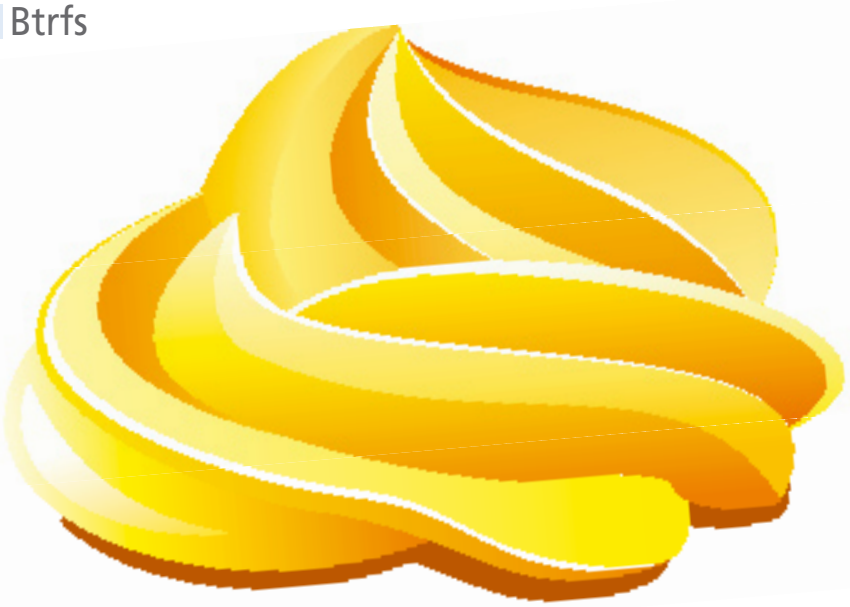

Managing storage volumes with Btrfs

Smooth as Butter

Btrfs is winning over the experts with advanced features like subvolumes, snapshots, and dynamic volume resizing.

By Petros Koutoupis

Btrfs [1] is a next-generation filesystem destined to become the default filesystem of many future Linux distribution releases. Btrfs, initially developed by Oracle, is licensed under the General Public License (GPL) Version 2 and was accepted into the mainline Linux kernel as of 2.6.29-rc1. Btrfs is an acronym for B-tree filesystem, but the word is usually pronounced "Butter FS." The innovative Btrfs filesystem was partly inspired by Sun Microsystem's (now Oracle) ZFS filesystem, which has never officially been ported to the Linux kernel because of licensing conflicts between the Common Development and Distribution License (CDDL) and the GPL.

Btrfs is known as a Copy-On-Write (COW) filesystem. When data is modified, it is never modified in place. A new data block or series of data blocks are allocated to store the new data. This concept helps Btrfs support enhanced features, such as snapshots and subvolumes. A volume can refer to one or multiple grouped disk devices or partitions.

The Btrfs filesystem provides the capabilities for pooling drives together into a single and (if specified) redundant RAID volume; dynamic volume resizing; online defragmentation, checksumming, and compression; and more. Although still feature incomplete, Btrfs is enabled in a default vanilla kernel configuration and is officially supported in various Linux distributions, including Fedora and Ubuntu. If you are using a system that does not offer Btrfs support, you need to rebuild the Linux kernel to enable Btrfs and build the btrfs-progs tool kit to manage it.

In this article, I look at how to create and manage a Btrfs filesystem. Along the way, you'll get a glimpse at some of the advanced features exciting the experts. However, keep in mind that Btrfs is relatively new and still a work in progress. (See the box titled "Errors and fsck.")

The Management Utility

At the time of this writing, Btrfs is not a bootable filesystem; therefore, it cannot host the `/boot` path, where all kernel and boot images reside.

The two main utilities you'll need to create and manage a Btrfs volume are `mkfs.btrfs(8)` and `btrfs(8)`. (Many Btrfs guides also refer to `btrfsctl`, an older tool that has now been replaced by `btrfs`.) The `btrfs` utility is a general-purpose tool you can use to manage Btrfs-enabled volumes to monitor usage, create subvolumes and snapshots, create new volumes, add and remove devices from existing volumes, and enable online defragmentation and volume balancing.

