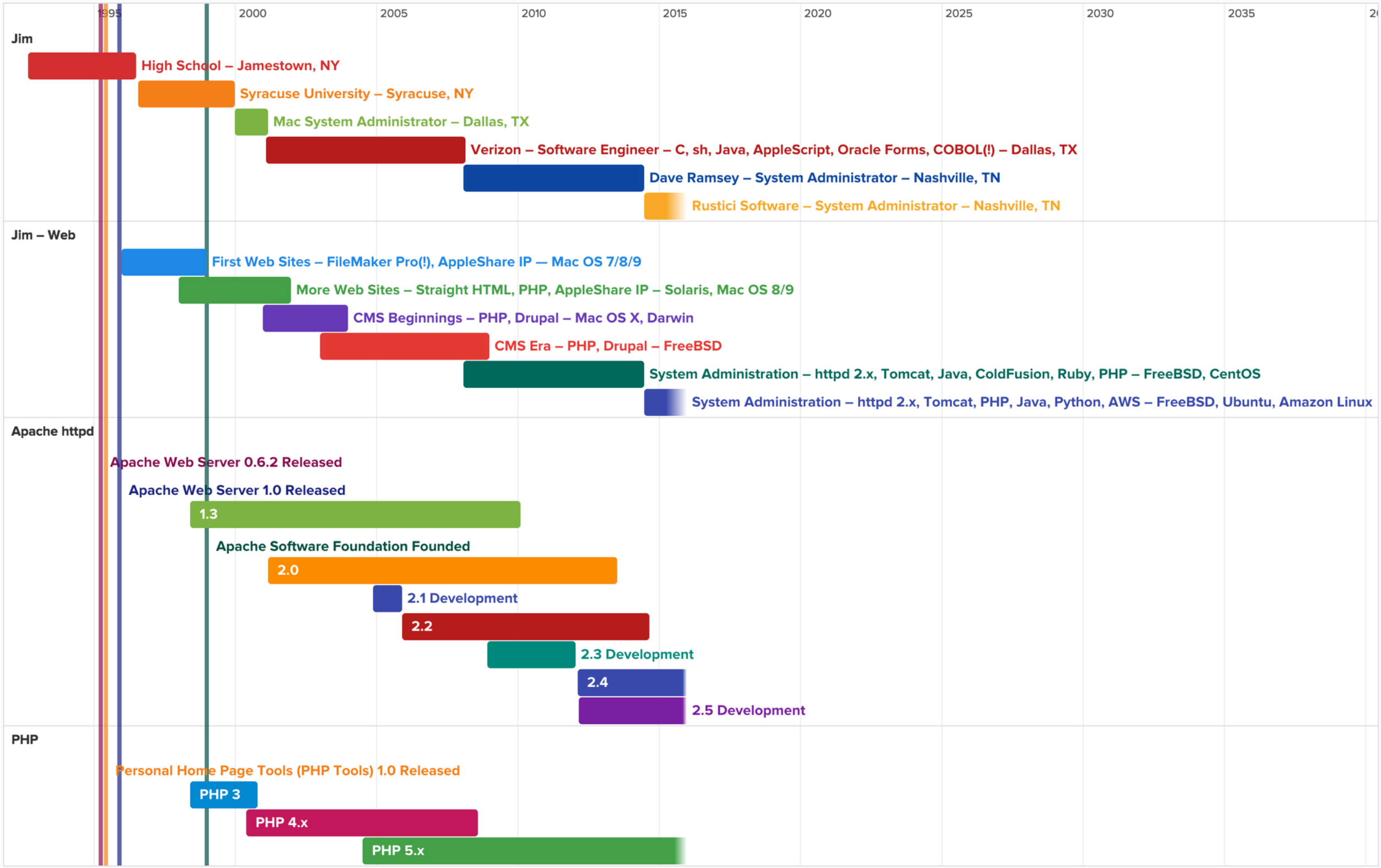
Jim Riggs, System Administrator, Rustici Software

Begone mod_php!

It is time to finally treat PHP like the application that it is!

