Oracle® Solaris ZFS 管理指南

文件号码 E38893–03
2013 年 6 月

ORACLE

版权所有 © 2006, 2013, Oracle 和/或其附属公司。 保留所有权利。

本软件和相关文档是根据许可证协议提供的，该许可证协议中规定了关于使用和公开本软件和相关文档的各种限制，并受知识产权法的保护。除非在许可证协议中明确许可或适用法律明确授权，否则不得以任何形式、任何方式使用、拷贝、复制、翻译、广播、修改、授权、传播、分发、展示、执行、发布或显示本软件和相关文档的任何部分。除非法律要求实现互操作，否则严禁对本软件进行逆向工程设计、反汇编或反编译。

此文档所含信息可能随时被修改，恕不另行通知，我们不保证该信息没有错误。如果贵方发现任何问题，请书面通知我们。

如果将本软件或相关文档交付给美国政府，或者交付给以美国政府名义获得许可证的任何机构，必须符合以下规定：

U.S. GOVERNMENT END USERS:

Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

本软件或硬件是为了在各种信息管理应用领域内的一般使用而开发的。它不应被应用于任何存在危险或潜在危险的应用领域，也不是为此而开发的，其中包括可能会产生人身伤害的应用领域。如果在危险应用领域内使用本软件或硬件，贵方应负责采取所有适当的防范措施，包括备份、冗余和其它确保安全使用本软件或硬件的措施。对于因在危险应用领域内使用本软件或硬件所造成的一切损失或损害，Oracle Corporation 及其附属公司概不负责。

Oracle 和 Java 是 Oracle 和/或其附属公司的注册商标。其他名称可能是各自所有者的商标。

Intel 和 Intel Xeon 是 Intel Corporation 的商标或注册商标。所有 SPARC 商标均是 SPARC International, Inc 的商标或注册商标，并应按照许可证的规定使用。AMD、Opteron、AMD 徽标以及 AMD Opteron 徽标是 Advanced Micro Devices 的商标或注册商标。UNIX 是 The Open Group 的注册商标。

本软件或硬件以及文档可能提供了访问第三方内容、产品和服务的方式或有关这些内容、产品和服务的信息。对于第三方内容、产品和服务，Oracle Corporation 及其附属公司明确表示不承担任何种类的担保，亦不对其承担任何责任。对于因访问或使用第三方内容、产品或服务所造成的任何损失、成本或损害，Oracle Corporation 及其附属公司概不负责。

130701@25097

目录

前言

《Oracle Solaris ZFS 管理指南》提供有关设置和管理 Oracle Solaris ZFS 文件系统的信息。

本指南同时包含了适用于基于 SPARC 和基于 x86 的系统的信息。

注 – 此 Oracle Solaris 发行版支持使用 SPARC 和 x86 系列处理器体系结构的系统。支持的系统可以在 http://www.oracle.com/webfolder/technetwork/hcl/index.html 上的《Oracle Solaris Hardware Compatibility List》（《Oracle Solaris 硬件兼容性列表》）中找到。本文档列举了在不同类型的平台上进行实现时的所有差别。

目标读者

本指南适用于对设置和管理 Oracle Solaris ZFS 文件系统感兴趣的任何用户。最好具有使用 Oracle Solaris 操作系统 (Operating System, OS) 或其他 UNIX 版本的经验。

本书的结构

下表介绍了本书中的各章。

章	说明
第 1 章，Oracle Solaris ZFS 文件系统（介绍）	概述 ZFS 及其功能和优点。本章还介绍了一些基本概念和术语。
第 2 章，Oracle Solaris ZFS 入门	提供通过基本池和文件系统设置基本 ZFS 配置的逐步说明。本章还介绍了创建 ZFS 文件系统所需的硬件和软件。
第 3 章，管理 Oracle Solaris ZFS 存储池	提供有关如何创建和管理 ZFS 存储池的详细说明。
第 4 章，安装和引导 Oracle Solaris ZFS 根文件系统	介绍如何安装和引导 ZFS 文件系统。同时还对使用 Oracle Solaris Live Upgrade 将 UFS 根文件系统迁移到 ZFS 根文件系统进行了介绍。
第 5 章，管理 Oracle Solaris ZFS 文件系统	提供有关管理 ZFS 文件系统的详细信息，其中包括分层次的文件系统布局、属性继承以及自动挂载点管理和共享交互等概念。

章	说明
第 6 章，使用 Oracle Solaris ZFS 快照和克隆	介绍如何创建和管理 ZFS 快照和克隆。
第 7 章，使用 ACL 和属性保护 Oracle Solaris ZFS 文件	介绍如何使用访问控制列表 (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。
第 8 章，Oracle Solaris ZFS 委托管理	介绍如何使用 ZFS 委托管理来允许非特权用户执行 ZFS 管理任务。
第 9 章，Oracle Solaris ZFS 高级主题	提供有关使用 ZFS 卷、在安装了区域的 Oracle Solaris 系统中使用 ZFS 以及使用备用根池的信息。
第 10 章，Oracle Solaris ZFS 故障排除和池恢复	介绍如何确定 ZFS 故障以及如何从中进行恢复。本章还介绍了防止故障的步骤。
第 11 章，建议的 Oracle Solaris ZFS 做法	介绍了创建、监视和维护 ZFS 存储池和文件系统的建议做法。
附录 A，Oracle Solaris ZFS 版本说明	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。

相关书籍

以下书籍提供了有关常规 Oracle Solaris 系统管理主题的相关信息：

- 《系统管理指南：高级管理》
- 《System Administration Guide: Devices and File Systems》
- 《System Administration Guide: Security Services》

获取 Oracle 支持

Oracle 客户可以通过 My Oracle Support 获取电子支持。有关信息，请访问 `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info`，或访问 `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs`（如果您听力受损）。

印刷约定

下表介绍了本书中的印刷约定。

表 P–1 印刷约定

字体或符号	含义	示例
`AaBbCc123`	命令、文件和目录的名称；计算机屏幕输出	编辑 `.login` 文件。 使用 `ls -a` 列出所有文件。 `machine_name% you have mail.`
`AaBbCc123`	用户键入的内容，与计算机屏幕输出的显示不同	`machine_name%` **`su`** `Password:`
aabbcc123	要使用实名或值替换的命令行占位符	删除文件的命令为 `rm` *filename*。
AaBbCc123	保留未译的新词或术语以及要强调的词	这些称为 *Class* 选项。 **注意**：有些强调的项目在联机时以粗体显示。
新词术语强调	新词或术语以及要强调的词	**高速缓存**是存储在本地的副本。 请**勿**保存文件。
《书名》	书名	阅读《用户指南》的第 6 章。

命令中的 shell 提示符示例

下表显示了 Oracle Solaris OS 中包含的缺省 UNIX shell 系统提示符和超级用户提示符。请注意，在命令示例中显示的缺省系统提示符可能会有所不同，具体取决于 Oracle Solaris 发行版。

表 P–2 shell 提示符

shell	提示符
Bash shell、Korn shell 和 Bourne shell	`$`
Bash shell、Korn shell 和 Bourne shell 超级用户	`#`
C shell	`machine_name%`
C shell 超级用户	`machine_name#`

◆ ◆ ◆ 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：

ZFS 中的新增功能

本节概述了 ZFS 文件系统的新增功能。

ZFS 命令使用方面的增强功能

Oracle Solaris 10 1/13：zfs 和 zpool 命令具有 help 子命令，可用来提供有关 zfs 和 zpool 子命令及其所支持选项的更多信息。例如：

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

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

有关更多信息，请参见 zfs(1M) 和 zpool(1M)。

ZFS 快照增强功能

Oracle Solaris 10 1/13：此发行版包含以下 ZFS 快照增强功能：

- zfs snapshot 命令具有 snap 别名，该别名提供了此命令的缩写语法。例如：

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

- zfs diff 命令提供了枚举选项 -e，可标识在两个快照之间添加或修改的所有文件。命令生成的输出中会标识所添加的所有文件，但不提供可能的删除项。例如：

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

  还可以使用 -o 选项来标识要显示的选定字段。例如：

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

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

改进的 aclmode 属性

Oracle Solaris 10 1/13：每当在 chmod 操作期间修改文件的 ACL 权限时，aclmode 属性即会修改访问控制列表 (Access Control List, ACL) 的行为。已重新引入了 aclmode 属性，它具有以下属性值：

- 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。

有关更多信息，请参见示例 7–13。

Oracle Solaris ZFS 安装功能

Oracle Solaris 10 8/11：在此发行版中，提供了以下新增安装功能：

- 可以使用文本模式安装方法来通过 ZFS Flash 归档文件安装系统。有关更多信息，请参见示例 4–3。
- 可以使用 Oracle Solaris Live Upgrade luupgrade 命令安装 ZFS 根 Flash 归档文件。有关更多信息，请参见示例 4–8。
- 可以使用 Oracle Solaris Live Upgrade 的 lucreate 命令指定单独的 /var 文件系统。有关更多信息，请参见示例 4–5。

ZFS 发送流增强功能

Oracle Solaris 10 8/11：在此发行版中，您可以设置在快照流中发送和接收的文件系统属性。通过这些增强功能，可以灵活地将发送流中的文件系统属性应用到接收方文件系统，或确定接收时是否应忽略本地文件系统属性（如 mountpoint 属性值）。

有关更多信息，请参见第 196 页中的 “向 ZFS 快照流应用不同的属性值”。

ZFS 快照差异 (zfs diff)

Oracle Solaris 10 8/11：在此发行版中，您可以使用 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/cindy
fileA
$ zfs snapshot tank/cindy@0913
$ ls /tank/cindy
fileA  fileB
$ zfs snapshot tank/cindy@0914
```

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/cindy@0913 tank/cindy@0914
M       /tank/cindy/
+       /tank/cindy/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。

有关更多信息，请参见第 189 页中的 “确定 ZFS 快照的差异 (zfs diff)”。

ZFS 存储池恢复和性能增强功能

Oracle Solaris 10 8/11：在此发行版中，提供了以下新增的 ZFS 存储池功能：

- 可以使用 zpool import -m 命令导入缺少日志的池。有关更多信息，请参见第 87 页中的 “导入缺少日志设备的池”。
- 可以在只读模式下导入池。此功能主要用于池恢复。如果由于底层设备受损而无法访问受损的池，可在只读模式下导入池以恢复数据。有关更多信息，请参见第 88 页中的 “在只读模式下导入池”。
- 在此发行版中创建的 RAID-Z（raidz1、raidz2 或 raidz3）存储池会对某些对延迟敏感的元数据自动进行镜像，以提高读取 I/O 的吞吐量性能。对于升级到池版本 29 或以上版本的现有 RAID-Z 池，将对所有新写入的数据的某些元数据进行镜像。

  RAID-Z 池中的镜像元数据不对硬件故障提供额外的防护，这与镜像存储池类似。镜像元数据占用额外的空间，但是 RAID-Z 保护措施与先前发行版中的相同。此增强功能仅出于性能考虑。

ZFS 同步行为调优

Solaris 10 8/11：在此发行版中，您可以使用 sync 属性确定 ZFS 文件系统的同步行为。

缺省同步行为是将所有同步文件系统事务写入意图日志，并刷新所有设备以确保数据稳定。建议不要禁用缺省同步行为。依赖于同步支持的应用程序可能会受影响，并可能发生数据丢失的情况。

sync 属性可以在创建文件系统之前或之后设置。无论何种情况，属性值都将立即生效。例如：

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

在包含 sync 属性的 Oracle Solaris 发行版中，zil_disable 参数不再可用。

有关更多信息，请参见表 5–1。

改进了 ZFS 池消息

Oracle Solaris 10 8/11：在此发行版中，您可以使用 -T 选项提供时间间隔和计数值，使 zpool list 和 zpool status 命令显示额外信息。

此外，zpool status 命令还提供更多的池清理和重新同步信息，如下所示：

- 重新同步进度报告。例如：

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

- 清理进度报告。例如：

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

- 重新同步完成消息。例如：

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

- 清理完成消息。例如：

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

- 取消正在进行的清理消息。例如：

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

- 清理和重新同步完成消息在系统重新引导后仍存在

以下语法使用时间间隔和计数选项显示正在进行的池重新同步的信息。您可以使用 -T d 值以标准日期格式显示信息，或使用 -T u 以内部格式显示信息。

```
# zpool status -T d tank 3 2
Wed Nov 14 15:44:34 MST 2012
  pool: tank
 state: DEGRADED
status: One or more devices is currently being resilvered.  The pool will
        continue to function in a degraded state.
action: Wait for the resilver to complete.
  scan: resilver in progress since Wed Nov 14 15:44:34 2012
    2.96G scanned out of 4.19G at 189M/s, 0h0m to go
    1.48G resilvered, 70.60% done
config:

        NAME                       STATE     READ WRITE CKSUM
        tank                       DEGRADED     0     0     0
          mirror-0                 ONLINE       0     0     0
            c0t5000C500335F95E3d0  ONLINE       0     0     0
```

```
            c0t5000C500335F907Fd0  ONLINE       0     0     0
          mirror-1                 DEGRADED     0     0     0
            c0t5000C500335BD117d0  ONLINE       0     0     0
            c0t5000C500335DC60Fd0  DEGRADED     0     0     0  (resilvering)

errors: No known data errors
```

ZFS ACL 互操作性增强功能

Oracle Solaris 10 8/11：在此发行版中，提供了以下 ACL 增强功能：

- 除了特殊权限，普通 ACL 不需要 deny 访问控制条目 (Access Control Entry, ACE)。例如，0644、0755 或 0664 模式不需要 deny ACE，但某些模式（如 0705、0060 等）需要 deny ACE。

  以前的行为是普通 ACL（如 644）包含 deny ACE。例如：

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

  新的行为是普通 ACL（如 644）不包含 deny ACE。例如：

  ```
  # ls -v file.1
  -rw-r--r--   1 root     root      206663 Jun 22 14:30 file.1
       0:owner@:read_data/write_data/append_data/read_xattr/write_xattr
           /read_attributes/write_attributes/read_acl/write_acl/write_owner
           /synchronize:allow
       1:group@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
       2:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
           :allow
  ```

- 在继承过程中，ACL 不再被拆分成多个 ACE 以尝试保留未经修改的原始权限。而是根据需要修改权限以强制进入文件创建模式。
- aclinherit 属性行为包括当属性设置为 restricted 时减少权限，这意味着继承过程中不再将 ACL 拆分成多个 ACE。
- 一种新的权限模式计算规则，即，如果 ACL 的 *user* ACE 同时也是文件所有者，则这些权限将包括在权限模式计算中。如果 *group* ACE 是文件的组所有者，将适用同样的规则。

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

分割镜像 ZFS 存储池 (zpool split)

Oracle Solaris 10 9/10：在此发行版中，您可以使用 zpool split 命令分割镜像的存储池，从原镜像池中分离一个或多个磁盘，创建另一个完全相同的池。

有关更多信息，请参见第 59 页中的 “通过分割镜像 ZFS 存储池创建新池”。

新 ZFS 系统进程

Oracle Solaris 10 9/10：在此发行版中，每个 ZFS 存储池都有一个关联的进程 zpool-*poolname*。此进程中的线程是用来处理与池相关的 I/O 任务（如压缩和校验和验证）的池 I/O 处理线程。此进程的作用是使各存储池的 CPU 利用情况具有可见性。

使用 ps 和 prstat 命令可以查看有关这些运行进程的信息。这些进程仅在全局区域中可用。有关更多信息，请参见 SDC(7)。

ZFS 设备替换增强功能

Oracle Solaris 10 9/10：在此发行版中，使用较大磁盘替换池中的磁盘时会提供系统事件或 *sysevent*。ZFS 的功能已得到增强，可以识别这些事件，并根据磁盘的新大小调整池，具体取决于 autoexpand 属性的设置。可以使用 autoexpand 池属性允许或禁止在较大磁盘替换了较小磁盘时进行自动池扩展。

这些增强功能使您可以增大池的大小，而不必导出和导入池或重新引导系统。

例如，对 tank 池启用自动 LUN 扩展功能。

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

或者，可以在启用 autoexpand 属性的情况下创建池。

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

缺省情况下，autoexpand 属性是禁用的，以便您可以决定，当较大磁盘替换较小磁盘时是否要扩展池的大小。

使用 zpool online - e 命令也可以扩展池的大小。例如：

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

或者，您可以在附加了较大磁盘或使其变为可用之后使用 zpool replace 命令重置 autoexpand 属性。例如，下例利用一个 8-GB 磁盘 (c0t0d0) 创建了一个池。8-GB 磁盘被一个 16-GB 磁盘 (c1t13d0) 替换，但直到启用 autoexpand 属性之后，池大小才会扩展。

```
# zpool create pool c0t0d0
# zpool list
NAME    SIZE    ALLOC   FREE    CAP    HEALTH  ALTROOT
```

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

另一种不启用 autoexpand 属性即可扩展磁盘的方法是使用 zpool online -e 命令，即使设备已经处于联机状态。例如：

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

此发行版中的其他设备替换增强功能包括：

- 在先前的发行版中，如果替换磁盘的大小与现有磁盘稍有不同，ZFS 就无法替换现有磁盘或附加新磁盘。在此发行版中，只要池未满，就可以用几乎同样大小的新磁盘替换现有磁盘或附加新磁盘。
- 在此发行版中，扩展池大小无需重新引导系统或者导出并导入一个池。如前所述，可以启用 autoexpand 属性或者使用 zpool online -e 命令扩展池的大小。

有关替换设备的更多信息，请参见第 64 页中的 “替换存储池中的设备”。

ZFS 和 Flash 安装支持

Solaris 10 10/09：在此发行版中，您可以设置 JumpStart 配置文件来标识 ZFS 根池的 Flash 归档文件。有关更多信息，请参见第 103 页中的 “安装 ZFS 根文件系统（Oracle Solaris Flash 归档文件安装）”。

ZFS 环境中的区域迁移

Solaris 10 5/09：此发行版扩展了以下支持：使用 Oracle Solaris Live Upgrade 在 ZFS 环境中迁移区域。有关更多信息，请参见第 123 页中的 “使用 Oracle Solaris Live Upgrade 迁移或升级具有区域的系统（最低 Solaris 10 5/09）”。

有关此发行版的已知问题列表，请参见《Solaris 5 10/09 发行说明》。

ZFS 安装和引导支持

Solaris 10 10/08：此发行版提供了安装并引导 ZFS 根文件系统的功能。您可以使用初始安装选项或 JumpStart 功能来安装 ZFS 根文件系统。或者，可以使用 Oracle Solaris Live Upgrade 功能将 UFS 根文件系统迁移到 ZFS 根文件系统。此外，还提供了对交换和转储设备的 ZFS 支持。有关更多信息，请参见第 4 章，安装和引导 Oracle Solaris ZFS 根文件系统。

有关此发行版的已知问题列表，请参见《Solaris 10 10/08 发行说明》。

基于 Web 的 ZFS 管理

Solaris 10 6/06 发行版：ZFS 管理控制台是一个基于 Web 的 ZFS 管理工具，允许您执行以下管理任务：

- 创建新存储池。
- 为现有池添加功能。
- 将存储池移动（导出）到另一个系统。
- 导入以前导出的存储池，使其可在另一个系统中使用。
- 查看有关存储池的信息。
- 创建文件系统。
- 创建卷。
- 创建文件系统或卷的快照。
- 将文件系统回滚到以前的快照。

通过安全 Web 浏览器访问以下网址，可以访问 ZFS 管理控制台：

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

如果键入了适当的 URL 但无法访问 ZFS 管理控制台，则表明可能未启动服务器。要启动服务器，请运行以下命令：

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

如果希望服务器在系统引导时自动启动，请运行以下命令：

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

注 – 不能使用 Solaris Management Console (`smc`) 管理 ZFS 存储池或文件系统。

什么是 Oracle Solaris ZFS？

ZFS 文件系统从根本上改变了文件系统的管理方式，它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。此设计增加了更多复杂性，并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数据在虚拟卷上的物理放置。

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。由于数据需要写入两次，因此该过程会引入不必要的开销，而且通常会导致一组新问题，例如在无法正确地重放日志时。

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。

简化的管理

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：

alternate boot environment（备用引导环境）
通过 `lucreate` 命令创建且可能通过 `luupgrade` 命令更新的引导环境，但它不是活动环境或主引导环境。通过运行 `luactivate` 命令，可以使备用引导环境成为主引导环境。

checksum（校验和）
文件系统块中数据的 256 位散列。校验和功能的范围可以从简单快速的 fletcher4（缺省值）到强加密散列（如 SHA256）。

clone（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，请参见第 190 页中的 “ZFS 克隆概述”。

dataset（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。

file system（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。

mirror（镜像）
在两个或更多磁盘上存储相同数据副本的虚拟设备。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都可以提供相同的数据。

pool（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。

primary boot environment（主引导环境）
`lucreate` 命令生成备用引导环境所用的引导环境。缺省情况下，主引导环境是当前引导环境。可以使用 `lucreate -s` 选项覆盖此缺省设置。

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，请参见第 41 页中的 “RAID-Z 存储池配置”。

resilvering（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，请参见第 264 页中的 “查看重新同步状态”。

snapshot（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，请参见第 183 页中的 “ZFS 快照概述”。

virtual device（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，请参见第 48 页中的 “显示存储池虚拟设备信息”。

volume（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，请参见第 237 页中的 “ZFS 卷”。

ZFS 组件命名要求

每个 ZFS 组件（如数据集和池等）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 与传统文件系统之间的差别

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，请参见第 34 页中的 “创建 ZFS 文件系统分层结构”。

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，请参见第 177 页中的 “设置 ZFS 文件系统的配额”。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，请参见第 180 页中的 “设置 ZFS 文件系统的预留空间”。此模型与从同一文件系统（例如 `/home`）挂载多个目录的 NFS 模型非常相似。

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，每个文件浪费大约 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 条目。

有关挂载和共享 ZFS 文件系统的更多信息，请参见第 171 页中的 “挂载 ZFS 文件系统”。

传统卷管理

如第 24 页中的 “ZFS 池存储” 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 草案 ACL 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

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，则可采用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《System Administration Guide: Security Services》。

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` 全部都可供使用。

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

有关冗余 ZFS 池配置的更多信息，请参见第 40 页中的 “ZFS 存储池的复制功能”。

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` 中。

```
# 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，请参见第 34 页中的 “创建 ZFS 文件系统分层结构”。

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。设备可以是预先格式化的磁盘上的单个分片，也可以是 ZFS 格式化为单个大分片的整个磁盘。

在第 33 页中的 “如何创建 ZFS 存储池”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，请参见第 37 页中的 “使用 ZFS 存储池中的磁盘”。

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）。

第 33 页中的 “如何创建 ZFS 存储池”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，请参见第 40 页中的 “ZFS 存储池的复制功能”。

▼ 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，请参见第 31 页中的 “ZFS 权限配置文件”。

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，但是必须符合第 27 页中的 “ZFS 组件命名要求”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，请参见第 43 页中的 “创建 ZFS 存储池”。有关如何确定设备使用情况的更多信息，请参见第 49 页中的 “检测使用中的设备”。

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，请参见第 73 页中的 “查询 ZFS 存储池的状态”。

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。使用过文件系统的任何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。

第 35 页中的 “如何创建 ZFS 文件系统”中创建了两个 ZFS 文件系统：`jeff` 和 `bill`。

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。

在第 35 页中的 “如何创建 ZFS 文件系统”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。

在第 35 页中的 “如何创建 ZFS 文件系统”的示例中，所有起始目录都挂载在 `/export/zfs/user` 中，都通过使用 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，请参见第 154 页中的 “ZFS 属性介绍”。

▼ 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，请参见第 31 页中的 “ZFS 权限配置文件”。

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 sharenfs=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 sharenfs=on -o compression=on tank/home
```

有关属性和属性继承的更多信息，请参见第 154 页中的 “ZFS 属性介绍”。

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，请参见第 152 页中的 “创建 ZFS 文件系统”。

有关挂载和共享文件系统的更多信息，请参见第 171 页中的 “挂载 ZFS 文件系统”。

5 **设置文件系统特定的属性。**

在本示例中，为用户 `jeff` 指定了 10 GB 的配额。此属性可对该用户可以使用的空间量施加限制，而无需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，请参见第 164 页中的 “查询 ZFS 文件系统信息”。

有关磁盘空间的使用和计算方法的更多信息，请参见第 28 页中的 “ZFS 磁盘空间记帐”。

◆ ◆ ◆ 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
```

在 ZFS 存储池中使用整个的磁盘时，请检查以下注意事项：

- 使用整个的磁盘时，通常使用 `/dev/dsk/cNtNdN` 命名约定对磁盘进行命名。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《System Administration Guide: Devices and File Systems》中的"Setting Up Disks for ZFS File Systems (Task Map)"。
- 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《System Administration Guide: Devices and File Systems》中的"EFI (GPT) Disk Label"。

可以使用全路径（如 `/dev/dsk/c1t0d0`）或构成 `/dev/dsk` 目录中设备名称的缩略名称（如 `c1t0d0`）来指定磁盘。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

当使用一个磁盘分片创建存储池时，可以为磁盘加上 Solaris VTOC (SMI) 标签，但不建议对一个池使用多个磁盘分片，因为多个磁盘分片的管理将更加困难。

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。
  - 可以使用基于软件的卷管理器（如 Solaris Volume Manager (SVM) 或 Veritas Volume Manager (VxVM)）所提供的卷来为 ZFS 构建逻辑设备。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。

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

ZFS 存储池的复制功能

ZFS 在镜像配置和 RAID-Z 配置中提供数据冗余和自我修复属性。

- 第 40 页中的 “镜像存储池配置”
- 第 41 页中的 “RAID-Z 存储池配置”
- 第 42 页中的 “冗余配置中的自我修复数据”
- 第 42 页中的 “存储池中的动态条带化”
- 第 42 页中的 “ZFS 混合存储池”

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，简单的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，请参见第 43 页中的 “创建镜像存储池”。

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 配置最好分割为两个 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，请参见第 44 页中的 “创建 RAID-Z 存储池”。

有关基于性能和磁盘空间考虑在镜像配置或 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 不仅会从另一个冗余副本中提取正确的数据，还会通过将错误数据替换为正确的副本对其进行修复。

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。由于是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：

可以快速轻松地创建和销毁池。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`。这个简单的命令会产生严重的后果。

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。池名称必须符合第 27 页中的 “ZFS 组件命名要求”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，并正确使用 ZFS 作为标签以包含单个大分片。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` 关键字表示将指定新的顶层虚拟设备。数据通过这两个镜像以动态方式进行条带化，并会相应地在每个磁盘之间创建冗余数据。

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，请参见第 53 页中的 “向存储池中添加设备”。
- 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，请参见第 57 页中的 “附加和分离存储池中的设备”。

- 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，请参见第 64 页中的 “替换存储池中的设备”。
- 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，请参见第 57 页中的 “附加和分离存储池中的设备”。
- 通过分离其中一个磁盘来分割镜像配置，以创建新的相同池。有关更多信息，请参见第 59 页中的 “通过分割镜像 ZFS 存储池创建新池”。

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

请考虑以下根池配置要求：

- 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，安装和引导 Oracle Solaris ZFS 根文件系统。

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由 5 个磁盘组成的单个 RAID-Z 设备的池：

```
# 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，使用 `raidz2` 或 `raidz3` 关键字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，请参见第 53 页中的 “向存储池中添加设备”。
- 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，请参见第 64 页中的 “替换存储池中的设备”。

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，请参见第 41 页中的 “RAID-Z 存储池配置”。

使用日志设备创建 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 和其他应用程序还可以使用 `fsync()` 来确保数据稳定性。

缺省情况下，ZIL 是通过主池中的块分配的。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，请考虑以下几点：

- 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。

以下示例显示如何使用镜像日志设备创建镜像存储池：

```
# 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，请参见示例 10–2。

使用高速缓存设备创建 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。

您可以使用高速缓存设备创建一个存储池，来缓存存储池数据。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。

