How many files can I put in a directory?

529

216

Does it matter how many files I keep in a single directory? If so, how many files in a directory is too many, and what are the impacts of having too many files? (This is on a Linux server.)

Background: I have a photo album website, and every image uploaded is renamed to an 8-hex-digit id (say, a58f375c.jpg). This is to avoid filename conflicts (if lots of "IMG0001.JPG" files are uploaded, for example). The original filename and any useful metadata is stored in a database. Right now, I have somewhere around 1500 files in the images directory. This makes listing the files in the directory (through FTP or SSH client) take a few seconds. But I can't see that it has any effect other than that. In particular, there doesn't seem to be any impact on how quickly an image file is served to the user.

I've thought about reducing the number of images by making 16 subdirectories: 0-9 and a-f. Then I'd move the images into the subdirectories based on what the first hex digit of the filename was. But I'm not sure that there's any reason to do so except for the occasional listing of the directory through FTP/SSH.

filesystems limit

edited Feb 28 '16 at 6:04

poolie
6,690 1 33 63

asked Jan 21 '09 at 18:58

Kip
68.2k 78 210 246

21 Answers

688

+50

FAT32:

- Maximum number of files: 268,173,300
- Maximum number of files per directory: 2^{16} - 1 (65,535)
- Maximum file size: 2 GiB - 1 without LFS, 4 GiB - 1 with

NTFS:

- Maximum number of files: 2^{32} - 1 (4,294,967,295)
- Maximum file size
  - Implementation: 2^{44} - 2^{6} bytes (16 TiB - 64 KiB)
  - Theoretical: 2^{64} - 2^{6} bytes (16 EiB - 64 KiB)
- Maximum volume size

ext2:

- Maximum number of files: 10^{18}
- Maximum number of files per directory: ~1.3×10^{20} (performance issues past 10,000)
- Maximum file size
  - 16 GiB (block size of 1 KiB)
  - 256 GiB (block size of 2 KiB)
  - 2 TiB (block size of 4 KiB)
  - 2 TiB (block size of 8 KiB)
- Maximum volume size
  - 4 TiB (block size of 1 KiB)
  - 8 TiB (block size of 2 KiB)
  - 16 TiB (block size of 4 KiB)
  - 32 TiB (block size of 8 KiB)

ext3:

- Maximum number of files: min(volumeSize / 2^{13}, numberOfBlocks)
- Maximum file size: *same as ext2*
- Maximum volume size: *same as ext2*

ext4:

- Maximum number of files: 2^{32} - 1 (4,294,967,295)
- Maximum number of files per directory: unlimited
- Maximum file size: 2^{44} - 1 bytes (16 TiB - 1)
- Maximum volume size: 2^{48} - 1 bytes (256 TiB - 1)

edited Nov 20 '18 at 7:19
hackingpro
3 3

answered Jan 21 '09 at 19:16
ISW
9,053 3 21 26

23 I assume these are the maximum number of files for the entire partition, not a directory. Thus, this information isn't too useful regarding the problem, because there'd be an equal number of files regardless of the method (unless you count directories as files). – strager Jan 21 '09 at 19:28

18 Since we're in 2012 now, I think its time to make clear that ext4 doesn't have any limit concerning the number of subdirectories. Also maximum filesize grew to 16 TB. Furthermore, the overall size of the filesystem may be up to 1 EB = 1,048,576 TB. – devsnd Jun 25 '12 at 23:13

7 Apparently, ext3 also has a limit of 60,000 files(or directories or links) per directory. I found out the hard way

7 hard filesystem limits do not answer the question "*Does it matter how many files I keep in a single directory?*" – Etki Dec 30 '16 at 10:30

176

I have had over 8 million files in a single ext3 directory. libc `readdir()` which is used by `find`, `ls` and most of the other methods discussed in this thread to list large directories.

