Anatomy of Linux journaling file systems

Journaling today and tomorrow

Skill Level: Intermediate

M. Tim Jones (mtj@mtjones.com)
Consultant Engineer
Emulex Corp.

04 Jun 2008

In recent history, journaling file systems were viewed as an oddity and thought of primarily in terms of research. But today, a journaling file system (ext3) is the default in Linux®. Discover the ideas behind journaling file systems, and learn how they provide better integrity in the face of a power failure or system crash. Learn about the various journaling file systems in use today, and peek into the next generation of journaling file systems.

You can define journaling file systems in many ways, but let's get right to the point. Journaling file systems are for people who tire of watching the boot-time `fsck`, or file system consistency check process. (Journaling file systems are also for anyone who likes the idea of a fault-resilient file system.) When a system using a traditional, non-journaling file system is improperly shut down, the operating system detects this and performs a consistency check using the `fsck` utility. This utility scans the file system (which can take a considerable amount of time) and fixes any issues that can be safely corrected. In some cases, the file system can be in such bad shape that the operating system boots into single user mode to allow the user to further the repair process.

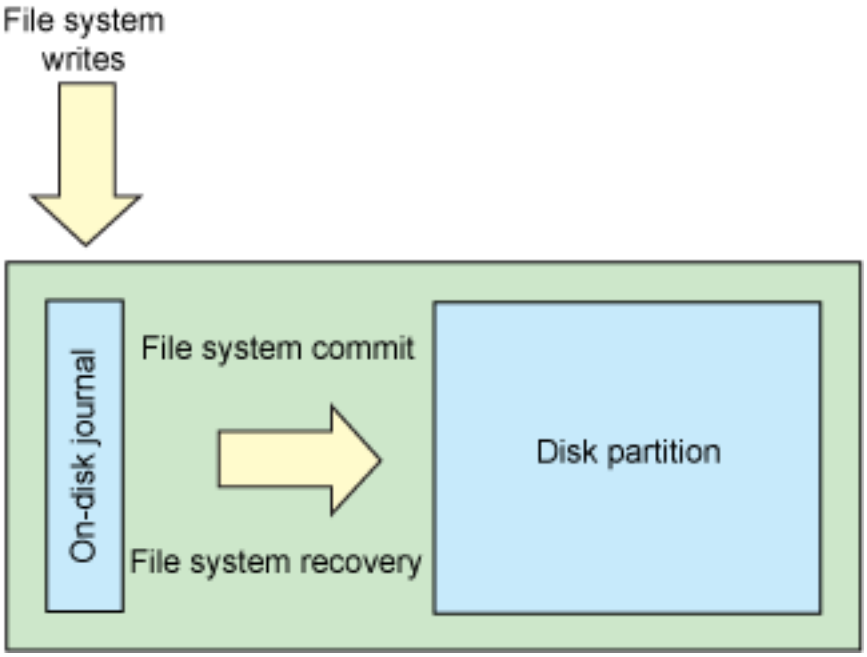
Pronouncing fsck
To add insult to injury, the `fsck` process can be initiated automatically by the operating system at mount time to ensure that the file system metadata is correct (even if no corruption is detected). Therefore, removing the need for file system consistency checks is an obvious area for improvement.

So, now you know for whom journaling file systems were created, but how do they obviate the need for `fsck`? In general, journaling file systems avoid file system corruption by maintaining a journal. The journal is a special file that logs the changes destined for the file system in a circular buffer. At periodic intervals, the journal is committed to the file system. If a crash occurs, the journal can be used as a checkpoint to recover unsaved information and avoid corrupting file system metadata.

To sum up, journaling file systems are fault-resilient file systems that use a journal to log changes before they're committed to the file system to avoid metadata corruption (see Figure 1). But like many Linux solutions, more than one option is available to you. Let's take a short walk through journaling file system history, and then review the file systems available and how they differ.

