

Help! I need more servers! What do I do?

Scaling a PHP application



Introduction

- A real world example
 - The wonderful world of startups
- Who am I?



Presentation Overview

- Scalability
- Network infrastructure and components
- Keeping server farms in sync
- Static content delivery
- Of databases and replications...
- Caching and code acceleration
- Code planning, design and implementation techniques



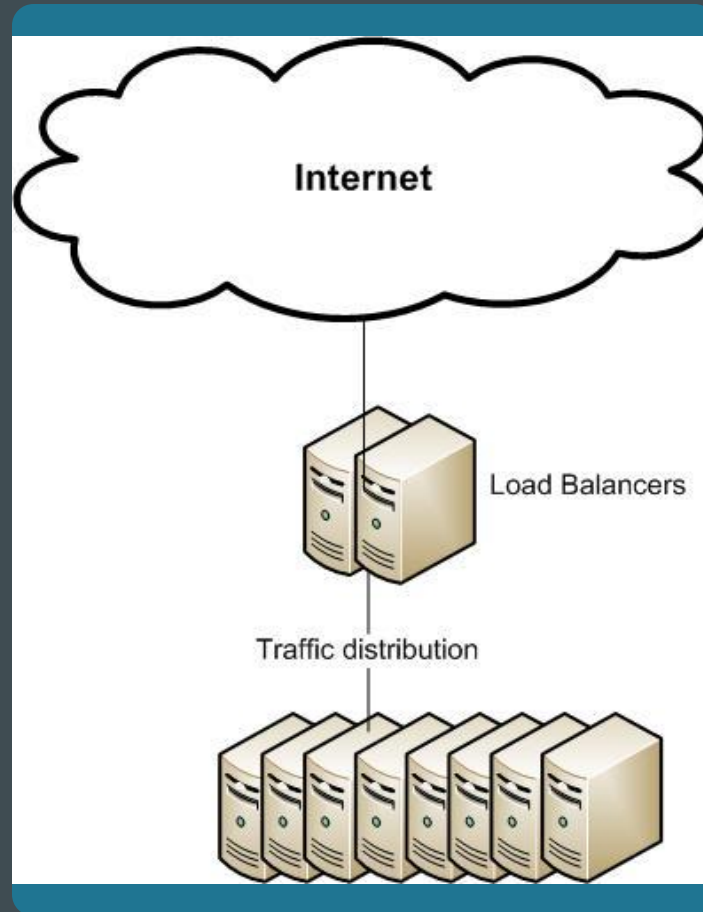
Scalability

- Performance v/s high availability
- No right or wrong technique
- Having the right tools for the job



Load Balancers

- In the network



Load Balancers

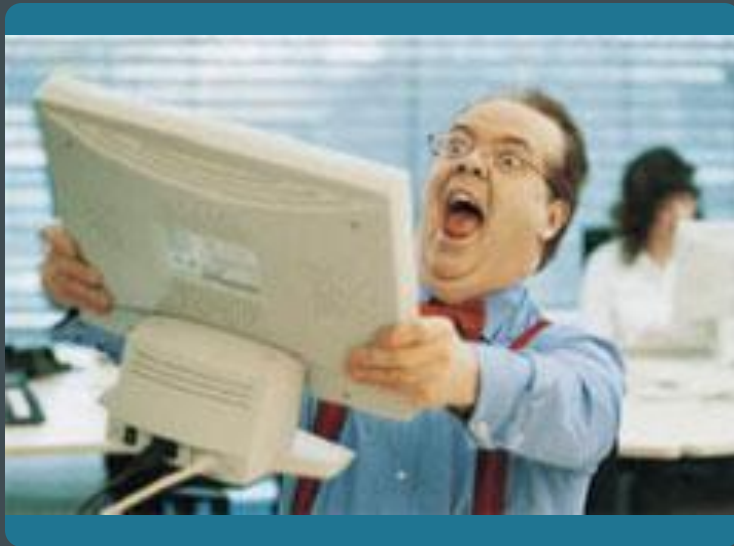
- Software v/s hardware load balancers

	Pros	Cons
Software load balancer (eg: UltraMonkey)	Easy setup Flexible (more configuration options) Cheaper	Most packages cannot handle large or complex networks Large amount of hardware
Hardware load balancer	More robust Processes traffic at hardware level Tech agnostic: works with any OS	More expensive



Session Persistence

- I'm losing my sessions!