For instance, if you are logging into a Linux system and you want to learn whether the system contains any Btrfs-labeled devices, you could type the following command:

```
$ sudo btrfs device scan
```

To discover all multidevice filesystems on the machine, you will have to execute this command after rebooting or reload-

ERRORS AND FSCK

The Btrfs wiki [1] contains the following warning:

Note that Btrfs does not yet have a fsck tool that can fix errors. While Btrfs is stable on a stable machine, it is currently possible to corrupt the filesystem irrecoverably if your machine crashes or loses power on disks that don't handle flush requests correctly. This will be fixed when the fsck tool is ready.

LISTING 1: Listing Btrfs Filesystems

```
$ sudo btrfs filesystem show
Label: none  uuid: f2346c58-64fd-42a5-afdb-10e9e134d0a1
    Total devices 1 FS bytes used 2.40GB
    devid    1 size 7.64GB used 4.60GB path /dev/sda6
```

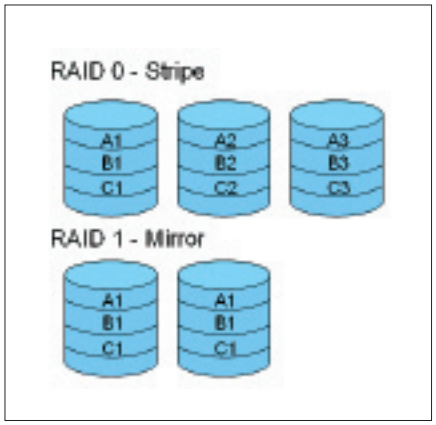

Figure 1: A general layout showing how data is striped across a RAID 0 volume and how data is mirrored across a RAID 1 volume.

ing the `btrfs` module. To list device types (Listing 1), use:

```
$ sudo btrfs filesystem show
```

Creating a New Volume

To use the new Btrfs filesystem, format and label a volume consisting of one or more disk devices or partitions. Creating a new Btrfs-enabled volume is quite simple. To create a single disk volume, type:

```
$ sudo mkfs.btrfs /dev/sdb
```

To stripe across multiple disk devices, type the following command, listing all the disk devices.

```
$ sudo mkfs.btrfs /dev/sdb ➥
    /dev/sdc /dev/sdd
```

(For Btrfs RAID options, see the box titled "Btrfs and RAID.") In Listing 2, `mkfs.btrfs` configures the drives to mirror the metadata across all disk devices and stripe the data across all disk devices. To stripe both the metadata and data (i.e., no mirroring), type:

```
$ sudo mkfs.btrfs -m raid0 ➥
    /dev/sdb /dev/sdc /dev/sdd
```

To mirror both metadata and file data across all attached disk devices, use:

```
$ sudo mkfs.btrfs -m raid10 -d raid10 ➥
    /dev/sdb /dev/sdc /dev/sdd /dev/sde
```

After you create a new Btrfs volume, a new entry is appended to the list of Btrfs filesystems on the machine. To list all Btrfs filesystems and which devices they include, use the `btrfs` command. You will notice the newly created Btrfs volume (Listing 3).

To read and write, you'll need to mount the volume from any of the block device node names specified when the volume was created. Listings 3 and 4 use `/dev/sdb` to signify the entire volume.

Adding and Removing a Device

One of the best features of the Btrfs filesystem is being able to add or remove disk devices dynamically from an existing pool. If you have a failed disk device, or a disk device to use someplace else, remove the device with:

```
$ sudo btrfs device delete /dev/sdd /mnt/
```

When you invoke the `filesystem show` option, the `btrfs` utility will show that a device is missing (Listing 5).

To add a disk device, use:

```
$ sudo btrfs device add /dev/sdd /mnt/
```

Listing 6 shows the updated listing of the filesystem.

