Making Simple Things Really Complicated

High Availability for WordPress

(or just about anything else)

David Smith @dsXLII https://xlii.io/wcstl16

Level-Setting

- This is NOT a deep comprehensive study of HA
- Most of this presentation is NOT specific to WordPress

There’s seriously a whole branch of computer science devoted to high availability and scalability, and related traits. So no, you’re not gonna be an expert after one quick WordCamp. But we’ll try to at least give you some basic vocabulary. This way, you’ll have an idea of what’s going on, and what questions to ask, for when your site becomes REALLY popular.

High Availability

- “likely to operate continuously without failure for a long time”
- “a characteristic of a system, which aims to ensure an agreed level of operational performance for a higher than normal period”
- “the ability to avoid unplanned outages”

Source: technopedia.com, Wikipedia, a different page on Wikipedia

These definitions are all a bit fiddly, the key point (IMO) is that you’re taking extraordinary measures to ensure your (whatever) is available in the wake of system failures.

Key aspects of HA:

* Avoid single points of failure
* Reliable crossover (between replicated components)
* Detection of failure

Do you *need* HA?

- What is the impact of your site being unavailable?
- Business-critical (online shopping, hosting company)
- Life or health at risk (emergency communications)

https://docs.oracle.com/cd/B28359_01/server.111/b28281/hadesign.htm

There's a LOT of added cost and complexity to HA — more servers, ideally geographically distributed; more personnel (because it's very rare to find one person who knows enough about every component to tune them properly). These are a few reasons where you might really need HA.

Your mom's blog probably doesn't need HA.

Determining this likely require some serious business impact analysis, cost/benefit, etc. Again, a really broad subject.

Reputation is sorta subjective.
Business-critical: think "Amazon" or any major online shopping venue. Or a hosting company, though that could be more embarrassing than anything.
Life/health: emergency.wustl.edu, used heavily for incident mgmt, as a local example.

Measuring Availability

- 99.9%: 8.76 hours/year (43 minutes/month)
- 99.99%: 52.5 minutes/year (4.3 minutes/month)
- 99.999%: 5.26 minutes/year (26 seconds/month)

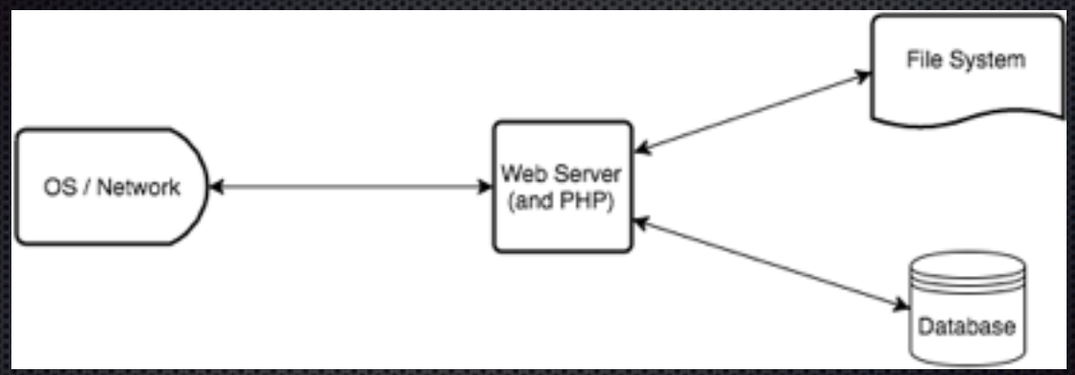
NINES

Seriously, most modern hosting, well-managed, can limit downtime to five or ten minutes, a couple times a month, for bouncing services, reboots, etc. (Probably around “three nines,” 99.9% uptime, ~43 minutes/month, or about 8 hours total downtime per year.)

Basic Web Environment

Note that for really small basic stuff (personal sites, dev/test) having your whole environment in one place is perfectly alright! See also $5 droplets from Digital Ocean, most super-budget hosts. My home dev environment is a Raspberry Pi. Just make sure you back up anything important. Which you should be doing anyway…

Zooming In: The Web Server



Operating System: Presumably Linux or Windows, but could be so many other things…
Web server process: Apache httpd, nginx, maybe IIS
PHP: Might technically be part of the Web server (mod_php) or not (fastcgi, etc)
File System: (where the files are, silly) - Assuming it's "local" for a really basic setup
DB: Separate process, maybe (or not) running on the same server

And if ANY ONE of those pieces fails, your WordPress site goes splat.

