phase2 TECHNOLOGY

THE LEADER IN DRUPAL PLATFORM DESIGN AND DEVELOPMENT

SHORT STACK

INGREDIENTS FOR A SHORTER, SWEETER DRUPAL HOSTING STACK

phase:II TECHNOLOGY

http://www.flickr.com/photos/crobj/3306760492/

YOUR HOST

STEVEN MERRILL

Director of Engineering

smerrill@phase2technology.com

@stevenmerrill

https://github.com/smerrill

phase:II TECHNOLOGY

ABOUT ME

- Big fan of virtualization and configuration management
- Architected hosting solutions for big Drupal platforms
  - Cloud
  - Dedicated
  - Hybrid

WHAT WILL BE COVERED?

- Conventional Drupal hosting stacks
- Deep tactical detail on:
  - PHP-FPM
  - nginx to run your whole web tier
    - Replicate a common Varnish configuration

phase:II TECHNOLOGY

WHAT WILL BE COVERED?, CONT'D

- More theoretical detail on:
  - Postgres
  - Redis

phase:II TECHNOLOGY

A TWITTER EXCHANGE

PEOPLE HAVE OPINIONS!

phase:II TECHNOLOGY

Can I just say that I love @nginxorg as an SSL terminating proxy and @varnishcache as a reverse proxy cache? They make a very powerful pair.

@stevenmerrill

Fan of Varnish and nginx

phase:II TECHNOLOGY

> @stevenmerrill Any reason not to skip varnish and use Nginx directly for everything with [upstream memcached for Drupal page cache] ? Nginx cache is very good.

@cweagans

Fan of nginx

> **@stevenmerrill Why the need for Varnish? #nginx has a highly performant cache also. Why complicate what can be simple?**

@perusio

Maintainer of a great Drupal nginx configuration

> **@perusio @cweagans I like Varnish's CLI tools (varnishtop / varnishhist) and the ability to do targeted bans on response headers.**

@stevenmerrill

Already had this talk planned

phase:II TECHNOLOGY

THE BIG STACK

WHAT PROBLEMS DO WE HAVE TO SOLVE IN A HOSTING ENVIRONMENT?

phase:II TECHNOLOGY

http://www.flickr.com/photos/waytru/528949873/

phase:II TECHNOLOGY

THE WEB SERVER AND PHP

- Generating PHP pages
- Compressing static resources
- Serving content to end-users or CDNs
- SSL termination

LOAD BALANCING

- HTTP
  - LRU
  - Lowest connections
- TCP
  - MySQL read balancing / HTTPS traffic

CACHING

- Object caching
- Reverse proxy caching
- Content distribution
- DRY
- Clearing x layers of caching

phase:II TECHNOLOGY

DATA STORAGE

- Relational data
- Counters / statistics
- Full-text search
- Faceting

THE BUFFET

WHAT SOFTWARE TYPICALLY GETS USED?

phase:II TECHNOLOGY

LOAD BALANCING AS A SERVICE

- Amazon ELBs
- Rackspace Cloud Load Balancers
- Linode NodeBalancers
- Zeus or F5 Hardware Load Balancers

WEB / PROXY / LOAD BALANCING

- Apache
- Varnish
- HAProxy

phase:II TECHNOLOGY

APACHE AND PHP

- Version 2.2
- mod_php
- Prefork MPM
  - Thread-safety of PHP core and modules
- mod_proxy

VARNISH

- Reverse proxy cache
- HTTP load balancer
- Memory or disk (ephemeral) storage
- Clear cache by URL or headers
- CLI tools

MEMCACHED

- Key-value store
- Small vocabulary of simple commands
- Two PHP extensions

HAPROXY

- Advanced HTTP load balancing
- TCP load balancing
- DDoS mitigation
- Web interface
- SSL termination in beta

SOLR

- Java-based web service atop Lucene
- Configurable query- and index-time features
  - Stemming
  - Tokenization
- Boosting of various parts of a query

phase:II TECHNOLOGY

THE RECIPE

A MODEST PROPOSAL FOR SOME NEW INGREDIENTS

phase:II TECHNOLOGY

http://www.flickr.com/photos/theknowlesgallery/5482189257/

REDIS

KEY/VALUE STORAGE WITH PERSISTANCE

DATA STRUCTURE SERVER

phase:II TECHNOLOGY

REDIS IN A NUTSHELL

- Data structures servers
- Persistent to disk; configurable intervals
- Pub/sub and queueing framework
- Lua built in since 2.6 for extra commands
- Connect with pure PHP driver or PHP extension

POSTGRES

HISTORY W/ POSTGRES

- TBM in 2006
  - Views and Date were very rough
- PGSQL support in 2012
  - Quite good out of the box
  - Install a distro!