The reason `ls` and `find` are slow in this case is that `readdir()` only reads 32K of directory entries at a time, so on slow disks it will require many many reads to list a directory. There is a solution to this speed problem. I wrote a pretty detailed article about it at: http://www.olark.com/spw/2011/08/you-can-list-a-directory-with-8-million-files-but-not-with-ls/

The key take away is: use `getdents()` directly -- http://www.kernel.org/doc/man-pages/online/pages/man2/getdents.2.html rather than anything that's based on libc `readdir()` so you can specify the buffer size when reading directory entries from disk.

edited Jun 9 '16 at 8:53

lord_t
2,056 2 21 47

answered Aug 11 '11 at 20:19

Ben
1,916 1 11 3

6 Interesting read! Can I ask in what situation you had 8 millions files in one directory? haha – Acheronfail Aug 3 '16 at 10:52

56

It depends a bit on the specific filesystem in use on the Linux server. Nowadays the default is ext3 with dir_index, which makes searching large directories very fast.

So speed shouldn't be an issue, other than the one you already noted, which is that listings will take longer.

There is a limit to the total number of files in one directory. I seem to remember it definitely working up to 32000 files.

answered Jan 21 '09 at 19:07

Bart Schuller
2,483 16 18

4 Gnome and KDE load large directories at a snails pace, windows will cache the directory so its reasonable. I love Linux, but kde and gnome are poorly written. – rook Apr 20 '10 at 2:26

1 And ext4 seems to have the equivalent of dir_index on by default. – Prof. Falken Feb 22 '12 at 13:22

22 There is a limit of around 32K *subdirectories* in one directory in ext3, but the OP is talking about image files. There is no (practical?) limit on files in an ext3 file system with Dir Index enabled. – Peter N Lewis May 31 '12 at 4:41

1 This answer is outdated, nowadays the default is ext4. – Boris Jan 11 at 9:44

56

Linux server.

Listed files via FTP or a php function is slow yes, but there is also a performance hit on displaying the file. e.g. www.website.com/thumbdir/gh3hg4h2b4h234b3h2.jpg has a wait time of 200-400 ms. As a comparison on another site I have with a around 100 files in a directory the image is displayed after just ~40ms of waiting.

I've given this answer as most people have just written how directory search functions will perform, which you won't be using on a thumb folder - just statically displaying files, but will be interested in performance of how the files can actually be used.

answered Jul 7 '12 at 8:33

S..
4,080 26 32

4 This is the only useful answer. We've made similar experiences. Our limit is 1.000 files to reduce problems with backups (too much directories slow down, too). – mgutt Aug 1 '12 at 18:15

1 It can be useful to mount a drive with noatime as well: howtoforge.com/... and read this, too: serverfault.com/questions/354017/... – mgutt Aug 1 '12 at 18:29

2 What filesystem are you using where it slows down so much? XFS, for example, should be able to easily handle 100,000 files in a directory without any noticeable slowdown. – Ethan Mar 21 '13 at 15:38

1 Contradicting the opinion of most others, I want to confirm this answer. We have hundreds of thousands of images in our social network website. In order to improve the performance we were forced to have 100 (or 1000 for some files) sub directories and distribute the files into them (ext3 on linux+ Apache for us). – wmac Jul 24 '14 at 17:58

47

Keep in mind that on Linux if you have a directory with too many files, the shell may not be able to expand wildcards. I have this issue with a photo album hosted on Linux. It stores all the resized images in a single directory. While the file system can handle many files, the shell can't. Example:

```
-shell-3.00$ ls A*
-shell: /bin/ls: Argument list too long
```

or

```
-shell-3.00$ chmod 644 *jpg
-shell: /bin/chmod: Argument list too long
```

answered Jan 21 '09 at 19:57

Steve Kuo
33k 71 178 239

33 @Steve, use find(1) and/or xargs(1) for these cases. For the same reason it's a good idea to use such tools in scripts instead of command line expansion. – Dave C Jan 21 '09 at 21:25

the shell, but of the system's `exec` implementation. The shell typically can expand the wildcard just fine - it's the call to `exec` with that many arguments that returns the error. – jw013 Nov 30 '12 at 20:34