Timeline


1995
2000
2005
2010
2015
2020
2025
2030
2035
Jim
High School – Jamestown, NY
Syracuse University – Syracuse, NY
Mac System Administrator – Dallas, TX
Verizon – Software Engineer – C, sh, Java, AppleScript, Oracle Forms, COBOL(!) – Dallas, TX
Dave Ramsey – System Administrator – Nashville, TN
Rustici Software – System Administrator – Nashville, TN
Jim – Web
First Web Sites – FileMaker Pro(!), AppleShare IP — Mac OS 7/8/9
More Web Sites – Straight HTML, PHP, AppleShare IP – Solaris, Mac OS 8/9
CMS Beginnings – PHP, Drupal – Mac OS X, Darwin
CMS Era – PHP, Drupal – FreeBSD
System Administration – httpd 2.x, Tomcat, Java, ColdFusion, Ruby, PHP – FreeBSD, CentOS
System Administration – httpd 2.x, Tomcat, PHP, Java, Python, AWS – FreeBSD, Ubuntu, Amazon Linux
Apache httpd
Apache Web Server 0.6.2 Released
Apache Web Server 1.0 Released
1.3
Apache Software Foundation Founded
2.0
2.1 Development
2.2
2.3 Development
2.4
2.5 Development
PHP
Personal Home Page Tools (PHP Tools) 1.0 Released
PHP 3
PHP 4.x
PHP 5.x

Bothersome Things

```
User www
Group www

DocumentRoot /var/www
```

```
% sudo chmod -R 777 /var/www
```

```
% sudo chown -R www:www /var/www
```

```
% ls -la /var/www/*

/var/www/example.com:
total 16
drwxrwxrwx  5 www  www   170 Apr  9 2001 .
drwxrwxrwx  4 www  www   136 Apr 10 2001 ..
-rwxrwxrwx  1 www  www   537 Apr  9 2001 config.php
drwxrwxrwx  2 www  www    68 Apr 10 2001 data
-rwxrwxrwx  1 www  www  1723 Apr  6 2001 index.php

/var/www/example2.com:
total 16
drwxrwxrwx  5 www  www  170 Apr 10 2001 .
drwxrwxrwx  4 www  www  136 Apr 10 2001 ..
-rwxrwxrwx  1 www  www  395 Apr  9 2001 config.php
drwxrwxrwx  2 www  www   68 Apr 12 2001 data
-rwxrwxrwx  1 www  www  938 Apr  9 2001 index.php
```

Thus begins…

The Quest

Surely, there must be some way to:

- isolate applications or virtual hosts from each other (user/group)
- not have the public-facing web server own or have write access to code, data, or configuration

Crusade #1

safe_mode, open_basedir, disable_functions, etc.

Pros	Cons
❖ uses existing mod_php setup ❖ relatively simple to implement ❖ doesn't break (too much) code	❖ still sharing a user / group ❖ prone to configuration error ❖ frustration results in disabling restrictions

Crusade #2

CGI, setuid / setgid, suEXEC

Pros	Cons
❖ distinct user / group ❖ (fairly?) well tested / verified / validated for security	❖ setuid / setgid ❖ *so many restrictions* ❖ easy to misconfigure

Crusade #3

per-user, per-vhost, user-/group-changing MPMs, etc.

Pros	Cons
❖ distinct user/group	❖ worker(s) running as root ❖ core patches ❖ stalled development ❖ not production-ready

Crusade #4

master-slave proxy httpd instances

Pros	Cons
❖ distinct user/group per slave instance ❖ -D + IfDefine allows for shared master/slave configuration files	❖ resource usage ❖ tricky configuration ❖ lots of proxying and listening

"So, what does it look like today?"

— All of you

```
DirectoryIndex "index.php"
AddType application/x-httpd-php .php

php_admin_flag engine off
php_admin_value disable_functions "chgrp, chown, dl, exec, link, passthru, popen, shell_exec, system"
php_admin_value open_basedir "/nonexistent/"
php_admin_value max_execution_time 120
php_admin_value memory_limit "64M"
php_admin_value post_max_size "64M"
php_admin_value upload_max_filesize "64M"

<VirtualHost *:80>
    ServerName "example.com"
    DocumentRoot "/var/www/example.com/public"

    <Directory "/var/www/example.com/public">
        Require all granted
        php_admin_flag engine on
        php_admin_value open_basedir "/var/www/example.com/"
        php_admin_value upload_tmp_dir "/var/www/example.com/tmp/"
        php_admin_value session.save_path "/var/www/example.com/tmp/"
        php_admin_value memory_limit "128M"
    </Directory>
</VirtualHost>

<VirtualHost *:80>
    ServerName "example2.com"
    DocumentRoot "/var/www/example2.com/public"

    <Directory "/var/www/example2.com/public">
        Require all granted
        php_admin_flag engine on
        php_admin_value open_basedir "/var/www/example2.com/"
        php_admin_value upload_tmp_dir "/var/www/example2.com/tmp/"
        php_admin_value session.save_path "/var/www/example2.com/tmp/"
    </Directory>
</VirtualHost>
```