确定是否使用高速缓存设备创建 ZFS 存储池时，请考虑以下几点：

- 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，请参见示例 3–4。
- 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。
- 未使用 ZFS 冗余（RAIDZ 或镜像）创建的池只能报告数据不一致性，无法修复数据不一致性。
- 虽然使用 ZFS 冗余创建池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).
```

使用 `-f` 选项可以覆盖其中的一些错误，但是无法覆盖大多数错误。使用 `-f` 选项无法覆盖下列条件，必须手动对这些错误进行更正：

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。

Live Upgrade 正在将磁盘用作 Oracle Solaris Live Upgrade 的替换引导环境。

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。

不匹配的复制级别

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。如果尝试创建具有这样配置的池，则会显示类似以下的错误：

```
# 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。要使用不同的缺省挂载点创建池，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，请参见第 171 页中的 “管理 ZFS 挂载点”。

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，请参见第 89 页中的 “恢复已销毁的 ZFS 存储池”。

如果使用 `zpool destroy` 命令销毁池，该池仍可用于导入，如第 89 页中的 “恢复已销毁的 ZFS 存储池”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。如果希望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。

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，请参见第 78 页中的 “确定 ZFS 存储池的运行状况”。

有关导入池的更多信息，请参见第 86 页中的 “导入 ZFS 存储池”。

管理 ZFS 存储池中的设备

第 37 页中的 “ZFS 存储池的组件”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。

- 第 53 页中的 “向存储池中添加设备”
- 第 57 页中的 “附加和分离存储池中的设备”
- 第 59 页中的 “通过分割镜像 ZFS 存储池创建新池”
- 第 61 页中的 “使存储池中的设备联机和脱机”
- 第 63 页中的 “清除存储池设备错误”
- 第 64 页中的 “替换存储池中的设备”
- 第 66 页中的 “在存储池中指定热备件”

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。将对设备进行检查以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，请参见第 49 页中的 “检测使用中的设备”。

示例 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
```

示例 3–1 向镜像 ZFS 配置中添加磁盘 （续）

```
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
```

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

示例 3–2 向 RAID-Z 配置中添加磁盘 （续）

```
errors: No known data errors
```

示例 3–3 添加和删除镜像日志设备

以下示例展示了如何将镜像日志设备添加到镜像存储池。

```
# 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       0     0     0
```

示例 3–3 添加和删除镜像日志设备 （续）

```
          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
```

示例 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。

示例 3–4 添加和删除高速缓存设备 （续）

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，请参见第 47 页中的 “使用高速缓存设备创建 ZFS 存储池”。

附加和分离存储池中的设备

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，请参见第 102 页中的 “如何创建镜像 ZFS 根池（安装后）”。

如果要替换 ZFS 根池中的磁盘，请参见第 143 页中的 “恢复 ZFS 根池或根池快照”。

示例 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
```

示例 3–5 将双向镜像存储池转换为三向镜像存储池 （续）

```
          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。

示例 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
```

但是，如果不存在该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
```

通过分割镜像 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。您可以使用此功能分隔镜像根池，但是分隔出来的池不可引导，除非另外执行其他步骤。

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。

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。分割操作完成后，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
```

发生实际的分割操作之前，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。新池 GUID 即产生，以便能在分割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，则无法分割池。
- 分割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（分隔）池是非冗余的。
- 如果现有池是一个三向池，则在分割操作之后，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。
- 在分割操作期间，保持数据冗余性的一个好方法是分割由三个磁盘组成的镜像存储池，这样在分割操作之后，原池由两个镜像磁盘组成。
- 在分割镜像池之前，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，请参见第 275 页中的 “一般系统做法”。

示例 3–7 分割镜像 ZFS 存储池 (zpool split)

在以下示例中，对名为 mothership 的镜像存储池（包含三个磁盘）进行分割。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。

示例 3–7 分割镜像 ZFS 存储池 (zpool split) （续）

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 不会向已脱机的设备发送任何请求。

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，请考虑以下几点：

- 不能将池脱机到它成为 `UNAVAIL` 的点。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，请参见第 64 页中的 “替换存储池中的设备”。

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，请参见第 73 页中的 “查询 ZFS 存储池的状态”。

有关设备运行状况的更多信息，请参见第 78 页中的 “确定 ZFS 存储池的运行状况”。

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 c1t0d0
warning: device 'c1t0d0' onlined, but remains in faulted state
use 'zpool replace' to replace devices that are no longer present
```

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-D3, TYPE: Fault, VER: 1, SEVERITY: Major
EVENT-TIME: Wed Jun 20 11:35:26 MDT 2012
PLATFORM: SUNW,Sun-Fire-880, CSN: -, HOSTNAME: neo
SOURCE: zfs-diagnosis, REV: 1.0
EVENT-ID: 504a1188-b270-4ab0-af4e-8a77680576b8
DESC: A ZFS device failed.  Refer to http://sun.com/msg/ZFS-8000-D3 for more information.
AUTO-RESPONSE: No automated response will occur.
IMPACT: Fault tolerance of the pool may be compromised.
REC-ACTION: Run 'zpool status -x' and replace the bad device.
```

有关更换故障设备的更多信息，请参见第 253 页中的 “解决缺少设备或设备被移除的问题”。

如果向池附加了较大的磁盘或使用较大磁盘替换了较小磁盘，可以使用 zpool online -e 命令扩展池的大小。缺省情况下，添加至池中的磁盘不会扩展到其完整大小，除非启用了 autoexpand 池属性。使用 zpool online -e 命令可以自动扩展池，即使替换磁盘已经联机或者磁盘目前脱机。例如：

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

清除存储池设备错误

如果设备因出现故障（导致在 zpool status 输出中列出错误）而脱机，则可以使用 zpool clear 命令清除错误计数。

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，则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，请参见第 258 页中的 “清除瞬态设备错误”。

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，要通过删除磁盘并在同一位置替换该磁盘来替换出现故障的磁盘 (c1t1d0)，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，请参见第 143 页中的 “恢复 ZFS 根池或根池快照”。

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，请参见示例 10–1。

替换 ZFS 存储池中的设备时，请考虑以下几点：

- 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。autoexpand 池属性的值决定向池添加了大型磁盘后池是否会扩展。缺省情况下，autoexpand 属性禁用。您可以在较大的磁盘添加到池中之前或之后，启用此属性以扩展池的大小。

  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，请参见第 68 页中的 “在存储池中激活和取消激活热备件”。

有关更换设备的更多信息，请参见第 253 页中的 “解决缺少设备或设备被移除的问题”和第 256 页中的 “更换或修复损坏的设备”。

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**意味着该设备不是池中的活动设备，但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，请考虑以下几点：

- 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
  ```

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: Attach the missing device and online it using 'zpool online'.
   see: http://www.sun.com/msg/ZFS-8000-2Q
  scan: resilvered 3.15G in 0h0m with 0 errors on Mon Nov 12 15:53:42 2012
config:

        NAME                         STATE     READ WRITE CKSUM
        zeepool                      DEGRADED     0     0     0
          mirror-0                   ONLINE       0     0     0
            c0t5000C500335F95E3d0    ONLINE       0     0     0
            c0t5000C500335F907Fd0    ONLINE       0     0     0
          mirror-1                   DEGRADED     0     0     0
            c0t5000C500335BD117d0    ONLINE       0     0     0
            spare-1                  DEGRADED   449     0     0
              c0t5000C500335DC60Fd0  UNAVAIL      0     0     0
              c0t5000C500335E106Bd0  ONLINE       0     0     0
        spares
          c0t5000C500335E106Bd0      INUSE

errors: No known data errors
```

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。请参见示例 3–8。
- 临时或永久性地换入另一热备件。请参见示例 3–9。

示例 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
```

示例 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: Attach the missing device and online it using 'zpool online'.
   see: http://www.sun.com/msg/ZFS-8000-2Q
  scan: resilver in progress since Mon Nov 12 16:04:12 2012
     4.80G scanned out of 12.0G at 55.8M/s, 0h2m to go
     4.80G scanned out of 12.0G at 55.8M/s, 0h2m to go
     4.77G resilvered, 39.97% done
config:

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

示例 3–9 分离故障磁盘并使用热备件 （续）

```
          c0t5000C500335E106Bd0    AVAIL

errors: No known data errors
# zpool detach zeepool c0t5000C500335DC60Fd0
# zpool status zeepool
  pool: zeepool
 state: ONLINE
  scan: resilvered 11.3G in 0h3m with 0 errors on Mon Nov 12 16:07:12 2012
config:

        NAME                      STATE     READ WRITE CKSUM
        zeepool                   ONLINE       0     0     0
          mirror-0                ONLINE       0     0     0
            c0t5000C500335F95E3d0 ONLINE       0     0     0
            c0t5000C500335F907Fd0 ONLINE       0     0     0
          mirror-1                ONLINE       0     0     0
            c0t5000C500335BD117d0 ONLINE       0     0     0
            c0t5000C500335E106Bd0 ONLINE       0     0     0

errors: No known data errors
(Original failed disk c0t5000C500335DC60Fd0 is physically replaced)
# zpool add zeepool spare c0t5000C500335DC60Fd0
# zpool status zeepool
  pool: zeepool
 state: ONLINE
  scan: resilvered 11.2G in 0h3m with 0 errors on Mon Nov 12 16:07:12 2012

config:

        NAME                      STATE     READ WRITE CKSUM
        zeepool                   ONLINE       0     0     0
          mirror-0                ONLINE       0     0     0
            c0t5000C500335F95E3d0 ONLINE       0     0     0
            c0t5000C500335F907Fd0 ONLINE       0     0     0
          mirror-1                ONLINE       0     0     0
            c0t5000C500335BD117d0 ONLINE       0     0     0
            c0t5000C500335E106Bd0 ONLINE       0     0     0
        spares
          c0t5000C500335DC60Fd0    AVAIL

errors: No known data errors
```

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 tank
tank  size           68G                  -
tank  capacity       0%                   -
tank  altroot        -                    default
tank  health         ONLINE               -
tank  guid           15560293364730146756 -
tank  version        32                   default
tank  bootfs         -                    default
tank  delegation     on                   default
tank  autoreplace    off                  default
tank  cachefile      -                    default
tank  failmode       wait                 default
tank  listsnapshots  on                   default
tank  autoexpand     off                  default
tank  free           68.0G                -
tank  allocated      124K                 -
tank  readonly       off                  -
```

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。检查未知池时，如果不能信任挂载点，或挂载点在备用根环境中（其中典型的路径无效），则可以使用此属性。
autoreplace	布尔值	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的同一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。

表 3–1 ZFS 池属性说明 （续）

属性名称	类型	缺省值	说明
bootfs	布尔值	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	控制在何处缓存池配置信息。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	用于标识已用池空间百分比的只读值。 此属性的缩写为 cap。
delegation	布尔值	on	控制是否可以向非特权用户授予为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	用于标识池的唯一标识符的只读属性。
health	字符串	N/A	用于标识池的当前运行状况（例如 ONLINE、DEGRADED、SUSPENDED、REMOVED 或 UNAVAIL）的只读属性。
listshares	字符串	off	控制使用 zfs list 命令时是否显示此池中的共享信息。缺省值为 off。

表 3–1 ZFS 池属性说明 （续）

属性名称	类型	缺省值	说明
listsnapshots	字符串	on	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。
size	数字	N/A	用于标识存储池总大小的只读属性。
version	数字	N/A	标识池的当前盘上版本。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。

- 第 73 页中的 “显示有关 ZFS 存储池的信息”
- 第 76 页中的 “查看 ZFS 存储池的 I/O 统计信息 ”
- 第 78 页中的 “确定 ZFS 存储池的运行状况”

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，请参见第 28 页中的 “ZFS 磁盘空间记帐”。

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，请参见第 78 页中的 “确定 ZFS 存储池的运行状况”。
ALTROOT	池的备用根（如果有）。 有关备用根池的更多信息，请参见第 244 页中的 “使用 ZFS 备用根池”。

另外，也可以通过指定池的名称来收集特定池的统计信息。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  -
```

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
```

列名称与第 73 页中的 “显示有关所有存储池或某个特定池的信息”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 会自动记录成功的 zfs 和 zpool 命令（用于修改池状态信息）。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':
2010-05-11.10:18:54 zpool create -f -o failmode=continue -R /a -m legacy -o
cachefile=/tmp/root/etc/zfs/zpool.cache rpool mirror c1t0d0s0 c1t1d0s0
2010-05-11.10:18:55 zfs set canmount=noauto rpool
2010-05-11.10:18:55 zfs set mountpoint=/rpool rpool
2010-05-11.10:18:56 zfs create -o mountpoint=legacy rpool/ROOT
2010-05-11.10:18:57 zfs create -b 8192 -V 2048m rpool/swap
2010-05-11.10:18:58 zfs create -b 131072 -V 1536m rpool/dump
2010-05-11.10:19:01 zfs create -o canmount=noauto rpool/ROOT/zfsBE
2010-05-11.10:19:02 zpool set bootfs=rpool/ROOT/zfsBE rpool
2010-05-11.10:19:02 zfs set mountpoint=/ rpool/ROOT/zfsBE
2010-05-11.10:19:03 zfs set canmount=on rpool
2010-05-11.10:19:04 zfs create -o mountpoint=/export rpool/export
2010-05-11.10:19:05 zfs create rpool/export/home
2010-05-11.11:11:10 zpool set bootfs=rpool rpool
2010-05-11.11:11:10 zpool set bootfs=rpool/ROOT/zfsBE rpool
```

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。报告的统计信息如下：

`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，请参见第 28 页中的 “ZFS 磁盘空间记帐”。

`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）。

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 out of 71.8G at 36.4M/s, 0h30m to go
    7.43G resilvered, 10.35% done
```

- 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摘要。例如：

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

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

errors: No known data errors
```

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（用于了解更多信息）的链接。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已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 tank c1t0d0
Bringing device c1t0d0 online
# zpool status -x
all pools are healthy
```

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

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

errors: No known data errors

# 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，请参见第 192 页中的 “发送和接收 ZFS 数据”。

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 ZFS 卷在池中处于使用状态，即使使用 -f 选项，也无法导出池。要导出包含 ZFS 卷的池，请首先确保卷的所有使用者都不再处于活动状态。

有关 ZFS 卷的更多信息，请参见第 237 页中的 “ZFS 卷”。

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
```

在本示例中，池 tank 可用于在目标系统中导入。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，其中包含有关禁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: 11809215114195894163
 state: DEGRADED
status: One or more devices are missing from the system.
action: The pool can be imported despite missing or damaged devices.  The
        fault tolerance of the pool may be compromised if imported.
   see: http://www.sun.com/msg/ZFS-8000-2Q
config:

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

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

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
```

如果该池的名称与现有的池名称冲突，则可以使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。

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。

也可以使用 `-R` 选项在备用根下导入池。有关备用根池的更多信息，请参见第 244 页中的 “使用 ZFS 备用根池”。

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
The devices below are missing, use '-m' to import the pool anyway:
            c3t3d0 [log]

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

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 could not be opened.  Sufficient replicas exist for
        the pool to continue functioning in a degraded state.
action: Attach the missing device and online it using 'zpool online'.
   see: http://www.sun.com/msg/ZFS-8000-2Q
  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: http://www.sun.com/msg/ZFS-8000-2Q
 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，由于包含以下状态信息，因此可以将池 tank 确定为已销毁的池：

```
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
```

如果已销毁池中的某个设备不可用，通过添加 `-f` 选项也许能够恢复已销毁的池。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 could not be opened.  Sufficient replicas exist for
        the pool to continue functioning in a degraded state.
action: Attach the missing device and online it using 'zpool online'.
   see: http://www.sun.com/msg/ZFS-8000-2Q
config:
        dozer              DEGRADED
          raidz2-0         DEGRADED
            c8t0d0         ONLINE
            c8t1d0         ONLINE
            c8t2d0         ONLINE
            c8t3d0         UNAVAIL  cannot open
            c8t4d0         ONLINE
errors: No known data errors
# zpool import -Df dozer
# zpool status -x
  pool: dozer
 state: DEGRADED
status: One or more devices could not be opened.  Sufficient replicas exist for
        the pool to continue functioning in a degraded state.
action: Attach the missing device and online it using 'zpool online'.
   see: http://www.sun.com/msg/ZFS-8000-2Q
   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` 命令会在运行旧版本池时通知您。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 22.

The following versions are supported:

VER  DESCRIPTION
---  --------------------------------------------------------
 1   Initial ZFS version
 2   Ditto blocks (replicated metadata)
 3   Hot spares and double parity RAID-Z
 4   zpool history
 5   Compression using the gzip algorithm
 6   bootfs pool property
 7   Separate intent log devices
 8   Delegated administration
 9   refquota and refreservation properties
 10  Cache devices
 11  Improved scrub performance
 12  Snapshot properties
 13  snapused property
 14  passthrough-x aclinherit
 15  user/group space accounting
 16  stmf property support
 17  Triple-parity RAID-Z
 18  Snapshot user holds
 19  Log device removal
 20  Compression using zle (zero-length encoding)
 21  Reserved
 22  Received properties
```

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

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。

如果要在包含来自先前 Solaris 发行版的池的系统上使用 ZFS 管理控制台，请确保在使用控制台之前先升级池。要确定池是否需要升级，请使用 `zpool status` 命令。

◆ ◆ ◆ 第 4 章

安装和引导 Oracle Solaris ZFS 根文件系统

本章介绍如何安装和引导 Oracle Solaris ZFS 根文件系统。同时还对使用 Oracle Solaris Live Upgrade 功能将 UFS 根文件系统迁移到 ZFS 文件系统进行了介绍。

本章包含以下各节：

有关此发行版的已知问题列表，请参见《Oracle Solaris 10 1/13 发行说明》。

安装和引导 Oracle Solaris ZFS 根文件系统（概述）

您可以通过以下方式安装 ZFS 根文件系统并从中进行引导：

- **Oracle Solaris 初始安装（交互式文本模式安装方法）**
  - 选择并安装 ZFS 作为根文件系统。
  - 安装 ZFS Flash 归档文件。
- **Oracle Solaris Live Upgrade 功能**
  - 将 UFS 根文件系统迁移到 ZFS 根文件系统。
  - 在新 ZFS 根池中创建新引导环境。
  - 在现有的 ZFS 根池中创建或更新引导环境。

  - 使用 ZFS Flash 归档文件升级备用引导环境 (Boot Environment, BE)。
- **Oracle Solaris JumpStart 功能**。
  - 创建配置文件以自动安装具有 ZFS 根文件系统的系统。
  - 创建配置文件以自动安装具有 ZFS Flash 归档文件的系统。

基于 SPARC 或基于 x86 的系统安装了 ZFS 根文件系统或迁移到 ZFS 根文件系统后，系统将从 ZFS 根文件系统自动引导。有关引导方面的变化的更多信息，请参见第 136 页中的“从 ZFS 根文件系统引导”。

ZFS 安装功能

此 Oracle Solaris 发行版中提供了以下 ZFS 安装功能：

- 使用交互式文本安装程序功能，您可以安装 UFS 根文件系统或 ZFS 根文件系统。在此发行版中，缺省文件系统仍为 UFS。您可以通过以下方式访问交互式文本安装程序：
  - SPARC：对于 Oracle Solaris 安装 DVD，请使用以下语法：

    ```
    ok boot cdrom - text
    ```
  - SPARC：从网络引导时，请使用以下语法：

    ```
    ok boot net - text
    ```
  - x86：选择文本模式安装方法。
- 定制 JumpStart 配置文件提供以下功能：
  - 您可以设置配置文件以创建 ZFS 存储池并指定可引导的 ZFS 文件系统。
  - 您可以设置配置文件来安装 ZFS 根池的 Flash 归档文件。
- 使用 Live Upgrade，您可以将 UFS 根文件系统迁移到 ZFS 根文件系统。
- 可以通过在安装期间选择两个磁盘来设置镜像 ZFS 根池。或者，可以通过在安装后附加其他磁盘来创建镜像 ZFS 根池。
- 将会在 ZFS 根池中的 ZFS 卷上自动创建交换和转储设备。

此发行版中不提供以下安装功能：

- 当前未提供用于安装 ZFS 根文件系统的 GUI 安装功能。您必须选择文本模式安装方法来安装 ZFS 根文件系统。
- 不能使用标准升级程序将 UFS 根文件系统升级到 ZFS 根文件系统。

支持 ZFS 所要满足的 Oracle Solaris 安装要求和 Oracle Solaris Live Upgrade 要求

尝试安装具有 ZFS 根文件系统的系统或将 UFS 根文件系统迁移到 ZFS 根文件系统之前，请确保满足以下要求。

Oracle Solaris 发行版要求

您可以通过以下方式安装和引导 ZFS 根文件系统或迁移到 ZFS 根文件系统：

- 安装 ZFS 根文件系统－从 Solaris 10 10/08 发行版开始可用。
- 使用 Live Upgrade 从 UFS 根文件系统迁移到 ZFS 根文件系统－必须已安装了 Solaris 10 10/08 或以上的发行版，或已升级到 Solaris 10 10/08 或以上的发行版。

ZFS 根池的一般要求

以下部分介绍 ZFS 根池空间和配置要求。

ZFS 根池的磁盘空间要求

ZFS 根文件系统所需的最小可用池空间量大于 UFS 根文件系统所需的最小可用池空间量，因为交换设备和转储设备在 ZFS 根环境中必须是单独的设备。缺省情况下，交换和转储设备在 UFS 根文件系统中为同一设备。

在系统中安装 ZFS 根文件系统或将其升级为 ZFS 根文件系统时，交换区域和转储设备的大小取决于物理内存量。可引导的 ZFS 根文件系统的最小可用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。

检查 ZFS 存储池的以下磁盘空间要求：

- 安装 ZFS 根文件系统所需的最小内存量为 1536 MB。
- 要获得更好的性能（ZFS 的全部性能），建议使用 1536 MB 或更大容量的内存。
- 建议至少使用 16 GB 的磁盘空间。磁盘空间的使用情况如下所述：
  - **交换区域和转储设备**－Oracle Solaris 安装程序创建的交换卷和转储卷的缺省大小如下：
    - **初始安装**－在新的 ZFS 引导环境中，缺省交换卷大小按照物理内存的一半计算，一般在 512 MB 至 2 GB 的范围内。可以在初始安装过程中调整交换卷的大小。
    - 缺省转储卷的大小由内核基于 `dumpadm` 信息和物理内存大小进行计算。可以在初始安装过程中调整转储卷的大小。
    - **Live Upgrade**－将 UFS 根文件系统迁移到 ZFS 根文件系统时，ZFS BE（Boot Environment，引导环境）的缺省交换卷大小按 UFS BE 的交换设备的大小来计算。计算缺省交换卷大小时，会将 UFS BE 中所有交换设备的大小加总，并在 ZFS BE 中创建一个该大小的 ZFS 卷。如果 UFS BE 中未定义交换设备，则缺省交换卷大小设为 512 MB。
    - 在 ZFS BE 中，缺省转储卷大小设置为物理内存的一半，范围介于 512 MB 至 2 GB 之间。

    可以将交换卷和转储卷的大小调整为所选择的大小，只要新的大小支持系统运作。有关更多信息，请参见第 134 页中的 “调整 ZFS 交换设备和转储设备的大小”。

- **引导环境 (Boot Environment, BE)** – 除了新的交换和转储空间要求或调整的交换和转储设备大小外，从 UFS BE 迁移的 ZFS BE 还需要大约 6 GB 空间。从其他 ZFS BE 克隆的每个 ZFS BE 都不需要额外的磁盘空间，但是请考虑到以下情况：应用修补程序时，BE 大小会增加。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** – 作为 OS 映像一部分的根文件系统的所有子目录（除 `/var` 之外）必须与根文件系统处于同一数据集。此外，除了交换和转储设备之外，所有其他 OS 组件必须驻留在根池中。

  有关使用 Live Upgrade 更改缺省交换和转储设备的信息，请参见第 135 页中的 “定制 ZFS 交换卷和转储卷”。

  另一个限制是 `/var` 目录或数据集必须是单个数据集。例如，如果您还想使用 Live Upgrade 迁移或修补 ZFS BE，或者创建此池的 ZFS Flash 归档文件，则无法创建后代 `/var` 数据集，如 `/var/tmp`。

例如，磁盘空间为 12 GB 的系统对于可引导的 ZFS 环境来说可能会太小，因为每个交换和转储设备都需要 2 GB 磁盘空间，而且从 UFS BE 迁移的 ZFS BE 需要大约 6 GB 磁盘空间。

ZFS 根池配置要求

请查看以下 ZFS 根池配置要求：

- 要用作根池的池必须具有 SMI 标签。使用磁盘分片创建池时，通常可以满足此要求。
- 池必须存在于磁盘分片或被镜像的磁盘分片上。如果在 Live Upgrade 迁移期间试图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，请参见第 44 页中的 “创建 ZFS 根池”。
- x86：磁盘必须包含 Oracle Solaris `fdisk` 分区。该 `fdisk` 分区是在安装基于 x86 的系统时自动创建的。有关 Solaris `fdisk` 分区的更多信息，请参见《System Administration Guide: Devices and File Systems》中的“Guidelines for Creating an fdisk Partition”。
- 在基于 SPARC 和基于 x86 的系统上，ZFS 根池中指定用于引导的磁盘必须小于 2 TB。
- 只有在安装完根池后，才能在根池中启用压缩。在安装期间，无法在根池中启用压缩。根池不支持 `gzip` 压缩算法。
- 初始安装创建根池后，或者在 Solaris Live Upgrade 迁移到 ZFS 根文件系统后，请勿重命名根池。重命名根池可能会导致系统无法引导。

  此外，如果要使用 Live Upgrade，请勿更改根池组件的缺省挂载点。
- 如果要更改交换和转储设备并使用 Live Upgrade，请查看第 135 页中的 “定制 ZFS 交换卷和转储卷”。

安装 ZFS 根文件系统（Oracle Solaris 初始安装）

在本 Oracle Solaris 发行版中，您可以使用以下方法执行初始安装：

- 使用交互式文本安装程序安装包含可引导 ZFS 根文件系统的新 ZFS 存储池。如果希望将现有 ZFS 存储池用于 ZFS 根文件系统，则必须使用 Live Upgrade 将现有 UFS 根文件系统迁移到现有 ZFS 存储池中，通过迁移建立 ZFS 根文件系统。有关更多信息，请参见第 110 页中的 “迁移到 ZFS 根文件系统或更新 ZFS 根文件系统 (Live Upgrade)”。
- 使用交互式文本安装程序从 ZFS Flash 归档文件安装包含可引导 ZFS 根文件系统的新 ZFS 存储池。

在开始初始安装以创建 ZFS 存储池之前，请查看第 94 页中的 “支持 ZFS 所要满足的 Oracle Solaris 安装要求和 Oracle Solaris Live Upgrade 要求”。

如果要在完成 ZFS 根文件系统的初始安装后配置区域，并且计划修补或升级系统，请参见第 119 页中的 “使用 Live Upgrade 迁移或升级具有区域的系统 (Solaris 10 10/08)”或第 123 页中的 “使用 Oracle Solaris Live Upgrade 迁移或升级具有区域的系统（最低 Solaris 10 5/09）”。

如果系统上已经具有 ZFS 存储池，则会通过以下消息告知。然而，这些池保持不动，除非您选择现有池中的磁盘来创建新存储池。

```
There are existing ZFS pools available on this system.  However, they can only be upgraded
using the Live Upgrade tools.  The following screens will only allow you to install a ZFS root system,
not upgrade one.
```

注意 – 如果选择将现有池中的任何磁盘用于新池，则现有池将被销毁。

示例 4–1 可引导的 ZFS 根文件系统的初始安装

交互式文本安装过程与先前的 Oracle Solaris 发行版中的过程基本相同，区别在于系统会提示您创建 UFS 还是 ZFS 根文件系统。在此发行版中，缺省文件系统仍为 UFS。如果选择 ZFS 根文件系统，系统会提示您创建 ZFS 存储池。安装 ZFS 根文件系统的步骤如下：

1. 插入 Oracle Solaris 安装介质或从安装服务器引导系统，然后选择交互式文本安装方法创建可引导的 ZFS 根文件系统。
   - SPARC：对于 Oracle Solaris 安装 DVD，请使用以下语法：

     ```
     ok boot cdrom - text
     ```
   - SPARC：从网络引导时，请使用以下语法：

     ```
     ok boot net - text
     ```
   - x86：选择文本模式安装方法。

   您还可以使用以下方法创建要安装的 ZFS Flash 归档文件：

示例 4–1 可引导的 ZFS 根文件系统的初始安装 （ 续 ）

- JumpStart 安装。有关更多信息，请参见示例 4–2。
- 初始安装。有关更多信息，请参见示例 4–3。

您可以执行标准升级来升级现有的可引导 ZFS 文件系统，但不能使用此选项创建新的可引导 ZFS 文件系统。从 Solaris 10 10/08 发行版开始，只要已安装 Solaris 10 10/08 或以上的发行版，就可以将 UFS 根文件系统迁移到 ZFS 根文件系统。有关迁移到 ZFS 根文件系统的更多信息，请参见第 110 页中的 “迁移到 ZFS 根文件系统或更新 ZFS 根文件系统 (Live Upgrade)”。

2. 要创建 ZFS 根文件系统，请选择 ZFS 选项。例如：