I had the same error last night (Fedora 15) with "rm" (somefiles*) with about ~400,000 files in a directory. I was able to trim the older files with "find" to the point where I could "rm" with a wildcard. – PJ Brunet Mar 15 '13 at 3:47

10.000.000 files to a directory on etx4 works fine. Not much of a performance hit when accessing. But rather slow with wildcard. Be careful when using shell programs that likes to sort filenames! :) – Simon Rigét Dec 3 '18 at 18:30

22

I'm working on a similar problem right now. We have a hierarchichal directory structure and use image ids as filenames. For example, an image with `id=1234567` is placed in

```
..../45/67/1234567_<...>.jpg
```

using last 4 digits to determine where the file goes.

With a few thousand images, you could use a one-level hierarchy. Our sysadmin suggested no more than couple of thousand files in any given directory (ext3) for efficiency / backup / whatever other reasons he had in mind.

answered Jan 21 '09 at 20:52

armandino
8,857 14 57 70

1 This is a pretty nice solution. Every level of your directory down to the file would have at most 100 entries in it if you stick with the 2 digit breakdown, and the bottom most directory would just have 1 file. – RobKohr May 31 '16 at 12:51

c# implementation github.com/acrobit/AcroFS – Ghominejad Dec 21 '17 at 10:10

16

For what it's worth, I just created a directory on an `ext4` file system with 1,000,000 files in it, then randomly accessed those files through a web server. I didn't notice any premium on accessing those over (say) only having 10 files there.

This is *radically* different from my experience doing this on `ntfs` a few years back.

answered Nov 10 '13 at 18:39

T.J. Crowder
716k 130 1295 1372

what kind of files? text or images?i am on ext4 and have to import 80000 images in a single directory under wordpress and would like to know if it will be okay – Yvon Huynh Aug 19 '17 at 20:29

12

The biggest issue I've run into is on a 32-bit system. Once you pass a certain number, tools like 'ls' stop working.

Trying to do anything with that directory once you pass that barrier becomes a huge problem.

edited Aug 24 '14 at 0:34

Peter Mortensen
14.2k 19 88 114

answered Jan 21 '09 at 19:01

Mike Paterson
391 1 5

6

It absolutely depends on the filesystem. Many modern filesystems use decent data structures to store the contents of directories, but older filesystems often just added the entries to a list, so retrieving a file was an O(n) operation.

Even if the filesystem does it right, it's still absolutely possible for programs that list directory contents to mess up and do an O(n^2) sort, so to be on the safe side, I'd always limit the number of files per directory to no more than 500.

answered Jan 21 '09 at 20:08

Michael Borgwardt
301k 66 434 669

6

It really depends on the filesystem used, and also some flags.

For example, ext3 can have many thousands of files; but after a couple of thousands, it used to be very slow. Mostly when listing a directory, but also when opening a single file. A few years ago, it gained the 'htree' option, that dramatically shortened the time needed to get an inode given a filename.

Personally, I use subdirectories to keep most levels under a thousand or so items. In your case, I'd create 256 directories, with the two last hex digits of the ID. Use the last and not the first digits, so you get the load balanced.

edited Aug 24 '14 at 0:36

Peter Mortensen
14.2k 19 88 114

answered Jan 21 '09 at 19:08

Javier
52.9k 7 72 115

5 If the filenames were completely random, it wouldn't matter which digits were used. – strager Jan 21 '09 at 19:30

Indeed, these filenames are generated randomly. – Kip Jan 21 '09 at 19:58

1 Or use the first N bytes of the SHA-1 digest of the filename. – gawi Mar 4 '15 at 20:49

As an example, F-Spot stores photo files as YYYY\MM\DD\filename.ext, which means the largest directory I have had to deal with while manually manipulating my ~20000-photo collection is about 800 files. This also makes the files more easily browsable from a third party application. Never assume that your software is the only thing that will be accessing your software's files.

answered Jan 21 '09 at 19:55

Sparr
6,874 24 44

4 I advertise against partitioning by date because bulk imports might cluster files at a certain date. – max Jan 27 '09 at 21:31

A good point. You should definitely consider your use cases before picking a partitioning scheme. I happen to import photos over many days in a relatively broad distribution, AND when I want to manipulate the photos outside F-Spot date is the easiest way to find them, so it's a double-win for me. – Sparr Jan 28 '09 at 0:28

4

The question comes down to what you're going to do with the files.