POSTGRES 9.2 FEATURES

- External data wrappers
- Better replication
- JSON as a native datatype
- Lua programmability

POSTGRES 9.2 FEATURES

- PostGIS geographic data framework
- Great support for full-text search and tokenization
  - `tsearch2` integrated since 8.0.3
- 9.2 released recently

POSTGRES ISSUES

- No testbot!
- http://drupal.org/node/1668644
  - Changing some field types breaks
  - Cannot test automatically

PHP-FPM

phase:II TECHNOLOGY

PHP AS A FASTCGI DAEMON

- More advanced than other FastCGI solutions (spawn-fcgi)
  - Can share APC cache among processes
- Listen on a port (usually 9000)
- Listen on a socket
  - No TCP overhead

PHP-FPM, CONT'D

- Availability
  - In Ubuntu 12.04 LTS
  - In IUS for RHEL and CentOS 5 and 6

UPGRADE YOUR APACHE

- Consider using PHP-FPM with your Apache setup
- mod_fcgid
    - Wants to handle process management
    - Can't connect to an external daemon
- mod_fastcgi

MOD_ FASTCGI CONFIGURATION

```
<IfModule mod_fastcgi.c>
    Alias /php5.fcgi /var/www/php5.fcgi
    FastCGIExternalServer /var/www/php5.fcgi \
      -flush -port 127.0.0.1:9000
    AddType application/x-httpd-fastphp5 .php
    Action application/x-httpd-fastphp5 /php5.fcgi
</IfModule>
```

MOD_FASTCGI CONFIGURATION, CONT'D

```
<IfModule mod_fastcgi.c>
    Alias /php5.fcgi /var/www/php5.fcgi
    FastCGIExternalServer /var/www/php5.fcgi \
      -flush -socket /var/run/php-fpm.sock
    AddType application/x-httpd-fastphp5 .php
    Action application/x-httpd-fastphp5 /php5.fcgi
</IfModule>
```

BENEFITS

- Run another MPM
  - Worker on Apache 2.x
  - Event MPM on Apache 2.4
- No need to size Apache based on `memory_limit`
- Control security with FPM pools, not `safe_mode`

phase:II TECHNOLOGY

SIZING PHP-FPM RESOURCE USAGE

- `pm.static` will have `pm.max_children` children
- `pm.ondemand` and `pm.dynamic` start `0` or `pm.start_servers` children and go up to `pm.max_children` children
- Like prefork MPM

phase:II TECHNOLOGY

MORE PHP-FPM CONFIGURATION

- Have an HTTP-based health check for PHP-FPM
- Set `ping.path = /ping` and pass it through

```
<LocationMatch "/ping">
  SetHandler application/x-httpd-fastphp5
  Action application/x-httpd-fastphp5 /php5.fcgi virtual
</LocationMatch>
```

EVEN MORE PHP-FPM CONFIGURATION

- Replicate `mod_status` for your FPM processes
- Set `pm.status_path = /status` and pass it through

```
<LocationMatch "/status">
  SetHandler application/x-httpd-fastphp5
  Action application/x-httpd-fastphp5 /php5.fcgi virtual
</LocationMatch>
```

PHP-FPM STATUS, CONT'D

- Many different outputs based on the query string
- http://localhost/status
- http://localhost/status?full
- http://localhost/status?json&full

phase:II TECHNOLOGY

PHP-FPM STATUS OUTPUT

```
pool:                 www
process manager:      ondemand
start time:           03/Nov/2012:11:33:26 -0700
start since:          980
accepted conn:        24
idle processes:       0
active processes:     1
total processes:      1
max active processes: 1
max children reached: 0
```

NGINX

WEB SERVER

LOAD BALANCER

REVERSE PROXY CACHE

phase:II TECHNOLOGY

REVERSE PROXY CACHING

HOW DOES IT WORK?