At this point, the filesystem includes three devices, but all the metadata and data are still stored on `/dev/sdb` and `/dev/sdc`. Now you need to balance the filesystem and spread the files across all of the devices. The whole concept of balancing re-stripes the allocated extents across all of the existing devices.

```
$ sudo btrfs filesystem balance /mnt/
```

leading to the configuration in Listing 7.

The balancing operation will take some time because it requires that all of the filesystem data and metadata are

LISTING 2: Formatting Multiple Disk Devices

```
$ sudo mkfs.btrfs /dev/sdb /dev/sdc /dev/sdd

WARNING! - Btrfs Btrfs v0.19 IS EXPERIMENTAL
WARNING! - see http://btrfs.wiki.kernel.org before using

adding device /dev/sdc id 2
adding device /dev/sdd id 3
fs created label (null) on /dev/sdb
    nodesize 4096 leafsize 4096 sectorsize 4096 size 6.00GB
```

BTRFS AND RAID

RAID (Redundant Array of Independent Disks) is a method by which multiple independent hard disk drives attached to a computer appear as a single disk. Depending on the RAID type, performance can improve dramatically, especially as you stripe/balance the data across multiple disks, thus removing the bottleneck of a single disk for write/read operations.

Most RAID algorithms offer a form of redundancy that can withstand a finite number of disk failures. When a disk fails in a redundant array, the array will operate in a degraded mode until it is recovered. The technology is capable of rebuilding itself to a point before the failure. Numerous RAID types exist. To date, Btrfs officially supports RAID 0, RAID 1, and RAID 10, and it can also duplicate metadata on a single disk. A patch does exist to add support for RAID 5 and RAID 6, but it hasn't been officially merged into the project.

A RAID 0 array stripes data, interleaving across all drives in the array by writing in a round-robin fashion. Both read and write operations access the data the same way. Writing or reading the data concurrently across several disks takes less time than writing it all to one disk, which means that a RAID 0 system can dramatically improve performance, but RAID 0 offers no form of redundancy. If a hard disk drive in the array were to fail, the entire array fails. In a RAID 1 (disk mirroring) array, one hard disk stores an exact replica of another hard disk. If one drive fails, the second steps in and resumes where the first left off. Overall drive performance is consistent with the performance of a single disk. In some RAID 1 implementations, read performance is tuned to achieve faster speeds through a mechanism known as read-balancing. Read-balancing provides the ability to retrieve data from both disks in the mirror.

The hybrid RAID 10 uses striping of mirrored sets. Figure 1 shows an example of how data is written in chunks across multiple disk devices in a RAID volume.

read and rewritten across the full array, including across the newly added device.

Subvolumes and Volume Snapshots

A single Btrfs volume can contain multiple subvolumes. If you define a subvolume as the "default" for the volume, when it is mounted, the subvolume is presented as root; you can even mount a subvolume when the parent volume is not mounted. Each subvolume can operate as an independent filesystem. In Btrfs, all the storage is in the "pool," and subvolumes are created from the pool – you do not need to partition anything. As long your storage capacity hold out, you can create as many subvolumes as you want. To create a subvolume, type:

```
$ sudo btrfs subvolume create /mnt/subvol
Create subvolume '/mnt/subvol'
```

To verify that the subvolume exists in the root of the Btrfs volume, use the `list` option:

```
$ sudo btrfs subvolume list /mnt
ID 256 top level 5 path subvol
```

Listing 8 shows the contents from the root of the filesystem.

If you want to set a specific subvolume as the default of the volume when it is mounted, you need to obtain the subvolume ID (as seen in the preceding command) and invoke the `btrfs` utility with the following options:

```
$ sudo btrfs subvolume ➥
    set-default 256 /mnt
```

Alternatively, you can mount the subvolume under a separate directory path:

```
$ sudo mount -t btrfs -o subvol=subvol ➥
   /dev/sdb /subvol
```

Although you might think the data snapshot is a subvolume, it differs in implementation. A snapshot is a single state of a storage volume at a particular point in time. Snapshots are usually used for data archiving. Most traditional volume managers require that the snapshot be taken across the entire logical volume. Btrfs, on the other hand, lets you create snapshots on individual files or entire directories anywhere in the Btrfs volume. (Remember, this is all made possible by the COW design of the Btrfs filesystem.) To see how the snapshot process works, start by creating a file with `touch` or `dd` (Listing 9).