```
Choose Filesystem Type

  Select the filesystem to use for your Solaris installation


           [ ] UFS
           [X] ZFS
```

3. 选择要安装的软件后，系统会提示您选择要用于创建 ZFS 存储池的磁盘。该屏幕与先前发行版中的屏幕类似。

```
Select Disks
  On this screen you must select the disks for installing Solaris software.
  Start by looking at the Suggested Minimum field; this value is the
  approximate space needed to install the software you've selected. For ZFS,
  multiple disks will be configured as mirrors, so the disk you choose, or the
  slice within the disk must exceed the Suggested Minimum value.
  NOTE: ** denotes current boot disk

  Disk Device                                                Available Space
  ============================================================================
  [X] ** c1t0d0                                            139989 MB  (F4 to edit
) [ ]    c1t1d0                                            139989 MB
  [ ]    c1t2d0                                            139989 MB
  [ ]    c1t3d0                                            139989 MB
  [ ]    c2t0d0                                            139989 MB
  [ ]    c2t1d0                                            139989 MB
  [ ]    c2t2d0                                            139989 MB
  [ ]    c2t3d0                                            139989 MB
                                 Maximum Root Size: 139989 MB
                                 Suggested Minimum:  11102 MB
```

您可以选择要用于 ZFS 根池的一个或多个磁盘。如果选择两个磁盘，则会为根池设置镜像双磁盘配置。双磁盘或三磁盘镜像池为最佳。如果您有八个磁盘并选择了所有磁盘，则这八个磁盘将作为一个大的镜像用于根池。此配置并非最佳。还可以选择在完成初始安装后创建镜像根池。不支持对根池的 RAID-Z 池配置。

有关配置 ZFS 存储池的更多信息，请参见第 40 页中的 “ZFS 存储池的复制功能”。

4. 要选择两个磁盘来创建镜像的根池，请使用光标控制键选择第二个磁盘。

示例 4–1 可引导的 ZFS 根文件系统的初始安装 （续）

在以下示例中，选择了 `c1t0d0` 和 `c1t1d0` 用作根池磁盘。这两个磁盘都必须具有 SMI 标签和分片 0。如果磁盘没有 SMI 标签或不包含分片，则必须退出安装程序，使用 `format` 实用程序重新对磁盘设置标签和分区，然后重新启动安装程序。

```
Select Disks
  On this screen you must select the disks for installing Solaris software.
  Start by looking at the Suggested Minimum field; this value is the
  approximate space needed to install the software you've selected. For ZFS,
  multiple disks will be configured as mirrors, so the disk you choose, or the
  slice within the disk must exceed the Suggested Minimum value.
  NOTE: ** denotes current boot disk

  Disk Device                                                Available Space
  ============================================================================
  [X] ** c1t0d0                                            139989 MB  (F4 to edit
) [X]    c1t1d0                                            139989 MB
  [ ]    c1t2d0                                            139989 MB
  [ ]    c1t3d0                                            139989 MB
  [ ]    c2t0d0                                            139989 MB
  [ ]    c2t1d0                                            139989 MB
  [ ]    c2t2d0                                            139989 MB
  [ ]    c2t3d0                                            139989 MB

                                  Maximum Root Size: 139989 MB
                                  Suggested Minimum:  11102 MB
```

如果 "Available Space"（可用空间）栏显示 0 MB，则该磁盘很有可能具有 EFI 标签。如果您想使用带 EFI 标签的磁盘，则必须退出安装程序，使用 `format -e` 命令以 SMI 标签重新标记该磁盘，然后重新启动安装程序。

如果在安装期间没有创建镜像根池，您可以在安装之后轻松创建。有关信息，请参见第 102 页中的 “如何创建镜像 ZFS 根池（安装后）”。

为 ZFS 存储池选择了一个或多个磁盘后，将出现一个屏幕，显示类似如下的内容：

```
Configure ZFS Settings
  Specify the name of the pool to be created from the disk(s) you have chosen.
  Also specify the name of the dataset to be created within the pool that is
  to be used as the root directory for the filesystem.


              ZFS Pool Name: rpool
      ZFS Root Dataset Name: s10nameBE
      ZFS Pool Size (in MB): 139990
  Size of Swap Area (in MB): 4096
  Size of Dump Area (in MB): 1024
        (Pool size must be between 7006 MB and 139990 MB)

                          [X] Keep / and /var combined
                          [ ] Put /var on a separate dataset
```

5. 在此屏幕中，您可以通过在各项之间移动光标控制键并使用新值替换缺省值来更改 ZFS 池名称、数据集名称、池大小以及交换和转储设备大小（可选操作）。或者，您可以接受缺省值。另外，您可以修改创建和挂载 `/var` 文件系统的方式。

示例 4–1 可引导的 ZFS 根文件系统的初始安装 （续）

在此示例中，根数据集名称更改为 zfsBE。

```
             ZFS Pool Name: rpool
     ZFS Root Dataset Name: zfsBE
     ZFS Pool Size (in MB): 139990
  Size of Swap Area (in MB): 4096
  Size of Dump Area (in MB): 1024
       (Pool size must be between 7006 MB and 139990 MB)
```

6. 在此最后的安装屏幕中，您可以更改安装配置文件（可选操作）。例如：

```
Profile

  The information shown below is your profile for installing Solaris software.
  It reflects the choices you've made on previous screens.

  ============================================================================

               Installation Option: Initial
                       Boot Device: c1t0d0
             Root File System Type: ZFS
                   Client Services: None

                           Regions: North America
                     System Locale: C ( C )

                          Software: Solaris 10, Entire Distribution
                         Pool Name: rpool
             Boot Environment Name: zfsBE
                         Pool Size: 139990 MB
                   Devices in Pool: c1t0d0
                                    c1t1d0
```

7. 安装完成后，查看生成的 ZFS 存储池和文件系统信息。例如：

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

        NAME          STATE     READ WRITE CKSUM
        rpool         ONLINE       0     0     0
          mirror-0    ONLINE       0     0     0
            c1t0d0s0  ONLINE       0     0     0
            c1t1d0s0  ONLINE       0     0     0

errors: No known data errors
# zfs list
NAME                USED  AVAIL  REFER  MOUNTPOINT
rpool              10.1G   124G   106K  /rpool
rpool/ROOT         5.01G   124G    31K  legacy
rpool/ROOT/zfsBE   5.01G   124G  5.01G  /
rpool/dump         1.00G   124G  1.00G  -
rpool/export         63K   124G    32K  /export
rpool/export/home    31K   124G    31K  /export/home
rpool/swap         4.13G   124G  4.00G  -
```

示例 4–1 可引导的 ZFS 根文件系统的初始安装 （续）

zfs list 输出样例标识了根池组件，例如 rpool/ROOT 目录，该目录在缺省情况下不可访问。

8. 要在同一存储池中创建其他 ZFS 引导环境 (Boot Environment, BE)，可以使用 lucreate 命令。

   在以下示例中，创建了名为 zfs2BE 的新 BE。当前 BE 命名为 zfsBE，如 zfs list 输出所示。但是，lustatus 输出不会确认当前 BE，直到新 BE 创建完成。

```
# lustatus
ERROR: No boot environments are configured on this system
ERROR: cannot determine list of all boot environment names
```

   如果在同一池中创建新 ZFS BE，请使用类似如下的语法：

```
# lucreate -n zfs2BE
INFORMATION: The current boot environment is not named - assigning name <zfsBE>.
Current boot environment is named <zfsBE>.
Creating initial configuration for primary boot environment <zfsBE>.
The device </dev/dsk/c1t0d0s0> is not a root device for any boot environment; cannot get BE ID.
PBE configuration successful: PBE name <zfsBE> PBE Boot Device </dev/dsk/c1t0d0s0>.
Comparing source boot environment <zfsBE> file systems with the file
system(s) you specified for the new boot environment. Determining which
file systems should be in the new boot environment.
Updating boot environment description database on all BEs.
Updating system configuration files.
Creating configuration for boot environment <zfs2BE>.
Source boot environment is <zfsBE>.
Creating boot environment <zfs2BE>.
Cloning file systems from boot environment <zfsBE> to create boot environment <zfs2BE>.
Creating snapshot for <rpool/ROOT/zfsBE> on <rpool/ROOT/zfsBE@zfs2BE>.
Creating clone for <rpool/ROOT/zfsBE@zfs2BE> on <rpool/ROOT/zfs2BE>.
Setting canmount=noauto for </> in zone <global> on <rpool/ROOT/zfs2BE>.
Population of boot environment <zfs2BE> successful.
Creation of boot environment <zfs2BE> successful.
```

   在同一池内创建 ZFS BE 时，使用 ZFS 克隆和快照功能可立即创建 BE。有关使用 Live Upgrade 进行 ZFS 根迁移的更多详细信息，请参见第 110 页中的 “迁移到 ZFS 根文件系统或更新 ZFS 根文件系统 (Live Upgrade)”。

9. 接下来，验证新引导环境。例如：

```
# lustatus
Boot Environment           Is       Active Active    Can    Copy
Name                       Complete Now    On Reboot Delete Status
-------------------------- -------- ------ --------- ------ ----------
zfsBE                      yes      yes    yes       no     -
zfs2BE                     yes      no     no        yes    -
# zfs list
NAME                      USED  AVAIL  REFER  MOUNTPOINT
rpool                    10.1G   124G   106K  /rpool
rpool/ROOT               5.00G   124G    31K  legacy
rpool/ROOT/zfs2BE         218K   124G  5.00G  /
rpool/ROOT/zfsBE         5.00G   124G  5.00G  /
rpool/ROOT/zfsBE@zfs2BE   104K      -  5.00G  -
```

示例 4–1 可引导的 ZFS 根文件系统的初始安装 （续）

```
rpool/dump               1.00G   124G  1.00G  -
rpool/export               63K   124G    32K  /export
rpool/export/home          31K   124G    31K  /export/home
rpool/swap               4.13G   124G  4.00G  -
```

10. 要从备用 BE 引导，请使用 `luactivate` 命令。

    - SPARC－如果引导设备包含 ZFS 存储池，请使用 `boot -L` 命令确定可用的 BE。

      例如，在基于 SPARC 的系统上，可使用 `boot - L` 命令显示可用的 BE 列表。要从新 BE `zfs2BE` 引导，请选择选项 2。然后，键入显示的 `boot -Z` 命令。

```
ok boot -L
Executing last command: boot -L
Boot device: /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/disk@0  File and args: -L
1 zfsBE
2 zfs2BE
Select environment to boot: [ 1 - 2 ]: 2

To boot the selected entry, invoke:
boot [<root-device>] -Z rpool/ROOT/zfs2BE
ok boot -Z rpool/ROOT/zfs2BE
```

    - x86－通过 GRUB 菜单确定要引导的 BE。

有关引导 ZFS 文件系统的更多信息，请参见第 136 页中的 “从 ZFS 根文件系统引导”。

▼ 如何创建镜像 ZFS 根池（ 安装后 ）

如果在安装期间没有创建 ZFS 镜像根池，可以在安装之后轻松创建。

有关替换根池中的磁盘的信息，请参见第 143 页中的 “如何替换 ZFS 根池中的磁盘”。

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
          c1t0d0s0  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 c1t0d0s0 c1t1d0s0
Make sure to wait until resilver is done before rebooting.
```

3 **查看根池状态，确认重新同步已完成。**

```
# zpool status rpool
  pool: rpool
 state: ONLINE
status: One or more devices is currently being resilvered.  The pool will
        continue to function, possibly in a degraded state.
action: Wait for the resilver to complete.
 scrub: resilver in progress for 0h1m, 24.26% done, 0h3m to go
config:

        NAME          STATE     READ WRITE CKSUM
        rpool         ONLINE       0     0     0
          mirror-0    ONLINE       0     0     0
            c1t0d0s0  ONLINE       0     0     0
            c1t1d0s0  ONLINE       0     0     0  3.18G resilvered

errors: No known data errors
```

在前面的输出中，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 10.0G in 0h10m with 0 errors on Thu Nov 15 12:48:33 2012
```

4 **验证您是否可以从第二个磁盘成功引导。**

5 **如有必要，设置系统自动从新磁盘引导。**

- SPARC－在 SPARC 引导 PROM 下使用 `eeprom` 命令或 `setenv` 命令重置缺省引导设备。
- x86－重新配置系统 BIOS。

安装 ZFS 根文件系统（Oracle Solaris Flash 归档文件安装）

从 Solaris 10 10/09 发行版开始，可以在具有 UFS 根文件系统或 ZFS 根文件系统的系统中创建 Flash 归档文件。ZFS 根池的 Flash 归档文件包含整个池分层结构，但交换卷和转储卷以及任何已排除的数据集除外。交换卷和转储卷是在安装 Flash 归档文件时创建的。您可以按如下方式使用 Flash 归档文件安装方法：

- 创建一个 Flash 归档文件，该归档文件可用于安装和引导具有 ZFS 根文件系统的系统。
- 使用 ZFS Flash 归档文件执行 JumpStart 安装或克隆系统的初始安装。创建 ZFS Flash 归档文件会克隆整个根池，而不是各个引导环境。将 `-D` 选项用于 `flarcreate` 和 `flar` 命令可以排除池内的单个数据集。

在考虑使用 ZFS Flash 归档文件安装系统之前，请先查看以下限制：

- 从 Oracle Solaris 10 8/11 发行版开始，可以使用交互式安装的 Flash 归档文件选项安装具有 ZFS 根文件系统的系统。此外，可以使用 Flash 归档文件通过 `luupgrade` 命令更新备用的 ZFS BE。

  - 运行 Solaris 10 9/10 发行版的系统必须添加修补程序 124630-51 (SPARC) 或修补程序 124631-51 (x86) 才可将 Flash 归档文件安装到 ABE。
  - 创建 Flash 归档文件的主系统和要安装 Flash 归档文件的克隆系统必须处于相同的内核修补程序级别。例如，如果在运行 Solaris 10 8/11 发行版的系统上创建了一个 ZFS Flash 归档文件，请确保克隆系统的内核修补程序级别也是 Solaris 10 8/11。否则，Flash 归档文件安装会失败，并且 `zfs receive` 命令会返回错误。
  - 运行 Solaris 10 9/10 发行版的具有后代根文件系统的主系统（如单独的 `/var` 文件系统）应先升级到 Solaris 10 8/11 发行版，然后再创建 Flash 归档文件和将其应用到 ABE。否则，Flash 归档文件安装将失败。
- 只能在体系结构与创建 ZFS Flash 归档文件的系统相同的系统上安装 Flash 归档文件。例如，在 `sun4v` 系统上创建的归档文件无法安装到 `sun4u` 系统上。
- 只支持使用 ZFS Flash 归档文件执行完整初始安装。无法安装 ZFS 根文件系统的 Flash 差别归档文件，也无法安装混合 UFS/ZFS 归档文件。
- 从 Solaris 10 8/11 发行版开始，可以使用 UFS Flash 归档文件安装 ZFS 根文件系统。例如：
  - 如果在 JumpStart 配置文件中使用 `pool` 关键字，UFS Flash 归档文件将安装到 ZFS 根池中。

    ```
    pool rpool auto auto auto mirror c0t0d0s0 c0t1d0s0
    ```
  - 使用 UFS Flash 归档文件执行交互式安装期间，选择 ZFS 作为文件系统类型。
- 虽然归档并安装了整个根池（任何显示排除的数据集除外），但是在安装 Flash 归档文件后，只有在创建归档文件时引导的 ZFS BE 才可用。但是，使用 `flarcreate` 或 `flar` 命令的 `-R` *rootdir* 选项归档池时，可以对当前引导的根池以外的其他根池进行归档。
- ZFS Flash 归档文件不支持用于包括和排除单个文件的 `flarcreate` 和 `flar` 命令选项。您只能从 ZFS Flash 归档文件中排除整个数据集。
- ZFS Flash 归档文件不支持 `flar info` 命令。例如：

  ```
  # flar info -l zfs10upflar
  ERROR: archive content listing not supported for zfs archives.
  ```

在主系统上安装或升级到 Solaris 10 10/09 或以上的发行版后，您可以创建用于安装目标系统的 ZFS Flash 归档文件。基本过程如下：

- 在主系统上使用 `flarcreate` 命令创建 ZFS Flash 归档文件。根池中的所有数据集（除了交换卷和转储卷）都包括在 ZFS Flash 归档文件中。
- 在安装服务器上创建一个包括 Flash 归档文件信息的 JumpStart 配置文件。
- 在目标系统上安装 ZFS Flash 归档文件。

使用 Flash 归档文件安装 ZFS 根池时，支持以下归档选项：

- 使用 `flarcreate` 或 `flar` 命令从指定的 ZFS 根池创建 Flash 归档文件。如果未指定，则会创建缺省根池的 Flash 归档文件。

- 使用 `flarcreate -D` *dataset* 从 Flash 归档文件中排除指定的数据集。可以多次使用此选项来排除多个数据集。

安装完 ZFS Flash 归档文件后，将按如下方式配置系统：

- 创建了 Flash 归档文件的系统上的整个数据集分层结构会在目标系统上重新创建，但在创建归档文件时明确排除的任何数据集除外。交换卷和转储卷不包括在 Flash 归档文件中。
- 根池名称与用于创建归档文件的池的名称相同。
- 创建 Flash 归档文件时处于活动状态的 BE 在部署系统上是活动 BE，同时也是缺省 BE。

示例 4–2 使用 ZFS Flash 归档文件安装系统（JumpStart 安装）

在主系统上安装或升级到 Solaris 10 10/09 或以上的发行版后，就可创建 ZFS 根池的 Flash 归档文件。例如：

```
# flarcreate -n zfsBE zfs10upflar
Full Flash
Checking integrity...
Integrity OK.
Running precreation scripts...
Precreation scripts done.
Determining the size of the archive...
The archive will be approximately 6.77GB.
Creating the archive...
Archive creation complete.
Running postcreation scripts...
Postcreation scripts done.

Running pre-exit scripts...
Pre-exit scripts done.
```

在要用作安装服务器的系统上，创建用于安装任何系统的 JumpStart 配置文件。例如，以下配置文件用于安装 `zfs10upflar` 归档文件。

```
install_type flash_install
archive_location nfs system:/export/jump/zfs10upflar
partitioning explicit
pool rpool auto auto auto mirror c0t1d0s0 c0t0d0s0
```

示例 4–3 可引导的 ZFS 根文件系统的初始安装（Flash 归档文件安装）

您可以选择 Flash 安装选项来安装 ZFS 根文件系统。此选项假定 ZFS Flash 归档文件已经创建并且可用。

1. 在 "Solaris Interactive Installation"（Solaris 交互式安装）屏幕中，选择 F4_Flash 选项。
2. 在 "Reboot After Installation"（安装后重新引导吗?）屏幕中，选择 "Auto Reboot"（自动重新引导）或 "Manual Reboot"（手动重新引导）选项。
3. 在 "Choose Filesystem Type"（选择文件系统类型）屏幕中，选择 ZFS。

示例 4–3 可引导的 ZFS 根文件系统的初始安装（Flash 归档文件安装） **（续）**

4. 在 "Flash Archive Retrieval Method"（Flash 归档检索方法）屏幕中，选择检索方法，如 "HTTP"、"FTP"、"NFS"、"Local File"（本地文件）、"Local Tape"（本地磁带）或 "Local Device"（本地设备）。

   例如，如果 ZFS Flash 归档文件是从 NFS 服务器共享的，请选择 NFS。

5. 在 "Flash Archive Addition"（Flash 归档附加）屏幕中，指定 ZFS Flash 归档文件的位置。

   例如，如果位置是 NFS 服务器，请使用其 IP 地址标识该服务器，然后指定到 ZFS Flash 归档文件的路径。

```
NFS Location: 12.34.567.890:/export/zfs10upflar
```

6. 在 "Flash Archive Selection"（Flash 归档选项）屏幕中，确认检索方法和 ZFS BE 名称。

```
Flash Archive Selection

  You selected the following Flash archives to use to install this system.  If
  you want to add another archive to install select "New".

        Retrieval Method      Name
        ========================================================================
        NFS                   zfsBE
```

7. 查看接下来的一组屏幕（与初始安装类似），然后选择与您的配置相符的选项。
   - Select Disks（选择磁盘）
   - Preserve Data?（保留数据吗?）
   - Configure ZFS Settings（配置 ZFS 设置）

     查看摘要信息，然后选择 "Continue"（继续）选项。

     例如：

```
Configure ZFS Settings
  Specify the name of the pool to be created from the disk(s) you have chosen.
  Also specify the name of the dataset to be created within the pool that is
  to be used as the root directory for the filesystem.


              ZFS Pool Name: rpool
      ZFS Root Dataset Name: s10zfsBE
      ZFS Pool Size (in MB): 69995
  Size of Swap Area (in MB): 2048
  Size of Dump Area (in MB): 1024
        (Pool size must be between 7591 MB and 69995 MB)
```

     如果 Flash 归档文件是 ZFS 发送流，将不显示合并的或单独的 `/var` 文件系统选项。在这种情况下，`/var` 是否合并取决于它在主系统上的配置方式。

   - 在 "Mount Remote File Systems?"（装配远程文件系统吗?）屏幕中，按下 "Continue"（继续）。

示例 4–3 可引导的 ZFS 根文件系统的初始安装（Flash 归档文件安装） （续）

- 查看 "Profile"（配置文件）屏幕，然后按 F4 进行任何更改。否则按 Begin_Installation (F2)。

  例如：

```
Profile

  The information shown below is your profile for installing Solaris software.
  It reflects the choices you've made on previous screens.

  ============================================================================

              Installation Option: Flash
                      Boot Device: c1t0d0
            Root File System Type: ZFS
                 Client Services: None

                        Software: 1 Flash Archive
                                    NFS: zfsBE
                       Pool Name: rpool
          Boot Environment Name: s10zfsBE
                       Pool Size: 69995 MB
                 Devices in Pool: c1t0d0
```

安装 ZFS 根文件系统（JumpStart 安装）

可以创建 JumpStart 配置文件以安装 ZFS 根文件系统或 UFS 根文件系统。

ZFS 特定的 JumpStart 配置文件必须包含新的 `pool` 关键字。`pool` 关键字会安装一个新的根池，并缺省创建一个新的引导环境 (Boot Environment, BE)。您可以使用 `bootenv installbe` 关键字以及 `bename` 和 `dataset` 选项提供 BE 的名称和创建单独的 `/var` 数据集。

有关使用 JumpStart 功能的一般信息，请参见《Oracle Solaris 10 1/13 安装指南：JumpStart 安装》。

如果要在完成 ZFS 根文件系统的 JumpStart 安装后配置区域，并且计划修补或升级系统，请参见第 119 页中的 “使用 Live Upgrade 迁移或升级具有区域的系统 (Solaris 10 10/08)”或第 123 页中的 “使用 Oracle Solaris Live Upgrade 迁移或升级具有区域的系统（最低 Solaris 10 5/09）”。

ZFS 的 JumpStart 关键字

ZFS 特定的 JumpStart 配置文件中允许使用以下关键字：

`auto` 自动指定池、交换卷或转储卷的分片大小。将会检查磁盘的大小以验证是否可以容纳最小大小。在给定约束条件下（如磁盘大小、保留的分片大小等），如果可以容纳最小大小，则会分配尽可能最大的池大小。

例如，如果指定了 `c0t0d0s0`，则在指定 `all` 或 `auto` 关键字时会创建尽可能大的根池分片。或者，可以指定特定大小的分片、交换卷或转储卷。

用于 ZFS 根池时，`auto` 关键字的工作方式与 `all` 关键字类似，因为池没有未使用的磁盘空间。

`bootenv` 标识引导环境特征。

使用以下 `bootenv` 关键字语法可创建可引导的 ZFS 根环境：

`bootenv installbe bename` *BE-name* [*dataset mount-point*]

`installbe` 创建并安装由 `bename` 选项和 *BE-name* 项标识的新 BE。

`bename` *BE-name* 标识要安装的 *BE-name*。

如果 `bename` 没有与 `pool` 关键字一起使用，则会创建缺省 BE。

`dataset` *mount-point* 使用可选的 `dataset` 关键字可标识与根数据集分离的 `/var` 数据集。*mount-point* 值当前仅限于 `/var`。例如，单独的 `/var` 数据集的 `bootenv` 语法行将类似如下：

```
bootenv installbe bename zfsroot dataset /var
```

`pool` 定义要创建的新根池。必须提供以下关键字语法：

`pool` *poolname poolsize swapsize dumpsize vdevlist*

poolname 标识要创建的池的名称。将使用指定的池大小 (*poolsize*) 和指定的一个或多个物理设备 (*vdevlist*) 创建该池。*poolname* 值不应标识现有池的名称，因为这样将会覆盖现有池。

poolsize 指定要创建的池的大小。该值可以为 `auto` 或 `existing`。在给定的约束条件（如磁盘大小等）下，`auto` 值分配尽可能最大的池大小。除非通过 `g`（表示 GB）指定，否则大小采用 MB 为单位。

swapsize 指定要创建的交换卷的大小。`auto`值表示使用缺省的交换大小。您可以使用 *size* 值指定大小。除非通过 `g`（表示 GB）指定，否则大小以 MB 为单位。

dumpsize 指定要创建的转储卷的大小。`auto` 值表示将使用缺省的转储卷大小。您可以使用 *size* 值指定大小。除非通过 `g`（表示 GB）指定，否则大小采用 MB 为单位。

vdevlist 指定要用于创建池的一个或多个设备。*vdevlist* 的格式与 `zpool create` 命令的格式相同。此时，如果指定了多个设备，则仅

支持镜像配置。对于根池，*vdevlist* 中的设备必须是分片。any 值表示安装软件会选择一个合适的设备。

您可以镜像任意多个磁盘，但创建的池大小由指定磁盘中最小的一个确定。有关创建镜像存储池的更多信息，请参见第 40 页中的“镜像存储池配置”。

ZFS 的 JumpStart 配置文件示例

本节提供了 ZFS 特定的 JumpStart 配置文件的示例。

以下配置文件在由 pool newpool 标识的新池（其大小由 auto 关键字自动设置为指定磁盘的大小）中执行初始安装（由 install_type initial_install 指定）。交换区域和转储设备的大小自动确定（由 auto 关键字标识），采用磁盘镜像配置（由 mirror 关键字标识，磁盘指定为 c0t0d0s0 和 c0t1d0s0）。使用 bootenv 关键字设置引导环境特征，以关键字 installbe 表示安装新 BE，创建名为 s10-xx 的 BE。

```
install_type initial_install
pool newpool auto auto auto mirror c0t0d0s0 c0t1d0s0
bootenv installbe bename s10-xx
```

以下配置文件在名为 newpool 的新池（大小为 80 GB）中执行初始安装（由关键字 install_type initial_install 指定，使用 SUNWCall 元簇）。创建该池时使用 2 GB 的交换卷和 2 GB 的转储卷，采用由任意两个足以创建 80 GB 池的可用设备构成的镜像配置。如果没有这样的两个设备，则安装失败。使用 bootenv 关键字将引导环境特征设置为通过关键字 installbe 安装新 BE，并创建名为 s10–xx 的 *bename*。

```
install_type initial_install
cluster SUNWCall
pool newpool 80g 2g 2g mirror any any
bootenv installbe bename s10-xx
```

JumpStart 安装语法允许在同时包括 ZFS 根池的磁盘上保留或创建 UFS 文件系统。生产系统不建议使用此配置。但是，此配置可以用于满足小型系统（如手提电脑）上的转换或迁移需求。

ZFS 的 JumpStart 问题

开始可引导的 ZFS 根文件系统的 JumpStart 安装之前，请考虑以下问题：

- 您不能将现有 ZFS 存储池用于 JumpStart 安装以创建可引导的 ZFS 根文件系统。必须使用类似如下的语法创建新 ZFS 存储池：

  ```
  pool rpool 20G 4G 4G c0t0d0s0
  ```

