一般存储池做法

- 使用整个磁盘来启用磁盘写入高速缓存并使维护更轻松。在分片上创建池会增加磁盘管理和恢复工作的复杂性。
- 使用 ZFS 冗余以便 ZFS 可以修复数据不一致性。
  - 创建非冗余池时，将显示以下消息：

    ```
    # zpool create tank c4t1d0 c4t3d0
    'tank' successfully created, but with no redundancy; failure
    of one device will cause loss of the pool
    ```
  - 对于镜像池，使用镜像磁盘对
  - 对于 RAID-Z 池，为每个 VDEV 组合 3-9 个磁盘
  - 不要在同一个池中混用 RAID-Z 和镜像组件。这些池更难以管理，并且性能可能会受到影响。
- 使用热备件以减少因硬件故障而导致的停机时间
- 使用大小近似的磁盘以便在各个设备之间平衡 I/O
  - 较小的 LUN 可以扩展为大的 LUN
  - 扩展 LUN 时请避免大小差异极大（如 128 MB 到 2 TB），以保持最佳 metaslab 大小
- 考虑创建一个小的根池和较大的数据池，以支持更快的系统恢复
- 建议的最小池大小为 8 GB。虽然最小池大小为 64 MB，但是小于 8 GB 会使空闲池空间的分配和回收比较困难。
- 建议根据您的工作负荷和数据大小确定最大池大小。不要尝试存储比您可以例行定期备份的数据更多的数据。否则，您的数据可能因某种无法预料的事件而处于风险中。

另请参见“在本地或网络连接存储阵列上创建池的做法” [271]。

根池创建做法

- SPARC (SMI (VTOC))：通过使用 `s*` 标识符使用分片创建根池。不要使用 `p*` 标识符。通常，在安装系统时会创建系统的 ZFS 根池。如果要创建第二个根池或者重新创建根池，请在 SPARC 系统上使用类似如下内容的语法：

  ```
  # zpool create rpool c0t1d0s0
  ```

  或者，创建一个镜像根池。例如：

  ```
  # zpool create rpool mirror c0t1d0s0 c0t2d0s0
  ```

- Solaris 11.1 x86 (EFI (GPT))：通过指定 `d*` 标识符使用整个磁盘创建根池。不要使用 `p*` 标识符。通常，在安装系统时会创建系统的 ZFS 根池。如果要创建另一个根池或者重新创建根池，请使用类似如下内容的语法：

  ```
  # zpool create rpool c0t1d0
  ```

  或者，创建一个镜像根池。例如：

  ```
  # zpool create rpool mirror c0t1d0 c0t2d0
  ```
- 根池必须作为镜像配置或单磁盘配置创建。不支持 RAID-Z 和条带化配置。不能使用 `zpool add` 命令添加其他磁盘以创建多个镜像顶层虚拟设备，但可以使用 `zpool attach` 命令扩展镜像虚拟设备。
- 根池不能有单独的日志设备。
- 在 AI 安装期间可以设置池属性，但是在根池上不支持 `gzip` 压缩算法。
- 通过初始安装创建了根池后，请勿对根池重命名。重命名根池可能会导致系统无法引导。
- 由于根池磁盘对连续操作至关重要（特别是在企业环境中），因此对于生产系统请勿在 USB 存储器上创建根池。可以考虑将系统的内部磁盘用作根池，或至少使用与非根目录数据所要使用磁盘的质量相同的磁盘。此外，USB 存储器 (USB stick) 的空间可能不足以支持转储卷大小，转储卷大小至少为物理内存大小的 1/2。
- 考虑创建双向或三向镜像根池，而不是向根池添加热备件。此外，不要在根池和数据池之间共享热备件。
- 请勿将 VMware 精简置备的设备用作根池设备。

非根池创建做法

- 通过使用 `d*` 标识符使用整个的磁盘创建非根池。不要使用 `p*` 标识符。
  - ZFS 在没有任何其他卷管理软件的情况下工作最佳。
  - 为了获得更出色的性能，请使用单个磁盘，至少也要使用仅由少数几个磁盘组成的 LUN。通过在 LUN 设置中为 ZFS 提供更大的可见性，ZFS 能够更好地进行 I/O 调度决策。
- 在多个控制器之间创建冗余的池配置，以减少因控制器故障而导致的停机时间。
  - **镜像存储池** – 占用更多磁盘空间，但通常情况下，对于小的随机读取，性能更好。

