



# 3 Non-Code Production Performance Improvements For Developers

How I turned  
broken into working  
without changing  
my code



Created with love by  
**Jon Cram**

<https://github.com/webignition/udiff-jan-2013>

Slide 1 of X

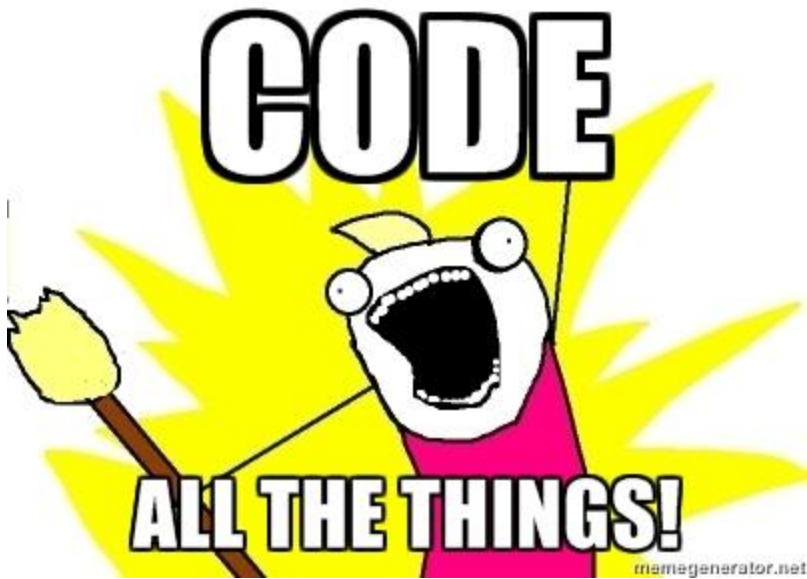


**Jon Cram**

- Professional software developer (Box UK, LGT)
- Got bored (made redundant)
- Built [SimplyTestable.com](http://SimplyTestable.com) for easy automated front-end web testing
- Encountered many an oddity when you build your own production environments

# The Developer Problem-Solving Mindset

# The Developer Problem-Solving Mindset



You use code to solve problems with your code!

# The Developer Problem-Solving Mindset: Example!!!1!

## Problem:

A DB-driven web page listing 20 articles takes 40 seconds to build; each article takes 2 seconds to load from DB.

# The Developer Problem-Solving Mindset: Example!!!1!

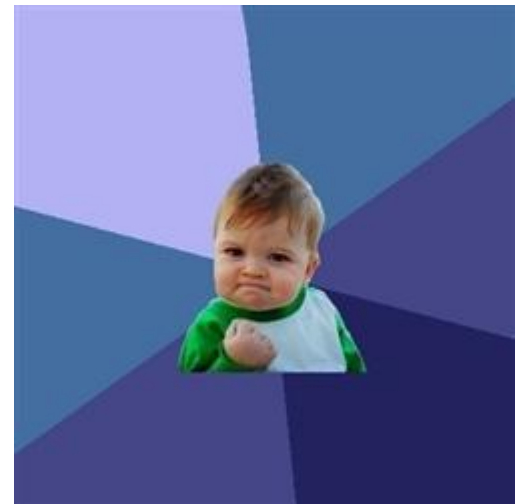
## Problem:

A DB-driven web page listing 20 articles takes 40 seconds to build; each article takes 2 seconds to load from DB.

## Solution:

Examine article model, read ORM docs; make article load in 0.1 seconds; page loads in 2 seconds.

<https://github.com/webignition/udiff-jan-2013>



# Non-Code Changes > Code Changes

I fixed major system-wide performance problems without touching a line of code.

The results were orders of magnitude greater than any code change.

# SimplyTestable.com Overview

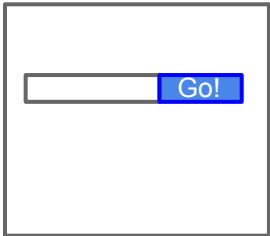
- full-site front-end testing
- performs a set of tests against every URL of a site
- co-ordinated effort between 7 applications
- all applications communicate over HTTP



# Simply Testable Scale By Numbers

- 20k visits per month
- 20k full-site tests performed
- 12 million individual tests
- 70GB MySQL database

# SimplyTestable.com Test Lifecycle



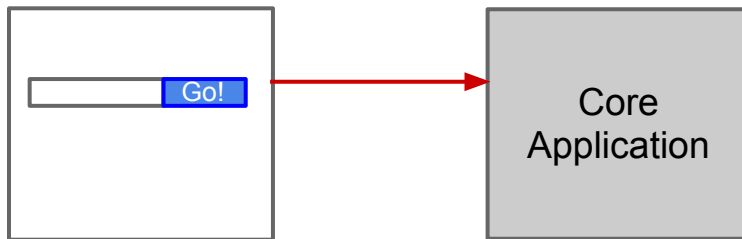
Internal HTTP  
request count:

