Solaris System Administration Series

Solaris™ 10 ZFS Essentials

Scott Watanabe

Library of Congress Cataloging-in-Publication Data

Watanabe, Scott.
Solaris 10 ZFS essentials / Scott Watanabe.
p. cm.
Includes index.
ISBN 978-0-13-700010-4 (pbk. : alk. paper)
1. Solaris (Computer file) 2. File organization (Computer science) 3. Database management.
I. Title.
QA76.9.F5W38 2009
005.74—dc22

2009044423

ISBN-13: 978-0-13-700010-4
ISBN-10: 0-13-700010-3
Text printed in the United States on recycled paper at Courier in Stoughton, Massachusetts.
First printing, December 2009

Preface

Solaris™ 10 ZFS Essentials is part of a new series of books on Solaris system administration. It presents the revolutionary ZFS file system that was introduced in the Solaris 10 release. ZFS is a file system that is elegant in its simplicity and in the ease with which it allows system administrators to manage data and storage. Other books in the series are *Solaris™ 10 System Administration Essentials* and *Solaris™ 10 Security Essentials*. The former covers all the breakthrough features of the Solaris 10 operating system in one place, and the latter covers all the features of the Solaris 10 operating system that make it the best choice for meeting the present-day challenges to robust and secure computing.

ZFS offers a dramatic advance in data management with an innovative approach to data integrity, near-zero administration, and a welcome integration of file system and volume management capabilities. The centerpiece of this new architecture is the concept of a virtual storage pool, which decouples the file system from physical storage in the same way that virtual memory abstracts the address space from physical memory, allowing for much more efficient use of storage devices.

In ZFS, storage space is shared dynamically between multiple file systems from a single storage pool and is parceled out of the pool as file systems request it. Physical storage can be added to storage pools dynamically without interrupting services, which provides new levels of flexibility, availability, and performance. And in terms of scalability, ZFS is a 128-bit file system. Its theoretical limits are truly mind-boggling—2,128 bytes of storage and 264 for everything else, including file systems, snapshots, directory entries, devices, and more. And ZFS implements an improvement on RAID-5 by introducing RAID-Z, which uses parity, striping, and atomic operations to ensure reconstruction of corrupted data. It is ideally suited for managing industry-standard storage servers.

Books in the Solaris System Administration Series

The following sections talk more about the other two books in the series.

Solaris™ 10 System Administration Essentials

Solaris™ 10 System Administration Essentials covers all the breakthrough features of the Solaris 10 operating system in one place. It does so in a straightforward way that makes an enterprise-level operating system accessible to system administrators at all levels.

The book provides a comprehensive overview along with hands-on examples of the key features that have made Solaris the leading UNIX operating system for years, and it covers the significant new features of Solaris 10 that put it far ahead of its competitors, such as Zones, the ZFS file system, Fault Management Architecture, Service Management Facility, and DTrace (the dynamic tracing tool for troubleshooting OS and application problems on production systems in real time).

Solaris™ 10 Security Essentials

Solaris™ 10 Security Essentials covers all the security features and technologies in the Solaris 10 operating system that make it the OS of choice for IT environments that require optimum security.

The book explains the strengths of Solaris operating system security—its scalability and adaptability—in a simple, straightforward way. It explains how security features in Solaris can protect a single-user system with login authentication as well as how those features can protect Internet and intranet configurations.

Intended Audience

The three books in the Solaris System Administration Series can benefit anyone who wants to learn more about the Solaris 10 operating system. They are written to be particularly accessible to system administrators who are new to Solaris, as well as to people who are perhaps already serving as administrators in companies running Linux, Windows, and other UNIX systems.

If you are not currently a practicing system administrator but want to become one, this series, starting with *Solaris™ 10 System Administration Essentials*, provides an excellent introduction. In fact, most of the examples used in the books are suited to or can be adapted to small learning environments such as a home setup. So, even before you venture into corporate system administration or deploy Solaris 10 in your existing IT installation, these books will help you experiment in a small test environment.

OpenSolaris

In June 2005, Sun Microsystems introduced OpenSolaris, a fully functional Solaris operating system release built from open source. Although the books in this series focus on Solaris 10, they often incorporate aspects of OpenSolaris. Now that Solaris has been open-sourced, its evolution has accelerated even beyond its normally rapid pace. The authors of this series have often found it interesting to introduce features or nuances that are new in OpenSolaris. At the same, many of the enhancements introduced into OpenSolaris are finding their way into Solaris 10. So, whether you are learning Solaris 10 or already have an eye on OpenSolaris, the books in this series are for you.

2

Managing Storage Pools

ZFS storage pools are the basis of the ZFS system. In this chapter, I cover the basic concepts, configurations, and administrative tasks of ZFS pools. In Chapter 5, I cover advanced configuration topics in ZFS pool administration.

