

ext4 File System: Introduction and Benchmarks

Jeffrey B. Layton

Destined to become the default file system for the more popular Linux distributions, ext4 is out of experimental mode and gearing up for production environments. Here's what you need to know.

If you have spent enough time around Linux it's almost certain you know about the file systems ext2 and ext3, and have probably heard of ext4. Get ready to hear some more.

On October 11, 2008, the "experimental" label for ext4 was removed. While this doesn't necessarily mean that you should change all of your file systems over to ext4 immediately, it does mean that you should consider using ext4 moving forward. With the "experimental" label gone and openSUSE (among others) considering it for the default file system in a late-2009 release, it's a good time to review ext4 so you have a solid working knowledge of what it is and what features it brings to the table.

A Bit of Linux History

If you go back to the Linux days of yore, you might remember that early distributions used the Minix File System. While MinixFS (my abbreviation) allowed Linux to get up and functioning quickly because a new file system did not have to be developed, it had a few limitations. It used 16-bit offsets internally resulting in a maximum file size limit of 64 Megabytes (MB) and only allowed file names of 14 characters. It's pretty obvious that this wasn't an ideal file system so work quickly began on the Extended File System (ext) by Remy Card and others.

ext was added to Linux 0.96c. It was able to handle file systems up to 2GB with files up to 255 characters. But there were still some issues with the file system so work on the second-generation version of ext, called ext2, was begun. It quickly became the most popular file system in Linux with a 4TB file system limit, a 2GB maximum file size, files with up to 255 characters, and 10^{18} files. To this day, you can still use ext2 and it's likely to be used for many years to come.

But just like everything in Linux, the ext2 file system was not standing still. Stephen Tweedie was evolving ext2 by adding, among other things, a journaling capability. Journaling improves the reliability of the file system and eliminates the need to check the file system after an unclean shutdown. In addition to journaling, the ability to resize the file system while it was on-line was added. Also, since, 64-bit computing was coming quickly, the b-tree algorithm was replaced with h-trees, allowing a larger number of files in a single directory.

Ext3 quickly became, arguably, the most popular file system in Linux. One attribute that contributed to it's popularity is that you could upgrade from ext2 to ext3 very easily (basically it just added a journal to the existing ext2 file system). So you didn't lose any data in the upgrade process from ext2 to ext3. By adding a journal to ext2 you increased it's reliability and also significantly reduced the need for a file system check (fsck) in the event of an unclean umount.

However, ext3 still has limitations that people were not happy with. The biggest complaints were the size of the file systems that is limited to 16TB and the performance was not on-par with other file systems such as XFS and JFS. The first complaint, the limited size of the file system, is perhaps the biggest complaint given that fact you can buy 1.5TB SATA drives and soon will be able to buy 2TB drives. It's pretty easy to create a simple RAID system in your home system that hits the 16TB limit. But there are other disadvantage as well.

Enter ext4, Stage Left

In 2006, the uber Linux developer, Theodore Ts'o, who was, at the time, the ext3 maintainer, began work on ext4. Unlike ext3, which just added some features to ext2 while keeping the on-line format and approach of ext2, ext4 is a fork of ext3 that is a deep code change affecting the data structures

ext4 File System: Introduction and Benchmarks

Jeffrey B. Layton

used in ext4 to make it a better file system - faster, more reliable, more features, better code, etc. Ext4 brought ext3 into the world of 64-bits allowing individual files of 16TB (assuming 4KB blocks), as well file systems of 1 Exabyte (EB) by using 48-bit data structures. One EB is the same as 1,048,576 Terabytes (TB).

While past predictions have been wrong about the amount of memory we would need (640KB) as well as storage, it is likely that our home machines won't get to 1 EB for a long time. But just in case, ext4 is set to go to 64-bits but the surgery to get there is likely to be deep enough to require some fundamental changes in the file system.

From the perspective of many, one of the most positive features of ext4 is that it is backward compatible with ext2 and ext3, allowing you to take the ext2 or ext3 file systems, change a few options, and mount them as ext4 file systems. The existing data is not lost and ext4 will use the new data structures only on new data (pretty nifty feature if you ask me).

Additionally, there is a nice upgrade capability that will allow you to take an ext2 or ext3 file system and upgrade it to ext4 without a loss of data (but — as always! — back up your data just in case). However, ext4 has limited forward compatibility with ext3. That is you can't always take an ext4 file system and mount it using ext3 because the data structures are completely different.

The hard work that went into ext4 added new features such as, extents, journaling checksumming, block allocation, delayed allocation, faster fsck, on-line defragmentation, and larger directory sizes (up to 64,000 files). Let's look at a few of these:

Extents:

Extents are a feature that describe how the blocks are laid out on the drive in order to store the data for the file.

