

# Practical Performance Tuning and Testing

Nick Lewis  
Senior Dev, Chapter Three

# I love to talk about myself

- Worked with the Drupal since version 4.4
- Accidental Programmer
- Nick Lewis: The blog
- Former Austinite; Kidnapped by Chapter 3.
- I am not @drupaltruth. I will make no further comments on such baseless rumors

# Performance Tuning...

- Not a science
- Not a strict process
- Easy to pick up
- Hard to master

# Performance tuning is sort of like medicine

- Diagnosis comes before treatment
- Aim to treat the root cause
- Gestaltish
- Run tests to gauge success



# It's also like detective work

- Sometimes, the beginning feels like a murder scene
- Trust and follow vague hunches
- Make note of anything “odd.”



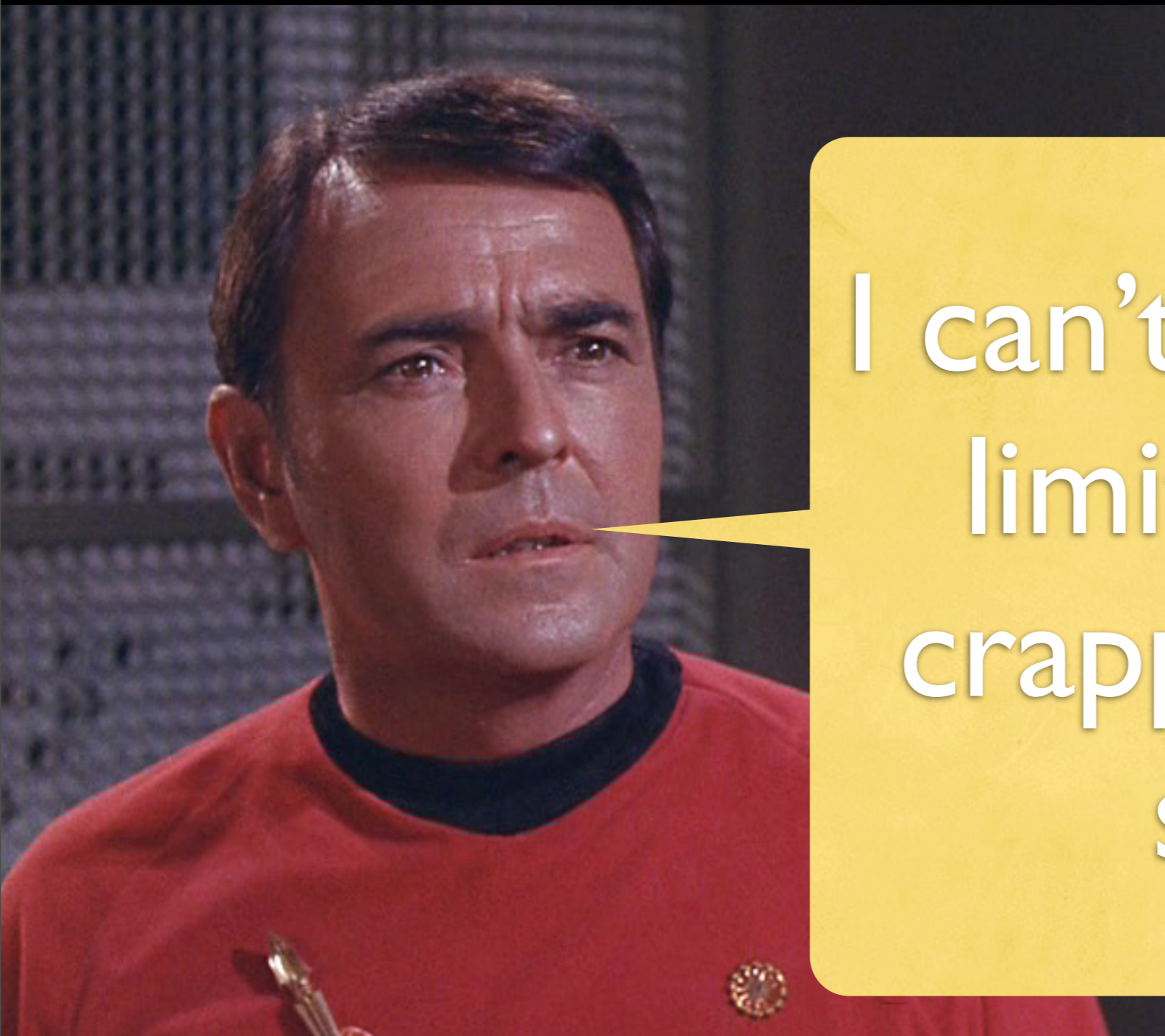
# It's also like being an airline pilot

- Be present at all times.
- In emergencies, its essential you stay calm and focused.
- Failure means the plane crashes into the mount.





And of course, you'll  
feel like him at times



I can't change the  
limits of your  
crappy godaddy  
server.

# The Patient

- Site get's tons of traffic ( > 200,000 an hour during spikes)
- 250,000 nodes
- Server frequently crashes
- Built by an outsourced firm
- Client thinks Drupal is to blame



# The Plot Twist

- Varnish running? Check.
- Memcache running? Check.
- Have they turned on the (insert whatever name you're thinking) cache? Yep.



Switch to nginx, idiot.

MongoDB!

Just add more  
web nodes.



A large herd of wildebeest is crossing a river. The animals are packed closely together, and the water is splashing around them. The scene is captured from an elevated perspective, showing the scale of the migration. The text "FEAR THE STAMPEDE" is overlaid in a black box in the center of the image.

# FEAR THE STAMPEDE

**BAD PERFORMANCE  
IS USUALLY DUE TO  
BAD PRACTICE**

# Turn on the lights

- Create a cache free test environment.
- Seriously, turn query cache, and APC off.
- We'll test the performance of our cache later...

# Test I: MySQL

- Include all public page types, e.g. taxonomy, front, node in a file
- Wimpy siege test: `siege -c 1`
- Observe queries:  
`mysql -u root -proot`  
`show processlist;`
- Parse yer slow slow query log with maatkit  
`~/mk-query-digest /path/to/log/slow.log`