The `zpool` *command manages ZFS storage pools. The* `zpool` *command creates, modifies, and destroys ZFS pools.*

Redundant configurations supported in ZFS are mirror (RAID-1), RAID-Z (similar to RAID-5), and RAID-Z2 with a double parity (similar to RAID-6). All traditional RAID-5-like algorithms (RAID-4, RAID-6, RDP, and EVEN-ODD, for example) suffer from a problem known as the ***RAID-5 write hole****. If only part of a RAID-5 stripe is written and power is lost before all blocks have made it to disk, the parity will remain out of sync with the data (and is therefore useless) forever, unless a subsequent full-stripe write overwrites it. In RAID-Z, ZFS uses variable-width RAID stripes so that all writes are full-stripe writes. This design is possible only because ZFS integrates file system and device management in such a way that the file system's metadata has enough information about the underlying data redundancy model to handle variable-width RAID stripes. RAID-Z is the world's first software-only solution to the RAID-5 write hole.*

2.1 ZFS Pool Concepts

ZFS pools are comprised of virtual devices. ZFS abstracts every physical device into a virtual device. A vdev can be a disk, a slice of a disk, a file, or even a logical volume presented by another volume manager such as Solaris Volume Manager (SVM) or a LUN from a hardware RAID device.

These are the virtual devices types:

- **Dynamic stripe**: A dynamic stripe is a nonredundant configuration of a simple disk or concatenation of disks.
- **Redundant group (mirror, RAID-Z1, or RAID-Z2)**: A mirror can be a two-way or three-way mirror. RAID-Z groups are recommended to have up to nine disks in the group. If there are more, then multiple vdevs are recommended. Two disks minimum are needed for RAID-Z, and three disks at a minimum are needed for RAID-Z2. (Note that *RAID-Z* and *RAID-Z1* are interchangeable terms. With the introduction of the RAID-Z2 feature, the term *RAID-Z* evolved into *RAID-Z1* to differentiate it from RAID-Z2. I use both terms in this book.)
- **Spare**: A spare vdev is for a hot standby disk replacement.
- **Log**: A log vdev is for ZFS Intent Log (ZIL). The ZIL increases the write performance on ZFS. Only dynamic stripe and mirrored vdev configurations are supported for this vdev type.
- **Cache**: A cache vdev is used to speed up random reads from a RAID-Z-configured pool. Its intended use is for read-heavy workloads. There is no redundancy support at this point for this vdev type. If there is a read error, then ZFS will read from the original storage pool.

Log and cache vdevs are used with solid-state disks (SSDs) in the Sun Storage 7000 series in hybrid storage pools (HSPs; see `http://blogs.sun.com/ahl/entry/fishworks_launch`).

Best-practice guidelines for ZFS pools include the following:

- Mirrored configuration beyond a three-way mirror should not be used. Think about using a RAID-Z configuration instead.
- Use RAID-Z or RAID-Z2 virtual device groups with fewer than ten disks in each vdev.
- Using whole disks is best. ZFS works best with “just a bunch of disks” (JBOD).
- Use slices for vdev groups only for boot disks.
- Use disks of a terabyte or less for boot devices.
- Use matched-capacity disks (mixed geometry is OK) for the best maximum storage results.
- Use matching sizes of vdevs in a ZFS pool. Match the number of disks and redundancy groups in each vdev in a pool for best performance.

Creating/adding new vdevs to a ZFS pool is the most unforgiving part about ZFS. Once committed, some operations cannot be undone. The `zpool` command will warn you, however, if the operation is not what's expected. There is a `force` option in `zpool` to bypass any of the warnings, but it is not recommended that you use the `force` option unless you are sure you will not need to reverse the operation.

These are the rules for ZFS pools:

- Once a normal (dynamic stripe) vdev is added to a ZFS pool, it cannot be removed.
- Only the special-use vdevs can be removed: spares, log, and cache.
- Disks the same size or larger can be replaced within a vdev.
- Disks can be added to a single disk or mirrored vdev to form a mirror or a three-way mirror.
- New disks cannot be added to an existing RAID-Z or RAID-Z2 vdev configuration.

2.2 Creating a Dynamic Stripe

A dynamic stripe is the most basic pool that can be created. There is no redundancy in this configuration. If any disk fails, then the whole pool is lost. Any pool created with multiple vdevs will dynamically stripe across each vdev or physical device.

