Using Btrfs with Multiple Devices

From btrfs Wiki

Contents

Multiple devices

A Btrfs filesystem can be created on top of many devices, and more devices can be added after the FS has been created.

By default, metadata will be mirrored across two devices and data will be striped across all of the devices present. This is equivalent to mkfs.btrfs -m raid1 -d raid0.

If only one device is present, metadata will be duplicated on that one device. For HDD mkfs.btrfs -m dup -d single, for SSD (or non-rotational device) mkfs.btrfs -m single -d single.

Current status

Btrfs can add and remove devices online, and freely convert between RAID levels after the FS has been created.

Btrfs supports RAID 0, RAID 1, RAID 10, RAID 5 and RAID 6 (but see the section below about RAID 5/6), and it can also duplicate metadata or data on a single spindle or multiple disks. When blocks are read in, checksums are verified. If there are any errors, Btrfs tries to read from an alternate copy and will repair the broken copy if the alternative copy succeeds.

See the Gotchas page for some current issues when using btrfs with multiple volumes of differing sizes in a RAID 1 style setup.

The RAID 1 profile also allows for 2, 3, or 4 copies of redundant data copies, called RAID 1, RAID 1C3, and RAID 1C4 respectively. See Manpage/mkfs.btrfs for more details on this and other RAID profiles.

RAID 5 and RAID 6

Please read the parity RAID status page first: RAID56.

Note that the minimum number of devices required for RAID 5 is 3. For a RAID 6, the minimum is 4 devices.

Filesystem creation

mkfs.btrfs will accept more than one device on the command line. It has options to control the RAID configuration for data (-d) and metadata (-m). Valid choices are raid0, raid1, raid10, raid5, raid6, single and dup. The option -m single means that no duplication is done, which may be desired when using hardware RAID.

RAID 10 requires at least 4 devices.

```
# Create a filesystem across four drives (metadata mirrored, linear data allocation)
mkfs.btrfs -d single /dev/sdb /dev/sdc /dev/sdd /dev/sde

# Stripe the data without mirroring, metadata are mirrored
mkfs.btrfs -d raid0 /dev/sdb /dev/sdc

# Use raid10 for both data and metadata
mkfs.btrfs -m raid10 -d raid10 /dev/sdb /dev/sdc /dev/sdd /dev/sde

# Don't duplicate metadata on a single drive (default on single SSDs)
mkfs.btrfs -m single /dev/sdb
```

If you want to use devices of different sizes, striped RAID levels (RAID 0, RAID 10, RAID 5, RAID 6) may not use all of the available space on the devices. (See btrfs-space-calculator (https://www.mankier.com/1/btrfs-space-calculator) or btrfs disk usage calculator (https://carfax.org.uk/btrfs-usage/)) Non-striped equivalents

may give you a more effective use of space (single instead of RAID 0, RAID 1 instead of RAID 10).

```
# Use full capacity of multiple drives with different sizes (metadata mirrored, data not mirrored and not striped)
mkfs.btrfs -d single /dev/sdb /dev/sdc
```

Once you create a multi-device filesystem, you can use any device in the FS for the mount command:

```
mkfs.btrfs /dev/sdb /dev/sdc /dev/sde
mount /dev/sde /mnt
```

If you want to mount a multi-device filesystem using a loopback device, it's not sufficient to use mount -o loop. Instead, you'll have to set up the loopbacks manually:

```
# Create and mount a filesystem made of several disk images
mkfs.btrfs img0 img1 img2
losetup /dev/loop0 img0
losetup /dev/loop1 img1
losetup /dev/loop2 img2
mount /dev/loop0 /mnt/btrfs
```

After a reboot or reloading the btrfs module, you'll need to use btrfs device scan to discover all multi-device filesystems on the machine (see below)

The UseCases page gives a few quick recipes for filesystem creation.

Device scanning

btrfs device scan is used to scan all of the block devices under /dev and probe for Btrfs volumes. This is required after loading the btrfs module if you're running with more than one device in a filesystem.

```
# Scan all devices
btrfs device scan

# Scan a single device
btrfs device scan /dev/sdb
```

btrfs filesystem show will print information about all of the btrfs filesystems on the machine.

Adding new devices

btrfs filesystem show gives you a list of all the btrfs filesystems on the systems and which devices they include.

btrfs device add is used to add new devices to a mounted filesystem.

btrfs balance can balance (restripe) the allocated extents across all of the existing devices. For example, with an existing filesystem mounted at /mnt, you can add the device /dev/sdc to it with:

```
btrfs device add /dev/sdc /mnt
```

At this point we have added our device to the filesystem, but all of the metadata and data are still stored on the original device(s). The filesystem can be balanced to spread the extents across all the devices.

```
btrfs balance start /mnt
```

The btrfs balance operation will take some time. It reads in all of the FS data and metadata and rewrites it across all the available devices. Depending on the usage scenario that is not needed, one example would be adding a new device to a RAID 1 filesystem whose existing devices still have enough space left. However, in other cases a balancing might be needed. Take a closer look at available filter options that can help reducing the needed time and scope of the balance.

