Red Hat
Enterprise Linux

ZFS/BTRFS/VxFS: Equivalent functions in RHEL

Lunch ‘n Learn

Mike Pagan
Sr. Platform Strategist

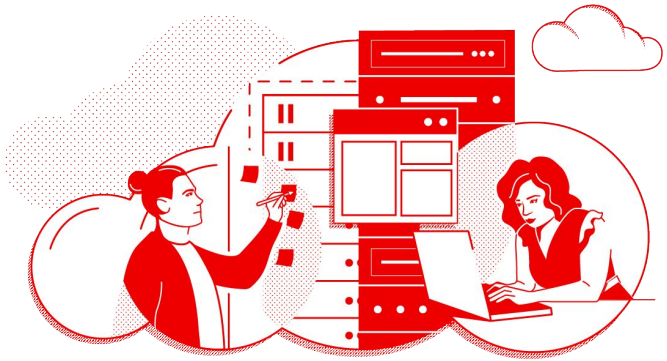
Red Hat

Red Hat Enterprise Linux 8

The purpose of this presentation: To help Linux engineering and operations be aware of the features in RHEL which allow equivalent functionality to legacy filesystems like ZFS, BTRFS, and VxFS

Why do we need alternatives to these filesystems?

What are the factors driving the need



- ZFS
  - ZFS is feature-rich
  - Solaris shops depend on some of its features
  - ...but ZFS is Oracle-proprietary and not open source
  - ...and OpenZFS is not exactly the same as the Solaris version, with less hardening and enterprise usage
- BTRFS
  - Red Hat kept BTRFS in tech preview for many revisions of RHEL
  - In all that time, BTRFS never achieved enough stability for production use
  - There is no plan to go back and incorporate BTRFS into RHEL
- VxFS
  - Commercial product of Veritas (Symantec)
  - Commonly carried forward as a part of a Linux SOE from legacy Unix (Solaris, HP-UX, AIX)
  - Feature-rich but fairly expensive
  - VxFS experienced admins are getting scarce

ZFS-to-RHEL Feature Comparison

	ZFS	RHEL
01	Encryption	LUKS 2.0
02	Pooling	LVM
03	SSD hierarchical storage	dm-cache
04	Data scrubbing	dm-integrity
05	Deduplication & Compression	VDO
06	Unified configuration	Stratis

ZFS-to-RHEL Feature Comparison (continued)

	ZFS	RHEL
07	Copy-on-Write	LVM snapshots & thinp
08	Scalability	XFS
09	Snapshots	LVM
10	Data versioning	N/A (LVM snapshots)
11	RAID-Z	LVM
12	Delegated Permissions	N/A

Encryption

ZFS encryption vs. RHEL encryption

Encrypting root disks

- ZFS can **not** encrypt root disks
- RHEL dm-crypt **can** encrypt root disks

Integration

- ZFS disk encryption is built in to the filesystem
- RHEL disk encryption is achieved by device mapper (dm-crypt) and integrated via RHEL Storage Roles (Ansible) and Stratis

Disk encryption at scale

- RHEL includes network-based disk encryption (NBDE), which allows disk decryption at boot time without storing encryption keys “in the clear”
- ZFS encrypted disk “wrapper” keys can be stored on a centralized server, with security dependant on that server.

Ciphers

- ZFS disk encryption uses AES
- RHEL disk encryption uses AES, twofish, or serpent

Setting up a LUKS2-encrypted volume in RHEL

```
# umount /vgXX/<volume>

# lvextend -L+32M /vgXX/<volume>

# cryptsetup reencrypt --encrypt --init-only \
  Reduce-device-size 32M /vgXX/<volume> <volume>_encrypted

# mount /dev/mapper/<volume>_encrypted \
  /mnt/<volume>_encrypted

# cryptsetup reencrypt --resume-only /vgXX/<volume>
```

Start with an unmounted filesystem
Subsequent reencryption can be done with the filesystem mounted

