OVERVIEW OF MODERN FILE SYSTEMS

Boban Joksimoski
Faculty of Electrical Engineering and Information Technology
Skopje, Republic of Macedonia

Suzana Loskovska
Faculty of Electrical Engineering and Information Technology
Skopje, Republic of Macedonia

ABSTRACT

Data storage has always been of keen interest to the computer society. There have been remarkable ideas and implementation for overcoming the problems in data storage. New file systems, like ZFS and Btrfs, are challenging the old ideas and introduce new concepts for storage, regardless of the underlying hardware technology. Dynamical expansion of storage, easy backup, fast snapshots, cloning, mirroring and preventing data corruption will be key issues that will shape file systems in the future. An overview of modern file system is presented in this paper.

I. INTRODUCTION

We live in an age where the most valuable possession is information. Every byte of data is considered important and needs to be stored for further use. Thus, storage of data is important in today's technological society. From such demands, it is expected that the amount of data that needs to be stored, is growing exponentially. The importance of keeping the data safe so far surpasses every other aspect of functioning. Techniques used for storing data are many and every file system has purpose and field where it excels.

At the beginning of modern computing, every major computer system had its own file system. Today the trend is to allow the users to choose their own file system, of course if it's supported by the operating system. Lately, open source software (OSS) has gained momentum and its flagship products are becoming more accepted for widespread use. Companies that are leading the technological design (IBM®, Oracle®, Sun Microsystems® (now under Oracle), Red Hat®, etc), have learned to embrace an use the OSS for their own purpose. And almost all of them have developed or supported some kind of file system that is being used in open source software. This paper is a review of such efforts, and the next major file systems that should be used.

II. BACKGROUND

File systems have a long history and have been developing for a long time. We could separate file systems in many categories, developed for special purposes, for special hardware, etc. With many different storage types, like Solid State Drives (SSD) and RAID, file systems are being revised and improvements are being made to gain maximum performance while maintaining reliability. In fact, performance is not the key issue, but the reliability of the data. A lot of techniques were devised to keep the reliability of the data, and a lot of file systems support them. These techniques vary from logs [1], journals [2], checksums, etc. Depending of the underlying purpose, file systems can be categorized as disk file systems, network file systems and special purpose file systems. The interest of this article is the advanced disk file systems. We'll keep this paper close to disk file systems, but with a thought that network and special purpose file systems are of immense interest for future file systems.

The most commonly used disk file systems are XFS[3], ReiserFS, Ext family [4][5], NTFS, FAT family, HFS+, VxFS and so on. Almost every one of them is implemented under some license that allows it to be used in Unix-like operating system (distributions based on the GNU Linux[6], different BSD variants and/or proprietary Unix systems). All of them are well established, but lately a lot of buzz is being created about new type of file system that allows advanced techniques for scalability and reliability. One of the new file system announced is called btrfs [7] (B-Tree FS, or Butter FS) and is catching the interest of the developers. The file system was announced 2007 and it's an answer to Sun Microsystems® ZFS™ [8][9] (initially Zettabyte File System), introduced in 2005.

III. FEATURES OF A MODERN FILE SYSTEM

The computer industry has walked a long way from the primitive file systems that were developed for early computers. Now we can't imagine a disk file system that doesn't support hierarchy, security (either capability based security [10] of access list security - ACL), metadata information and data recovery. With the technology breakthroughs and advances made, new modern file systems should extend the maximum storage size they provide, should allow easy data encryption and advanced management of the file system. Advanced users should have more control over the properties of the file system and there should be minimum maintenance for the data, easy backup, strategies for snapshots, and so on. The file system must satisfy future demands in file size and volume size. As a introduction to both file systems, they provide maximum file size and volume size of 16 EiB (1 exbibyte = 2^{60} bytes), with 2^{64} and 2^{48} maximum number of files for Btrfs and ZFS respectively. To see how much that is, the total number of atoms in the earth's crust is approximately about 2^{60}.

IV. OVERVIEW OF ORACLE (SUN) ZFS™

The developers at Sun Microsystems have redesigned the whole working process and idea of a file system. The source code is published under Sun's Common Development and Distribution License (CDDL). Sun's ZFS should not be mistaken for IBM's zFS.