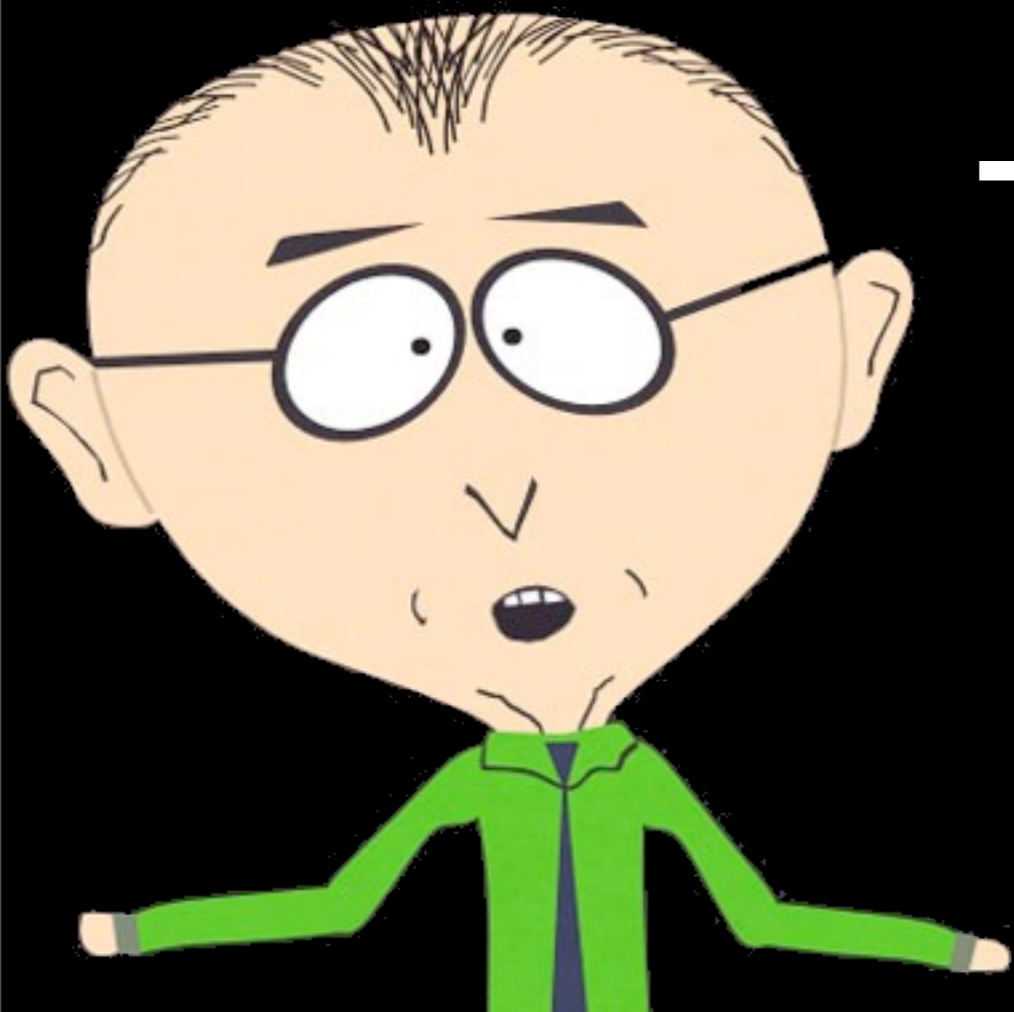




We'll do it live!

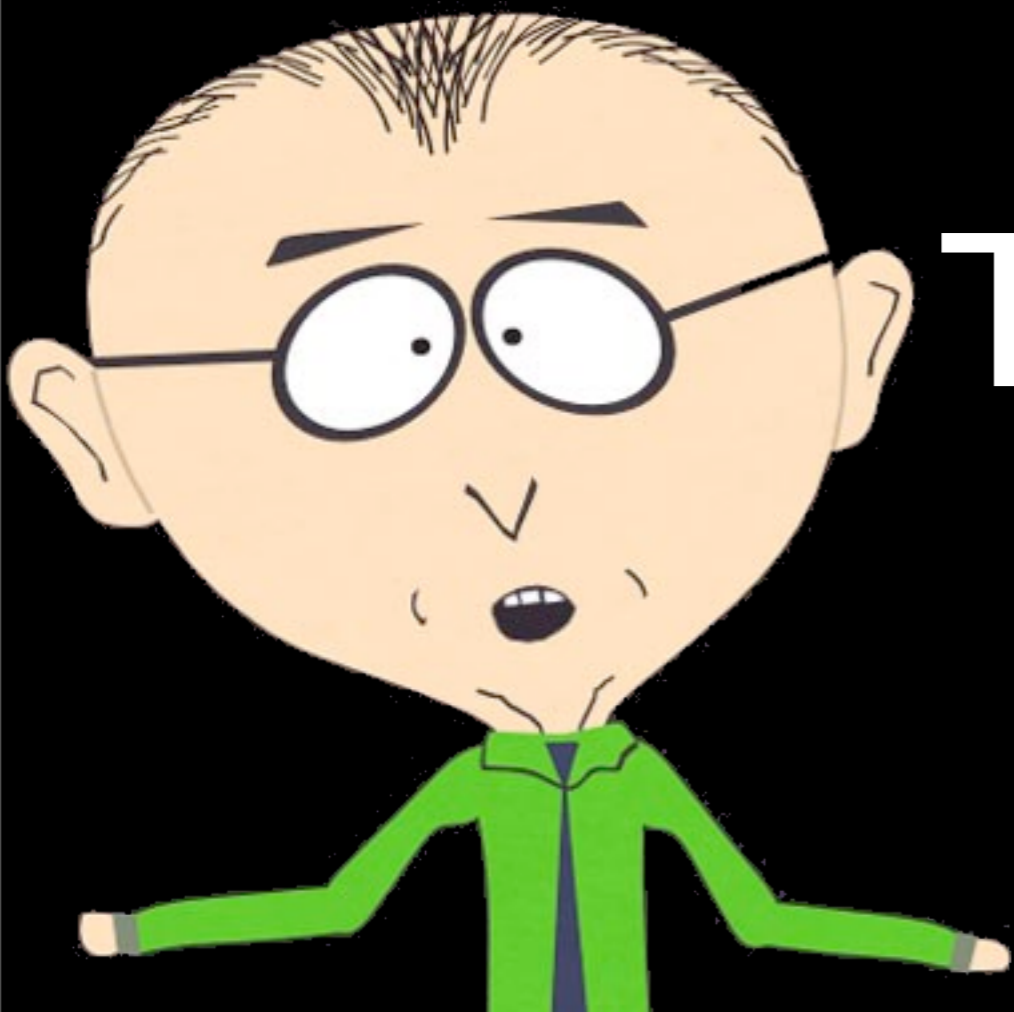
\$ siege -c l

```
SELECT n.title,c.totalcount  
FROM node n  
LEFT JOIN node_counter c ON n.nid = c.nid  
ORDER BY c.totalcount DESC  
LIMIT 0,5
```



