http://www.sun.com/bigadmin/features/articles/zfs_overview.jsp
Dec 14, 2007

BigAdmin System Administration Portal

Feature Article

ZFS Overview and Guide

Don Turnbull, March 2007

ZFS is Sun Microsystems' newest method of managing storage in the Solaris 10 Operating System. ZFS incorporates a volume management system with a POSIX-compliant file system and a rich set of management tools. From the perspective of applications, the file system behaves exactly as traditional file systems behave but offers much improved performance and capacity. Finally, ZFS affords the system administrator with a wide range of tools comparable to the best volume management and backup solutions.

ZFS Resources

Training Resources

- Solaris 10: Ten Moves Ahead of the Competition (FREE)
- Solaris 10 New Features for Experienced Solaris System Administrators
- Training and Certification Hub

This guide is intended to provide an overview that is useful when configuring ZFS for the first time. ZFS incorporates many features and options not covered in this guide. These features are useful in specific circumstances not suitable for inclusion in this general overview. Please refer to the man pages, `zfs`(1M) and `zpool`(1M), for more detailed information. Additional documentation may be found at docs.sun.com. Another excellent resource is the OpenSolaris ZFS Community. For more resources, see the For More Information section below.

General Concepts

Storage Pools

ZFS organizes physical devices into logical pools called *storage pools*. Both individual disks and array logical unit numbers (LUNs) visible to the operating system may be included in a ZFS pools. ZFS can be based on other less traditional storage structures as well, but these features are not covered in this guide.

Storage pools can be sets of disks striped together with no redundancy (RAID 0), mirrored disks (RAID 1), striped mirror sets (RAID 1 + 0), or striped with parity (RAID Z). Additional disks can be added to pools at any time but they must be added with the same RAID level. For example, if a pool is configured with RAID 1, disks may be added only to the pool in mirrored sets in the same number as was used when the pool was created. As disks are added to pools, the additional storage is automatically used from that point forward.

Note: Adding disks to a pool causes data to be written to the new disks as writes are performed on the pool. Existing data is not redistributed automatically, but is redistributed when modified.

When organizing disks into pools, the following issues should be considered:

- *Disk contention*. Use whole disks in storage pools. If individual disk partitions are used to build a storage pool, contention could occur between the pool and the partitions not included in the pool. Never include a disk in more than one storage pool.
- *Controller contention*. Try to balance anticipated load across the available controllers. For example, when configuring disks for an Oracle database, build the pool that will hold the data table spaces with disks assigned to different controllers than the pool that will hold index table spaces. A disk's controller number is represented by the number after the "c" in the disk's device file name (`/dev/dsk/c1t0d0` is a disk on controller 1).
- *File system layout.* More than one file system can be built in a single storage pool. Plan the size and composition of pools to accommodate similar file systems. For example, rather than building 10 pools for 10 file systems, performance will likely be better if you build two pools and organize the file systems in each pool to prevent contention within the pools (that is, do not use the same pool for both indexes and table data).

Note: RAID-Z is a special implementation of RAID-5 for ZFS allowing stripe sets to be more easily expanded with higher performance and availability.

ZFS File System

Tip 1:

Storage pools perform better as more disks are included. Include as many disks in each pool as possible and build multiple file systems on each pool.

ZFS offers a POSIX-compliant file system interface to the operating system. In short, a ZFS file system looks and acts exactly like a UFS file system except that ZFS files can be much larger, ZFS file systems can be much larger, and ZFS will perform much better when configured properly.

Note: It is not necessary to know how big a file system needs to be to create it.

ZFS file systems will grow to the size of their storage pools automatically.

ZFS file systems must be built in one and only one storage pool, but a storage pool may have more than one defined file system. Each file system in a storage pool has access to all the unused space in the storage pool. As any one file system uses space, that space is reserved for that file system until the space is released back to the pool by removing the file(s) occupying the space. During this time, the available free space on all the file systems based on the same pool will decrease.

ZFS file systems are not necessarily managed in the `/etc/vfstab` file. Special, logical device files can be constructed on ZFS pools and mounted using the `vfstab` file, but that is outside the scope of this guide. The common way to mount a ZFS file system is to simply define it against a pool. All defined ZFS file systems automatically mount at boot time unless otherwise configured.

