ext4

From Wikipedia, the free encyclopedia

The **ext4**, or **fourth extended filesystem** is a journaling file system in development, designed as a backwards-compatible replacement of the ext3 filesystem.

Contents

ext4	
Developer	Mingming Cao, Andreas Dilger, Alex Tomas, Dave Kleikamp, Theodore Ts'o, Eric Sandeen, Sam Naghshineh, others
Full name	Fourth extended file system
Introduced	**Stable:** Yet to be released **Unstable:** October 10, 2006 (Linux 2.6.19)
Partition identifier	0x83 (MBR) EBD0A0A2-B9E5-4433-87C0-68B6B72699C7 (GPT)
Structures	
Directory contents	Table, htree
File allocation	Extents/Bitmap
Bad blocks	Table
Limits	
Max file size	1 EiB
Max number of files	4 billion (specified at filesystem creation time)
Max filename length	256
Max volume size	1 EiB
Allowed characters in filenames	All bytes except NUL and '/'
Features	
Dates recorded	modification (mtime), attribute modification (ctime), access (atime), delete (dtime), create (crtime)
Date range	December 14, 1901 - April, 25 2514

History

Ext4 began as a fork of ext3. The developers of ext3 had designed a set of new features for ext3. Some of those were implemented in ext3 but others were not, so that only newer versions would be able to read the file system structure created with the new features. Patches to implement those features were proposed[1] on the Linux kernel development mailing list, but some developers argued that it would be better to keep ext3 stable, and continue the development in a forked version to be called ext4.[2] The ext3 developers argued that maintaining a separate code base for the new file system would be harder than a shared code

base for both file systems. But arguments for forking, including ones by Linus Torvalds[3], convinced the ext3 developers to start a new file system.

On June 28, 2006 Theodore Ts'o, the ext3 maintainer, announced[4] the new plan of development. A preliminary development snapshot of ext4 was included in version 2.6.19 of the Linux kernel which was released on November 29, 2006.

The file system is currently marked as developmental and is titled "ext4dev".[5] It is considered unstable and so is not recommended for use in production environments, since data corruption is possible in this early stage of development.

Date resolution	Nanosecond
Forks	No
Attributes	extents, noextents, mballoc, nomballoc, delalloc, nodelalloc, data=journal, data=ordered, data=writeback, commit=nrsec, orlov, oldalloc, user_xattr, nouser_xattr, acl, noacl, bsddf, minixdf, bh, nobh, journal_dev
File system permissions	POSIX
Transparent compression	No
Transparent encryption	No
Single Instance Storage	No
Supported operating systems	Linux

Features

Large filesystem

The ext4 filesystem can support volumes with sizes up to 1 exabyte. Since Linux kernel version 2.6.25, it also supports files as large as the file system.

Extents

Extents are introduced to map a range of contiguous physical blocks into a single descriptor. A single extent can map up to 128MiB of contiguous space with a 4KiB block size.[6]

Backward compatibility

The ext4 filesystem is backward compatible with ext3, making it possible to mount an ext3 filesystem as ext4 (using the “ext4dev” filesystem type, “ext4” after final release).

Forward compatibility

The ext4 file system is forward compatible with ext3, that is, it can be mounted as an ext3 partition (using “ext3” as the filesystem type when mounting). However, if the ext4 partition uses extents (a major new feature of ext4), then the ability to mount the file system as ext3 is lost. Extents were enabled by default in the 2.6.23 kernel. Previously, the “extents” option was explicitly required (e.g. `mount /dev/sda1 /mnt/point -t ext4dev -o extents`).

Persistent pre-allocation

The ext4 filesystem allows for pre-allocation of on disk space for a file. The current

methodology for this on most file systems is to write the file full of 0's to reserve the space when the file is created (although XFS has an ioctl to allow for true pre-allocation as well). This method would no longer be required for ext4; instead, a new preallocate() system call was added for use by filesystems, including ext4 and XFS, that have this capability. The space allocated for files such as these would be guaranteed and would likely be contiguous. This has applications for media streaming and databases.

Delayed allocation

Ext4 uses a "delayed allocation" feature to delay block allocation as long as possible. During this delay pending writes will only change the free space counter. This improves performance and reduces fragmentation by virtue of improving block allocation decisions based on the actual file size.

Break 32000 subdirectory limit

In ext3 the number of subdirectories that a directory can contain is limited to 32000. This limit has been raised to 64000 in ext4, and with the "dir_nlink" feature it can go beyond this (although it will stop increasing the link count on the parent). To allow for continued performance given the possibility of much larger directories, htree indexes (a specialized version of a Btree) is turned on by default in ext4. This feature is implemented in 2.6.23. htree is also available in ext3 when the dir_index feature is enabled.