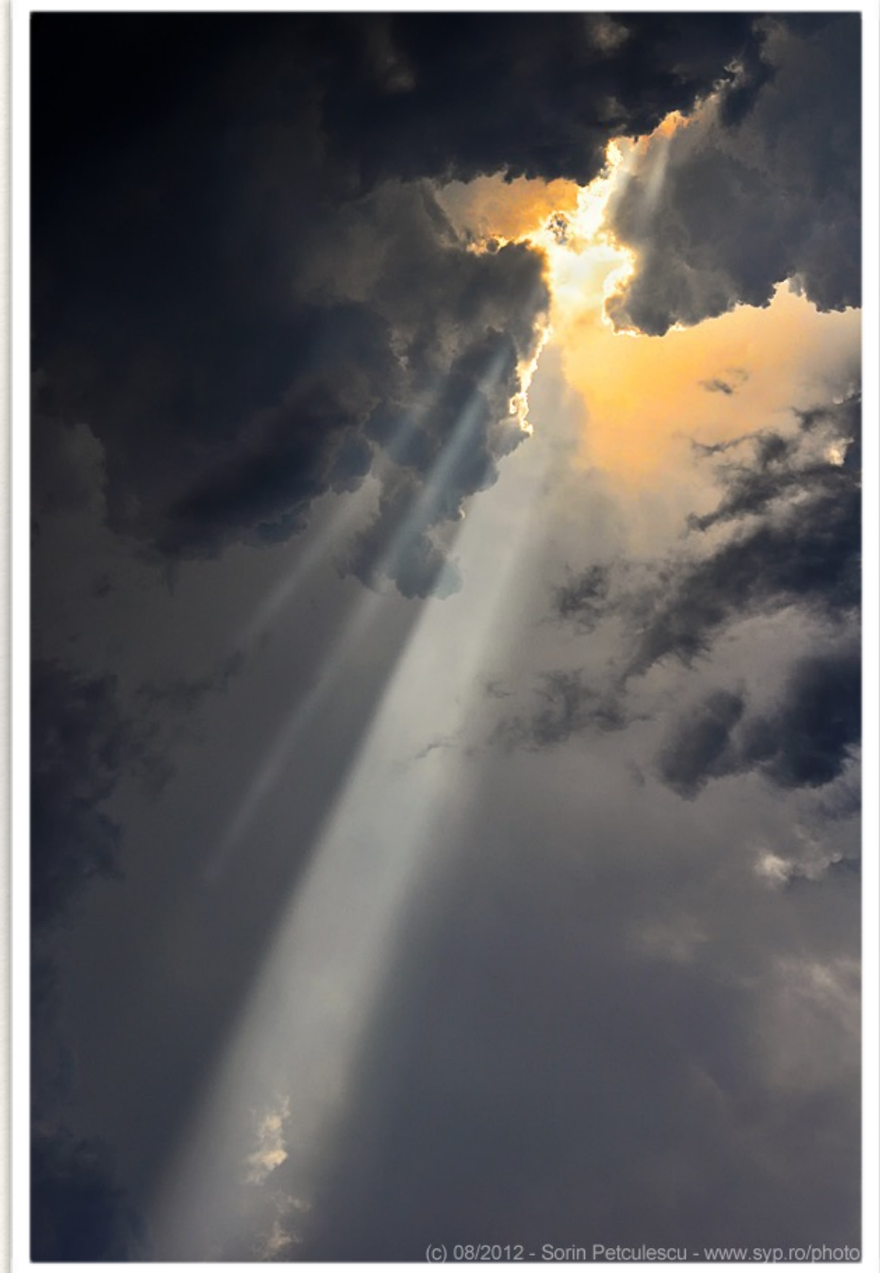
Sample Results

JMeter test: 50 threads, 50 loops

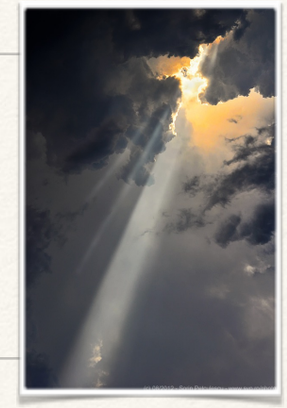
Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
HTTP Request	2500	1008	1026	1094	198	4058	0.00%	48.4/sec	5266.0
TOTAL	2500	1008	1026	1094	198	4058	0.00%	48.4/sec	5266.0