What is metadata?
Metadata refers to the managing structures for data on a disk. Metadata represents file creation and removal, directory creation and removal, growing a file, truncating a file, and so on.

Figure 1. A typical journaling file system



Linux journaling file system history

The first journaled file system was the IBM® Journaled File System (JFS). JFS was first released in 1990, but the current version supported in Linux is the later-developed JFS2. In 1994, Silicon Graphics introduced the high-performance XFS for the IRIX operating system. XFS was ported into Linux in 2001. The Smart

File System (SFS) was developed for the Amiga in 1998, but then released under the GNU Lesser General Public License (LGPL) and supported under Linux in 2005. The most commonly used journaling file system, ext3fs (or *third extended file system*) is an extension of ext2 with journaling capabilities. The ext3fs has been supported in Linux since 2001. Finally, the ReiserFS journaling file system blazed many new trails when it was introduced and found wide adoption. Its evolution is now diminished because of the legal issues of its original author.

Variations on journaling

More in Tim's Anatomy of... series on developerWorks

- Anatomy of Linux flash file systems
- Anatomy of Security-Enhanced Linux (SELinux)
- Anatomy of real-time Linux architectures
- Anatomy of the Linux SCSI subsystem
- Anatomy of the Linux file system
- Anatomy of the Linux networking stack
- Anatomy of the Linux kernel
- Anatomy of the Linux slab allocator
- Anatomy of Linux synchronization methods
- All of Tim's *Anatomy of...* articles
- All of Tim's articles on developerWorks

Journaling file systems use a journal to buffer changes to the file system (which is also used in crash recovery) but can use different strategies for when and what is journaled. Three of the most common strategies are writeback, ordered, and data.

In *writeback mode*, only the metadata is journaled, and the data blocks are written directly to their location on the disk. This preserves the file system structure and avoids corruption, but data corruption can occur (for example, if the system crashes after the metadata is journaled but before the data block is written). To solve this problem, you can use ordered mode. *Ordered mode* is metadata journaling only but writes the data before journaling the metadata. In this way, data and file system are guaranteed consistent after a recovery. Finally, data journaling can also be supported. In *data mode*, both metadata and data are journaled. This mode offers the greatest protection against file system corruption and data loss but can suffer from performance degradation, as all data is written twice (first to the journal, then to the disk).

The journal commit policy can also differ in the various approaches. For example, is the journal committed when it nears full, or through a timeout?

Journaling file systems today

Today, several journaling file systems are actively used. Each has its own benefits and disadvantages. Here are the four most popular journaling file systems available today.

JFS2

JFS2 (also called the *enhanced journaled file system*) was the first journaled file system and has many years of use in the IBM AIX® operating system before being ported to Linux. JFS2 is a 64-bit file system that, although based on the original JFS, was enhanced to be more scalable and support multi-processor architectures.

JFS2 supports ordered journaling for high performance with sub-second file system recovery. JFS2 also provides extent-based file allocation for performance. *Extent-based allocation* means that instead of allocating a single block, a contiguous set of blocks is allocated. Because these blocks are contiguous on the disk, there's better read and write performance for them. An additional advantage to extent-based allocation is minimization of metadata management. Allocating space by block means metadata updates per block. Using an extent, metadata is only updated for the extent (which can represent many blocks).

JFS2 also makes use of B+ trees for fast directory lookups as well as managing extent descriptors. JFS2 has no internal journal commit policy but instead relies on the timeout of the kupdate daemon.

XFS

XFS is one of the other early journaling file systems that was originally developed by Silicon Graphics for the IRIX operating system in 1995. XFS was ported to Linux in 2001 and, therefore, was already mature and reliable.

XFS supports full 64-bit addressing and provides very high performance using B+ trees both for directories and for file allocation. XFS also uses extent-based allocation with variable block size support (from 512 bytes to 64KB). Along with extents, XFS uses delayed allocation, in which allocation of disk blocks is delayed until the blocks are to be written to disk. This functionality improves the chances that sequential disk blocks are allocated, because the total number needed will be known.

