NAME

btrfs-device − manage devices of btrfs filesystems

SYNOPSIS

btrfs device <*subcommand*> <*args*>

DESCRIPTION

The **btrfs device** command group is used to manage devices of the btrfs filesystems.

DEVICE MANAGEMENT

Btrfs filesystem can be created on top of single or multiple block devices. Data and metadata are organized in allocation profiles with various redundancy policies. There's some similarity with traditional RAID levels, but this could be confusing to users familiar with the traditional meaning. Due to the similarity, the RAID terminology is widely used in the documentation. See **mkfs.btrfs**(8) for more details and the exact profile capabilities and constraints.

The device management works on a mounted filesystem. Devices can be added, removed or replaced, by commands provided by **btrfs device** and **btrfs replace**.

The profiles can be also changed, provided there's enough workspace to do the conversion, using the **btrfs balance** command and namely the filter *convert*.

Type
: The block group profile type is the main distinction of the information stored on the block device. User data are called *Data*, the internal data structures managed by filesystem are *Metadata* and *System*.

Profile
: A profile describes an allocation policy based on the redundancy/replication constraints in connection with the number of devices. The profile applies to data and metadata block groups separately. Eg. *single*, *RAID1*.

RAID level
: Where applicable, the level refers to a profile that matches constraints of the standard RAID levels. At the moment the supported ones are: RAID0, RAID1, RAID10, RAID5 and RAID6.

See the section **TYPICAL USECASES** for some examples.

SUBCOMMAND

add [−Kf] <*device*> [<*device*>*...*] <*path*>

Add device(s) to the filesystem identified by <*path*>.

If applicable, a whole device discard (TRIM) operation is performed prior to adding the device. A device with existing filesystem detected by **blkid**(8) will prevent device addition and has to be forced. Alternatively the filesystem can be wiped from the device using eg. the **wipefs**(8) tool.

The operation is instant and does not affect existing data. The operation merely adds the device to the filesystem structures and creates some block groups headers.

Options

−K|−−nodiscard
: do not perform discard (TRIM) by default

−f|−−force
: force overwrite of existing filesystem on the given disk(s)

−−enqueue
: wait if there's another exclusive operation running, otherwise continue

remove [options] <*device*>|<*devid*> [<*device*>|<*devid*>*...*] <*path*>

Remove device(s) from a filesystem identified by <*path*>

Device removal must satisfy the profile constraints, otherwise the command fails. The filesystem must be converted to profile(s) that would allow the removal. This can typically happen when going down from 2 devices to 1 and using the RAID1 profile. See the **TYPICAL USECASES** section below.

The operation can take long as it needs to move all data from the device.

It is possible to delete the device that was used to mount the filesystem. The device entry in the mount table will be replaced by another device name with the lowest device id.

If the filesystem is mounted in degraded mode (−o degraded), special term *missing* can be used for *device*. In that case, the first device that is described by the filesystem metadata, but not present at the mount time will be removed.

Note

In most cases, there is only one missing device in degraded mode, otherwise mount fails. If there are two or more devices missing (e.g. possible in RAID6), you need specify *missing* as many times as the number of missing devices to remove all of them.

Options

−−enqueue
: wait if there's another exclusive operation running, otherwise continue

delete *<device>*|*<devid>* [*<device>*|*<devid>...*] *<path>*
: Alias of remove kept for backward compatibility

ready *<device>*
: Wait until all devices of a multiple−device filesystem are scanned and registered within the kernel module. This is to provide a way for automatic filesystem mounting tools to wait before the mount can start. The device scan is only one of the preconditions and the mount can fail for other reasons. Normal users usually do not need this command and may safely ignore it.

scan [options] [*<device>* [*<device>...*]]
: Scan devices for a btrfs filesystem and register them with the kernel module. This allows mounting multiple−device filesystem by specifying just one from the whole group.

If no devices are passed, all block devices that blkid reports to contain btrfs are scanned.

The options *−−all−devices* or *−d* can be used as a fallback in case blkid is not available. If used, behavior is the same as if no devices are passed.

The command can be run repeatedly. Devices that have been already registered remain as such. Reloading the kernel module will drop this information. There's an alternative way of mounting multiple−device filesystem without the need for prior scanning. See the mount option *device*.

Options

−d|−−all−devices
: Enumerate and register all devices, use as a fallback in case blkid is not available.

−u|−−forget
: Unregister a given device or all stale devices if no path is given, the device must be unmounted otherwise it's an error.

stats [options] *<path>*|*<device>*
: Read and print the device IO error statistics for all devices of the given filesystem identified by *<path>* or for a single *<device>*. The filesystem must be mounted. See section **DEVICE STATS** for more information about the reported statistics and the meaning.

Options

−z|−−reset
: Print the stats and reset the values to zero afterwards.

−c|−−check
: Check if the stats are all zeros and return 0 if it is so. Set bit 6 of the return code if any of the statistics is no−zero. The error values is 65 if reading stats from at least one device failed, otherwise it’s 64.

usage [options] <*path*> [<*path*>...]

Show detailed information about internal allocations on devices.