Unlike any other file system, ZFS is designed from 7 different layers that interact one with another [18].

The whole concept of disk drives and partitions is obsolete and it is replaced by so called "storage pools" or "zpools" [8].

©2010 Institute of Informatics.

Zpools provide easy method for adding more storage and the user isn't aware of the number of devices that are being used. All storage from all devices is viewed as a single storage unit (or pool) (Fig. 1). Pools are implemented to use virtual devices (vdevs) [9] thus making typical volumes unnecessary. A concept of virtual volume is present but mainly to provide export of the pool onto a disk device. Creating and managing zpool is relatively easy [11].

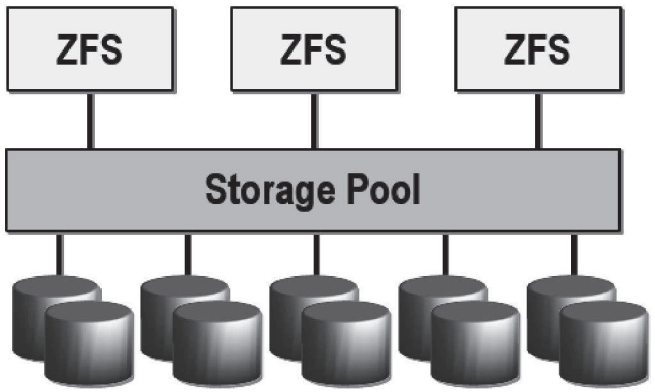

Figure 1: Storage pools and the ZFS file system

Another important feature is the usage of COW (Copy on Write) technique [12]. The technique is implemented in several other modern file systems (VxFS, Ext3, Btrfs) and manages concurrent access to the data. Every time when the data is modified, it is written to a new location. After the writhing is done, the inode pointers are updated so to the modified data (Fig. 2). Copy on Write provides mechanism for easy creation of file system snapshots (read only structures) and clones (read and write structures). And because of the COW technique, creating of a snapshot is in constant time – O(1), but deleting and incrementing a snapshot takes longer time – O(Δ).

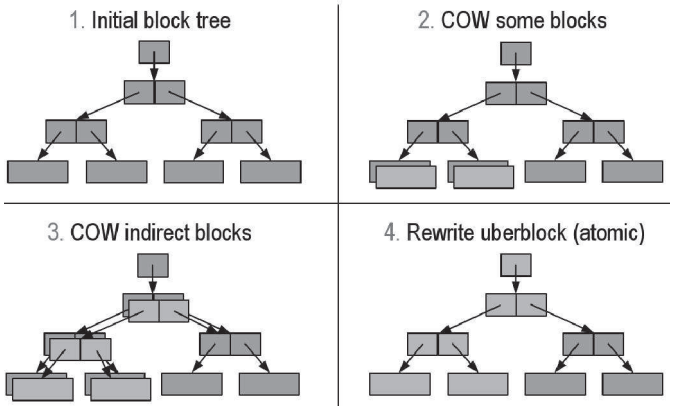

Figure 2: The Copy on Write (COW) Process

As a consequence of the COW process, the data on the disk is always valid, making administration and maintenance of the file system very easy. ZFS has a pipelined I/O engine that provides score-boarding, priority, deadline scheduling, out-of-order issue and I/O aggregation of the drives bandwidth. Also all the data is check summed and live disk scrubbed to prevent bit rots, phantom writes, misdirected reads and writes, DMA parity errors, driver bugs and accidental overwrite.
In addition to the data reliability, a non-standard RAID systems have been developed, dubbed RAID-Z, that optimize RAID-5 and RAID-6 for use with ZFS. Mirroring is also supported.
Security is implemented using NFSv4 Access Lists [13], making it incompatible with POSIX Access List present in most UNIX successors. Also, there is a native support for compression and encryption.

