Scaling PHP

… in an “Enterprise” environment

(what’s that?)

Sarel van der Walt

- I’m from the Internet … a.k.a. afrihost.com

 facebook.com/sarelvdwalt

 twitter.com/sfvdwalt

“Scalability is about the **entire architecture**,
not some minor code optimisations”

– Dustin Whittle

Is PHP really the problem?

- Have you looked at your **DB**?
- Have you looked at your **network**?
- Have you looked at your **assets**?

That's but to name a few!

Vertical = expensive

versus…

Horizontal = inexpensive,
but complex

Why Afrihost cared (suddenly!)

From this…

Right now

60

active users on site

…to this!

Right now

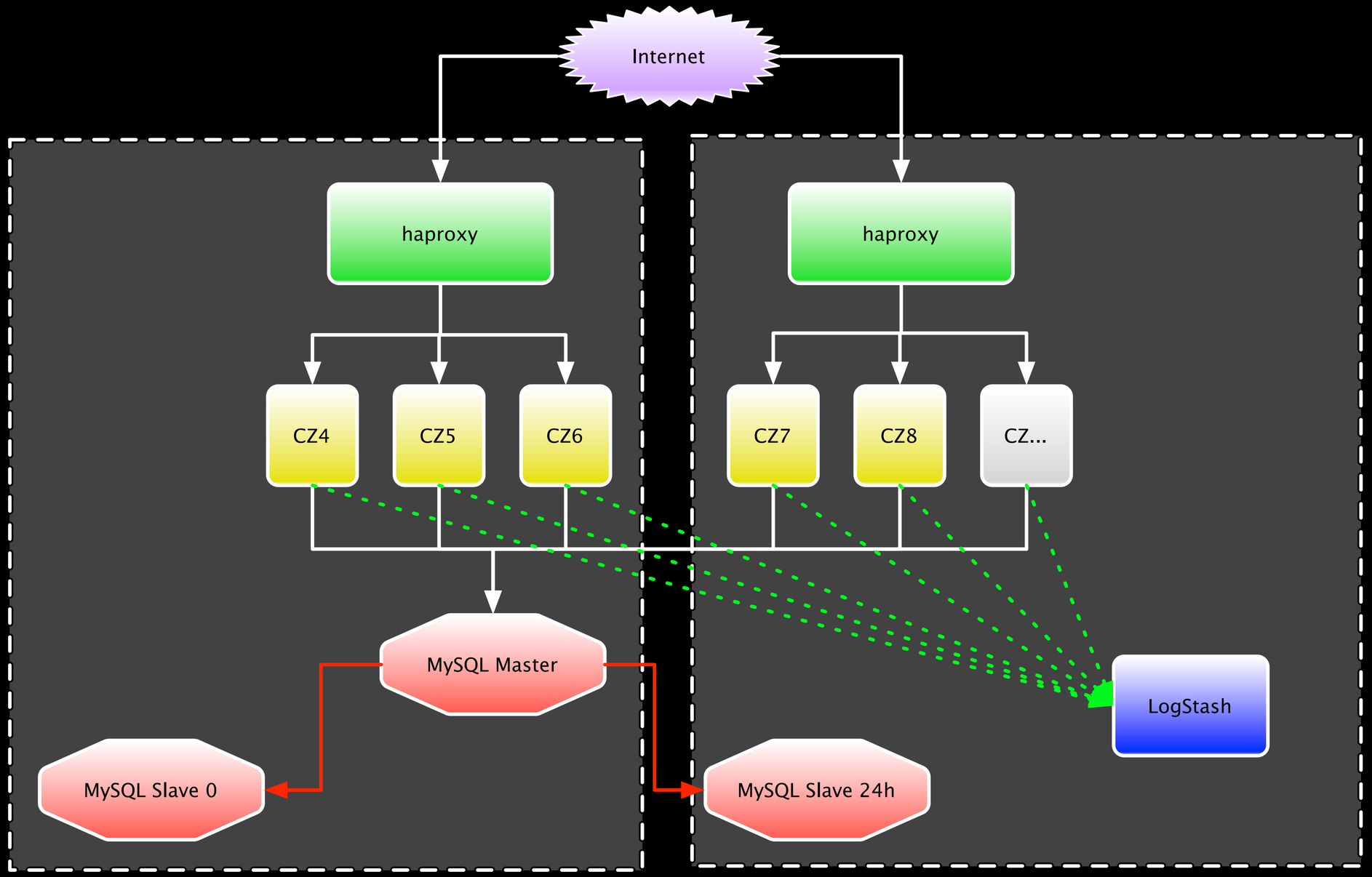
8114

active visitors on site

RETURNING NEW

71% 29%

Our architecture, more/less (actually very high level)



$_SESSION

- File system is **bad** :: you can only use ONE server!
- Database (MySQL, Postgres) is **ok** :: vertical scaling needed

$_SESSION

- IN-Memory Cache is **better -** think memcached / redis
- You could also put it in **mongodb**, or **casandra**…
  …session data nicely lend itself to serialisation
- Browser based is the **BEST** - but be careful of the 4k limit of older browsers (and security)

