Living with BTRFS

KWLug - April 2015

Chris Irwin

With what?

- “Butter F S”
- “Better F S”
- “B-tree F S”
- “Bee Tee Arr Eff Ess”

Why should I consider a new filesystem?

ext3/ext4/ntfs/etc work fine for me!

My trip to Ottawa!

Figure 1: Trip to Ottawa

BTRFS Benefits

- Data & metadata checksums
- Subvolumes
- Copy on Write
- Snapshots
- Defragmentation support
- Deduplication support
- Multi-device support (“RAID”)
- SSD optimizations
- send/recieve support

Data & metadata checksums

All data is checksummed at write, and verified at read.

The files you save are guaranteed* to be the files you get!

So you don't need ECC like ZFS?

Strictly speaking, you don't need ECC memory for either ZFS or BTRFS. ECC memory provides error detection and correction in-memory. It would be ideal to have and use, but not all machines support it.

Not having ECC doesn't mean you shouldn't use a filesystem that provides checksums. It will still protect you from disk errors.

Car analogy: If you don't have airbags, you still wear your seat belt.

Subvolumes

One btrfs filesystem (disk/partition/etc) can contain multiple subvolumes (root, home, etc).

- Logical separation for data (like partitions or logical volumes)
- Can be mounted directly (like partitions or logical volumes)
- No division of free space (unlike partitions or logical volumes)
- Can be mounted as one (unlike partitions or logical volumes)

While they have special abilities, subvolumes act like directories.

Copy on Write

Changes don't physically overwrite what they're logically overwriting.

```
AAAA -> ABBA
```

```
AAAA      AAAABB
[1-4]     [1,5-6,4]
```

Alternative copy on write method

ex: LVM

```
AAAA -> ABBA
```

```
AAAA     AAAAAA    ABBAAA
[1-4]              [1-4]
```

Due to having to sync the disk to save AA before writing BB, this affects write performance.

Note: BTRFS doesn't do this, but other things do (LVM...)

Copy on Write tricks

File copies can use no additional storage*

```
$ cp --reflink=always original.jpg copy.jpg
```

Unlike a hardlink, modifications to copy.jpg don't touch original.jpg

Snapshots

Snapshots can be read-only or read-write.

- Read-only are ideal for *facilitating* backups
- Read-write can be used for testing, etc.

Snapshots don't use any additional disk space, and don't snapshot free space, unlike LVM.

```
$ btrfs subvolume snapshot [-r] ./home ./home-20150413
```

What about LVM?

LVM does provide snapshots:

- Negatively affect performance and throughput – one write becomes two writes and at least one sync
- Require a pre-allocated amount of space
- become corrupt when that space is used
- even if that space was “free” on the original volume

Fragmentation!

Yes, BTRFS does cause fragmentation. For this reason, BTRFS developers suggest not using databses (mysql), or virtual machines on BTRFS filesystems because they will become fragmented over time.

The bright side is with SSDs, fragmentation is less of an issue. SSDs themselves are already fragmented by design, so file contiguity is not important. However, a very heavily fragmented file can cause some CPU spikes when reading on an SSD.

For files that are not modified (images, music. . .), or overwritten completely (os updates, atomic file switches), there is no impact.

Defragment support

There is on-line defragmentation support, which can defragment at the filesystem, subvolume, directory, or file level.

```
$ btrfs filesystem defragment [-r] [-t 5M] /home
```

Maybe don't defragment a SSD?. It could cause unecessary erase cycles on your SSD, but it doesn't relocate files unneccessarily. It *probably* is fine with current SSDs.

Deduplication support

BTRFS supports on-line, out-of-band deduplication. This is accessed using third-party tools that look for duplicate blocks, and then tell btrfs to merge them.

- bededup
- dupremove

I have not used these tools.

Live deduplication support is being worked on, but will require large amounts of RAM. Large amounts of storage is typically cheaper than large amounts of RAM, but your needs may vary.

I will not use that, either.

Defragment and deduplication

Dedup two files, then defrag ment them. What happens?

