VIVA-Tech International Journal for Research and Innovation *Volume 1, Issue 5 (2022)*
ISSN(Online): 2581-7280
VIVA Institute of Technology
10th National Conference on Role of Engineers in Nation Building – 2022 (NCRENB-2022)

VIVA-TECH INTERNATIONAL JOURNAL FOR RESEARCH AND INNOVATION
ANNUAL RESEARCH JOURNAL
ISSN(ONLINE): 2581-7280

A review on use of btrfs file system in Linux operating systems

Rohit Jha[1], Arun Nair[2]
[1](MCA, VIVA Institute of Technology/ Mumbai University, India)
[2](MCA, VIVA Institute of Technology/ Mumbai University, India)

Abstract : *BTRFS (B-tree Filesystem) is a modern file system developed for Linux Operating System. It has been adopted as the default file system in many distributions like SUSE and OpenSUSE, and almost all major distributions ship it as an optional file system. It is based on copy-on-write which allows snapshots and clones. It uses a B-tree structure as its main data structure. Its main features and benefits include Snapshots, RAID support, Self-healing capabilities, among others. A linux filesystem could be put in on an smartphone as well as on the enterprise servers. This demands the resolution of many challenges like Scalability, Data integrity and Disk diversity to make any file system usable on a particular system. This paper describes the basic idea, algorithms, data structures, current development, and changes used in this file system. It also discuss the challenges posed by disk defragmentation in the presence of snapshots.*
Keywords -*B-Tree, BTRFS, Encryption, Filesystem, Linux.*

I. INTRODUCTION

Btrfs is an open source filesystem that was started in 2007. It was initially designed by Oracle Corporation to use in Linux. But since November 2013, the file system has been declared as stable in the Linux kernel. Along with Oracle BTRFS was developed in cooperation with Fujitsu, Fusion-IO, Intel, Red Hat, Strato, SUSE, and many others [1].According to Oracle, Btrfs "is not a true acronym".
Its main features are [2]:

- Snapshotsthey do not make full copies of files.
- Built-in volume management, support for software-based RAID 0, RAID 1, RAID 10 and others
- Self-healing capablity - detection of silent data corruptions
- Multi-device support
- Online resize and defragmentation
- Compression
- Swapfile support

It aims to solve scalability problems for file storage, while also adding new features that existing filesystems lacked. Btrfs is able to detect bad copies of block and use inbuilt RAID to pull correct data as it maintains both data and metadata integrity checksum [3].

II. HISTORY

The original idea of using the copy-on-write B-tree was proposed by IBM researcher Ohad Rodeh at a presentation at USENIX 2007 [4].BTRFS was finally accepted into the Linux kernel mainline in 2009 [5]. Several Linux distributions began offering Btrfs as an experimental choice of root file system during installation at that time BTRFS was extremly unstable.
In 2011, Btrfs automatic defragmentation and scrubbing features were merged into Linux kernel 3.0 mainline [6].In 2012, Oracle Linux and SUSE Linux Enterprise Server provided support for BTRFS [7][8]. In 2015, Btrfs was adopted as the default filesystem for SUSE Linux Enterprise Server 12 [9]. Red Hat removed BTRFS support from RHEL 8 (RedHat Enterprise Linux) [10].BTRFS was selected as default filesystem for Fedora 33 desktop in 2020 [11].

III. FEATURES

As of version 5.0 of the Linux kernel, Btrfs implements the following features [12]

3.1 Major Features Currently Implemented

- Extent based file storage
- 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, xxhash, sha256, blake2b)
- Compression (ZLIB, LZO, ZSTD), heuristics
- Integrated multiple device support
- File Striping
- File Mirroring
- File Striping+Mirroring
- Single and Dual Parity implementations (experimental, not production-ready)
- SSD (flash storage) awareness
- TRIM/Discard for reporting free blocks for reuse
- Optimizations
- Background scrub process for finding and repairing errors of files with redundant copies
- Online filesystem defragmentation
- Offline filesystem check
- In-place conversion of existing ext2/3/4 and reiserfs file systems
- Seeding devices.
- Subvolume-aware quota support
- Send/receive of subvolume changes, efficient incremental filesystem mirroring and backup
- Batch, or out-of-band deduplication (happens after writes, not during)
- Swapfile support
- Tree-checker, post-read and pre-write metadata verification
- Zoned mode support (SMR/ZBC/ZNS friendly allocation)
- fsverity integration