Under Windows, any directory with more than 2k files tends to open slowly for me in Explorer. If they're all image files, more than 1k tend to open very slowly in thumbnail view.

At one time, the system-imposed limit was 32,767. It's higher now, but even that is way too many files to handle at one time under most circumstances.

answered Jan 21 '09 at 19:07

Yes - that Jake.
11.9k 12 62 93

4

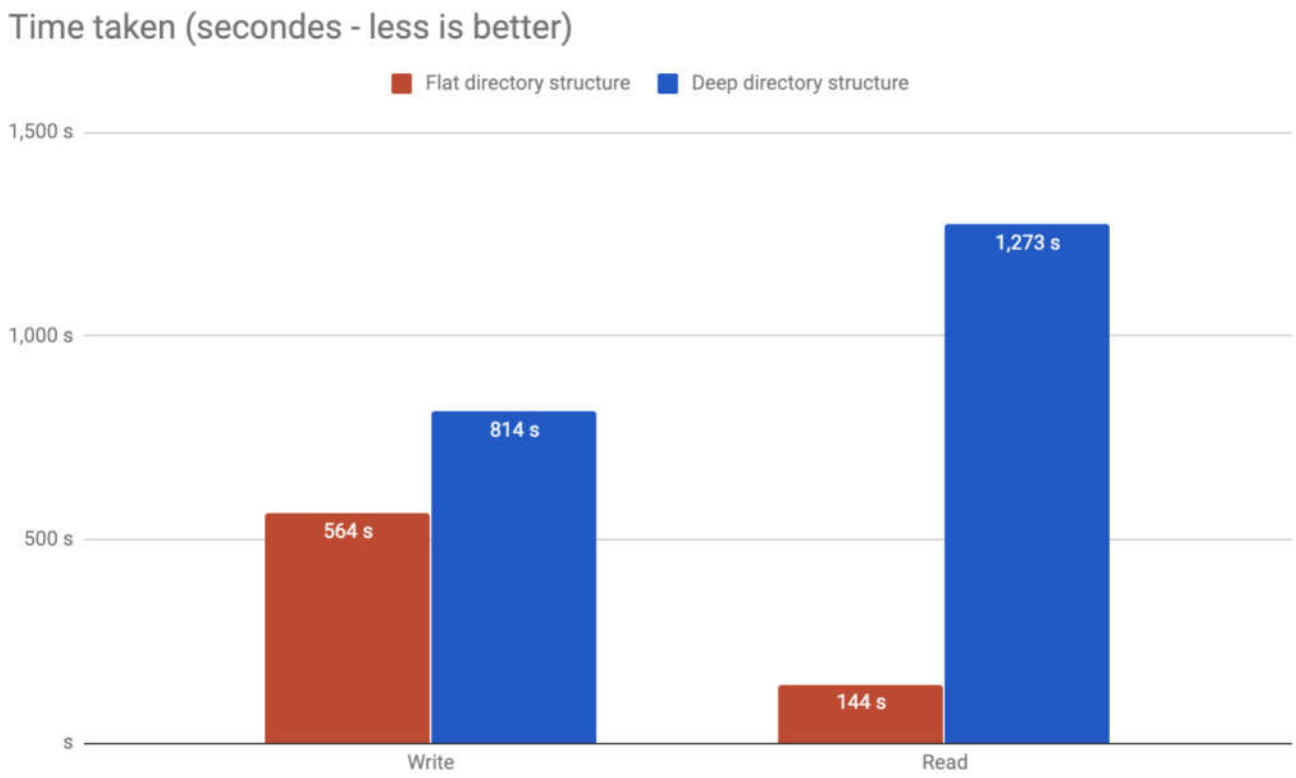
ext3 does in fact have directory size limits, and they depend on the block size of the filesystem. There isn't a per-directory "max number" of files, but a per-directory "max number of blocks used to store file entries". Specifically, the size of the directory itself can't grow beyond a b-tree of height 3, and the fanout of the tree depends on the block size. See this link for some details.