IN-Memory Cache

- File-System Cache - old, but trusted - vertical
- memcached is **better** :: multiple servers through sharding

```php
<?php
$m = new Memcached();
$m->addServer('localhost', 11211);

$m->set('foo', 100);
var_dump($m->get('foo'));
```

IN-Memory Cache *[continued]*

- redis is **awesome** (but more complicated) :: also has sharding
  / redis can do so much more */*

```php
<?php
$redis = new PredisClient(array(
            "host" => "127.0.0.1",
            "port" => 6379)
);

$redis->set("My Cool Key !!", "Say something awesome!");
$value = $redis->get("My Cool Key !!"); // flexible keys!
var_dump($value);
```

… more redis (I love redis!)

```php
// counters - and they don't mess with your head!
$redis->incr("page_views_123"); // increments by 1
$redis->incr("page_views_123", 5); // by 5
$redis->decr("page_views_123"); // you can even decrement??
$redis->decr("article_views_237", 5); // decr by 5
```

#atomic

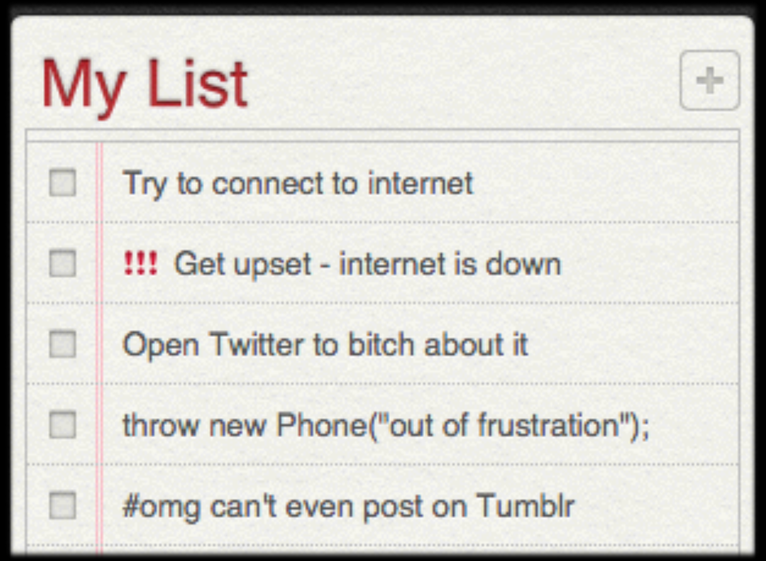
#fancy_replacing_your_auto_increments_with_this??

. . . more redis (I love redis!)

```php
// it has hashes, great for storing session data (or cats)!!
$redis->hset("my_cat", "color", "like a cow, with spots and stuff");
$redis->hset("my_cat", "sex", "male");
$redis->hset("my_cat", "temper", "extreme");
$redis->hset("my_cat", "primary aim in life", "sleeping");
```

. . . more redis (I love redis!)

```
// really great is lists...
$redis->rpush("mylist", "bravo"); // creates, and puts "bravo" at the end
$redis->rpush("mylist", "charlie"); // puts "charlie" after "bravo" (right)
$redis->lpush("mylist", "alpha"); // puts "alpha" in front of list
```



#wondering_about_message_queues

… more redis (I love redis!)

```php
// expiring stuff...
$redis->set("session_d41d8cd98f", json_encode($_SESSION));
$redis->expireat("session_d41d8cd98f", 3600); // expires in 1 hour
$redis->expireat("session_d41d8cd98f", strtotime('2014-12-31 23:59:59')); // exact
echo $redis->ttl("session_d41d8cd98f"); // gets time left
$redis->persist("session_d41d8cd98f"); // removes all expiry, VIVA session_d41d8cd98f
```

Background Work (non-blocking)

- 3rd Party Calls (tweeting, api-calls, emailing, etc)
- Expensive DB (or any other) work - especially unimportant ones
- **Q:** What's "expensive" ??
  A: 150+ milliseconds
- Ask yourself: **Is it important for the next request?**
- **Q:** So what do I do?
  A: Queue it!!
  Q: How?
  A: Why with REDIS, off course?! :)
  … or something else… like MySQL… or not?

During billing run (25th to 2nd) Afrihost runs…

550 000 background CLI's - broken up over 253 scripts

Some of these actions, are called...

75k times

during a month, consuming...

9.7 days of server time !!!

