Running a High Performance LAMP stack on a $20 Virtual Server

Jay Janssen
Percona, Inc.

Simplified Uptime

- Started a side-business selling customized hosting to small e-commerce and other web sites
- Spent a lot of time optimizing for RAM utilization
- Had several sites appear in places like:
  - Digg -- Patterntap.com
  - Makezine -- Simplifiedbuilding.com

The framework

- Debian stable + backports.org + dotdeb
- Puppetized configuration
- Apache 2.2, PHP 4.x, MySQL 5.1.xx, Innodb plugin
- Redundant VPS providers/regions for Business continuity goals

Tuning Goals

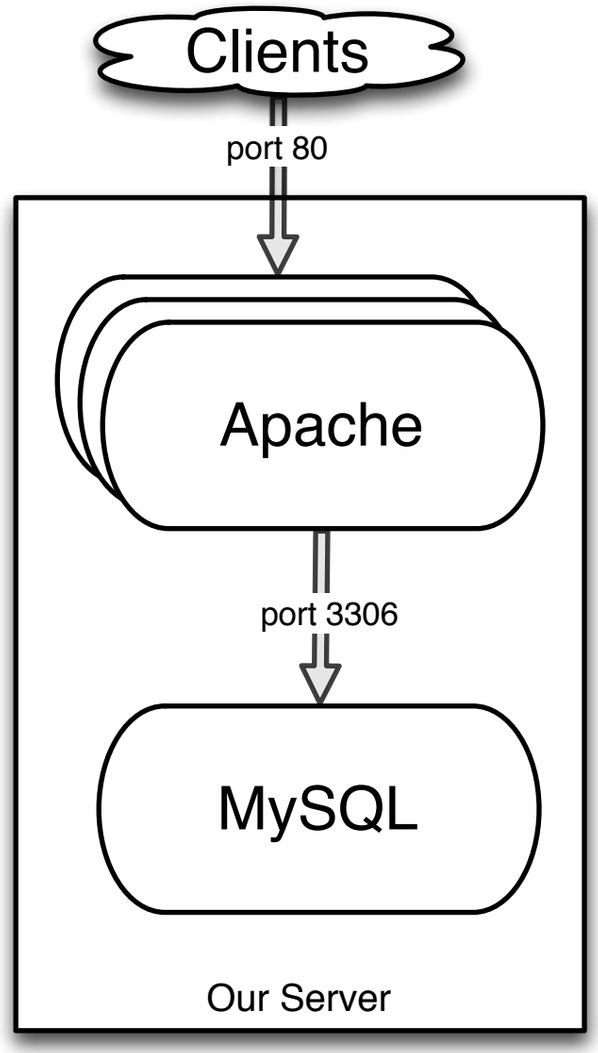
- Allow users to connect with Keep-alive
- Serve static files efficiently
- Prevent swapping on the server
- Tune the server to prevent self-DOS
- Work with most* open source PHP apps