3.2 Features by kernel version

Table 1: List of features by kernel version

Feature	Version	Description
scrub	3	Read all data and verify checksums, if possible repair it.
store otime	4	Save Save creation time (otime) while creating new files and directories.
swapfile	5	With some limitations where COW design does not work well with the swap implementation btrfs will try to improve that in the future
metadata uuid	5	An optional incompat feature to assign a new filesystem UUID without overwriting all metadata blocks.

3.3 Features Currently in Development or Planned for Future Implementation

- DAX/persistent memory support
- The file/directory -level encryption support using fscrypt

3.4 Cloning

Btrfs provides a cloning feature that creates a copy-on-write snapshot of a file [13]. By cloning, the file system does not create a new link pointing to an existing inode; instead, it creates a new inode that shares the same disk blocks of original file. Eailier cloning worked only within the boundaries of the same Btrfs file system, but in version kernel 3.6 it may cross the boundaries of subvolumes under certain circumstances [14].

3.5 Subvolumes and snapshots

A subvolume is a part of filesystem with its own independent file hirarchy. It can be accessed by mounting the top-level volume, in which subvolumes are visible as their subdirectories [15]. A subvolume looks like a normal directory. Subvolumes can be renamed or moved like a normal directory, but the numeric id (called subvolid or rootid) of the subvolume is persistent and cannot be changed. Sub-volumes could be created at any place in filesystem hirarchy and they can also be nested. Nested subvolumes will appear as subdirectories within their parent subvolumes, similarly to the traditional directory-subdirectory architecture. Deleting a subvolume is not possible until all subvolumes below it are cleared and deleted, hence the top-level sub-volumes cannot be deleted [16].

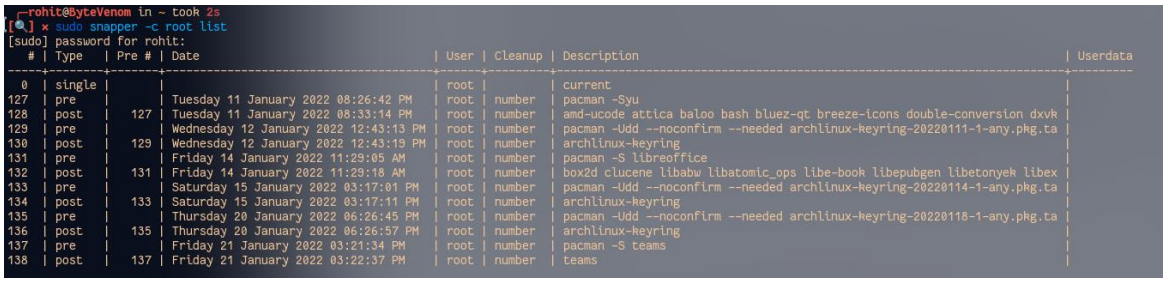

Fig. 1: Example of a BTRFS file system subvolume

A snapshot is a sub-volume that shares its data (and metadata) with other subvolumes using Btrfs' copy-on-write capabilities, modifications to a snapshot are not visible in the original subvolume. Once a writable snapshot is created, it is treated as an alternate version of the original file system. Hence to roll back to a snapshot, the original (modified) subvolume needs to be unmounted and the older snapshot needs to be mounted in its place. After sucessful roll-back the original subvolume may also be deleted [17].
The copy-on-write (CoW) nature of Btrfs means that snapshots are regularry created hence while system initially consumes very little disk space, their size increases exponrntially. As a snapshot is a subvolume in itself, nested snapshots could also be created. Taking snapshots of a subvolume is not a recursive process hence when a sanapshot of a volume/subvolume is created every sub-volume under that volume are mapped to an empty directory with the same name [18].