> composer.phar install chrisboulton/php-resque

```php
<?php
Resque::setBackend('localhost:6379');

while (true) {
    $args = array(
        'name' => 'Sarel-' . rand(0, 1000000)
    );

    Resque::enqueue('my-cool-queue', 'My_Job', $args);
}
```

40million records added in 42 minutes

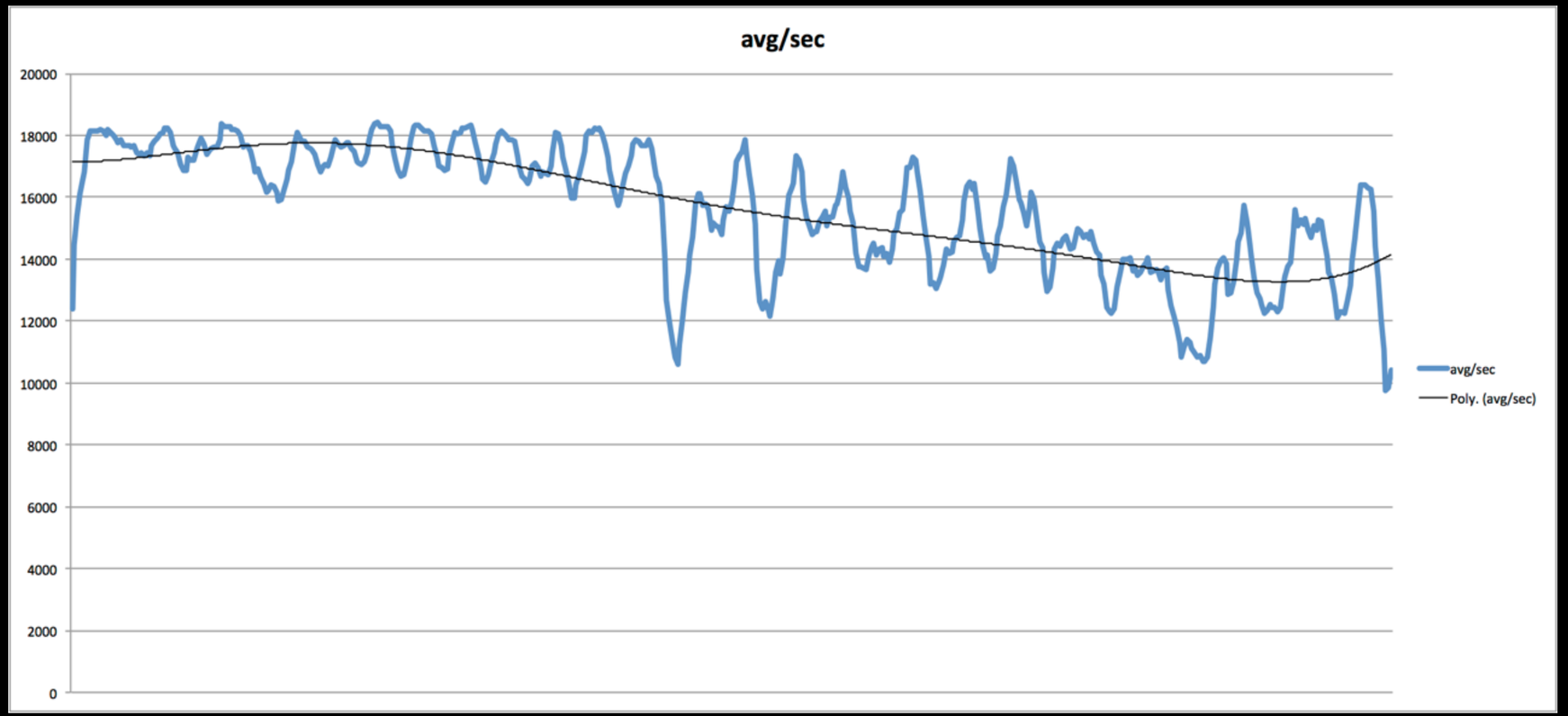
... that's **954k** in a minute...

... that's **15906** in a second!! *(MySQL took **34 seconds** to do this)*

... that's **35 seconds** (rounded up, lol) to insert the entire billing-run to Afrihost Job Queue!!

MySQL didn't stand a chance!

(1177 seconds = 19.6 minutes!!)



HTTP Caching

- Are you caching **assets** on the browsers?
- Should they really download new **CSS** that often?
- How do you **cache-bust**?
- **eTag** much? No, it's not this…

Reverse Proxy

- varnish - varnish-cache.org
- nginx - nginx.org
- haproxy - haproxy.org
- squid - squid-cache.org
- PS: Most Frameworks has reverse proxy built in!!

Reverse Proxy - haproxy

srv_http

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
cz4	0	0	-	0	0		0	0	8	0	0	?	0	0		0		0	0	0	0	5d3h UP	L6OK in 4ms	1	-	Y	242	11	1m22s	-
cz5	0	0	-	0	0		0	0	8	0	0	?	0	0		0		0	0	0	0	10d3h UP	L6OK in 3ms	1	-	Y	184	4	43s	-
cz6	0	0	-	0	0		0	0	8	0	0	?	0	0		0		0	0	0	0	20d4h UP	L6OK in 3ms	1	-	Y	140	0	0s	-
cz7	0	0	-	4	30		0	8	8	884 142	882 900	0s	847 115 120	8 979 609 507		0		0	501	0	0	20d4h UP	L4OK in 0ms	1	Y	-	0	0	0s	-
cz8	0	0	-	3	34		0	8	8	884 372	882 934	0s	842 542 614	8 809 497 249		0		0	528	0	0	16d17h UP	L4OK in 0ms	1	Y	-	36	2	7m43s	-
Backend	0	107		7	54		0	123	200	1 768 514	1 765 834	0s	1 689 657 734	17 789 106 756	0	0		0	1 029	0	0	20d4h UP		2	2	3		0	0s	