PERCONA LIVE

Serving Static Files

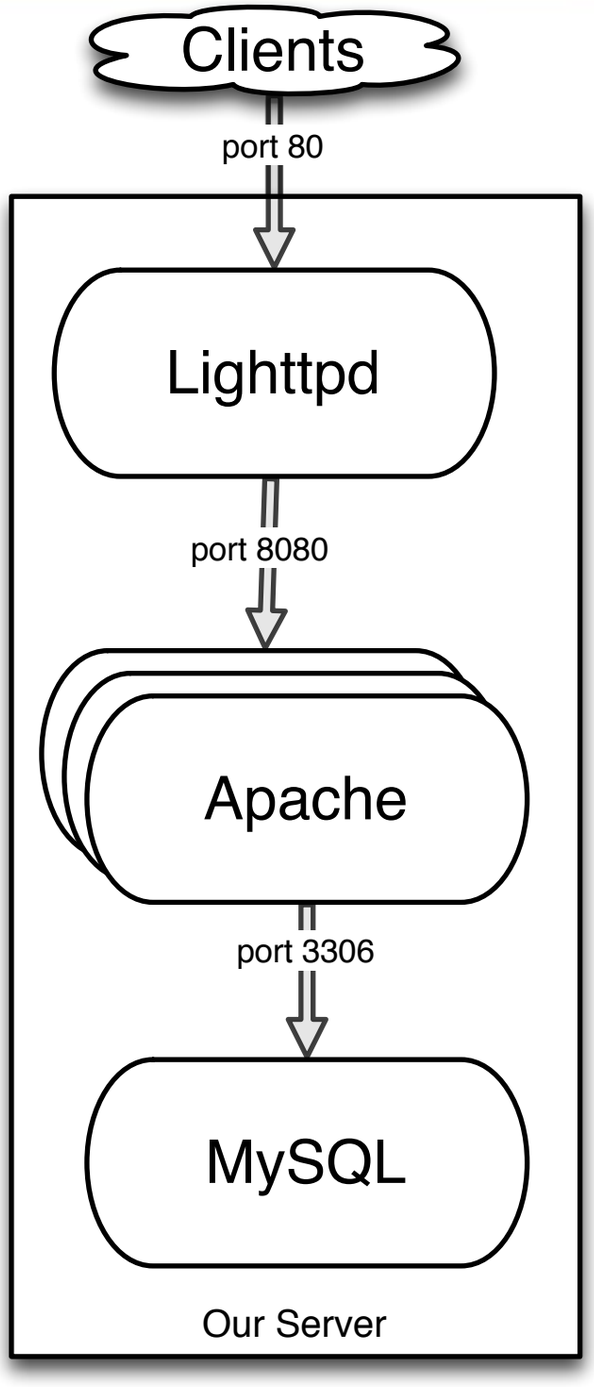
Standard forked Apache + mod_php

- Single process per connection served
- Expensive for memory utilization
- Pros:
  - "Just works" with most OSS
  - .htaccess and php ini_set
- Cons
  - Can't easily use keep-alive
  - Not going to scale