Session Persistence

- Sticky sessions
- Session sharing
 - Database
 - Session server
 - Sharedance
- Session clustering in Zend Platform



Session Persistence

- **Overriding php's default session handler**

```
<?php
function session_handler_open($save_path, $session_name)
{
...
}
function session_handler_close() {
...
}
function session_handler_store($key, $data) {
...
}
function session_handler_fetch($key) {
...
}
function session_handler_delete($key) {
...
}
function session_handler_gc($timeout) {

}
session_set_save_handler('session_handler_open', 'session_handler_close',
                        'session_handler_fetch', 'session_handler_store',
                        'session_handler_delete', 'session_handler_gc');
?>
```



Server farms: Keeping servers in sync

- Code base

- Vendor tools

- PHP script using rsync

```
#!/usr/local/php5/bin/php
<?php
$servers = array(
'10.1.42.11', //web2
'10.1.42.12', //web3
'10.1.42.13', //web4
'10.1.42.14', //web5
'10.1.42.15', //web6
'10.1.42.16', //web7
'10.1.42.17', //web8
);

foreach($servers as $server) {
//rsync /www/doc_root
$command = "rsync -avz /www/doc_root/ $server:/www/doc_root";
echo "Syncing /www/doc_root on $server...\n";
echo "-----\n";
echo shell_exec($command) . "\n";
?>
```



Server farms: Keeping servers in sync

- Server configurations
 - Need to restart servers after config changes
 - Apache configs
 - PHP configs
 - Zend platform synchronises php.ini files across servers and restarts them



Static content separation

- Why separate static content?
 - Memory consumption
 - Blocking Apache processes for large files
- How to separate static content?
 - Types of static content: js, css, images, videos, audio, html
 - Changing application's urls for static content
 - Using a low memory footprint server to serve static content (lighttpd, thttpd...)
- Zend platform's download server
- Content delivery network (CDN)



Static content separation

Typical prefork MPM Apache config

```
<IfModule mpm_prefork_module>  
  ServerLimit 1024  
  StartServers 128  
  MinSpareServers 256  
  MaxSpareServers 512  
  MaxClients 1024  
  MaxRequestsPerChild 2000  
</IfModule>
```



Static content separation

- Comparing the different content separation ways

	Pros	Cons
Zend Download server	Easy integration	Still the same server
Static content server	Total content separation Optimal resources utilization	Harder to integrate Extra set of servers to maintain
Content Delivery Network	Speed!	Harder to integrate (change all links) Cost



Lighttpd benchmark

Concurrency Level: 10,000
Time taken for tests: 5.420 seconds
Complete requests: 100,000
Failed requests: 0
Write errors: 0
Keep-Alive requests: 0
Total transferred: 419,968,990 bytes
HTML transferred: 388,510,351 bytes
Requests per second: 18,449.53 [#/sec] (mean)
Time per request: 542.019 [ms] (mean)
Time per request: 0.054 [ms] (mean, across all concurrent requests)
Transfer rate: 75,666.30 [Kbytes/sec] received



Application database infrastructure

- High performance and high availability for MySQL
 - Master-slave replication
 - Dual master replication
 - DRBD + Heartbeat
 - MySQL Cluster
- Vertical and horizontal partitioning
- Shards



Application database infrastructure

- How does all this affect your code?
 - Some technologies implement scalability transparently
 - Replication, to take advantage of it for performance, requires separation of read/write queries
 - One way to do that would be to implement a wrapper with methods: `mysql_safe_reader()`, `mysql_safe_writer()`
 - MySQL proxy