You can use the `zpool` command with the subcommand to create a dynamic stripe. After the `create` subcommand is the name of the new ZFS pool, dstripe, and the disks that will comprise the pool.

```
# zpool create dstripe c5t0d0 c5t1d0
```

The following listing presents the results of creating the ZFS pool dstripe. On line 2, `zpool  list` is executed to list all the ZFS pools on the system. Line 3 starts a list of the available pools. The command gives you general information about the ZFS pools.

```
1   # zpool create dstripe c5t0d0 c5t1d0
2   # zpool list
3   NAME       SIZE   USED  AVAIL    CAP  HEALTH  ALTROOT
4   dstripe    234M    75K   234M     0%  ONLINE  -
5   rpool     15.9G  3.21G  12.7G    20%  ONLINE  -
```

Next on line 6, `zpool  status` is issued to inquire about the status of the ZFS pools. Starting at line 7, the status of the ZFS pool dstripe is displayed, with a normal status. Reading the `config:` section of the output starting at line 10, the pool dstripe is shown as two concatenated disks. Lines 14 and 15 list the vdevs (c5t0d0 and c5t1d0) that belong to the pool dstripe. The second pool listed is made of a single disk called *rpool,* configured as a dynamic stripe with only a single vdev (c3d0s0). It was created as part of the OS installation process.

```
6   # zpool status
7   pool: dstripe
8   state: ONLINE
9   scrub: none requested
10  config:
11
12          NAME        STATE     READ WRITE CKSUM
13          dstripe     ONLINE       0     0     0
14            c5t0d0    ONLINE       0     0     0
15            c5t1d0    ONLINE       0     0     0
16
17  errors: No known data errors
18
19    pool: rpool
20   state: ONLINE
21   scrub: none requested
22  config:
23
24          NAME        STATE     READ WRITE CKSUM
25          rpool       ONLINE       0     0     0
26            c3d0s0    ONLINE       0     0     0
27
28 errors: No known data errors
```

Figure 2.1 illustrates the resulting dynamic pool with its two vdevs of single nonredundant disks. Any problems with the disks (sector errors or disk failure) may result in the loss of the whole pool or data.

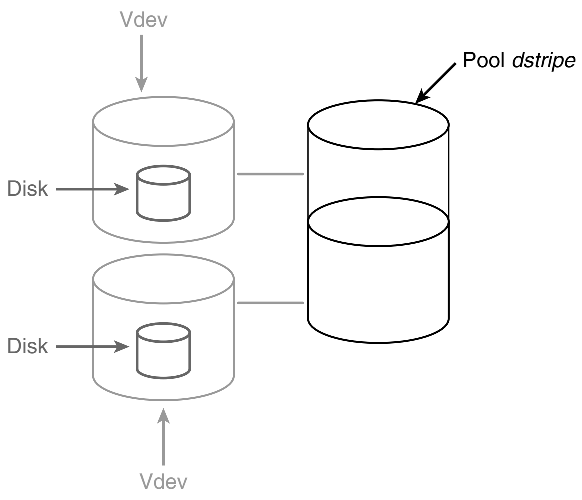

Figure 2.1 Dynamic stripe pool with two vdevs

2.3 Creating a Pool with Mirrored Devices

The following command creates a ZFS mirrored pool called *mpool* with a mirrored vdev. As expected, the new pool mpool is about half the capacity of dstripe. The pool dstripe is a concatenation of two disks of the same capacity, and mpool is a mirror of a disk of the same capacity.

The following command line shows how to use the `zpool` command with the subcommand `create` to create a pool with mirrored vdevs. After the `create` subcommand is the name of the new ZFS pool, mpool. The `mirror` subcommand will create a mirrored vdev with the disks c5t2d0 and c5t3d0.

```
# zpool create mpool mirror c5t2d0 c5t3d0
```

The following output is the creation of a mirrored ZFS pool called *mpool*. Using the `zpool list` command, you now can see the capacity of the newly created pool on line 5. Notice it is half the capacity of the dstripe pool.

```
1   # zpool create mpool mirror c5t2d0 c5t3d0
2   # zpool list
3   NAME       SIZE   USED  AVAIL    CAP  HEALTH  ALTROOT
4   dstripe    234M    75K   234M     0%  ONLINE  -
5   mpool      117M  73.5K   117M     0%  ONLINE  -
6   rpool     15.9G  3.21G  12.7G    20%  ONLINE  -
```

Starting at line 20 (in the following part of the listing) is the status information of mpool. The disks c5t2d0 and c5t3d0 are configured as a mirror vdev, and the mirror is part of *mpool*. This is an important concept in reading the status information of ZFS pools. Notice the indentations on the pool name notations. The first is the name of the ZFS pool. Then at the first indentation of the name are the vdevs that are part of the pool. In the case of a dynamic stripe, this is the physical device. If the pool is created with redundant vdev(s), the first indentation will be `mirror`, `raidz1`, or `raidz2`. Then the next indentation will be the physical devices that comprise the redundant vdev. On lines 13 and 25 are the names dstripe and mpool, respectively. On lines 14 and 15 are the disks that belong to dstripe on the first indentation. On line 25 is the mirror configuration, and the next indented line lists the disks belonging to the mirror configuration. Compare Figures 2.1 and 2.2 for a graphical representation of each pool's configuration.

```
7   # zpool status
8   pool: dstripe
9   state: ONLINE
10  scrub: none requested
11  config:
```