Now that you have a file, you can create a snapshot of the filesystem root:

```
$ sudo btrfs subvolume snapshot ➥
    /mnt /mnt/snapshot_of_root
```

A listing of all subvolumes for the Btrfs filesystem will now have the snapshot appended to it:

```
$ sudo btrfs subvolume list /mnt
ID 256 top level 5 path subvol
ID 257 top level 5 path snapshot_of_root
```

A listing of the contents of the newly created snapshot will display an image of what the root of the Btrfs volume looked like when the snapshot was taken, with the exception of the `snapshot subvolume` directory (Listing 10).

Now create a new file at the root of the volume. The snapshot will still preserve the original data at the time it was taken (see the comparison in Listing 11).

To mount the newly created snapshot to a separate directory path, use:

```
$ sudo mkdir /btrfs_snapshot
$ sudo mount -t btrfs -o subvol=➥
    snapshot_of_root ➥
    /dev/sdb /btrfs_snapshot/
```

Full backups of an entire volume can take a long time and use large amounts of storage space, even for files that remain unchanged. Also, when performing a data backup of entire volumes or subsets of volumes in a symmetric multiprocessing environment, write operations might continue to modify the file data on that volume, with the possibility of data corruption. Several strategies provide some protection. For instance, the volume can be taken offline or marked as read-only before the archival process, but in high-availability production environments, this approach might not be practical.

A snapshot provides a more complete solution. You can use a snapshot to avoid downtime and retain atomicity when archiving entire files, directories, or filesystems. In a production environment, it is not uncommon for a system

LISTING 3: A New Btrfs Filesystem

```
$ sudo btrfs filesystem show
Label: none  uuid: f2346c58-64fd-42a5-afdb-
10e9e134d0a1
    Total devices 1 FS bytes used 2.40GB
    devid    1 size 7.64GB used 4.60GB path /dev/sda6

Label: none  uuid: 0fa5bbee-6f69-4d10-a316-ac373e8b5f64
    Total devices 3 FS bytes used 28.00KB
    devid    1 size 2.00GB used 531.94MB path /dev/sdb
    devid    2 size 2.00GB used 212.75MB path /dev/sdc
    devid    3 size 2.00GB used 519.94MB path /dev/sdd
```

LISTING 4: Mounting and Verifying

```
$ sudo mount /dev/sdb /mnt/
$ df -t btrfs
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/sda6              8011776   2745480   5266296  35% /
/dev/sdb               6291456        56   6291400   1% /mnt
```

LISTING 5: After Removing a Device

```
$ sudo btrfs filesystem show /dev/sdb

Label: none  uuid: 0fa5bbee-6f69-4d10-a316-ac373e8b5f64
    Total devices 3 FS bytes used 36.00KB
    devid    1 size 2.00GB used 156.00MB path /dev/sdb
    devid    2 size 2.00GB used 136.00MB path /dev/sdc
    *** Some devices missing
```

administrator to create a scheduled `cron` job to create hourly, daily, weekly, or monthly snapshots of various files and directories, including the end user's home directory. That way, the admin can retrieve a recently modified or deleted file with few or no headaches for all parties involved.