That query is bad,  
m'kay

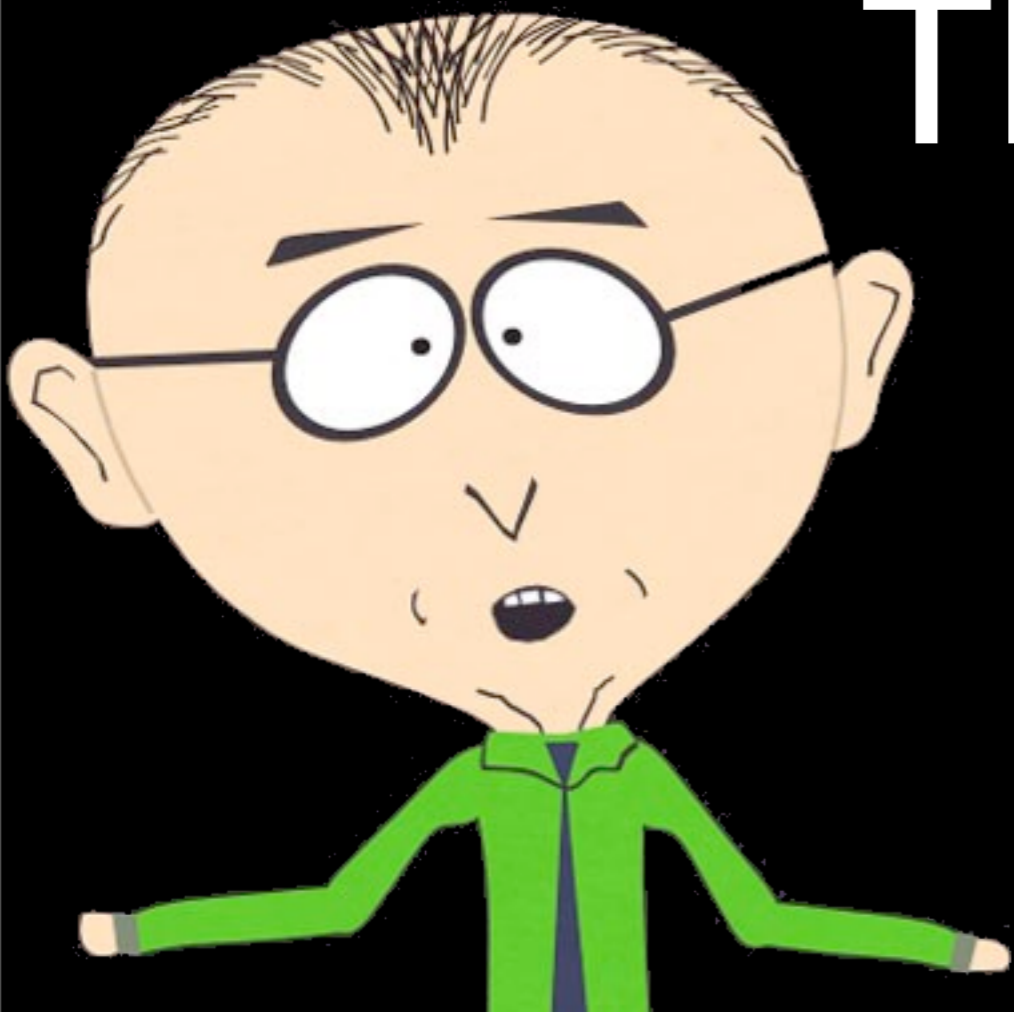
```
SELECT n.title,c.totalcount  
FROM node n  
INNER JOIN node_counter c ON n.nid = c.nid  
ORDER BY totalcount DESC
```



