ZFS Best Practices Guide

From Siwiki

Contents

ZFS Administration Considerations

ZFS Storage Pools Recommendations

This section describes general recommendations for setting up ZFS storage pools.

System/Memory/Swap Space

- Run ZFS on a system that runs a 64-bit kernel
- One GB or more of memory is recommended.
- Approximately 64 KB of memory is consumed per mounted ZFS file system. On systems with 1,000s of ZFS file systems, provision 1 GB of extra memory for every 10,000 mounted file systems including snapshots. Be prepared for longer boot times on these systems as well.

- Because ZFS caches data in kernel addressable memory, the kernel sizes will likely be larger than with other file systems. You might configure additional disk-based swap areas to account for this difference on systems with limited RAM. You can use the size physical memory size as an upper bound for the extra amount of swap space that might be required. In any case, monitor swap space usage to determine if swapping occurs.
- For additional memory considerations, see Memory and Dynamic Reconfiguration Recommendations (http://www.solarisinternals.com/wiki/index.php /ZFS_Best_Practices_Guide#Memory_and_Dynamic_Reconfiguration_Recommendations)

Storage Pools

- Set up one storage pool using whole disks per system, if possible.
- For production systems, use whole disks rather than slices for storage pools for the following reasons:
  - Allows ZFS to enable the disk's write cache for those disks that have write caches. If you are using a RAID array with a non-volatile write cache, then this is less of an issue and slices as vdevs should still gain the benefit of the array's write cache.
    - Note: How did the above comment get in here? It seems to be incorrect for two reasons:
      - Even if using whole disks, ZFS does not seem to enable the disk write cache by default. It seems that by default, the disk cache is enable for read, and disabled for write, and the behavior is the same for whole disks, or slices. At least this is true for the one system I tested, using "format -e" and "cache" to display and set the cache settings.
      - Also, the statement implies that enabling the caches would be a good thing. However, there seems to be no benefit of enabling either the disk read or write cache. For async writes, ZFS aggregates operations into large sequential blocks before they reach the disk, hence the write cache does not offer any significant benefit. For sync writes, since the disk write cache is volatile, the data must be flushed from cache onto the platter before the sync operation can complete anyway. Hence there is no significant benefit to enabling the disk write cache. As for read cache: ZFS is more intelligent about read cache than the disk is. The disk can only cache things which it has already read recently (and therefore ZFS has already cached in system ram) or the disk can readahead sequential blocks before the OS requests to read them, but ZFS already has a more intelligent readahead and already requests those blocks if they're related to the present read operation. Hence the disk read cache offers no significant benefit either.
    - The one point which stands is: For sync write operations, you definitely benefit by having nonvolatile writeback in your HBA, or by using dedicated log devices. Neither one of these is a significant benefit for async writes.
  - The recovery process of replacing a failed disk is more complex when disks contain both ZFS and UFS file systems on slices.
  - ZFS pools (and underlying disks) that also contain UFS file systems on slices cannot be easily migrated to other systems by using zpool import and export features.
  - In general, maintaining slices increases administration time and cost. Lower your administration costs by simplifying your storage pool configuration model.
    - Note: See the Additional Cautions section below prior to Nevada, build 117 bug id 6844090 (http://bugs.opensolaris.org/bugdatabase/view_bug.do?bug_id=6844090) .
- If you must use slices for ZFS storage pools, review the following:
  - Consider migrating the pools to whole disks after a transition period.
  - Use slices on small systems, such as laptops, where experts need access to both UFS and ZFS file systems.
  - However, take great care when reinstalling OSes in different slices so you don't accidentally clobber your ZFS pools.
  - Managing data on slices is more complex than managing data on whole disks.
- For production environments, configure ZFS so that it can repair data inconsistencies. Use ZFS redundancy, such as RAIDZ, RAIDZ-2, RAIDZ-3, mirror, or copies > 1, regardless of the RAID level implemented on the underlying storage device. With such redundancy, faults in the underlying storage device or its connections to the host can be discovered and repaired by ZFS.
- Avoid creating a RAIDZ, RAIDZ-2, RAIDZ-3, or a mirrored configuration with one logical device of 40+ devices. See the sections below for examples of redundant configurations.
- In a replicated pool configuration, leverage multiple controllers to reduce hardware failures and to improve performance. For example:

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

- Set up hot spares to speed up healing in the face of hardware failures. Spares are critical for high mean time to data loss (MTTDL) environments. One or two spares for a 40-disk pool is a commonly used configuration. For example:

```
# zpool create tank mirror c1t0d0 c2t0d0 [mirror cxtydz ...] spare c1t1d0 c2t1d0
```

- Run zpool scrub on a regular basis to identify data integrity problems. If you have consumer-quality drives, consider a weekly scrubbing schedule. If you have datacenter-quality drives, consider a monthly scrubbing schedule. You should also run a scrub prior to replacing devices or temporarily reducing a pool's redundancy to ensure that all devices are currently operational.
- ZFS works well with the following devices:
  - Solid-state storage devices that emulate disk drives (SSDs). You might wish to enable compression on storage pools that contain such devices because of their relatively high cost per byte.
  - iSCSI devices. For more information, see the ZFS Administration Guide and the following blog: x4500_solaris_zfs_iscsi_perfect (http://blogs.sun.com/constantin/entry/x4500_solaris_zfs_iscsi_perfect)
  - Storage based protected LUNs (RAID-5 or mirrored LUNs from intelligent storage arrays). However, ZFS cannot

heal corrupted blocks that are detected by ZFS checksums.