Finally, the default mount point for a ZFS file system is based on the name of the pool and the name of the file system. For example, a file system named `data1` in pool `indexes` would mount as `/indexes/data1` by default. This default can be overridden either when the file

system is created or later if desired.

Command-Line Interface

The command-line interface consists primarily of the `zfs` and `zpool` commands.. Using these commands, all the storage devices in any system can be configured and made available. A graphical interface is available through the Sun Management Center. Please see the SMC documentation at docs.sun.com for more information.

For example, assume that a new server named `proddb.mydomain.com` is being configured for use as a database server. Tables and indexes must be on separate disks but the disks must be configured for highly available service resulting in the maximum possible usable space. On a traditional system, at least two arrays would be configured on separate storage controllers, made available to the server by means of hardware RAID or logical volume management (such as Solaris Volume Manager) and UFS file systems built on the device files offered from the RAID or logical volume manager. This section describes how this same task would be done with ZFS.

Planning for ZFS

The following steps must be performed prior to configuring ZFS on a new system. All commands must be issued by root or by a user with root authority:

Tip 2:

Use the `format` command to determine the list of available devices and to address configuration problems with those devices.

- Determine which devices are available to the local system.

  Before configuring ZFS, it is important to be certain what disks are available to the system.

- Ensure that these devices are properly configured in the operating system.

  Different platforms and devices require different configuration steps before they can be reliably used with the Solaris OS. Consult the documentation with your storage devices and OS for more information.

- Plan what pools and file systems will be necessary.

  Review the expected use for the system and determine what pools will be necessary and what file systems will be in each pool. File systems can be migrated from pool to pool so this does not have to be precise; expect to experiment until the right balance is struck.

- Determine which devices should be included in which pool.

  Match the list of pools with the list of devices and account for disk and controller contention issues as well as any hardware RAID already applied to the available devices.

Additional planning information can be found at docs.sun.com.

In the running example, two bodies of JBOD ("just a bunch of disks" or non-RAID managed storage) are attached to the server. Though there is no reason to avoid hardware RAID systems when using ZFS, this example is clearer without hardware RAID systems. The following table lists the physical devices presented from attached storage.

c2t0d0	c4t0d0	c3t0d0	c5t0d0
c2t1d0	c4t1d0	c3t1d0	c5t1d0
c2t0d0	c4t2d0	c3t2d0	c5t2d0
c2t1d0	c4t3d0	c3t3d0	c5t3d0

Based on the need to separate indexes from data, it is decided to use two pools named `indexes` and `tables`, respectively. In order to avoid controller contention, all the disks from controllers 2 and 4 will be in the `indexes` pool and those from controllers 3 and 5 will be in the `tables` pool. Both pools will be configured using RAID-Z for maximum usable capacity.

Creating a Storage Pool

Storage pools are created with the `zpool` command. Please see the man page, `zpool` (1M), for information on all the command options. However, the following command syntax builds a new ZFS pool:

```
#  zpool create <pool_name> [<configuration>] <device_files>
```

The command requires the user to supply a name for the new pool and the disk device file names without path (`c#t#d#` as opposed to `/dev/dsk/c#t#d#`). In addition, if a configuration flag, such as `mirror` or `raidz`, is used, the list of devices will be configured using the requested configuration. Otherwise, all disks named are striped together with no parity or other highly available features.

Tip 3:

Check out the `-m` option for defining a specific mount point or the `-R` option for redefining the relative root path for the default mount point.

Continuing the example, the `zpool` commands to build two RAID-Z storage pools of eight disks, each with minimum controller contention, would be as follows:

```
# zpool create indexes raidz c2t0d0 c2t1d0 c2t2d0 \
  c2t3d0 c4t0d0 c4t1d0 c4t2d0 c4t3d0
# zpool create tables raidz c3t0d0 c3t1d0 c3t2d0 \
  c3t3d0 c5t0d0 c5t1d0 c5t2d0 c5t3d0
```

The effect of these commands will be to create two pools named `indexes` and `tables`, respectively, each with RAID-Z striping and data redundancy. ZFS pool names can be named anything starting with a letter except the strings `mirror`, `raidz`, `spare`, or any string starting with `c#` where `#` is any digit 0 through 9. ZFS pool names can include only letters, digits, dashes, underscores, or periods.