srv_http

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
cz4	0	0	-	0	0		0	0	8	0	0	?	0	0		0		0	0	0	0	5d3h UP	L6OK in 7ms	1	-	Y	242	11	1m22s	-
cz5	0	0	-	0	0		0	0	8	0	0	?	0	0		0		0	0	0	0	10d3h UP	L6OK in 5ms	1	-	Y	184	4	43s	-
cz6	0	0	-	0	0		0	0	8	0	0	?	0	0		0		0	0	0	0	20d4h UP	L6OK in 3ms	1	-	Y	140	0	0s	-
cz7	0	0	-	6	30		0	8	8	884 579	883 337	0s	847 528 167	8 983 160 295		0		0	502	0	18	20d4h UP 1/3 ↓	L4CON in 0ms	1	Y	-	2	0	0s	-
cz8	0	0	-	6	34		1	8	8	884 810	883 372	0s	843 642 297	8 813 448 469		0		0	528	0	0	16d17h UP	L4OK in 0ms	1	Y	-	36	2	7m43s	-
Backend	0	107		6	54		1	123	200	1 769 371	1 766 709	0s	1 691 170 464	17 796 608 764	0	0		0	1 030	0	18	20d4h UP		2	2	3		0	0s	

srv_http

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
cz4	0	0	-	0	0		0	0	8	0	0	?	0	0		0		0	0	0	0	5d3h UP	L6OK in 4ms	1	-	Y	242	11	1m22s	-
cz5	0	0	-	0	0		0	0	8	0	0	?	0	0		0		0	0	0	0	10d3h UP	L6OK in 3ms	1	-	Y	184	4	43s	-
cz6	0	0	-	0	0		0	0	8	0	0	?	0	0		0		0	0	0	0	20d4h UP	L6OK in 7ms	1	-	Y	140	0	0s	-
cz7	0	0	-	0	30		0	8	8	884 584	883 342	12s	847 528 167	8 983 160 295		0		0	502	0	23	12s DOWN	L4CON in 0ms	1	Y	-	3	1	12s	-
cz8	0	0	-	2	34		2	8	8	884 854	883 416	1s	843 684 490	8 813 770 301		0		0	528	0	0	16d17h UP	L4OK in 0ms	1	Y	-	36	2	7m43s	-
Backend	0	107		2	54		2	123	200	1 769 415	1 766 758	1s	1 691 212 657	17 796 930 596	0	0		0	1 030	0	23	20d4h UP		1	1	3		0	0s	

Database Optimisation

- SQL **Slow Query** Log - look at it regularly!
- Add slaves for **read-scaling**
- Side-line: **Disaster Recovery** :: Slave **0**-seconds + Slave **24**-hour
- Archive!! - makes reading AND writing faster
- Offload!! - perhaps relational DB isn't the right place for data?

Database Optimisation [continued…]

id	username	in_bytes	out_bytes	created_at
123	sarel@afrihost.co.za	1024	2048	2014-07-15 10:01:15
124	dude@afrihost.co.za	971826	12823	2014-07-15 10:01:23
125	geek@afrihost.co.za	17263712	7126373	2014-07-15 10:01:58
126	27838127372@afrihost	12382	9563	2014-07-15 10:02:07

2009-06 :: 3110 records
2009-10 :: 1.7 million records
2010-10 :: 11.9 million records
2011-10 :: 17.4 million records
2012-10 :: 26.6 million records
2013-10 :: 59.6 million records
2014-06 :: 101.1 million records

Database Optimisation [continued…]

- A good strategy that worked for us was **ztableYYYYMM**
- You could use any strategy you like, the basics of it:
  - **Select** a bunch of rows
  - **Insert** them into the applicable archive table
  - **Delete** original
  - Check out: https://github.com/sarelvdwalt/SFDatabaseArchiveBundle

Framework Optimisation

- Did you just install it and leave? **#default_install**
- Go through your **configs**, switch things on and off for production
- PRO Hint :: Switch debugging off in production
- PRO Hint #2 :: Switch APC / OpCache on INSIDE your Framework (if it has it) for things like Doctrine / ORM

Doing Essential work Up-Front

- Pre-cache data - hey, Redis / Casandra might be a good place to put this?? :)
- Warm up your cache during rollouts
- Careful about rollout cache busting!

Move work to the client

- The math is simple: 100k clients times 4 cores = 400 000 CPU cores! Use them … use them ALL!!!!
- Seriously, **render calculations** belong on the **browser**. CSS3, JavaScript
- Do you have an IF statement in your HTML generation (on server)? Now you have two problems!
- Send data via API, with templates - can utilise CDN for templates - **SOA (Service Oriented Architecture)**
- AngularJS, BackboneJS, check out TodoMVC.com

Learn to debug… LIVE!

