NIM'S NONSENSE

Whispers from the geeky side

2013-01-13 BY NIM

Repeat after me: MySQL is not a filesystem

I came across this gem on DZone this morning. It's a tutorial on storing images in a MySQL database (using PHP). There are several things in the tutorial that I don't agree with, but I'll let those slide. What really bugs me, is how it fails to mention that **this is a *very* bad idea**.

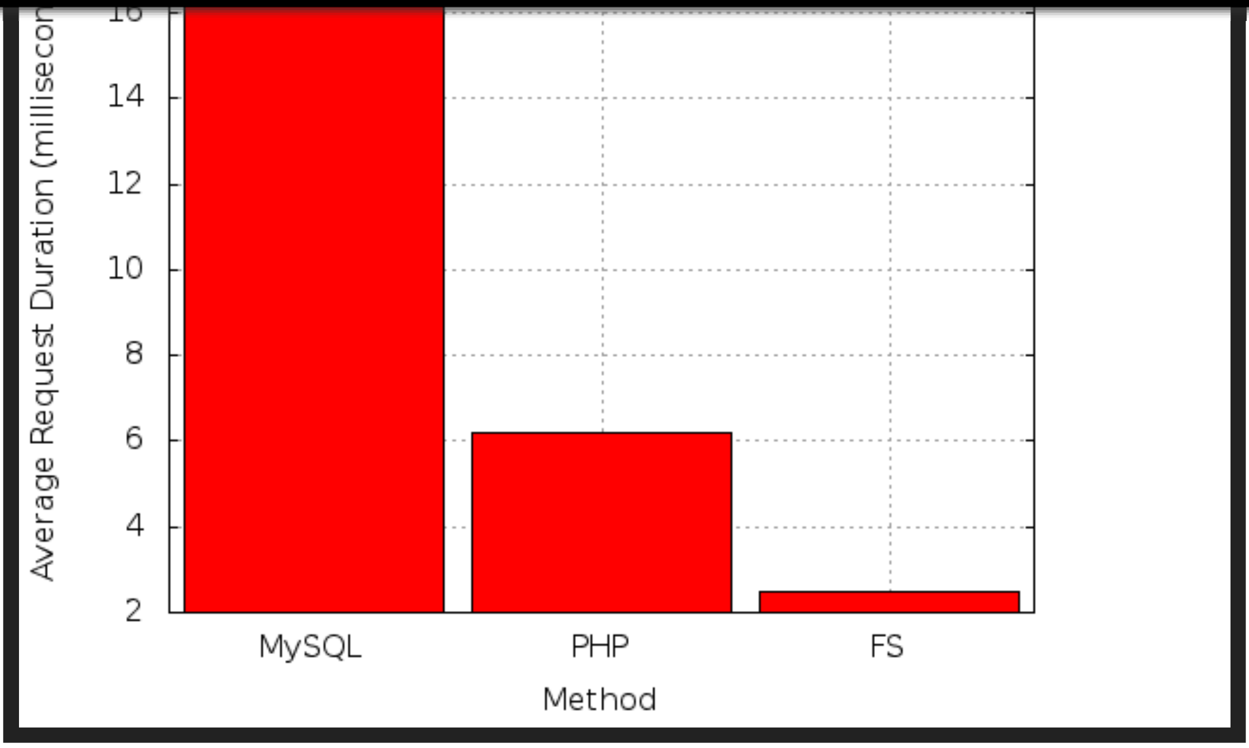
A relational database is **not** a filesystem. Files go on a filesystem. Relational data goes in an RDBMS. Repeat that a couple of times.

The most compelling argument for this, is performance. I did a quick test. I did a google image search on stupidity and downloaded the first 10 images. I then wrote PHP scripts to serve them up in two ways:

1. From a MySQL (MyISAM) table with 2 columns: ID (int, auto_increment) and DATA (mediumblob)
2. Using readfile.

The third test method, "FS", simply loads the image over HTTP directly, without any intermediary scripts.

The results are the average of running Apache Benchmark 10 times: 10 concurrent requests, 1000 requests per run.



As you can see, the MySQL approach is a hell of a lot slower than the more sensible FS approach.

The best way to store your images (or other binary files) is on the filesystem. Every modern web server does a good (or excellent) job of serving up static content. Storing them in a database is by far the worst possible solution. Not only because it's slow, but also because it complicates database backups: MySQL dumps with binary data don't compress very well, causing the whole database backup to be slower and larger than needs be.