(IN MOST DRUPAL / VARNISH CONFIGURATIONS)

phase:II TECHNOLOGY

THE FASTEST ANONYMOUS REQUESTS

- Don't bootstrap Drupal*
  - What about statistics.module?
- Reverse proxy caches (including CDNs like Akamai) serve anonymous and static content _fast_
- Keep your backend alive under a traffic surge

phase:II TECHNOLOGY

VARNISH FTW

- Stores (non-persistent) cache on disk or in memory
- Can purge a specific URL based on an HTTP request
- Can ban a set of content based on header matches
- Has "grace mode" to avoid the thundering herd problem
- Works w/ most Vary headers natively

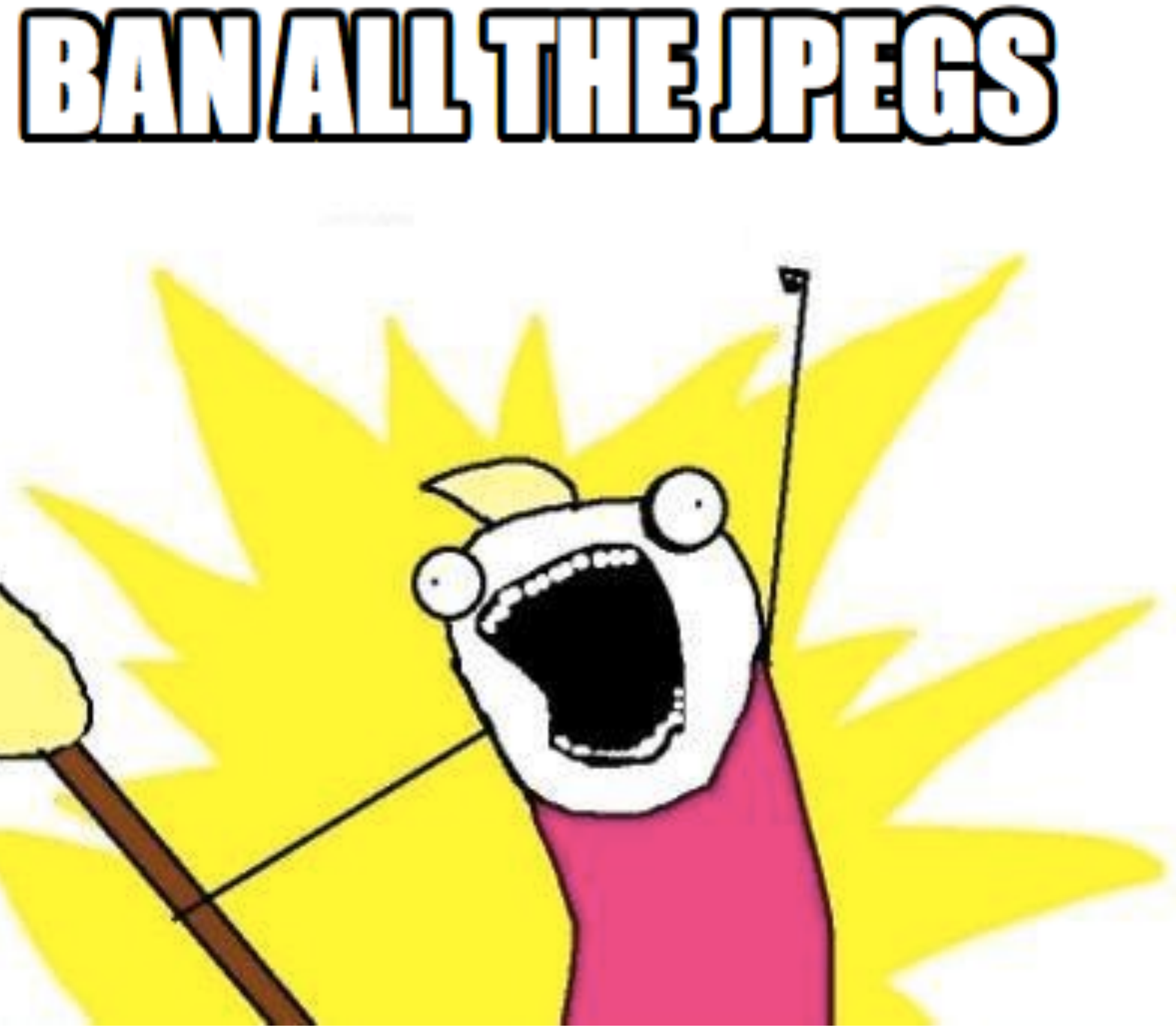
VARNISH CLI TOOLS

- Varnish has great CLI tools
- `varnishadm` to interact w/ the control terminal
- `varnishstat` to see cache hitrate over time
- `varnishtop` to view weighted totals of varnishlog entries
- `varnishlog` to view streaming Varnish logs