Hybrid Storage Pools (or Pools with SSDs)

- There are two possible ways to accelerate your ZFS pool through hybrid storage. By using "cache" devices, you may accelerate read operations. By using "log" devices, you may accelerate synchronous write operations.
- For more information about cache devices, please see #Separate_Cache_Devices
- For more information about log devices, please see #Separate_Log_Devices

Using Storage Arrays in Pools

- With MPxIO
  - Running the Solaris 10 5/09 release is recommended.
  - Enable MPxIO by using the stmsboot command. The paths will change (under /scsi_vhci), but ZFS can handle this change.
- ZFS and Array Replication Interactions Template:Draft
  - ZFS does not support the ability for a Solaris host to have both the the ZFS storage pool contained on the Master Volume and a controller-based (or host-based) snapshot of said ZFS storage pool accessible on the Shadow Volume. This Shadow Volume can be accessed on another Solaris host, if the storage array supports multiple hosts, or the snapshot Shadow Volume is used as the source of remote replication, where ZFS storage pool can then be accessed on the secondary node.
  - If the SNDR unit of replication is a ZFS storage pool (replicated as an SNDR I/O consistency group), all ZFS storage pool and file system properties, such as compression, are replicated too.
  - The TrueCopy snapshot feature does not retain write-order consistency across all volumes in a single ZFS storage pool. To address this issue, within TrueCopy, you must create a single I/O consistency group for all volumes in a "named" ZFS storage pool. The other solution is to do the following:

```
# zpool export <entire ZFS storage pool>
# TrueCopy snapshot
# zpool import <entire ZFS storage pool>
```

Additional Cautions for Storage Pools

Review the following cautions before building your ZFS storage pool:

- Do not create a storage pool that contains components from another storage pool. Deadlocks can occur in this unsupported configuration.
- A pool created with a single slice or single disk has no redundancy and is at risk for data loss. A pool created with multiple slices but no redundancy is also at risk for data loss. A pool created with multiple slices across disks is harder to manage than a pool created with whole disks.
- A pool that is not created with ZFS redundancy (RAIDZ or mirror) can only report data inconsistencies. It cannot repair data inconsistencies. A pool created without ZFS redundancy is harder to manage because you cannot replace or detach disks in a non-redundant ZFS configuration.
- Although a pool that is created with ZFS redundancy can help reduce down time due to hardware failures, it is not immune to hardware failures, power failures, or disconnected cables. Make sure you backup your data on a regular basis. Performing routine backups of pool data on non-enterprise grade hardware is important.
- A pool cannot be shared across systems. ZFS is not a cluster file system.
- The size of the replacements vdev, measured by usable sectors, must be the same or greater than the vdev being replaced. This can be confusing when whole disks are used because different models of disks may provide a different number of usable sectors. For example, if a pool was created with a "500 GB" drive and you need to replace it with another "500 GB" drive, then you may not be able to do so if the drives are not of the same make, model, and firmware revision. Consider planning ahead and reserving some space by creating a slice which is smaller than the whole disk instead of the whole disk. In Nevada, build 117, it might be possible to replace or attach a disk that is slight smaller than the other disks in a pool. This is CR 6844090.
- Today, vdevs cannot shrink in size. If your plans include replacing vdevs with smaller vdevs, then you will also need to plan to grow the file system as additional capacity is needed or implement a plan to copy the data out of the file system before shrinking. CR 4852783 (http://bugs.opensolaris.org/view_bug.do?bug_id=4852783) addresses the ability to shrink vdevs.
- A disk that is part of a pool cannot be relabeled or repartitioned.
- Consider that a BIOS or firmware upgrade might inadvertently relabel a disk so carefully review the upgrade changes that might impact the disks of your pool before the upgrade.
- Other hardware upgrades or changes might change the device paths of the devices in your pool. In general, a ZFS storage pool on Sun hardware can handle these changes, but review your hardware manual to see if the pool should be imported or possibly other preparatory steps taken, before upgrading the hardware.

Simple or Striped Storage Pool Limitations

Simple or striped storage pools have limitations that should be considered. Expansion of space is possible by two methods:

- Adding another disk to expand the stripe. This method should also increase the performance of the storage pool because more devices can be utilized concurrently. Be aware that for current ZFS implementations, once vdevs are added, they cannot be removed.

```
# zpool add tank c2t2d0
```

- Replacing an existing vdev with a larger vdev. For example:

```
# zpool replace tank c0t2d0 c2t2d0
```

- ZFS can tolerate many types of device failures.
  - For simple storage pools, metadata is dual redundant, but data is not redundant.
  - You can set the redundancy level for data using the ZFS copies property.
  - If a block cannot be properly read and there is no redundancy, ZFS will tell you which files are affected.
  - Replacing a failing disk for a simple storage pool requires access to both the old and new device in order to put the old data onto the new device.

```
   # zpool replace tank c0t2d0         ### wrong: cannot recreate data because there is no redundancy
   # zpool replace tank c0t2d0 c2t2d0  ### ok
```

Multiple Storage Pools on the Same System

- The pooling of resources into one ZFS storage pool allows different file systems to get the benefit from all resources at different times. This strategy can greatly increase the performance seen by any one file system.

- If some workloads require more predictable performance characteristics, then you might consider separating workloads into different pools.

- For instance, Oracle log writer performance is critically dependent on I/O latency and we expect best performance to be achieved by keeping that load on a separate small pool that has the lowest possible latency.