    ```
    # zpool create tank mirror c1d0 c2d0 mirror c3d0 c4d0
    ```
  - RAID-Z 存储池 – 可以使用 3 个奇偶校验策略创建，其中奇偶校验等于 1 (`raidz`)、2 (`raidz2`) 或 3 (`raidz3`)。RAID-Z 配置最大限度地利用了磁盘空间，通常情况下，在以大块（128K 或更大）写入和读取数据时性能良好。
    - 考虑一个单奇偶校验 RAID-Z (`raidz`) 配置，其中 2 个 VDEV 各有 3 个磁盘 (2+1)。

      ```
      # zpool create rzpool raidz1 c1t0d0 c2t0d0 c3t0d0 raidz1 c1t1d0 c2t1d0 c3t1d0
      ```

- RAIDZ-2 配置可提供更高的数据可用性，其性能与 RAID-Z 类似。与 RAID-Z 或双向镜像相比，RAIDZ-2 的数据丢失平均时间 (mean time to data loss, MTTDL) 要好得多。在 6 个磁盘 (4+2) 上创建双奇偶校验 RAID-Z (`raidz2`) 配置。

  ```
  # zpool create rzpool raidz2 c0t1d0 c1t1d0 c4t1d0 c5t1d0 c6t1d0 c7t1d0
  raidz2 c0t2d0 c1t2d0 c4t2d0 c5t2d0 c6t2d0 c7t2d
  ```

- RAIDZ-3 配置最大限度地利用了磁盘空间并提供了极佳的可用性，因为它可以承受 3 个磁盘故障。在 9 个磁盘 (6+3) 上创建三重奇偶校验 RAID-Z (`raidz3`) 配置。

  ```
  # zpool create rzpool raidz3 c0t0d0 c1t0d0 c2t0d0 c3t0d0 c4t0d0
  c5t0d0 c6t0d0 c7t0d0 c8t0d0
  ```

在本地或网络连接存储阵列上创建池的做法

在本地或远程连接的存储阵列上创建 ZFS 存储池时，请考虑以下存储池做法。

- 如果在 SAN 设备上创建池，且网络连接较慢，池的设备可能会有一段时间处于 `UNAVAIL` 状态。需要评估网络连接是否适合以连续的方式提供数据。另外请注意，如果将 SAN 设备用于根池，这些设备在系统引导后可能不会马上可用，从而可能使根池的设备变为 `UNAVAIL`。
- 向您的阵列供应商确认磁盘阵列未在 ZFS 发出刷新写入高速缓存请求后刷新其高速缓存。
- 将整个磁盘（而非磁盘分片）用作存储池设备，以允许 Oracle Solaris ZFS 激活小型本地磁盘高速缓存，从而在适当时间进行刷新。
- 为获得最佳性能，请为阵列中的每个物理磁盘创建一个 LUN。仅使用一个大型 LUN 可能会导致 ZFS 中排队的读取 I/O 操作过少，实际上无法获得最佳存储性能。与之相反，使用许多小型 LUN 可能会造成大量待处理的读取 I/O 操作，使存储器无法及时处理。
- 对于 Oracle Solaris ZFS，不建议存储阵列使用动态（或瘦）置备软件实现虚拟空间分配。Oracle Solaris ZFS 将已修改的数据写入空闲空间时，它将写入整个 LUN。从存储阵列的角度来看，Oracle Solaris ZFS 写入过程分配了所有虚拟空间，使得动态置备的优势无法体现。

  使用 ZFS 时可能无需考虑动态置备软件：

  - 可以将现有 ZFS 存储池中的 LUN 扩展为使用新空间。
  - 使用较大的 LUN 替换较小的 LUN 时的行为与此类似。
  - 如果评估池的存储需求，并使用可满足必需的存储需求的较小 LUN 创建池，则在需要更多空间时，始终可以将 LUN 扩展为更大大小。
- 如果阵列可以提供单个设备（JBOD 模式），则考虑在此类型的阵列上创建冗余 ZFS 存储池（镜像或 RAID-Z），以允许 ZFS 报告并更正不一致的数据。