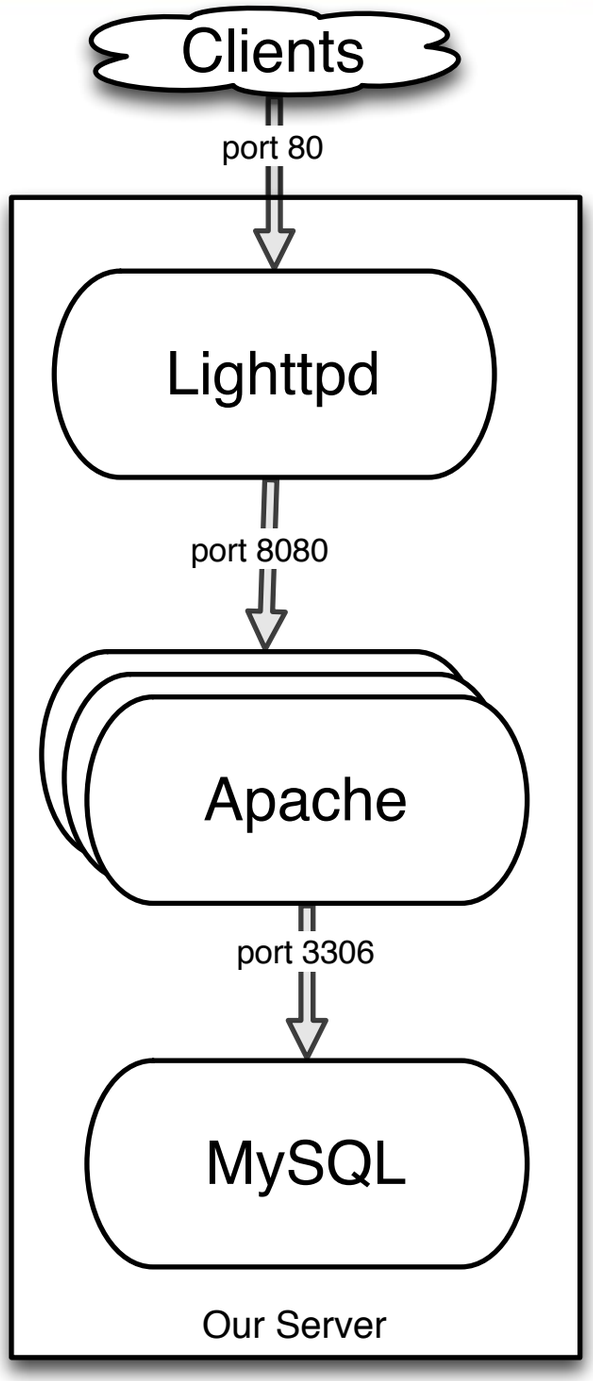
Static Files in Front

- End users connect to lighttpd or nginx
- Match static files in config by filename (*.jpg|*.gif|*.png|*.cs|*.js) and serve those directly
- Everything else is proxied to apache running on another port



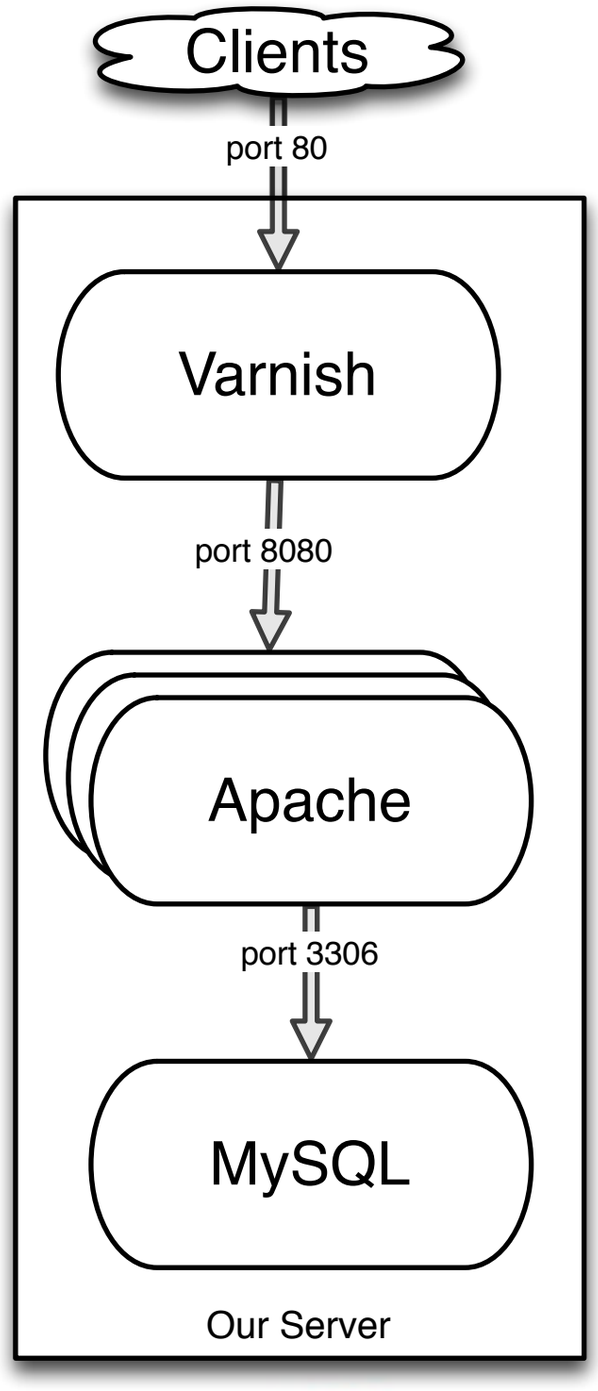
Static Files in Front - Pros / Cons

- Pros:
  - keep-alive
  - Static files served uber-fast
- Cons:
  - Stupid regex -- inefficient
    - Files that appear static, but really are php
    - Files that are static, but are served by a *.php
  - Multiple webserver configs to maintain
  - Can confuse some apps, apps won't see client ips



