File / Image Replication

Asked 16 years ago Modified 16 years ago Viewed 1k times

Part of PHP Collective

5

I have a simple question and wish to hear others' experiences regarding which is the best way to replicate images across multiple hosts.

I have determined that storing images in the database and then using database replication over multiple hosts would result in maximum availability.

The worry I have with the filesystem is the difficulty synchronising the images (e.g I don't want 5 servers all hitting the same server for images!).

Now, the only concerns I have with storing images in the database is the extra queries hitting the database and the extra handling i'd have to put in place in apache if I wanted 'virtual' image links to point to database entries. (e.g AddHandler)

As far as my understanding goes:

- If you have a script serving up the images: Each image would require a database call.
- If you display the images inline as binary data: Which could be done in a single database call.

- To provide external / linkable images you would have to add a addHandler for the extension you wish to 'fake' and point it to your scripting language (e.g php, asp).

I might have missed something, but I'm curious if anyone has any better ideas?

Edit: Tom has suggested using mod_rewrite to save using an AddHandler, I have accepted as a proposed solution to the AddHandler issue; however I don't yet feel like I have a complete solution yet so please, please, keep answering ;)

A few have suggested using lighttpd over Apache. How different are the ISAPI modules for lighttpd?

PHP php mysql sql-server apache image

Share

Improve this question

Follow

edited Dec 16, 2008 at 17:12

asked Dec 16, 2008 at 15:55

Jay

833 ● 7 ● 15

4 Answers

Sorted by: Highest score (default)

3

✓

If you store images in the database, you take an extra database hit *plus* you lose the innate caching/file serving optimizations in your web server. Apache will serve a static image much faster than PHP can manage it.

In our large app environments, we use up to 4 clusters:

- App server cluster
- Web service/data service cluster
- Static resource (image, documents, multi-media) cluster
- Database cluster

You'd be surprised how much traffic a static resource server can handle. Since it's not really computing (no app logic), a response can be optimized like crazy. If you go with a separate static resource cluster, you also leave yourself open to change just that portion of your architecture. For instance, in some benchmarks lighttpd is even faster at serving static resources than apache. If you have a separate cluster, you can change your http server there without changing anything else in your app environment.

I'd start with a 2-machine static resource cluster and see how that performs. That's another benefit of separating functions - you can scale out only where you need it. As far as synchronizing files, take a look at existing file synchronization tools versus rolling your own. You may find something that does what you need without having to write a line of code.

Share Improve this answer Follow

answered Dec 16, 2008 at 16:23

Corbin March
25.7k ● 6 ● 76 ● 100

Hi Corbin, Thanks for your reply. To be honest I hadn't even thought of the webserver cache, you are right that the webserver may respond with a HTTP 200 rather than a HTTP 304 for the images. Are you 100% sure that the webserver wont see the the URI and the filesize and respond with a 304. – Jay Dec 16, 2008 at 16:29

I wasn't thinking so much of 200 vs 304. Even if the server doesn't catch it, you can code your handler to return a 304. When I mention file serving optimization, I'm thinking more that your server is optimized to handle static files - caching commonly requested files and avoiding script exec. – Corbin March Dec 16, 2008 at 16:42

Hi Corbin, thanks and I agree. However, it may hit a pitfall with 'faking' a HTTP 304 if the record has actually changed. I would definately prefer that the image request hit a cache before it hit the database, however there would have to be some sort of cache invalidation in place. – Jay Dec 16, 2008 at 17:05

2

Serving the images from wherever you decide to store them is a trivial problem; I won't discuss how to solve it.

Deciding where to store them is the real decision you need to make. You need to think about what your goals are:

- Redundancy of hardware
- Lots of cheap storage
- Read-scaling
- Write-scaling

The last two are not the same and will definitely cause problems.

If you are confident that the size of this image library will not exceed the disc you're happy to put on your web servers (say, 200G at the time of writing, as being the largest high speed server-grade discs that can be obtained; I assume you want to use 1U web servers so you won't be able to store more than that in raid1, depending on your vendor), then you can get very good read-scaling by placing a copy of all the images on every web server.

Of course you might want to keep a master copy somewhere too, and have a daemon or process which syncs them from time to time, and have monitoring to check that they remain in sync and this daemon works, but these are details. Keeping a copy on every web server will make read-scaling pretty much perfect.

But keeping a copy everywhere will ruin write-scalability, as every single web server will have to write every changed / new file. Therefore your total write throughput will be limited to the slowest single web server in the cluster.

"Sharding" your image data between many servers will give good read/write scalability, but is a nontrivial exercise. It may also allow you to use cheap(ish) storage.

Having a single central server (or active/passive pair or something) with expensive IO hardware will give better write-throughput than using "cheap" IO hardware everywhere, but you'll then be limited by read-scalability.

Share Improve this answer Follow

answered Dec 17, 2008 at 6:29

MarkR
63.5k ● 15 ● 119 ● 154

Thanks for the response MarkR, I am currently mulling over the responses here to see which solution fits me best. Last thing I want is 4 different servers with out of sync data!! I've heard some good things about Stackless IO, what is your opinion? – Jay Dec 19, 2008 at 10:26

1

Having your images in a database doesn't necessarily mean a database call for each one; you could cache these separately on each host (e.g. in temporary files) when they are retrieved. The source images would still be in the database and easy to synchronise across servers.

You also don't really need to add Apache handlers to serve an image through a PHP script whilst maintaining nice urls- you can make urls like http://server/image.php/param1/param2/param3.JPG and read the parameters through `$_SERVER['PATH_INFO']` . You could also remove the 'image.php' portion of the URL (if you needed to) using mod_rewrite.

Share Improve this answer Follow

edited Dec 16, 2008 at 16:28

answered Dec 16, 2008 at 16:20

Tom Haigh
57.8k ● 21 ● 117 ● 143

Hi Tom, Thanks for your reply. Using mod_rewrite would certainly avoid the use of an apache AddHandler. I am already using mod rewrite so it would not be too much effort to implement this. – Jay Dec 16, 2008 at 16:23

1

What you are looking for already exists and is called MogileFS Target setup involves mogilefsd, replicated mysql databases and lighttd/perlbal for serving files; It will bring you failover, fine grained file replication (for exemple, you can decide to duplicate end-user images on several physical devices, and to keep only one physical instance of thumbnails). Load balancing can also be achieved quite easily.

Share Improve this answer Follow

answered Dec 16, 2008 at 17:01

MatthieuP
1,126 ● 5 ● 12

Hi MatthieuP, Thanks for a very interesting reply! I will have to look into this, but it does sound ideal! – Jay Dec 16, 2008 at 17:11

I have updated my question, I am curious: How different are the ISAPI modules for lighttpd? – Jay Dec 16, 2008 at 17:13