Light Shines into the Darkness

- **December 2005:**
  mod_proxy_fcgi appears in trunk courtesy of pquerna shortly after 2.2.0 is released
- **2006:**
  jim, rooneg, and others pick up the torch
- **2007–2009:**
  The Quiet Years™
- **2010–present:**
  development accelerates, and the code is production-ready as part of the 2.3 and 2.4 development cycle



mod_proxy_fcgi

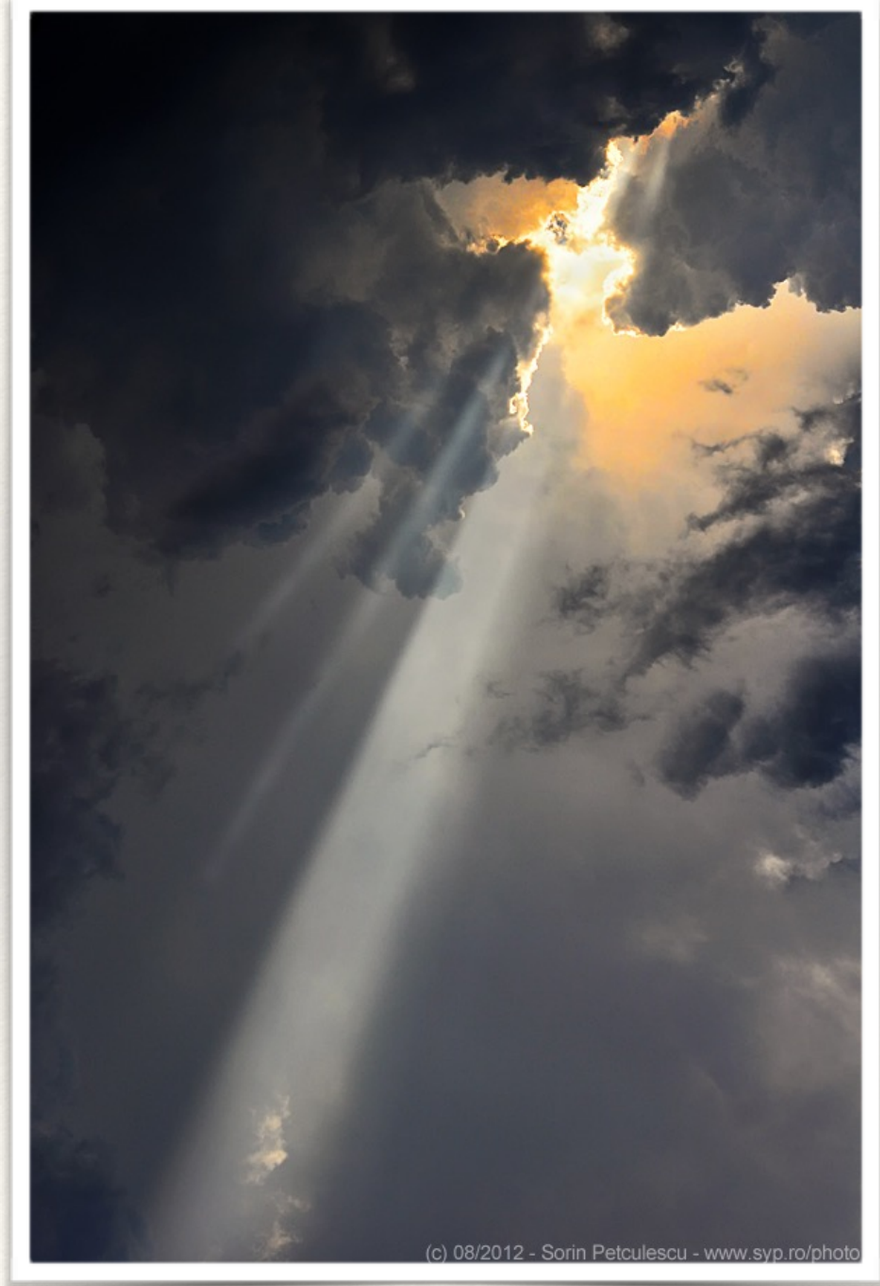


- use familiar mod_proxy directives:
  - `ProxyPass` & `ProxyPassReverse`
  - `BalancerMember fcgi://...`*
  - `RewriteRule ... fcgi://... [P]`
- run FastCGI process locally or remotely, so you can:
  - move application processing to application tier
  - spread resource load across backend servers
  - share backend server(s) with multiple frontend servers