3.6 Quota groups

A quota group (or qgroup) restricts a subvolume to use too much space by imposes an upper limit to the space it may consume. A new snapshot initially has no quota because its data is shared with its parent, but thereafter copy-on-write and new files increases its size. A quota group is automatically created whenever a new subvolume or snapshot is created. These initial quota groups are used to implement quota pools [19].
Quota groups are applied to subvolumes and snapshots only, enforcing quota groups to users, or user groups is not possible. However, it could be achived by using different subvolumes for all users or user groups that require a quota to be enforced.

3.7 Encryption

Chris Mason stated that thesupport for encryption was planned for Btrfs during his interview in 2009. As of 2020 the developers were working to add keyed-hash message authentication code (HMAC) to the Btrfs[20]. Currently, full-disk encryption mechanism such as dm-crypt / LUKS could be installed on underlying devices and Btrfs filesystem could be installed on top of that layer.

IV. PERFORMANCE

There are no standard for testing filesystem performance. NFA and CISF protocols are considered to be industry benchmark, they are not accurate for higher workloads. Each filesystem may perform diffrently under different application loads. Best way to check which filesystem is best for particular use case, is to try several filesystems, and see which one works best.
As to it is impossible to cover all use cases, we chose several common benchmarks, to show that BTRFS performs comparably with its alternatives such as XFS and Ext4. These are much more mature systems than Btrfs. For our comparision we chose two storage configurations : a hard disk, and an SSD.

4.1 Hard disk

Hard disk tests where run on a single socket 3.2 Ghz quad core x86 processor with 8 gigabytes of ram on a single SATA drive with a 6 gbps link. The test was a Linux kernel build, starting from a clean source file tree. For this test Ext4 has slightly higher throughput than BTRFS and XFS. The compile times are all within a few seconds of each other on each filesystem. The kernel compile test tends to be a bit meta-data intensive, and it is a good benchmark for an application that has a heavy meta-data load [1].

4.2 Flash disk (SSD)

Flash drives are are becoming more common in modern laptops and desktops. Smartphones and embedded devices use flash disks and are generally linux based. Thus it is important to optimize BTRFS for Solid State Drives (SSDs).
SSD test was also done by compilng linux kernal 3.3. Performance in the kernel 3.3. File creation rate is unsteady. The number of reads had to be reduced to improve performance. The core problem was the way the Linux virtual memory was used. Linux virtual memory system assumes that the allocated pages will be used for a while but Btrfs uses many temporary pages due to its copy-on-write nature, where data move around on the disk rapidly. This was causing the VM to hold many out of date pages in memory, reducing cache effectiveness. The fix was to discard invalid pages from VM as soon as they have become invalid. After applying the propsed solution throughput was steady at about 125MB/sec, and the rate of file creation is 150,000 files per second [1].

V. CONCLUSION

BTRFS is a relatively young filesystem. Although it is still in development but tests have proven that it could be used as a default file system in desktop as well as on servers. It has been adopde as default file system in many distributions like SUSE and OpenSUSE, and almost all major distributions ship it as an optional file system at the time of installatioon.
BTRFS is based on copy-on-write that supports efficient snapshots and strong data integrity. BTRFS was finally accepted into the Linux kernel mainline in 2009. BTRFS works a wide range of hardware, from enterprise servers to smartphones and embedded systems. BTRFS is a modern copy-on-write (CoW) filesystem for Linux. It's objective was to impliment advanced and modern features while focusing on fault tolerance, repair and easy administration. Its main features and benefits are Snapshots, RAID support, Self-healing etc..
Btrfs provides a cloning feature that creates a copy-on-write snapshot of a file. By cloning, the file system does not create a new link pointing to an existing inode; instead, it creates a new inode that shares the same disk blocks of original file. A subvolume is a part of filesystem with its own independent file hirarchy. It can be accessed by mounting the top-level volume, in which subvolumes are visible as their subdirectories. Sub-volumes could be created at any place in filesystem hirarchy and they can also be nested. A quota group (or qgroup) restricts a subvolume to use too much space by imposes an upper limit to the space it may consume. A quota group is automatically created whenever a new subvolume or snapshot is created. These initial quota groups are used to implement quota pools. Btrf encryption is under development As of 2020 the developers were working to add keyed-hash message authentication code (HMAC) to the Btrfs. Currently, full-disk encryption mechanism such as dm-crypt / LUKS could be installed on underlying devices and Btrfs filesystem could be installed on top of that layer.
Performance tests were run on a single socket 3.2 Ghz quad core x86 processor with 8 gigabytes of ram on a single SATA drive with a 6 gbps link. The test was a Linux kernel build, starting from a clean source file tree.