- 必须使用磁盘分片而不是整个磁盘创建池，如第 94 页中的 “支持 ZFS 所要满足的 Oracle Solaris 安装要求和 Oracle Solaris Live Upgrade 要求”中所述。例如，以下示例中的粗体部分语法是不可接受的：

```
install_type initial_install
cluster SUNWCall
pool rpool all auto auto mirror c0t0d0 c0t1d0
bootenv installbe bename newBE
```

以下示例中的粗体部分语法是可以接受的：

```
install_type initial_install
cluster SUNWCall
pool rpool all auto auto mirror c0t0d0s0 c0t1d0s0
bootenv installbe bename newBE
```

迁移到 ZFS 根文件系统或更新 ZFS 根文件系统 (Live Upgrade)

与 UFS 组件相关的 Live Upgrade 功能仍然可用，并且其工作方式与先前发行版中的一样。

提供了下列功能：

- **UFS BE 到 ZFS BE 的迁移**
  - 将 UFS 根文件系统迁移到 ZFS 根文件系统时，必须使用 `-p` 选项指定现有 ZFS 存储池。
  - 如果 UFS 根文件系统在不同的分片上具有组件，则会将这些组件迁移到 ZFS 根池。
  - 在 Oracle Solaris 10 8/11 发行版中，将 UFS 根文件系统迁移到 ZFS 根文件系统时可以指定单独的 `/var` 文件系统
  - 将 UFS 根文件系统迁移到 ZFS 根文件系统的基本过程如下：
    1. 安装必需的 Live Upgrade 修补程序（如果需要）。
    2. 在任何基于 SPARC 或基于 x86 的受支持系统上，安装当前的 Oracle Solaris 10 发行版（Solaris 10 10/08 到 Oracle Solaris 10 8/11），或使用标准升级程序从先前的 Oracle Solaris 10 发行版进行升级。
    3. 运行 Solaris 10 10/08 或以上的发行版时，为 ZFS 根文件系统创建 ZFS 存储池。
    4. 使用 Live Upgrade 将 UFS 根文件系统迁移到 ZFS 根文件系统。
    5. 使用 `luactivate` 命令激活 ZFS BE。
- **修补或升级 ZFS BE**
  - 可以使用 `luupgrade` 命令修补或升级现有的 ZFS BE，还可以使用 `luupgrade` 通过 ZFS Flash 归档文件升级备用 ZFS BE。有关信息，请参见示例 4–8。

  - 在同一池中创建新的 ZFS BE 时，Live Upgrade 可以使用 ZFS 快照和克隆功能。因此，创建 BE 比在以前的发行版中快得多。
- **区域迁移支持** – 可以迁移具有区域的系统，但在 Solaris 10 10/08 发行版中支持的配置有限。从 Solaris 10 5/09 发行版开始，支持更多的区域配置。有关更多信息，请参见以下各节：
  - 第 119 页中的 “使用 Live Upgrade 迁移或升级具有区域的系统 (Solaris 10 10/08)”
  - 第 123 页中的 “使用 Oracle Solaris Live Upgrade 迁移或升级具有区域的系统（最低 Solaris 10 5/09）”

  如果要迁移没有区域的系统，请参见第 112 页中的 “使用 Live Upgrade 迁移或更新 ZFS 根文件系统（不具有区域）”。

有关 Oracle Solaris 安装和 Oracle Solaris Live Upgrade 功能的详细信息，请参见《Oracle Solaris 10 1/13 安装指南：Live Upgrade 和升级规划》。

有关 ZFS 和 Live Upgrade 要求的信息，请参见第 94 页中的 “支持 ZFS 所要满足的 Oracle Solaris 安装要求和 Oracle Solaris Live Upgrade 要求”。

Live Upgrade 的 ZFS 迁移问题

使用 Live Upgrade 将 UFS 根文件系统迁移到 ZFS 根文件系统之前，请查看以下问题：

- Oracle Solaris 安装程序 GUI 的标准升级选项不可用于从 UFS 根文件系统到 ZFS 根文件系统的迁移。要从 UFS 文件系统进行迁移，必须使用 Live Upgrade。
- 在进行 Live Upgrade 操作之前，必须创建将用于引导的 ZFS 存储池。此外，由于当前的引导限制，必须使用分片而不是整个磁盘创建 ZFS 根池。例如：

  ```
  # zpool create rpool mirror c1t0d0s0 c1t1d0s0
  ```

  创建新池之前，请确保要用于池中的磁盘具有 SMI (VTOC) 标签而不是 EFI 标签。如果使用 SMI 标签重新对磁盘设置标签，请确保标签设置过程中未更改分区方案。在大多数情况下，所有磁盘容量应位于将用于根池的分片中。
- 您不能使用 Oracle Solaris Live Upgrade 从 ZFS BE 创建 UFS BE。如果将 UFS BE 迁移到 ZFS BE，并且保留 UFS BE，则可以从 UFS BE 或 ZFS BE 引导。
- 请勿使用 `zfs rename` 命令重命名 ZFS BE，因为 Live Upgrade 无法检测到名称更改。后续命令（如 `ludelete`）将会失败。实际上，如果有要继续使用的现有 BE，则请勿重命名 ZFS 池或文件系统。
- 创建备用 BE（主 BE 的克隆）时，不能使用 `-f`、`-x`、`-y`、`-Y` 和 `-z` 选项在主 BE 中包括或从中排除文件。在以下情况下，您仍可以使用包括和排除选项设置：

  ```
  UFS -> UFS
  UFS -> ZFS
  ZFS -> ZFS (different pool)
  ```

- 虽然可以使用 Live Upgrade 将 UFS 根文件系统升级到 ZFS 根文件系统，但不能使用 Live Upgrade 升级非根文件系统或共享文件系统。
- 不能使用 `lu` 命令创建或迁移 ZFS 根文件系统。
- 如果要将系统的交换和转储设备置于非根池，请查看第 135 页中的 “定制 ZFS 交换卷和转储卷”。

使用 Live Upgrade 迁移或更新 ZFS 根文件系统（不具有区域）

以下示例说明如何将 UFS 根文件系统迁移到 ZFS 根文件系统，以及如何更新 ZFS 根文件系统。

如果要迁移或更新具有区域的系统，请参见以下各节：

- 第 119 页中的 “使用 Live Upgrade 迁移或升级具有区域的系统 (Solaris 10 10/08)”
- 第 123 页中的 “使用 Oracle Solaris Live Upgrade 迁移或升级具有区域的系统（最低 Solaris 10 5/09）”

示例 4–4 使用 Live Upgrade 将 UFS 根文件系统迁移到 ZFS 根文件系统

以下示例说明如何从 UFS 根文件系统迁移到 ZFS 根文件系统。包含 UFS 根文件系统的当前 BE `ufsBE` 由 `-c` 选项标识。如果不包括可选的 `-c` 选项，则当前 BE 名称将缺省为设备名称。新 BE `zfsBE` 由 `-n` 选项标识。在执行 `lucreate` 操作之前，ZFS 存储池必须存在。

必须使用分片而不是整个磁盘创建 ZFS 存储池，才能使 ZFS 存储池可升级和可引导。创建新池之前，请确保要用于池中的磁盘具有 SMI (VTOC) 标签而不是 EFI 标签。如果使用 SMI 标签重新对磁盘设置标签，请确保标签设置过程中未更改分区方案。在大多数情况下，所有磁盘容量应位于打算用于根池的分片中。

```
# zpool create rpool mirror c1t2d0s0 c2t1d0s0
# lucreate -c ufsBE -n zfsBE -p rpool
Analyzing system configuration.
No name for current boot environment.
Current boot environment is named <ufsBE>.
Creating initial configuration for primary boot environment <ufsBE>.
The device </dev/dsk/c1t0d0s0> is not a root device for any boot environment; cannot get BE ID.
PBE configuration successful: PBE name <ufsBE> PBE Boot Device </dev/dsk/c1t0d0s0>.
Comparing source boot environment <ufsBE> file systems with the file
system(s) you specified for the new boot environment. Determining which
file systems should be in the new boot environment.
Updating boot environment description database on all BEs.
Updating system configuration files.
The device </dev/dsk/c1t2d0s0> is not a root device for any boot environment; cannot get BE ID.
Creating configuration for boot environment <zfsBE>.
Source boot environment is <ufsBE>.
Creating boot environment <zfsBE>.
```

示例 4–4 使用 Live Upgrade 将 UFS 根文件系统迁移到 ZFS 根文件系统 （续）

```
Creating file systems on boot environment <zfsBE>.
Creating <zfs> file system for </> in zone <global> on <rpool/ROOT/zfsBE>.
Populating file systems on boot environment <zfsBE>.
Checking selection integrity.
Integrity check OK.
Populating contents of mount point </>.
Copying.
Creating shared file system mount points.
Creating compare databases for boot environment <zfsBE>.
Creating compare database for file system </rpool/ROOT>.
Creating compare database for file system </>.
Updating compare databases on boot environment <zfsBE>.
Making boot environment <zfsBE> bootable.
Creating boot_archive for /.alt.tmp.b-qD.mnt
updating /.alt.tmp.b-qD.mnt/platform/sun4u/boot_archive
Population of boot environment <zfsBE> successful.
Creation of boot environment <zfsBE> successful.
```

`lucreate` 操作完成后，使用 `lustatus` 命令查看 BE 状态。例如：

```
# lustatus
Boot Environment           Is       Active Active    Can    Copy
Name                       Complete Now    On Reboot Delete Status
-------------------------- -------- ------ --------- ------ ----------
ufsBE                      yes      yes    yes       no     -
zfsBE                      yes      no     no        yes    -
```

然后，查看 ZFS 组件列表。例如：

```
# zfs list
NAME                   USED  AVAIL  REFER  MOUNTPOINT
rpool                 7.17G  59.8G  95.5K  /rpool
rpool/ROOT            4.66G  59.8G    21K  /rpool/ROOT
rpool/ROOT/zfsBE      4.66G  59.8G  4.66G  /
rpool/dump               2G  61.8G    16K  -
rpool/swap             517M  60.3G    16K  -
```

接下来，使用 `luactivate` 命令激活新 ZFS BE。例如：

```
# luactivate zfsBE
A Live Upgrade Sync operation will be performed on startup of boot environment <zfsBE>.

**********************************************************************

The target boot environment has been activated. It will be used when you
reboot. NOTE: You MUST NOT USE the reboot, halt, or uadmin commands. You
MUST USE either the init or the shutdown command when you reboot. If you
do not use either init or shutdown, the system will not boot using the
target BE.

**********************************************************************
.
.
```

示例 4–4 使用 Live Upgrade 将 UFS 根文件系统迁移到 ZFS 根文件系统 （续）

```
.
Modifying boot archive service
Activation of boot environment <zfsBE> successful.
```

接下来，将系统重新引导至 ZFS BE。

```
# init 6
```

确认 ZFS BE 是否处于活动状态。

```
# lustatus
Boot Environment           Is       Active Active    Can    Copy
Name                       Complete Now    On Reboot Delete Status
-------------------------- -------- ------ --------- ------ ----------
ufsBE                      yes      no     no        yes    -
zfsBE                      yes      yes    yes       no     -
```

如果切换回 UFS BE，必须重新导入在引导 ZFS BE 时创建的所有 ZFS 存储池，因为它们在 UFS BE 中不会自动可用。

如果不再需要 UFS BE，可以使用 `ludelete` 命令将其删除。

示例 4–5 使用 Live Upgrade 从 UFS BE 创建 ZFS BE（具有单独的 `/var`）

在 Oracle Solaris 10 8/11 发行版中，可以使用 `lucreate - D` 选项指定在将 UFS 根文件系统迁移至 ZFS 根文件系统时，要创建单独的 `/var` 文件系统。在以下示例中，现有的 UFS BE 迁移到一个具有单独 `/var` 文件系统的 ZFS BE。

```
# lucreate -n zfsBE -p rpool -D /var
Determining types of file systems supported
Validating file system requests
Preparing logical storage devices
Preparing physical storage devices
Configuring physical storage devices
Configuring logical storage devices
Analyzing system configuration.
No name for current boot environment.
INFORMATION: The current boot environment is not named - assigning name <c0t0d0s0>.
Current boot environment is named <c0t0d0s0>.
Creating initial configuration for primary boot environment <c0t0d0s0>.
INFORMATION: No BEs are configured on this system.
The device </dev/dsk/c0t0d0s0> is not a root device for any boot environment; cannot get BE ID.
PBE configuration successful: PBE name <c0t0d0s0> PBE Boot Device </dev/dsk/c0t0d0s0>.
Updating boot environment description database on all BEs.
Updating system configuration files.
The device </dev/dsk/c0t1d0s0> is not a root device for any boot environment; cannot get BE ID.
Creating configuration for boot environment <zfsBE>.
Source boot environment is <c0t0d0s0>.
Creating file systems on boot environment <zfsBE>.
Creating <zfs> file system for </> in zone <global> on <rpool/ROOT/zfsBE>.
Creating <zfs> file system for </var> in zone <global> on <rpool/ROOT/zfsBE/var>.
```

示例 4–5 使用 Live Upgrade 从 UFS BE 创建 ZFS BE（具有单独的 /var） （续）

```
Populating file systems on boot environment <zfsBE>.
Analyzing zones.
Mounting ABE <zfsBE>.
Generating file list.
Copying data from PBE <c0t0d0s0> to ABE <zfsBE>
100% of filenames transferred
Finalizing ABE.
Fixing zonepaths in ABE.
Unmounting ABE <zfsBE>.
Fixing properties on ZFS datasets in ABE.
Reverting state of zones in PBE <c0t0d0s0>.
Making boot environment <zfsBE> bootable.
Creating boot_archive for /.alt.tmp.b-iaf.mnt
updating /.alt.tmp.b-iaf.mnt/platform/sun4u/boot_archive
Population of boot environment <zfsBE> successful.
Creation of boot environment <zfsBE> successful.
# luactivate zfsBE
A Live Upgrade Sync operation will be performed on startup of boot environment <zfsBE>.
.
.
.
Modifying boot archive service
Activation of boot environment <zfsBE> successful.
# init 6
```

查看新创建的 ZFS 文件系统。例如：

```
# zfs list
NAME                   USED  AVAIL  REFER  MOUNTPOINT
rpool                 6.29G  26.9G  32.5K  /rpool
rpool/ROOT            4.76G  26.9G    31K  legacy
rpool/ROOT/zfsBE      4.76G  26.9G  4.67G  /
rpool/ROOT/zfsBE/var  89.5M  26.9G  89.5M  /var
rpool/dump             512M  26.9G   512M  -
rpool/swap            1.03G  28.0G    16K  -
```

示例 4–6 使用 Live Upgrade 从 ZFS BE 创建 ZFS BE

在同一池中从 ZFS BE 创建 ZFS BE 非常快捷，因为该操作使用 ZFS 快照和克隆功能。如果当前的 BE 驻留在同一 ZFS 池中，则可以省略 -p 选项。

如果有多个 ZFS BE，请执行以下操作来选择从哪个 BE 引导：

- SPARC：可以使用 boot -L 命令标识可用的 BE。然后，使用 boot -Z 命令选择一个要从中引导的 BE。
- x86：您可以从 GRUB 菜单中选择一个 BE。

有关更多信息，请参见示例 4–12。

```
# lucreate -n zfs2BE
Analyzing system configuration.
```

示例 4–6 使用 Live Upgrade 从 ZFS BE 创建 ZFS BE （续）

```
No name for current boot environment.
INFORMATION: The current boot environment is not named - assigning name <zfsBE>.
Current boot environment is named <zfsBE>.
Creating initial configuration for primary boot environment <zfsBE>.
The device </dev/dsk/c1t0d0s0> is not a root device for any boot environment; cannot get BE ID.
PBE configuration successful: PBE name <zfsBE> PBE Boot Device </dev/dsk/c1t0d0s0>.
Comparing source boot environment <zfsBE> file systems with the file
system(s) you specified for the new boot environment. Determining which
file systems should be in the new boot environment.
Updating boot environment description database on all BEs.
Updating system configuration files.
Creating configuration for boot environment <zfs2BE>.
Source boot environment is <zfsBE>.
Creating boot environment <zfs2BE>.
Cloning file systems from boot environment <zfsBE> to create boot environment <zfs2BE>.
Creating snapshot for <rpool/ROOT/zfsBE> on <rpool/ROOT/zfsBE@zfs2BE>.
Creating clone for <rpool/ROOT/zfsBE@zfs2BE> on <rpool/ROOT/zfs2BE>.
Setting canmount=noauto for </> in zone <global> on <rpool/ROOT/zfs2BE>.
Population of boot environment <zfs2BE> successful.
Creation of boot environment <zfs2BE> successful.
```

示例 4–7 更新 ZFS BE (`luupgrade`)

您可以使用其他软件包或修补程序更新 ZFS BE。

基本过程如下：

- 使用 `lucreate` 命令创建备用 BE。
- 激活该备用 BE 并从中引导。
- 使用 `luupgrade` 命令添加软件包或修补程序以更新主 ZFS BE。

```
# lustatus
Boot Environment           Is       Active Active    Can    Copy
Name                       Complete Now    On Reboot Delete Status
-------------------------- -------- ------ --------- ------ ----------
zfsBE                      yes      no     no        yes    -
zfs2BE                     yes      yes    yes       no     -
# luupgrade -p -n zfsBE -s /net/system/export/s10up/Solaris_10/Product SUNWchxge
Validating the contents of the media </net/install/export/s10up/Solaris_10/Product>.
Mounting the BE <zfsBE>.
Adding packages to the BE <zfsBE>.

Processing package instance <SUNWchxge> from </net/install/export/s10up/Solaris_10/Product>

Chelsio N110 10GE NIC Driver(sparc) 11.10.0,REV=2006.02.15.20.41
Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.

This appears to be an attempt to install the same architecture and
version of a package which is already installed.  This installation
will attempt to overwrite this package.

Using </a> as the package base directory.
## Processing package information.
```

示例 4–7 更新 ZFS BE (luupgrade) （续）

```
## Processing system information.
   4 package pathnames are already properly installed.
## Verifying package dependencies.
## Verifying disk space requirements.
## Checking for conflicts with packages already installed.
## Checking for setuid/setgid programs.

This package contains scripts which will be executed with super-user
permission during the process of installing this package.

Do you want to continue with the installation of <SUNWchxge> [y,n,?] y
Installing Chelsio N110 10GE NIC Driver as <SUNWchxge>

## Installing part 1 of 1.
## Executing postinstall script.

Installation of <SUNWchxge> was successful.
Unmounting the BE <zfsBE>.
The package add to the BE <zfsBE> completed.
```

或者，可以创建新的 BE 来更新到较新的 Oracle Solaris 发行版。例如：

```
# luupgrade -u -n newBE -s /net/install/export/s10up/latest
```

其中 -s 选项指定 Solaris 安装介质的位置。

示例 4–8 使用 ZFS Flash 归档文件创建 ZFS BE (luupgrade)

在 Oracle Solaris 10 8/11 发行版中，可以使用 luupgrade 命令从现有的 ZFS Flash 归档文件创建 ZFS BE。基本过程如下所示：

1. 为具有 ZFS BE 的主系统创建 Flash 归档文件。

   例如：

   ```
   master-system# flarcreate -n s10zfsBE /tank/data/s10zfsflar
   Full Flash
   Checking integrity...
   Integrity OK.
   Running precreation scripts...
   Precreation scripts done.
   Determining the size of the archive...
   The archive will be approximately 4.67GB.
   Creating the archive...
   Archive creation complete.
   Running postcreation scripts...
   Postcreation scripts done.

   Running pre-exit scripts...
   Pre-exit scripts done.
   ```

2. 将基于主系统创建的 ZFS Flash 归档文件设为可供克隆系统使用。

   可能的 Flash 归档文件位置包括本地文件系统、HTTP、FTP、NFS 等等。

示例 4–8 使用 ZFS Flash 归档文件创建 ZFS BE (luupgrade) （续）

3. 在克隆系统上创建一个空白的备用 ZFS BE。

   使用 -s - 选项指定这是要使用 ZFS Flash 归档文件内容填充的空白 BE。

   例如：

```
clone-system# lucreate -n zfsflashBE -s - -p rpool
Determining types of file systems supported
Validating file system requests
Preparing logical storage devices
Preparing physical storage devices
Configuring physical storage devices
Configuring logical storage devices
Analyzing system configuration.
No name for current boot environment.
INFORMATION: The current boot environment is not named - assigning name <s10zfsBE>.
Current boot environment is named <s10zfsBE>.
Creating initial configuration for primary boot environment <s10zfsBE>.
INFORMATION: No BEs are configured on this system.
The device </dev/dsk/c0t0d0s0> is not a root device for any boot environment; cannot get BE ID.
PBE configuration successful: PBE name <s10zfsBE> PBE Boot Device </dev/dsk/c0t0d0s0>.
Updating boot environment description database on all BEs.
Updating system configuration files.
The device </dev/dsk/c0t1d0s0> is not a root device for any boot environment; cannot get BE ID.
Creating <zfs> file system for </> in zone <global> on <rpool/ROOT/zfsflashBE>.
Creation of boot environment <zfsflashBE> successful.
```

4. 将 ZFS Flash 归档文件安装到备用 BE 中。

   例如：

```
clone-system# luupgrade -f -s /net/server/export/s10/latest -n zfsflashBE -a /tank/data/zfs10up2flar
miniroot filesystem is <lofs>
Mounting miniroot at </net/server/s10up/latest/Solaris_10/Tools/Boot>
Validating the contents of the media </net/server/export/s10up/latest>.
The media is a standard Solaris media.
Validating the contents of the miniroot </net/server/export/s10up/latest/Solaris_10/Tools/Boot>.
Locating the flash install program.
Checking for existence of previously scheduled Live Upgrade requests.
Constructing flash profile to use.
Creating flash profile for BE <zfsflashBE>.
Performing the operating system flash install of the BE <zfsflashBE>.
CAUTION: Interrupting this process may leave the boot environment unstable or unbootable.
Extracting Flash Archive: 100% completed (of 5020.86 megabytes)
The operating system flash install completed.
updating /.alt.tmp.b-rgb.mnt/platform/sun4u/boot_archive

The Live Flash Install of the boot environment <zfsflashBE> is complete.
```

5. 激活备用 BE。

```
clone-system# luactivate zfsflashBE
A Live Upgrade Sync operation will be performed on startup of boot environment <zfsflashBE>.
.
.
.
Modifying boot archive service
```

示例 4–8 使用 ZFS Flash 归档文件创建 ZFS BE (luupgrade) （续）

```
Activation of boot environment <zfsflashBE> successful.
```

6. 重新引导系统。

```
clone-system# init 6
```

使用 Live Upgrade 迁移或升级具有区域的系统 (Solaris 10 10/08)

可以使用 Live Upgrade 迁移具有区域的系统，但在 Solaris 10 10/08 发行版中支持的配置有限。如果安装或升级到 Solaris 10 5/09 或以上的发行版，可支持更多区域配置。有关更多信息，请参见第 123 页中的 “使用 Oracle Solaris Live Upgrade 迁移或升级具有区域的系统（最低 Solaris 10 5/09）”。

本节介绍如何安装和配置具有区域的系统，以便使用 Live Upgrade 升级和修补该系统。如果要迁移到没有区域的 ZFS 根文件系统，请参见第 112 页中的 “使用 Live Upgrade 迁移或更新 ZFS 根文件系统（不具有区域）”。

如果要在 Solaris 10 10/08 发行版中迁移具有区域的系统或配置具有区域的系统，请查看以下过程：

- 第 119 页中的 “如何将 UFS 上具有区域根的 UFS 根文件系统迁移到 ZFS 根文件系统 (Solaris 10 10/08)”
- 第 121 页中的 “如何配置 ZFS 上具有区域根的 ZFS 根文件系统 (Solaris 10 10/08)”
- 第 122 页中的 “如何升级或修补 ZFS 上具有区域根的 ZFS 根文件系统 (Solaris 10 10/08)”
- 第 140 页中的 “解决妨碍成功引导的 ZFS 挂载点问题 (Solaris 10 10/08)”

按照建议的过程在具有 ZFS 根文件系统的系统上设置区域，以确保可以在该系统上使用 Live Upgrade。

▼ 如何将 UFS 上具有区域根的 UFS 根文件系统迁移到 ZFS 根文件系统 (Solaris 10 10/08)

此过程解释如何将安装了区域的 UFS 根文件系统迁移到 ZFS 根文件系统，以及如何升级或修补的 ZFS 区域根配置。

在后面的步骤中，示例池名称为 `rpool`，活动引导环境 (Boot Environment, BE) 的示例名称以 `s10BE*` 开头。

1. **如果系统当前运行的是先前的 Solaris 10 发行版，请将其升级到 Solaris 10 10/08 发行版。**

   有关对运行 Solaris 10 发行版的系统进行升级的更多信息，请参见《Oracle Solaris 10 1/13 安装指南：Live Upgrade 和升级规划》。

2. **创建根池。**

   ```
   # zpool create rpool mirror c0t1d0 c1t1d0
   ```

   有关根池要求的信息，请参见第 94 页中的 “支持 ZFS 所要满足的 Oracle Solaris 安装要求和 Oracle Solaris Live Upgrade 要求”。

3. **确认已引导 UFS 环境中的区域。**

4. **创建新 ZFS 引导环境。**

   ```
   # lucreate -n s10BE2 -p rpool
   ```

   此命令将在根池中为新 BE 建立数据集，并将当前的 BE（包括区域）复制到这些数据集。

5. **激活新 ZFS 引导环境。**

   ```
   # luactivate s10BE2
   ```

   现在，系统正在运行 ZFS 根文件系统，但 UFS 上的区域根仍在 UFS 根文件系统中。需要执行后续步骤将 UFS 区域完全迁移到支持的 ZFS 配置。

6. **重新引导系统。**

   ```
   # init 6
   ```

7. **将区域迁移到 ZFS BE。**

   a. **引导区域。**

   b. **在池中创建另一个 ZFS BE。**

      ```
      # lucreate s10BE3
      ```

   c. **激活新引导环境。**

      ```
      # luactivate s10BE3
      ```

   d. **重新引导系统。**

      ```
      # init 6
      ```

      此步骤用于验证是否已引导 ZFS BE 和区域。

8 解决任何潜在的挂载点问题。

由于 Live Upgrade 中的错误，非活动 BE 可能无法引导，因为该 BE 中的 ZFS 数据集或区域的 ZFS 数据集有无效的挂载点。

a. 查看 `zfs list` 输出。

查找不正确的临时挂载点。例如：

```
# zfs list -r -o name,mountpoint rpool/ROOT/s10up

NAME                                MOUNTPOINT
rpool/ROOT/s10up                    /.alt.tmp.b-VP.mnt/
rpool/ROOT/s10up/zones              /.alt.tmp.b-VP.mnt//zones
rpool/ROOT/s10up/zones/zonerootA    /.alt.tmp.b-VP.mnt/zones/zonerootA
```

根 ZFS BE (`rpool/ROOT/s10up`) 的挂载点应为 `/`。

b. 重置 ZFS BE 及其数据集的挂载点。

例如：

```
# zfs inherit -r mountpoint rpool/ROOT/s10up
# zfs set mountpoint=/ rpool/ROOT/s10up
```

c. 重新引导系统。

出现引导特定 BE 的选项（在 OpenBoot PROM 提示符下或 GRUB 菜单中）时，选择刚刚更正了其挂载点的 BE。

▼ 如何配置 ZFS 上具有区域根的 ZFS 根文件系统 (Solaris 10 10/08)

此过程解释如何设置 ZFS 根文件系统和可以进行升级或修补的 ZFS 区域根配置。在此配置中，ZFS 区域根创建为 ZFS 数据集。

在后面的步骤中，示例池名称为 `rpool`，活动引导环境的示例名称为 `s10BE`。区域数据集的名称可以为任何有效的数据集名称。在以下示例中，区域数据集名称为 `zones`。

1 使用交互式文本安装程序或 JumpStart 安装方法安装具有 ZFS 根的系统。

根据您选择的安装方法，请参见第 97 页中的 “安装 ZFS 根文件系统（Oracle Solaris 初始安装）”或第 107 页中的 “安装 ZFS 根文件系统（JumpStart 安装）”。