- **xdebug** + webgrind - this will hurt, it's **slow** and eats CPU's (and HDD space)
- **xhprof** + xhprofGUI - **fast**, can run on production (best to keep it off until you need it)
- Sometimes **logging** is good enough! Use **LogStash** for history and trend… which brings me to monitoring…

Learn to debug… LIVE!

Have you tried:

```
echo "WTF! Why is this sh!t not working??" . $variable. PHP_EOL;
```

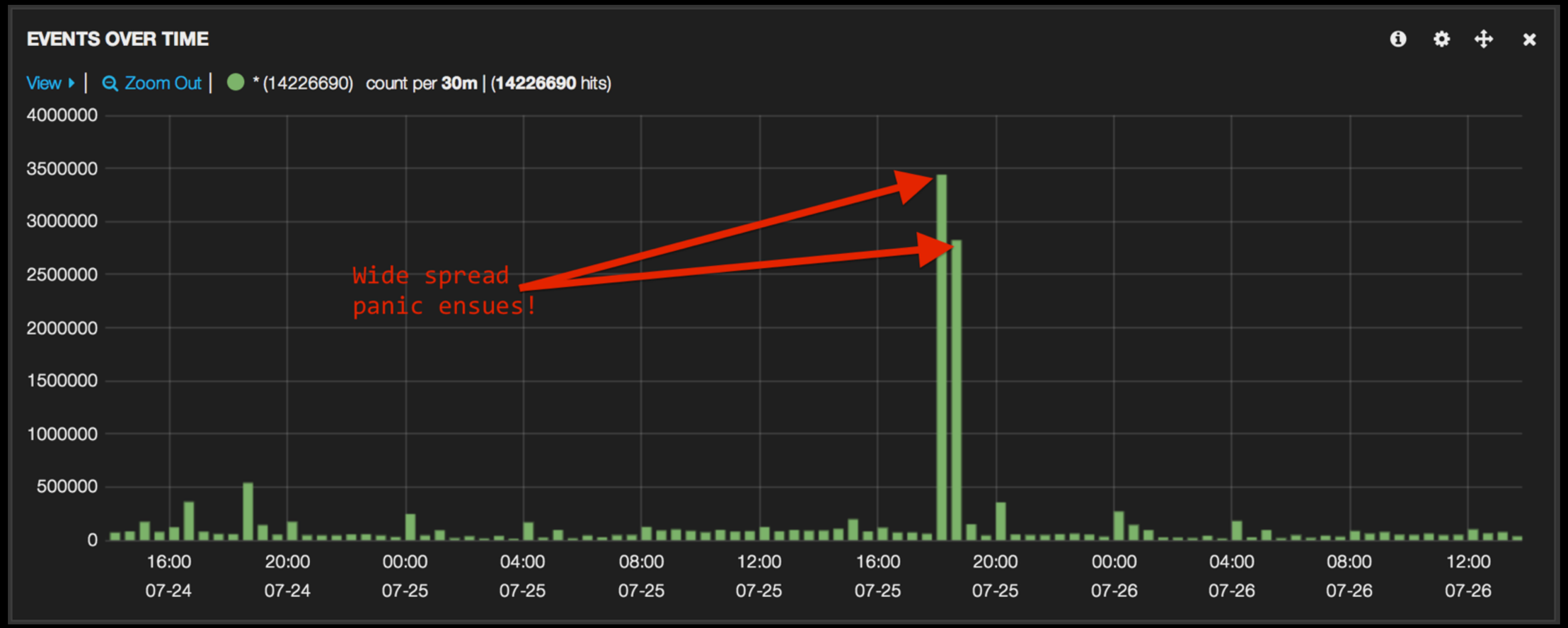
Perhaps better to wrap that…

```
function secretEcho ($v) {
    if ($this->isAdminUserOrOnVPN) {
        echo $v.PHP_EOL;
    }
}

secretEcho("WTF!?" . $variable);
```