On HDD Ext4 has slightly higher throughput than BTRFS and XFS. For SSDs to improve btrfs performance and increase throughput, the solution was to discard pages from virtual memmory as soon as they became invalid.
This paper describes the basic idea, algorithms, data structures, current development, and changes used in this file system. It also discuss the challenges posed by disk defragmentation in the presence of snapshots.

Acknowledgements

I would like to extend my special thanks to our Head of Department - Prof. Ms. Chandani Patel Ma'am, research coordinator Prof. Ms. Neha Lodhe Ma'am and our research guide Prof. Ms. Sonia Dubey Ma'am for the opportunity to work on this excellent research on the topic of A review on use of btrfs file system in Linux operating systems, while working on this paper I have also done a lot of research and improved my understanding about the topic. I am confident that the knowlwdge gained during this reasearch work will help me in forthcoming future. Secondly, I would like to thank the members of my research team for helping me carry out my duties at each stage of development and for contributing in the same way and with determination, and thank my parents and friends for helping me complete this research paper on time.

REFERENCES

[1] https://dominoweb.draco.res.ibm.com/6e1c5b6a1b6edd9885257a38006b6130.html
[2] https://btrfs.wiki.kernel.org/index.php/Main_Page
[3]McPherson, Amanda (22 June 2009) https://web.archive.org/web/20120627065427/http://www.linuxfoundation.org/news-media/blogs/browse/2009/06/conversation-chris-mason-btrfs-next-generation-file-system-linux
[4]https://www.usenix.org/legacy/events/lsf07/tech/rodeh.pdf
[5]https://www.linux-magazine.com/Online/News/Kernel-2.6.29-Corbet-Says-Btrfs-Next-Generation-Filesystem
[6]https://kernelnewbies.org/Linux_3.0#head-3e596e03408e1d32a7cc381d6f54e87feee22ee4
[7] https://blogs.oracle.com/linux/post/unbreakable-enterprise-kernel-release-2-has-been-released
[8]http://www.novell.com/linux/releasenotes/x86_64/SUSE-SLES/11-SP2/#fate-306585
[9]https://www.suse.com/releasenotes/x86_64/SUSE-SLES/12/index.html#fate-317221
[10]https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/considerations_in_adopting_rhel_8/file-systems-and-storage_considerations-in-adopting-rhel-8#btrfs-has-been-removed_file-systems-and-storage
[11]https://fedoramagazine.org/btrfs-coming-to-fedora-33/
[12]https://btrfs.wiki.kernel.org/index.php/Main_Page#Features
[13]https://lwn.net/Articles/331808/
[14]https://btrfs.wiki.kernel.org/index.php/UseCases
[15]https://btrfs.wiki.kernel.org/index.php/SysadminGuide
[16]https://docs.oracle.com/cd/E37670_01/E37355/html/ol_use_case3_btrfs.html
[17]https://btrfs.wiki.kernel.org/index.php/SysadminGuide
[18] http://docs.oracle.com/cd/E37670_01/E37355/html/ol_use_case3_btrfs.html
[19] http://sensille.com/qgroups.pdf
[20]Jansen, Arne (2011), http://sensille.com/qgroups.pdf