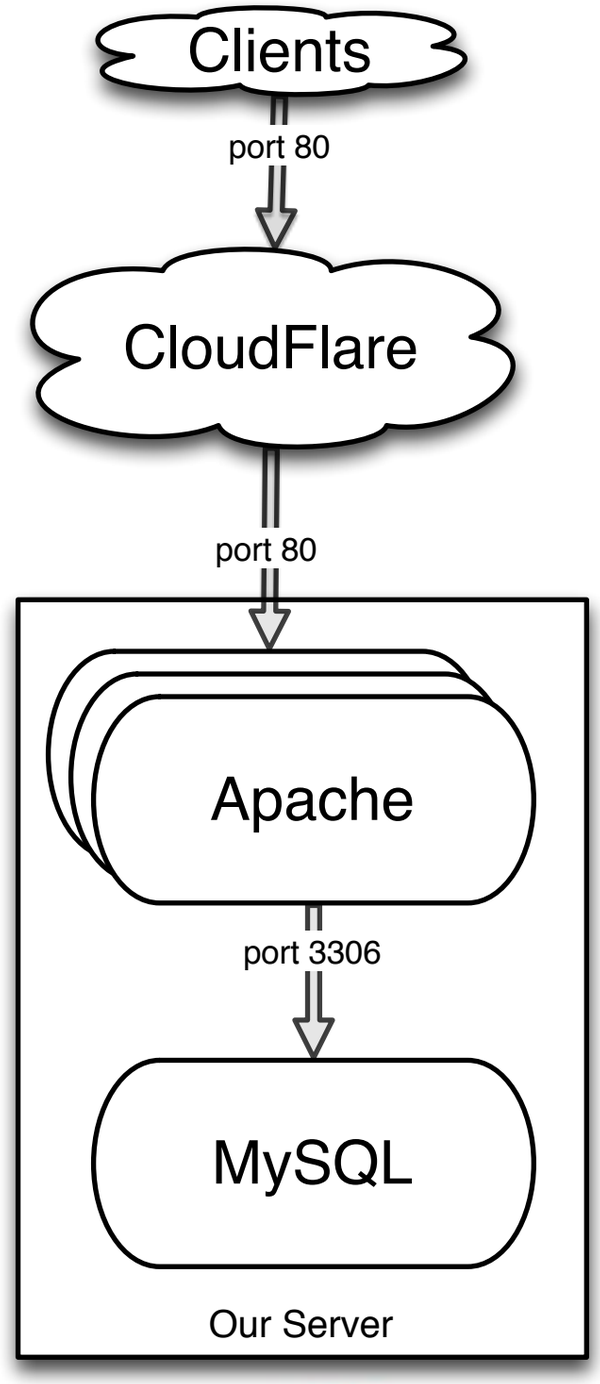
Caching Server in front

- Configured varnish to cache by Content-type
- Pros:
  - More accurate by-type caching
  - Application can override cache with headers
- Cons:
  - Larger memory footprint
  - Varnish config can be tricky
  - App still can't see client ip and may get confused about its hostname



Caching service in front

- Run Apache on port 80
- Use an external caching service (e.g., CloudFlare) and setup your domain name to point at them
- Cloudflare caches static content for you, obeys cache headers, etc.



