The sys admin's daily grind: Converting to Btrfs

Btr's Better

Btrfs might not be complete, but conceptually, it combines so many useful filesystem functions that Charly couldn't wait to play around with it. By Charly Kühnast

My desktop machine uses an old ext4-formatted hard disk, which Linux mounts as `/opt/data`. This is where I build and try out software. Every few months, when the disk is full, I dump the content of `/opt/data` into the trash. Because I never store any important data here, I would survive the long-anticipated demise of this ancient hard disk without shedding a tear. Today, I'm going to try out the Btrfs [1] [2] filesystem conversion on this disk.

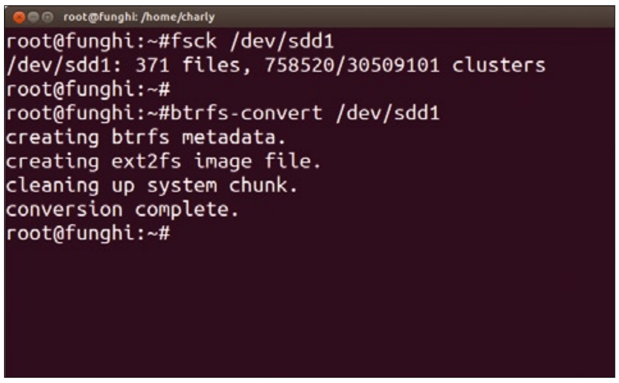
The copy-on-write filesystem created as a prototype in 2007 by ReiserFS and Oracle guru Chris Mason autonomously creates checksums, compresses data, and creates snapshots, while also using dynamic inodes and providing highly efficient storage for small files, which will probably help it heal Linux's open wounds from ZFS someday.

But back to my legacy disk: I first need to unmount the volume and run a filesystem check to make sure the filesystem is in good working order:

```
umount /dev/sdd1
fsck /dev/sdd1
```

The conversion itself (Figure 1) is handled by the following command:

```
btrfs-convert /dev/sdd1
```



```
root@funghi:~#fsck /dev/sdd1
/dev/sdd1: 371 files, 758520/30509101 clusters
root@funghi:~#
root@funghi:~#btrfs-convert /dev/sdd1
creating btrfs metadata.
creating ext2fs image file.
cleaning up system chunk.
conversion complete.
root@funghi:~#
```

Figure 1: Converting an ext2 filesystem to Btrfs is a surprisingly fast process.

I had planned a mandatory coffee break at this point, but the conversion tool came right back with the results. It seems as though the converter doesn't touch the data blocks at all; rather, it just rewrites the Btrfs metadata. It also creates a new subvolume named `ext2_saved`.

Put simply, a subvolume is a directory within the new filesystem that acts like a separate volume. Every Btrfs-formatted partition contains the `default` subvolume and can contain (almost) an arbitrary number of other subvolumes. The `ext2_saved` subvolume created by the converter contains an image (snapshot) of the previous ext volume. It doesn't take much in the line of storage space on the disk originally because it only stores the data changed since it was created – in other words, the more I mess around, the bigger the snapshot will become.

Rollback Command

If I'm unhappy with my Btrfs, I can undo the conversion by typing:

```
btrfs-convert -r /dev/sdd1
```

This command would restore my previous ext filesystem from the snapshot but would lose the changes since the snapshot. Instead, I want to try out my new Btrfs and proceed to mount it by typing:

```
mount -o compress /dev/sdd1 /opt/data
```

The `-o compress` mount option enables the native Btrfs compression function, which is disabled by default. From now on, it will be enabled on write access, although Btrfs will not touch the existing data for the time being. Compression doesn't cause any noticeable CPU load. The engine is clever enough to identify compressed files such as JPEG and MP3 and doesn't waste any time trying to compress them again.

Although it might take a while for Btrfs to make the version jump to 1.0 and permanently replace my venerable ext filesystems, my experiments clearly show that this is a date to await eagerly. ■■■

INFO

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

[2] "Btrfs" by Petros Koutoupis, *Linux Magazine*, March 2011, pg. 32

AUTHOR

Charly Kühnast is a Unix operating system administrator at the Data Center in Moers, Germany. His tasks include firewall and DMZ security and availability. He divides his leisure time into hot, wet, and eastern sectors, where he enjoys cooking, freshwater aquariums, and learning Japanese, respectively.