Linux File Systems

Linux supports a wide range of file systems, from older ones like **Ext2** to more modern, feature-rich options like **Btrfs** and **ZFS**. Each file system offers a unique set of features, including support for journaling (protecting file system integrity), compression, encryption, and snapshots. The choice of a file system can affect system performance, reliability, scalability, and ease of management. Selecting the right file system is key to ensuring your system operates efficiently and reliably.

Linux's open-source nature allows for flexibility in choosing the right file system, whether you're looking for performance, data integrity, fault tolerance, or ease of use. This diversity provides many options to tailor the system's storage to specific requirements.

- Ext4: the reliable all-rounder
- Btrfs: modern and feature-packed
- XFS: the workhorse
- ZFS: the file system to end all file systems

Ext4: the reliable all-rounder

Ext4 (Fourth Extended File System) is a high-performance, journaling file system widely used in Linux environments. It is an evolution of its predecessors, **Ext3** and **Ext2**. Ext4 is the default file system for many Linux distributions, as it is battle-tested and a good choice for general purpose desktop computing needs.

Key features of Ext4 include:

1. **Journaling**: Like Ext3, Ext4 is a journaling file system, meaning it keeps a log (or journal) of changes to the file system before committing them to disk. This helps prevent data corruption in the event of a system crash or power failure, as the journal can be used to roll back or replay incomplete operations. Additionally, Ext4 uses checksums in the journal to improve reliability. This feature has a side benefit: it can safely avoid a disk I/O wait during journaling, improving performance slightly.
2. **Larger File and Volume Support**: Ext4 supports file sizes between 16 - 256 TiB and volumes up to 1 EiB, making it suitable for modern storage needs. This was a significant improvement over Ext3, which, depending on block size, was limited to 2 - 32 TiB volumes and 16 GiB - 2 TiB file sizes.
3. **Extents**: Extents replace the traditional block mapping scheme used by ext2 and ext3. An extent is a range of contiguous physical blocks, improving large-file performance and reducing fragmentation.
4. **Backward Compatibility**: Ext4 is backward compatible with Ext3, meaning you can mount an Ext3 file system as Ext4 and take advantage of the newer features without needing to reformat the partition.
5. **Delayed Allocation**: Ext4 provides better performance compared to Ext3, thanks to delayed allocation, which improves write performance and multi-block allocation, optimizing space usage. This further helps to reduce fragmentation.
6. **Extended Attributes**: Ext4 supports extended attributes, allowing additional metadata to be attached to files (e.g. security labels, file system flags, user tags). This is useful for extending file properties with arbitrary application metadata to offer advanced file management.
7. **Online Defragmentation**: Unlike Ext3, Ext4 provides the ability to perform online defragmentation, which allows the file system to be defragmented while the system is running, reducing the impact on performance.

Ext4 is a solid choice for every-day computing needs. It's simple, robust and well tested in the field. However, its developers have stated that Ext4 is just a stop-gap until more modern file systems like Btrfs mature and reach the same performance and robustness levels as Ext4.

Btrfs: modern and feature-packed

Btrfs (B-tree File System) is a modern, copy-on-write (COW) file system for Linux designed to address the limitations of older file systems like Ext4. It provides advanced features such as snapshots, data integrity verification, built-in volume management, and more. Btrfs aims to combine the functionalities of traditional file systems and logical volume managers, making it a versatile and powerful tool for managing storage.

Key features of Btrfs include:

1. **Copy-on-Write (COW)**: Btrfs uses copy-on-write (COW) for data and metadata, meaning that when data is modified, the changes are written to new locations on disk instead of overwriting the original data. This approach ensures that the previous version of the data is preserved, which is crucial for features like snapshots and data integrity. It also makes copies of data on the same volume instantaneous, because both refer to the same block of data on disk. COW provides benefits such as *atomic writes*, where operations are either fully completed or not done at all, reducing the risk of data corruption.
2. **Snapshots**: Snapshots are read-only or read-write copies of the file system at a specific point in time. Snapshots are efficient and fast because they share common data blocks until changes to the data on the disk are made. Snapshots are ideal for backup, system rollback, and testing purposes. They allow you to capture the state of the file system and revert back to it later if necessary. With certain system configurations it is even possible to boot into a snapshot.
3. **Data Integrity and Checksumming**: Btrfs provides checksumming for both data and metadata. Every block of data and metadata is verified with a checksum when read or written. If corruption is detected (e.g. due to hardware failure), Btrfs can attempt to recover the data using its checksums. This feature significantly improves the file system's reliability, especially in environments where data integrity is critical.
4. **Built-in Volume Management**: Btrfs includes volume management features natively, which means it can handle multiple physical devices and manage them as a single logical volume. This feature eliminates the need for external volume managers like LVM (Logical Volume Manager). It supports RAID-like configurations (RAID 0, RAID 1, RAID 10) natively, allowing users to create storage pools with redundancy and striping.
5. **Compression and Deduplication**: Btrfs natively supports transparent compression, allowing data to be stored in a compressed format to save space. Several algorithms (such as LZO and zstd) are available for use. Deduplication (removal of duplicate data) can also be performed, making it ideal for environments with large amounts of redundant data where storage space is precious.