Caching service in front -- Pros / Cons

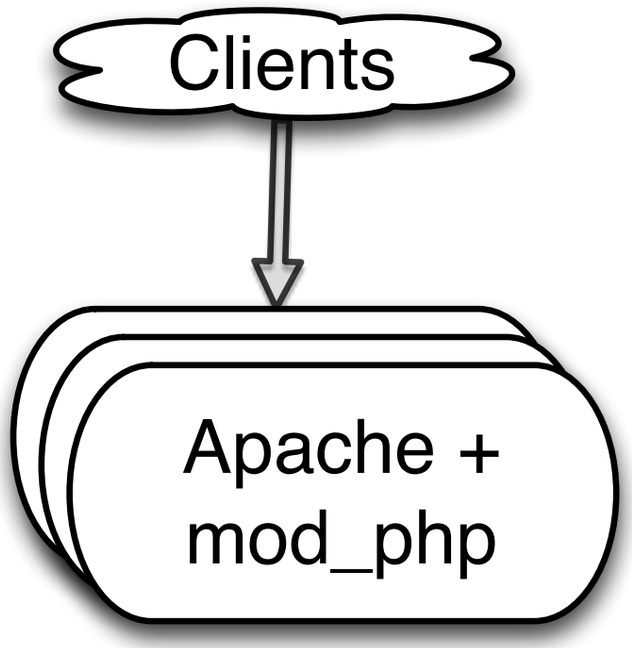
- Pros:
  - End-users use keep-alive, have geo-affinity with Cloudflare's proxies
  - No extra daemon on your server
  - HTTP and HTTPS can be served by Cloudflare
  - App server thinks it's serving directly (and actually can)
  - DOS attacks
- Disadvantages:
  - Dependency on external service
  - No transparency into caching methodology
  - Still can't see client IP w/o extra config

PERCONA LIVE

Serving Dynamic files

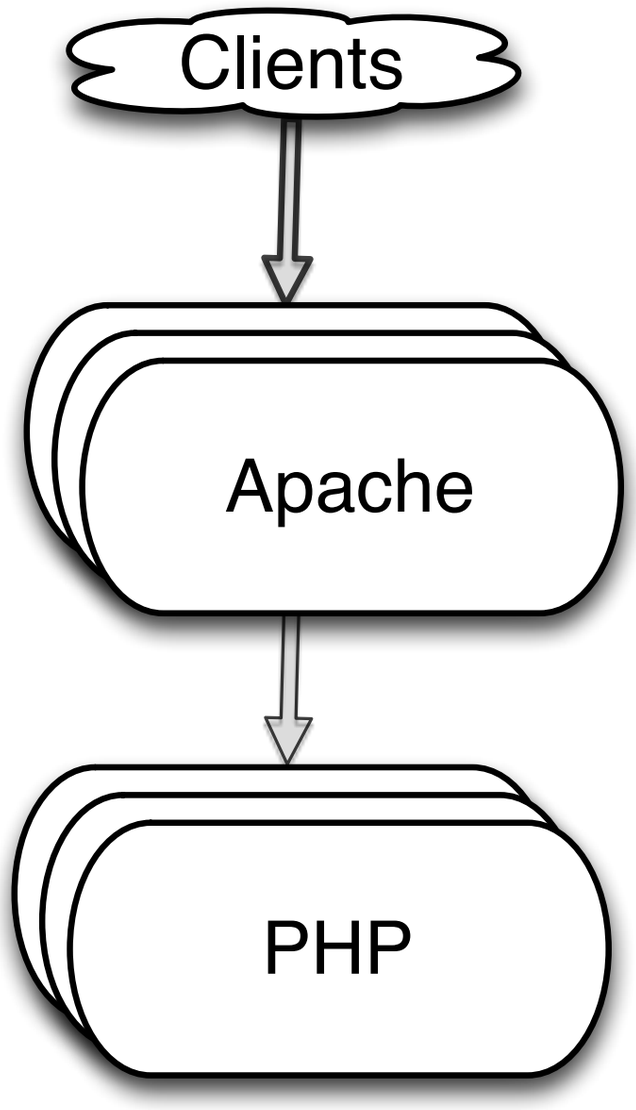
mod_php

- Pros:
  - Allows standard .htaccess files to be used, including PHP ini_set
  - Default PHP, most things “just work”
- Cons:
  - ties PHP with preforked Apache, problematic for serving a lot of connections with Apache



fastCGI + php-fpm (or similar)

- Pros:
  - Allows PHP to be scaled independently of concurrent connections on the webserver
  - Webserver automatically serves non-php content directly -- one config
- Cons:
  - One more knob to turn
  - Breaks sites that rely on .htaccess for php config
  - Sometimes breaks php in weird ways