Disable COW on a file

You can disable COW on a directory or file. This will avoid fragmentation, but at a cost:

- Data checksumming is disabled.
- Snapshots won't work (because they rely on COW)

You very probably do not want to do this.

```
$ chattr +C /dir/file
```

Multi-device support ("RAID")

BTRFS implements it's own multi-device support.

"RAID1" just means to make sure at least two copies exist for all data. It can work with mismatched drives (1TB, 500GB, 500GB)

"Single" means to keep only one copy of data, ensuring you can use all available space (but with no redundancy/recovery)

Note: RAID5+6 are still in heavy development. Until recently, there was no drive replace support. I've not used, experimented, or investigated these modes.

Versus traditional RAID

From Data Corruption on Wikipedia:

> *As an example, ZFS creator Jeff Bonwick stated that the fast database at Greenplum – a database software company specializing in large-scale data warehousing and analytics – faces silent corruption every 15 minutes.[9] As another example, a real-life study performed by NetApp on more than 1.5 million HDDs over 41 months found more than 400,000 silent data corruptions, out of which more than 30,000 were* ***not detected by the hardware RAID controller****. Another study, performed by CERN over six months and involving about 97 petabytes of data, found that about 128 megabytes of data became permanently corrupted.*

BTRFS will detect a bad read and recover from a second copy, or cause the read to fail, hopefully making you restore your backups (rather than silently accepting data corruption)

Simulating Data Corruption

From raid.wiki.kernel.org

> *RAID (be it hardware or software), assumes that if a write to a disk doesn't return an error, then the write was successful. Therefore, if your disk corrupts data without returning an error, your data will become corrupted. This is of course very unlikely to happen, but it is possible, and it would result in a corrupt filesystem.*

Simulating Data Corruption

. . .

RAID cannot, and is not supposed to, guard against data corruption on the media. Therefore, it doesn't make any sense either, to purposely corrupt data (using dd for example) on a disk to see how the RAID system will handle that. It is most likely (unless you corrupt the RAID superblock) that the RAID layer will never find out about the corruption, but your filesystem on the RAID device will be corrupted.

This is the way things are supposed to work. RAID is not a guarantee for data integrity, it just allows you to keep your data if a disk dies (that is, with RAID levels above or equal one, of course).

BTRFS Integrity

BTRFS supports “scrubbing”, checking the integrity of all data. This can be done on a path, or a specific device in a multi-device BTRFS.

```
$ btrfs start [-B] [-c #] [-n #] <path>
```

It can report any errors found. If another copy is available (“RAID1”), it will automatically fix the bad copy.

DEMO!

Demonstration of corruption on plain ext4, ext4 on mdadm raid 1, ext4 on mdadm raid 5, btrfs (single device), and btrfs “raid1”

SSD optimizations

BTRFS has some useful mount options

autodefrag : Detect random writes and defragments the affected files

degraded : Allow you to mount a "RAID" filesystem missing a device.

ssd : "Avoiding unnecessary optimizations. This results in larger write operations and faster write throughput.

discard : Enables TRIM support. This is disabled by default as it affects generational rollbacks (not covered in this talk), and many drives reserve space for garbage collection on their own. There is also some concern about "queued TRIM support", which didn't exist before SATA 3.1. (Also, queued TRIM support seems buggy, since Windows doesn't use it)

Send/Receive support

BTRFS can perform it's own backups. Much like the `dump` utility for ext2/3/4, this has the advantage of working at the filesystem level. It will preserve all data btrfs knows, including the checksum data, and shared blocks.

```
$ btrfs send ./home-20150413 \
  | btrfs receive /mnt/backups/
```

My BTRFS-formatted USB backup disk now contains an exact copy of my snapshot – right down to which blocks are shared between files, and the data checksums of those blocks.

Send incremental

Send/receives can also share space (and save time) as well by providing a 'parent' reference that exists on both drives:

```
$ btrfs send -p ./home-20150412 ./home-20150413 \
  | btrfs receive /mnt/backups/
```

Because all data is checksummed, you can be guaranteed to have a valid "full" home-20150413 on your backup drive.