2 从新创建的根池引导系统。

3 创建数据集以用于对区域根进行分组。

例如：

```
# zfs create -o canmount=noauto rpool/ROOT/s10BE/zones
```

将 `canmount` 属性的值设置为 `noauto` 可防止以 Live Upgrade 显式操作和系统启动代码以外的其他方式挂载数据集。

4 **挂载新创建的区域数据集。**

```
# zfs mount rpool/ROOT/s10BE/zones
```

数据集挂载在 /zones。

5 **为每个区域根创建并挂载数据集。**

```
# zfs create -o canmount=noauto rpool/ROOT/s10BE/zones/zonerootA
# zfs mount rpool/ROOT/s10BE/zones/zonerootA
```

6 **在区域根目录上设置适当的权限。**

```
# chmod 700 /zones/zonerootA
```

7 **配置区域，如下所示设置区域路径：**

```
# zonecfg -z zoneA
    zoneA: No such zone configured
    Use 'create' to begin configuring a new zone.
    zonecfg:zoneA> create
    zonecfg:zoneA> set zonepath=/zones/zonerootA
```

可通过使用以下语法在引导系统时自动引导区域：

```
zonecfg:zoneA> set autoboot=true
```

8 **安装区域。**

```
# zoneadm -z zoneA install
```

9 **引导区域。**

```
# zoneadm -z zoneA boot
```

▼ 如何升级或修补 ZFS 上具有区域根的 ZFS 根文件系统 (Solaris 10 10/08)

当您需要升级或修补 ZFS 上具有区域根的 ZFS 根文件系统时，请执行此过程。这些更新可以是升级系统，也可以是应用修补程序。

在后面的步骤中，newBE 是所升级或修补的 BE 的示例名称。

1 **创建 BE 以进行升级或修补。**

```
# lucreate -n newBE
```

将克隆现有的 BE，包括所有区域。将为原始 BE 中的每个数据集创建一个数据集。将在与当前根池相同的池中创建新数据集。

2 **选择以下操作之一来升级系统或将修补程序应用到新 BE：**

- 升级系统。

  ```
  # luupgrade -u -n newBE -s /net/install/export/s10up/latest
  ```

其中 -s 选项指定 Oracle Solaris 安装介质的位置。

- 将修补程序应用到新 BE。

```
# luupgrade -t -n newBE -t -s /patchdir 139147-02 157347-14
```

3 **激活新 BE。**

```
# luactivate newBE
```

4 **从新激活的 BE 引导。**

```
# init 6
```

5 **解决任何潜在的挂载点问题。**

由于 Live Upgrade 中的错误，非活动 BE 可能无法引导，因为该 BE 中的 ZFS 数据集或区域的 ZFS 数据集有无效的挂载点。

a. **查看 `zfs list` 输出。**

查找不正确的临时挂载点。例如：

```
# zfs list -r -o name,mountpoint rpool/ROOT/newBE

NAME                                MOUNTPOINT
rpool/ROOT/newBE                    /.alt.tmp.b-VP.mnt/
rpool/ROOT/newBE/zones              /.alt.tmp.b-VP.mnt/zones
rpool/ROOT/newBE/zones/zonerootA    /.alt.tmp.b-VP.mnt/zones/zonerootA
```

根 ZFS BE (`rpool/ROOT/newBE`) 的挂载点应为 `/`。

b. **重置 ZFS BE 及其数据集的挂载点。**

例如：

```
# zfs inherit -r mountpoint rpool/ROOT/newBE
# zfs set mountpoint=/ rpool/ROOT/newBE
```

c. **重新引导系统。**

出现引导特定引导环境的选项（在 OpenBoot PROM 提示符下或 GRUB 菜单中）时，选择刚刚更正了其挂载点的引导环境。

使用 Oracle Solaris Live Upgrade 迁移或升级具有区域的系统（最低 Solaris 10 5/09）

从 Solaris 10 10/08 发行版开始，您可以使用 Oracle Solaris Live Upgrade 功能迁移或升级具有区域的系统。从 Solaris 10 5/09 发行版开始，Live Upgrade 支持更多稀疏（根和完全）区域配置。

本节介绍从 Solaris 10 5/09 发行版开始，如何配置具有区域的系统，以便使用 Live Upgrade 升级和修补该系统。如果要迁移到没有区域的 ZFS 根文件系统，请参见第 112 页中的“使用 Live Upgrade 迁移或更新 ZFS 根文件系统（不具有区域）”。

从 Solaris 10 5/09 发行版开始，对 ZFS 和区域使用 Oracle Solaris Live Upgrade 时，请考虑以下几点：

- 要对从 Solaris 10 5/09 发行版开始支持的区域配置使用 Live Upgrade，首先必须使用标准升级程序将系统升级到 Solaris 10 5/09 或以上的发行版。
- 然后，您可以使用 Live Upgrade 将具有区域根的 UFS 根文件系统迁移到 ZFS 根文件系统，也可以升级或修补 ZFS 根文件系统和区域根。
- 您无法将不受支持的区域配置从先前的 Solaris 10 发行版直接迁移到 Solaris 10 5/09 或以上的发行版。

从 Solaris 10 5/09 发行版开始，如果您要迁移或配置具有区域的系统，请查看以下信息：

- 第 124 页中的“支持的 ZFS 和区域根配置信息（最低 Solaris 10 5/09）”
- 第 125 页中的“如何创建具有 ZFS 根文件系统和区域根的 ZFS BE（最低 Solaris 10 5/09）”
- 第 127 页中的“如何升级或修补具有区域根的 ZFS 根文件系统（最低 Solaris 10 5/09）”
- 第 130 页中的“如何将具有区域根的 UFS 根文件系统迁移到 ZFS 根文件系统（最低 Solaris 10 5/09）”

支持的 ZFS 和区域根配置信息（最低 Solaris 10 5/09）

在使用 Oracle Solaris Live Upgrade 迁移或升级具有区域的系统之前，需要先查看支持的区域配置。

- **将 UFS 根文件系统迁移到 ZFS 根文件系统**－支持以下区域根配置：
  - UFS 根文件系统中的目录
  - UFS 根文件系统中的挂载点的子目录
  - UFS 根文件系统的区域根位于 UFS 根文件系统目录中或 UFS 根文件系统挂载点的子目录中，ZFS 非根池具有区域根

  不支持以区域根作为挂载点的 UFS 根文件系统。
- **迁移或升级 ZFS 根文件系统**－支持以下区域根配置：
  - 位于 ZFS 根池或非根池中的文件系统内。例如，`/zonepool/zones` 是可以接受的。在某些情况下，如果在执行 Live Upgrade 操作前未提供区域根的文件系统，Live Upgrade 将会创建区域根的文件系统 (`zoneds`)。
  - 位于 ZFS 文件系统的后代文件系统或子目录中（只要不同区域路径未嵌套）。例如，`/zonepool/zones/zone1` 和 `/zonepool/zones/zone1_dir` 是可以接受的。

在以下示例中，zonepool/zones 是一个包含区域根的文件系统，rpool 包含 ZFS BE：

```
zonepool
zonepool/zones
zonepool/zones/myzone
rpool
rpool/ROOT
rpool/ROOT/myBE
```

如果使用下列语法，Live Upgrade 会对 zonepool 和 rpool BE 中的区域实施快照并进行克隆：

```
# lucreate -n newBE
```

创建 newBE BE，位于 rpool/ROOT/newBE 中。激活后，newBE 提供访问 zonepool 组件的途径。

在上面的示例中，如果 /zonepool/zones 是一个子目录，而不是单独的文件系统，则 Live Upgrade 会将其作为根池 rpool 的组件进行迁移。

- **不支持以下 ZFS 和区域配置：**
  - 当源 BE 具有非全局区域并且其区域路径设置为顶层池文件系统的挂载点时，无法使用 Live Upgrade 创建备用 BE。例如，如果 zonepool 池有一个挂载为 /zonepool 的文件系统，则无法拥有一个区域路径设置为 /zonepool 的非全局区域。
  - 不要在全局区域的 /etc/vfstab 文件中为非全局区域添加文件系统条目。请改用 zonecfg 的 add fs 功能为非全局区域添加文件系统。
- **UFS 和 ZFS 区域的迁移或升级信息** – 查看下列可能会影响 UFS 或 ZFS 环境的迁移或升级的因素：
  - 如果您在 Solaris 10 10/08 发行版中按照第 119 页中的 “使用 Live Upgrade 迁移或升级具有区域的系统 (Solaris 10 10/08)”所述的方法配置了区域并已升级到 Solaris 10 5/09 或以上的发行版，将可以迁移到 ZFS 根文件系统，或者使用 Live Upgrade 升级到 Solaris 10 5/09 或以上的发行版。
  - 请勿在嵌套目录中创建区域根，例如 zones/zone1 和 zones/zone1/zone2。否则，引导时挂载可能失败。

▼ 如何创建具有 ZFS 根文件系统和区域根的 ZFS BE（最低 Solaris 10 5/09）

执行 Solaris 10 5/09 或以上的发行版的初始安装之后，使用此过程创建 ZFS 根文件系统。使用 luupgrade 命令将 ZFS 根文件系统升级到 Solaris 10 5/09 或以上的发行版之后，也可以使用此过程。然后，可以升级或修补使用此过程创建的 ZFS BE。

在后面的步骤中，Oracle Solaris 10 9/10 示例系统在 /rpool/zones 中具有 ZFS 根文件系统和区域根数据集。将创建名为 zfs2BE 的 ZFS BE，然后可以对其进行升级或修补。

1 查看现有的 ZFS 文件系统。

```
# zfs list
NAME                   USED  AVAIL  REFER  MOUNTPOINT
rpool                 7.26G  59.7G    98K  /rpool
rpool/ROOT            4.64G  59.7G    21K  legacy
rpool/ROOT/zfsBE      4.64G  59.7G  4.64G  /
rpool/dump            1.00G  59.7G  1.00G  -
rpool/export            44K  59.7G    23K  /export
rpool/export/home       21K  59.7G    21K  /export/home
rpool/swap               1G  60.7G    16K  -
rpool/zones            633M  59.7G   633M  /rpool/zones
```

2 确保安装和引导了区域。

```
# zoneadm list -cv
  ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              native   shared
   2 zfszone          running    /rpool/zones                   native   shared
```

3 创建 ZFS BE。

```
# lucreate -n zfs2BE
Analyzing system configuration.
No name for current boot environment.
INFORMATION: The current boot environment is not named - assigning name <zfsBE>.
Current boot environment is named <zfsBE>.
Creating initial configuration for primary boot environment <zfsBE>.
The device </dev/dsk/c1t0d0s0> is not a root device for any boot environment; cannot get BE ID.
PBE configuration successful: PBE name <zfsBE> PBE Boot Device </dev/dsk/c1t0d0s0>.
Comparing source boot environment <zfsBE> file systems with the file
system(s) you specified for the new boot environment. Determining which
file systems should be in the new boot environment.
Updating boot environment description database on all BEs.
Updating system configuration files.
Creating configuration for boot environment <zfs2BE>.
Source boot environment is <zfsBE>.
Creating boot environment <zfs2BE>.
Cloning file systems from boot environment <zfsBE> to create boot environment <zfs2BE>.
Creating snapshot for <rpool/ROOT/zfsBE> on <rpool/ROOT/zfsBE@zfs2BE>.
Creating clone for <rpool/ROOT/zfsBE@zfs2BE> on <rpool/ROOT/zfs2BE>.
Setting canmount=noauto for </> in zone <global> on <rpool/ROOT/zfs2BE>.
Population of boot environment <zfs2BE> successful.
Creation of boot environment <zfs2BE> successful.
```

4 激活 ZFS BE。

```
# lustatus
Boot Environment           Is       Active Active    Can    Copy
Name                       Complete Now    On Reboot Delete Status
-------------------------- -------- ------ --------- ------ ----------
zfsBE                      yes      yes    yes       no     -
zfs2BE                     yes      no     no        yes    -
# luactivate zfs2BE
A Live Upgrade Sync operation will be performed on startup of boot environment <zfs2BE>.
.
.
.
```

5 **引导 ZFS BE。**

```
# init 6
```

6 **确认在新 BE 中创建了 ZFS 文件系统和区域。**

```
# zfs list
NAME                                USED  AVAIL  REFER  MOUNTPOINT
rpool                              7.38G  59.6G    98K  /rpool
rpool/ROOT                         4.72G  59.6G    21K  legacy
rpool/ROOT/zfs2BE                  4.72G  59.6G  4.64G  /
rpool/ROOT/zfs2BE@zfs2BE           74.0M      -  4.64G  -
rpool/ROOT/zfsBE                   5.45M  59.6G  4.64G  /.alt.zfsBE
rpool/dump                         1.00G  59.6G  1.00G  -
rpool/export                         44K  59.6G    23K  /export
rpool/export/home                    21K  59.6G    21K  /export/home
rpool/swap                            1G  60.6G    16K  -
rpool/zones                        17.2M  59.6G   633M  /rpool/zones
rpool/zones-zfsBE                   653M  59.6G   633M  /rpool/zones-zfsBE
rpool/zones-zfsBE@zfs2BE           19.9M      -   633M  -
# zoneadm list -cv
  ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              native   shared
   - zfszone          installed  /rpool/zones                   native   shared
```

▼ 如何升级或修补具有区域根的 ZFS 根文件系统（最低 Solaris 10 5/09）

在 Solaris 10 5/09 或以上的发行版中，当您需要升级或修补具有区域根的 ZFS 根文件系统时，请执行此过程。这些更新可以是升级系统，也可以是应用修补程序。

在后面的步骤中，`zfs2BE` 是所升级或修补的 BE 的示例名称。

1 **查看现有的 ZFS 文件系统。**

```
# zfs list
NAME                                USED  AVAIL  REFER  MOUNTPOINT
rpool                              7.38G  59.6G   100K  /rpool
rpool/ROOT                         4.72G  59.6G    21K  legacy
rpool/ROOT/zfs2BE                  4.72G  59.6G  4.64G  /
rpool/ROOT/zfs2BE@zfs2BE           75.0M      -  4.64G  -
rpool/ROOT/zfsBE                   5.46M  59.6G  4.64G  /
rpool/dump                         1.00G  59.6G  1.00G  -
rpool/export                         44K  59.6G    23K  /export
rpool/export/home                    21K  59.6G    21K  /export/home
rpool/swap                            1G  60.6G    16K  -
rpool/zones                        22.9M  59.6G   637M  /rpool/zones
rpool/zones-zfsBE                   653M  59.6G   633M  /rpool/zones-zfsBE
rpool/zones-zfsBE@zfs2BE           20.0M      -   633M  -
```

2 **确保安装和引导了区域。**

```
# zoneadm list -cv
  ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              native   shared
   5 zfszone          running    /rpool/zones                   native   shared
```

3 创建 ZFS BE 以进行升级或修补。

```
# lucreate -n zfs2BE
Analyzing system configuration.
Comparing source boot environment <zfsBE> file systems with the file
system(s) you specified for the new boot environment. Determining which
file systems should be in the new boot environment.
Updating boot environment description database on all BEs.
Updating system configuration files.
Creating configuration for boot environment <zfs2BE>.
Source boot environment is <zfsBE>.
Creating boot environment <zfs2BE>.
Cloning file systems from boot environment <zfsBE> to create boot environment <zfs2BE>.
Creating snapshot for <rpool/ROOT/zfsBE> on <rpool/ROOT/zfsBE@zfs2BE>.
Creating clone for <rpool/ROOT/zfsBE@zfs2BE> on <rpool/ROOT/zfs2BE>.
Setting canmount=noauto for </> in zone <global> on <rpool/ROOT/zfs2BE>.
Creating snapshot for <rpool/zones> on <rpool/zones@zfs10092BE>.
Creating clone for <rpool/zones@zfs2BE> on <rpool/zones-zfs2BE>.
Population of boot environment <zfs2BE> successful.
Creation of boot environment <zfs2BE> successful.
```

4 选择以下操作之一来升级系统或将修补程序应用到新 BE：

- 升级系统。

  ```
  # luupgrade -u -n zfs2BE -s /net/install/export/s10up/latest
  ```

  其中 -s 选项指定 Oracle Solaris 安装介质的位置。

  此过程要花费很长时间。

  有关 luupgrade 过程的完整示例，请参见示例 4–9。

- 将修补程序应用到新 BE。

  ```
  # luupgrade -t -n zfs2BE -t -s /patchdir patch-id-02 patch-id-04
  ```

5 激活新引导环境。

```
# lustatus
Boot Environment           Is       Active Active    Can    Copy
Name                       Complete Now    On Reboot Delete Status
-------------------------- -------- ------ --------- ------ ----------
zfsBE                      yes      yes    yes       no     -
zfs2BE                     yes      no     no        yes    -
# luactivate zfs2BE
A Live Upgrade Sync operation will be performed on startup of boot environment <zfs2BE>.
.
.
.
```

6 从新激活的引导环境引导。

```
# init 6
```

示例 4–9 将具有区域根的 ZFS 根文件系统升级到 Oracle Solaris 10 9/10 ZFS 根文件系统

在此示例中，将在具有 ZFS 根文件系统和非根池的区域根的 Solaris 10 10/09 系统上创建的 ZFS BE (zfsBE) 升级到 Oracle Solaris 10 9/10 发行版。此过程要花费较长时间。然后，激活升级的 BE (zfs2BE)。请确保在尝试升级之前，安装并引导了区域。

在此示例中，zonepool 池、/zonepool/zones 数据集和 zfszone 区域是按如下方式创建的：

```
# zpool create zonepool mirror c2t1d0 c2t5d0
# zfs create zonepool/zones
# chmod 700 zonepool/zones
# zonecfg -z zfszone
zfszone: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:zfszone> create
zonecfg:zfszone> set zonepath=/zonepool/zones
zonecfg:zfszone> verify
zonecfg:zfszone> exit
# zoneadm -z zfszone install
cannot create ZFS dataset zonepool/zones: dataset already exists
Preparing to install zone <zfszone>.
Creating list of files to copy from the global zone.
Copying <8960> files to the zone.
.
.
.

# zoneadm list -cv
  ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              native   shared
   2 zfszone          running    /zonepool/zones                native   shared

# lucreate -n zfsBE
.
.
.
# luupgrade -u -n zfsBE -s /net/install/export/s10up/latest
40410 blocks
miniroot filesystem is <lofs>
Mounting miniroot at </net/system/export/s10up/latest/Solaris_10/Tools/Boot>
Validating the contents of the media </net/system/export/s10up/latest>.
The media is a standard Solaris media.
The media contains an operating system upgrade image.
The media contains <Solaris> version <10>.
Constructing upgrade profile to use.
Locating the operating system upgrade program.
Checking for existence of previously scheduled Live Upgrade requests.
Creating upgrade profile for BE <zfsBE>.
Determining packages to install or upgrade for BE <zfsBE>.
Performing the operating system upgrade of the BE <zfsBE>.
CAUTION: Interrupting this process may leave the boot environment unstable
or unbootable.
Upgrading Solaris: 100% completed
```

```
Installation of the packages from this media is complete.
Updating package information on boot environment <zfsBE>.
Package information successfully updated on boot environment <zfsBE>.
Adding operating system patches to the BE <zfsBE>.
The operating system patch installation is complete.
INFORMATION: The file </var/sadm/system/logs/upgrade_log> on boot
environment <zfsBE> contains a log of the upgrade operation.
INFORMATION: The file </var/sadm/system/data/upgrade_cleanup> on boot
environment <zfsBE> contains a log of cleanup operations required.
INFORMATION: Review the files listed above. Remember that all of the files
are located on boot environment <zfsBE>. Before you activate boot
environment <zfsBE>, determine if any additional system maintenance is
required or if additional media of the software distribution must be
installed.
The Solaris upgrade of the boot environment <zfsBE> is complete.
Installing failsafe
Failsafe install is complete.
# luactivate zfs2BE
# init 6
# lustatus
Boot Environment           Is       Active Active    Can    Copy
Name                       Complete Now    On Reboot Delete Status
-------------------------- -------- ------ --------- ------ ----------
zfsBE                      yes      no     no        yes    -
zfs2BE                     yes      yes    yes       no     -
# zoneadm list -cv
  ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              native   shared
   - zfszone          installed  /zonepool/zones                native   shared
```

▼ 如何将具有区域根的 UFS 根文件系统迁移到 ZFS 根文件系统（最低 Solaris 10 5/09）

使用此过程将具有 UFS 根文件系统和区域根的系统迁移到 Solaris 10 5/09 或以上的发行版。然后，使用 Live Upgrade 创建 ZFS BE。

在后面的步骤中，示例 UFS BE 名称为 `c1t1d0s0`，UFS 区域根为 `zonepool/zfszone`，ZFS 根 BE 为 `zfsBE`。

1 如果系统当前运行的是先前的 Solaris 10 发行版，请将其升级到 Solaris 10 5/09 或以上的发行版。

有关对运行 Solaris 10 发行版的系统进行升级的信息，请参见《Oracle Solaris 10 1/13 安装指南：Live Upgrade 和升级规划》。

2 创建根池。

有关根池要求的信息，请参见第 94 页中的 “支持 ZFS 所要满足的 Oracle Solaris 安装要求和 Oracle Solaris Live Upgrade 要求”。

3 确认已引导 UFS 环境中的区域。

```
# zoneadm list -cv
  ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              native   shared
   2 zfszone          running    /zonepool/zones                native   shared
```

4 创建新的 ZFS BE。

```
# lucreate -c c1t1d0s0 -n zfsBE -p rpool
```

此命令将在根池中为新 BE 建立数据集，并将当前的 BE（包括区域）复制到这些数据集。

5 激活新的 ZFS BE。

```
# lustatus
Boot Environment           Is       Active Active    Can    Copy
Name                       Complete Now    On Reboot Delete Status
-------------------------- -------- ------ --------- ------ ----------
c1t1d0s0                   yes      no     no        yes    -
zfsBE                      yes      yes    yes       no     -         #
luactivate zfsBE
A Live Upgrade Sync operation will be performed on startup of boot environment <zfsBE>.
.
.
.
```

6 重新引导系统。

```
# init 6
```

7 确认在新 BE 中创建了 ZFS 文件系统和区域。

```
# zfs list
NAME                              USED  AVAIL  REFER  MOUNTPOINT
rpool                            6.17G  60.8G    98K  /rpool
rpool/ROOT                       4.67G  60.8G    21K  /rpool/ROOT
rpool/ROOT/zfsBE                 4.67G  60.8G  4.67G  /
rpool/dump                       1.00G  60.8G  1.00G  -
rpool/swap                        517M  61.3G    16K  -
zonepool                          634M  7.62G    24K  /zonepool
zonepool/zones                    270K  7.62G   633M  /zonepool/zones
zonepool/zones-c1t1d0s0           634M  7.62G   633M  /zonepool/zones-c1t1d0s0
zonepool/zones-c1t1d0s0@zfsBE     262K      -   633M  -
# zoneadm list -cv
  ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              native   shared
   - zfszone          installed  /zonepool/zones                native   shared
```

示例 4–10 将具有区域根的 UFS 根文件系统迁移到 ZFS 根文件系统

在此示例中，将具有 UFS 根文件系统和区域根 (/uzone/ufszone)、ZFS 非根池 (pool) 以及区域根 (/pool/zfszone) 的 Oracle Solaris 10 9/10 系统迁移到 ZFS 根文件系统。在尝试迁移之前，请确保创建了 ZFS 根池并且安装和引导了区域。

```
                # zoneadm list -cv
                  ID NAME             STATUS     PATH                           BRAND    IP
                   0 global           running    /                              native   shared
                   2 ufszone          running    /uzone/ufszone                 native   shared
                   3 zfszone          running    /pool/zones/zfszone            native   shared

# lucreate -c ufsBE -n zfsBE -p rpool
Analyzing system configuration.
No name for current boot environment.
Current boot environment is named <zfsBE>.
Creating initial configuration for primary boot environment <zfsBE>.
The device </dev/dsk/c1t0d0s0> is not a root device for any boot environment; cannot get BE ID.
PBE configuration successful: PBE name <ufsBE> PBE Boot Device </dev/dsk/c1t0d0s0>.
Comparing source boot environment <ufsBE> file systems with the file
system(s) you specified for the new boot environment. Determining which
file systems should be in the new boot environment.
Updating boot environment description database on all BEs.
Updating system configuration files.
The device </dev/dsk/c1t1d0s0> is not a root device for any boot environment; cannot get BE ID.
Creating configuration for boot environment <zfsBE>.
Source boot environment is <ufsBE>.
Creating boot environment <zfsBE>.
Creating file systems on boot environment <zfsBE>.
Creating <zfs> file system for </> in zone <global> on <rpool/ROOT/zfsBE>.
Populating file systems on boot environment <zfsBE>.
Checking selection integrity.
Integrity check OK.
Populating contents of mount point </>.
Copying.
Creating shared file system mount points.
Copying root of zone <ufszone> to </.alt.tmp.b-EYd.mnt/uzone/ufszone>.
Creating snapshot for <pool/zones/zfszone> on <pool/zones/zfszone@zfsBE>.
Creating clone for <pool/zones/zfszone@zfsBE> on <pool/zones/zfszone-zfsBE>.
Creating compare databases for boot environment <zfsBE>.
Creating compare database for file system </rpool/ROOT>.
Creating compare database for file system </>.
Updating compare databases on boot environment <zfsBE>.
Making boot environment <zfsBE> bootable.
Creating boot_archive for /.alt.tmp.b-DLd.mnt
updating /.alt.tmp.b-DLd.mnt/platform/sun4u/boot_archive
Population of boot environment <zfsBE> successful.
Creation of boot environment <zfsBE> successful.
# lustatus
Boot Environment           Is       Active Active    Can    Copy
Name                       Complete Now    On Reboot Delete Status
-------------------------- -------- ------ --------- ------ ----------
ufsBE                      yes      yes    yes       no     -
zfsBE                      yes      no     no        yes    -
# luactivate zfsBE
.
.
.
# init 6
.
.
.
# zfs list
NAME                                USED  AVAIL  REFER  MOUNTPOINT
```

```
pool                                    628M  66.3G    19K  /pool
pool/zones                              628M  66.3G    20K  /pool/zones
pool/zones/zfszone                     75.5K  66.3G   627M  /pool/zones/zfszone
pool/zones/zfszone-ufsBE                628M  66.3G   627M  /pool/zones/zfszone-ufsBE
pool/zones/zfszone-ufsBE@zfsBE           98K      -   627M  -
rpool                                  7.76G  59.2G    95K  /rpool
rpool/ROOT                             5.25G  59.2G    18K  /rpool/ROOT
rpool/ROOT/zfsBE                       5.25G  59.2G  5.25G  /
rpool/dump                             2.00G  59.2G  2.00G  -
rpool/swap                              517M  59.7G    16K  -
# zoneadm list -cv
  ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              native   shared
   - ufszone          installed  /uzone/ufszone                 native   shared
   - zfszone          installed  /pool/zones/zfszone            native   shared
```

管理 ZFS 交换和转储设备

Oracle Solaris OS 初始安装期间，或者使用 Live Upgrade 从 UFS 文件系统迁移之后，会在 ZFS 根池中的 ZFS 卷上创建交换区域。例如：

```
# swap -l
swapfile                  dev  swaplo  blocks   free
/dev/zvol/dsk/rpool/swap 256,1      16 4194288 4194288
```

Oracle Solaris OS 初始安装或者使用 Live Upgrade 从 UFS 文件系统迁移期间，会在 ZFS 根池中的 ZFS 卷上创建转储设备。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/t2000
  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，必须使用 `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，请参见第 94 页中的 “支持 ZFS 所要满足的 Oracle Solaris 安装要求和 Oracle Solaris Live Upgrade 要求”。

在安装期间和安装后，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，请参见第 134 页中的 “调整 ZFS 交换设备和转储设备的大小”。

使用 ZFS 交换和转储设备时，请考虑以下问题：

- 必须将单独的 ZFS 卷用于交换区域和转储设备。
- 当前，不支持在 ZFS 文件系统上使用交换文件。

- 如果在安装或升级系统后需要更改交换区域或转储设备，请像在先前的发行版中那样使用 swap 和 dumpadm 命令。有关更多信息，请参见《System Administration Guide: Devices and File Systems》中的第 16 章 “Configuring Additional Swap Space (Tasks)”和《系统管理指南：高级管理》中的第 17 章 “管理系统故障转储信息（任务）”。