continues

```
12
13        NAME        STATE     READ WRITE CKSUM
14        dstripe     ONLINE       0     0     0
15          c5t0d0    ONLINE       0     0     0
16          c5t1d0    ONLINE       0     0     0
17
18  errors: No known data errors
19
20    pool: mpool
21   state: ONLINE
22   scrub: none requested
23  config:
24          NAME        STATE     READ WRITE CKSUM
25          mpool       ONLINE       0     0     0
26            mirror    ONLINE       0     0     0
27              c5t2d0  ONLINE       0     0     0
28              c5t3d0  ONLINE       0     0     0
29
30  errors: No known data errors
31
32    pool: rpool
33   state: ONLINE
34   scrub: none requested
35  config:
36
37          NAME        STATE     READ WRITE CKSUM
38          rpool       ONLINE       0     0     0
39            c3d0s0    ONLINE       0     0     0
40  errors: No known data errors
```

Figure 2.2 illustrates the results of creating the ZFS pool mpool with one mirrored vdev.

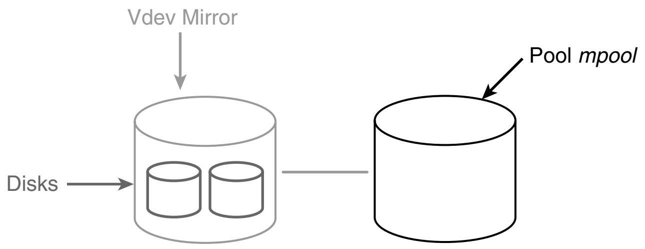

Figure 2.2 Pool *mpool* with a mirrored vdev

The following sequence adds a second mirror vdev to the pool, doubling the size of pool mpool. Notice that after the addition of the second mirror vdev, the dstripe and mpool pools are the same capacity. Line 30 is the mirrored vdev with the physical disks c5t4d0 and c5t5d0 indented.

```
1.  # zpool add mpool mirror c5t4d0 c5t5d0
2.  # zpool list
3.  NAME       SIZE   USED  AVAIL    CAP  HEALTH  ALTROOT
4.  dstripe    234M    75K   234M     0%  ONLINE  -
```

continues

```
mpool     234M    78K   234M     0%  ONLINE  -
rpool    15.9G  3.21G  12.7G    20%  ONLINE  -
# zpool status
     pool: dstripe
    state: ONLINE
 scrub: none requested
config:

NAME        STATE     READ WRITE CKSUM
dstripe     ONLINE       0     0     0
  c5t0d0    ONLINE       0     0     0
  c5t1d0    ONLINE       0     0     0

errors: No known data errors

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

NAME        STATE     READ WRITE CKSUM
mpool       ONLINE       0     0     0
  mirror    ONLINE       0     0     0
    c5t2d0  ONLINE       0     0     0
    c5t3d0  ONLINE       0     0     0
  mirror    ONLINE       0     0     0
    c5t4d0  ONLINE       0     0     0
    c5t5d0  ONLINE       0     0     0

errors: No known data errors

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

NAME        STATE     READ WRITE CKSUM
rpool       ONLINE       0     0     0
  c3d0s0    ONLINE       0     0     0

errors: No known data errors
```

Figure 2.3 shows the results of adding a second mirrored vdev to ZFS pool *mpool*.

2.4 Creating a Pool with RAID-Z Devices

In ZFS you can also create redundant vdevs similar to RAID-5, called *RAID-Z*. To create a pool with double parity, you would use RAID-Z2 instead.

The following command line will create a ZFS pool named *rzpool* with two RAID-Z1 vdevs, each with four disks:

```
# zpool create rzpool raidz1 c5t6d0 c5t7d0 c5t8d0 c5t9d0 raidz1
c5t10d0 c5t11d0 c5t12d0 c5t13d0
```

The first RAID-Z1 vdev consists of disks c5t6d0, c5t7d0, c5t8d0, and c5t9d0, and the second RAID-Z1 vdev has c5t6d0, c5t7d0, c5t8d0, and c5t9d0 as members.

