NAME

zfsconcepts - overview of ZFS concepts

DESCRIPTION

ZFS File System Hierarchy

A ZFS storage pool is a logical collection of devices that provide space for datasets. A storage pool is also the root of the ZFS file system hierarchy.

The root of the pool can be accessed as a file system, such as mounting and unmounting, taking snapshots, and setting properties. The physical storage characteristics, however, are managed by the zpool(8) command.

See zpool(8) for more information on creating and administering pools.

Snapshots

A snapshot is a read-only copy of a file system or volume. Snapshots can be created extremely quickly, and initially consume no additional space within the pool. As data within the active dataset changes, the snapshot consumes more data than would otherwise be shared with the active dataset.

Snapshots can have arbitrary names. Snapshots of volumes can be cloned or rolled back, visibility is determined by the **snapdev** property of the parent volume.

File system snapshots can be accessed under the *.zfs/snapshot* directory in the root of the file system. Snapshots are automatically mounted on demand and may be unmounted at regular intervals. The visibility of the *.zfs* directory can be controlled by the **snapdir** property.

Bookmarks

A bookmark is like a snapshot, a read-only copy of a file system or volume. Bookmarks can be created extremely quickly, compared to snapshots, and they consume no additional space within the pool. Bookmarks can also have arbitrary names, much like snapshots.

Unlike snapshots, bookmarks can not be accessed through the filesystem in any way. From a storage standpoint a bookmark just provides a way to reference when a snapshot was created as a distinct object. Bookmarks are initially tied to a snapshot, not the filesystem or volume, and they will survive if the snapshot itself is destroyed. Since they are very light weight there’s little incentive to destroy them.

Clones

A clone is a writable volume or file system whose initial contents are the same as another dataset. As with snapshots, creating a clone is nearly instantaneous, and initially consumes no additional space.

Clones can only be created from a snapshot. When a snapshot is cloned, it creates an implicit dependency between the parent and child. Even though the clone is created somewhere else in the dataset hierarchy, the original snapshot cannot be destroyed as long as a clone exists. The **origin** property exposes this dependency, and the **destroy** command lists any such dependencies, if they exist.

The clone parent-child dependency relationship can be reversed by using the **promote** subcommand. This causes the "origin" file system to become a clone of the specified file system, which makes it possible to destroy the file system that the clone was created from.

Mount Points

Creating a ZFS file system is a simple operation, so the number of file systems per system is likely to be numerous. To cope with this, ZFS automatically manages mounting and unmounting file systems without the need to edit the */etc/fstab* file. All automatically managed file systems are mounted by ZFS at boot time.

By default, file systems are mounted under */path*, where *path* is the name of the file system in the ZFS namespace. Directories are created and destroyed as needed.

A file system can also have a mount point set in the **mountpoint** property. This directory is created as needed, and ZFS automatically mounts the file system when the **zfs mount -a** command is invoked (without editing */etc/fstab*). The **mountpoint** property can be inherited, so if *pool/home* has a mount point of */export/stuff*, then *pool/home/user* automatically inherits a mount point of */export/stuff/user*.

A file system **mountpoint** property of **none** prevents the file system from being mounted.

If needed, ZFS file systems can also be managed with traditional tools (**mount**, **umount**, */etc/fstab*). If a file system's mount point is set to **legacy**, ZFS makes no attempt to manage the file system, and the administrator is responsible for mounting and unmounting the file system. Because pools must be imported before a legacy mount can succeed, administrators should ensure that legacy mounts are only attempted after the zpool import process finishes at boot time. For example, on machines using systemd, the mount option

x-systemd.requires=zfs-import.target

will ensure that the zfs-import completes before systemd attempts mounting the filesystem. See systemd.mount(5) for details.

Deduplication

Deduplication is the process for removing redundant data at the block level, reducing the total amount of data stored. If a file system has the **dedup** property enabled, duplicate data blocks are removed

synchronously. The result is that only unique data is stored and common components are shared among files.

Deduplicating data is a very resource-intensive operation. It is generally recommended that you have at least 1.25 GiB of RAM per 1 TiB of storage when you enable deduplication. Calculating the exact requirement depends heavily on the type of data stored in the pool.

Enabling deduplication on an improperly-designed system can result in performance issues (slow I/O and administrative operations). It can potentially lead to problems importing a pool due to memory exhaustion. Deduplication can consume significant processing power (CPU) and memory as well as generate additional disk I/O.

Before creating a pool with deduplication enabled, ensure that you have planned your hardware requirements appropriately and implemented appropriate recovery practices, such as regular backups. Consider using the **compression** property as a less resource-intensive alternative.

Block cloning

Block cloning is a facility that allows a file (or parts of a file) to be "cloned", that is, a shallow copy made where the existing data blocks are referenced rather than copied. Later modifications to the data will cause a copy of the data block to be taken and that copy modified. This facility is used to implement "reflinks" or "file-level copy-on-write".

Cloned blocks are tracked in a special on-disk structure called the Block Reference Table (BRT). Unlike deduplication, this table has minimal overhead, so can be enabled at all times.

Also unlike deduplication, cloning must be requested by a user program. Many common file copying programs, including newer versions of **/bin/cp**, will try to create clones automatically. Look for "clone", "dedupe" or "reflink" in the documentation for more information.

There are some limitations to block cloning. Only whole blocks can be cloned, and blocks can not be cloned if they are not yet written to disk, or if they are encrypted, or the source and destination **recordsize** properties differ. The OS may add additional restrictions; for example, most versions of Linux will not allow clones across datasets.