Ext3 (and ext2) use an indirect method of keeping track of the blocks used by a particular file. This means they have to keep track of every single block. For example, for a 100MB file, you have 25,600 4KB blocks. So for that file, ext3 has to keep track of all 25,600 blocks and how they are ordered.

Ext4 allows the blocks for a particular file to be stored as an extent. An extent is just a contiguous set of blocks. So the file system only has to store two bits of information, the starting block, and how many contiguous blocks are in the extent. Extents also help prevent file fragmentation improving performance because you are storing the data in contiguous blocks. Extents also help with file deletion because you have much less metadata information to change.

Journaling Checksumming:

One of the big developments in ext3 was the implementation of a journal. A journal is just a list of the changes that need to be done to a file system (e.g. reads, writes, deletes, etc.). So a file system just "plays" this journal to commit the changes to the file system. If there is a crash, the journal, which is stored on disk, is just "replayed" and the file system is brought into a consistent state. But don't forget that the journal is stored on the disk and is subject to disk failures.

Journaling Checksumming creates a checksum of the journal data so that ext4 can tell if the area of the disk where the journal is kept is failing or going corrupt. This improves reliability but can also improve performance because it allows faster commits of the journal compared to ext3.

Multi-block Allocation:

Ext3 allocates blocks for a file one at a time (typically using 4KB blocks). For very large files, the

ext4 File System: Introduction and Benchmarks

Jeffrey B. Layton

associated function that does the allocation will have to be called thousands of times. ext4 uses “multi-block allocation” which allows multiple blocks (hence the name) to be allocated during one function call. This can greatly improve the performance of ext4 relative to ext3, particularly for large files.

Delayed Allocation:

In ext3 and other traditional file systems, blocks are allocated as soon as they are needed by a write function. But, in reality they may not be needed right away because the data may be in cache for some time. So delayed allocation allows blocks to be allocated only when they are actually needed to write the data. This can improve the performance of ext4 because during that time the allocator can be optimizing the allocation of blocks to minimize fragmentation and improve performance. It also has great benefits when coupled with extents and multi-block allocations.

The first development snapshot of ext4 was added to the 2.6.19 kernel but was marked as experimental. The patches that marked ext4 as a stable file system were merged on Oct. 11, 2008 in the 2.6.28 kernel source (and the peasants rejoiced). The only bad spot as of the writing of this article is that you can't use ext4 as the boot partition because grub doesn't understand it (although that is being actively worked upon).

Recently, Fedora Core 11 has announced that ext4 will be the default file system. Also Ubuntu 9.0.4 will support ext4 but not necessarily use it as the default file system. I'm sure other distributions will enable it by default. Really, all you need is a kernel 2.6.28 or newer and a very recent version of e2fsprogs and you're off to the races.

Creating ext4 File Systems and Benchmarking

To get you started, let's take a fresh SATA II drive and show how to format ext4 and then take it for a spin and compare it to ext3.

Creating an ext4 file system from scratch is very easy (as easy as ext3). I will assume that you have a kernel that is ext4 capable (2.6.28 or greater) along with the latest version of e2fsprogs (see this link for even more help). For the examples that follow, I'm using a couple of Seagate ST3500641AS 500GB disks that appear as /dev/sda and /dev/sdb in my little home box (it's old enough that I'm embarrassed to say what it is). The box also has 1GB of memory and is running CentOS 5.2. I'm using a 2.6.28.7 kernel built by hand along with version 1.41.4 of e2fsprogs.

A simple way to start is by using,

```
% mke2fs -t ext4 /dev/sda1
```

This is a pretty easy way to create an ext4 file system but it assumes all of the defaults.

To learn a bit more about what defaults are in ext4 you can use tune2fs.

```
% tune2fs -l /dev/sda1 | more
```

Here's a quick sample of what the output might look like:

```
tune2fs 1.41.4 (27-Jan-2009)
```

```
Filesystem volume name:
```

```
Last mounted on:
```

```
Filesystem UUID:          9db1edc1-69bf-4644-be16-78262d6ffb3a
```