Other interesting properties of XFS are guaranteed rate input/output (I/O—through bandwidth reservation for file system users) and direct I/O, where data is copied

directly between the disk and the user space buffer (rather than being staged through multiple buffers). XFS uses the writeback journaling policy.

Third extended file system (ext3fs)

The third extended file system (ext3fs) is the most popular journaling file system and is the evolution of the popular ext2 file system. Ext3fs is actually compatible with ext2fs, because ext3fs uses the same structure from ext2fs and simply adds a journal. It's even possible to mount an ext3fs partition as an ext2 file system or convert an ext2 file system to an ext3 file system (using the `tune2fs` utility).

Ext3fs permits three types of journaling (writeback, ordered, and data) but uses ordered as the default mode. The journal commit policy is configurable but by default is based on filling 1/4 of the journal or through timeout of one of the commit timers.

One of the primary disadvantages of ext3fs is that it was not designed from the ground up as a journaling file system. Being based on ext2fs, it lacks some of the more recent advanced features found in other journaling file systems (such as extents). It also typically scores worse in performance when compared to ReiserFS, JFS, and XFS but requires less CPU and memory than competing solutions.

ReiserFS

What is tail packing?
In many cases, files exist whose size is less than the size of a logical block. Rather than waste space allocating a logical block for each small file (called a *tail*), multiple files are packed within a single logical block. This has been found to increase disk space by 5% over competing file systems (with a performance penalty).

ReiserFS is a journaling file system that was developed from the ground up with journaling in mind. ReiserFS was introduced in 2001 in the mainline 2.4 kernel (the first journaling file system to be adopted by Linux). The default method for journaling is ordered and supports online resizing to grow the file system. ReiserFS also included *tail packing* to dynamically reduce fragmentation. For smaller files, ReiserFS tends to be much faster than ext3fs (when tail packing is enabled).

ReiserFS (also called ReiserFS v3) includes many modern features, such as B+ trees. The fundamental format of the file system is based on a single B+ tree, which makes search operations efficient and very scalable. The commit policy depends on the journal size but is based on the number of blocks to commit.

ReiserFS was plagued by several issues—most recently, by the legal troubles of its author (see Resources for details).

Journaling file systems tomorrow

Now that you've seen the journaling file systems of today (and yesterday), let's look at what's ahead (and what's not).

Reiser4

After successfully getting ReiserFS merged into the Linux kernel and adopted by many Linux distributions, Namesys (the company behind ReiserFS) began work on a new journaling file system. Reiser4 was designed from scratch as a new journaling file system with many advanced features.

Resier4 was designed for better journaling through the use of wandering logs and delayed allocation of blocks until the journal is committed (as was done in XFS). Reiser4 was also designed with a flexible plug-in architecture (to support capabilities such as compression and encryption) but was rejected by the Linux community, as these capabilities were viewed best in the virtual file system (VFS).

Since the conviction of the owner of Namesys, all commercial activity on Reiser4 has stopped.

Fourth extended file system

The fourth extended journaling file system (ext4fs) is the evolution of ext3fs. The ext4 file system is designed as a backward- and forward-compliant replacement for ext3fs but with many new advanced features (some of which break the compatibility). This means that you can mount an ext4fs partition as ext3fs or vice versa.

First, ext4fs is a 64-bit file system and is designed to support very large volumes (1 exabyte). It has also been designed to use extents, but if this is used, then compatibility with ext3fs is lost. Like XFS and Reiser4, ext4fs includes delayed allocation to allocate blocks on the disk only when needed (which reduces fragmentation). The contents of the journal are also checksummed to make the journal more reliable. Instead of the standard B+ or B* tree, ext4fs uses a variation of the B tree, called the *H tree,* which allows much larger subdirectories (ext3 was limited to 32KB).