有关更多信息，请参见以下各章节：

- 第 134 页中的 “调整 ZFS 交换设备和转储设备的大小”
- 第 136 页中的 “ZFS 转储设备故障排除”

调整 ZFS 交换设备和转储设备的大小

在安装后可能需要调整交换和转储设备的大小，也可能需要重新创建交换卷和转储卷。

- 您可以在初始安装期间调整交换和转储卷的大小。有关更多信息，请参见示例 4–1。
- 您可以在执行 Live Upgrade 操作之前创建交换卷和转储卷并确定其大小。例如：

  1. 创建存储池。

     ```
     # zpool create rpool mirror c0t0d0s0 c0t1d0s0
     ```

  2. 创建转储设备。

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

  3. 启用转储设备。

     ```
     # dumpadm -d /dev/zvol/dsk/rpool/dump
           Dump content: kernel pages
            Dump device: /dev/zvol/dsk/rpool/dump (dedicated)
     Savecore directory: /var/crash/t2000
       Savecore enabled: yes
        Save compressed: on
     ```

  4. 创建交换卷：

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

  5. 添加或更改新的交换设备时，必须启用交换区域。

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

  6. 针对交换卷在 /etc/vfstab 文件中添加一项。

  Live Upgrade 不会调整现有交换卷和转储卷的大小。

- 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          -
  ```

- 如果当前交换区域未在使用中，可以调整当前交换卷的大小，但必须重新引导系统，交换空间的大小才会增长。

```
# zfs get volsize rpool/swap
NAME        PROPERTY  VALUE    SOURCE
rpool/swap  volsize   4G       local
# zfs set volsize=8g rpool/swap
# zfs get volsize rpool/swap
NAME        PROPERTY  VALUE    SOURCE
rpool/swap  volsize   8G       local
# init 6
```

- 您可以尝试调整交换卷的大小，但最好先将交换设备删除。然后再创新创建它。例如：

```
# swap -d /dev/zvol/dsk/rpool/swap
# zfs create -V 2g rpool/swap
# swap -a /dev/zvol/dsk/rpool/swap
```

- 您可以使用类似如下的配置文件语法在 JumpStart 配置文件中调整交换和转储卷的大小。

```
install_type initial_install
cluster SUNWCXall
pool rpool 16g 2g 2g c0t0d0s0
```

  在此配置文件中，两个 2g 项将交换卷和转储卷的大小各设置为 2 GB。

- 如果已安装的系统上需要更多交换空间，只需添加另一个交换卷。例如：

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

  然后，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
```

  最后，针对第二交换卷添加一项到 `/etc/vfstab` 文件。

定制 ZFS 交换卷和转储卷

如果要删除缺省交换卷和转储卷并在非根（数据）池中重新创建这些卷，请注意以下几点:

- 如果要在非根池中创建交换和转储设备，请勿在 RAIDZ 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。
- 如果使用 Live Upgrade 升级系统，请使用 `-P` 选项将转储设备从 PBE 保留到 ABE。例如：

```
# lucreate -n newBE -P
```

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，会自动创建转储卷。大多数情况下，如果缺省转储卷太小，您只需要调整转储卷的大小。例如，在一个大存储器系统中，转储卷大小增大到 40 GB，如下所示：

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

  调整大转储卷的大小可能是一个耗时的过程。

  如果由于某种原因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/t2000
    Savecore enabled: yes
  ```

- 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，将会显示类似于以下内容的消息：

  ```
  # 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
  ```

  有关确定交换和转储设备大小的信息，请参见《System Administration Guide: Devices and File Systems》中的“Planning for Swap Space”。

- 目前无法将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 的系统都使用新型的引导方式，即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 根文件系统引导系统时，将会在选择用来进行引导的根文件系统中解析引导归档文件和内核文件的路径名。

引导系统进行安装时，在整个安装过程中，RAM 磁盘用于根文件系统。

从 ZFS 文件系统引导不同于从 UFS 文件系统引导，因为对于 ZFS，引导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**可引导的数据集**或 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数据集是由池的 `bootfs` 属性标识的。通过在 `boot -Z` 命令中指定备用可引导数据集可以覆盖此缺省选择。

从镜像 ZFS 根池中的备用磁盘引导

您可以在安装系统时创建镜像 ZFS 根池，也可以在安装后通过附加磁盘来创建镜像 ZFS 根池。有关更多信息，请参见：

- 第 97 页中的“安装 ZFS 根文件系统（Oracle Solaris 初始安装）”
- 第 102 页中的“如何创建镜像 ZFS 根池（安装后）”

请查看以下有关镜像 ZFS 根池的已知问题：

- 如果您使用 `zpool replace` 命令替换根池磁盘，必须使用 `installboot` 或 `installgrub` 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。`installboot` 和 `installgrub` 命令语法如下：
  - SPARC：

    ```
    sparc# installboot -F zfs /usr/platform/‘uname -i‘/lib/fs/zfs/bootblk
    ```

  - x86：

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

- 您可以从镜像 ZFS 根池中的不同设备引导。根据硬件配置，可能需要更新 PROM 或 BIOS 以指定不同的引导设备。

  例如，您可以从以下池中的任一磁盘（`c1t0d0s0` 或 `c1t1d0s0`）进行引导。

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

          NAME          STATE     READ WRITE CKSUM
          rpool         ONLINE       0     0     0
            mirror-0    ONLINE       0     0     0
              c1t0d0s0  ONLINE       0     0     0
              c1t1d0s0  ONLINE       0     0     0
  ```

- SPARC：在 `ok` 提示符下指定备用磁盘。例如：

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

  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@0,0:a'
```

- x86：从相应的 BIOS 菜单中选择镜像 ZFS 根池中的备用磁盘。

  然后，使用类似以下的语法确认系统是从备用磁盘引导：

```
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：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 `luactivate` 命令从任何 BE 引导。

在 Oracle Solaris OS 安装过程中或执行 Live Upgrade 过程中，将会通过 `bootfs` 属性自动指定缺省的 ZFS 根文件系统。

一个池中可能存在多个可引导的数据集。缺省情况下，*/pool-name*`/boot/menu.lst` 文件中的可引导数据集项由池的 `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备用数据集。这样，`menu.lst` 文件就可以包含池中多个根文件系统的项。

当系统中安装了 ZFS 根文件系统或迁移到 ZFS 根文件系统时，类似如下的项会添加到 `menu.lst` 文件：

```
title zfsBE
bootfs rpool/ROOT/zfsBE
title zfs2BE
bootfs rpool/ROOT/zfs2BE
```

创建新 BE 时，将会自动更新 `menu.lst` 文件。

在基于 SPARC 的系统上，提供了两个 ZFS 引导选项：

- 激活 BE 后，您可以使用 boot `-L` 命令显示 ZFS 池中可引导的数据集列表。然后，您可以在列表中选择可引导的数据集之一。此时将会显示有关引导该数据集的详细说明。您可以按照这些说明来引导选定的数据集。
- 您可以使用 boot `-Z` *dataset* 命令引导特定的 ZFS 数据集。

示例 4–11 SPARC：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 `luactivate` 命令指定缺省 BE。

例如，以下 `lustatus` 输出显示有两个 ZFS BE 可用：

示例 4–11 SPARC：从特定的 ZFS 引导环境引导 （续）

```
# lustatus
Boot Environment           Is       Active Active    Can    Copy
Name                       Complete Now    On Reboot Delete Status
-------------------------- -------- ------ --------- ------ ----------
zfsBE                      yes      no     no        yes    -
zfs2BE                     yes      yes    yes       no     -
```

如果基于 SPARC 的系统上有多个 ZFS BE，您可以使用 `boot -L` 命令从非缺省 BE 引导。然而，从 `boot -L` 会话引导的 BE 不会向缺省 BE 一样重置，`bootfs` 属性也不会进行更新。如果要使从 `boot -L` 会话引导的 BE 成为缺省 BE，必须使用 `luactivate` 命令激活它。

例如：

```
ok boot -L
Rebooting with command: boot -L
Boot device: /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/disk@0  File and args: -L

1 zfsBE
2 zfs2BE
Select environment to boot: [ 1 - 2 ]: 1
To boot the selected entry, invoke:
boot [<root-device>] -Z rpool/ROOT/zfsBE

Program terminated
ok boot -Z rpool/ROOT/zfsBE
```

示例 4–12 SPARC：在故障安全模式下引导 ZFS 文件系统

在基于 SPARC 的系统上，您可以按如下方式从位于 `/platform/‘uname -i‘/failsafe` 的故障安全归档文件引导：

```
ok boot -F failsafe
```

要从特定的 ZFS 可引导数据集引导故障安全归档文件，请使用类似如下的语法：

```
ok boot -Z rpool/ROOT/zfsBE -F failsafe
```

x86：从 ZFS 根文件系统引导

在 Oracle Solaris OS 安装过程中或执行 Live Upgrade 过程期间，以下项会添加到 */pool-name* `/boot/grub/menu.lst` 文件以自动引导 ZFS：

```
title Solaris 10 1/13  X86
findroot (rootfs0,0,a)
kernel$ /platform/i86pc/multiboot -B $ZFS-BOOTFS
module /platform/i86pc/boot_archive
```

```
title Solaris failsafe
findroot (rootfs0,0,a)
kernel /boot/multiboot kernel/unix -s -B console=ttya
module /boot/x86.miniroot-safe
```

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为 ZFS 数据集，则会添加以下选项：

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

示例 4–13 x86：引导 ZFS 文件系统

当系统从 ZFS 文件系统引导时，根设备将由 `-B $ZFS-BOOTFS` 引导参数指定。例如：

```
title Solaris 10 1/13  X86
findroot (pool_rpool,0,a)
kernel /platform/i86pc/multiboot -B $ZFS-BOOTFS
module /platform/i86pc/boot_archive
title Solaris failsafe
findroot (pool_rpool,0,a)
kernel /boot/multiboot kernel/unix -s -B console=ttya
module /boot/x86.miniroot-safe
```

示例 4–14 x86：在故障安全模式下引导 ZFS 文件系统

x86 故障安全归档文件为 `/boot/x86.miniroot-safe`，可以通过从 GRUB 菜单选择 Solaris 故障安全项来引导。例如：

```
title Solaris failsafe
findroot (pool_rpool,0,a)
kernel /boot/multiboot kernel/unix -s -B console=ttya
module /boot/x86.miniroot-safe
```

解决妨碍成功引导的 ZFS 挂载点问题 (Solaris 10 10/08)

更改活动引导环境 (Boot Environment, BE) 的最佳方法是使用 `luactivate` 命令。如果由于错误的修补程序或配置错误而导致活动 BE 引导失败，则从其他 BE 引导的唯一方法是在引导时选择该 BE。您可以选择备用 BE，方法为：在基于 SPARC 的系统上从 PROM 显式引导该备用 BE；在基于 x86 的系统上从 GRUB 菜单显式引导该备用 BE。

由于 Solaris 10 10/08 发行版 Live Upgrade 功能中的错误，非活动 BE 可能无法引导，因为该 BE 中的 ZFS 数据集或区域的 ZFS 数据集有无效的挂载点。如果 BE 有单独的 `/var` 数据集，则上述错误还会阻止该 BE 挂载。

如果区域的 ZFS 数据集有无效的挂载点，通过下列步骤可以更正该挂载点。

▼ 如何解决 ZFS 挂载点问题

1. **从故障安全归档文件引导系统。**

2. **导入池。**

   例如：

   ```
   # zpool import rpool
   ```

3. **查找不正确的临时挂载点。**

   例如：

   ```
   # zfs list -r -o name,mountpoint rpool/ROOT/s10up

       NAME                                MOUNTPOINT
       rpool/ROOT/s10up                    /.alt.tmp.b-VP.mnt/
       rpool/ROOT/s10up/zones              /.alt.tmp.b-VP.mnt//zones
       rpool/ROOT/s10up/zones/zonerootA    /.alt.tmp.b-VP.mnt/zones/zonerootA
   ```

   根 BE (`rpool/ROOT/s10up`) 的挂载点应为 `/`。

   如果由于 `/var` 挂载问题而导致引导失败，请为 `/var` 数据集查找类似不正确的临时挂载点。

4. **重置 ZFS BE 及其数据集的挂载点。**

   例如：

   ```
   # zfs inherit -r mountpoint rpool/ROOT/s10up
   # zfs set mountpoint=/ rpool/ROOT/s10up
   ```

5. **重新引导系统。**

   出现引导特定 BE 的选项（在 OpenBoot PROM 提示符下或 GRUB 菜单中）时，选择刚刚更正了其挂载点的引导环境。

在 ZFS 根环境中进行引导以恢复系统

如果您需要通过引导系统来解决 `root` 口令丢失或类似问题，请按以下过程操作。

您必须引导故障安全模式或从备用介质引导，具体取决于错误的严重程度。一般而言，引导故障安全模式可以解决 `root` 口令丢失或未知问题。

- 第 142 页中的 “如何引导 ZFS 故障安全模式”
- 第 142 页中的 “如何从备用介质引导 ZFS”

如果您需要恢复根池或根池快照，请参见第 143 页中的 “恢复 ZFS 根池或根池快照”。

▼ 如何引导 ZFS 故障安全模式

1 **引导故障安全模式。**

- 在基于 SPARC 的系统上，在 ok 提示符下键入以下内容：

```
ok boot -F failsafe
```

- 在 x86 系统上，从 GRUB 菜单中选择故障安全模式。

2 **当出现提示时，将 ZFS BE 挂载于 /a。**

```
.
.
.
ROOT/zfsBE was found on rpool.
Do you wish to have it mounted read-write on /a? [y,n,?] y
mounting rpool on /a
Starting shell.
```

3 **转到 /a/etc 目录。**

```
# cd /a/etc
```

4 **如有必要，设置 TERM 类型。**

```
# TERM=vt100
# export TERM
```

5 **修正 passwd 或 shadow 文件。**

```
# vi shadow
```

6 **重新引导系统。**

```
# init 6
```

▼ 如何从备用介质引导 ZFS

如果某个问题妨碍系统成功引导，或者发生其他严重问题，您必须从网络安装服务器或者 Oracle Solaris 安装 DVD 引导，导入根池，挂载 ZFS BE，然后尝试解决问题。

1 **从安装 DVD 或者从网络进行引导。**

- SPARC－选择以下引导方法之一：

```
ok boot cdrom -s
ok boot net -s
```

  如果您未使用 -s 选项，则必须退出安装程序。

- x86－选择网络引导选项或者从本地 DVD 引导。

2 导入根池并指定备用挂载点。例如：

```
# zpool import -R /a rpool
```

3 挂载 ZFS BE。例如：

```
# zfs mount rpool/ROOT/zfsBE
```

4 从 /a 目录访问 ZFS BE 内容。

```
# cd /a
```

5 重新引导系统。

```
# init 6
```

恢复 ZFS 根池或根池快照

以下各节描述了如何执行下列任务：

- 第 143 页中的 “如何替换 ZFS 根池中的磁盘”
- 第 145 页中的 “如何创建根池快照”
- 第 147 页中的 “如何重新创建 ZFS 根池和恢复根池快照”
- 第 148 页中的 “如何从故障安全引导回滚根池快照”

▼ 如何替换 ZFS 根池中的磁盘

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较小的磁盘。
- 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，必须在替换根池磁盘前从备用介质（如 DVD 或网络）进行引导。

在镜像根池配置中，您可以尝试替换磁盘而不必从备用介质引导。可以使用 zpool replace 命令替换发生故障的磁盘。或者，如果有额外的磁盘，可以使用 zpool attach 命令。有关附加额外磁盘和分离根池磁盘的示例，请参见本节中的过程。

有些硬件要求您在尝试通过 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
# zpool replace rpool c1t0d0s0
# zpool online rpool c1t0d0s0
# zpool status rpool
```

```
<Let disk resilver before installing the boot blocks>
SPARC# installboot -F zfs /usr/platform/‘uname -i‘/lib/fs/zfs/bootblk /dev/rdsk/c1t0d0s0
x86# installgrub /boot/grub/stage1 /boot/grub/stage2 /dev/rdsk/c1t9d0s0
```

对于某些硬件，插入替换磁盘后不必使其联机或进行重新配置。

您必须标识当前磁盘和新磁盘的引导设备路径名，以便测试从替换磁盘引导，如果替换磁盘发生故障，也可以手动从现有磁盘引导。在以下过程的示例中，当前根池磁盘 (c1t10d0s0) 的路径名为：

```
/pci@8,700000/pci@3/scsi@5/sd@a,0
```

替换引导磁盘 c1t9d0s0 的路径名为：

```
/pci@8,700000/pci@3/scsi@5/sd@9,0
```

1. **物理连接替换磁盘或新磁盘。**

2. **确认新磁盘具有 SMI 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，请参见以下参考文档：

   - SPARC：《System Administration Guide: Devices and File Systems》中的“How to Set Up a Disk for a ZFS Root File System”
   - x86：《System Administration Guide: Devices and File Systems》中的“How to Set Up a Disk for a ZFS Root File System”

3. **将新磁盘连接到根池。**

   例如：

   ```
   # zpool attach rpool c1t10d0s0 c1t9d0s0
   ```

4. **确认根池状态。**

   例如：

   ```
   # zpool status rpool
     pool: rpool
    state: ONLINE
   status: One or more devices is currently being resilvered.  The pool will
           continue to function, possibly in a degraded state.
   action: Wait for the resilver to complete.
    scrub: resilver in progress, 25.47% done, 0h4m to go
   config:

           NAME           STATE     READ WRITE CKSUM
           rpool          ONLINE       0     0     0
             mirror-0     ONLINE       0     0     0
               c1t10d0s0  ONLINE       0     0     0
               c1t9d0s0   ONLINE       0     0     0

   errors: No known data errors
   ```

5 **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  -
```

6 **验证您是否可以从新磁盘引导。**

例如，在基于 SPARC 的系统上，应使用类似以下的语法：

```
ok boot /pci@8,700000/pci@3/scsi@5/sd@9,0
```

7 **如果系统从新磁盘引导，则分离旧磁盘。**

例如：

```
# zpool detach rpool c1t10d0s0
```

8 **重置缺省引导设备，设置系统自动从新磁盘引导。**

- SPARC－在 SPARC 引导 PROM 下使用 eeprom 命令或 setenv 命令。
- x86－重新配置系统 BIOS。

▼ 如何创建根池快照

可以创建根池快照以便进行恢复。创建根池快照的最好方法是对根池执行递归快照。

下面的过程将创建递归根池快照，并将快照以文件和快照的形式存储到远程系统上的池中。如果根池失败，可以使用 NFS 挂载远程数据集，快照文件可以放入重新创建的池中。您也可以将根池快照以实际快照的形式存储到远程系统上的池中。从远程系统发送和接收快照比较复杂，因为在从 Oracle Solaris OS miniroot 引导要修复的系统时，您必须配置 ssh 或使用 rsh。

在根池恢复中，以文件或快照形式验证远程存储的快照是一个重要步骤。无论何种方法，都应按照例程重新创建快照，例如在池配置更改或 Solaris OS 升级时。

在以下过程中，系统从 zfsBE 引导环境引导。

1. 在远程系统上创建池和文件系统以存储快照。

   例如：

   ```
   remote# zfs create rpool/snaps
   ```

2. 与本地系统共享文件系统。

   例如：

   ```
   remote# zfs set sharenfs='rw=local-system,root=local-system' rpool/snaps
   # share
   -@rpool/snaps   /rpool/snaps   sec=sys,rw=local-system,root=local-system   ""
   ```

3. 创建根池的递归快照。

   ```
   local# zfs snapshot -r rpool@snap1
   local# zfs list -r rpool
   # NAME                      USED  AVAIL  REFER  MOUNTPOINT
   rpool                    15.1G   119G   106K  /rpool
   rpool@snap1                  0      -   106K  -
   rpool/ROOT               5.00G   119G    31K  legacy
   rpool/ROOT@snap1             0      -    31K  -
   rpool/ROOT/zfsBE         5.00G   119G  5.00G  /
   rpool/ROOT/zfsBE@snap1       0      -  5.00G  -
   rpool/dump               2.00G   120G  1.00G  -
   rpool/dump@snap1             0      -  1.00G  -
   rpool/export               63K   119G    32K  /export
   rpool/export@snap1           0      -    32K  -
   rpool/export/home          31K   119G    31K  /export/home
   rpool/export/home@snap1      0      -    31K  -
   rpool/swap               8.13G   123G  4.00G  -
   rpool/swap@snap1             0      -  4.00G  -
   ```

4. 将根池快照发送到远程系统。

   例如，要将根池快照作为文件发送到远程池，请使用类似以下的语法：

   ```
   local# zfs send -Rv rpool@snap1 > /net/remote-system/rpool/snaps/rpool.snap1
   sending from @ to rpool@snap1
   sending from @ to rpool/ROOT@snap1
   sending from @ to rpool/ROOT/s10zfsBE@snap1
   sending from @ to rpool/dump@snap1
   sending from @ to rpool/export@snap1
   sending from @ to rpool/export/home@snap1
   sending from @ to rpool/swap@snap1

   local# zfs send -Rv rpool@snap1 > /net/remote-system/rpool/snaps/rpool.snap1
   sending from @ to rpool@snap1
   sending from @ to rpool/export@snap1
   sending from @ to rpool/export/home@snap1
   sending from @ to rpool/ROOT@snap1
   sending from @ to rpool/ROOT/zfsBE@snap1
   sending from @ to rpool/dump@snap1
   sending from @ to rpool/swap@snap1
   ```

要将根池快照作为快照发送到远程池，请使用类似以下的语法：

```
local# zfs send -Rv rpool@snap1 | ssh remote-system zfs receive
-Fd -o canmount=off tank/snaps
sending from @ to rpool@snap1
sending from @ to rpool/export@snap1
sending from @ to rpool/export/home@snap1
sending from @ to rpool/ROOT@snap1
sending from @ to rpool/ROOT/zfsBE@snap1
sending from @ to rpool/dump@snap1
sending from @ to rpool/swap@snap1
```

▼ 如何重新创建 ZFS 根池和恢复根池快照

在此过程中，假设下列条件存在：

- 无法恢复 ZFS 根池。
- ZFS 根池快照存储在远程系统上并通过 NFS 共享。

在本地系统上执行所有步骤。

1. **从安装 DVD 或者从网络进行引导。**
   - SPARC－选择以下引导方法之一：

     ```
     ok boot net -s
     ok boot cdrom -s
     ```

     如果您未使用 `-s` 选项，则需要退出安装程序。
   - x86－选择从 DVD 或网络进行引导的选项。然后，退出安装程序。
2. **如果已经将根池快照作为文件发送到远程系统，请挂载远程快照文件系统。**
   例如：

   ```
   # mount -F nfs remote-system:/rpool/snaps /mnt
   ```

   如果您尚未配置网络服务，则可能需要指定 *remote-system* 的 IP 地址。
3. **如果根池磁盘被替换，并且不包含 ZFS 可用的磁盘标签，则必须为磁盘重新设置标签。**
   有关为磁盘重新设置标签的更多信息，请参见以下参考文档：
   - SPARC：《System Administration Guide: Devices and File Systems》中的“How to Set Up a Disk for a ZFS Root File System”
   - x86：《System Administration Guide: Devices and File Systems》中的“How to Set Up a Disk for a ZFS Root File System”

4 **重新创建根池。**

例如：

```
# zpool create -f -o failmode=continue -R /a -m legacy -o cachefile=
/etc/zfs/zpool.cache rpool c1t1d0s0
```

5 **恢复根池快照。**

此步骤可能会花费一些时间。例如：

```
# cat /mnt/rpool.snap1 | zfs receive -Fdu rpool
```

使用 `-u` 选项表示 `zfs receive` 操作完成时不挂载恢复的归档文件。

要恢复存储在远程系统上的池中的实际根池快照，请使用类似以下的语法：

```
# ssh remote-system zfs send -Rb tank/snaps/rpool@snap1 | zfs receive -F rpool
```

6 **验证根池数据集是否已恢复。**

例如：

```
# zfs list
```

7 **设置根池 BE 上的 `bootfs` 属性。**

例如：

```
# zpool set bootfs=rpool/ROOT/zfsBE rpool
```

8 **在新磁盘上安装引导块。**

- SPARC：

  ```
  # installboot -F zfs /usr/platform/‘uname -i‘/lib/fs/zfs/bootblk /dev/rdsk/c1t1d0s0
  ```

- x86：

  ```
  # installgrub /boot/grub/stage1 /boot/grub/stage2 /dev/rdsk/c1t1d0s0
  ```

9 **重新引导系统。**

```
# init 6
```

▼ 如何从故障安全引导回滚根池快照

此过程假设现有根池快照可用。在本例中，本地系统提供这些根池快照。

```
# zfs snapshot -r rpool@snap1
# zfs list -r rpool
NAME                       USED  AVAIL  REFER  MOUNTPOINT
rpool                     7.84G  59.1G   109K  /rpool
rpool@snap1                 21K      -   106K  -
rpool/ROOT                4.78G  59.1G    31K  legacy
rpool/ROOT@snap1             0      -    31K  -
```

```
rpool/ROOT/s10zfsBE        4.78G  59.1G  4.76G  /
rpool/ROOT/s10zfsBE@snap1  15.6M      -  4.75G  -
rpool/dump                 1.00G  59.1G  1.00G  -
rpool/dump@snap1             16K      -  1.00G  -
rpool/export                 99K  59.1G    32K  /export
rpool/export@snap1           18K      -    32K  -
rpool/export/home            49K  59.1G    31K  /export/home
rpool/export/home@snap1      18K      -    31K  -
rpool/swap                 2.06G  61.2G    16K  -
rpool/swap@snap1              0      -    16K  -
```

1 关闭系统并引导故障安全模式。

```
ok boot -F failsafe
ROOT/zfsBE was found on rpool.
Do you wish to have it mounted read-write on /a? [y,n,?] y
mounting rpool on /a

Starting shell.
```

2 回滚各根池快照。

```
# zfs rollback rpool@snap1
# zfs rollback rpool/ROOT@snap1
# zfs rollback rpool/ROOT/s10zfsBE@snap1
```

3 重新引导至多用户模式。

```
# init 6
```

◆ ◆ ◆ 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（概述）

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。由于文件系统是轻量型的，并且是 ZFS 中的管理中心点，因此可能要创建许多文件系统。

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，请参见第 237 页中的 “ZFS 卷”。

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。将文件系统名称指定为从池名称开始的路径名，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第 27 页中的 “ZFS 组件命名要求”中所述的命名要求。