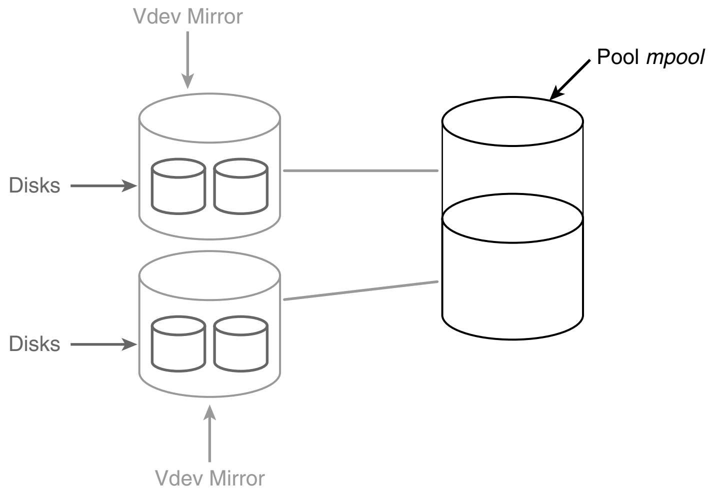

Figure 2.3 Pool *mpool* with the added mirror vdev

The `zpool list` command shows the summary status of the new pool rzpool. The output of `zpool status`, on lines 16 and 21, shows the RAID-Z1 virtual devices and physical disk devices that make up each RAID-Z1 virtual device.

```
1   # zpool create rzpool raidz1 c5t6d0 c5t7d0 c5t8d0 c5t9d0
raidz1 c5t10d0 c5t11d0 c5t12d0 c5t13d0
2   # zpool list
3   NAME      SIZE   USED  AVAIL    CAP  HEALTH  ALTROOT
4   dstripe   234M    75K   234M     0%  ONLINE  -
5   mpool     234M    78K   234M     0%  ONLINE  -
6   rpool    15.9G  3.21G  12.7G    20%  ONLINE  -
7   rzpool    936M   138K   936M     0%  ONLINE  -
8   # zpool status
    …lines deleted…
9   pool: rzpool
10   state: ONLINE
11   scrub: none requested
12  config:
13
14          NAME         STATE     READ WRITE CKSUM
15          rzpool       ONLINE       0     0     0
16            raidz1     ONLINE       0     0     0
17              c5t6d0   ONLINE       0     0     0
18              c5t7d0   ONLINE       0     0     0
19              c5t8d0   ONLINE       0     0     0
20              c5t9d0   ONLINE       0     0     0
21            raidz1     ONLINE       0     0     0
22              c5t10d0  ONLINE       0     0     0
23              c5t11d0  ONLINE       0     0     0
24              c5t12d0  ONLINE       0     0     0
25              c5t13d0  ONLINE       0     0     0
26
27  errors: No known data errors
```

Figure 2.4 has two RAID-Z1 vdevs with four physical disks to each vdev. The vdevs are concatenated to form the ZFS pool rzpool.

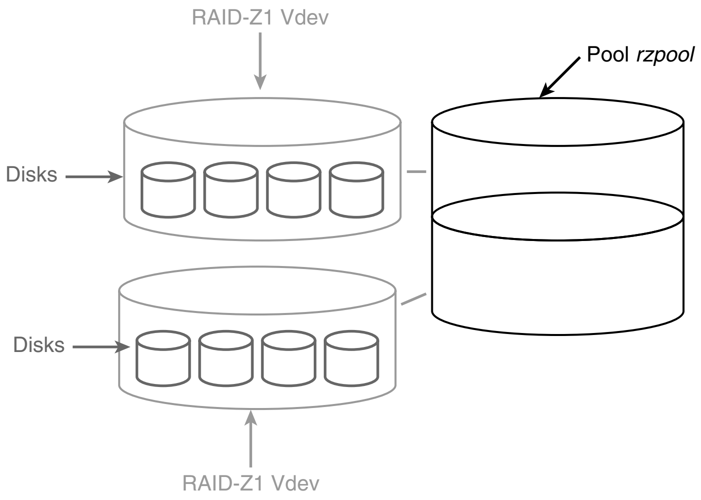

Figure 2.4 Pool *rzpool* using two RAID-Z1 redundant vdevs

2.5 Creating a Spare in a Storage Pool

You can add spare vdevs to a pool to increase its reliability and maintain performance. In case of disk failure, ZFS can replace failed disks with spare disks automatically. This gives the administrator time to make repairs without compromising the reliability and performance of the pool. Spares can be shared among multiple pools. The following example adds a spare disk device to the pool *mpool*. On line 22, the spare vdev is now listed as part of the pool *mpool*.

```
1    # zpool add mpool spare c5t14d0
2    # zpool list
3    NAME       SIZE   USED  AVAIL    CAP  HEALTH  ALTROOT
4    dstripe    234M    75K   234M     0%  ONLINE  -
5    mpool      234M    84K   234M     0%  ONLINE  -
6    rpool     15.9G  3.20G  12.7G    20%  ONLINE  -
7    rzpool     936M   141K   936M     0%  ONLINE  -
8    # zpool status mpool
9      pool: mpool
10    state: ONLINE
11    scrub: none requested
```

continues

```
config:

      NAME        STATE     READ WRITE CKSUM
      mpool       ONLINE       0     0     0
        mirror    ONLINE       0     0     0
          c5t2d0  ONLINE       0     0     0
          c5t3d0  ONLINE       0     0     0
        mirror    ONLINE       0     0     0
          c5t4d0  ONLINE       0     0     0
          c5t5d0  ONLINE       0     0     0
      spares
        c5t14d0   AVAIL

errors: No known data errors
```

In Figure 2.5, an additional vdev is added to the ZFS pool mpool. A spare vdev can be shared with multiple ZFS pools. The spares must have at least the same capacity of the smallest disk in the other vdev devices.