V. OVERVIEW OF BTRFS™

Btrfs was first announced in 2007, and its initial idea is to make it GNU GPL compatible file system that could rival the ZFS features. The main developer of Btrfs is Cris Mason, and main sponsor of the project is Oracle. It is not completed but a lot of effort is made to make it stable. As of Linux 2.6.29 it is part of the Linux Kernel as an unstable feature The whole idea of Btrfs is build around B-Trees and the implementation of COW on B-trees [15]. Actually all structures present in a Btrfs are stored using B-trees (like inodes, files, directory entries, block pointers). Unlike ZFS, Btrfs is keeping large data chunks as extents and allows keeping the small data chunks in the same block as the metadata, making more efficient use of the available disk space (Fig 3). It support dynamic inode allocation, creating writable snapshots (clones), creation of subvolumes, striping and check summing of the data, fast file system checking, compression, defragmentation and mirroring. As we can see, almost all of the features that are supported by ZFS are implemented in Btrfs. Btrfs also has a native support for RAID 0, RAID 1, RAID 5, RAID 6 and RAID 10.

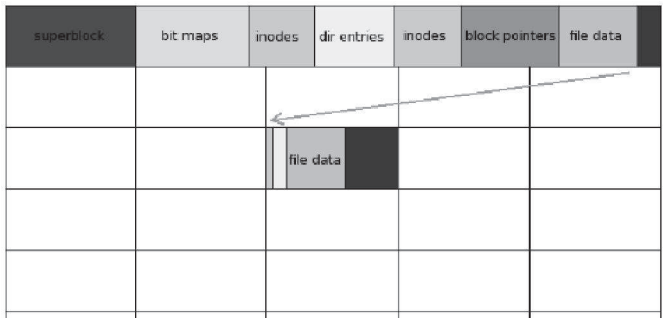

Figure 3: Simplified graph of the disk layout in Btrfs

Security is based on ACL , but with the difference that they are POSIX compatible.
The compact design of Btrfs is somehow “borrowed” from ReiserFS, but it is expected that Btrfs will be more stable. The features of Btrfs that are very similar to those of ZFS, but have completely different implementation (for comparison see 16 and 18).

VI. ISSUES WITH BOTH FILE SYSTEMS

The concepts that ZFS implemented were strong in favour of using it as the next everyday file system. But keeping ZFS as a technology under one company, made the other computer industry giants to react (it is surprising how easily big companies are scared). Also, distribution under the CDDL license made implementation for the Linux Systems hard, mainly because the Linux kernel is licensed under GNU General Public License and implementation of source code with CDDL is against GPL rules of compliance. A workaround is to implement ZFS in user space with the fuse library [14], but loss in performance would be the main drawback. Aside from Linux, ZFS was successfully ported to Open Solaris® and FreeBSD® [13], with efforts made for porting on Mac OS X® and NetBSD®.

With Linux being most widespread open source operating system, Btrfs is surely to be the biggest competitor to ZFS. The usage of these advanced file systems will shape data management in years to come.

When talking about file systems, there is the problem of networking and making the file system distributed. Both file system do not support native distribution, but ongoing projects are set to implement those features. Sun Microsystems bought Cluster File Systems in 2007, a key player in distributed file systems with their Lustre object based distributed file system. Since their acquisition, there was announcement of porting ZFS to work with Lustre, but still there are no significant results. Oracle on the other hand announced its own networking protocol, Coherent Remote File System (CRFS) [17], also distributed under GPL, specially designed for the Btrfs file system. CRFS is in its early stages of development, and we cannot expect any significant release in a year.

ZFS has implemented the role of Logical volume management (LVM) in its design, while Btrfs is driven by more common ideas and does not implement virtual devices and pools. In addition, it is currently impossible to reduce the available size of ZFS pool or remove a virtual device. These issues are not a problem with Btrfs. ZFS, unlike Btrfs, does not require the use of file system check (fsck), thus making the system easier to administer. ZFS is not efficient as Btrfs in disk usage, providing bigger overhead and disk head seeking. Btrfs has a native support for Solid State Drives, making use of techniques like wear levelling. Engineers of ZFS have implemented hybrid storage model, consisting of SSDs and typical hard disk drives, for better performance.

Performance of both file systems is disputable. Benchmarks were given for Btrfs and it is less compatible in speed with all used file systems. Being still in beta, it is believable that performance will increase as the project matures. ZFS on the other hand have made improvements and its benchmarks are officially available [19], but are made by the company that created it. Both file systems are generally slower than their rivals, but that an easy trade-off for the features given.