The pools are automatically brought online and mounted as ZFS file systems against `/data1` and `/data2`, respectively.

Creating File Systems

If the default file system that is created is not adequate to suit the needs of the system, additional file systems can be created using the `zfs` command. Please see the man page, `zfs` (1M), for detailed information on the command's options.

Suppose, in the running example, two databases were to be configured on the new storage and for management purposes, each database needed to have its own mount points in the `indexes` and `tables` pools. Use the `zfs` command to create the desired file systems as follows:

```
# zfs create indexes/db1
# zfs create indexes/db2
# zfs create tables/db1
# zfs create tables/db2
```

Note: Be careful when naming file systems. It is possible to reuse the same name for different file systems in different pools, which might be confusing.

The effect is to add a separate mount point for `db1` and `db2` under each of `/indexes` and `/tables`. In the mount output, something like the following would be shown:

```
/indexes on indexes read/write/setuid/devices/exec/atime/dev=2f90002 on Tue Oct 17
/tables on tables read/write/setuid/devices/exec/atime/dev=2f90002 on Tue Oct 17 0
/indexes/db1 on indexes/db1 read/write/setuid/devices/exec/atime/dev=2f90003 on Tu
/indexes/db2 on indexes/db2 read/write/setuid/devices/exec/atime/dev=2f90003 on Tu
/tables/db1 on tables/db1 read/write/setuid/devices/exec/atime/dev=2f90003 on Tue
/tables/db2 on tables/db2 read/write/setuid/devices/exec/atime/dev=2f90003 on Tue
```

The space available to `/indexes`, `/indexes/db1`, and `/indexes/db2` is all of the space defined in the `indexes` pool. Likewise, the space available to `/tables`, `/tables/db1`, and `/tables/db2` is all of the space defined in the `tables` pool. The file systems `db1` and `db2` in each pool are mounted as separate file systems in order to provide distinct control and management interfaces for each defined file system.

Displaying Information

Tip 4:

Check out the `set` options of the `zfs` command to manipulate the mount point and other properties of each file system.

Information on the pools and file systems can be displayed using the `list` commands for `zpool` and `zfs`. Other commands exist as well. Please read the man pages for `zfs` and `zpool` for the complete list.

```
# zpool list

NAME            SIZE    USED   AVAIL    CAP  HEALTH     ALTROOT
indexes         240M    110K    240M     0%  ONLINE     -
tables          240M    110K    240M     0%  ONLINE     -

# zfs list

NAME            USED    AVAIL   REFER  MOUNTPOINT
```

```
indexes          107K    208M   25.5K  /indexes
indexes/db1      24.5K   208M   24.5K  /indexes/db1
indexes/db2      24.5K   208M   24.5K  /indexes/db2
tables           107K    208M   25.5K  /tables
tables/db1       24.5K   208M   24.5K  /indexes/db1
tables/db2       24.5K   208M   24.5K  /indexes/db2
```

Monitoring

Though a detailed discussion of monitoring is out of this document's scope, this overview would be incomplete without some mention of the ZFS built-in monitoring. As with management, the command to monitor the system is simple:

```
# zpool iostat <pool_name> <interval> <count>
```

This command works very much like the `iostat` command found in the operating system. If the pool name is not specified, the command reports on all defined pools. If no count is specified, the command reports until stopped. A separate command was needed as the `iostat` command in the operating system cannot see the true reads and writes performed by ZFS; it can see only those submitted to and requested from file systems.

The command output is as follows:

```
# zpool iostat test_pool 5 10

               capacity     operations    bandwidth
pool          used  avail   read  write   read  write
----------  -----  -----  -----  -----  -----  -----
test_pool     80K  1.52G      0      7      0   153K
test_pool     80K  1.52G      0      0      0      0
test_pool     80K  1.52G      0      0      0      0
test_pool     80K  1.52G      0      0      0      0
test_pool     80K  1.52G      0      0      0      0
test_pool     80K  1.52G      0      0      0      0
test_pool     80K  1.52G      0      0      0      0
test_pool     80K  1.52G      0      0      0      0
test_pool     80K  1.52G      0      0      0      0
test_pool     80K  1.52G      0      0      0      0
```

Other commands can be used to contribute to an administrator's understanding of the status, performance, options, and configuration of running ZFS pools and file systems. Please read the man pages for `zfs` and `zpool` for more information.