https://www.mail-archive.com/cwelug@googlegroups.com/msg01944.html

I was bitten by this recently on a filesystem formatted with 2K blocks, which was inexplicably getting directory-full kernel messages `warning: ext3_dx_add_entry: Directory index full!` when I was copying from another ext3 filesystem. In my case, a directory with a mere 480,000 files was unable to be copied to the destination.

answered Jan 21 '14 at 22:24

dataless
281 2 9



Wrote an article.

answered Dec 22 '18 at 3:42

Hartator
2,343 2 30 65

A link to a solution is welcome, but please ensure your answer is useful without it: add context around the link so your fellow users will have some idea what it is and why it's there, then quote the most relevant part of the page you're linking to in case the target page is unavailable. Answers that are little more than a link may be deleted. – Samuel Liew ♦ Dec 22 '18 at 4:32

3

I recall running a program that was creating a huge amount of files at the output. The files were sorted at 30000 per directory. I do not recall having any read problems when I had to reuse the produced output. It was on an 32-bit Ubuntu Linux laptop, and even Nautilus displayed the directory contents, albeit after a few seconds.

ext3 filesystem: Similar code on a 64-bit system dealt well with 64000 files per directory.

edited Aug 24 '14 at 0:38

Peter Mortensen
14.2k 19 88 114

answered Jan 21 '09 at 19:13

user54579
2,609 3 19 19

some limit you are comfortable with for performance, aesthetic or whatever reason, you just create a second folder and start dropping files there...

answered Jan 21 '09 at 20:49

Goyuix
18k 14 73 124

2

I ran into a similar issue. I was trying to access a directory with over 10,000 files in it. It was taking too long to build the file list and run any type of commands on any of the files.

I thought up a little php script to do this for myself and tried to figure a way to prevent it from time out in the browser.

The following is the php script I wrote to resolve the issue.

Listing Files in a Directory with too many files for FTP

How it helps someone

answered Nov 26 '10 at 15:37

Swhistlesoft
39 1

2

I prefer the same way as @armandino. For that I use this little function in PHP to convert IDs into a filepath that results 1000 files per directory:

```
function dynamic_path($int) {
    // 1000 = 1000 files per dir
    // 10000 = 10000 files per dir
    // 2 = 100 dirs per dir
    // 3 = 1000 dirs per dir
    return implode('/', str_split(intval($int / 1000), 2)) . '/';
}
```

or you could use the second version if you want to use alpha-numeric:

```
function dynamic_path2($str) {
    // 26 alpha + 10 num + 3 special chars (._-) = 39 combinations
    // -1 = 39^2 = 1521 files per dir
    // -2 = 39^3 = 59319 files per dir (if every combination exists)
    $left = substr($str, 0, -1);
    return implode('/', str_split($left ? $left : $str[0], 2)) . '/';
}
```

results:

```
<?php
```

```
    echo dynamic_path(basename($file, '.jpg')) . $file . PHP_EOL;
}
?>

1/1.jpg
1/12.jpg
1/123.jpg
1/999.jpg
1/1000.jpg
2/1234.jpg
2/1999.jpg
2/2000.jpg
13/12345.jpg
12/4/123456.jpg
12/35/1234567.jpg
12/34/6/12345678.jpg
12/34/57/123456789.jpg

<?php
$files = array_merge($files, explode(',',
'a.jpg,b.jpg,ab.jpg,abc.jpg,ddd.jpg,af_ff.jpg,abcd.jpg,akkk.jpg,bf.ff.jpg,abc-
de.jpg,abcdef.jpg,abcdefg.jpg,abcdefgh.jpg,abcdefghi.jpg'));
foreach ($files as $file) {
    echo dynamic_path2(basename($file, '.jpg')) . $file . PHP_EOL;
}
?>

1/1.jpg
1/12.jpg
12/123.jpg
99/999.jpg
10/0/1000.jpg
12/3/1234.jpg
19/9/1999.jpg
20/0/2000.jpg
12/34/12345.jpg
12/34/5/123456.jpg
12/34/56/1234567.jpg
12/34/56/7/12345678.jpg
12/34/56/78/123456789.jpg
a/a.jpg
b/b.jpg
a/ab.jpg
ab/abc.jpg
dd/ddd.jpg
af/_f/af_ff.jpg
ab/c/abcd.jpg
ak/k/akkk.jpg
bf/.f/bf.ff.jpg
ab/c-/d/abc-de.jpg
ab/cd/e/abcdef.jpg
ab/cd/ef/abcdefg.jpg
ab/cd/ef/g/abcdefgh.jpg
ab/cd/ef/gh/abcdefghi.jpg
```