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
```

如果新文件系统创建成功，则 ZFS 会自动挂载该文件系统。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，请参见第 171 页中的 “管理 ZFS 挂载点”。

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，请参见第 154 页中的 “ZFS 属性介绍”。

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，请参见第 172 页中的 “自动挂载点”。

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，那么即使是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 – 使用 `zfs destroy` 命令的 `-f`、`-r` 或 `-R` 选项时不会出现确认提示，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，请参见第 177 页中的 “设置 ZFS 配额和预留空间”。

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，请参见第 186 页中的 “重命名 ZFS 快照”。

ZFS 属性介绍

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。

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，请参见第 163 页中的 “ZFS 用户属性”。

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源，表明属性是如何获得的。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，如第 167 页中的 “设置 ZFS 属性”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，请参见第 163 页中的 “ZFS 用户属性”。

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 项的方法。该属性的值包括 discard、noallow、secure 和 passthrough。有关这些值的说明，请参见第 206 页中的 “ACL 属性”。
aclmode	字符串	groupmask	控制在 chmod 操作过程中修改 ACL 项的方法。该属性的值包括 discard、groupmask 和 passthrough。有关这些值的说明，请参见第 206 页中的 “ACL 属性”。
atime	布尔值	on	控制文件被读取后是否更新该文件的访问时间。关闭此属性可避免在读取文件时产生写入流量，并可显著提高性能，但可能会使邮件程序和类似的实用程序相混淆。
available	数字	N/A	只读属性，用于标识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。由于磁盘空间是在池内共享的，因此可用空间会受到多种因素的限制，包括物理池大小、配额、预留空间和池内的其他数据集。 此属性的缩写为 avail。 有关磁盘空间记帐的更多信息，请参见第 28 页中的 “ZFS 磁盘空间记帐”。
canmount	布尔值	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。 有关更多信息，请参见第 162 页中的 “canmount 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。该属性的值包括 on、off、fletcher2、fletcher4 和 sha256。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。

表 5–1 ZFS 本机属性说明 （续）

属性名称	类型	缺省值	说明
compression	字符串	off	对数据集启用或禁用压缩。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。现有数据保持未压缩状态。 此属性的缩写为 compress。
compressratio	数字	N/A	只读属性，用于标识针对数据集实现的压缩比例，表示为乘数。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，用于标识创建数据集的日期和时间。
devices	布尔值	on	控制是否可以打开文件系统中的设备文件。
exec	布尔值	on	控制是否允许执行文件系统中的程序。另外，设置为 off 时，将不允许执行带有 PROT_EXEC 的 mmap(2) 调用。
mounted	布尔值	N/A	只读属性，用于指明文件系统、克隆或快照当前是否已挂载。该属性不适用于卷。此属性的值可以是 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取消挂载状态。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，请参见第 171 页中的“管理 ZFS 挂载点”。

表 5–1 ZFS 本机属性说明 （续）

属性名称	类型	缺省值	说明
primarycache	字符串	all	控制在主高速缓冲存储器 (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。
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）。 非克隆文件系统具有 none 源。
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，请参见第 177 页中的 “设置 ZFS 文件系统的配额”。
readonly	布尔值	off	控制是否可以修改数据集。设置为 on 时，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，请参见第 163 页中的 “recordsize 属性”。
referenced	数字	N/A	只读属性，用于标识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	设置数据集可以使用的磁盘空间量。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。

表 5–1 ZFS 本机属性说明 （续）

属性名称	类型	缺省值	说明
refreservation	数字（或 none）	none	设置为数据集（不包括后代，如快照和克隆）预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	设置确保供某个文件系统及其后代使用的最小磁盘空间量。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，请参见第 180 页中的 “设置 ZFS 文件系统的预留空间”。
secondarycache	字符串	all	控制在二级高速缓存 (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。
setuid	布尔值	on	控制文件系统中是否会标记 setuid 位。
shareiscsi	字符串	off	控制 ZFS 卷是否共享为 iSCSI 目标。该属性的值包括 on、off 和 type=disk。您可能希望对一个文件系统设置 shareiscsi=on，使得该文件系统中的所有 ZFS 卷的缺省设置为共享。但是，对一个文件系统设置此属性没有直接影响。
sharenfs	字符串	off	控制文件系统是否可用于 NFS 中以及使用的选项。如果设置为 on，则会调用不带任何选项的 zfs share 命令。否则，将调用带有与该属性的内容等效的选项的 zfs share 命令。如果设置为 off，则使用传统的 share 和 unshare 命令以及 dfstab 文件来管理文件系统。 有关共享 ZFS 文件系统的更多信息，请参见第 175 页中的 “共享和取消共享 ZFS 文件系统”。

表 5–1 ZFS 本机属性说明 （续）

属性名称	类型	缺省值	说明
snapdir	字符串	hidden	控制 .zfs 目录在文件系统根目录中是隐藏还是可见。有关使用快照的更多信息，请参见第 183 页中的 “ZFS 快照概述”。
type	字符串	N/A	只读属性，用于将数据集类型标识为 filesystem（文件系统或克隆）、volume 或 snapshot。
used	数字	N/A	只读属性，用于标识数据集及其所有后代占用的磁盘空间 量。 有关详细说明，请参见第 161 页中的 “used 属性”。
usedbychildren	数字	off	只读属性，用于标识此数据集的子项占用的磁盘空间量；如果所有数据集子项都被销毁，将释放该空间。此属性的缩写为 usedchild。
usedbydataset	数字	off	只读属性，用于标识数据集本身占用的磁盘空间量；如果在先销毁所有快照并删除所有 refreservation 预留空间后销毁数据集，将释放该空间。此属性的缩写为 usedds。
usedbyrefreservation	数字	off	只读属性，用于标识针对数据集设置的 refreservation 占用的磁盘空间量；如果删除 refreservation，将释放该空间。此属性的缩写为 usedrefreserv。
usedbysnapshots	数字	off	只读属性，用于标识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	表示文件系统在磁盘上的版本，它与池版本无关。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。
volsize	数字	N/A	可为卷指定卷的逻辑大小。 有关详细说明，请参见第 163 页中的 “volsize 属性”。
volblocksize	数字	8 KB	可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。范围位于 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。

表 5–1 ZFS 本机属性说明 （续）

属性名称	类型	缺省值	说明
zoned	布尔值	N/A	指示是否已将文件系统添加至非全局区域。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，请参见第 239 页中的 “在安装了区域的 Solaris 系统中使用 ZFS”。
xattr	布尔值	on	指明此文件系统是否启用 (on) 或禁用 (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 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 对其进行了描述。

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

  有关详细信息，请参见第 161 页中的 “used 属性”。
- 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，请参见第 28 页中的 “ZFS 磁盘空间记帐”。

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，请参见第 29 页中的 “空间不足行为”。

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。

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 zfs create 命令进行设置，请分别参见第 167 页中的 “设置 ZFS 属性”和第 152 页中的 “创建 ZFS 文件系统”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。有关配额和预留空间的更多信息，请参见第 177 页中的 “设置 ZFS 配额和预留空间”。

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 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 对其进行了描述。

- aclinherit

  有关详细说明，请参见第 206 页中的 “ACL 属性”。

- aclmode

  有关详细说明，请参见第 206 页中的 “ACL 属性”。

- atime
- canmount
- checksum
- compression

- copies
- devices
- exec
- mountpoint
- primarycache
- quota
- readonly
- recordsize

  有关详细说明，请参见第 163 页中的 “recordsize 属性”。
- refquota
- refreservation
- reservation
- secondarycache
- shareiscsi
- setuid
- snapdir
- version
- volsize

  有关详细说明，请参见第 163 页中的 “volsize 属性”。
- 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
```

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 128 KB。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。

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。

尽管并不建议，但可以通过为 -zfs create -V 指定 s 标志或在创建卷后更改预留空间来创建稀疏卷。**稀疏卷**指预留空间与卷大小不相等的卷。对于稀疏卷，预留空间中不会反映对 volsize 的更改。

有关使用卷的更多信息，请参见第 237 页中的 “ZFS 卷”。

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，请参见第 154 页中的 “ZFS 属性介绍”。

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
```

以下示例展示了如何显示关于 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，请参见第 154 页中的 “ZFS 属性介绍”。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 sharenfs 和 mountpoint 属性值。

```
# zfs list -r -o name,sharenfs,mountpoint users/home
NAME                        SHARENFS  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
```

可以使用 -t 选项指定要显示的数据集的类型。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 选项可后跟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
```

使用 -H 选项可从生成的输出中省略 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，请参见第 161 页中的 “可设置的 ZFS 本机属性”。

zfs set 命令采用 *property=value* 格式的属性/值序列，然后是数据集名称。zfs set 的每次调用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 和 sharenfs 除外。这两个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，请参见第 164 页中的“列出基本 ZFS 信息”。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)

源值	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，请参见第 174 页中的 “使用临时挂载属性”。
-（无）	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:

注 – `casesensitivity`、`nbmand`、`normalization`、`sharesmb`、`utf8only` 和 `vscan` 属性在 Oracle Solaris 10 发行版中并不能全面使用，因为 Oracle Solaris 10 发行版不支持 Oracle Solaris SMB 服务。

```
# 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
```

通过 `zfs get` 的 `-s` 选项，可以按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`。以下示例显示了已在 `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 选项省去标头信息并用 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，如第 154 页中的 “ZFS 属性介绍”部分所述。
- 输出逗号分隔的字面字段、name、value、property 和 source 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 选项为 export/home 及其后代检索文件系统名称和 used 属性值，同时忽略标题输出：

```
# 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，请参见第 43 页中的 “创建 ZFS 存储池”。

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。

要阻止文件系统被挂载，须将 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，请参见第 162 页中的 “canmount 属性”。

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，请参见第 173 页中的 “传统挂载点”。

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                       -
```

mountpoint 属性更改时，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 文件中的项可能看起来如下例所示：

```
#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，请参见第 24 页中的 “事务性语义”。

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
```

缺省情况下，ZFS 不允许在非空目录的顶层进行挂载。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。

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 命令可以采用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

只要设置 sharenfs 属性，ZFS 便能自动共享文件系统。使用此属性时，不必在共享新文件系统时修改 /etc/dfs/dfstab 文件。sharenfs 属性是要传递给 share 命令的选项的逗号分隔列表。值 on 是缺省的共享选项的别名，它向所有用户提供 read/write 权限。值 off 指明文件系统不是由 ZFS 进行管理，但可通过传统方法（如 /etc/dfs/dfstab 文件）来共享。在引导过程中将会共享 sharenfs 属性不为 off 的所有文件系统。

控制共享语义

缺省情况下，所有文件系统都不共享。要共享新文件系统，请使用类似如下的 zfs set 语法：

```
# zfs set sharenfs=on tank/home/eric
```

sharenfs属性是继承的，如果文件系统继承的属性不为 off，则这些文件系统在创建时会自动进行共享。例如：

```
# zfs set sharenfs=on tank/home
# zfs create tank/home/bill
# zfs create tank/home/mark
# zfs set sharenfs=ro tank/home/bob
```

tank/home/bill 和 tank/home/mark 最初以可写方式共享，因为它们从 tank/home 继承了 sharenfs 属性。将该属性设置为 ro（只读）后，无论为 tank/home 设置的 sharenfs 属性为何值，都会以只读方式共享 tank/home/mark。

取消共享 ZFS 文件系统

尽管大多数文件系统都可在引导、创建和销毁过程中自动共享或取消共享，但文件系统有时候需要显式取消共享。为此，请使用 zfs unshare 命令。例如：

```
# zfs unshare tank/home/mark
```

此命令会取消共享 tank/home/mark 文件系统。要取消共享系统中的所有 ZFS 文件系统，需要使用 -a 选项。

```
# zfs unshare -a
```

共享 ZFS 文件系统

通常而言，ZFS 在引导和创建时共享文件系统的自动行为足以满足一般操作的需要。如果由于某些原因取消共享了某个文件系统，则可使用 zfs share 命令再次将其共享。例如：

```
# zfs share tank/home/mark
```

另外，还可以通过使用 -a 选项共享系统中的所有 ZFS 文件系统。

```
# zfs share -a
```

传统共享行为

如果 sharenfs 属性设置为 off，则 ZFS 在任何时候都不会尝试共享或取消共享文件系统。借助此值，可以通过传统方法（如 /etc/dfs/dfstab 文件）来管理文件系统共享。

与传统的 mount 命令不同，传统的 share 和 unshare 命令在 ZFS 文件系统中仍然可以运行。因此，可以使用与 sharenfs 属性的选项不同的选项来手动共享文件系统。不鼓励使用这种管理模型。请选择完全通过 ZFS 或完全通过 /etc/dfs/dfstab 文件来管理 NFS 共享内容。ZFS 管理模型与传统模型相比，设计更为简单，所需进行的工作越少。

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，请考虑以下几点：

- 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 属性时无效。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，请参见第 177 页中的 “设置 ZFS 文件系统的配额”和第 180 页中的 “设置 ZFS 文件系统的预留空间”。

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。

不能将配额设置为比文件系统当前使用的空间小的数量。例如：

```
# zfs set quota=10K tank/home/jeff
cannot set property for 'tank/home/jeff':
size is less than current used or reserved space
```

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，它会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。

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，此命令会在文件系统运行老版本时通知您。

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.
```

◆ ◆ ◆ 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：

- 可在系统重新引导后存留下来。
- 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，请参见第 192 页中的 “发送和接收 ZFS 数据”。

- 第 184 页中的 “创建和销毁 ZFS 快照”
- 第 187 页中的 “显示和访问 ZFS 快照”
- 第 188 页中的 “回滚 ZFS 快照”

创建和销毁 ZFS 快照

快照是使用 `zfs snapshot` 命令创建的，该命令将要创建的快照的名称用作其唯一参数。快照名称按如下方式指定：

filesystem@snapname
volume@snapname

快照名称必须满足第 27 页中的 “ZFS 组件命名要求”中所述的命名要求。

在以下示例中，将创建一个 `tank/home/cindy` 的快照，其名称为 `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，请参见第 152 页中的 “销毁 ZFS 文件系统”。

保持 ZFS 快照

如果存在不同的原子快照策略，导致旧的快照由于不再存在于发送侧而被 zfs receive 不小心销毁，则可能需要考虑使用本 Solaris 发行版中的快照保持功能。

保持快照可以防止它被销毁。此外，当一个带有克隆的快照等待删除最后一个克隆时，该功能允许使用 zfs destroy - d 命令删除该快照。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

您可以通过 listsnapshots 池属性启用或禁用 zfs list 输出中的快照列表显示。缺省情况下，此属性处于启用状态。

如果禁用了此属性,则可以使用 zfs list -t snapshot 命令来显示快照信息。或者，启用 listsnapshots 池属性。例如：

```
# zpool get listsnapshots tank
NAME  PROPERTY       VALUE      SOURCE
tank  listsnapshots  on         default
# zpool set listsnapshots=off tank
# zpool get listsnapshots tank
NAME  PROPERTY       VALUE      SOURCE
tank  listsnapshots  off          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，如果在 /home/cindy 处挂载了 tank/home/cindy，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 属性值如何被占用的附加信息。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                      59.1G  7.84G       21K    109K              0       7.84G
rpool@snap1                    -    21K         -       -              -          -
rpool/ROOT                 59.1G  4.78G         0     31K              0       4.78G
rpool/ROOT@snap1               -     0          -       -              -          -
rpool/ROOT/zfsBE           59.1G  4.78G     15.6M   4.76G              0          0
rpool/ROOT/zfsBE@snap1         -  15.6M         -       -              -          -
rpool/dump                 59.1G  1.00G       16K   1.00G              0          0
rpool/dump@snap1               -    16K         -       -              -          -
rpool/export               59.1G    99K       18K     32K              0         49K
rpool/export@snap1             -    18K         -       -              -          -
rpool/export/home          59.1G    49K       18K     31K              0          0
rpool/export/home@snap1        -    18K         -       -              -          -
rpool/swap                 61.2G  2.06G         0     16K           2.06G          0
rpool/swap@snap1               -     0          -       -              -          -
```

有关这些属性的说明，请参见表 5–1。

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，会将 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，请参见第 167 页中的“设置 ZFS 属性”。

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，将从 `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 sharenfs=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，请参见第 191 页中的 “销毁 ZFS 克隆”。

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，请参见第 183 页中的 “ZFS 快照概述”。
- **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 Solaris 10 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。不过，下面讨论的自包含递归流是以没有外部相关项的方式创建的。

  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，请牢记以下要点：

- 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
  ```

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（如果有）。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，请参见第 193 页中的 “识别 ZFS 快照流”。
- 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。

示例 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
```

示例 6–1 发送和接收复杂的 ZFS 快照流 （续）

```
# 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 草案 ACL 规范。基于 POSIX 样式的 ACL 用来保护 UFS 文件，并通过 NFSv4 之前的 NFS 版本进行转换。

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。
- 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，请参见第 205 页中的 “ACL 继承”。

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 也由多个访问控制项 (Access Control Entry, ACE) 构成。

POSIX 样式的 ACL 使用单个项来定义允许和拒绝的权限。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，不能根据任何定义一组权限的单个 ACE 来推断是否允许或拒绝该 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，请参见第 192 页中的 “使用其他备份产品保存 ZFS 数据”。

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。
- **非普通 ACL**－不仅包含所有者、组和用户的条目，还包含其他条目，或者包含继承标志集，或者是其中的条目以非传统顺序排列。

用于设置普通 ACL 的语法

`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
```

用于设置非普通 ACL 的语法

```
chmod [options] A[index]{+|=}user|group:name:access-permissions
/...[:inheritance-flags]:deny | allow file

chmod [options] A-user|group:name:access-permissions /...[:inheritance-flags]:deny |
allow file ...

chmod [options] A[index]- file
```

owner@, group@, everyone@
: 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。

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。

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。

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。

deny | allow
: 标识授予还是拒绝访问权限。

在以下示例中，owner@、group@ 或 everyone@ 没有 *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:gozer:list_directory/read_data/execute:allow
```

显示的 ACL 项与以下内容类似：

```
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
```

下表介绍了 ACL 项类型，即所有者、组和其他对象的 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。
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。
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。

表 7–2 ACL 访问特权 （续）

访问特权	缩写访问特权	说明
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。 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

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其本来要继承的 ACL，但不忽略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。

下表介绍了可选的继承标志。

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。

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。该属性的值包括：
  - 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，请参见示例 7–13。

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 和第 213 页中的 “以详细格式对 ZFS 文件设置 ACL 继承”。

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
```

在本例中，每个用户类别（owner@、group@、everyone@）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* **规范**来确定 ACL 的格式。有关 *acl* **规范**的说明，请参见第 202 页中的 “ACL 设置语法的说明”。

- 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，请参见第 218 页中的 “以缩写格式设置和显示 ZFS 文件的 ACL”。

示例 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 中将只包含传统的 UNIX 条目、用户、组以及其他内容。

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
```

示例 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 `gozer` 添加了对 `test.dir` 目录的 `read_data/execute` 权限。

示例 7–2 设置 ZFS 文件的非普通 ACL （续）

```
# chmod A+user:gozer:read_data/execute:allow test.dir
# ls -dv test.dir
drwxr-xr-x+  2 root     root           2 Jul 20 14:23 test.dir
     0:user:gozer:list_directory/read_data/execute:allow
     1:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
         /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 gozer 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
```

示例 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
```

示例 7–3 与 ZFS 文件权限的 ACL 交互 （续）

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 gozer 的读取权限。

```
# chmod A=user:gozer:read_data:allow file.3
# ls -v file.3
----------+  1 root     root        2440 Jul 20 14:28 file.3
     0:user:gozer:read_data:allow
```

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
```

示例 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:gozer:read_data:file_inherit:deny
     2:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
```

示例 7–4 恢复 ZFS 文件的普通 ACL （续）

```
         /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 gozer 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
```

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，请参见第 205 页中的 “ACL 继承”。

示例 7–5 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 gozer 应用了针对 test.dir 的非普通 ACE read_data/write_data/execute。

```
# chmod A+user:gozer:read_data/write_data/execute:allow test.dir
# ls -dv test.dir
drwxr-xr-x+  2 root     root           2 Jul 20 14:53 test.dir
     0:user:gozer:list_directory/read_data/add_file/write_data/execute:allow
     1:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
         /append_data/read_xattr/write_xattr/execute/delete_child
         /read_attributes/write_attributes/read_acl/write_acl/write_owner
```

示例 7–5 授予缺省 ACL 继承 （续）

```
        /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 gozer 的 ACE。如果对 sub.dir 的权限授予用户 gozer 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
```

示例 7–6 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 gozer 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:gozer:read_data/write_data:file_inherit:allow test2.dir
# ls -dv test2.dir
drwxr-xr-x+  2 root     root           2 Jul 20 14:55 test2.dir
     0:user:gozer:read_data/write_data:file_inherit:allow
     1:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
        /append_data/read_xattr/write_xattr/execute/delete_child
        /read_attributes/write_attributes/read_acl/write_acl/write_owner
        /synchronize:allow
     2:group@:list_directory/read_data/read_xattr/execute/read_attributes
        /read_acl/synchronize:allow
     3:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
        /read_acl/synchronize:allow
```

在以下示例中，用户 gozer 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 gozer 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:gozer:read_data:inherited:allow
     1:owner@:read_data/write_data/append_data/read_xattr/write_xattr
```

示例 7–6 对文件和目录授予 ACL 继承 （续）

```
        /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 gozer 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 gozer 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 Jun 23 15:21 test2.dir/subdir.2
     0:user:gozer:list_directory/read_data/add_file/write_data:file_inherit
        /inherit_only: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
```

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 gozer 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:gozer:read_data/write_data/execute:file_inherit/dir_inherit:allow
test3.dir
# ls -dv test3.dir
drwxr-xr-x+  2 root     root           2 Jul 20 15:00 test3.dir
     0:user:gozer:list_directory/read_data/add_file/write_data/execute
        :file_inherit/dir_inherit:allow
     1:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
        /append_data/read_xattr/write_xattr/execute/delete_child
        /read_attributes/write_attributes/read_acl/write_acl/write_owner
        /synchronize:allow
     2:group@:list_directory/read_data/read_xattr/execute/read_attributes
        /read_acl/synchronize:allow
     3:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
        /read_acl/synchronize:allow

# 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:gozer:read_data:allow
```

示例 7–6 对文件和目录授予 ACL 继承 （续）

```
     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:gozer: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 `gozer` 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 `gozer` 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:gozer:read_data/write_data/execute:file_inherit/no_propagate:allow
test4.dir
# ls -dv test4.dir
drwxr--r--+  2 root     root           2 Mar  1 12:11 test4.dir
     0:user:gozer:list_directory/read_data/add_file/write_data/execute
         :file_inherit/no_propagate:allow
     1:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
         /append_data/read_xattr/write_xattr/execute/delete_child
         /read_attributes/write_attributes/read_acl/write_acl/write_owner
         /synchronize:allow
     2:group@:list_directory/read_data/read_xattr/read_attributes/read_acl
         /synchronize:allow
     3:everyone@:list_directory/read_data/read_xattr/read_attributes/read_acl
         /synchronize:allow
```

如以下示例所示，基于所属组的权限降低了用户 `gozer` 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 Jun 23 15:28 test4.dir/file.4
     0:user:gozer: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
```

示例 7–6 对文件和目录授予 ACL 继承 （续）

```
     2:group@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
     3:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
         :allow
```

示例 7–7 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 `gozer` 将继承 `test4.dir` 上应用的 ACL，如下所示：

```
# zfs set aclinherit=passthrough tank/cindy
# touch test4.dir/file.4
# ls -v test4.dir/file.4
-rw-r--r--+  1 root     root           0 Jun 23 15:35 test4.dir/file.4
     0:user:gozer: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
```

示例 7–8 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:gozer:read_data/write_data/execute:dir_inherit:allow test5.dir
# ls -dv test5.dir
drwxr-xr-x+  2 root     root           2 Jul 20 14:18 test5.dir
     0:user:gozer:list_directory/read_data/add_file/write_data/execute
         :dir_inherit:allow
     1:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
         /append_data/read_xattr/write_xattr/execute/delete_child
         /read_attributes/write_attributes/read_acl/write_acl/write_owner
         /synchronize:allow
     2:group@:list_directory/read_data/read_xattr/execute/read_attributes
         /read_acl/synchronize:allow
     3:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
         /read_acl/synchronize:allow
```

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
```

示例 7–8 ACL 继承模式设置为 Discard 时的 ACL 继承 （续）

```
     2:everyone@:list_directory/read_data/read_xattr/read_attributes/read_acl
         /synchronize:allow
```

示例 7–9 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:gozer:read_data:file_inherit:deny,user:lp:read_data:file_inherit:allow
test6.dir
# ls -dv test6.dir
drwxr-xr-x+  2 root     root           2 Jul 20 14:22 test6.dir
     0:user:gozer:read_data:file_inherit:deny
     1:user:lp:read_data:file_inherit:allow
     2:owner@:list_directory/read_data/add_file/write_data/add_subdirectory
         /append_data/read_xattr/write_xattr/execute/delete_child
         /read_attributes/write_attributes/read_acl/write_acl/write_owner
         /synchronize:allow
     3:group@:list_directory/read_data/read_xattr/execute/read_attributes
         /read_acl/synchronize:allow
     4:everyone@:list_directory/read_data/read_xattr/execute/read_attributes
         /read_acl/synchronize:allow
```

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 Jun 15 12:19 test6.dir/file.6
     0:user:gozer:read_data:inherited:deny
     1:owner@:read_data/write_data/append_data/read_xattr/write_xattr
         /read_attributes/write_attributes/read_acl/write_acl/write_owner
         /synchronize:allow
     2:group@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
     3:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize
         :allow
```

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 和表 7–4 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      206663 Jun 23 15:06 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，请参见第 209 页中的 “以详细格式设置和显示 ZFS 文件的 ACL”。

示例 7–10 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      206663 Jun 23 15:06 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 gozer 添加了对 file.1 的 read_data/execute 权限。

```
# chmod A+user:gozer:rx:allow file.1
# ls -V file.1
-rw-r--r--+  1 root     root      206663 Jun 23 15:06 file.1
       user:gozer:r-x-----------:------:allow
           owner@:rw-p--aARWcCos:------:allow
           group@:r-----a-R-c--s:------:allow
```

示例 7–10 以缩写格式设置和显示 ACL （续）

```
         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 gozer 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:gozer:rwx:fd:allow dir.2
# ls -dV dir.2
drwxr-xr-x+  2 root     root           2 Jun 23 16:04 dir.2
        user:gozer:rwx-----------:fd----:allow
            owner@:rwxp--aARWcCos:------: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 gozer 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 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 Jun 23 16:04 dir.2
       user:cindy:rwx-----------:fd----:allow
        user:gozer:rwx-----------:fd----:allow
            owner@:rwxp--aARWcCos:------:allow
            group@:r-x---a-R-c--s:------:allow
         everyone@:r-x---a-R-c--s:------:allow
```

示例 7–11 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 Jun 23 16:10 test1.dir
```

示例 7–11 ACL 继承模式设置为 Passthrough 时的 ACL 继承 （续）

```
            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 Jun 23 16:11 file.1
            owner@:rwxpdDaARWcCos:------:allow
            group@:rwxp----------:------:allow
         everyone@:--------------:------: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 Jun 23 16:13 subdir.1
            owner@:rwxpdDaARWcCos:fd----:allow
            group@:rwxp----------:fd----:allow
         everyone@:--------------:fd----:allow
```

`fd----` 项用于传播继承，在访问控制期间不会被考虑。在本示例中，会使用继承 ACE 不存在的另一目录中的普通 ACL 创建文件。

```
# cd /tank/cindy
# mkdir test2.dir
# cd test2.dir
# touch file.2
# ls -V file.2
-rw-r--r--   1 root     root           0 Jun 23 16:15 file.2
            owner@:rw-p--aARWcCos:------:allow
            group@:r-----a-R-c--s:------:allow
         everyone@:r-----a-R-c--s:------:allow
```

示例 7–12 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。

示例 7–12 ACL 继承模式设置为 Passthrough-X 时的 ACL 继承 （续）

```
# chmod A=owner@:rwxpcCosRrWaAdD:fd:allow,group@:rwxp:fd:allow,everyone@::fd:allow test1.dir
# ls -Vd test1.dir
drwxrwx---+  2 root     root           2 Jun 23 16:17 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 Jun 23 16:18 test1.dir/file1
            owner@:rw-pdDaARWcCos:------:allow
            group@:rw-p----------:------:allow
         everyone@:--------------:------: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:------:allow
                group@:rwxp----------:------:allow
             everyone@:--------------:------: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。

示例 7–13 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
```

示例 7–13 ACL 与对 ZFS 文件的 chmod 操作的交互 （续）

```
                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
```

◆ ◆ ◆ 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《System Administration Guide: Security Services》中的第 III 部分, “Roles, Rights Profiles, and Privileges”。

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。

委托 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。
- 可将各权限的组作为**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	相关项
`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。	

权限（子命令）	说明	相关项
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。	
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 sharenfs 才能创建 NFS 共享。 必须同时具有 share 和 sharesmb 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
- 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
- devices
- exec
- logbias
- mountpoint
- nbmand
- normalization

- primarycache
- quota
- readonly
- recordsize
- refquota
- refreservation
- reservation
- rstchown
- secondarycache
- setuid
- sharenfs
- sharesmb
- 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，请参见第 154 页中的“ZFS 属性介绍”。

授予 ZFS 权限 (zfs allow)

zfs allow 语法如下所示：

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，假定您按如下方式授予了 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
```

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
```

委托 ZFS 权限（示例）

示例 8–1 向单个用户委托权限

在向单个用户授予 create 和 mount 权限时，必须确保该用户对底层挂载点有权限。

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：

示例 8–1 向单个用户委托权限 （续）

```
# su mark
mark$ zfs create tank/home/mark
mark$ ^D
# su lp
$ zfs create tank/home/lp
cannot create 'tank/home/lp': permission denied
```