Add some extra free space (in case you don't have it)
This makes room for the encryption header. This header can be external to the volume if necessary

Run "cryptsetup" with --init-only
This prepares the volume and creates the entry in device-mapper

Mount & re-run "cryptsetup
This performs the actual encryption

Storage Pooling
Snapshots
RAID

ZFS pooling & snapshots vs. RHEL pooling & snapshots

Integration

- ZFS disk pooling & snapshots are built in to the filesystem
- RHEL disk pooling, RAID & snapshots are provided by LVM
- Integration in RHEL provided through Stratis
- API-based management of pools, RAID, and snapshots provided via System Roles (Ansible)

Functionality

Pooling, RAID, and snapshots have been long-standing features of LVM in RHEL and are well-adopted across the user base. .

RAID

ZFS offers RAID-Z

Proprietary software RAID format with several interesting features

RHEL provides RAID in LVM

Software RAID has been a core feature in RHEL since RHEL 3, providing:

- RAID-0
- RAID-1
- RAID-10
- RAID-5
- RAID-6

Performance Considerations

RAID-0 and RAID-1 provide improved or neutral performance vis-a-vis individual volumes. However, RAID-5 and -6 (as well as their equivalent RHAID-Z options) impose a significant performance impact. **Real-world use has shown that higher RAID levels are typically implemented in hardware, not software.**

Storage pooling, RAID, and snapshot commands in RHEL

```
Create a typical volume of size 10GB named "myvolume"from
pooled storage in volume group "vgXX"
# lvcreate -L 10G -n myvolume vgXX

Create a RAID-0 volume across two physical devices with a
stripe size of 64MB
# lvcreate -L 10G -i2 -I64M -n striped_volume vgXX

Create a one-way RAID-1 (mirrored) volume across two
physical devices
# lvcreate -L 10G -m1 -n mirrored_volume vgXX

Create a snapshot of volume "lvolX" limited to 100MB
# lvcreate -size 100M --snapshot --name snap_lvolX \
  /dev/vgXX/lvolX
```

Volume group setup

The examples on the left presume a basic volume group setup previously done with the "vgcreate", "pvcreate", and "vgextend" commands

Pooling

Pooling is automatic within a volume group, with all "lvcreate" commands operating against the entire pool or a chosen subset.

RAID and mirroring

RAID volumes (including mirroring, aka "RAID-1") draw from the same volume group pool

Snapshots

Snapshots create a new logical volume in the same pool.

Red Hat

SSD Hierarchical Storage

Cacheing/Hierarchical Storage: ZFS vs. RHEL

ZFS vs. RHEL dm-cache

- ZFS uses ZIL for all write operations (ZFS Intent Log), which can be placed onto a separate logging device such as an SSD
- ZFS read caching implemented as an L2 cache above RAM read cache
- RHEL implements hierarchical storage in device mapper (dm-cache)

RHEL SSD cache administration options

- You can work directly with dm-cache…
- However, it is easier to do within LVM as part of the “lvcreate” command, which will front-end dm-cache for you

Using hierarchical storage in RHEL

```
Create a a 100GB logical volume on an SSD and name it
"cachedisk"
# lvcreate -L 100G -n cachedisk vgXX /dev/<SSD>

Create a 4GB logical volume on an SSD and name it
"metadisk"
# lvcreate -L 4G -n metadisk vgXX /dev/<SSD>

Add the SSD cache and meta volumes to an already existing
magnetic disk (named "magdisk")
# lvconvert --type cache-pool /dev/VGXX/cachedisk \
--poolmetadata /dev/data/metadisk
# lvconvert --type cache /dev/vgXX/magdisk --cachepool \
/dev/data/cachedisk
```

Volume group setup

As with earlier slides, the examples on the left presume a basic volume group setup previously

LVM method

In the first example, we use "lvcreate" with options to include an SSD cache

Cache and meta volumes

Note that the operation requires a cache volume (relatively large) and a meta volume (relatively small). Both should be on fast (SSD) devices

Data Scrubbing

Data Scrubbing in RHEL 8

What is data scrubbing?