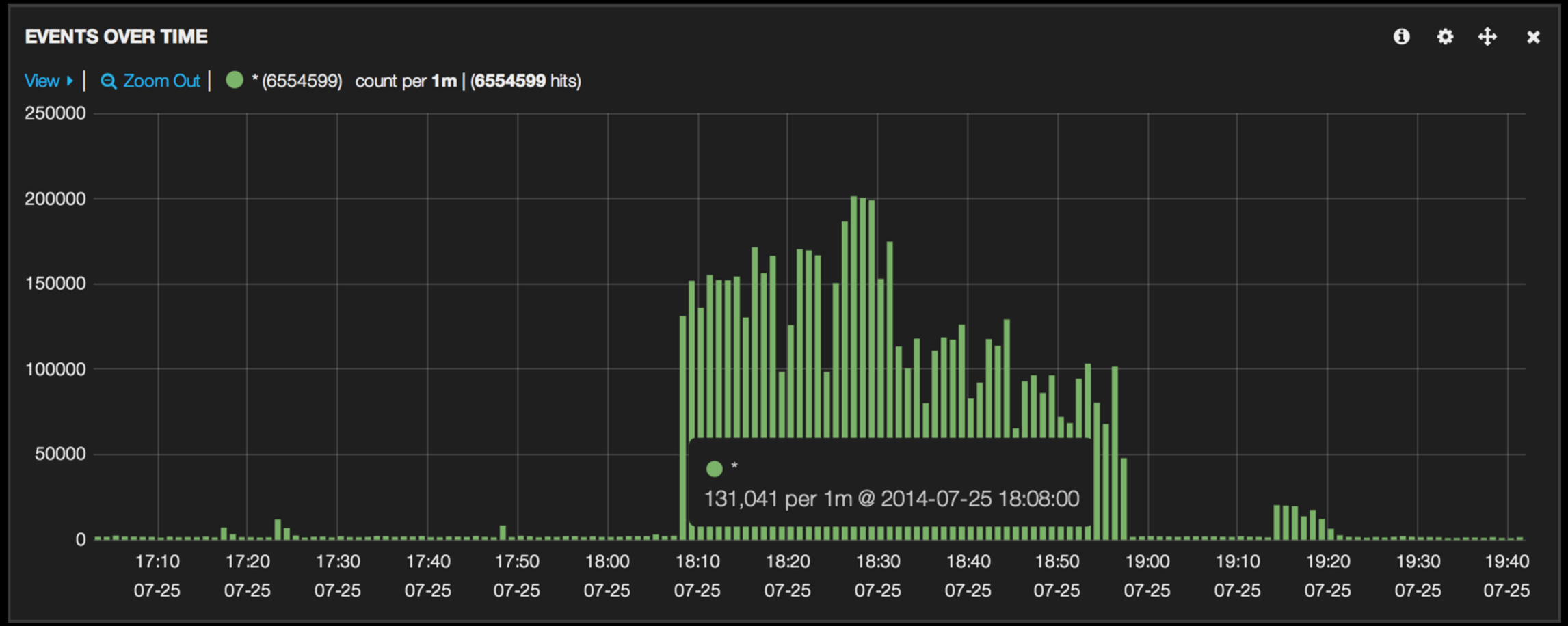
Monitoring!

LogStash - aggregate logs from multiple servers into central location.