VARNISH BANNING

- Ban anything from cache via headers or request criteria
- `ban.url ~ "^/node/1"`
- `ban req.url ~ "^/node/1"`
- `ban obj.http.x-host ~ "^/node/1"`

phase:II TECHNOLOGY


BAN ALL THE JPEGS

VARNISH WILDCARD BANNING

- `ban req.url ~ "\\.jpg"`
- `ban obj.http.Content-Type == "image/jpeg"`

phase:II TECHNOLOGY

VARNISH FTL

- ESI
- No way to have a persistent cache
- Streaming support is very recent
- Cannot terminate SSL

A STANDARD DRUPAL VARNISH CONFIG

- Don't cache anything but GET/HEAD requests
- Strip non-essential cookies (all but `S?SESS[a-z0-9]+`)
- Don't cache `cron.php` or `update.php`
- Munge `Accept-Encoding` headers for high hit rates
- Allow stale content to be served while updating it

CAN NGINX DO THIS?

- Yes, with one server on port 80
- Shortcomings:
  - No Age header
  - No bans
  - Re-gzip all the things

NGINX CACHING

- Two different modes
  - Cache an upstream server with `proxy_*` directives
  - Cache output from PHP using `fastcgi_*` directives
- We'll examine `fastcgi_*` to just cache dynamic content

VARNISH VS NGINX SETUP

- Replicate the salient bits
- Point by point
- Unless otherwise specified:
  - nginx code is inside `location ~ \.php { }`
  - Varnish code is in `vcl_recv`

phase:II TECHNOLOGY

SETUP: CACHE 10M OF DATA IN RAM

- Varnish
  - Add `-s malloc,10m` to the startup options
- nginx
  - `fastcgi_cache_path /dev/shm/nginx levels=1:2 keys_zone=one:10m;`

SETUP: DEFAULT 10 MINUTE TTL

- Varnish
  - Add `-t 600` to the startup options
- nginx
  - `fastcgi_cache_valid 200 302 301 10m;`

SETUP: CONFIGURE NGINX TO USE FPM

```
root           /var/www;
fastcgi_pass   127.0.0.1:9000;
fastcgi_index  index.php;
fastcgi_param  SCRIPT_FILENAME
  $document_root$fastcgi_script_name;
include        fastcgi_params;
```

CACHE GET/HEAD REQUESTS: VARNISH

```
if (req.request != "GET" && req.request != "HEAD") {
  /* We only deal with GET and HEAD by default */
  return (pass);
}
```

CACHE GET/HEAD REQUESTS: NGINX

```
# Instead of return (pass), we have a $cache_bypass
variable.
set $cache_bypass "";

# Only cache GET or HEAD requests.
if ($request_method !~ ^(GET|HEAD)$) {
  set $cache_bypass "1";
}
```

DON’T CACHE *.PHP: VARNISH

```
// Skip the cache for install, update, and cron
if (req.url ~ "(install|update|cron)\.php") {
  return (pass);
}
```

DON'T CACHE *.PHP: NGINX

```
# Don't cache install, update, or cron
if ($request_uri ~* "(install|update|cron)\.php") {
  set $cache_bypass "1";
}
```

STRIP COOKIES: VARNISH

```
// Whitelist the Drupal session cookie and "p2_" cookies.
if (req.http.Cookie) {
  set req.http.Cookie = ";" + req.http.Cookie;
  set req.http.Cookie = regsuball(req.http.Cookie, "; +", ";");
  set req.http.Cookie = regsuball(req.http.Cookie,
    ";(S?SESS[a-z0-9]+|p2_)=", "; \1=");
  set req.http.Cookie = regsuball(req.http.Cookie, ";[^ ][^;]*", "");
  set req.http.Cookie = regsuball(req.http.Cookie, "^[; ]+|[; ]+$", "");
}
```

