ZFS and the Infinite Incremental Backup

A brief description of ZFS and how to use snapshots and clones for data backup and recovery

Some Basic Storage Conventions

- Disks
  - A Disk is a physical disk either spinning or solid state
- LUN
  - Logical unit or block device
  - Can be a disk
  - Can be a SAN or managed aggregate device
- Pool
  - A collection of disks managed by a “volume manager”
- Volume
  - A unit of managed storage
- Volume Manager
  - Manages multiple disks in a “pool” or “volume group”
  - LVM
  - ZFS
  - BTRFS

Linux Volume Managers

- LVM – Logical Volume Manager
  - Physical Volumes
  - Volume Groups
  - Logical Volumes
- ZFS – Zettabyte File System
  - Zpools
  - ZFS Objects (volumes)
- BTRFS – Better File System
  - Removed from RedHat 8
  - Similar in capability to ZFS

What is the Zettabyte File System?

- ZFS is both a file system and a volume manager.
  - Takes physical disks and creates a “pool” of storage (like LVM VG)
  - Top level of a pool is a file system (unlike LVM VG)
  - Allocate objects out of the pool
    - File systems
    - Block devices
    - Snapshots
    - clones

ZFS Objects

- A ZFS object is internally represented as a Merkel tree
  - https://en.wikipedia.org/wiki/Merkle_tree
- The Merkel tree points to blocks in the zpool
- ZFS is a “redirect-on-write” system.
  - Blocks are replaced not updated
  - Blocks may be shared across multiple objects
  - Blocks have a reference count
    - When count goes to zero, it is reused

File Systems

- A ZFS file system looks like a common Linux file system
- File systems are sparsely allocated out of the pool.
- They only use space as it is used.
- You can have many filesystems
- A filesystem can have a mount point, e.g. /home
- A ZFS file system is as large as the pool it is allocated out of

ZVOLS

- A ZVOL is a block device allocated out of a ZFS pool
  - This is similar to an LVM logical volume
- A ZVOL is typically pre-allocated out of the pool
  - Can be “sparse” with the -s flag at creation time
    - Only allocates storage as used
    - A sparse volume may be huge but use almost no storage
- A ZVOL has a fixed size
- A ZVOL can be used like any other Linux block device
  - You can format them as EXT3, XFS, and even swap

Snapshots

- A snapshot is a read-only representation of a ZFS object at a specific point in time
- Effectively a copy of an object’s Merkel tree
  - Not really, but it works to think of it that way
- Can not be changed.
- Shares data blocks with object of which it is a snapshot
- An object may have many snapshots

clone

- A clone creates a usable volume from a snapshot
- Uses snapshot for initial contents
- Retains new blocks outside the snapshot
  - A snapshot may have multiple clones
- A zfs object snapshot is used as the basis of a new object using clone

Setup SSH on two systems

- Use ssh-keygen to create an RSA key on main system
- Copy public key to root’s .ssh/authorized_keys on the replication target
- Using ssh from origin system to root on the replication target should proceed with no user interaction

Create Source Pool and Object

- Devel file system
  - `zpool create -o ashift=12 zpool1 /dev/vdb`
  - `mkdir /devel`
  - `zfs create -o compression=lz4 zpool1/devel`
  - `zfs set mountpoint=/devel zpool1/devel`
- /devel is now a ZFS file system

Create Destination Pool

- Devel file system
  - zpool create -o ashift=12 zpool1 /dev/vdb

Perform Initial Backup

- Create Snapshot
  - zfs snapshot zpool1/devel@snap1
- Use send to send first copy
  - zfs send zpool1/devel@snap1 | ssh demo2 "zfs receive -F zpool1/devel"

Create Some Data

```
dd if=/dev/urandom bs=4096 count=20
of=/devel/data
```

Perform Next Backup

- Create snapshot
  - zfs snapshot zpool1/devel@snap2
- Use “send” and ssh to send the changed blocks
  - zfs send -i zpool1/devel@snap1 zpool1/devel@snap2 | ssh demo2 “zfs receive -F zpool1/devel”
    - Sends only data changed since @snap1
    - Snap1 and snap2 are compared and blocks that have changed will be sent. All others will be ignored
    - The receiving object will be identical to the sending object.
- Destroy snap1 and rename snap2 to snap1
  - zfs destroy zpool1/devel@snap1
  - zfs rename zpool1/devel@snap2 zpool1/devel@snap1
  - ssh demo2 ‘zfs destroy zpool1/devel@snap1’
  - ssh demo2 ‘zfs rename zpool1/devel@snap2 zpool1/devel@snap1’

A simple script

```
#!/bin/bash

zfs snapshot zpool1/devel@snap2
zfs send -i zpool1/devel@snap1 zpool1/devel@snap2 | ssh demo2 'zfs receive -F zpool1/devel'

zfs destroy zpool1/devel@snap1
zfs rename zpool1/devel@snap2 zpool1/devel@snap1
ssh demo2 "zfs destroy zpool1/devel@snap1"
ssh demo2 "zfs rename zpool1/devel@snap2 zpool1/devel@snap1"
```

What if you don’t delete the snapshots?

- ZFS snapshots are light weight and unlike LVM do not duplicate data
- They only retain the old changed blocks.
- You can have thousands or tens of thousands of snapshots
- There is little or no impact on performance for snapshots

Are snapshots just for backup?

- Snapshots are an effective tool to “freeze” a volume for backup
- Snapshots can be used for “clones”
- A clone is a read/write volume that uses the snapshot for its original data
- As data is written to a clone, only the clone’s Merkel tree is changed, leaving the snapshot unchanged

Example 1, accident recovery

- Have you ever deleted a file and immediately cursed yourself, and dread re-writing the code?
- You could snapshot every 15 minutes and never lose more than that (assuming you save)
- Use a clone with a snapshot and get your data back
  - Or use zfs rollback to got back to the state of a snapshot
- No noticeable affect on performance

Example 2: Historical Archive

- A snapshot represents a point in time for a volume
- You can create a strategy for long term retention of data, just by using snapshots
- Create snapshots for important dates
  - Once a week
  - Once a month
  - Once a year
  - Release dates
  - Etc.

Example 3: Multiple Volumes that Share Common Base

- Create one volume, take a snapshot and create multiple clones
- The data on the base object is shared with the clones via the snapshot
- Can save data
- Unfortunately, not updatable