G-Technology®

BTRFs
ESSENTIAL GUIDE

G|RACK™ 12
High-Performance NAS

g-technology.com

G|RACK™ 12

G-RACK 12 BTRfs Backgrounder

Table of Contents

What is BTRfs?

B-tree file system. Pronounced as "butter F S", "better F S", "b-tree F S", or simply by spelling it out.

How long has it been around? Initially designed at Oracle Corporation for use in Linux. The development of BTRfs began in 2007. August 2014, the file system's on-disk format has been marked as stable.

Is it stable? August 2014, the file system's on-disk format has been marked as stable. Since its inception there have been over 45+ official releases. It has a very active community of developers.

BTRfs is a file system which is completely made from scratch. The BTRfs exists because its developers firstly wanted to expand the file system functionality in order to include snapshots, pooling, as well as checksums among the other things. While it is independent from the ext4, it also wants to build off the ideas present in the ext4 that are great for the consumers and the businesses alike as well as incorporate those additional features that will benefit everybody, but specifically the enterprises.

The role file systems play in storage devices.

Each file system has a different take on how and where to put data on the disks. Where files are stored and how they are organized is the essential role of any filesystem. Speed of retrieval is a key factor in both the design of filesystems and the selection of them for your computer.

Just in case if you are unfamiliar about what file systems really do, it is actually simple when it is summarized. The file systems are mainly used in order for controlling how the data is stored after any program is no longer using it, how access to the data is controlled, what other information (metadata) is attached to the data itself, etc. I know that it does not sound like an easy thing to be programmed, and it is definitely not. The file systems are continually still being revised for including more functionality while becoming more efficient in what it simply needs to do. Therefore, however, it is a basic need for all computers, it is not quite as basic as it sounds like.

Why is BTRfs a good choice for a file system?

The key features of BTRfs are highly desirable to most computer systems administrators:

- Snapshots
- Sub-volumes
- Quotas
- Copy on Write Logging

Data and metadata in BTRfs are protected with copy on write logging (COW). Once the transaction that allocated the space on disk has committed, any new writes to that logical address in the file or btree will go to a newly allocated block, and block pointers in the btrees and super blocks will be updated to reflect the new location.

Some of the BTRfs trees do not use reference counting for their allocated space. This includes the root tree, and the extent trees. As blocks are replaced in these trees, the old block is freed in the extent tree. These blocks are not reused for other purposes until the transaction that freed them commits.

All subvolume (and snapshot) trees are reference counted. When a COW operation is performed on a btree node, the reference count of all the blocks it points to is increased by one. For leaves, the reference counts of any file extents in the leaf are increased by one. When the transaction commits, a new root pointer is inserted in the root tree for each new subvolume root. The key used has the form:

```
Subvolume inode number   BTRfs_ROOT_ITEM_KEY Transaction ID
```

The updated btree blocks are all flushed to disk, and then the super block is updated to point to the new root tree. Once the super block has been properly written to disk, the transaction is considered complete. At this time the root tree has two pointers for each subvolume changed during the transaction. One item points to the new tree and one points to the tree that existed at the start of the last transaction.

Any time after the commit finishes, the older subvolume root items may be removed. The reference count on the subvolume root block is lowered by one. If the reference count reaches zero, the block is freed and the reference count on any nodes the root points to is lowered by one. If a tree node or leaf can be freed, it is traversed to free the nodes or extents below it in the tree in a depth first fashion.

The traversal and freeing of the tree may be done in pieces by inserting a progress record in the root tree. The progress record indicates the last key and level touched by the traversal so the current transaction can commit and the traversal can resume in the next transaction. If the system crashes before the traversal completes, the progress record is used to safely delete the root on the next mount.

Comparison of features between the most popular file system ext4.

BTRfs vs ext4

ext4 and BTRfs are modern file system types which can be used with a Linux operating system.

ext4 advantages

ext4 is the default for major Linux distributions such as Ubuntu. ext4 is in use by a very large number of Linux users and has proven to be useable. BTRfs is not considered ready for normal (production) use, and has not been widely tested by end users. One of the primary problems with BTRfs today is its incomplete fsck (filesystem check) implementation, which may not be able to successfully fix file system problems when they appear. (According to the BTRfs wiki as of 2012/7/3, «while this tool should be able to repair broken filesystems, it is still relatively new code, and has not seen widespread testing on a large range of real-life breakage. It is possible that it may cause additional damage in the process of repair.»)

Benchmarks have shown that ext4 provides superior read-write speed to BTRfs. Since BTRfs is under development, it may (or may not) improve performance to match ext4 in the future. BTRfs has shown high disk consumption on small files.

BTRfs advantages

BTRfs will provide new features, such as the «copy-on-write» concept to improve performance and reliability. BTRfs systems will be able to «roll back», meaning all changes made to the hard disk since a certain point in time would be reversed.

BTRfs and ext4 both support online grow, but BTRfs supports online shrink, while ext4 only supports offline shrink. This means BTRfs users do not have to unmount the filesystem to shrink it.

The way to the future

Theodore Ts›o, principal developer of ext3 and a maintainer of ext4, says that BTRfs is the way forward due to new features in its design.

Public Opinion from /r/BTRfs on reddit.

Love:

- Online everything! Dedup, defrag, shrink, grow, raid level migration, adding/removing drives. Think about this: you can move the root partition to a different drive without rebooting.
- Snapshots. I use these every day, and sometimes when I use them they›re a lifesaver.
- Send/receive. In combination with Snapshots an über awesome base to build a backup system.
- Checksumming. I have lost data in the past to bit rot and I don't want it to happen again.