6. **Dynamic Sizing and Subvolumes**: Btrfs allows the dynamic resizing of file systems, making it easy to increase or decrease the size of volumes as needed, without requiring reformatting or complex migrations. Subvolumes are logical divisions of a file system that can be treated like separate file systems. Each subvolume can have its own set of snapshots, making them useful for organizing data, creating backups, or managing different applications in isolation. Subvolumes can also be used to install multiple different operating systems on the same file system without getting into the way of one another.
7. **Online Defragmentation**: Btrfs supports online defragmentation, meaning that it can reorganize and defragment the file system while the system is running, which can help improve performance over time, especially on systems with high churn of small files.
8. **Self-healing**: In combination with its checksumming capabilities, Btrfs is able to self-heal data when used with RAID configurations. For example, in a RAID 1 setup, if one disk has corrupted data, Btrfs can detect the issue and restore the correct data from the mirrored copy.

The rich feature set of Btrfs makes it a modern and versatile choice for data storage on Linux. Built-in volume management at the file system level eliminates having to think about partition layouts and sizes, while the RAID capabilities make an underlying software RAID layer redundant. Snapshots allow for quick rollbacks to a previous state of the system without wasting precious space. Transparent compression and deduplication maximize efficient use of storage space. When used effectively, Btrfs is an excellent and future-proof choice.

However, because Btrfs is still a relatively new file system, some features are still not considered ready for daily use, e.g. integrated RAID 5/6 support currently being considered unstable. It also often ranks worse than other file systems in areas where high throughput is important, such as database applications or as backing storage for virtual machines. The complex feature set may also pose a high learning curve for people unfamiliar with its concepts.

XFS: the workhorse

XFS is a high-performance journaling file system created by Silicon Graphics, Inc. for their IRIX workstations. XFS is particularly proficient at parallel I/O due to its allocation group based design. This enables extreme scalability of I/O threads, filesystem bandwidth, file and filesystem size when spanning multiple storage devices.

Key features of XFS include:

1. **Journaling**: Like other journaling file systems, XFS keeps a log (journal) of file system changes before they are written to disk. This helps ensure data integrity in the event of a system crash or power failure, as the file system can use the journal to recover incomplete operations.
2. **Scalability**: XFS is designed to handle large volumes and large files efficiently. It supports file systems as large as 16 EiB and individual file sizes up to 8 EiB, making it suitable for modern data centers and high-performance applications that require managing vast amounts of data.
3. **Parallel I/O**: XFS is optimized for parallel I/O operations, which improves performance for applications with heavy read/write demands like databases or large media files.
4. **Extents-based Allocation**: XFS uses extents (contiguous blocks of storage) to manage files, which reduces fragmentation and improves throughput when working with large files.
5. **Online Defragmentation**: XFS supports online defragmentation, allowing for defragmentation of the file system without having to unmount it. This is particularly useful for maintaining performance on systems with minimal downtime.
6. **Dynamic Allocation**: XFS supports dynamic inode allocation, meaning that inodes (the data structures used to store metadata about files) can be created as needed, improving the file system’s efficiency when dealing with a large amount of files.
7. **Data Integrity**: In addition to journaling, XFS also provides checksumming of metadata, which helps ensure data integrity and detect corruption. This is important for environments where data reliability is critical.
8. **Efficient Space Management**: XFS is known for its efficient space management. It uses techniques like delayed allocation and aggressive pre-allocation to minimize disk fragmentation and optimize disk space usage

XFS is the file system of choice when high throughput and performance are important. It's the file system of choice for storage arrays found in servers and NAS systems. It is a good choice for the backing storage for virtual machine environments and high availability database applications. While it lacks some of the advanced features found in other file systems (like snapshots or compression), it excels in areas such as scalability, performance, and data integrity. XFS is well-suited for applications requiring fast, reliable access to large datasets.