ZFS Root Pool Considerations

- A root pool must be created with disk slices rather than whole disks. Allocate the entire disk capacity for the root pool to slice 0, for example, rather than partition the disk that is used for booting for many different uses. A root pool must be labeled with a VTOC (SMI) label rather than an EFI label.
- A disk that contains the root pool or any pool cannot be repartitioned while the pool is active. If the entire disk's capacity is allocated to the root pool, then it is less likely to need more disk space.
- Consider keeping the root pool separate from pool(s) that are used for data. Several reasons exist for this strategy:
  - Only mirrored pools and pools with one disk are supported. No RAIDZ or unreplicated pools with more than one disk are supported.
  - You cannot add additional disks to create multiple mirrored vdevs but you can expand a mirrored vdev by using the zpool attach command.
  - Data pools can be architecture-neutral. It might make sense to move a data pool between SPARC and Intel. Root pools are pretty much tied to a particular architecture.
  - In general, it's a good idea to separate the "personality" of a system from its data. Then, you can change one without having to change the other.
- A root pool cannot be exported on the local system. For recovery purposes, you can import a root pool when booted from the network or alternate media.
- Consider using descendent datasets in the root pool for non-system related data because you cannot rename or destroy the top-level pool dataset. Using the top-level pool dataset as a container for descendent datasets provide more flexibility if you need to snapshot, clone, or promote datasets in the root pool or a top-level dataset.
- Keep all root pool components, such as the /usr and /var directories, in the root pool.
- Create a mirrored root pool to reduce downtime due to hardware failures.
- For more information about setting up a ZFS root pool, see ZFS Root Pool Recommendations (http://www.solarisinternals.com/wiki/index.php /ZFS_Troubleshooting_Guide#ZFS_Root_Pool_Recommendations_and_Requirements) .
- For more information about migrating to a ZFS root file system, see Migrating to a ZFS Root File System (http://www.solarisinternals.com/wiki/index.php/ZFS_Best_Practices_Guide#Migrating_to_a_ZFS_Root_File_System) .

ZFS Mirrored Root Pool Disk Replacement

If a disk in a mirrored root pool fails, you can either replace the disk or attach a replacement disk and then detach the failed disk. The basic steps are like this:

- Identify the disk to be replaced by using the zpool status command.
- You can do a live disk replacement if the system supports hot-plugging. On some systems, you might need to offline and unconfigure the failed disk first. For example:

```
# zpool offline rpool c1t0d0s0
# cfgadm -c unconfigure c1::dsk/c1t0d0
```

- Physically replace the disk.
- Reconfigure the disk. This step might not be necessary on some systems.

```
# cfgadm -c configure c1::dsk/c1t0d0
```

- Confirm that the replacement disk has an SMI label and a slice 0 to match the existing root pool configuration.
- Let ZFS know that the disk is replaced.

```
# zpool replace rpool c1t0d0s0
```

- Bring the disk online.

```
# zpool online rpool c1t0d0s0
```

- Install the bootblocks after the disk is resilvered.
- Confirm that the replacement disk is bootable by booting the system from the replacement disk.
- For information about formatting a disk that is intended for the root pool and installing boot blocks, see [[1] (http://www.solarisinternals.com/wiki/index.php/ZFS_Troubleshooting_Guide#Replacing.2FRelabeling_the_Root_Pool_Disk.)]

Storage Pool Performance Considerations

General Storage Pool Performance Considerations

- For better performance, use individual disks or at least LUNs made up of just a few disks. By providing ZFS with more visibility into the LUNs setup, ZFS is able to make better I/O scheduling decisions.
- Depending on workloads, the current ZFS implementation can, at times, cause much more I/O to be requested than other page-based file systems. If the throughput flowing toward the storage, as observed by iostat, nears the capacity of the channel linking the storage and the host, tuning down the zfs recordsize should improve performance. This tuning is dynamic, but only impacts new file creations. Existing files keep their old recordsize.
- Tuning recordsize does not help sequential type loads. Tuning recordsize is aimed at improving workloads that intensively manage large files using small random reads and writes.
- Keep pool space under 80% utilization to maintain pool performance. Currently, pool performance can degrade when a pool is very full and file systems are updated frequently, such as on a busy mail server. Full pools might cause a performance penalty, but no other issues. If the primary workload is immutable files (write once, never remove), then you can keep a pool in the 95-98% utilization range. Keep in mind that even with mostly static content in the 95-98% range, write, read, and resilvering performance might suffer.
- For better performance, do not build UFS components on top of ZFS components. For ZFS performance testing, make sure you are not running UFS on top of ZFS components.

See also ZFS for Databases (http://www.solarisinternals.com/wiki/index.php/ZFS_for_Databases) .

Separate Log Devices

The ZFS intent log (ZIL) is provided to satisfy POSIX requirements for synchronous writes. By default, the ZIL is allocated from blocks within the main storage pool. Better performance might be possible by using dedicated nonvolatile log devices such as NVRAM, SSD drives, or a dedicated disk.

- If your server hosts a database, virtual machines, iSCSI targets, acts as a NFS server with the clients mounting in "sync" mode, or in any way has heavy synchronous write requests, then you may benefit by using a dedicated log device for ZIL.
- The benefit of a dedicated ZIL depends on your usage. If you always do async writes, it won't matter at all, because the log device can only accelerate sync writes to be more similar to async writes. If you do many small sync writes, you will benefit a lot. If you do large continuous sync writes, you may see some benefit, but it's not clear exactly how significant.
- If you add a log device to your storage pool, it cannot be removed, prior to zpool version 19. (http://hub.opensolaris.org/bin/view/Community+Group+zfs/19) You can find out your pool version by running the "zpool upgrade" command. The latest pool version in Solaris 10 releases, fully patched, is pool version 15. The latest OpenSolaris 2009.06 release includes pool version 14. The only way to get pool version 19 or greater is to update from OpenSolaris 2009.06 to the developer builds, according to instructions on OpenSolaris Download Center (http://hub.opensolaris.org/bin/view/Main/downloads) or by installing one of the developer builds from genunix.org (http://genunix.org) .
- With two or more nonvolatile storage devices, you can create a mirrored set of log devices.

```
# zpool add tank log mirror c0t4d0 c0t6d0
```

- In a mirrored log configuration, you can always detach (unmirror) devices, but as mentioned above, you cannot remove your last unmirrored log device prior to pool version 19.
- Log devices can be unreplicated or mirrored, but RAIDZ is not supported for log devices.
- Prior to pool version 19, if you have an unmirrored log device that fails, your whole pool is permanently lost.
  - Prior to pool version 19, mirroring the log device is highly recommended.
- In pool version 19 or greater, if an unmirrored log device fails during operation, the system reverts to the default behavior, using blocks from the main storage pool for the ZIL, just as if the log device had been gracefully removed via the "zpool remove" command.
  - In pool version 19 or greater, it may be ok not to mirror log devices. Instead, just add non-mirrored log devices for maximum performance.

```
# zpool add tank log c0t4d0 c0t6d0
```