Scaling Out: Database

- MySQL Replication (master/slave)
- Amazon RDS
- MariaDB/Galera or Percona XtraDB
- Oracle MySQL Enterprise HA
- *WordPress HyperDB plugin*

Standard DB setup: Single node

MySQL standard master/slave: Works pretty well, but requires manual failover and slave promotion (or some sort of fancy wrapper script to do that for you)

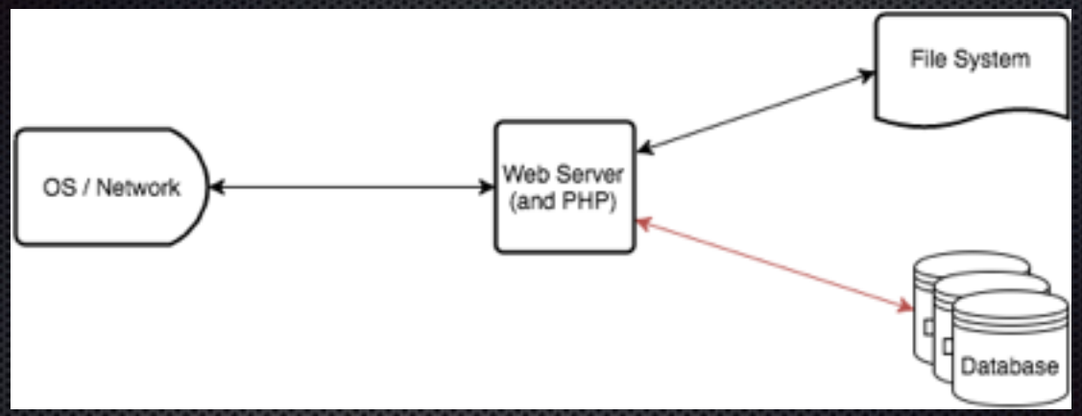
Oracle Enterprise HA: multi-master, probably really expensive (Oracle)

MariaDB and Percona are MySQL-alikes, and Galera and Xtra are those vendors' respective multi-master DB products. MariaDB has been standard since RHEL 7, so you may well already be using it.

Amazon RDS (Relational Database Service): MySQL, with the failover parts automated — redundancy is literally a single checkbox. But since you're maintaining two DB instances, they charge you twice as much.

HyperDB: If you use a multi-master solution, you'll probably want this (maybe also touch on sharding) unless you have a load balancer (q.v. later stuff)

Scaling Out: Database



We’ve scaled out the database, but there’s still only one copy of our files (WordPress itself, the /wp-content/ dir with its themes and uploads, etc…)

Sadly, scaling this out likely means separating the content from the Web server entirely.

Scaling Out: File System

- Dependent on where/how you're hosting
- Amazon? Elastic File System (where available)
- Windows shop? DFS
- Linux shop? GlusterFS or DRBD
- Enterprise? SAN filers (hardware or software)

Amazon EFS - not bad but presently only available in a couple of AWS regions

Windows DFS (Distributed File System) - requires a pretty hefty investment in Active Directory

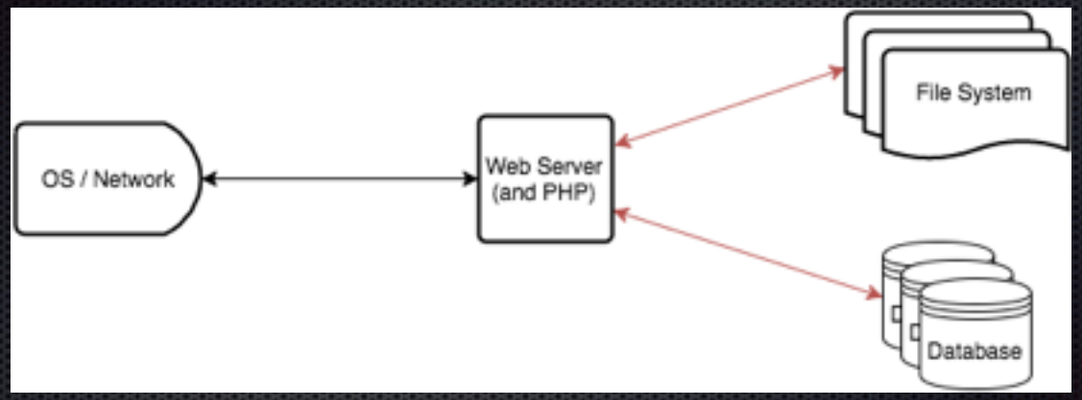
GlusterFS or DRBD - Both pretty reliable, but can be hard to set up. GlusterFS is a FUSE file system, DRBD is a block-based kernel module

NFS SAN filers are great, but ridiculously expensive (like, tens of thousands of dollars). OpenFiler may be a viable solution too.

Basically, there are no good answers here. :(

Sadly, there's not much WordPress can do here.