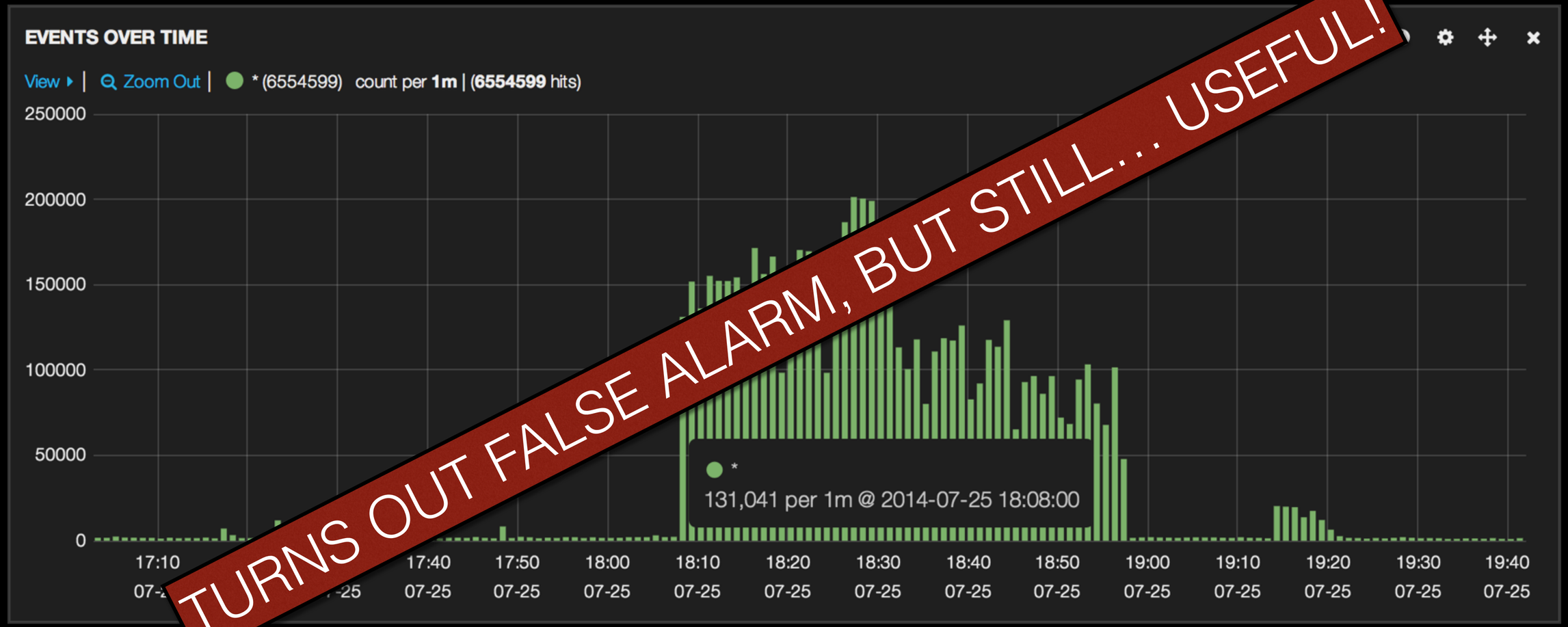
… let's zoom and see WTF…

Monitoring… looking closer…



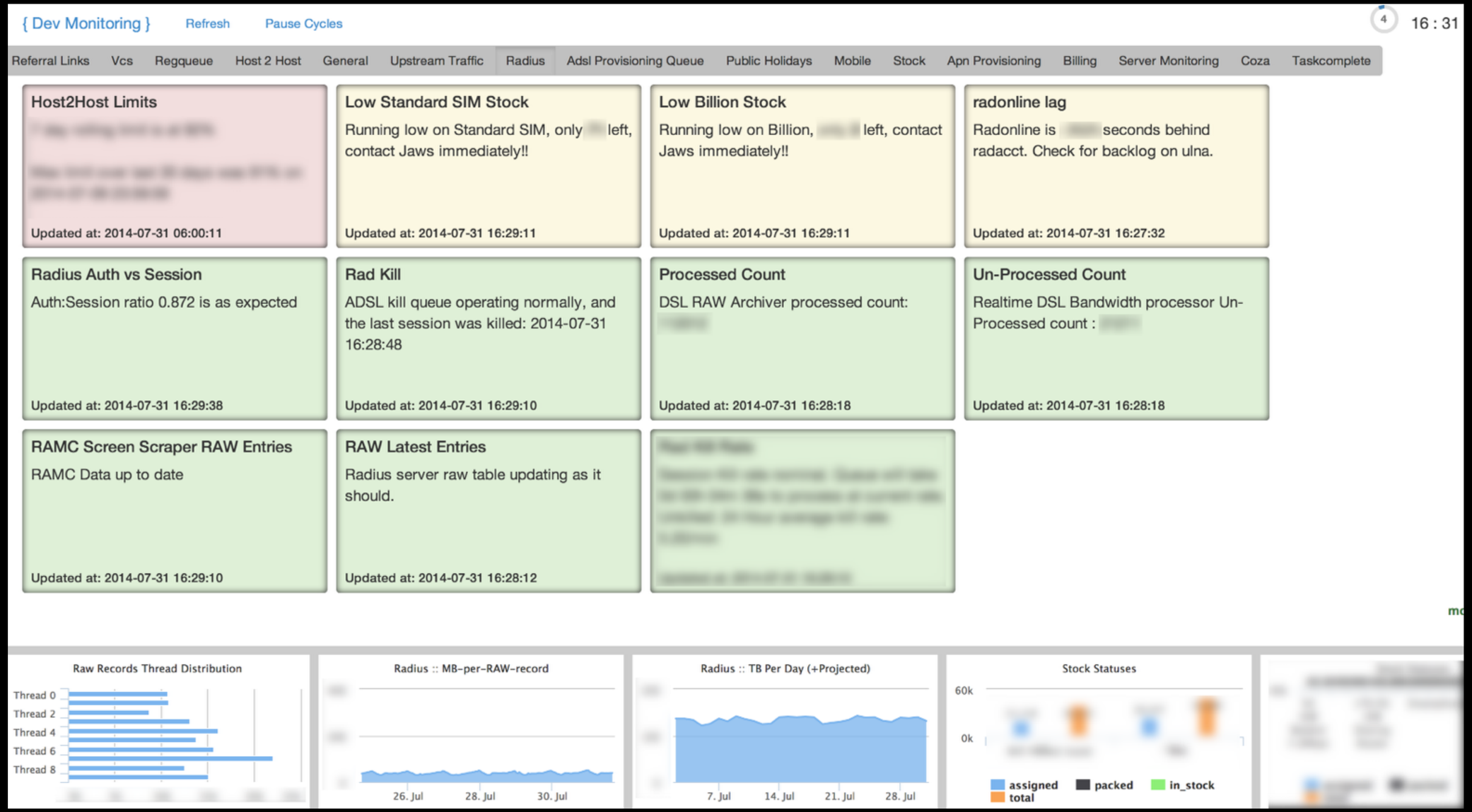
… hmmmm… what started around 18:08 on the 25th?

Monitoring… looking closer…



… hmmmm… what started around 18:08 on the 25th?

Monitoring… **DevOps** Board…



Monitoring… **DevOps** Board…

SMS's (Texts)…

… lots and lots of notifications!

ON-Call Much?