-
  - If you use unmirrored log devices in pool version 19 or greater, you run the following risks:
    - During normal operation, a log device is normally only written. It is normally only read during bootup, or zpool import. It is possible for a drive to fail in such a way that it does not report any errors during all those writes, and only reports failure during the first read.
    - If you're unlucky enough to have that situation, and discover it after an ungraceful shutdown, while there

  was data supposedly in the log device... If your device was unmirrored, then you will lose the sync writes that were supposedly in the log device. This means you could possibly lose up to several seconds worth of writes that were thought to be complete before the crash.
    - Fortunately, unlike pool versions less than 19, even if the above scenario happens to you, you do not lose your whole pool. You only lose the unplayed writes that were in the failed log device, and the damage is limited to the last seconds before system crash.
- The minimum size of a log device is the same as the minimum size of device in pool, which is 64 MB. The amount of in-play data that might be stored on a log device is relatively small. Log blocks are freed when the log transaction (system call) is committed.
- The maximum size of a log device should be approximately 1/2 the size of physical memory because that is the maximum amount of potential in-play data that can be stored. For example, if a system has 16 GB of physical memory, consider a maximum log device size of 8 GB.
- For a target throughput of X MB/sec and given that ZFS pushes transaction groups every 5 seconds (and have 2 outstanding), we also expect the ZIL to not grow beyond X MB/sec * 10 sec. So to service 100MB/sec of synchronous writes, 1 GB of log device should be sufficient.

Memory and Dynamic Reconfiguration Recommendations

The ZFS adaptive replacement cache (ARC) tries to use most of a system's available memory to cache file system data. The default is to use all of physical memory except 1 GB. As memory pressure increases, the ARC relinquishes memory.

Consider limiting the maximum ARC memory footprint in the following situations:

- When a known amount of memory is always required by an application. Databases often fall into this category.
- On platforms that support dynamic reconfiguration of memory boards, to prevent ZFS from growing the kernel cage onto all boards.
- A system that requires large memory pages might also benefit from limiting the ZFS cache, which tends to breakdown large pages into base pages.
- Finally, if the system is running another non-ZFS file system, in addition to ZFS, it is advisable to leave some free memory to host that other file system's caches.

The trade off is to consider that limiting this memory footprint means that the ARC is unable to cache as much file system data, and this limit could impact performance.

In general, limiting the ARC is wasteful if the memory that now goes unused by ZFS is also unused by other system components. Note that non-ZFS file systems typically manage to cache data in what is nevertheless reported as free memory by the system.

For information about tuning the ARC, see the following section:

http://www.solarisinternals.com/wiki/index.php/ZFS_Evil_Tuning_Guide#Limiting_the_ARC_Cache

Separate Cache Devices

In addition to the in-memory ARC cache, ZFS employs a second level, L2ARC on-disk cache. In a typical configuration, there would be large pool of spindle disks, and a smaller number of SSD's or other high performance devices dedicated to cache. Using L2ARC cache devices may accelerate read operations, especially when some data is read repeatedly, and cannot fit in the system memory ARC cache. This is particularly likely when active processes are consuming the system memory, and in high performance machines which may already be maxed out for RAM. For example, if a machine maxes out at 128G ram, and requires 120G of ram for active processes, and frequently needs to read some data files from disk, then performance could likely be increased by adding a few hundred G of SSD cache devices.

- You can add cache devices with the "zpool add" command.

```
zpool add tank cache c0t5d0 c0t6d0
```

- It is not possible to mirror or use raidz on cache devices, nor is it necessary. If a cache device fails, the data will simply be read from the main pool storage devices instead.

RAIDZ Configuration Requirements and Recommendations

A RAIDZ configuration with N disks of size X with P parity disks can hold approximately (N-P)*X bytes and can withstand P device(s) failing before data integrity is compromised.

- Start a single-parity RAIDZ (raidz) configuration at 3 disks (2+1)
- Start a double-parity RAIDZ (raidz2) configuration at 5 disks (3+2)
- Start a triple-parity RAIDZ (raidz3) configuration at 8 disks (5+3)
- (N+P) with P = 1 (raidz), 2 (raidz2), or 3 (raidz3) and N equals 2, 4, or 8
- The recommended number of disks per group is between 3 and 9. If you have more disks, use multiple groups.

For a RAIDZ configuration example, see x4500 with RAID-Z2 (http://www.solarisinternals.com/wiki/index.php /ZFS_Configuration_Guide#ZFS_Configuration_Example_.28x4500_with_raidz2.29) and Recipe for a ZFS RAID-Z Storage Pool on Sun Fire X4540 (http://blogs.sun.com/timthomas/entry/recipe_for_a_zfs_raid) .

Mirrored Configuration Recommendations

- No currently reachable limits exist on the number of devices
- On a Sun Fire X4500 server, do not create a single mirror with 48 devices. Consider creating 24 2-device mirrors. This

configuration reduces the disk capacity by 1/2, but up to 24 disks or 1 disk in each mirror could be lost without a failure.
- If you need better data protection, a 3-way mirror has a significantly greater MTTDL than a 2-way mirror. Going to a 4-way (or greater) mirror may offer only marginal improvements in data protection. Concentrate on other methods of data protection if a 3-way mirror is insufficient.

For a mirrored ZFS configuration examples, see x4500 with mirror (http://www.solarisinternals.com/wiki/index.php /ZFS_Configuration_Guide#ZFS_Configuration_Example_.28x4500_with_mirror.29) and x4200 with mirror (http://www.solarisinternals.com/wiki/index.php /ZFS_Configuration_Guide#ZFS_Configuration_Example_.28x4200_with_mirror.29) .

Should I Configure a RAIDZ, RAIDZ-2, RAIDZ-3, or a Mirrored Storage Pool?