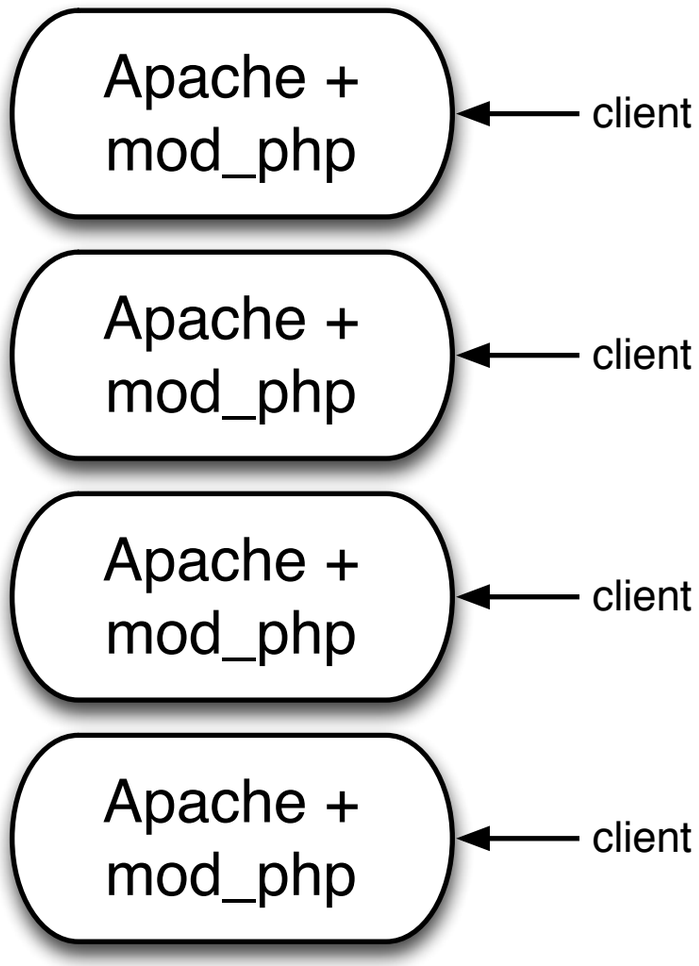


PERCONA LIVE

Tuning Apache

With mod_php

- Must use prefork Apache because mod_php is not threadsafe
- Keep MaxClients LOW -- max 5-10
- StartServers/MinSpareServers -- keep large enough to serve normal traffic
- MaxSpareServers -- small enough to keep extra procs cleaned up
- KeepAlive off!!



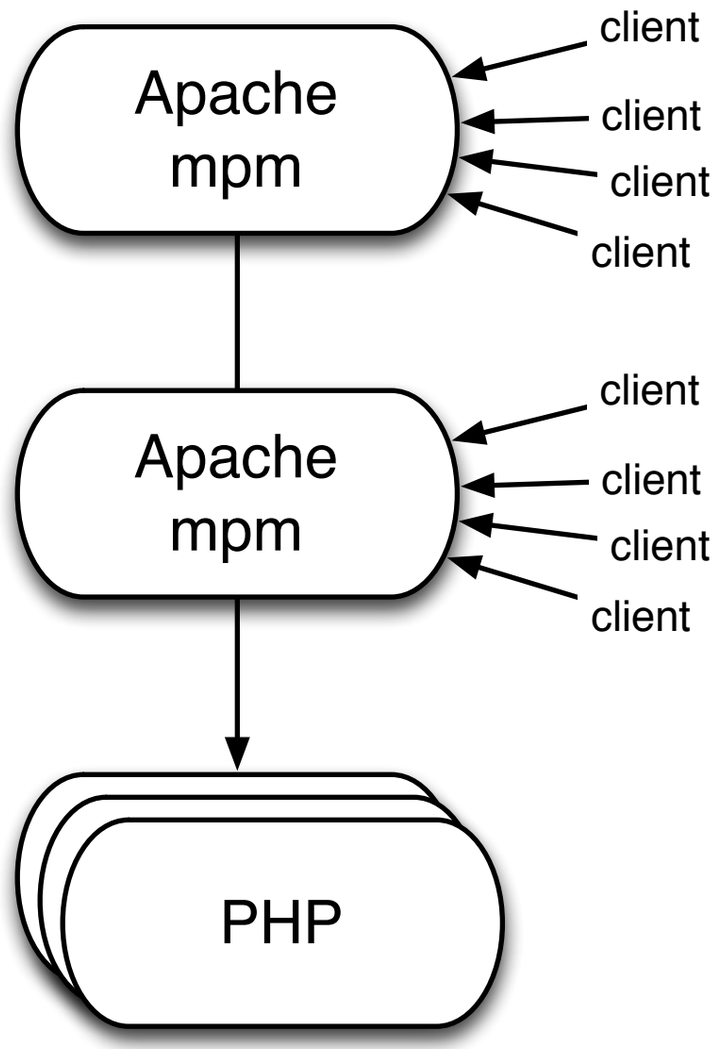
Apache mod_php configuration

```
1.<IfModule mpm_prefork_module>
2.     StartServers              3
3.     MinSpareServers           3
4.     MaxSpareServers           4
5.     MaxClients               10
6.     MaxRequestsPerChild    1000
7.</IfModule>

9.KeepAlive Off
```