示例 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任何人的文件系统。

```
# 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
```

示例 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：

示例 8–3 在正确的文件系统级别授予权限 （续）

```
# 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。

示例 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
```

示例 8–4 定义和使用复杂委托权限 （续）

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
```

示例 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
```

示例 8–5 定义和使用 ZFS 委托权限集 （ **续** ）

```
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 委托权限（ 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

示例 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
```

示例 8–7 显示复杂委托管理权限

此示例中的输出指示针对 `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。

示例 8–7 显示复杂委托管理权限 （ 续 ）

- 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 委托权限（ 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。

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，请参见第 250 页中的 “确定 ZFS 存储池的问题”。

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，请参见第 242 页中的 “向非全局区域中添加 ZFS 卷”。

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，请参见第 134 页中的 “调整 ZFS 交换设备和转储设备的大小”。

使用 ZFS 卷作为 Solaris iSCSI 目标

通过设置卷的 `shareiscsi` 属性，可以轻松创建 ZFS 卷作为 iSCSI 目标。例如：

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

创建 iSCSI 目标后，应设置 iSCSI 启动器。有关 Solaris iSCSI 目标和启动器的更多信息，请参见《System Administration Guide: Devices and File Systems》中的第 12 章 “Configuring Oracle Solaris iSCSI Targets (Tasks)”。

注 – 也可以使用 `iscsitadm` 命令来创建和管理 Solaris iSCSI 目标。如果对 ZFS 卷设置 `shareiscsi` 属性，请勿使用 `iscsitadm` 命令再创建同一目标设备。否则将为同一设备创建重复的目标信息。

可以像管理其他 ZFS 数据集一样来管理作为 iSCSI 目标的 ZFS 卷。不过，对于 iSCSI 目标而言，`rename`、`export` 和 `import` 操作的工作方式略有不同。

- 重命名 ZFS 卷时，iSCSI 目标名将保持不变。例如：

  ```
  # zfs rename tank/volumes/v2 tank/volumes/v1
  # iscsitadm list target
  Target: tank/volumes/v1
      iSCSI Name: iqn.1986-03.com.sun:02:984fe301-c412-ccc1-cc80-cf9a72aa062a
      Connections: 0
  ```

- 导出包含共享 ZFS 卷的池会导致目标被删除。导入包含共享 ZFS 卷的池会导致目标被共享。例如：

  ```
  # zpool export tank
  # iscsitadm list target
  # zpool import tank
  # iscsitadm list target
  Target: tank/volumes/v1
      iSCSI Name: iqn.1986-03.com.sun:02:984fe301-c412-ccc1-cc80-cf9a72aa062a
      Connections: 0
  ```

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 Zones 的系统中使用 ZFS。

- 第 243 页中的 “了解 zoned 属性”

有关在具有 ZFS 根文件系统（使用 Oracle Solaris Live Upgrade 进行迁移或修补）的系统上配置区域的信息，请参见第 119 页中的 “使用 Live Upgrade 迁移或升级具有区域的系统 (Solaris 10 10/08)”或第 123 页中的 “使用 Oracle Solaris Live Upgrade 迁移或升级具有区域的系统（最低 Solaris 10 5/09）”。

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级的完全控制。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中的数据集，包括不能超过对委托数据集设置的任何顶层配额。

在安装了 Oracle Solaris Zones 的系统中使用 ZFS 时，请考虑以下事项：

- 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`。
- 当配置了非全局区域时，不要将 ZFS 数据集添加到非全局区域中。请在安装区域后添加 ZFS 数据集。
- 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《系统管理指南：Oracle Solaris Containers－资源管理和 Oracle Solaris Zones》中的第 II 部分, “区域”。
- 如果您将 ZFS 文件系统委托给非全局区域，则必须在使用 Oracle Solaris Live Upgrade 之前从非全局区域删除该文件系统。否则，Oracle Live Upgrade 会因只读文件系统错误而失败。

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
```

此语法可向挂载在 /opt/data 且已配置的 zion 区域中添加 ZFS 文件系统 tank/zone/zion。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 命令以及使用 zonecfg 配置资源类型的更多信息，请参见《系统管理指南：Oracle Solaris Containers－资源管理和 Oracle Solaris Zones》中的第 II 部分, “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 子命令向非全局区域中添加数据集。

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> 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。区域管理员可以设置文件系统属性，也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。

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。

如果使用 Oracle Solaris Live Upgrade 升级带有非全局区域的 ZFS BE，首先请删除所有委托数据集。否则，Oracle Solaris Live Upgrade 会因只读文件系统错误而失败。例如：

```
zonecfg:zion>
zonecfg:zion> remove dataset name=tank/zone/zion
zonecfg:zion1> exit
```

有关区域内所允许的操作的更多信息，请参见第 242 页中的 “在区域内管理 ZFS 属性”。

向非全局区域中添加 ZFS 卷

不能使用 `zonecfg` 命令的 `add dataset` 子命令向非全局区域中添加 ZFS 卷。但是，可以使用 `zonecfg` 命令的 `add device` 子命令向区域中添加卷。

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 卷：

```
# zonecfg -z zion
zion: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:zion> create
zonecfg:zion> add device
zonecfg:zion:device> set match=/dev/zvol/dsk/tank/vol
zonecfg:zion:device> end
```

此语法将 `tank/vol` 卷添加到 `zion` 区域中。

请注意，即使卷不与物理设备对应，向区域中添加原始卷仍然可能存在安全风险。具体来说，区域管理员可能会创建格式错误的文件系统，这在尝试挂载时会使系统发出警告音。有关向区域中添加设备以及相关的安全风险的更多信息，请参见第 243 页中的 “了解 zoned 属性”。

有关向区域添加设备的更多信息，请参见《系统管理指南：Oracle Solaris Containers－资源管理和 Oracle Solaris Zones》中的第 II 部分, “区域”。

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，请参考以下配置：

```
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	是	否	-
tank/home	否	-	-
tank/data	是	否	-
tank/data/matrix	否	-	-
tank/data/zion	是	是	sharenfs、zoned、quota、reservation
tank/data/zion/home	是	是	sharenfs、zoned

请注意，tank/zone/zion 的每个父级都会显示为只读，所有后代都可写，不属于父级分层结构的数据集完全不可见。由于非全局区域不能充当 NFS 服务器，因此区域管理员无法更改 sharenfs 属性。区域管理员不能更改 zoned 属性，否则将产生下节所述的安全风险。

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 sharenfs 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，以便不在全局区域的上下文中解释特定属性。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。

使用 ZFS 备用根池

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**池功能。系统重新引导之后备用根池不会保留，并且所有挂载点都会被修改以与该池的根相关。

创建 ZFS 备用根池

创建备用根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。使用 zpool create -R 选项创建备用根池时，根文件系统的挂载点将自动设置为 /，这与备用根值等效。

在以下示例中，名为 morpheus 的池是通过将 /mnt 作为备用根路径来创建的：

```
# 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 /mnt 的备用根。存储在磁盘上的挂载点是 /，/mnt 的全路径仅在池创建的初始上下文中才会进行解释。然后可以使用 -R **备用根值**语法在不同系统的任意备用根池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
```

导入备用根池

也可以使用备用根来导入池。如果挂载点不是在当前根的上下文中而是在可以执行修复的某个临时目录下解释的，则可以使用此功能进行恢复。在挂载可移除介质时，也可以使用此功能，如上一节所述。

在以下示例中，名为 morpheus 的池是通过将 /mnt 作为备用根路径来导入的。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
```

◆ ◆ ◆ 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：

有关完整的根池恢复的信息，请参见第 143 页中的 “恢复 ZFS 根池或根池快照”。

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**
  - 第 250 页中的 “确定 ZFS 存储池的问题”
  - 第 253 页中的 “解决 ZFS 存储设备问题”
- 第 265 页中的 “解决 ZFS 文件系统问题”
- 第 273 页中的 “修复损坏的 ZFS 配置”
- 第 274 页中的 “修复无法引导的系统”

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` 上的错误。使用以下语法可显示更详细的设备错误。

```
# iostat -En
c0t5000C500335F95E3d0 Soft Errors: 0 Hard Errors: 0 Transport Errors: 0
Vendor: SEAGATE  Product: ST930003SSUN300G Revision: 0B70 Serial No: 110672QFSB
Size: 300.00GB <300000000000 bytes>
Media Error: 0 Device Not Ready: 0 No Device: 0 Recoverable: 0
Illegal Request: 0 Predictive Failure Analysis: 0
c0t5000C500335FC3E7d0 Soft Errors: 0 Hard Errors: 0 Transport Errors: 0
Vendor: SEAGATE  Product: ST930003SSUN300G Revision: 0B70 Serial No: 110672TE67
Size: 300.00GB <300000000000 bytes>
Media Error: 0 Device Not Ready: 0 No Device: 0 Recoverable: 0
Illegal Request: 0 Predictive Failure Analysis: 0
c0t5000C500335BA8C3d0 Soft Errors: 0 Hard Errors: 0 Transport Errors: 0
Vendor: SEAGATE  Product: ST930003SSUN300G Revision: 0B70 Serial No: 110672SDF4
Size: 300.00GB <300000000000 bytes>
Media Error: 0 Device Not Ready: 0 No Device: 0 Recoverable: 0
Illegal Request: 0 Predictive Failure Analysis: 0
c2t0d0           Soft Errors: 0 Hard Errors: 12 Transport Errors: 0
Vendor: AMI      Product: Virtual CDROM    Revision: 1.00 Serial No:
Size: 0.00GB <0 bytes>
Media Error: 0 Device Not Ready: 12 No Device: 0 Recoverable: 0
Illegal Request: 2 Predictive Failure Analysis: 0
```

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系统日志消息。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：

- 第 251 页中的 “确定 ZFS 存储池中是否存在问题”
- 第 251 页中的 “查看 zpool status 输出”
- 第 249 页中的 “ZFS 错误消息的系统报告”

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。

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
```

如果没有 -x 标志，则该命令显示所有池（如果在命令行上指定了池，则为请求的池）的完整状态，即使池的运行状况良好也是如此。

有关 zpool status 命令的命令行选项的更多信息，请参见第 73 页中的 “查询 ZFS 存储池的状态”。

查看 zpool status 输出

完整的 zpool status 输出与以下内容类似：

```
# zpool status tank
   pool: tank
 state: DEGRADED
status: One or more devices could not be opened.  Sufficient replicas exist for
        the pool to continue functioning in a degraded state.
action: Attach the missing device and online it using 'zpool online'.
   see: http://www.sun.com/msg/ZFS-8000-2Q
 scan: scrub repaired 0 in 0h3m with 0 errors on Mon Nov 12 15:17:02 2012
config:

        NAME        STATE     READ WRITE CKSUM
        tank        DEGRADED     0     0     0
          mirror-0  DEGRADED     0     0     0
            c1t1d0  ONLINE       0     0     0
            c1t2d0  UNAVAIL      0     0     0  cannot open

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（其中一些字段仅针对出现问题的池显示）：

pool　　确定池的名称。

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，请参见第 256 页中的 “确定设备故障的类型”。

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，请参见第 264 页中的 “查看重新同步状态”。

ZFS 存储池清理状态

`zpool status` 输出的清理部分描述任何显式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。

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，请参见第 266 页中的 “检查 ZFS 文件系统完整性”。

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
  pool: tank
 state: UNAVAIL
status: One or more devices are faulted in response to IO failures.
action: Make sure the affected devices are connected, then run 'zpool clear'.
   see: http://www.sun.com/msg/ZFS-8000-HC
 scrub: scrub completed after 0h0m with 0 errors on Tue Feb  2 13:08:42 2010
config:

        NAME        STATE     READ WRITE CKSUM
        tank        UNAVAIL      0     0     0  insufficient replicas
          c1t0d0    ONLINE       0     0     0
          c1t1d0    UNAVAIL      4     1     0  cannot open

errors: Permanent errors have been detected in the following files:

/tank/data/aaa
/tank/data/bbb
/tank/data/ccc
```

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，请参见第 270 页中的 “确定数据损坏的类型”。

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，请参见第 258 页中的“替换 ZFS 存储池中的设备”。

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**－第 137 页中的“从镜像 ZFS 根池中的备用磁盘引导”
- **替换根池中的磁盘**
  - 第 143 页中的“如何替换 ZFS 根池中的磁盘”
- **完整的根池灾难恢复**－第 143 页中的“恢复 ZFS 根池或根池快照”

例如，设备出现故障后，可能会在 fmd 的输出中看到类似于以下内容的消息：

```
SUNW-MSG-ID: ZFS-8000-FD, TYPE: Fault, VER: 1, SEVERITY: Major
EVENT-TIME: Thu Jun 24 10:42:36 PDT 2010
PLATFORM: SUNW,Sun-Fire-T200, CSN: -, HOSTNAME: daleks
SOURCE: zfs-diagnosis, REV: 1.0
EVENT-ID: a1fb66d0-cc51-cd14-a835-961c15696fcb
DESC: The number of I/O errors associated with a ZFS device exceeded
acceptable levels.  Refer to http://sun.com/msg/ZFS-8000-FD for more information.
AUTO-RESPONSE: The device has been offlined and marked as faulted.  An attempt
will be made to activate a hot spare if available.
IMPACT: Fault tolerance of the pool may be compromised.
REC-ACTION: Run 'zpool status -x' and replace the bad device.
```

要查看有关设备问题和解决办法的更详细信息，请使用 zpool status -x 命令。例如：

```
# zpool status -x
  pool: tank
 state: DEGRADED
status: One or more devices could not be opened.  Sufficient replicas exist for
        the pool to continue functioning in a degraded state.
action: Attach the missing device and online it using 'zpool online'.
   see: http://www.sun.com/msg/ZFS-8000-2Q
  scan: scrub repaired 0 in 0h0m with 0 errors on Tue Sep 27 16:59:07 2011
config:

        NAME        STATE     READ WRITE CKSUM
        tank        DEGRADED     0     0     0
          mirror-0  DEGRADED     0     0     0
            c2t2d0  ONLINE       0     0     0
            c2t1d0  UNAVAIL      0     0     0  cannot open

errors: No known data errors
```

从此输出中可以看到，设备 c2t1d0 未正常运行。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 tank c2t1d0
```

如果 fmadm faulty 的输出标识出了该设备错误，请让 FMA 知道设备已被替换。例如：

```
# fmadm faulty
--------------- ------------------------------------  -------------- ---------
TIME            EVENT-ID                              MSG-ID         SEVERITY
--------------- ------------------------------------  -------------- ---------
Sep 27 16:58:50 e6bb52c3-5fe0-41a1-9ccc-c2f8a6b56100  ZFS-8000-D3    Major

Host        : neo
Platform    : SUNW,Sun-Fire-T200        Chassis_id  :
Product_sn  :

Fault class : fault.fs.zfs.device
Affects     : zfs://pool=tank/vdev=c75a8336cda03110
                  faulted and taken out of service
Problem in  : zfs://pool=tank/vdev=c75a8336cda03110
                  faulted and taken out of service

Description : A ZFS device failed.  Refer to http://sun.com/msg/ZFS-8000-D3 for
              more information.

Response    : No automated response will occur.

Impact      : Fault tolerance of the pool may be compromised.

Action      : Run 'zpool status -x' and replace the bad device.

# fmadm repaired zfs://pool=tank/vdev=c75a8336cda03110
```

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。在以下情况下，池可能会变为 UNAVAIL 状态，即无法访问数据，除非重新附加设备：

- 镜像的所有组件都被删除
- RAID-Z (raidz1) 设备中有一个以上设备被删除
- 单磁盘配置中移除了顶层设备

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。按照第 256 页中的 “更换或修复损坏的设备”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'.
    see: http://www.sun.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，请参见第 62 页中的 “使设备联机”。

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://www.sun.com/msg/ZFS-8000-8A
  scan: scrub in progress since Tue Sep 27 17:12:40 2011
    63.9M scanned out of 528M at 10.7M/s, 0h0m to go
    0 repaired, 12.11% done
config:

        NAME        STATE     READ WRITE CKSUM
        tank        ONLINE       2     0     0
          mirror-0  ONLINE       2     0     0
            c2t2d0  ONLINE       2     0     0
            c2t1d0  ONLINE       2     0     0

errors: Permanent errors have been detected in the following files:

        /tank/words
```

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，请参见第 269 页中的 “修复损坏的数据”。

清除瞬态设备错误

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，请参见第 63 页中的 “清除存储池设备错误”。

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，除非存在包含正确数据的足够副本，否则会将设备上的任何损坏数据复制到新设备。

在以下配置中，可以替换磁盘 c1t1d0，而且将从完好的副本 c1t0d0 复制池中的任何数据：

```
    mirror            DEGRADED
    c1t0d0             ONLINE
    c1t1d0             FAULTED
```

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。

```
    raidz             FAULTED
    c1t0d0            ONLINE
    c2t0d0            FAULTED
    c3t0d0            FAULTED
    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，请参见第 272 页中的 “修复 ZFS 存储池范围内的损坏”。

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，请参见第 64 页中的 “替换存储池中的设备”。

示例 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，则请参见第 64 页中的 “替换存储池中的设备”。

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）之前，请确保蓝色的 Ready to Remove（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的同一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
  ```

示例 10–1 替换 ZFS 存储池中的 SATA 磁盘 （续）

- 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。

如果您打算将一个磁盘 (`c1t3d0`) 替换为另一个磁盘 (`c4t3d0`)，则只需运行 `zpool replace` 命令。例如：

示例 10–1 替换 ZFS 存储池中的 SATA 磁盘 （续）

```
# 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
```

示例 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：

- 查看 `zpool status -x` 输出和 FMA 诊断消息。有关说明，请参见以下页面：

示例 10–2 更换出现故障的日志设备 （续）

```
   https://support.oracle.com/
   CSP/main/
   article?cmd=show&type=NOT&doctype=REFERENCE&alias=EVENT:ZFS-8000-K4
```

- 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。

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

        NAME        STATE     READ WRITE CKSUM
        pool        FAULTED      0     0     0 bad intent log
          mirror    ONLINE       0     0     0
            c0t1d0  ONLINE       0     0     0
            c0t4d0  ONLINE       0     0     0
        logs        FAULTED      0     0     0 bad intent log
          c0t5d0    UNAVAIL      0     0     0 cannot open
<Physically replace the failed log device>
# zpool online pool c0t5d0
# zpool clear pool
```

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>
```

示例 10–2 更换出现故障的日志设备 （续）

```
# 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。

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。
- 重新同步是可中断的和安全的。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: DEGRADED
status: One or more devices is currently being resilvered.  The pool will
        continue to function, possibly in a degraded state.
action: Wait for the resilver to complete.
 scrub: resilver in progress for 0h0m, 22.60% done, 0h1m to go
config:
```

```
        NAME                 STATE     READ WRITE CKSUM
        tank              DEGRADED     0     0     0
          mirror-0        DEGRADED     0     0     0
            replacing-0   DEGRADED     0     0     0
              c1t0d0      UNAVAIL      0     0     0  cannot open
              c2t0d0      ONLINE       0     0     0  85.0M resilvered
            c1t1d0        ONLINE       0     0     0

errors: No known data errors
```

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。

解决 ZFS 文件系统问题

解决 ZFS 存储池中的数据问题

数据问题的示例包括以下各项：

- 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。

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。ZFS 配置中的磁盘上存在不一致数据的唯一原因是出现硬件故障（在这种情况下，应该已创建冗余池）或 ZFS 软件中存在错误。

`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，请参见第 272 页中的 “修复 ZFS 存储池范围内的损坏”。

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**，在内存和其他系统中经常将它用作一种在错误导致硬件或软件故障之前检测和防止错误的方法。

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的方法是，对池中所有数据启动显式清理操作。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
 scrub: scrub completed after 0h7m with 0 errors on Tue Tue Feb  2 12:54:00 2010
config:
        NAME        STATE     READ WRITE CKSUM
        tank        ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c1t0d0  ONLINE       0     0     0
            c1t1d0  ONLINE       0     0     0

errors: No known data errors
```

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。如果系统通常一直执行 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。

有关解释 zpool status 输出的更多信息，请参见第 73 页中的 “查询 ZFS 存储池的状态”。

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，请参见第 264 页中的 “查看重新同步状态”。

ZFS 数据损坏

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。

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。另请查看第 28 页中的 “ZFS 磁盘空间记帐”。

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（如果有）。

- **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 池空间之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
```

修复损坏的数据

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）。

- 第 270 页中的 “确定数据损坏的类型”
- 第 271 页中的 “修复损坏的文件或目录”
- 第 272 页中的 “修复 ZFS 存储池范围内的损坏”

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，请参见第 266 页中的 “检查 ZFS 文件系统完整性”。

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 monkey
  pool: monkey
 state: ONLINE
status: One or more devices has experienced an error resulting in data
        corruption.  Applications may be affected.
action: Restore the file in question if possible.  Otherwise restore the
        entire pool from backup.
   see: http://www.sun.com/msg/ZFS-8000-8A
 scrub: scrub completed after 0h0m with 8 errors on Tue Jul 13 13:17:32 2010
config:

        NAME        STATE     READ WRITE CKSUM
        monkey      ONLINE       8     0     0
          c1t1d0    ONLINE       2     0     0
          c2t5d0    ONLINE       6     0     0

errors: 8 data errors, use '-v' for a list
```

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: 1422736890544688191
 state: FAULTED
status: The pool metadata is corrupted.
```

```
action: The pool cannot be imported due to damaged devices or data.
   see: http://www.sun.com/msg/ZFS-8000-72
config:

        morpheus    FAULTED   corrupted data
          c1t10d0   ONLINE
```

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 -v
  pool: monkey
 state: ONLINE
status: One or more devices has experienced an error resulting in data
        corruption.  Applications may be affected.
action: Restore the file in question if possible.  Otherwise restore the
        entire pool from backup.
   see: http://www.sun.com/msg/ZFS-8000-8A
 scrub: scrub completed after 0h0m with 8 errors on Tue Jul 13 13:17:32 2010
config:

        NAME        STATE     READ WRITE CKSUM
        monkey      ONLINE       8     0     0
          c1t1d0    ONLINE       2     0     0
          c2t5d0    ONLINE       6     0     0

errors: Permanent errors have been detected in the following files:

/monkey/a.txt
/monkey/bananas/b.txt
/monkey/sub/dir/d.txt
monkey/ghost/e.txt
/monkey/ghost/boo/f.txt
```

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) 中的对象已损坏，则会显示特殊标签 <metadata>，后跟该对象的编号。

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额外的损坏数据路径：

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: FAULTED
  status: The pool metadata is corrupted and the pool cannot be opened.
  action: Recovery is possible, but will result in some data loss.
          Returning the pool to its state as of Wed Jul 14 11:44:10 2010
          should correct the problem.  Approximately 5 seconds of data
          must be discarded, irreversibly.  Recovery can be attempted
          by executing 'zpool clear -F tpool'.  A scrub of the pool
          is strongly recommended after recovery.
     see: http://www.sun.com/msg/ZFS-8000-72
   scrub: none requested
  config:

          NAME        STATE     READ WRITE CKSUM
           tpool      FAULTED      0     0     1  corrupted data
            c1t1d0    ONLINE       0     0     2
            c1t3d0    ONLINE       0     0     4
  ```

  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 Wed Jul 14 11:44:10 2010
        should correct the problem.  Approximately 5 seconds of data
        must be discarded, irreversibly.  Recovery can be attempted
        by executing 'zpool import -F tpool'.  A scrub of the pool
        is strongly recommended after recovery.
```

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 Wed Jul 14 11:44:10 2010.
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。如果池不在 zpool.cache 文件中，将无法成功导入或打开它，当您尝试导入该池时，会看到池受损的消息。

- 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，请参见第 88 页中的 “在只读模式下导入池”。
- 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，请参见第 87 页中的 “导入缺少日志设备的池”。
- 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，将描述数据集的布局和在本地设置的各种属性的文件保存在某个安全的位置（因为在使池无法访问后此信息将变得无法访问）。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，都可以通过先导出池（如果它确实是可见的）再重新导入它来恢复配置。

有关导入和导出池的信息，请参见第 83 页中的 “迁移 ZFS 存储池”。

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。为了从此情况恢复，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 =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：

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 发行版和修补程序使系统保持最新
- 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。

  另请参见第 278 页中的 “在本地或网络连接存储阵列上创建池的做法”。
- 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。

根池创建做法

- 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 c0t1d0s0
  ```

  或者，创建一个镜像根池。例如：

  ```
  # zpool create rpool mirror c0t1d0s0 c0t2d0s0
  ```

  - 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 (USB stick) 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。

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，请考虑是否要将 SAN 设备用作根池，这些设备在系统引导后可能不可用，从而可能使根池的设备也变为 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

有关更多信息，请参见以下白皮书：

http://blogs.oracle.com/storage/entry/new_white_paper_configuring_oracle

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> 是虚拟机的名称
  - <type> 是控制器类型，为 piix3ide（如果使用常用的 IDE 虚拟控制器）或 ahci（如果使用 SATA 控制器）
  - <n> 是磁盘编号

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 80% 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已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 80% 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 80%。
- 监视池运行状况
  - 对于冗余池，使用 `zpool status` 和 `fmdump` 监视池，每周监视一次
  - 对于非冗余池，使用 `zpool status` 和 `fmdump` 监视池，每周监视两次
- 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，使用 `zpool status` 和 `fmdump` 监视池运行状况，每周一次
  - 对于非冗余池，使用 `zpool status` 和 `fmdump` 监视池运行状况，每两周一次
- 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。

- 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。

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。
- 在归档池中创建归档日志文件系统，启用压缩和缺省的 recordsize 值，并将 primarycache 设置为 metadata。

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，而不要使用 du 和 df 命令，因为传统命令不会计入子孙文件系统或快照所占用的空间。

有关更多信息，请参见第 268 页中的 “解决 ZFS 空间问题”。

- 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，请参见第 182 页中的 “升级 ZFS 文件系统”和第 91 页中的 “升级 ZFS 存储池”。

ZFS 池版本

下表列出了 Oracle Solaris 发行版提供的 ZFS 池版本。

版本	Solaris 10	说明
1	Solaris 10 6/06	初始 ZFS 版本
2	Solaris 10 11/06	同样的块（复制的元数据）
3	Solaris 10 11/06	热备件和双奇偶校验 RAID-Z
4	Solaris 10 8/07	`zpool` 历史记录

版本	Solaris 10	说明
5	Solaris 10 10/08	`gzip` 压缩算法
6	Solaris 10 10/08	`bootfs` 池属性
7	Solaris 10 10/08	不同用途的日志设备
8	Solaris 10 10/08	委托管理
9	Solaris 10 10/08	`refquota` 和 `refreservation` 属性
10	Solaris 10 5/09	缓存设备
11	Solaris 10 10/09	改进了清理性能
12	Solaris 10 10/09	快照属性
13	Solaris 10 10/09	`snapused` 属性
14	Solaris 10 10/09	`aclinherit passthrough-x` 属性
15	Solaris 10 10/09	用户和组空间记帐
16	Solaris 10 9/10	`stmf` 属性支持
17	Solaris 10 9/10	三奇偶校验 RAID-Z
18	Solaris 10 9/10	快照用户存放
19	Solaris 10 9/10	日志设备移除
20	Solaris 10 9/10	**使用** `zle`（zero-length encoding，零长度编码）压缩
21	Solaris 10 9/10	保留
22	Solaris 10 9/10	已接收属性
23	Solaris 10 8/11	精简 ZIL
24	Solaris 10 8/11	系统属性
25	Solaris 10 8/11	改进了清理统计信息
26	Solaris 10 8/11	提高了删除快照的性能
27	Solaris 10 8/11	提高了创建快照的性能
28	Solaris 10 8/11	多个 vdev 替换
29	Solaris 10 8/11	RAID-Z/镜像混合分配器
30	Solaris 10 1/13	保留
31	Solaris 10 1/13	改进了 `zfs list` 性能
32	Solaris 10 1/13	1 MB 的块大小

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	Solaris 10	说明
1	Solaris 10 6/06	初始 ZFS 文件系统版本
2	Solaris 10 10/08	增强了目录项
3	Solaris 10 10/08	大小写无关性和文件系统唯一标识符 (FUID)
4	Solaris 10 10/09	`userquota` 和 `groupquota` 属性
5	Solaris 10 8/11	系统属性

索引