A general consideration is whether your goal is to maximum disk space or maximum performance.

- A RAIDZ configuration maximizes disk space and generally performs well when data is written and read in large chunks (128K or more).
- A RAIDZ-2 configuration offers better data availability, and performs similarly to RAIDZ. RAIDZ-2 has significantly better mean time to data loss (MTTDL) than either RAIDZ or 2-way mirrors.
- A RAIDZ-3 configuration maximizes disk space and offers excellent availability because it can withstand 3 disk failures.
- A mirrored configuration consumes more disk space but generally performs better with small random reads.
- If your I/Os are large, sequential, or write-mostly, then ZFS's I/O scheduler aggregates them in such a way that you'll get very efficient use of the disks regardless of the data replication model.

For better performance, a mirrored configuration is strongly favored over a RAIDZ configuration particularly for large, uncacheable, random read loads.

For more information about RAIDZ considerations, see When to (and not to) use RAID-Z (http://blogs.sun.com/roch/entry /when_to_and_not_to) .

RAIDZ Configuration Examples

For RAIDZ configuration on a Thumper, mirror c3t0 and c3t4 (disks 0 and 1) as your root pool, with the remaining 46 disks available for user data. The following RAIDZ-2 configurations illustrate how to set up the remaining 46 disks.

```
 * 5x(7+2), 1 hot spare, 17.5 TB
 * 4x(9+2), 2 hot spares, 18.0 TB
 * 6x(5+2), 4 hot spares, 15.0 TB
```

ZFS Migration Strategies

Migrating to a ZFS Root File System

In the SXCE (Nevada) build 90 release or in the Solaris 10 10/08 release, you can migrate your UFS root file system to a ZFS root file system by upgrading to build 90 or the Solaris 10 10/08 release and then using the Solaris Live Upgrade feature to migrate to a ZFS root file system. You can create a mirrored ZFS root pool either during an initial installation or a JumpStart installation. Or, by using the zpool attach command to create a mirrored ZFS root pool after installation.

For information about installing a ZFS root file system or migrating a UFS root file system to a ZFS root file system, see the Installing and Booting a ZFS Root File System (http://docs.sun.com/app/docs/doc/819-5461/zfsboot-1?l=en&a=view)

Migrating a UFS Root File System With Zones to a ZFS Root File System

Keep the following points in mind when using ZFS datasets on a Solaris system with zones installed:

- In the Solaris 10 10/08 release, you can create a zone root path on a ZFS file system. However, supported configurations are limited when migrating a system with zones to a ZFS root file system by using the Solaris Live Upgrade feature. Review the following supported configurations before you begin migrating a system with zones.
  - Migrate System With UFS with Zones to ZFS Root (http://www.solarisinternals.com/wiki/index.php /ZFS_Troubleshooting_Guide#Migrate_a_UFS_Root_File_System_with_Zones_Installed_to_a_ZFS_Root_File_System)
  - Configure ZFS Root With Zone Roots on ZFS (http://www.solarisinternals.com/wiki/index.php /ZFS_Troubleshooting_Guide#Configure_a_ZFS_Root_File_System_With_Zone_Roots_on_ZFS)
  - Upgrade or Patch ZFS Root with Zone Roots on ZFS (http://www.solarisinternals.com/wiki/index.php /ZFS_Troubleshooting_Guide#Upgrade_or_Patch_a_ZFS_Root_File_System_With_Zone_Roots_on_ZFS)
- You can use ZFS as a zone root path in the Solaris Express releases, but keep in mind that patching or upgrading these zones is not supported.
- You cannot associate ZFS snapshots with zones at this time.

For more information about using ZFS with zones, see the Zones FAQ (http://opensolaris.org/os/community/zones/faq/) .

Manually Migrating Non-System Data to a ZFS File System

Consider the following practices when migrating non-system-related data from UFS file systems to ZFS file systems:

- Unshare the existing UFS file systems
- Unmount the existing UFS file systems from the previous mount points
- Mount the UFS file systems to temporary unshared mount points

- Migrate the UFS data with parallel instances of rsync running to the new ZFS file systems
- Set the mount points and the sharenfs properties on the new ZFS file systems

ZFS Interactions With Other Volume Management Products

- ZFS works best without any additional volume management software.
- If you must use ZFS with SVM because you need an extra level of volume management, ZFS expects that 1 to 4 MB of consecutive logical block map to consecutive physical blocks. Keeping to this rule allows ZFS to drive the volume with efficiency.
- You can construct logical devices for ZFS by using volumes presented by software-based volume managers, such as SolarisTM Volume Manager (SVM) or Veritas Volume Manager (VxVM). However, these configurations are not recommended. While ZFS functions properly on such devices, less-than-optimal performance might be the result.

General ZFS Administration Information

- ZFS administration is performed while the data is online.
- For information about setting up pools, see [ZFS Storage Pools Recommendations (http://www.solarisinternals.com /wiki/index.php/ZFS_Best_Practices_Guide#ZFS_Storage_Pools_Recommendations)].
- ZFS file systems are mounted automatically when created.
- ZFS file systems do not have to be mounted by modifying the /etc/vfstab file.
- Currently, ZFS doesn't provide a comprehensive backup/restore utility like ufsdump and ufsrestore commands. However, you can use the zfs send and zfs receive commands to capture ZFS data streams. You can also use the ufsrestore command to restore UFS data into a ZFS file system.
- For most ZFS administration tasks, see the following references:
  - zfs.1m (http://docs.sun.com/app/docs/doc/819-2240/zfs-1m) and zpool.1m (http://docs.sun.com/app/docs/doc/819-2240 /zpool-1m) , provide basic syntax and examples
  - ZFS Administration Guide (http://docs.sun.com/app/docs/doc/817-2271) , provides more detailed syntax and examples
  - zfs-discuss (http://opensolaris.org/os/community/zfs/discussions) , join this OpenSolaris discussion list to ask ZFS questions
- You can use "iostat -En" to display error information about devices that are part of a ZFS storage pool.
- A dataset is a generic term for a ZFS component, such as file system, snapshot, clone, or volume.
- When you create a ZFS storage pool, a ZFS file system is automatically created. For example, the following syntax created a pool named tank and top-level dataset named tank that is mounted at /tank.

```
# zpool create tank mirror c1t0d0 c2t0d0
# zfs list tank
NAME   USED  AVAIL  REFER  MOUNTPOINT
tank    72K  8.24G    21K  /tank
```