针对 Oracle 数据库的池创建做法

在创建 Oracle 数据库时，请考虑以下存储池做法。

- 为池使用镜像池或硬件 RAID
- 对于随机读取工作负荷，通常不建议使用 RAID-Z 池
- 为数据库恢复日志创建一个具有单独日志设备的小的单独池
- 为归档日志创建一个小的单独池

有关为 Oracle 数据库调优 ZFS 的更多信息，请参见《Oracle Solaris 11.2 可调参数参考手册 》中的“为 Oracle 数据库进行 ZFS 调优”。

在 VirtualBox 中使用 ZFS 存储池

- 缺省情况下，Virtual Box 配置为忽略来自底层存储器的高速缓存刷新命令。这意味着，系统崩溃或硬件发生故障时数据可能会丢失。
- 通过发出以下命令可在 Virtual Box 上启用高速缓存刷新：

  ```
  VBoxManage setextradata vm-name "VBoxInternal/Devices/type/0/LUN#n/Config/IgnoreFlush" 0
  ```

  - *vm-name* – 虚拟机的名称
  - *type* – 控制器类型，为 `piix3ide`（如果使用常用的 IDE 虚拟控制器）或 `ahci`（如果使用的是 SATA 控制器）
  - *n* – 磁盘编号

针对性能的存储池做法

- 为获得最佳性能，请将池容量保持在 90% 以下
- 对于随机的读取/写入工作负荷，建议使用镜像池，而不是 RAID-Z 池
- 单独的日志设备
  - 建议使用以提高同步写入性能
  - 在同步写入负荷很高时，可防止在主池中写入许多日志块而产生分段
- 建议使用单独的高速缓存设备以改善读取性能
- 清理/重新同步 – 具有许多设备的大型 RAID-Z 池需要较长的清理和重新同步时间
- 池性能很差 – 可以使用 `zpool status` 命令来排除导致池性能很差的硬件问题。如果在 `zpool status` 命令中未揭露出问题，请使用 `fmdump` 命令显示硬件故障，或者使用 `fmdump -eV` 命令查看已存在但尚未导致报告故障的任何硬件错误。

ZFS 存储池维护和监视做法

- 为获得最佳性能，请确保池容量在 90% 以下。

如果池非常满并且文件系统频繁更新（例如，在繁忙的邮件服务器上），池性能可能会降低。池处于已满状态可能会影响性能，但不会产生其他问题。如果主要工作负荷为不可改变的文件，则应将池保持在 95-96% 利用率范围内。即使将大部分静态内容保持在 95-96% 范围内，写入、读取和重新同步性能也会受到影响。

- 对池和文件系统空间进行监视以确保它们未满。
- 考虑使用 ZFS 配额和预留空间，以确保文件系统空间不超过池容量的 90%。

- 监视池运行状况
  - 每周至少使用 `zpool status` 和 `fmdump` 监视冗余池一次。
  - 每周至少使用 `zpool status` 和 `fmdump` 监视非冗余池两次。
- 定期运行 `zpool scrub` 以识别数据完整性问题。
  - 如果使用的是使用者质量的驱动器，请考虑制定每周清理计划。
  - 如果使用的是数据中心质量的驱动器，请考虑制定每月清理计划。
  - 在更换设备或暂时减小池的冗余以前也应当运行清理，以确保所有设备当前都是可运转的。
- 监视池或设备故障 – 按如下说明使用 `zpool status`。此外，使用 `fmdump` 或 `fmdump -eV` 查看是否已出现任何设备故障或错误。
  - 对于冗余池，每周使用 `zpool status` 和 `fmdump` 监视池运行状况一次
  - 对于非冗余池，每周使用 `zpool status` 和 `fmdump` 监视池运行状况两次
- 池设备状态为 `UNAVAIL` 或 `OFFLINE` – 如果池设备不可用，请检查在 `format` 命令输出中是否列出了该设备。如果在 `format` 输出中未列出该设备，则它对 ZFS 将是不可见的。