Scaling Out: File System



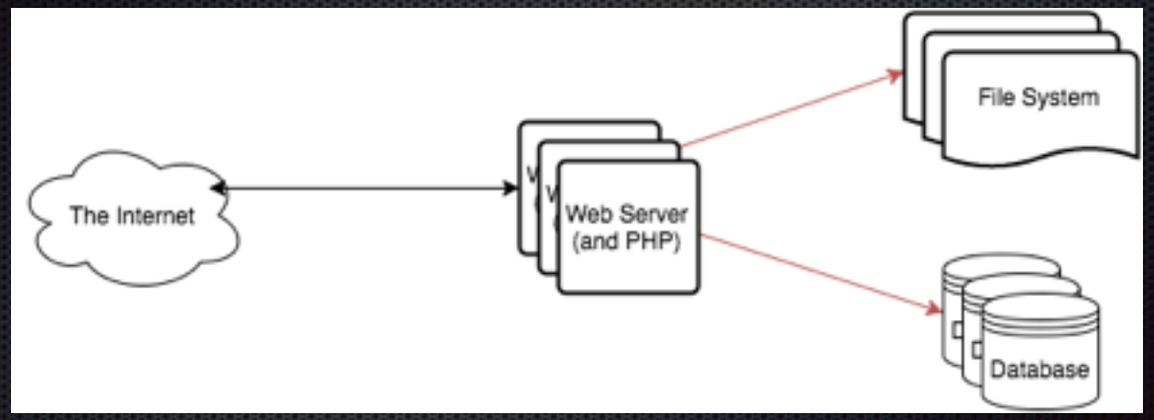
Scaling Out: Web Server

- You can't just copy "the Web server" — you really have to copy the whole machine, configs and all
- VM clones: Amazon AMI, VMware templates …
- Config management: Salt, Puppet, Ansible, Chef …

TO DO: Options for scaling out your Web server. Needs to be consistent and reproducible.

This could be a whole presentation unto itself…

Scaling Out: Web Server



Scaling Out: Load Balancing

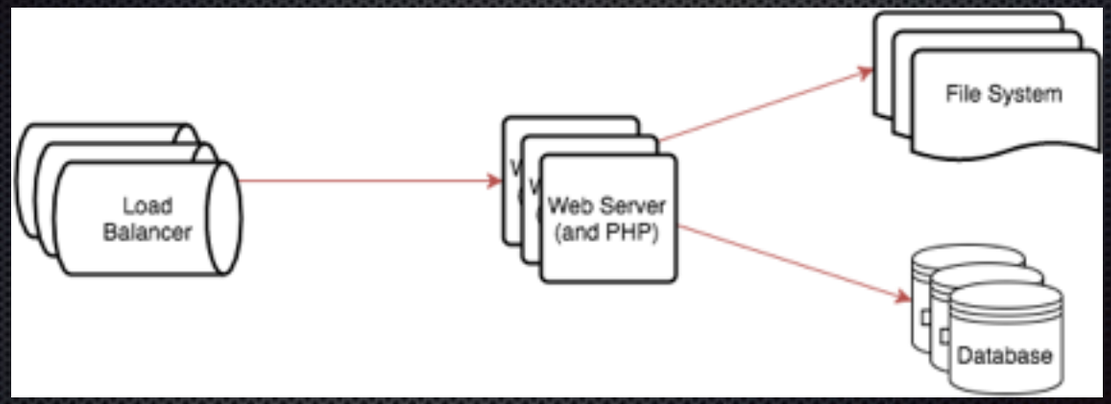
- Simple: DNS “round-robin”
- Software: Red Hat Load Balancer, Microsoft TMG
- Dedicated: F5, Netscaler, Barracuda, etc.

DNS: Hey, you have to have it anyway… but it’s not the most reliable approach — TTL/caching, no easy way to remove dead nodes, etc.

TMG — Threat Management Gateway. RH Load Balancer is their supported version of HAProxy, heartbeat, etc.

F5 LTM, Citrix/Netscaler, Barracuda (probably just another rebadged HAProxy)

Scaling Out: Load Balancing



Further Topics

- Geography: Are all of your data in the same building/ city/time zone?
- CDNs and content caching (where possible)
- Performance Tuning (PHP opcode caching, DB object caching…)

Geography: Are you trying to avoid regional disasters?

CDNs: One way to distribute your work geographically, assuming your content is static enough for this to work

Tuning: PHP opcode caching with APC or Zend; Object caching with memcached or Redis; maybe “local” content caching with Apache httpd mod_cache, Varnish or a WP plugin like WP Super Cache/W3 Total Cache

Conclusion/Questions

https://xlii.io/wcstl16 (in a day or two)