Application database infrastructure

- Some tools to optimize MySQL's performance
 - General query log with `mysqlsla`
 - Mysql slow query log (`log_slow_queries`)
 - Monyog



Application database infrastructure

- Comparison of MySQL scalability options

	Pros	Cons
Master-slave replication	Easy setup	No automatic failover Need to recode application
Dual Master replication	No app changes required High throughput	Need a load balancer If data falls out of sync, recovery very hard
DRBD	No app changes required Fast failover	Two servers max Always one inactive server
MySQL cluster	Extremely high throughput True redundancy No app changes required	Complex setup Big amounts of RAM required on servers Lots of servers required



Application database infrastructure

- More details in MySQL manual, chapter 14, high availability and scalability



Code acceleration

- How code accelerators work
- Xcache, APC, eaccelerator
- Zend platform Code accelerator



Code acceleration

- How code accelerators work
- Xcache, APC, eaccelerator
- Zend platform Code accelerator



Caching

- Database requests caching
 - Usually the primary bottleneck
 - Limit the number of requests to the db
 - Use memcached for frequently accessed data that does not change often
- HTML caching
 - At application level, generate HTML pages that do not change often, and serve them as HTML
 - Zend Platform caching



Caching

- Database requests caching
 - Usually the primary bottleneck
 - Limit the number of requests to the db
 - Use memcached for frequently accessed data that does not change often



Caching

- Memcache example

```
$memcache = new Memcache();  
$memcache->connect('yourserverhost', 11211) or die("Could not connect to memcache server");
```

```
function getTopFiles() {  
    global $memcache;
```

```
    $query = "SELECT TOP 100 id, filename, rank FROM files ORDER BY rank DESC";  
    $queryMd5 = md5($query);
```

```
    //Connect to memcache server
```

```
    $results = $memcache->get($queryMd5);
```

```
    if(!$results) {
```

```
        $results = mysql_safe_reader($query);
```

```
        //store for a day. 0: do not compress
```

```
        $memcache->set($queryMd5, $results, 0, 86400);
```

```
    }
```

```
    return $results;
```

```
}
```



Caching

- HTML caching
 - At application level, generate HTML pages that do not change often, and serve them as HTML
- Zend Platform caching



Caching

- Zend Platform caching screenshot

The screenshot displays the 'Performance' configuration page in the Zend Platform interface. The page is divided into several sections for different settings. The 'Dynamic Content Caching' section is currently expanded, showing a comparison between 'Current Settings' and 'New Settings'. The 'File Compression' section is also visible below it.

Setting	Current Value	New Value
Accelerator Memory	64 MB	64 MB
Memory Reclaim Threshold	5%	5%
Maximum Accelerated Files	2000	2000
Extensions For PHP Files	php	php
Dynamic Content Caching		
Dynamic Caching Enabled		
Dynamic Caching Enabled	On	<input checked="" type="radio"/> On <input type="radio"/> Off
Maximum Cache Size	Unlimited	0 MB
Minimum Free Diskspace	428 MB	428 MB
Maximum Cached File Size	500 KB	500 KB
Default Cache Lifetime	360 Seconds	360 Seconds
Default Dynamic Caching Conditions	ALLGET	Change Default Conditions
File Compression		
Compress Files	Only cached files	<input type="radio"/> None <input checked="" type="radio"/> Only cached files <input type="radio"/> All files



Caching

- Client side caching
 - Expired header
 - Etag



PHP Code design considerations

- Zend Framework

- What is Zend Framework?
- How can it help me in my development?



PHP Code design considerations

- How is the application split?
 - Modular design advantages
 - Segmentation across multiple servers
- Shared storage
 - Data centralization



Conclusion

- Plan, plan then plan some more
- Ensure compatibility



Thank you!

Maurice Kherlakian

Email: maurice.kherlakian@bell.ca

Skype: mkherlakian