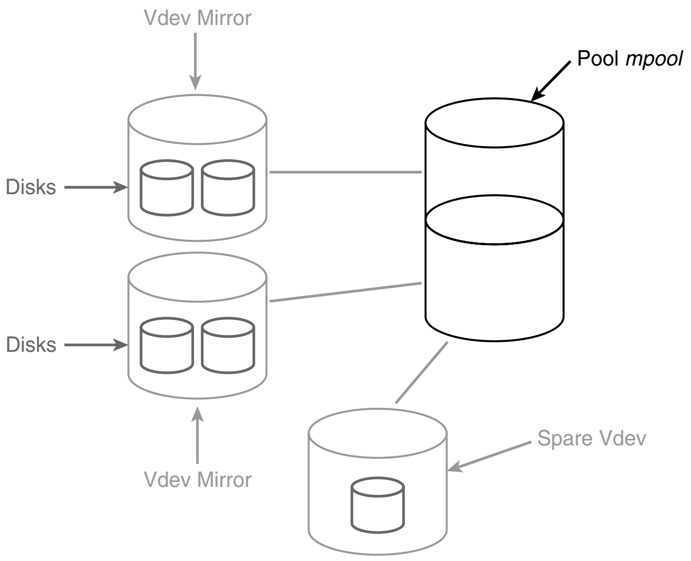

Figure 2.5 Pool *mpool* with a spare vdev added

2.6 Adding a Spare Vdev to a Second Storage Pool

As indicated earlier in the chapter, you can share a spare vdev with other pools on the same system. In the following example, the spare vdev (c5t14d0) disk is shared with ZFS pool *rzpool*. After you add the spare to the *rzpool,* the spare disk now appears in ZFS pools *mpool* and *rzpool,* namely, on lines 16 and 17 and lines 38 and 39.

```
1   # zpool add rzpool spare c5t14d0
2   # zpool status mpool rzpool
3   pool: mpool
4   state: ONLINE
5   scrub: none requested
6   config:
7
8   NAME      STATE     READ WRITE CKSUM
9   mpool     ONLINE       0     0     0
10  mirror    ONLINE       0     0     0
11    c5t2d0  ONLINE       0     0     0
12    c5t3d0  ONLINE       0     0     0
13  mirror    ONLINE       0     0     0
14    c5t4d0  ONLINE       0     0     0
15    c5t5d0  ONLINE       0     0     0
16  spares
17    c5t14d0   AVAIL
18
19  errors: No known data errors
20
21  pool: rzpool
22  state: ONLINE
23  scrub: none requested
24  config:
25
26  NAME         STATE     READ WRITE CKSUM
27  rzpool       ONLINE       0     0     0
28    raidz1     ONLINE       0     0     0
29      c5t6d0   ONLINE       0     0     0
30      c5t7d0   ONLINE       0     0     0
31      c5t8d0   ONLINE       0     0     0
32      c5t9d0   ONLINE       0     0     0
33    raidz1     ONLINE       0     0     0
34      c5t10d0  ONLINE       0     0     0
35      c5t11d0  ONLINE       0     0     0
36      c5t12d0  ONLINE       0     0     0
37      c5t13d0  ONLINE       0     0     0
38    spares
39      c5t14d0    AVAIL
40
41  errors: No known data errors
```

In Figure 2.6, the spare vdev is shared between the *mpool* and *rzpool* ZFS pools. Each pool can use the spare when needed.

2.7 Replacing Bad Devices Automatically

ZFS has the capability to replace a disk in a pool automatically without intervention by the administrator. This feature, known as **autoreplace**, is turned off by default. This feature will allow ZFS to replace a bad disk with a spare from the spares pool, automatically allowing the pool to operate at optimum performance.

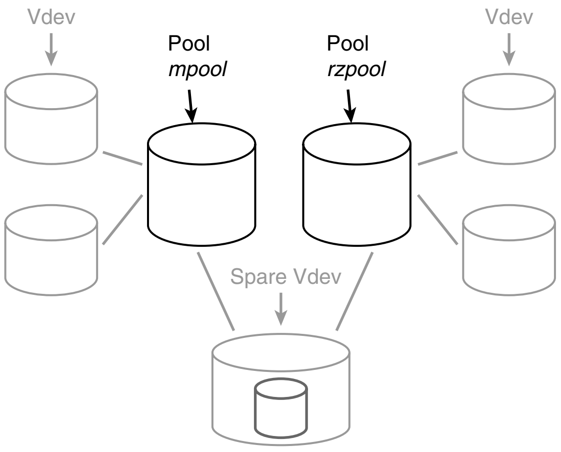

Figure 2.6 Pools *mpool* and *rzpool* sharing a spare vdev

This allows the administrator to replace the failed drive at a later time. The manual disk replacement procedure is covered later in this chapter.