As you can see for the `$int` -version every folder contains up to 1000 files and up to 99 directories containing 1000 files and 99 directories ...

Finally you should think about how to reduce the amount of files in total. Depending on your target you can use CSS sprites to combine multiple tiny images like avatars, icons, smilies, etc. or if you use many small non-media files consider combining them e.g. in JSON format. In my case I had thousands of mini-caches and finally I decided to combine them in packs of 10.

edited May 23 '17 at 11:47
Community ♦
1 1

answered Apr 17 '15 at 19:32

mgutt
3,224 1 32 50

1

What most of the answers above fail to show is that there is no "One Size Fits All" answer to the original question.

In today's environment we have a large conglomerate of different hardware and software -- some is 32 bit, some is 64 bit, some is cutting edge and some is tried and true - reliable and never changing. Added to that is a variety of older and newer hardware, older and newer OSes, different vendors (Windows, Unixes, Apple, etc.) and a myriad of utilities and servers that go along. As hardware has improved and software is converted to 64 bit compatibility, there has necessarily been considerable delay in getting all the pieces of this very large and complex world to play nicely with the rapid pace of changes.

IMHO there is no one way to fix a problem. The solution is to research the possibilities and then by trial and error find what works best for your particular needs. Each user must determine what works for their system rather than using a cookie cutter approach.

I for example have a media server with a few very large files. The result is only about 400 files filling a 3 TB drive. Only 1% of the inodes are used but 95% of the total space is used. Someone else, with a lot of smaller files may run out of inodes before they come near to filling the space. (On ext4 filesystems as a rule of thumb, 1 inode is used for each file/directory.) While theoretically the total number of files that may be contained within a directory is nearly infinite, practicality determines that the overall usage determine realistic units, not just filesystem capabilities.

I hope that all the different answers above have promoted thought and problem solving rather than presenting an insurmountable barrier to progress.

answered May 23 '16 at 23:30

computersavvy
11 1

0

Not an answer, but just some suggestions.

Select a more suitable FS (file system). Since from a historic point of view, all your issues were wise enough, to be once central to FSs evolving over decades. I mean more modern FS better support your issues. First make a comparison decision table based on your ultimate purpose from FS list.

I think its time to shift your paradigms. So I personally suggest using a distributed system aware FS, which means no limits at all regarding size, number of files and etc. Otherwise you will sooner or

I'm not sure to work, but if you don't mention some experimentation, give AUFS over your current file system a try. I guess it has facilities to mimic multiple folders as a single virtual folder.

To overcome hardware limits you can use RAID-0.

answered Dec 17 '13 at 5:37

shvahabi
838 6 6

0

There is no single figure that is "too many", as long as it doesn't exceed the limits of the OS. However, the more files in a directory, regardless of the OS, the longer it takes to access any individual file, and on most OS's, the performance is non-linear, so to find one file out of 10,000 takes more then 10 times longer then to find a file in 1,000.

Secondary problems associated with having a lot of files in a directory include wild card expansion failures. To reduce the risks, you might consider ordering your directories by date of upload, or some other useful piece of metadata.

answered Feb 16 '14 at 0:18

Paul Smith
384 6 11