Without PHP

- Use mpm_worker in Apache
- Relationship between “Servers” and “Threads”
- MaxClients - large, but not too big (50-100?)
- ThreadsPerChild - not too small to avoid stopping/starting Servers frequently
- KeepAlive=ON, KeepAliveTimeout not too big



Apache config w/o PHP

```
1. <IfModule mpm_worker_module>
2.     StartServers          2
3.     MaxClients           75
4.     ThreadsPerChild      25
5.     ServerLimit           3
6.     MinSpareThreads      25
7.     MaxSpareThreads      50
8.     MaxRequestsPerChild   1000
9. </IfModule>

11.KeepAlive On
12.MaxKeepAliveRequests 100
13.KeepAliveTimeout 10
```

Tuning PHP

- Keep your memory limits low
- Use PHP cli for crons (not wget http://yoursite.com/cron.php) and tune PHP cli with a higher memory limit
- Use APC or similar!!!

Tuning MySQL

- AVOID SWAPPING
- Innodb:
  - innodb_buffer_pool_size
  - innodb_flush_method = O_DIRECT|ALL_O_DIRECT
  - innodb_flush_log_at_trx_commit = 2
- Don't use query cache, but try it in a pinch
- Unused cache sizes small or 0: key_buffer_size, query_cache_size, etc.
- Keep various buffers as low as possible (and/or defaults):
  - *_buffer_size (sort, join, etc.)
  - max_tmp_table + max_heap_table_size low
  - table_cache, thread_cache_size, etc.

Progression of Architecture Choices

- Lighttpd + Apache + mod_php
- Varnish + Apache + mod_php
- Apache (threaded, keepalive) + php-fpm
- Cloudflare + Apache + php-fpm
  - Best of breed
  - Gives you standard LAMP stack w/o custom config
  - OSS .htaccess works as expected
  - Cloudflare works pretty transparently*

Non-intuitive Steps

- RAM is your most precious commodity, pinch every byte
- Some swap usage is inevitable.
  - VPS vendors won't help you when you have swap usage
- Don't run things as daemons you can run as cronjobs
  - puppetd
  - denyhosts

RAM Tuning

- Apache modules
- Apache MaxClients
- Varnish cache
- APC cache
- PHP processes
- Innodb buffer pool
- Extra daemons
- MyISAM key buffer
- Query cache
- MySQL per-connection buffers
- Innodb flush method
- Binlog cache size
- swappiness
- Filesystem cache
- 32-bit vs 64-bit OS
- Persistent connections
- Thread cache

The Results

SIM.PLIFIED.COM

Chris Pollock - web developer & ecommerce entrepreneur
undivided… my thoughts on world, family, church, business, technology and Jesus Christ (all in all)

HOME frontpage | ARCHIVES browse freely | WEB web devlopment | FAMILY my tribe | BODY the church | SCRIPTURES word of god | RSS syndicate | MAIN skip to content

AUG 17, 2008

Pattern Tap: We dig Jay because we got Dugg and not buried

On Sunday, August 10, one week ago, we experienced our first hit from Digg. We had talked about the possibility of being Dugg but were by no means prepared for what that meant. On our biggest day previous to this we saw a max of around 9k visitors. Looking at the graph below you can see we far exceeded that topping out at almost 50k visitors on that one day.