If you list the properties of the ZFS pool, you can see that the autoreplace feature is turned off using the `get` subcommand.

```
# zpool get all mpool
NAME   PROPERTY       VALUE       SOURCE
mpool  size           234M        -
mpool  used           111K        -
mpool  available      234M        -
mpool  capacity       0%          -
mpool  altroot        -           default
mpool  health         DEGRADED    -
mpool  guid           11612108450022771594  -
mpool  version        13          default
mpool  bootfs         -           default
mpool  delegation     on          default
mpool  autoreplace    off         default
mpool  cachefile      -           default
mpool  failmode       wait        default
mpool  listsnapshots  off         default
```

To turn on the autoreplace feature, use the following command line:

```
# zpool autoreplace=on mpool
```

To simulate a bad device, I shut down the system and removed c5t4d0 from the system. The system now cannot contact the disk and has marked the removed disk as unavailable. With the system rebooted, you can examine the output of the `zpool status` command:

```
1   $ zpool status mpool
2   pool: mpool
3   state: DEGRADED
4   status: One or more devices could not be opened.  Sufficient replicas exist for
5   the pool to continue functioning in a degraded state.
6   action: Attach the missing device and online it using 'zpool online'.
7   see: http://www.sun.com/msg/ZFS-8000-2Q
8   scrub: resilver completed after 0h0m with 0 errors on Mon Apr  6 00:52:36 2009
9   config:
10
11  NAME           STATE     READ WRITE CKSUM
12  mpool          DEGRADED     0     0     0
13    mirror       ONLINE       0     0     0
14      c5t2d0     ONLINE       0     0     0
15      c5t3d0     ONLINE       0     0     0
16    mirror       DEGRADED     0     0     0
17      spare      DEGRADED     0     0     0
18        c5t4d0   UNAVAIL      0    89     0  cannot open
19        c5t14d0  ONLINE       0     0     0  31K resilvered
20      c5t5d0     ONLINE       0     0     0  31K resilvered
21  spares
22    c5t14d0      INUSE     currently in use
23
24  errors: No known data errors
```

On line 3, the state of the pool has been degraded, and line 4 tells you that the pool can continue in this state. On lines 6 and 7, ZFS tells you what actions you will need to take, and by going to the Web site, a more detailed message tells you how to correct the problem. Line 19 tells you that the spare disk has been resilvered with disk c5t5d0. Line 22 now gives you the new status of the spare disk c5t14d0.

The original definition of **resilver** is the process of restoring a glass mirror with a new silver backing. In ZFS, it is a re-creation of data by copying from one disk to another. In other volume management systems, the process is called **resynchronization**. Continuing the example, you can shut down the system and attach a new disk in the same location of the missing disk. The new disk at location c5t4d0 is automatically resilvered to the mirrored vdev, and the spare disk is put back to an available state.

```
$ zpool status mpool
pool: mpool
state: ONLINE
scrub: resilver completed after 0h0m with 0 errors on Mon Apr  6 02:21:05 2009
config:

NAME        STATE     READ WRITE CKSUM
mpool       ONLINE       0     0     0
  mirror    ONLINE       0     0     0
    c5t2d0  ONLINE       0     0     0
    c5t3d0  ONLINE       0     0     0
  mirror    ONLINE       0     0     0
    c5t4d0  ONLINE       0     0     0  23K resilvered
    c5t5d0  ONLINE       0     0     0  23K resilvered
 spares
  c5t14d0   AVAIL

errors: No known data errors
```

If the original disk is reattached to the system, ZFS does not handle this case with the same grace. Once the system is booted, the original disk must be detached from the ZFS pool. Next, the spare disk (c5d14d0) must be replaced with the original disk. The last step is to place the spare disk back into the spares group.

```
$ pfexec zpool detach mpool c5t4d0
$ pfexec zpool replace mpool c5t14d0 c5t4d0
$ pfexec zpool add mpool spare c5t14d0
$ zpool status mpool
pool: mpool
state: ONLINE
scrub: resilver completed after 0h0m with 0 errors on Mon Apr  6 02:50:25 2009
config:

NAME         STATE     READ WRITE CKSUM
mpool        ONLINE       0     0     0
  mirror     ONLINE       0     0     0
    c5t2d0   ONLINE       0     0     0
    c5t3d0   ONLINE       0     0     0
  mirror     ONLINE       0     0     0
    c5t4d0   ONLINE       0     0     0  40.5K resilvered
    c5t5d0   ONLINE       0     0     0  40.5K resilvered
spares
  c5t14d0    AVAIL

errors: No known data errors
```

2.8 Locating Disks for Replacement

Locating the correct disk for replacement is crucial. If the wrong disk is replaced, it could corrupt the whole ZFS pool. There is a simple process for locating a disk in an array using the `format` command. This technique works best when the disks are relatively quiet and have disk-access LEDs that can be seen while in operation.