To delete a subvolume, type:

```
$ sudo btrfs subvolume delete /mnt/subvol
```

When listing all subvolumes of the parent volume, you will notice that the recently deleted subvolume will not be listed:

```
$ sudo btrfs subvolume list /mnt
ID 257 top level 5 path snapshot_of_root
```

To delete a snapshot subvolume use:

```
$ sudo btrfs subvolume delete ➥
    /mnt/snapshot_of_root/
```

Resizing a Volume

Unlike many filesystem alternatives, Btrfs lets you dynamically resize an existing volume. This feature is still somewhat limited and needs more time to mature, but it is still useful for some environments. If you want to decrease the size of a volume from 6 to 5GB, type:

```
$ sudo btrfs filesystem resize -1G /mnt/
Resize '/mnt/' of '-1G'
```

Now the volume is 5GB in size (see Listing 12). To return the filesystem to 6GB, type:

```
$ sudo btrfs filesystem resize +1G /mnt/
Resize '/mnt/' of '+1G'
```

If you want to use the maximum size of the volume, type:

```
$ sudo btrfs filesystem resize max /mnt
```

Listing 13 shows the `df` output after resizing back to 6GB (the volume's maximum capacity).

The current limitation, I find, is that the Btrfs module won't let you resize a subvolume without affecting the size of the parent volume. For instance, if the parent volume is mounted at `/mnt` and the subvolume is mounted at `/subvol`, if I attempt to decrease the size of the subvolume to leave more space to create a new subvolume in the future, the parent volume also decreases in total size (Listing 14). Btrfs doesn't let me keep the parent volume constant and liberate space by shrinking a subvolume. I imagine this problem will be addressed in the near future as Btrfs matures.

Commands and Configurations

For man pages for both the `mkfs.btrfs` and `btrfs` utilities, at the command line enter:

```
$ man 8 btrfs
```

Although I mainly relied on the `df` command to monitor the filesystem's overall size and usage, the `btrfs` tool has its own built-in version that conveniently displays more detailed usage of the desired filesystem.

```
$ btrfs filesystem df /
Data: total=3.06GB, used=2.24GB
Metadata: total=783.19MB, used=220.93MB
System: total=12.00MB, used=4.00KB
```

LISTING 6: After Adding a Device

```
$ sudo btrfs filesystem show /dev/sdb
Label: none  uuid: 0fa5bbee-6f69-4d10-a316-ac373e8b5f64
    Total devices 3 FS bytes used 36.00KB
    devid    1 size 2.00GB used 156.00MB path /dev/sdb
    devid    2 size 2.00GB used 136.00MB path /dev/sdc
    devid    3 size 2.00GB used 0.00 path /dev/sdd
```

LISTING 7: After Rebalancing

```
$ sudo btrfs filesystem show /dev/sdb
Label: none  uuid: 0fa5bbee-6f69-4d10-a316-ac373e8b5f64
    Total devices 3 FS bytes used 36.00KB
    devid    1 size 2.00GB used 352.75MB path /dev/sdb
    devid    2 size 2.00GB used 204.75MB path /dev/sdc
    devid    3 size 2.00GB used 340.75MB path /dev/sdd
```

LISTING 8: Listing the Contents

```
$ ls -l /mnt/
total 0
drwx------ 1 root root 0 2010-12-29 15:51 subvol
```

LISTING 9: Creating a Test File

```
$ sudo dd if=/dev/zero of=/mnt/test.dat count=4
4+0 records in
4+0 records out
2048 bytes (2.0 kB) copied, 0.000383095 s, 5.3 MB/s
$ ls /mnt/
subvol  test.dat
```

LISTING 10: Snapshot Comparison

```
$ ls -l /mnt/snapshot_of_root/
total 2
drwxr-xr-x 1 root root    0 2010-12-29 15:57 subvol
-rw-r--r-- 1 root root 2048 2010-12-29 15:54 test.dat
$ ls -l /mnt/
total 6
dr-xr-xr-x 1 root root   60 2010-12-29 15:54 snapshot_of_root
drwx------ 1 root root    0 2010-12-29 15:51 subvol
-rw-r--r-- 1 root root 2048 2010-12-29 15:54 test.dat
```

```
$ btrfs filesystem df /mnt/
Data: total=614.25MB, used=0.00
Metadata: total=128.00MB, used=32.00KB
System: total=12.00MB, used=4.00KB
```

As mentioned earlier, Btrfs supports online defragmentation. (Note that defragging takes longer on larger and more data-occupied volumes.) To defragment a volume, type:

```
$ sudo btrfs filesystem defragment /mnt/
```

As with any other Linux filesystems, Btrfs also supports mounting a Btrfs volume with one or more mount options. For instance, to disable checksumming CRC-32C while enabling zlib compression, you need to type:

```
$ sudo mount -t btrfs -o ➥
    nodatasum,compress /dev/sdb /mnt/
```

Btrfs also contains optimizations for the popular Flash-based Solid State Disk (SSD). To enable the SSD optimizations, mount with the `-o ssd` option.