So what.. people get dugg all the time, what's the big whip? Well, currently PT is sitting on a shared 256MB of RAM. This hit should of sent our server home crying and returning all sorts of "sorry, we v types of errors.

However, thanks to the good foresight of our kung fu server master, Jay Janssen, the server did not continued to gracefully serve pages as the onslaught of visitors continued. Jay is a mysql guru who also a good friend and my roommate from R.I.T.).

The secret sauce was lighttp, a server component that allows you to bypass apache for non dynami

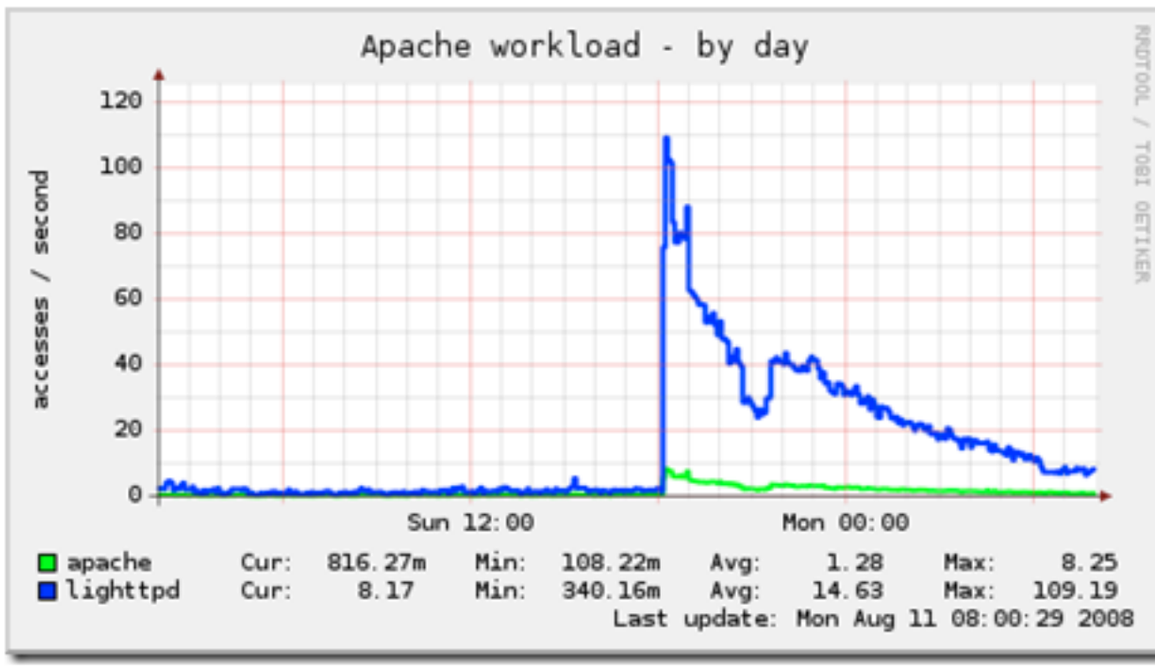
Things I'm Saying

Hey designers and friends of designers, here's our latest RFQ for simplified safety http://t.co/uRumsq64 # 18 hours ago

If you use jQuery cycle, you should be aware of this issue on chrome http://t.co/ySInzsG4 # 2012/02/28

A letter from the future! http://t.co/xgcWZobR # 2012/02/28

and now he's skyping me!!!! so funny!! #



Conclusion

- We vastly underutilize our servers
- Proper Systems administration is balancing resource utilization with good tuning
  - Give the best customer experience
  - For the lowest cost to your company
- These principles apply if you have a single server or 100k