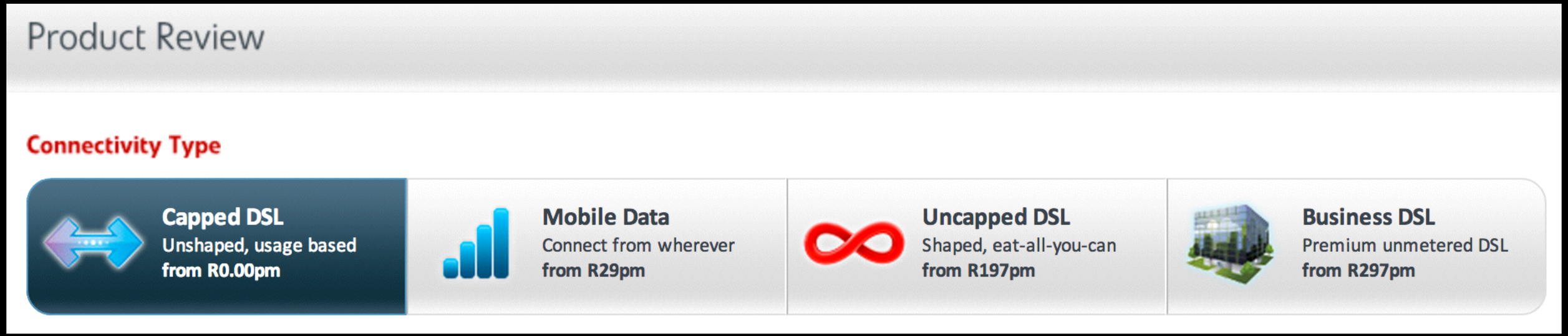
Accessibility

- How close are you to your servers?
- Do you NEED to log in to see **logs**?
- **Catch-22** :: Developers should be able to access anything (or close to) **without having to log into the servers**… but **don’t deny them access.**

Side-line…

The **most junior** developer should **do the rollouts**!

Let’s look at something **practical**…
the Afrihost order form.

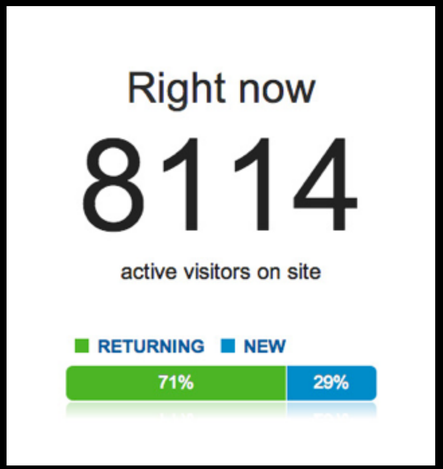


Each click, does:

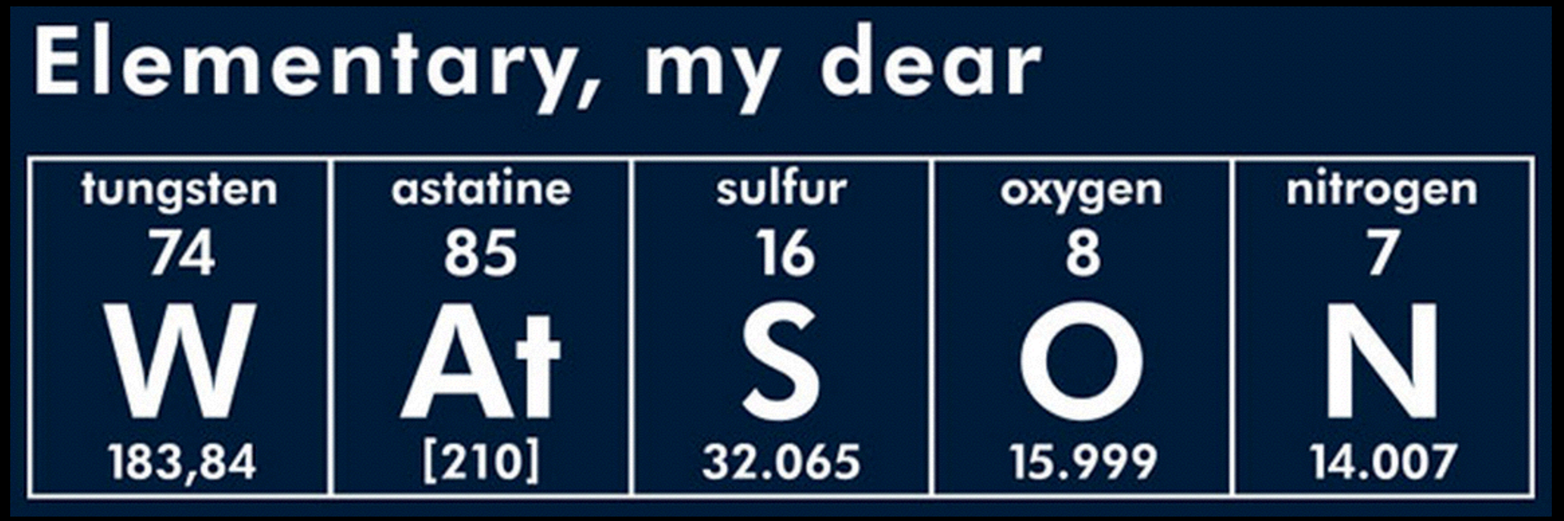
1. HTTP request (hits apache, hits symfony)
2. Asks the DB for the next products in the category
3. Returns the value, and renders it

Let's go look at it live: clientzone.afrihost.com/get-connected

Remember this?



Final thoughts…



… but sometimes it's **not** so elementary.

Thank you.

Charging beer… 36%

github.com/sarelvdwalt/php-jhb