  如果某个池设备具有 `UNAVAIL` 或 `OFFLINE` 状态，则这通常表示该设备已出现故障、电缆已断开或者出现某个其他硬件问题，如坏的电缆或坏的控制器已导致设备无法访问。
- 考虑配置 `smtp-notify` 服务，以便在硬件组件被诊断为有故障时通知您。有关更多信息，请参见 `smf(5)` 和 `smtp-notify(1M)` 的“通知参数”部分。

  缺省情况下，某些通知设置为自动发送给 root 用户。如果您作为 root 用户在 `/etc/aliases` 文件中为您的用户帐户添加了一个别名，则将收到类似以下内容的电子邮件通知：

```
From noaccess@tardis.space.com Fri Jun 29 16:58:59 2012
Date: Fri, 29 Jun 2012 16:58:58 -0600 (MDT)
From: No Access User <noaccess@tardis.space.com>
Message-Id: <201206292258.q5TMwwFL002753@tardis.space.com>
Subject: Fault Management Event: tardis:ZFS-8000-8A
To: root@tardis.central.com
Content-Length: 771

SUNW-MSG-ID: ZFS-8000-8A, TYPE: Fault, VER: 1, SEVERITY: Critical
EVENT-TIME: Fri Jun 29 16:58:58 MDT 2012
PLATFORM: ORCL,SPARC-T3-4, CSN: 1120BDRCCD, HOSTNAME: tardis
```

```
SOURCE: zfs-diagnosis, REV: 1.0
EVENT-ID: 76c2d1d1-4631-4220-dbbc-a3574b1ee807
DESC: A file or directory in pool 'pond' could not be read due to corrupt data.
AUTO-RESPONSE: No automated response will occur.
IMPACT: The file or directory is unavailable.
REC-ACTION: Use 'fmadm faulty' to provide a more detailed view of this event.
Run 'zpool status -xv' and examine the list of damaged files to determine what
has been affected. Please refer to the associated reference document at
http://support.oracle.com/msg/ZFS-8000-8A for the latest service procedures
and policies regarding this diagnosis.
```

- 镜像您的存储池空间 – 可以使用 `zpool list` 命令和 `zfs list` 命令来确定文件系统数据占用的磁盘空间。ZFS 快照会占用磁盘空间；如果 `zfs list` 命令未列出它们，则它们可能暗地里占用磁盘空间。可以使用 `zfs list -t` 快照命令来确定快照占用的磁盘空间。

建议的文件系统做法

以下各节介绍了建议的文件系统做法。

根文件系统最佳做法

- 考虑保持较小的根文件系统，并使其与其他非根相关数据隔离，以加快根池恢复速度。
- 不要在 rpool/ROOT 中包括文件系统，这是一个无需管理的特殊容器，不应包含任何其他组件。

文件系统创建做法

以下各节介绍了 ZFS 文件系统创建做法。

- 针对每个用户为起始目录创建一个文件系统
- 对于重要文件系统，考虑使用文件系统配额和预留空间来管理和保留磁盘空间
- 在具有许多用户的环境中，考虑使用用户和组配额来管理磁盘空间
- 使用 ZFS 属性继承来将属性应用于许多后代文件系统

针对 Oracle 数据库的文件系统创建做法

在创建 Oracle 数据库时，请考虑以下文件系统做法。

- 使 ZFS `recordsize` 属性与 Oracle `db_block_size` 匹配。
- 使用 8 KB `recordsize` 和缺省的 `primarycache` 值，在主数据库池中创建数据库表和索引文件系统。
- 使用缺省的 `recordsize` 和 `primarycache` 值，在主数据库池中创建临时数据和撤消表空间文件系统。
- 启用压缩和缺省的 `recordsize` 值，并将 `primarycache` 设置为 `metadata`，在归档池中创建归档日志文件系统。

有关更多信息，请参见以下白皮书：

http://blogs.oracle.com/storage/entry/new_white_paper_configuring_oracle

用于监视 ZFS 文件系统的做法

您应当对 ZFS 文件系统进行监视，以确保它们可用并确保可及时发现空间占用问题。

- 每周使用 `zpool list` 和 `zfs list` 命令监视文件系统空间可用性一次，而不要使用 `du` 和 `df` 命令，因为传统命令不会计入后代文件系统或快照所占用的空间。