Like:

- Performance - it›s usually good, and it generally «does the right thing» with ssds.
- It doesn›t eat RAM the way zfs does.
- Subvolumes make volume management and snapshot management easy.
- Can de-fragment (unlike zfs)
- Can dedupe out-of-band, so unlike zfs this can be done without huge memory requirements

Dislike:

- Not fully stable. Don't get me wrong. I do believe it's ready for day to day usage even for production systems. But you still have to be cautious. You should look up known problems or limitations before using some features. For example the conversion from ext seems to have problems since Linux 4. Some of them are not problems/bugs in BTRfs but other programs that have not been found yet.
- General growing pains (not everything supported on every configuration.)
- qgroups (every time I›ve hit a bug, it›s been here.)

Hate:

- Performance with virtual machine images and databases is simply bad. I know about NoCow but it turns off checksumming.
- Encryption has been talked about for years, but is nowhere on the road map.
- RAID is limited to two disks of redundancy, and this might be by choice since a patch to allow nearly arbitrary levels of redundancy in RAID was submitted years ago. I keep hoping this is just attributed to growing pains and BTRfs RAID›s immaturity, but I can›t be sure.

Why BTRfs is awesome?

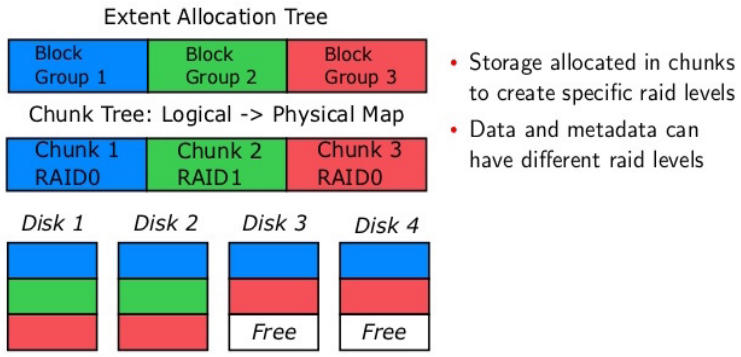
Ok, so BTRfs is stable enough to be trusted. Or at least with companies whose judgment has way more value than most like Facebook and SuSE Linux experts. At this point, if you still don't trust it, stop reading this post and keep using ext4 or xfs, no problem.

But if you are thinking "maybe I can use BTRfs on my next Linux deployment", why should you consider it? Well, because it has some great features! The page linked at the beginning has the complete list, here I'm going to list the ones I prefer the most.

BTRfs has been designed from the beginning to deal with modern data sources, and in fact is able to manage modern large hard disks and large disk groups, up to 2^64 byte. That number means 16 EiB of maximum file size and file system, and yes the E means Exabyte. This is possible thanks to the way it consumes space: other file systems use disks in a contiguous manner, layering their structure in a single space from the beginning to the end of the disk. This makes the rebuild of a disk, especially large ones,

extremely slow, and also there's no internal protection mechanism as one disk is seen as a single entity by the filesystem itself.

BTRfs instead uses "chunks". Each disk, regardless its size, is divided into pieces (the chunks) that are either 1 GiB in size (for data) or 256 MiB (for metadata). Chunks are then grouped in block groups, each stored on a different device. The number of chunks used in a block group will depend on its RAID level. And here comes another awesome feature of BTRfs: the volume manager is directly integrated into the filesystem, so it doesn't need anything like hardware or software raid, or volume managers like LVM. Data protection and striping is done directly by the filesystem, so you can have different volumes that have inner redundancy:



BTRfs allocation tree

For example, Block group 2 is configured for RAID1 redundancy. So, a chunk is consumed on disk1, and its mirror is stored in another device, Disk 2 in the picture. In this way, if we lose Disk1, another copy of the block is still available on Disk2, and another copy can be immediately recreated for example on Disk3 using the free chunk. You can configure BTRfs for File Striping, File Mirroring, File Striping+Mirroring, Striping with Single and Dual Parity.

Another aspect of BTRfs is its performance. Because of its modern design and the b-tree structure, BTRfs is damn fast. If you didn't already, look at the video above starting from 30:30. They have run a test against the same storage, formatted at different stages with XFS, EXT4 and BTRfs, and they wrote around 24 million files of different size and layout. XFS takes 430 seconds to complete the operations and it was performance bound by its log system; EXT4 took 200 seconds to complete the test, and its limit comes from the fixed inode locations. Both limits are the results of their design, and overcoming of those limits was one of the original goal of BTRfs. Did they succeed? The same test took 62 seconds to be completed on BTRfs, and the limit was the CPU and Memory of the test system, while both XFS and EXT4 were able to use only around 25% of the available CPU because they were quickly IO bound.

The main BTRfs features available the G-RACK 12 include:

- Extent based file storage
- 2^64 byte == 16 EiB maximum file size (practical limit is 8 EiB due to Linux VFS)
- Space-efficient packing of small files
- Space-efficient indexed directories
- Dynamic inode allocation
- Writable snapshots, read-only snapshots
- Subvolumes (separate internal filesystem roots)
- Checksums on data and metadata (crc32c)
- Compression (zlib and LZO)
- Integrated multiple device support
- File Striping, File Mirroring, File Striping+Mirroring, Striping with Single and Dual Parity implementations (RAID-5 and RAID-6)
- Background scrub process for finding and fixing errors on files with redundant copies
- Online filesystem defragmentation
- Offline filesystem check.
- Subvolume-aware quota support
- Online filesystem check BTRfs Links and references

G|RACK™ 12

Links and references

G-Technology®

Links and References

Informative Links about BTRfs:

- https://www.reddit.com/r/btrfs/
- https://btrfs.wiki.kernel.org/index.php/Main_Page
- https://en.wikipedia.org/wiki/Btrfs