STRIP COOKIES: VARNISH, CONT'D

```
// Remove a ";" prefix, if present.
set req.http.Cookie = regsub(req.http.Cookie, "^;\s*", "");

// Remove empty cookies.
if (req.http.Cookie ~ "^\s*$") {
  unset req.http.Cookie;
}
```

phase:II
TECHNOLOGY

STRIP COOKIES: VARNISH, CONT'D

```
sub vcl_hash {
  if (req.http.Cookie) {
    hash_data(req.http.Cookie);
  }
}
```

STRIP COOKIES: NGINX

```
# Emulate Varnish's cookie stripping behavior.
set $stripped_cookie "";

# Allow a session cookie or any p2_* cookie through.
if ($http_cookie ~* "(S?SESS[a-f0-9]+|p2+[a-z0-9]+)=([^;]+)(?:;|$)" ) {
  set $stripped_cookie "$1=$2";
}

fastcgi_param HTTP_COOKIE $stripped_cookie;
```

STRIP COOKIES: NGINX, CONT'D

```
# Cache by URL and since nginx cache does not use Vary headers, add any
# of the things that we know that we might be called upon to Vary on.
fastcgi_cache_key $request_method$scheme$host$request_uri
$stripped_cookie;
```

MUNGE ACCEPT-ENCODING: VARNISH

```
// Normalize the Accept-Encoding header
if (req.http.Accept-Encoding) {
  if (req.url ~ "\.(jpg|png|gif|gz|tgz|bz2|tbz|mp3|ogg)$") {
    remove req.http.Accept-Encoding;
  }
  elsif (req.http.Accept-Encoding ~ "gzip") {
    set req.http.Accept-Encoding = "gzip";
  }
  else {
    remove req.http.Accept-Encoding;
  }
}
```

MUNGE ACCEPT-ENCODING: NGINX

- One weak point for the `fastcgi_*` caching method
- PHP would have to gzip; nginx would have to add the `fastcgi_cache_key`

MUNGE ACCEPT-ENCODING: NGINX

```
# Munge Accept-Encoding.
set $munged_accept_encoding "";
if ($http_accept_encoding ~* "gzip") {
  set $munged_accept_encoding "gzip";
}

fastcgi_param HTTP_ACCEPT_ENCODING $munged_accept_encoding;

gzip on;
gzip_comp_level 1;
```

PURGE A URL: VARNISH

```
sub vcl_hit {
  if (req.http.X-Purge == "true") {
    purge;
    error 200 "Purged.";
  }
}
```

PURGE A URL: NGINX

```
# Allow cache to be cleared by setting an X-Purge: true header.
if ($http_x_purge = "true") {
  set $cache_bypass "1";

  # Use the ngx_cache_purge module. Purge from the "one" zone.
  fastcgi_cache_purge one \
    $request_method$scheme$host$request_uri$stripped_cookie;
}

# Bypass any cache if a user is uncacheable.
fastcgi_no_cache $cache_bypass;
fastcgi_cache_bypass $cache_bypass;
```

SHOW CACHE HIT/MISS: VARNISH

```
sub vcl_deliver {
  if (obj.hits > 0) {
    set resp.http.X-Cache = "HIT";
  } else {
    set resp.http.X-Cache = "MISS";
  }
}
```

SHOW CACHE HIT/MISS: NGINX

```
# If we're uncacheable, say so.
if ($cache_bypass = "1") {
  add_header X-Cache "NO";
}
# Otherwise, show the nginx caching status.
if ($cache_bypass = "") {
  add_header X-Cache $upstream_cache_status;
}
```

DO GRACE MODE: VARNISH

```
sub vcl_recv {
  // Allow a stale objects up to 2 hours.
  set req.grace = 2h;
}
sub vcl_fetch {
  // Allow a 2 hour grace period if
  // our backend is unhealthy.
  set beresp.grace = 2h;
}
```

DO GRACE MODE: NGINX

```
# Emulate Varnish's grace mode.
fastcgi_cache_use_stale updating;
```

MINIDEMO

MORE READING

phase:II
TECHNOLOGY

NGINX

- https://github.com/perusio/drupal-with-nginx
- https://github.com/perusio/nginx-cache-purge
- https://github.com/perusio/nginx-cache-inspector
- http://openresty.org/

phase:// TECHNOLOGY

phase2technology.com

@phase2tech