Areas that XFS does not perform well in is handling large amounts of small files, as it stays true to its data center and workstation roots. Another downside of XFS is that while it's easy to grow the file system, it is not (yet) possible to shrink it. XFS recovery, while reliable, can be more complex than other file systems in cases of severe corruption, and its repair tools are not as user-friendly as those for Ext4.

ZFS: the file system to end all file systems

ZFS (Zettabyte File System) is a high-performance, advanced file system originally developed by Sun Microsystems (now part of Oracle) for the **Solaris** operating system. ZFS combines both a file system and a volume manager, providing integrated management of storage. It is known for its robust features, such as data integrity, high scalability, fault tolerance, and ease of management, making it a popular choice in enterprise environments, as well as for personal use by those requiring advanced features in Linux or FreeBSD.

Key features of ZFS include:

1. **Data Integrity and Checksumming**: One of ZFS's standout features is its data integrity capabilities. It uses checksums for all data and metadata. Every block of data is verified when read and written, ensuring that corruption is detected and corrected. If corruption occurs (e.g. from disk errors), ZFS can automatically attempt to repair it using redundant copies or RAID-like configurations.
2. **Copy-on-Write (COW)**: ZFS utilizes copy-on-write, meaning that when data is modified, it is not overwritten in place. Instead, new data is written to a different location, and once the write operation is complete, the system updates its pointers to the new location. This ensures that the file system always remains in a consistent state, even in the event of a power failure or system crash.
3. **Snapshots**: ZFS supports snapshots, which are read-only or read-write copies of the file system at a particular point in time. Snapshots are efficient in terms of storage because they only store changes made after the snapshot was taken, not the entire data set. This makes snapshots an ideal tool for rollbacks.
4. **Pooled Storage**: Unlike traditional file systems, ZFS combines file system and volume management, allowing it to create storage pools (called *Zpools*). Zpools abstract physical devices into a single logical pool of storage and data is automatically distributed across multiple devices. This allows ZFS to easily manage devices of varying sizes, providing better flexibility and redundancy. Additionally, pools can have cache devices, combinging the capacity of spinning hard drives with the speed of solid state drives to increase performance.
5. **RAID-Z**: ZFS includes its own RAID-like functionality, called RAID-Z, which provides data redundancy and improved performance. RAID-Z can be thought of as a more advanced version of RAID 5, with better protection against data loss due to disk failures. RAID-Z offers different configurations: RAID-Z1 (single parity), RAID-Z2 (double parity), and RAID-Z3 (triple parity), providing varying levels of fault tolerance.
6. **Compression**: ZFS supports inline compression, meaning that data is compressed as it is written to disk. This can significantly reduce storage space usage. ZFS supports multiple

compression algorithms (e.g. LZ4, ZLE, zstd). Compression is transparent and data is automatically decompressed when read.
7. **Deduplication**: ZFS offers deduplication, which ensures that duplicate data is stored only once. This feature can be particularly useful in environments where many identical files are stored (e.g. backups or virtual machine images). However, deduplication can be resource-intensive and should be used with caution on systems with limited memory.
8. **High Scalability**: ZFS is highly scalable, supporting file systems and storage pools up to 256 trillion YiB (2^{128} bytes) in size with a maximum file size of 16 EiB. It can handle vast amounts of data and large numbers of files, making it ideal for enterprise-level storage solutions and large-scale data environments.
9. **Self-Healing**: ZFS provides self-healing capabilities. In case of data corruption (detected through checksums), ZFS can automatically repair data by accessing redundant copies (e.g. from RAID-Z). This ensures data reliability without requiring manual intervention.

With features like data integrity, compression, snapshots, RAID-Z, and self-healing, ZFS offers exceptional storage management capabilities, making it ideal for large-scale enterprise environments and applications requiring high availability and data protection. However, its resource requirements, complexity, and limited support on non-Solaris platforms makes it diffucult to use outside of these environments. For desktop computers, other file systems are generally a better choice. For those who need enterprise-grade features and are willing to manage its complexity, ZFS is the uncontested choice.