Conversion

A non-RAID filesystem is converted to RAID by adding a device and running a balance filter that will change the chunk allocation profile.

For example, to convert an existing single device system (/dev/sdb1) into a 2 device RAID 1 (to protect against a single disk failure):

```
mount /dev/sdb1 /mnt
btrfs device add /dev/sdc1 /mnt
btrfs balance start -dconvert=raid1 -mconvert=raid1 /mnt
```

If the metadata is not converted from the single-device default, it remains as DUP, which does not guarantee that copies of block are on separate devices. If data is

not converted it does not have any redundant copies at all.

Removing devices

btrfs device delete is used to remove devices online. It redistributes any extents in use on the device being removed to the other devices in the filesystem. Example:

```
mkfs.btrfs /dev/sdb /dev/sdc /dev/sdd /dev/sde
mount /dev/sdb /mnt
# Put some data on the filesystem here
btrfs device delete /dev/sdc /mnt
```

Replacing failed devices

Using btrfs replace

When you have a device that's in the process of failing or has failed in a RAID array you should use the btrfs replace command rather than adding a new device and removing the failed one. This is a newer technique that worked for me when adding and deleting devices didn't however it may be helpful to consult the mailing list of irc channel before attempting recovery.

First list the devices in the filesystem, in this example we have one missing device (5 devices out of 6 are shown). The devids in the example are not sequential since add and remove have been used previously instead of replace.

```
user@host:~$ sudo btrfs filesystem show
Label: none  uuid: 67b4821f-16e0-436d-b521-e4ab2c7d3ab7
  Total devices 6 FS bytes used 5.47TiB
  devid    1 size 1.81TiB used 1.71TiB path /dev/sda3
  devid    3 size 1.81TiB used 1.71TiB path /dev/sdb3
  devid    4 size 1.82TiB used 1.72TiB path /dev/sdc1
  devid    6 size 1.82TiB used 1.72TiB path /dev/sdd1
  devid    8 size 2.73TiB used 2.62TiB path /dev/sde1
  *** Some devices missing
```

Before replacing the device you will need to mount the array, if you have a missing device then you will need to use the following command:

```
sudo mount -o degraded /dev/sda3 /mnt
```

Now replace the absent device with the new drive /dev/sdf1 on the filesystem currently mounted on /mnt (since the device is absent, you can use any devid number that isn't present; 2,5,7,9 would all work the same):

```
sudo btrfs replace start 7 /dev/sdf1 /mnt
```

This will start the replacement process in the background, you can monitor the status using the btrfs replace status command. The status command updates the status continuously, you can ctrl+c to exit it at any time. At first progress will be shown, on completing you'll see the following output:

```
user@host:~$ sudo btrfs replace status /mnt
Started on 27.Mar 22:34:20, finished on 28.Mar 06:36:15, 0 write errs, 0 uncorr. read errs
```

Caveats

- If replacing an existing device with a smaller device, you should use btrfs fi resize first. If replacing a device with a larger device, you should use resize after.
- The error counters appear to be inaccurate, I had errors in my recovery reported in dmesg but not in the replace status output. The following command will parse out all the names of damaged files currently in the dmesg log, it only works on the messages currently in the kernel log ring buffer so if you have received too many log messages it will not be complete:

```
dmesg | grep BTRFS | grep path | sed -e 's/^.*path: //;s/)$//' | sort | uniq
```

Using add and delete

The example above can be used to remove a failed device if the super block can still be read. But, if a device is missing or the super block has been corrupted, the filesystem will need to be mounted in degraded mode:

```
mkfs.btrfs -m raid1 /dev/sdb /dev/sdc /dev/sdd /dev/sde
#
# sdd is destroyed or removed, use -o degraded to force the mount
```

```
# to ignore missing devices
#
mount -o degraded /dev/sdb /mnt
#
# 'missing' is a special device name
#
btrfs device delete missing /mnt
```

btrfs device delete missing tells btrfs to remove the first device that is described by the filesystem metadata but not present when the FS was mounted.

In case of *RAID XX* layout, you cannot go below the minimum number of the device required. So before removing a device (even the *missing* one) you may need to add a new one. For example if you have a RAID 1 layout with **two** devices, and a device fails, you must:

- mount in degraded mode
- add a new device
- remove the *missing* device

Registration in /etc/fstab

If you don't have an initrd, or your initrd doesn't perform a btrfs device scan, you can still mount a multi-volume btrfs filesystem by passing *all* the devices in the filesystem explicitly to the mount command. A suitable /etc/fstab entry would be:

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

> Using device is **not** recommended, as it is sensitive to device names changing that is affected by the order of discovery of the devices. You should *really* be using a initramfs. Most modern distributions will do this for you automatically if you install their own btrfs-progs package.

Using device=PARTUUID=... with GPT partition UUIDs would also work and be less fragile. All these options can also be set from the kernel command line (https://www.kernel.org/doc/Documentation/kernel-parameters.txt) , through root=/fstype=/rootflags=.