Summary

With just the two commands, `zpool` and `zfs`, the database server `proddb.mydomain.com` has four mounted file systems: two for indexes and two for tables. When the databases are created, the tables for `db1` would go into files in `/tables/db1` and the indexes for `db1` would go into `/indexes/db1`. The locations for the second database, `db2`, would be `/indexes/db2` and `/tables/db2`. Mount points under `/indexes` are based on distinctly different disks and controllers than those under `/tables`. Finally, using RAID-Z, one disk in each pool can fail before there is any data lost.

Please note that there is no requirement in ZFS that disks be separated by controller, be stand-alone drives, be internal or external, be managed by hardware RAID controllers, or be attached using any particular technology. ZFS will manage any storage device that represents itself all or in part as a device file in the Solaris OS. The example is meant to provide context for the information in this guide; it does not provide a complete demonstration of ZFS capabilities.

Sample Script

Here is a sample script that can be cut and pasted to perform the running example. Be sure to change anything in the script shown in **bold** to suit the local requirements:

```
#!/bin/sh
# This script must be run as root or as a user with appropriate
# role-based access control (RBAC) privileges.

#################################################################
#
# Create pools
#
# Creating pools also creates mount points. If you do not need
# more mount points than pools and the pool names are acceptable
# for mount points, you need not do more than create the pools.
#
# Create as many pools as you want.  Pools define the groups of
# disks under management. All disks in a pool must be managed
# in the same way (RAID-Z, mirrored, and so on), but each pool
# may have a different raid level. Use pools in the same way
# traditional RAID manager LUNs are used.
#
# Create first pool. Change the pool name (data1), the raid flag
# (raidz), and the disk names (c#t#d#) to suite your local
# requirements. Read the zpool man page to determine the available
# options for the raid flag.

zpool create data1 raidz c2t0d0 c2t1d0 c2t2d0 c2t3d0 c4t0d0 c4t1d0 c4t2d0 c4t3d0

# Create second pool.  Change the pool name (data2), the raid flag
# (raidz), and the disk names (c#t#d#) to suite your local
# requirements. Read the zpool man page to determine the available
# options for the raid flag.

zpool create data2 raidz c3t0d0 c3t1d0 c3t2d0 c3t3d0 c5t0d0 c5t1d0 c5t2d0 c5t3d0

#################################################################
#
# Create mounts
#
# Creating mounts may not be necessary as each pool forms a mount
# by default. However, if you need more than one mount point per
# pool, use the following commands to provide distinct mount
# points based on the pools created above.  All mount points have
# access to the full amount of free space in the mount point's
# designated pool. As any one mount point uses space, available
# space for all mount points decreases.
#
# Create the first mount point. The mount will be in the path
# shown from root unless changed later. Be sure to change the
# pool name (data1) to suit the pools created previously and the
# file system name (fs1) to suit your local needs.  Please see
```

```
# the zfs man page for information on options and settings.

zfs create data1/fs1

# Create the first mount point. The mount will be in the path
# shown from root unless changed later. Be sure to change the
# pool name (data1) to suit the pools created previously and the
# file system name (fs2) to suit your local needs.  Please
# see the zfs man page for information on options and settings.

zfs create data1/fs2

# Create the first mount point. The mount will be in the path
# shown from root unless changed later. Be sure to change
# the pool name (data2) to suit the pools created previously
# and the file system name (fs1) to suit your local needs.
# Please see the zfs man page for information on options and
# settings.

zfs create data2/fs1

# Create the first mount point. The mount will be in the path
# shown from root unless changed later. Be sure to change the
# pool name (data2) to suit the pools created previously and
# the file system name (fs2) to suit your local needs.  Please
# see the zfs man page for information on options and settings.

zfs create data2/fs2
```

For More Information

- ZFS, Sun's Cutting-Edge File System (Part 1: Storage Integrity, Security, and Scalability)
- ZFS, Sun's Cutting-Edge File System (Part 2: Ease of Administration and Future Enhancements)
- The OpenSolaris ZFS Community
- The ZFS Learning Center
- Solaris 10 OS Update 06/06 with ZFS