Note that, as of v2.6.31-rc1, this mount option will be enabled if Btrfs is able to detect non-rotating storage. The SSD is going to play a large part in the future of data storage, and it is nice to observe that the Btrfs developers have been preparing for its arrival. See the Btrfs wiki page for a full listing of mount options [2].

One last thing worth mentioning is that, if you want to add an entry into the `/etc/fstab` file to mount the Btrfs volume at system bootup, you can do so by entering a new line specifying one of the devices, the mount point, the filesystem, and a list of all the devices and options:

```
/dev/sdb     /mnt     btrfs ➥
   device=/dev/sdb,device=/dev/sdc,➥
   device=/dev/sdd,➥
   device=/dev/sde     0     0
```

Or, you can use the UUID (as it is displayed in the output of the `btrfs filesystem show` command) followed by the traditional `fstab` field entries (Listing 15).

LISTING 11: Updated Comparison

```
$ ls -l /mnt/
total 10
dr-xr-xr-x 1 root root   60 2010-12-29 15:54 snapshot_of_root
drwx------ 1 root root    0 2010-12-29 15:51 subvol
-rw-r--r-- 1 root root 2048 2010-12-29 15:59 test_again.dat
-rw-r--r-- 1 root root 2048 2010-12-29 15:54 test.dat
$ ls -l /mnt/snapshot_of_root/
total 2
drwxr-xr-x 1 root root    0 2010-12-29 15:57 subvol
-rw-r--r-- 1 root root 2048 2010-12-29 15:54 test.dat
```

LISTING 12: df Output

```
$ df -t btrfs
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/sda6              8011776   2804452   5207324  36% /
/dev/sdb               5242880        72   5242808   1% /mnt
```

LISTING 13: After Resizing

```
$ df -t btrfs
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/sda6              8011776   2804456   5207320  36% /
/dev/sdb               6291456        72   6291384   1% /mnt
```

LISTING 14: Resizing a Subvolume

```
$ df -t btrfs
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/sda6              8011776   2804768   5207008  36% /
/dev/sdb               8388608        80   8388528   1% /mnt
/dev/sdc               8388608        80   8388528   1% /subvol
$ sudo btrfs filesystem resize -1G /mnt/subvol/
Resize '/mnt/subvol/' of '-1G'
$ df -t btrfs
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/sda6              8011776   2804768   5207008  36% /
/dev/sdb               7340032        80   7339952   1% /mnt
/dev/sdc               7340032        80   7339952   1% /subvol
```

LISTING 15: UUID and fstab Field Entries

```
01 UUID-0fa5bbee-6f69-4d10-a316-ac373e8b5f64    /mnt    btrfs    defaults    0    0
```

Summary

The next-generation Linux filesystem is full of great features and functionality if you take the time to get used to the tools. Although Btrfs is still feature incomplete, the filesystem provides amazing flexibility. As Btrfs evolves [3], you can expect to find official support for RAID 5 and RAID 6, online filesystem integrity checks, and support for data deduplication (an ideal solution in virtualization environments). ■■■

INFO

[1] Btrfs wiki: *https://btrfs.wiki.kernel.org/*

[2] Btrfs mount options: *https://btrfs.wiki.kernel.org/index.php/Getting_started#Mount_Options*

[3] Wikipedia on Btrfs: *http://en.wikipedia.org/wiki/Btrfs*

AUTHOR

Petros Koutoupis is a full-time Linux kernel, device driver, and application developer for embedded and server platforms. He has worked in the data storage industry for more than six years and enjoys discussing storage technologies.