- Data scrubbing in this case means scanning a mounted filesystem for bit rot (silent data errors)
- It does **NOT** mean wiping disks for disposal
- Data scrubbing can become more important as the size of mounted filesystems increases; even if a silent data error is a vanishingly small possibility, when you have hundreds of Terabytes to Petabytes of data, it can become a problem.
- Data scrubbing can be needed even if you have other forms of data protections (like mirroring or RAID)

Data scrubbing in RHEL 8 vs. RHEL 7 and prior

- In RHEL 8, data scrubbing is provided by device mapper via **dm-integrity**
- In prior version of RHEL, data scrubbing is not provided

Consider using Ceph for extremely large data

- Data scrubbing is built in to Ceph Bluestore

How does data scrubbing work in RHEL 8?

It is provided by dm-integrity, which is part of dm-crypt (disk encryption)

- The goal of data scrubbing is “bit rot detection.” We want to be warned if a bit or bits have flipped in **data at rest**
- It is achieved by taking a checksum of the data on every write and periodically taking that checksum “at rest” to see if it matches

Pros and cons of data scrubbing in RHEL

- This feature is fully supported in RHEL 8, but **should be approached with caution**
- Data scrubbing imparts overhead on the system:
  - It imposes an overhead on every write (computation of checksum)
  - It consumes extra space for the checksum header
  - It imposes system overhead for periodically comparing checksums
- In RHEL data scrubbing is implemented at the device level, not the filesystem leve. So it can be used with XFS, ext4, etc.

Using data scrubbing in RHEL 8

```
Initial setup of a block storage device with data
scrubbbing
# integritysetup format /dev/vgXX/<LVOL>

Open a data-scrubbing enabled device, assigning it the name
<NAME> for use
# integritysetup open /dev/vgXX/<LVOL> <NAME>

Now create a filesystem on the scrubbing-enabled device
# mkfs.xfs /dev/mapper/<NAME>
```

First, enable a device for data scrubbing

Use the “format” opeiton on inegritysetup to initialize a storage device. This will reserve some space in the header and compute the checksum for the first time.

Open the device to access it as a scrubbed block device

Use the “open” option to assing a previously enabled device to a name from which you can access the now-monitored data.

Deduplication & Compression

What is Virtual Data Optimizer?

Modular Data Reduction for Red Hat Enterprise Linux

- A data reduction module for the Linux device mapper.
- Installs on top of a Linux block device to provide:
  - In-line, on-the-fly deduplication
  - Performance-optimized data compression
  - Thin provisioning and zero block elimination
- Managed via CLI or with Cockpit
- Based on technology assets acquired from Permabit Technology Corporation
- Fully supported feature in Red Hat Enterprise Linux 7.5

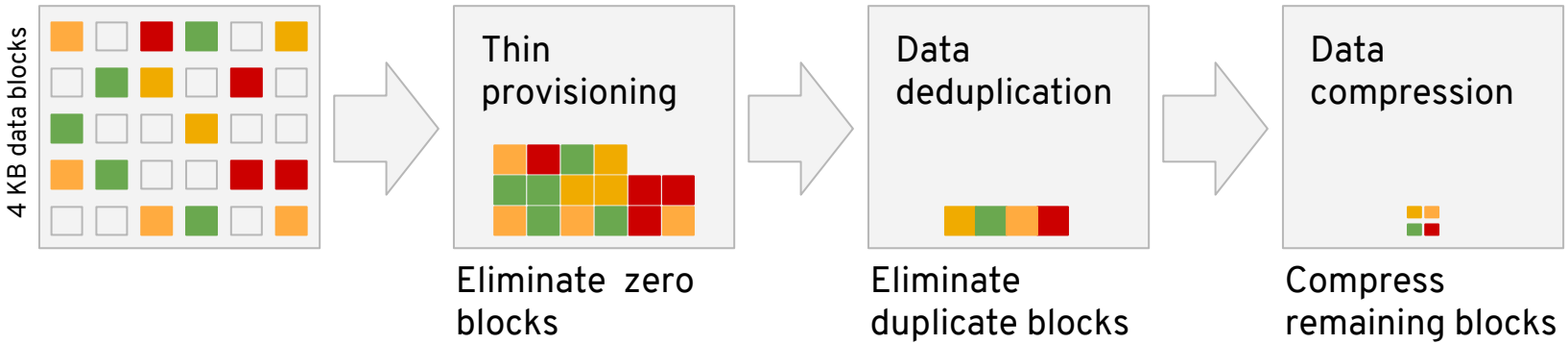
What are the benefits of VDO?