The level of detail can differ if the command is run under a regular or the root user (due to use of restricted ioctls). The first example below is for normal user (warning included) and the next one with root on the same filesystem:

```
WARNING: cannot read detailed chunk info, per−device usage will not be shown, run as root
/dev/sdc1, ID: 1
   Device size:           931.51GiB
   Device slack:              0.00B
   Unallocated:           931.51GiB

/dev/sdc1, ID: 1
   Device size:           931.51GiB
   Device slack:              0.00B
   Data,single:           641.00GiB
   Data,RAID0/3:            1.00GiB
   Metadata,single:        19.00GiB
   System,single:          32.00MiB
   Unallocated:           271.48GiB
```

- *Device size* — size of the device as seen by the filesystem (may be different than actual device size)
- *Device slack* — portion of device not used by the filesystem but still available in the physical space provided by the device, eg. after a device shrink
- *Data,single*, *Metadata,single*, *System,single* — in general, list of block group type (Data, Metadata, System) and profile (single, RAID1, ...) allocated on the device
- *Data,RAID0/3* — in particular, striped profiles RAID0/RAID10/RAID5/RAID6 with the number of devices on which the stripes are allocated, multiple occurrences of the same profile can appear in case a new device has been added and all new available stripes have been used for writes
- *Unallocated* — remaining space that the filesystem can still use for new block groups

Options

−b|−−raw
: raw numbers in bytes, without the *B* suffix

−h|−−human−readable
: print human friendly numbers, base 1024, this is the default

−H
: print human friendly numbers, base 1000

−−iec
: select the 1024 base for the following options, according to the IEC standard

−−si
: select the 1000 base for the following options, according to the SI standard

−k|−−kbytes
: show sizes in KiB, or kB with −−si

−m|−−mbytes
: show sizes in MiB, or MB with −−si

−g|−−gbytes
: show sizes in GiB, or GB with −−si

−t|−−tbytes
: show sizes in TiB, or TB with −−si

If conflicting options are passed, the last one takes precedence.

TYPICAL USECASES

STARTING WITH A SINGLE−DEVICE FILESYSTEM

Assume we've created a filesystem on a block device */dev/sda* with profile *single/single* (data/metadata), the device size is 50GiB and we've used the whole device for the filesystem. The mount point is */mnt*.

The amount of data stored is 16GiB, metadata have allocated 2GiB.

ADD NEW DEVICE

We want to increase the total size of the filesystem and keep the profiles. The size of the new device */dev/sdb* is 100GiB.

```
$ btrfs device add /dev/sdb /mnt
```

The amount of free data space increases by less than 100GiB, some space is allocated for metadata.

CONVERT TO RAID1

Now we want to increase the redundancy level of both data and metadata, but we'll do that in steps. Note, that the device sizes are not equal and we'll use that to show the capabilities of split data/metadata and independent profiles.

The constraint for RAID1 gives us at most 50GiB of usable space and exactly 2 copies will be stored on the devices.

First we'll convert the metadata. As the metadata occupy less than 50GiB and there's enough workspace for the conversion process, we can do:

```
$ btrfs balance start −mconvert=raid1 /mnt
```

This operation can take a while, because all metadata have to be moved and all block pointers updated. Depending on the physical locations of the old and new blocks, the disk seeking is the key factor affecting performance.

You'll note that the system block group has been also converted to RAID1, this normally happens as the system block group also holds metadata (the physical to logical mappings).

What changed:

- available data space decreased by 3GiB, usable roughly (50 − 3) + (100 − 3) = 144 GiB
- metadata redundancy increased

IOW, the unequal device sizes allow for combined space for data yet improved redundancy for metadata. If we decide to increase redundancy of data as well, we're going to lose 50GiB of the second device for obvious reasons.

```
$ btrfs balance start −dconvert=raid1 /mnt
```

The balance process needs some workspace (ie. a free device space without any data or metadata block groups) so the command could fail if there’s too much data or the block groups occupy the whole first device.

The device size of */dev/sdb* as seen by the filesystem remains unchanged, but the logical space from 50−100GiB will be unused.

REMOVE DEVICE

Device removal must satisfy the profile constraints, otherwise the command fails. For example:

```
$ btrfs device remove /dev/sda /mnt
ERROR: error removing device '/dev/sda': unable to go below two devices on raid1
```

In order to remove a device, you need to convert the profile in this case:

```
$ btrfs balance start −mconvert=dup −dconvert=single /mnt
$ btrfs device remove /dev/sda /mnt
```

DEVICE STATS

The device stats keep persistent record of several error classes related to doing IO. The current values are printed at mount time and updated during filesystem lifetime or from a scrub run.

```
$ btrfs device stats /dev/sda3
[/dev/sda3].write_io_errs   0
[/dev/sda3].read_io_errs    0
[/dev/sda3].flush_io_errs   0
[/dev/sda3].corruption_errs 0
[/dev/sda3].generation_errs 0
```

write_io_errs
: Failed writes to the block devices, means that the layers beneath the filesystem were not able to satisfy the write request.

read_io_errors
: Read request analogy to write_io_errs.

flush_io_errs
: Number of failed writes with the *FLUSH* flag set. The flushing is a method of forcing a particular order between write requests and is crucial for implementing crash consistency. In case of btrfs, all the metadata blocks must be permanently stored on the block device before the superblock is written.

corruption_errs
: A block checksum mismatched or a corrupted metadata header was found.

generation_errs
: The block generation does not match the expected value (eg. stored in the parent node).

Since kernel 5.14 the device stats are also available in textual form in */sys/fs/btrfs/FSID/devinfo/DEVID/error_stats*.

EXIT STATUS

btrfs device returns a zero exit status if it succeeds. Non zero is returned in case of failure.

If the *−s* option is used, **btrfs device stats** will add 64 to the exit status if any of the error counters is non−zero.

AVAILABILITY

btrfs is part of btrfs−progs. Please refer to the btrfs wiki **http://btrfs.wiki.kernel.org** for further details.

SEE ALSO

mkfs.btrfs(8), **btrfs−replace**(8), **btrfs−balance**(8)