  有关更多信息，请参见“解决 ZFS 空间问题” [257]。
- 通过使用 `zfs list -o space` 命令，显示文件系统空间的占用情况。
- 文件系统空间可能会不知不觉地被快照占用。通过使用以下语法，可以显示所有的数据集信息：

  ```
  # zfs list -t all
  ```
- 在安装系统时会自动创建单独的 `/var` 文件系统，但是您应该对此文件系统设置配额和预留空间，以确保它不会不知不觉地占用根池空间。
- 此外，可以使用 `fsstat` 命令显示 ZFS 文件系统的文件操作活动。可按挂载点或文件系统类型来报告活动。以下示例显示常规的 ZFS 文件系统活动：

  ```
  # 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 /
  ```
- 备份
  - 保存文件系统快照
  - 考虑对企业级软件进行每周和每月备份
  - 在远程系统上存储根池快照，以便进行裸机恢复

♦ ♦ ♦ 附 录 A

Oracle Solaris ZFS 版本说明

本附录介绍可用的 ZFS 版本、各版本的功能以及提供 ZFS 版本和功能的 Solaris OS。

本附录包含以下部分：

ZFS 版本概述

引入了新的 ZFS 池和文件系统功能，可利用 Solaris 发行版中提供的特定 ZFS 版本进行访问。可以使用 `zpool upgrade` 或 `zfs upgrade` 确定一个池或文件系统的版本是否低于当前运行的 Solaris 发行版提供的版本。还可以使用这些命令升级池和文件系统版本。

有关使用 `zpool upgrade` 和 `zfs upgrade` 命令的信息，请参见“升级 ZFS 文件系统” [166]和“升级 ZFS 存储池” [85]。

ZFS 池版本

下表列出了 Oracle Solaris 发行版提供的 ZFS 池版本。

版本	Oracle Solaris 11	说明
1	snv_36	初始 ZFS 版本
2	snv_38	同样的块（复制的元数据）
3	snv_42	热备件和双奇偶校验 RAID-Z
4	snv_62	`zpool history`
5	snv_62	`gzip` 压缩算法
6	snv_62	`bootfs` 池属性

版本	Oracle Solaris 11	说明
7	snv_68	不同用途的日志设备
8	snv_69	委托管理
9	snv_77	`refquota` 和 `refreservation` 属性
10	snv_78	缓存设备
11	snv_94	改进的清理性能
12	snv_96	快照属性
13	snv_98	`snapused` 属性
14	snv_103	`aclinherit passthrough-x` 属性
15	snv_114	用户和组空间记帐
16	snv_116	`stmf` 属性
17	snv_120	三奇偶校验 RAID-Z
18	snv_121	快照用户存放
19	snv_125	日志设备移除
20	snv_128	`zle`（零长度编码）压缩算法
21	snv_128	重复数据删除
22	snv_128	已接收属性
23	snv_135	精简 ZIL
24	snv_137	系统属性
25	snv_140	改进了清理统计信息
26	snv_141	提高了删除快照的性能
27	snv_145	提高了创建快照的性能
28	snv_147	多个 vdev 替换
29	snv_148	RAID-Z/镜像混合分配器
30	snv_149	加密
31	snv_150	改进了 "zfs list" 性能
32	snv_151	1 MB 的块大小
33	snv_163	改进了共享支持
34	S11.1	共享继承

ZFS 文件系统版本

下表列出了 Oracle Solaris 发行版提供的 ZFS 文件系统版本。请注意，特定文件系统版本中提供的功能需要使用特定的池版本。

版本	Oracle Solaris 11	说明
1	snv_36	初始 ZFS 文件系统版本
2	snv_69	增强的目录项
3	snv_77	大小写无关性和文件系统唯一标识符 (FUID)

版本	Oracle Solaris 11	说明
4	snv_114	`userquota` 和 `groupquota` 属性
5	snv_137	系统属性
6	S11.1	多级别文件系统支持

索引

T

U

V

W

X