Business impact of reduced data footprint

- Allows repurposing of existing storage resources
- Reduces cost of cloud consumption-based pricing
- Extends the life of storage hardware
- Increases affordability of high performance storage
- Reduces the time it takes to replicate storage devices

How does VDO work?

VDO data reduction processing

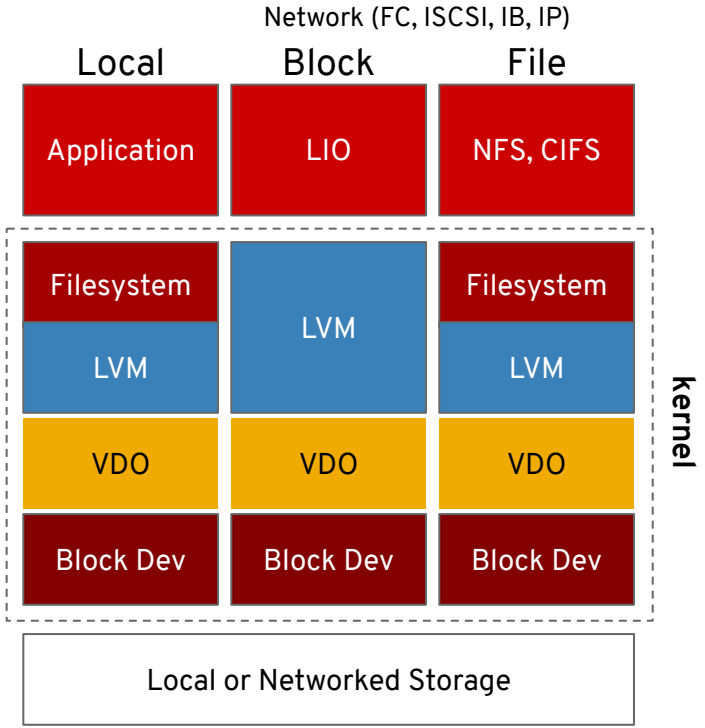


- Operates inline, on-the-fly
- Works at the block level, with the file system of your choice
- Each VDO volume supports up to 256 TB physical and 4 PB logical storage

Where is VDO deployed?

Virtual or physical systems, on-premise or in the cloud

- Hardware agnostic
- Addresses hardware and software approaches to storage
- Works directly with block devices and filesystems
- Functions in cloud or on premises
- Addresses the four storage consumption models



What System Resources does VDO require?

These are minimums

OS Requirements

- RHEL 7.5 or later

RAM

- Base: 500 MB
- Plus: 268 MB per TB of physical storage

CPU

- x86_64 (exploring other processor platforms for future releases)

How is VDO different?