- Consider using the top-level dataset as a container for other file systems. The top-level dataset cannot be destroyed or renamed. You can export and import the pool with a new name to change the name of the top-level dataset, but this operation would also change the name of the pool. If you want to snapshot, clone, or promote file system data then create separate file systems in your pool. File systems provide points of administration that allow you to manage different sets of data within the same pool.

OpenSolaris/ZFS Considerations

- Most of the general pool and storage recommendations apply to using ZFS in the OpenSolaris release.

OpenSolaris/ZFS/Virtual Box Recommendations

Template:Draft

- By default, Virtual Box is configured to ignore cache flush commands from the underlying storage. This means that in the event of a system crash or a hardware failure, data could be lost.
- To enable cache flushing on Virtual Box, issue the following command:

```
VBoxManage setextradata <VM_NAME> "VBoxInternal/Devices/<type>/0/LUN#<n>/Config/IgnoreFlush" 0
```

where:

- <VM_NAME> is the name of the virtual machine
- <type> is the controller type, either piix3ide (if you're using thenormal IDE virtual controller) or ahci (if you're using a SATAcontroller)
- <n> is the disk number.
  - For IDE disks, primary master is 0, primary slave is 1, secondary master is 2, secondary slave is 3.
  - For SATA disks, it's simply the SATA disk number.
- Additional notes:
  - This setting can only be enabled for (virtual) SATA/IDE disks. It cannot be enabled for CD/DVD drives. iSCSI behavior is unknown at this time.
  - You can only enable this setting for disks that are attached to the particularvirtual machine. If you enable it for any other disks (LUN#), it willfail to boot with a rather cryptic error message. This means ifyou detach the disk, you have to disable this setting.

Using ZFS for Application Servers Considerations

ZFS NFS Server Practices

Consider the following lessons learned from a UFS to ZFS migration experience:

- Existing user home directories were renamed but they were not unmounted. NFS continued to serve the older home directories when the new home directories were also shared.
- Do not mix UFS directories and ZFS file systems in the same file system hierarchy because this model is confusing to administer and maintain.
- Do not mix NFS legacy shared ZFS file systems and ZFS NFS shared file systems because this model is difficult to maintain. Go with ZFS NFS shared file systems.
- ZFS file systems are shared with the sharenfs file system property and zfs share command. For example:

```
 # zfs set sharenfs=on export/home
```

- This syntax shares the file system automatically. If ZFS file systems need to be shared, use the zfs share command. For example:

```
 # zfs share export/home
```

For information about ZFS-over-NFS performance, see ZFS and NFS Server Performance (http://www.solarisinternals.com/wiki/index.php/ZFS_Best_Practices_Guide#ZFS_and_NFS_Server_Performance) .

ZFS NFS Server Benefits

- NFSv4-style ACLs are available with ZFS file systems and ACL information is automatically available over NFS.
- ZFS snapshots are available over NFSv4 so NFS mounted-home directories can access their .snapshot directories.

ZFS file service for SMB (CIFS) or SAMBA

Many of the best practices for NFS also apply to CIFS or SAMBA servers.

- ZFS file systems can be shared using the SMB service, for those OS releases which support it.

```
 # zfs set sharesmb=on export/home
```

- If native SMB support is not available, then SAMBA offers a reasonable solution.

ZFS Home Directory Server Practices

Consider the following practices when planning your ZFS home directories:

- Set up one file system per user
- Use quotas and reservations to manage user disk space
- Use snapshots to back up users' home directories
- Beware that mounting 1000s of file systems, will impact your boot time

Consider the following practices when migrating data from UFS file systems to ZFS file systems:

- Unshare the existing UFS file systems
- Unmount the existing UFS file systems from the previous mount points
- Mount the UFS file systems to temporary unshared mount points
- Migrate the UFS data with parallel instances of rsync running to the new ZFS file systems
- Set the mount points and the sharenfs properties on the new ZFS file systems

See the ZFS/NFS Server Practices (http://www.solarisinternals.com/wiki/index.php/ZFS_Best_Practices_Guide#ZFS_NFS_Server_Practices) section for additional tips on sharing ZFS home directories over NFS.

ZFS Home Directory Server Benefits

- ZFS can handle many small files and many users because of its high capacity architecture.
- Additional space for user home directories is easily expanded by adding more devices to the storage pool.
- ZFS quotas are an easy way to manage home directory space.
- Use ZFS property inheritance to apply properties to many file systems.

Recommendations for Saving ZFS Data

Using ZFS Snapshots

- Using ZFS snapshots is a quick and easy way to protect files against accidental changes and deletion. In nearly all cases, availability of snapshots allows users to restore their own files, without administrator assistance, and without the need to access removable storage, such as tapes. For example:

```
$ rm reallyimportantfile /* D'oh!
$ cd .zfs/snapshot
$ cd .auto...
$ cp reallyimportantfile $HOME
$ ls $HOME/reallyimportantfile
/home/cindys/reallyimportantfile
```

- The following syntax creates recursive snapshots of all home directories in the tank/home file system. Then, you can use the zfs send -R command to create a recursive stream of the recursive home directory snapshot, which also includes the individual file system property settings.

```
# zfs snapshot -r tank/home@monday
# zfs send -R tank/home/@monday | ssh remote-system zfs receive -dvu pool
```

- You can create rolling snapshots and zfs-auto-snapshots to help manage snapshot copies. For more information, see the Rolling Snapshots Made Easy (http://blogs.sun.com/mmusante/entry/rolling_snapshots_made_easy) blog or the ZFS Automatic Snapshot (http://blogs.sun.com/timf/entry/zfs_automatic_snapshots_0_10) blog by Tim Foster.
- ZFS snapshots are accessible in the .zfs directories of the file system that was snapshot. Configure your backup product to skip these directories.