Although the delayed allocation method reduces fragmentation, over time, a large file system can become fragmented. An online defragmentation tool (e4defrag) has been developed to address this. You can use the tool to defragment individual files or an entire file system.

Another interesting difference between ext3fs and ext4fs is the date resolution for files. In ext3, the minimum resolution for timestamp was one second. Ext4fs is

looking toward the future: Where processor and interface speeds continue to increase, better resolution is needed. For this reason, the minimum timestamp resolution in ext4 is 1 nanosecond.

Ext4fs has been in the Linux kernel since 2.6.19 but is yet to be called stable. Development continues on this next generation; given its heritage, it will be the next generation in Linux journaling file systems.

Going further

Journaling file systems provide reliability and protect against corruption in the face of system crash or power loss. Additionally, the crash recovery time for journaling file systems is dramatically reduced compared to more traditional file system methods (such as those that rely on `fsck`). Development of new journaling capabilities continues to look to the future at new algorithms and structures as well as to the past, where features of JFS and XFS are incorporated. How journaling file systems will evolve in the future is unclear, but their usefulness is clear, and they are the new file system standard.

Resources

Learn

- The list of file systems on Wikipedia ranges from the earliest DEC file systems of the 1960s to the latest BufferFS from Oracle. To round out your file system knowledge, also check out this file system reading list, which covers a wide range of file system topics.
- JFS (and its successor, JFS2) were the earliest journaled file systems. They continue to be used in Linux and the AIX operating systems.
- XFS was the earliest journaling file system that focused on high performance. Learn more about the development and future of XFS at the SGI home page.
- The current leader in Linux journaling file systems (as far as deployments go) is the third extended file system (successor to the second extended file system). Read more about the transformation of ext2 to ext3 in the interesting paper, "Journaling the Linux ext2fs Filesystem" (PDF), or in this talk given by the ext3fs designer, Dr. Stephen Tweedie.
- The Reiser4 file system was the first journaling file system to be adopted into the mainline Linux kernel. You can also learn more about Reiser's internal structures and disk layout.
- On 28 April 2008, Hans Reiser (owner of Namesys, developer of Reiser file systems) was convicted for the murder of his estranged wife. Namesys has ceased to exist, and work on the Reiser4 file system has also stopped (although there is speculation about the future of Reiser4 in the Linux kernel).
- Tim's "Anatomy of the Linux file system" (developerWorks, Oct 2007) introduces you to the VFS and its major structures. The Linux VFS layer provides an abstraction using a common application program interface (API) to the various supported underlying file systems.
- The future of journaling file systems is ext4fs. The paper, "The new ext4 filesystem: current status and future plans" (PDF), along with the presentation, "Ext4: The Next Generation of Ext2/3 Filesystem" (PDF), provide a wealth of technical details for ext4fs. Finally, you can learn more about the development of ext4fs from the development wiki and also about the online defragmentation (PDF) approach.
- Read all of Tim's *Anatomy of...* articles on developerWorks.
- Read all of Tim's Linux articles on developerWorks.
- In the developerWorks Linux zone, find more resources for Linux developers, and scan our most popular articles and tutorials.
- See all Linux tips and Linux tutorials on developerWorks.

- Stay current with developerWorks technical events and Webcasts.

Get products and technologies

- Order the SEK for Linux, a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- With IBM trial software, available for download directly from developerWorks, build your next development project on Linux.

Discuss

- Get involved in the developerWorks community through blogs, forums, and podcasts.

About the author

M. Tim Jones
M. Tim Jones is an embedded firmware architect and the author of *Artificial Intelligence: A Systems Approach, GNU/Linux Application Programming* (now in its second edition), *AI Application Programming* (in its second edition), and *BSD Sockets Programming from a Multilanguage Perspective.* His engineering background ranges from the development of kernels for geosynchronous spacecraft to embedded systems architecture and networking protocols development. Tim is a Consultant Engineer for Emulex Corp. in Longmont, Colorado.

Trademarks