Send/Receive variations

You can send snapshots to another host:

```
$ btrfs send ./home-20150413 \
  | ssh my-server.com btrfs receive /mnt/backup/laptop/
```

Or if you're backing up to ext4, S3, or another non-btrfs volume:

```
$ btrfs send ./home-20150413 \
  | gzip /mnt/backup-ext4/home-20150413.gz
```

And restore at a later date:

```
$ zcat /mnt/backup-ext4/home-20150413.gz \
  | btrfs receive /path/to/restore
```

Snapshot utilities

snapper

- Lots of users. Weird (to me) snapshot layout
- Uses some logic to keep 24 hourly snapshots, then expire those and keep daily snapshots, etc.
- No support in itself for getting those snapshots to another disk/host
- Some issues with scheduled snapper-clean + discard

BTRFS Wiki on Backups

There are other tools, or you could roll your own cron-job.

So what about ZFS?

I thought ZFS did all these things?

Yes, it does

Why not just use ZFS?

Go ahead

Why don't you use ZFS?

- I don't want to use Solaris/FreeBSD
- I don't want to use a third-party filesystem on Linux
- I don't want to depend on COPRs/PPAs and dkms/akmods for filesystem drivers
- BTRFS is built-in

Why doesn't everybody use BTRFS?

- It is still a young filesystem. Some features are incomplete (deduplication), dangerously incomplete (RAID5/6), or missing (live deduplication).
- It doesn't support all use-cases (swap files won't work at all, fragmentation with databases/virtual machines/torrents).
- Some people have experienced problems that resulted in data loss (though not too recently). With extensive backups, this shouldn't be an issue. However, filesystems are expected to be fool-proof.
- There are some gotchas and quirks that are confusing.

BTRFS “Gotchas”

From Gotchas on BTRFS Wiki

- deadlock at mount time in 3.19.1 - 3.19.3
- deadlock on heavy (rsync) workloads in 3.15 - 3.16.1
- defragment/snapshot problems before 3.9 - 3.14
- defragmenting with kernels up to 2.6.37 will unlink COW-ed copies of data, don’t use it if you use snapshots, have de-duplicated your data or made copies with `cp --reflink`.

BTRFS Quirks

df is correct/incorrect/misleading

Assuming a 1TB+500GB “RAID1”, how much free space is there?

BTRFS Quriks

Assuming a 1TB+500GB "RAID1", how much free space is there?

- df reports 1.5TB.
- There is 1.5TB of free space
- You can only write 500GB of data to this drive. You will then get an out of space error, while df reports 500GB free space. Not intuitive or obvious.
- You also can't modify or overwrite files at that point, because it needs to write the new data before the old data is de-referenced and considered "free"

BTRFS Quirks

Why not have df report *usable* space instead of *free* space

That's too hard

BTRFS Quirks

What about 1x2TB + 2x500GB as "RAID1"?

- You could have 1.0TB of usable space: each 500GB as the "copy" for 1TB, with 1TB "free" but unusable
- You could have 500GB of usable space: one 500GB master, one 500GB copy, and 2TB "free" but unusable

It depends on how BTRFS decides to allocate it's data (prefer fewer drives vs spread out as many drives). It may depend on if the filesystem was grown (i.e. added 2TB drive to existing 2x500GB array). There may be other factors.

BTRFS Quirks

In theory, developers could figure out the above to list free space...
But the BTRFS roadmap has "Object-level mirroring and striping", so free space could depend on *where* your file was written.

BTRFS Quirks

There is no good solution to this I can think of, other than KIDS^{TM}: Keep Your Disks Simple

Should I stay away then?

- I haven't had a single BTRFS* issue that resulted in data loss*
- Backups solve all
- ext4-to-btrfs migrations are easy/hard

What am I using

- lvm
- btrfs for /, /home
- ext4/xfs for /boot, games (steam), torrents, Virtual Machines, etc.

References

- Official BTRFS Wiki
- BTRFS Mount options
- BTRFS Backups
- BTRFS on Arch Wiki
- Snapper article