Inline, granularity, scale and performance

- **Deduplication is always done inline, on-the-fly**
  - No guessing about storage consumption
  - No bursts of performance impact from optimization done in the background
- **VDO operates efficiently at 4 KB granularity**
  - A 4 KB change to a 128 KB data chunk will still allow you to save 124 KB (97%)
  - VDO is able keep track of 4 KB blocks with low memory overhead (0.03%*)
- **VDO operates at scale**
  - VDO supports up to 256 TB of physical (after protection) storage
  - VDO presents up to 4 PB of logical storage
- **VDO is designed to perform with random IO workloads**
  - Blocks only get compressed the first time they are seen (fewer CPU cycles)
  - Reading in 4 KB of data doesn't require more than 4 KB of IO

* Memory to physical storage overhead, 10 TB volume

For more information...

Red Hat documentation resources for VDO configuration, tuning and management

The Storage Administration Guide for Red Hat Enterprise Linux includes a section on VDO:

- https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/storage_administration_guide/vdo

Topics addressed include:

- Theoretical Overview of VDO
- System Requirements
- Getting Started with VDO
- Administering VDO
- Deployment Scenarios
- Tuning VDO
- VDO Commands
- Statistics Files in /sys

Unified Configuration

Project Stratis

Integrated volume and file system management

- Is easy to use
- Automates best practices
- Reduces complexity

What is Stratis?

- Stratis is a storage management too, not a filesystem
- Stratis is a local tool, running on each individual server
- Stratis integrates multiple layers of technology which already exist in RHEL
  - LVM
  - Device mapper
  - VDO
- The Stratis daemon manages collections of block devices via:
  - GUI
  - CLI
  - D-bus API

How Does Stratis Operate?

- Stratis manages three types of objects
  - Block devices
  - Pools
  - Filesystems
- The objects are organized according to this hierarchy:
  - Block devices include physical or virtual disks, LVM volumes, multipath devices, etc.
  - Pools are created from one or more block devices
  - Filesystems are created from pools

Major Stratis Features

- Snapshot management
- Thin provisioning
- Caching/hierarchical storage via SSDs
- Compression & deduplication
- Monitoring, notification, and maintenance of:
  - Avoidance of data corruption
  - Switching filesystems to read-only mode
  - Grow filesystems (shrink? - not yet)
  - Fstrim
  - Write throttling
- Future plans:
  - Encryption management
  - Integrity checking/scrubbing
  - Redundancy management

Stratis operations (CLI)

```
Installation
# yum install stratis-cli stratisd

Enabling
# systemctl start stratisd
# systemctl enable --now stratisd

Operations
# stratis pool create demo_pool /dev/vdb /dev/vdc
# stratis pool list
# stratis pool add-data demo_pool /dev/vdd
# stratis filesystem create demo_pool demo_filesystem1
```

One-command installation
"Yum install..."

Operates as a systemd service
Inherits all normal systemd features (journald, cgroups...)

Simple CLI commands
"Do What I Mean"

Scalability

Matching scalability in RHEL

RHEL XFS

- XFS is open-source and non proprietary
- It is the default filesystem starting with RHEL 8

	XFS	ZFS
Max single file size	2^{63} -1 bytes	16 EB (2^{64} bytes)
Max pool size	2^{63} -1 bytes	2^{128} Bytes
Max # files per directory	2^{64}	2^{48}

Data Versioning & Copy-on-Write

Data versioning in RHEL-supported filesystems

No data versioning currently in RHEL 8.3 or earllier

- Consider using snapshots?
- Could be provided by some external tools

Copy-on-Write in RHEL

RHEL partial support of copy-on-write filesystems

- Both RHEL and ZFS use copy-on-write to support snapshots
- RHEL does not support copy-on-write for all filesystem operations
- This feature is the underlying technology that supports data versioning in ZFS

Delegated Permissions

Delegated permissions in RHEL

What are delegated permissions in ZFS?

- Allows a non-root user with permissions on a directory or file to other users
- More fine-grained than just user, group, and world permissions

No delegated permissions on filesystemd currently in RHEL 8.3 or earlier

- But, RHEL does support **ACLs (Access Control Lists)**
- ACLs can be used to effect most of the features of delegated permissions

Questions?

redhat.

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos