

Bottom to Top Stack Optimization

With LAMP + CodeIgniter

Server side: It's About I/O

Network I/O

Disk I/O

CPU

Server/Client side:
Minimize and Cache Assets

Optimization Roadmap

- System
- MySQL
- Apache
- PHP
- Framework
- Application

Testing

- Frontend
 - Yslow
- Backend
 - `ab`
 - vmstat

LAMP Configuration

- By default, for “compatibility”
- You’re responsible for your boxes

System

- Don't swap
 - Swap in itself isn't bad, swapping out is
- Get rid of non-essential processes
- atime – Filesystem option to write 'last-accessed' time to everything
 - \$ cat /
- If you're a badass, check into XFS

atime

```
$ cat /etc/fstab
```

```
proc          /proc        proc         defaults     0 0
/dev/sda1     /            ext3        defaults,errors=remount-ro,noatime 0 1
/dev/sda2     none         swap        sw           0 0
```

MySQL

- Don't "SELECT *"
 - If the data you need is in an index, MySQL will grab that
- Column indexes
 - Foreign key type matching
 - Be smart about column size
 - Integrity checks? Try NOT deleting instead
- Batched queries
- Key Buffer
- Query Buffer
- Engine
 - InnoDB, or go crazy and use Percona XtraDB with XFS

Apache

- Kill unneeded modules
- Tune MaxClients
- Eliminate .htaccess, if possible
- Kill access logs
- Disable ETags if not used
- Expires Headers
- Compression
- Don't serve static content (if you can avoid it)

Benchmarking

- Apache bench is most common
 - Tests Requests Per Second
- Use vmstat to watch swap usage
- Send 1,000 requests, concurrency of 10

```
$ ab -n 1000 -c 10 http://getsparks.org
```

Finished 10000 requests

Server Software: Apache/2.2.12

Server Hostname: getsparks.org

Server Port: 80

Document Path: /

Document Length: 9865 bytes

Concurrency Level: 10

Time taken for tests: 6.040 seconds

Complete requests: 10000

...

Requests per second: 1655.63 [#/sec] (mean)

Time per request: 6.040 [ms] (mean)

Time per request: 0.604 [ms] (mean, across all concurrent requests)

Transfer rate: 16260.41 [Kbytes/sec] received

Percentage of the requests served within a certain time (ms)

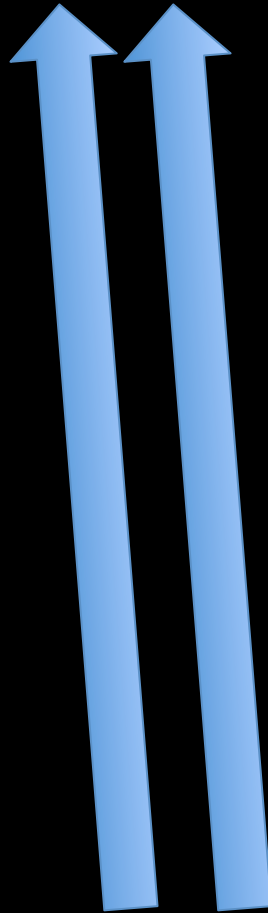
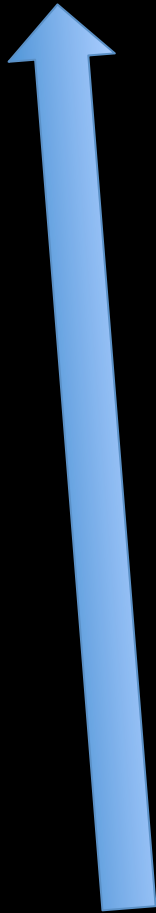
50% 10

...

100% 60 (longest request)

```
$ vmstat 5
```

```
procs -----memory----- ---swap-- -----io----- -system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo    in    cs  us  sy  id  wa
 0  0   55152  53212  15128  69172    0    0     0     1     0     1  0  0  100  0
 1  0   55152  54428  15132  69172    0    0     0    20    56    48  0  0  100  0
```



Killing Apache Modules

```
$ a2dismod
```

```
Your choices are: alias auth_basic  
authn_file authz_default authz_groupfile  
authz_host authz_user autoindex cgi deflate  
dir env expires mime negotiation php5  
rewrite setenvif status
```

Tuning MaxClients

“You can, and should, control the MaxClients setting so that your server does not spawn so many children it starts swapping.”

- Apache.org

Usually in httpd.conf or apache2.conf

Find average size of http process with `top` (x).
How much RAM to dedicate to Apache? (n).

$\text{MaxClients} = \text{floor}(n / x)$

Play around with that and Apache Bench `ab`

Adding Expires Headers

```
<IfModule mod_expires.c>
ExpiresActive on
ExpiresByType image/gif "access plus X days"
ExpiresByType image/jpeg "access plus X days"
ExpiresByType image/png "access plus X days"
ExpiresByType text/css "access plus X days"
ExpiresByType text/js "access plus X days"
ExpiresByType text/javascript "access plus X days"
ExpiresByType application/javascript "access plus X
days"
ExpiresByType application/x-javascript "access plus
X days"
ExpiresDefault "access plus X days"
</IfModule>
```

mod_deflate

- Compress your stuff, comes with Apache

```
$ a2enmod deflate
# ... configure in global or vhost config ...
AddOutputFilterByType DEFLATE text/html text/plain text/xml
# ... restart Apache
$ apache2ctl -k restart
```

Killing ETags

```
$ a2enmod header
```

```
# open apache.conf or httpd.conf, add:
```

```
Header unset ETag
```

```
FileETag None
```

```
$ apache2ctl -k restart
```

HTTP Accelerators

- Serving static content is VERY expensive for Apache
- Varnish, Squid

\$ apt-get install varnish

go to varnish-cache.org

PHP

- APC
 - Caches PHP Op-code, can use as cache
 - Faster than memcache for one box
- PHP 5.3
- mysqlnd
- Output buffering tuning, send data to client more frequently

APC Install

```
$ apt-get install php-pear
```

```
...
```

```
$ pecl install apc
```

Then take a look at cache size: `apc.shm_size`

Framework (CodeIgniter)

- Page Caching
 - Personalized info, uh oh.
- Query Result Caching
- Configuration
 - Disable verbose logging
 - Output compression
 - config/database.php: autoinit?
- Minifying, combining assets
 - ‘assets’ Spark

Application

- Most optimization does not occur within code
- Use built-in PHP functions
- Use associative arrays (hashes, dictionaries)
- Batch DB Inserts, Selects
- Don't use references unless it makes sense
 - Like... `$CI = & get_instance();`
- `require_once` is expensive
- `count()` elements before start 'for' loop

Application (cont'd)

- Don't copy variables if you don't have to

```
foreach($db_result as $row) {  
    $content = $row['content'];  
    $content = word_limited($content, 100);  
}
```

And the usual: `require_once`, etc.

Find one or two numbers in an array that add to 10.

The gross way — n^2

```
$set = array(1, 6, 7, 2, 5, 0, 8);
```

```
foreach($set as $a) {  
    foreach($set as $b) {  
        if($a + $b == 10)  
            exit("Found $a and $b");  
    }  
}
```

Find one or two numbers in an array that add to 10.

The real way — n

```
$set = array(1, 6, 7, 2, 5, 0, 8);
```

```
$flipped = array_flip($a);
```

```
foreach($set as $el) {  
    $needed = 10 - $el;  
    if(array_key_exists($needed, $flipped))  
        exit("Found $el and $needed");  
}
```

When You Get Big

- More machines! with load balancing
- memcached
- Master-slave db setup
 - Writes to master, reads to slave

Takeaways

- GetSpark's APC-based Output override
- These slides

codefury.net/cicon2011

Kenny Katzgrau

@_kennyk_