That query is good.

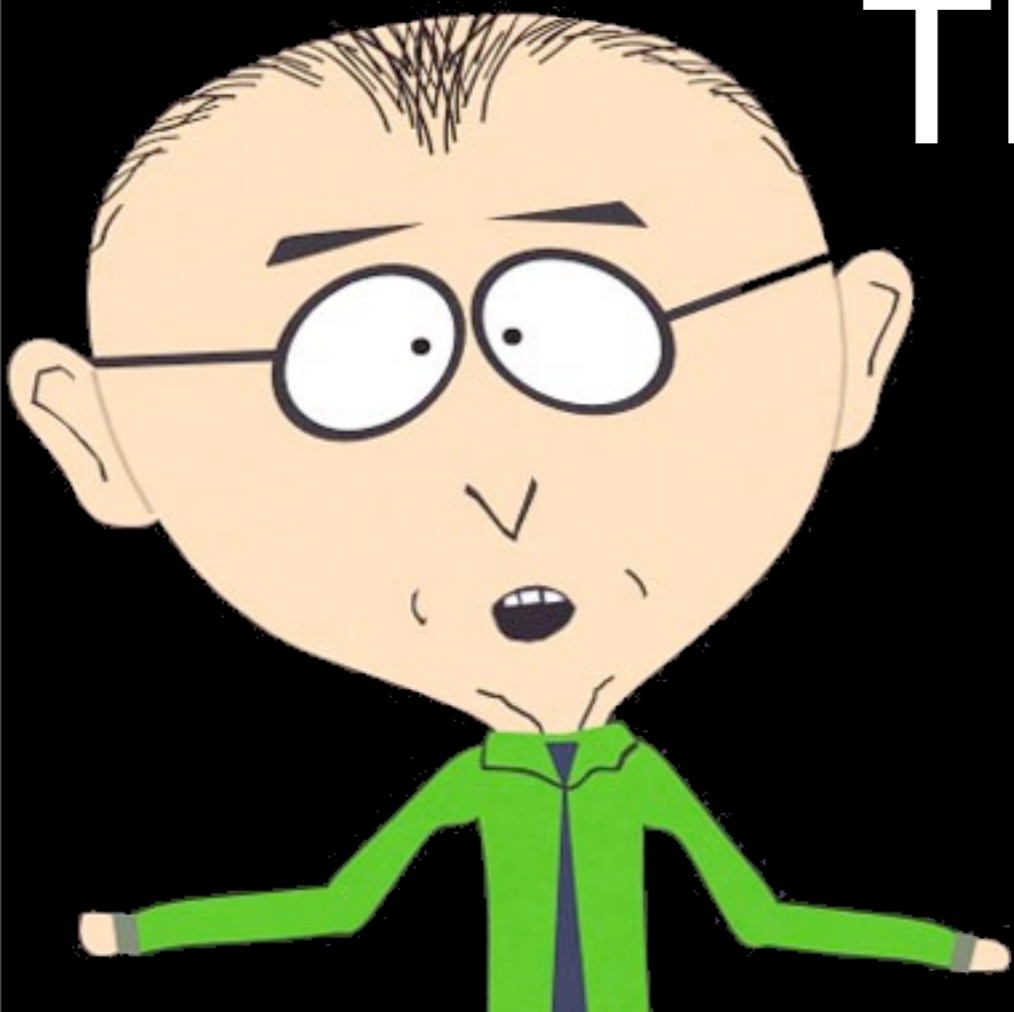
```
SELECT n.title,c.totalcount
FROM node n
LEFT JOIN node_counter c ON n.nid = c.nid
WHERE c.totalcount > 4000
ORDER BY c.totalcount DESC
```