ext4 File System: Introduction and Benchmarks

Jeffrey B. Layton

```
Filesystem magic number: 0xEF53
Filesystem revision #: 1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index filetype
needs_recovery extent    flex_bg sparse_super large_file huge_file uninit_bg
dir_nlink extra_isize
Filesystem flags:        signed_directory_hash
Default mount options:   (none)
Filesystem state:        clean
Errors behavior:         Continue
Filesystem OS type:      Linux
Inode count:             30531584
Block count:             122096000
Reserved block count:    6104800
Free blocks:             120129098
Free inodes:             30531573
First block:             0
Block size:              4096
Fragment size:          4096
Reserved GDT blocks:     994
Blocks per group:        32768
Fragments per group:    32768
Inodes per group:        8192
Inode blocks per group:  512
Flex block group size:   16
Filesystem created:      Sun Mar 15 15:57:09 2009
Last mount time:         Sun Mar 15 16:06:53 2009
Last write time:         Sun Mar 15 16:06:53 2009
Mount count:             1
Maximum mount count:     33
Last checked:            Sun Mar 15 15:57:09 2009
Check interval:          15552000 (6 months)
Next check after:        Fri Sep 11 15:57:09 2009
Reserved blocks uid:     0 (user root)
Reserved blocks gid:     0 (group root)
First inode:             11
Inode size:              256
Required extra isize:    28
Desired extra isize:     28
Journal inode:           8
Default directory hash:  half_md4
Directory Hash Seed:    3aabc48e-6505-41f7-9c62-06b106fe975c
Journal backup:          inode blocks
```

Benchmarking ext3 and ext4

I'm using several benchmarks for testing ext3 and ext4 (benchmarking file systems is a pet peeve of mine, but I digress). For now, I will be testing just iозone since it seems to be a popular benchmark that can produce a great deal of information.

I ran two sets of tests on ext3 and ext4. The first test builds the file systems with the defaults. The second test takes some of Ted Ts'o's advice and changes the mount options. iозone produces a great deal of information but I chose to just show the results for read, reread, write, and rewrite for a 2GB file with a record length of 16,384 bytes. This shows the performance for a very large file (perhaps a KC and the Sunshine Band album in ogg format).

ext4 File System: Introduction and Benchmarks

Jeffrey B. Layton

The results have 4 columns: Ext3 (default), Ext4 (default), Ext3 (performance), and Ext4 (performance). The “performance” options are not intended to produce the optimal performance but to illustrate what kind of performance you could obtain with some fairly easy changes. The commands to make the four file systems are:

```
% mke2fs -t ext3 /dev/sda1
% mke2fs -t ext4 /dev/sdb1
% mke2fs -t ext3 /dev/sda1
% mke2fs -t ext4 /dev/sdb1
```

The difference for the “performance” options are the mount options. For ext3, the entry in /etc/fstab is,

```
/dev/sda1 /data_ext3 ext3 defaults,data=writeback,noatime 0 0
```

For ext4, the mount options are:

```
/dev/sdb1 /data_ext4 ext4
defaults,data=writeback,noatime,barrier=0,extents,journal_checksum 0 0
```

The mount options used both the “noatime” and the “data=writeback” options. Whether these options are acceptable or not is up to you. By default ext3 does not use barriers (for good or bad). Ext4 uses barriers by default. So to make things equal I disabled barriers on ext4.

Benchmark	Ext3 (default) MB/s	Ext4 (default) MB/s	Ext3 (performance) MB/s	Ext4 (performance) MB/s
Write	28.307	30.228	28.047	30.127
Rewrite	28.001	29.626	26.432	29.336
Read	55.791	108.701	105.565	109.889
Reread	55.765	108.884	105.156	109.600

One of the more interesting things you can take from the results is that the defaults for ext4 produce the top (or near the top) results for write, rewrite, read, and reread. Once all of the mount options are used with ext3, it achieved about the same performance of ext4 with or without the mount options.

The benchmarks may or may not be applicable to your situation but I do think they point out that the ext4 defaults produce very good performance relative to ext3. Performance being one of the goals in the development of ext4, I think the dev team hit the mark.

Summary

Ext3 is arguably the most popular file system in Linux. It has good enough performance and is very stable. But there have been a number of complaints about it, primarily in the areas of performance and overall capacity. Ext4 was really a fork of ext3 to add features. It was merged into the mainstream kernel a while ago but was marked as experimental. More recently, it was merged into the kernel to get more testing. Now, with the experimental label removed, ext4 has arrived and is ready for prime time.

For more info check out our interview with Ted Ts'o's, which points out that ext4 has some fantastic features and provides links to some great blogs about ext4 — particularly using ext4 on SSDs (Solid State Disks).

ext4 File System: Introduction and Benchmarks

Jeffrey B. Layton

Should you move immediately to ext4? While you can migrate an ext3 file system to ext4, it requires a kernel capable of supporting ext4 (2.6.28 or newer or a kernel that has ext4 back-ported to it) and a relative new version of e2fsprogs. The quick benchmark I presented shows that ext4 has great performance using the defaults.

So while an immediate leap to ext4 may not be necessary, if you are currently using ext3 you should seriously start considering it for your default file system in the future.

Links and articles that are very helpful in learning more about ext4.

Ted Ts'o's blog

Ext4 Wiki

Ext4 Howto

E2fsprogs

Ext4 at Kernel Newbies

Anatomy of Ext4 from IBM

IBM Tech Center paper on ext4

Phoronix Ext4 Benchmarks

Migrating to Ext4 from IBM