A last, a final concern of the open source developer community is the acquisition of Sun Microsystems by Oracle, in April 2009. Now Oracle has a leading role in both file systems, thus making people nervous about the future of the projects. Btrfs main developer as a response added that Oracle will not drop support for its file system. ZFS has the advantage of being developed longer, and already commercially distributed and deployed in working environments.

VII. COMPARISON WITH OTHER FILE SYSTEMS

Other file systems are not the focus of this paper, but it is good to see where the competition is. The most popular file systems were mentioned before, and they make use of standard techniques to make user data secure. Ext3 and Ext4 implement journaling and provide convenient ways of data storing. NTFS provides partially implemented extents. All of them rely on disks and partitions, a constraint that is ignored in Btrfs and ZFS. Also Btrfs and ZFS are the only one to provide fault tolerance. Speed is traded for better overall system expandability. The only file system that approaches some of the features offered by these two is the VxFS from Veritas.

VIII. CONCLUSION

File systems are a must in modern computers. ZFS and Btrfs have implemented new and innovative design ideas and techniques to make user data more secure and available. They represent the new generation of file systems that should power the server or desktop computers and even grid computers (if the appropriate projects are successful). All together, they provide a nice overview of how innovative thinking can refine or rewrite old ideas.

REFERENCES

[1] M. Rosenblum and J. K. Ousterhout: The Design and Implementation of a Log-Structured File System, ACM Transactions on Computer Systems (TOCS), Vol 10, Issue 1, Feb 1992, pp. 26 – 52.

[2] C. Swenson, R. Phillips and S. Shenoi: File System Journal Forensics, Advances in Digital Forensics III, Springer 2007, pp. 231-244.

[3] J. Mostek, W. Earl et al.: Porting the SGI XFS File System to Linux, USENIX Annual Technical Conference, 2000.

[4] T. Ts'o: The Linux ext2/3/4 Filesystem: Past, Present, and Future, IBM Linux Technology Center, 2006

[5] S. C. Tweede: Journaling the Linux ext2fs Filesystem, The Fourth Annual Linux Expo, 1998

[6] S. French: Around the Linux File System World in 45 minutes, Proceedings of the Linux Symposium, Vol 1, 2008, pp. 129-135

[7] C. Mason: "The Btrfs Filesystem", LinuxConf Europe, 2007.s

[8] "ZFS On-Disk Specification" sun.com, retreived on 2009-05-12. http://opensolaris.org/os/community/zfs/docs/ondiskformat0822.pdf

[9] "ZFS: The Last Word In File Systems" sun.com, retreived on 2009-05-12. http://opensolaris.org/os/community/zfs/docs/zfs_last.pdf

[10] H. M. Levy: Capability-Based Computer Systems, Butterworth-Heineman, Newton, MA, USA, 1984
[11] H. J. Foxwell and C. Tran: The ZFS File System, Pro OpenSolaris, Apress, 2009, pp. 102-124
[12] F. J. T. Fábrega, J. D. Guttman: Copy on Write, 1995.
[13] P. J. Dawidek and M. K. McKusick: Porting the Solaris ZFS File System to the FreeBSD Operating System, AsiaBSDCon, Vol 31, June 2007, pp. 19-24
[14] "FUSE Design Document" Opensolaris.org, retrieved on 2009-08-09, http://cn.opensolaris.org/os/project/fuse/Documentation/FUSE_Design_Doc_0_6.pdf.
[15]O. Rodeh: B-trees, Shadowing, and Clones, IBM Haifa. http://www.cs.tau.ac.il/~ohadrode/papers/btree_TOS.pdf
[16]А. Пешеходов :Architecture and Implementation of btrfs", retreived on 2009-07-01. (in Russian). http://www.filesystems.nm.ru/my/btrfs.pdf
[17]"Project CRFS" oracle.com retrieved on 2009-08-08. http://oss.oracle.com/projects/crfs/
[18]А. Пешеходов :Architecture of ZFS", retreived on 2009-07-01. (in Russian).
http://lug.yaroslavl.ru/archive/files/zfs_arch.pdf
[19]http://www.sun.com/software/solaris/reference_resources.jsp#wp, retrieved on 2009-08-09