That query isn't bad  
It will work.



```
SELECT n.title,c.totalcount  
FROM node n  
LEFT JOIN node_counter c ON n.nid = c.nid  
WHERE c.totalcount > 4000  
ORDER BY c.totalcount DESC
```

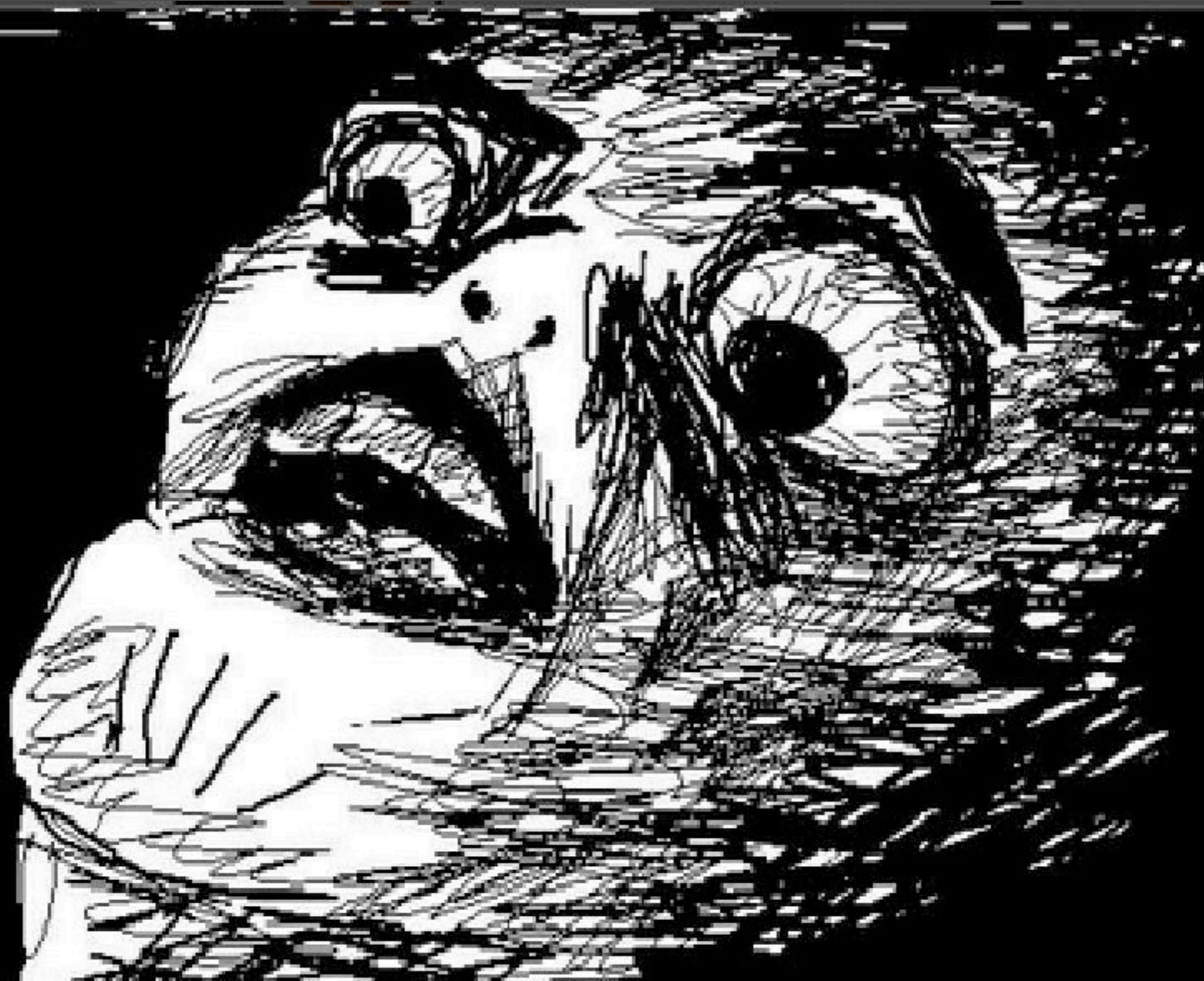
That query isn't bad  
It will work.