So please, be sensible. Store your files on a filesystem.

ENGLISH

# **MYSQL**, **PERFORMANCE**, **PHP**

9 Replies to "Repeat after me: MySQL is not a filesystem"

It's not about where you store images, it's about your strategy to serve them. You could have images in the database and use output cache on the web server.

Of course it slows backups, because your are backing up everything, which also simplifies backups, only one backup to do, you don't have to worry about files.

Justin Dalton

2013-01-14 AT 15:40 (UTC)

I agree, having tried this a long time ago. Any time you have a situation where you would need to "index" an image in a database you are better off to store the image in the file system and use the database to store its location (file name) in a database table and using the location to perform whatever operations you need to perform on the image.

Martyr2

2013-01-14 AT 18:13 (UTC)

Good article. I wrote about this topic back in 2011 in response to an increased "chatter" of storing images into MySQL.

http://www.coderslexicon.com/inserting-images-into-mysql-and-retrieving-them-using-php/

My research came to the same conclusions that large images should be left out of BLOB fields. I did notice that if the imagery was small, say around an icon size that most databases handle this amount fairly well.

However, my stance parallels yours. Leave image data out of it. You gain advantages to having path data to images as well like being able to search the file name etc.

Thanks! 🙂

Nim

2013-01-14 AT 22:13 (UTC)

True. Still doesn't make MySQL any more of a filesystem though. When I have the time I might have a look at the effect of large blobs on overal database performance under load. Might make for interesting results.

Maybe you want to link data from an existing database with some files that you can access and update through a filesystem interface. For example, storing digital pictures or other content that like to be seen as a filesystem and being able to select * from mysqlfs, existingtable... where JOIN-Condition-Here. IIRC there was a large company playing up how wonderful life could be when filesystem queries could be performed using SQL. Can't recall the name of the company off the top... microsomething?

Arthur

2014-08-11 AT 17:22 (UTC)

Do I understand the results correctly? According to your benchmark on your server, you can only serve 50 images per second from a db? That is very slow indeed. What sizes were the images? Did the difference increase with the image size? i.e if you stored tiny images on the db, did it perform on par with the fs?

spongebob

2015-08-25 AT 05:06 (UTC)

hi just so you know mysql CAN be a filesystem only problem i see is there was never a kernel level module developed and no one ever bothered to optimize the concept. mostly because it was slow going tru fuse , userspace , kernel calls and several other level of abstraction ... apparently ... (i am being a bit sarcastic here)

having a database as a filesystem has numerous advantages but people fail to understand that. it does have big disadvantages if like was said it has to go thru loops just to work the basics for sure its going to be slow just like anything that has to go thru those same loops. but in fact with massive amounts of data people acquire these days its actually a mystery why people are still on flat and inflexible file systems.

Xenofon

2016-06-08 AT 15:30 (UTC)

This is an ancient post but I found this searching for “Is storing blobs in rdbms still considered bad practice”, so my 2 cents:

– The most common use case people consider using a database for storing images or other blobs is updating dynamic content. The three main problems with using a filesystem are:

1) If you have many (more than a few thousands of) binary files, using a filesystem would

3) If you have a cluster of web server (the most typical scenario), you either need a clustered filesystem (much harder to setup compared to an RDBMS) or you need to synchronize your updates on your own (very hard to do and has room for inconsistencies).

– The benefits of an RDBMS in these areas outweigh the performance issues. Any web app concerned with performance will have a cache of some sort. It makes much better sense to use an RDBMS to persist the BLOBs, then the cache to serve them after the 1st time you load them.

– You could ofc use some highly available file/document server such as S3, but most people don't need this complexity and RDBMS should be just fine. In any case, if RDBMS is not good enough for you, then a simple filesystem definitely isn't.

Nim

2016-06-08 AT 16:32 (UTC)

1. It's true that some filesystems are slow when directories contain many files. This is easily solved, and isn't any harder to implement than throwing BLOBs at your database.
2. If you're not backing up your files, you're doing something wrong.
3. Again, this isn't any harder to configure than a database cluster. It's a matter of using the right tool for the job

The benefits of the RDBMS you speak of aren't benefits of an RDBMS at all. And if you're going to be introducing a cache layer, then you might as well go the whole distance and serve your static files from a separate webserver (such as thttpd). It'll save you boatloads on memory and disk space.