Journal checksumming

Ext4 uses checksums in the journal to improve reliability, since the journal is one of the most used files of the disk. This feature has a side benefit; it can safely avoid a disk IO wait during the journaling process, improving performance slightly. The technique of journal checksumming was inspired by research from Wisconsin on IRON File Systems (Section 6, called "transaction checksums").[7]

Online defragmentation

Ext4 will eventually also have an online defragmenter. Even with the various techniques used to avoid it, a long lived file system does tend to become fragmented over time. Ext4 will have a tool which can defragment individual files or entire file systems. Currently a work in progress (unreleased) version of the defragmentation program is located here (http://www.kernel.org/pub/linux/kernel/people/tytso/ext4-patches/LATEST/broken-out/ext4-online-defrag-command.patch) (compilable if you comment out the comments and give the file a .c extension)

Faster file system checking

In ext4, unallocated block groups and sections of the inode table are marked as such. This enables e2fsck to skip them entirely on a check and greatly reduce the time it takes to check a file system of the size ext4 is built to support. This feature is implemented in version 2.6.24 of the Linux kernel.

Multiblock allocator

ext4 will have a multiblock allocator. This allows many blocks to be allocated to a file in

single operation and therefore a better decision can be made on finding a chunk of free space where all the blocks can fit. The multiblock allocator is active when using O_DIRECT or if delayed allocation is on. This allows the file to have many dirty blocks submitted for writes at the same time, unlike the existing kernel mechanism of submitting each block to the filesystem separately for allocation.

Improved timestamps

As computers become faster in general and specifically Linux becomes used more for mission critical applications, the granularity of second-based timestamps becomes insufficient. To solve this, ext4 will have timestamps measured in nanoseconds. This feature is currently implemented in 2.6.23. In addition, 2 bits of the expanded timestamp field are added to the most significant bits of the seconds field of the timestamps to defer the year 2038 problem for an additional 500 years.

Support for date-created timestamps is added in ext4. But as Theodore Ts'o points out, while adding an extra creation date field in the inode is easy (thus technically enabling support for date-created timestamps in ext4), modifying or adding the necessary system calls, like stat() (which would probably require a new version), and the various libraries that depend on them (like glibc) is not trivial and would require the coordination of many different projects[8]. So even if ext4 developers implement initial support for creation date timestamps, this feature will not be available to user programs for now.[8]

See also

- List of file systems
- Comparison of file systems

References

1. ^ LKML: Mingming Cao: [RFC 0/13] extents and 48bit ext3 (http://lkml.org/lkml/2006/6/8/270)
2. ^ LKML: Jeff Garzik: Re: [RFC 0/13] extents and 48bit ext3 (http://lkml.org/lkml/2006/6/8/296)
3. ^ LKML: Linus Torvalds: Re: [Ext2-devel] [RFC 0/13] extents and 48bit ext3 (http://lkml.org/lkml/2006/6/9/183)
4. ^ LKML: "Theodore Ts'o": Proposal and plan for ext2/3 future development work (http://lkml.org/lkml/2006/6/28/454)
5. ^ "OLS 2007: Ext4 Paper (https://ols2006.108.redhat.com/2007/Reprints/mathur-Reprint.pdf) ". Retrieved on 2007-08-10.
6. ^ "Ext4 overview (https://ols2006.108.redhat.com/2007/Reprints/mathur-Reprint.pdf) ". Retrieved on 2008-01-15.
7. ^ Vijayan Prabhakaran, *et al*. "*IRON File Systems (http://www.cs.wisc.edu/wind/Publications/iron-sosp05.pdf)* " (PDF). CS Dept, University of Wisconsin.
8. ^ ***a b*** "Theodore Ts'o answer on creation time stamps for ext4 (http://osdir.com/ml/file-systems.ext3.user/2006-10/msg00015.html) ".

External links

- Theodore Ts'o's discussion on ext4 (http://kerneltrap.org/node/6776)
- First benchmarks of ext4 (http://www.linuxinsight.com/first_benchmarks_of_the_ext4_file_system.html)

- Ext4 Development Wiki (http://ext4.wiki.kernel.org)
- "Ext4 block and inode allocator improvements" (http://ols.fedoraproject.org/OLS/Reprints-2008/kumar-reprint.pdf) (materials from Ottawa Linux Symposium 2008)
- "The new ext4 filesystem: current status and future plans" (https://ols2006.108.redhat.com/2007/Reprints/mathur-Reprint.pdf) (materials from Ottawa Linux Symposium 2007)
- "ext4 online defragmentation" (https://ols2006.108.redhat.com/2007/Reprints/sato-Reprint.pdf) (materials from Ottawa Linux Symposium 2007)
- “Ext4: The Next Generation of Ext2/3 Filesystem” (http://www.usenix.org/event/lsf07/tech/cao_m.pdf)