Storing ZFS Snapshot Streams (zfs send/receive)

- You can use the zfs send and zfs receive commands to archive a snapshot stream but saving ZFS send streams is different from traditional backups, for the following reasons:
  - You cannot select individual files or directories to receive because the zfs receive operation is an all-or-nothing event. You can get all of a file system snapshot or none of it.
  - If you store ZFS send stream on a file or on tape, and that file becomes corrupted, then it will not be possible to receive it, and none of the data will be recoverable. However, Nevada, build 125 adds the zstreamdump(1m) command to verify a ZFS snapshot send stream. See also, RFE 6736794 (http://bugs.opensolaris.org/bugdatabase /view_bug.do?bug_id=6736794) .
  - You cannot restore individual files or directories from a ZFS send stream, although you can copy files or directories from a snapshot. This limitation means enterprise backup solutions and other archive tools, such as cp, tar, rsync, pax, cpio, are more appropriate for tape backup/restore because you can restore individual files or directories.
  - You cannot exclude directories or files from a ZFS send stream.
- You can create an incremental snapshot stream (see "zfs send -i" syntax). This is generally much faster than incremental backups performed by file-level tools, such as tar and rsync, because ZFS already knows which blocks have changed on disk, and it can simply read those blocks as large sequential disk read operations, to the extent physically possible. Archive tools, such as tar and rsync, must walk the file system, checking every file and directory for modifications, in order to choose which files have changed and need to be included in the incremental backup.
- Another advantage of using a ZFS send stream over a file-based backup tool is that you can send and receive a file system's property settings.
- If you have random access storage (not tape) to receive onto, and you don't need to exclude anything, then zfs send and receive can be used to store data, provided that you pipe the zfs send command directly into the zfs receive command.

Using ZFS With AVS

The Sun StorageTek Availability Suite (AVS), Remote Mirror Copy and Point-in-Time Copy services, previously known as SNDR (Sun Network Data Replicator) and II (Instant Image), are similiar to the Veritas VVR (volume replicator) and Flashsnap (point-in-time copy) products, is currently available in the Solaris Express release.

SNDR differs from the ZFS send and recv features, which are time-fixed replication features. For example, you can take a point-in-time snapshot, replicate it, or replicate it based on a differential of a prior snapshot. The combination of AVS II and SNDR features, also allows you to perform time-fixed replication. The other modes of the AVS SNDR replication feature allows you to obtain CDP (continuous data replication). ZFS doesn't currently have this feature.

For more information about AVS, see the OpenSolaris AVS Project (http://www.opensolaris.org/os/project/avs/) . View the AVS/ZFS Demos here (http://www.opensolaris.org/os/project/avs/Demos) .

Using ZFS With Enterprise Backup Solutions

- The robustness of ZFS does not protect you from all failures. You should maintain copies of your ZFS data either by taking regular ZFS snapshots, and saving them to tape or other offline storage. Or, consider using an enterprise backup solution.
- Sun StorEdge Enterprise Backup Software (Legato Networker 7.3.2) product can fully back up and restore ZFS files including ACLs.
- Symantec's Netbackup 6.5 product can fully back up and restore ZFS files including ACLs. Release 6.5.2A (http://www.symantec.com/business/support/overview.jsp?pid=15143) offers some fixes which make backup of ZFS file systems easier.
- IBM's TSM product can be used to back up ZFS files. However supportability is not absolutely clear. Based on TSM documentation on IBM website (http://publib.boulder.ibm.com/infocenter/tivihelp/v1r1/index.jsp?topic= /com.ibm.itsmc.doc_5.3.3/ans5000095.htm) , ZFS with ACLs is supported with TSM client 5.3. It has been verified (internally to Sun (http://opensolaris.org/os/community/zfs/faq/#backupsoftware)) to correctly work with 5.4.1.2.

```
tsm> q file /opt/SUNWexplo/output
 #     Last Incr Date      Type    File Space Name
---    --------------      ----    ---------------
 1   08/13/07   09:18:03   ZFS     /opt/SUNWexplo/output
                            ^
                            |__correct filesystem type
```

- For the latest information about enterprise-backup solution issues with ZFS, see the ZFS FAQ (http://opensolaris.org/os/community/zfs/faq/) .

Using ZFS With Open Source Backup Solutions

- ZFS snapshots are accessible in the .zfs directories of the file system that was snapshot. Configure your backup product to skip these directories.
- Amanda - Joe Little blogs about how he backs up ZFS file systems (http://jmlittle.blogspot.com/2008/08/amanda-simple-zfs-backup-or-s3.html) to Amazon's S3 (http://aws.amazon.com/s3/) using Amanda. (http://www.zmanda.com) Integration of ZFS snapshots with MySQL and Amanda Enterprise 2.6 Software (http://www.sun.com/bigadmin/features/articles/zmanda_sfx4540.pdf) can also take advantage of ZFS snapshot capabilities.
- Bacula - Tips for backing up ZFS data are as follows:
  - Make sure you create multiple jobs per host to allow multiple backups to occur in parallel rather than just one job per host. If you have several file systems and/or pools, running multiple jobs speeds up the backup process.
  - Create a non-global zone to do backups. Having multiple zones for backups means that you can delegate control of a backup server to a customer so they can perform restores on their own and only have access to their own data.
  - Use a large RAIDZ pool (20 TB), instead of tapes, to store all the backups, and which allows quick backup and restores.
  - Use opencsw (from opencsw.org (http://opencsw.org)) and/or blastwave (from blastwave.org (http://www.blastwave.org)) for packages, which makes it very easy to install and maintain. If using opencsw, run "pkg-get -i bacula" and it installs all the prerequisites. If using blastwave, run "pkgutil -i bacula" and it installs all the prerequisites. On the clients, install opencsw and bacula_client on the global zones, and backup the local zones from the global zone.
  - On the server, the "director" configuration file, /opt/csw/etc/bacula/bacula-dir.conf, contains the information you need about what clients are backed up. You can split configurations into sections, such as "core" for the base system, "raid" for my local larger pools, and "zones" for my zones. If you have several zones, break up "zones" to a per-zone job, which is easy to do.
  - For more information, see Bacula Configuration Example (http://www.solarisinternals.com/wiki/index.php/Baculaconfexample) .
- The OpenSolaris ndmp service project is proposed to take advantage of ZFS features, such as snapshots, to improve the backup and restore process. With this addition, an enterprise backup solution could take advantage of snapshots to improve data protection for large storage repositories. For more information, see ndmp service (http://arc.opensolaris.org/caselog/PSARC/2007/397) .