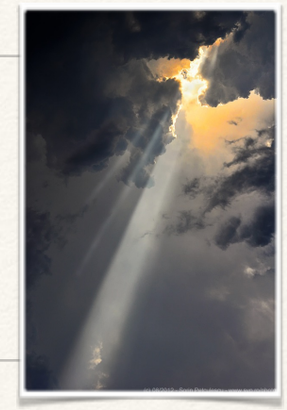
* use of mod_proxy_fcgi in a load balancer with FPM actually requires a patch to PHP

Meanwhile in PHP Land…

- **June 2007:**
  Andrei Nigmatulin releases PHP-FPM 0.1, a patch against PHP 4.4.7, with the intent of making a production-ready PHP FastCGI implementation
- **22 July 2010:**
  PHP 5.3.3 released with FPM included as an SAPI



PHP-FPM



- works as a process manager using a multi-process, child/worker model similar to the prefork MPM
- handles multiple "pools" of configuration settings (think of each pool as an application or domain or user) that can:
  - have their own configuration (user, group, listen settings, limits, PHP directives, etc.)
  - spawn, kill, and restart workers/children

Putting Them Together

How does this change things?
Is this the grail?

```
[example.com]
user = 8000
group = 8000

chroot = "/var/www/$pool"

listen = 127.0.0.1:8000
listen.allowed_clients = 127.0.0.1

pm = dynamic
pm.max_children = 5
pm.start_servers = 2
pm.min_spare_servers = 1
pm.max_spare_servers = 3

php_admin_value[disable_functions] = "chgrp, chown, dl, exec, link, passthru, popen, shell_exec, system"
php_admin_value[max_execution_time] = 120
php_admin_value[memory_limit] = "128M"
php_admin_value[post_max_size] = "64M"
php_admin_value[upload_max_filesize] = "64M"
php_admin_value[upload_tmp_dir] = "/tmp/"
php_admin_value[session.save_path] = "/tmp/"

[example2.com]
user = 8001
group = 8001

chroot = "/var/www/$pool"

listen = 127.0.0.1:8001
listen.allowed_clients = 127.0.0.1

...
```

```
DirectoryIndex "index.php"

<VirtualHost *:80>
    ServerName "example.com"
    DocumentRoot "/var/www/example.com/public"

    <Directory "/var/www/example.com/public">
        Require all granted
    </Directory>

    ProxyPassMatch "^/+(.*\.php)$" "fcgi://127.0.0.1:8000/public/$1"
</VirtualHost>

<VirtualHost *:80>
    ServerName "example2.com"
    DocumentRoot "/var/www/example2.com/public"

    <Directory "/var/www/example2.com/public">
        Require all granted
    </Directory>

    ProxyPassMatch "^/+(.*\.php)$" "fcgi://127.0.0.1:8001/public/$1"
</VirtualHost>
```

Sample Results

JMeter test: 50 threads, 50 loops

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
HTTP Request	2500	1198	1263	1315	94	7169	0.00%	39.2/sec	4259.2
TOTAL	2500	1198	1263	1315	94	7169	0.00%	39.2/sec	4259.2

Summary

mod_proxy_fcgi + PHP-FPM

Pros	Cons
❖ distinct user/group per FPM pool ❖ PHP *application* can reside, process, and utilize resources in a different address space, in a separate jail, or on a separate host/VM ❖ httpd can load-balance to multiple FPM backends or multiple httpd frontends can share a single FPM backend	❖ latency ❖ PHP-FPM process manager tuning ❖ mod_proxy configuration