If the disk is still accessible to Solaris, follow these steps:

1. Start the `format` command, and select the disk you want to locate by selecting the number before the drive.
2. Type **`analyze`**, and hit the Enter/Return key.
3. Select read test by typing **`read`**, and then hit Enter/Return.
4. Look at the array, and find the disk LED with constant access.
5. Once the disk is located, stop the read test by pressing Ctrl+C.
6. To exit the format utility, type **`quit`** and hit Return, and then type **`quit`** and hit Return again.
7. Replace the disk according to manufacturer's instructions.

If the damaged disk is not seen by `format`, try to light up the LEDs of the disks above and below the target number of the damaged disk. For example, if you were looking to find c5t6d0 and it was not seen by the `format` command, you would first locate c5t5d0 by using the format read test and then locate c5t7d0 by using the same method. The drive in between should be c5t6d0.

2.9 Example of a Misconfigured Pool

The following is an example of a ZFS pool that started as a RAID-Z pool and needed more space. The administrator just added the new disks to the pool. The status output follows.

```
$ zpool status -v
pool: mypool
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: resilver completed with 0 errors on Tue Nov 18 17:05:10 2008
config:

NAME         STATE      READ WRITE CKSUM
mypool       ONLINE        0     0    18
  raidz1     ONLINE        0     0     0
    c2t0d0   ONLINE        0     0     0
    c2t1d0   ONLINE        0     0     0
    c2t2d0   ONLINE        0     0     0
    c2t3d0   ONLINE        0     0     0
    c2t4d0   ONLINE        0     0     0
    c2t5d0   ONLINE        0     0     0
  c2t9d0     ONLINE        0     0     0
  c2t11d0    ONLINE        0     0     0
  c2t12d0    ONLINE        0     0    18
  c2t10d0    ONLINE        0     0     0
  c2t13d0    ONLINE        0     0     0
spares
  c2t8d0     AVAIL

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

        mypool/dataset/u01:<0x2a07a>
        mypool/dataset/u01:<0x2da66>
```

c2t9d0, c2t10d0, c2t11d0, c2t12d0, and c2t13d0 are single-disk vdevs. They have no redundancy. The spare vdev will not help because there is nothing to resilver with when one of the single-disk vdevs fails. In this case, ZFS has detected permanent errors in mypool/dataset/u01 and has suggested an action of restoring the files from backup or restoring the whole pool from backup.

The ZFS pool is in danger of a single disk failure that may destroy the whole pool. The only way to get this pool to be redundant is to mirror all the single-vdev drives.

Figure 2.7 represents the ZFS pool mypool. The figure shows five single-disk vdevs with a single RAID-Z vdev. Any damage to the single disks will cause ZFS to lose data, or, worse, the pool will fail.

You can correct this configuration for pool redundancy in two ways:

- If there are enough disks, you can mirror the single-disk vdevs.
- You can back up all the file systems and re-create the pool. Then do a full restore of the file systems.

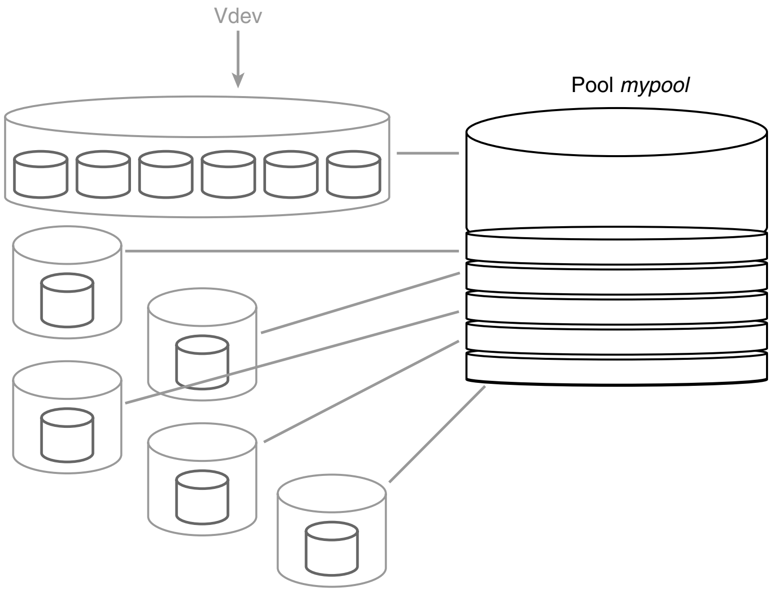

Figure 2.7 The ZFS pool *mypool* has a single RAID-Z vdev and five single-disk vdevs

The procedure to replace a single-disk vdev in the preceding example is to mirror the drive in question and then detach the bad drive. What follows is the command-line procedure:

1. Mirror the drive with bad sector problems:

   ```
   # zpool replace mypool <baddisk> <gooddisk>
   ```

2. Detach the bad disk:

   ```
   # zpool detach mypool <baddisk>
   ```

Index