# In a Nutshell...

- EXPLAIN will explain slow queries
- Make sure your selects are using keys.
- Avoid tmp tables at all cost!
- Use maatkit mk-query-digest to parse giant query logs





**\$ siege -c 1000!!!**

# Safe Load Testing

- Gradually bump -c levels
- Observe server using htop or top
- Know your load limit
- Generally, load of 1.0 is when the test is over, and you should get back to optimizing.



WE'LL DO IT LIVE!

\$ siege -c 3

\$ siege -c 5

\$ siege -c 8

\$ siege -c 10

\$ siege -c 12

Limit hit.

# What did we learn?

- Our MySQL fix made a HUGE impact!
- Something's up with taxonomy/term pages
- Our new limit is 12 before server overloads.
- MySQL isn't overloaded, the apache process is.



# Captain Obvious says



PHP just  
became a  
suspect.

© five

# XHProf

- Built by facebook
- Not terribly easy to install, but totally worth it.
- There is a session devoted entirely to XHProf tomorrow.





I'LL PROFILE IT. AND WE'LL  
DO IT LIVE.

# Captain Obvious says:



I think we  
found our man.





Cache it?

Memcache it.

Nah dude, he said  
optimize first  
cache later.



WE'RE DOING IT LIVE!  
\$.siege -c 12 (old limit)  
\$.siege -c 20 (new limit)

FORGET TO TURN  
OFF XHPROF AND  
THE MACBOOK  
DIES

# Let's Review

- The cache worked. We've nearly doubled our -c limit (GOOD!)
- We weren't able to optimize our function because of a 3rd party (We'll deal with it)
- PHP is still the bottle neck. Let's go back to profiling.



# Captain Obvious says:



Drupal, panels,  
and views are  
now all  
suspects.



# This is a major milestone

- No slow queries.
- No out of control PHP processes
- In short, nothing glaringly wrong.

# Time for APC

- APC reduces PHP overhead
- APC some claim good to use for cache tables that have small amounts of records, and change infrequently (e.g. cache, cache\_bootstrap, cache\_menu)



## settings.php

```
$conf['cache_backends'][] = 'sites/all/modules/apc/drupal_apc_cache.i  
$conf['cache_class_cache'] = 'DrupalAPCCache';  
$conf['cache_class_cache_bootstrap'] = 'DrupalAPCCache';
```

## php.inc

```
extension=apc.so
```

**WE'LL DO IT LIVE!**

I dunno, seemed  
okay.

Memcache it.

You suck.



# The first rule of APC: Don't talk about settings.

- Remember to check `apc.php` to make sure you have room
- `apc.shm_size = 64`
- `apc.stat = 0`
- Let's try again



# What's the verdict?

- APC has significantly stabilized performance.
- Not a magic bullet
- We'll want to keep an eye on it, and possibly assign more cache tables

# Memcache

- Memcache is good for storing things that change frequently.
- Very effective when used in conjunction with APC



## settings.php


```
include_once('./includes/cache.inc');  
include_once('./sites/all/modules/memcache/memcache.inc');  
$conf['cache_default_class'] = 'MemCacheDrupal';
```

## php.inc

```
extension=memcache.so
```

**DOING IT LIVE!**





Isn't 30 uncached hits a second a lot?

$30 \times 60 \times 60 \times 24 = 2,592,000$  daily

We're done. Can we go?

**We are not done.**



# What was wrong with our test?

- A real page request will ask apache for 10 files (images, etc).
- Real traffic tends to follow not random patterns.

# Further Optimization Strategies

- Panels or views cache on short time
- Develop “smart” panels cache plugins.

# The Future!

- Queue API
- Edge side includes
- Context based caching system



Your session in one word:  
I wish i knew about nginx

I propose that server tuning  
could have solved everything.

Can you show that graph again?  
That was awesome.