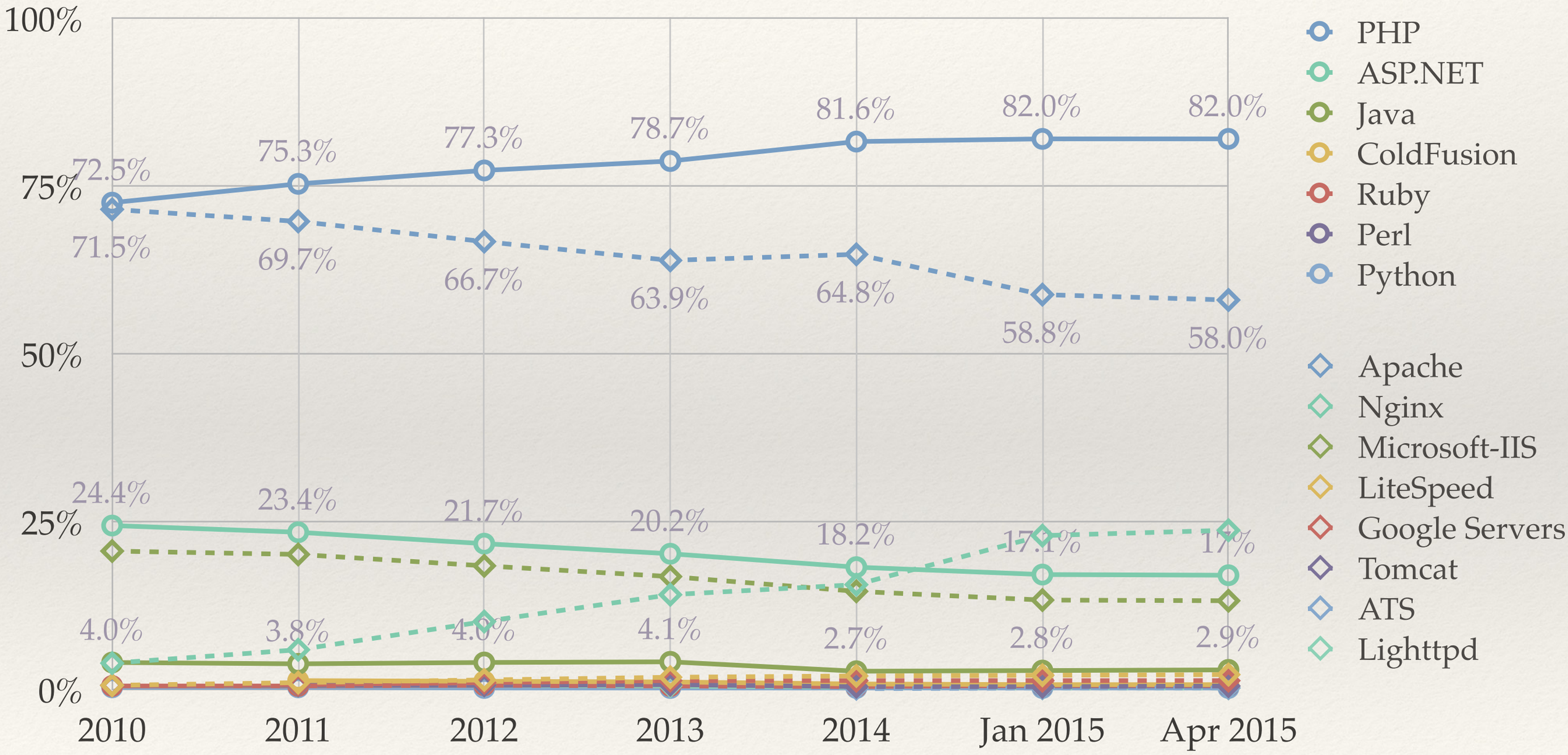
Considerations

- **File system:**
  If httpd and PHP-FPM are not sharing a file system, the files must be synchronized, proxied, or both.
- **Access control:**
  - Be sure that httpd only serves/allows access to the files it should.
  - Take extra care with Files, Directory, Location, and Require directives.
- **File ownership:**
  - Ensure httpd has read-only access. (And only to what it needs!)
  - Ensure PHP has read-only access to everything except things that ***require*** write access.

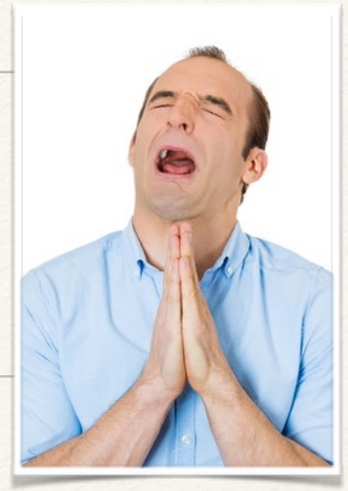
"Great, but who cares? This was a valuable conversation ten years ago. Who really needs this information today? PHP is so passé!"

— *All of you*

Still Relevant Today



Pleas from Your Friendly SysAdmin



- Treat PHP just like any other application language. *Would you run Java, Ruby, or Language X in the frontend?* Run it in the application "tier" (whatever that looks like in your environment).
- ACLs, ownership, and permissions matter, so limit write ***and read*** access to as-needed. ***When*** http/PHP/LangX gets compromised, what does it have access to?
- Let httpd do what it does best: serve static content quickly, efficiently, and securely. Proxy everything else to backend processes that can do their work in the same way.

Questions