ZFS and Database Recommendations

The information in this section has been consolidated into a separate ZFS for Databases (http://www.solarisinternals.com/wiki/index.php/ZFS_for_Databases) section.

ZFS and Complex Storage Considerations

- Certain storage subsystems stage data through persistent memory devices, such as NVRAM on a storage array, allowing them to respond to writes with very low latency. These memory devices are commonly considered as stable storage, in the sense that they are likely to survive a power outage and other types of breakdown. At critical times, ZFS is unaware of the persistent nature of storage memory, and asks for that memory to be flushed to disk. If indeed the memory devices are considered stable, the storage system should be configured to ignore those requests from ZFS.
- For potential tuning considerations, see: ZFS Evil Tuning Guide, Cache_Flushes (http://www.solarisinternals.com/wiki/index.php/ZFS_Evil_Tuning_Guide#Cache_Flushes)

Virtualization Considerations

ZFS and Virtual Tape Libraries (VTLs)

VTL solutions are hardware and software combinations that are used to emulate tapes, tape drives, and tape libraries. VTLs are used in backup/archive systems with a focus on reducing hardware and software costs.

- VTLs are big disk space eaters and we believe ZFS will allow them to more efficiently and securely manage the massive, online disk space.
  - OpenSolaris - the COMSTAR project delivers both tape and disk targets, so a ZFS volume can look like a tape drive.
  - Falconstor VTL - has been tested on Thumper running ZFS. For more information, see: Sun Puts Thumper To Work (http://www.internetnews.com/storage/article.php/3693306)
  - NetVault from BakBone - This backup solution includes a VTL feature that has been tested on Thumper running ZFS.

ZFS Performance Considerations

See the following sections for basic system, memory, pool, and replication recommendations:

- ZFS Storage Pools Recommendations (http://www.solarisinternals.com/wiki/index.php/ZFS_Best_Practices_Guide#ZFS_Storage_Pools_Recommendations)
- Should I Configure a RAIDZ, RAIDZ-2, RAIDZ-3, or a Mirrored Storage_Pool (http://www.solarisinternals.com/wiki/index.php/ZFS_Best_Practices_Guide#Should_I_Configure_a_RAIDZ.2C_RAIDZ-2.2C_RAIDZ-3.2C_or_a_Mirrored_Storage_Pool.3F)

ZFS and Application Considerations

ZFS and NFS Server Performance

ZFS is deployed over NFS in many different places with no reports of obvious deficiency. Many have reported disappointing performance, but those scenarios more typically relate to comparing ZFS-over-NFS performance with local file system performance. It is well known that serving NFS leads to significant slowdown compared to local or directly-attached file systems, especially for workloads that have low thread parallelism. A dangerous way to create better ZFS-over-NFS performance at the expense of data integrity is to set the kernel variable, zil_disable. Setting this parameter is not recommended.

Later versions of ZFS have implemented a separate ZIL log device option which improves NFS synchronous write operations. This is a better option than disabling the ZIL. Anecdotal evidence suggests good NFS performance improvement even if the log device does not have nonvolatile RAM. To see if your zpool can support separate log devices use *zpool upgrade -v* and look for version 7. For more information, see separate intent log (http://blogs.sun.com/perrin/entry/slog_blog_or_blogging_on) .

See also ZFS for Databases (http://www.solarisinternals.com/wiki/index.php/ZFS_for_Databases) .

For more detailed information about ZFS-over-NFS performance, see ZFS and NFS, a fine combination (http://blogs.sun.com/roch/entry/nfs_and_zfs_a_fine) .

ZFS Overhead Considerations

- Checksum and RAIDZ parity computations occur in concurrent threads in recent Solaris releases.
- Compression is no longer single-threaded due to integration of CR 6460622 (http://bugs.opensolaris.org/view_bug.do?bug_id=6460622) .

Data Reconstruction

Traditional RAID systems, where the context of the data is not known and data is reconstructed (also known as resilvering), blindly reconstruct the data blocks in block order. ZFS only needs to reconstruct data, so ZFS can be more efficient than traditional RAID systems when the storage pool is not full. ZFS reconstruction occurs top-down in a priority-based manner. Jeff Bonwick describes this in more detail in his Smokin' Mirrors (http://blogs.sun.com/bonwick/entry/smokin_mirrors) blog post.

Since ZFS reconstruction occurs on the host, some concern exists over the performance impact and availability trade-offs. Two competing RFEs address this subject:

- CR 6670933, resilver code should prefetch (http://bugs.opensolaris.org/bugdatabase/view_bug.do?bug_id=6678033)
- CR 6494473, ZFS needs a way to slow down resilvering (http://bugs.opensolaris.org/bugdatabase/view_bug.do?bug_id=6494473)

Retrieved from "http://www.solarisinternals.com/wiki/index.php/ZFS_Best_Practices_Guide"

Category: ZFS

- This page was last modified 21:29, 10 August 2010.