0

<https://github.com/webignition/udiff-jan-2013>

Slide 1 of X

# SimplyTestable.com Test Lifecycle



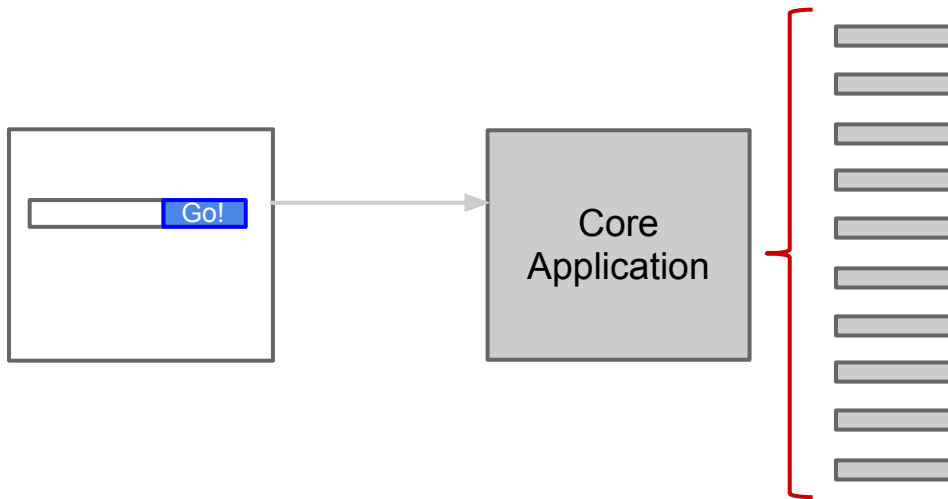
Internal HTTP  
request count:

1

<https://github.com/webignition/udiff-jan-2013>

Slide 1 of X

# SimplyTestable.com Test Lifecycle



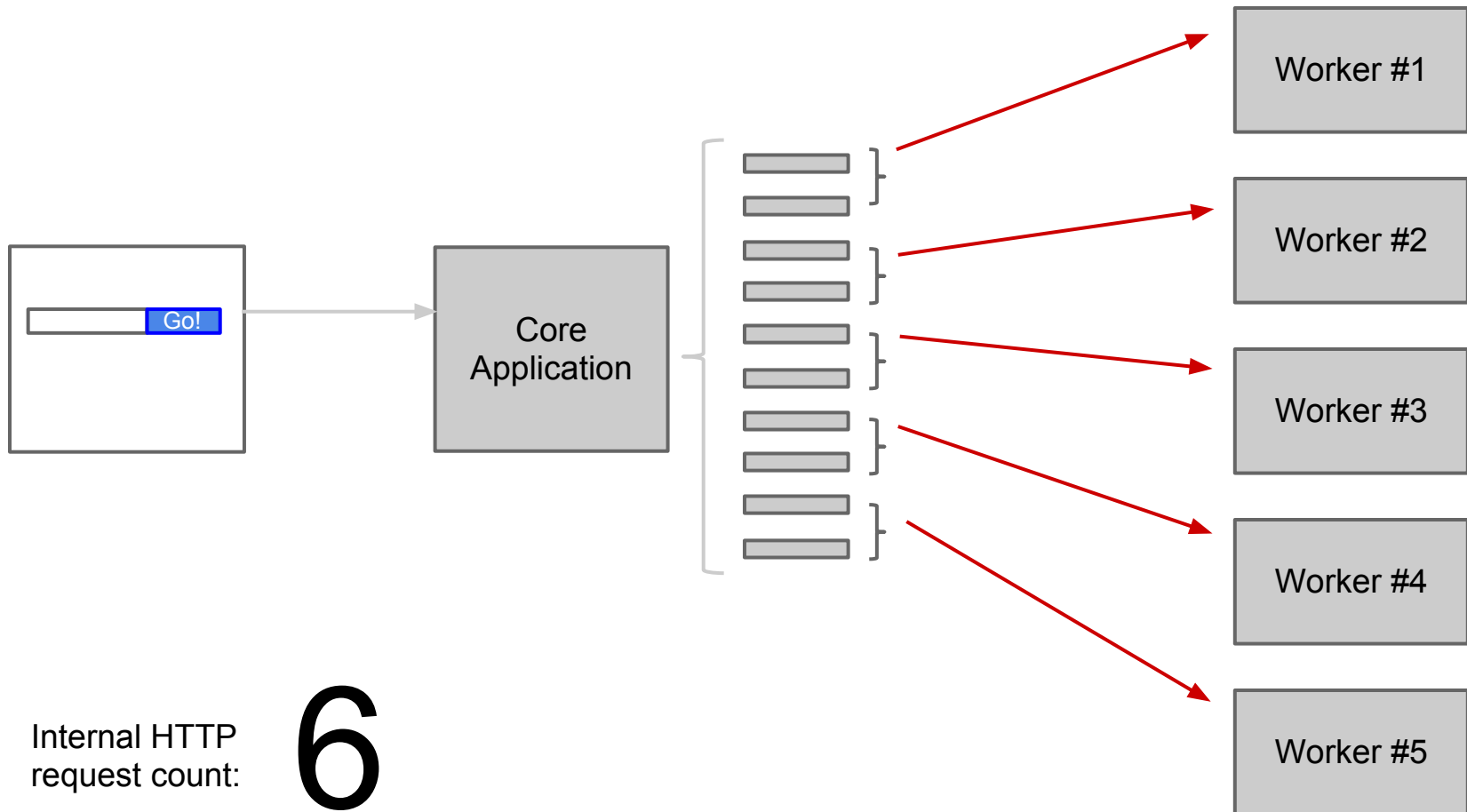
Internal HTTP  
request count:

1

<https://github.com/webignition/udiff-jan-2013>

Slide 1 of X

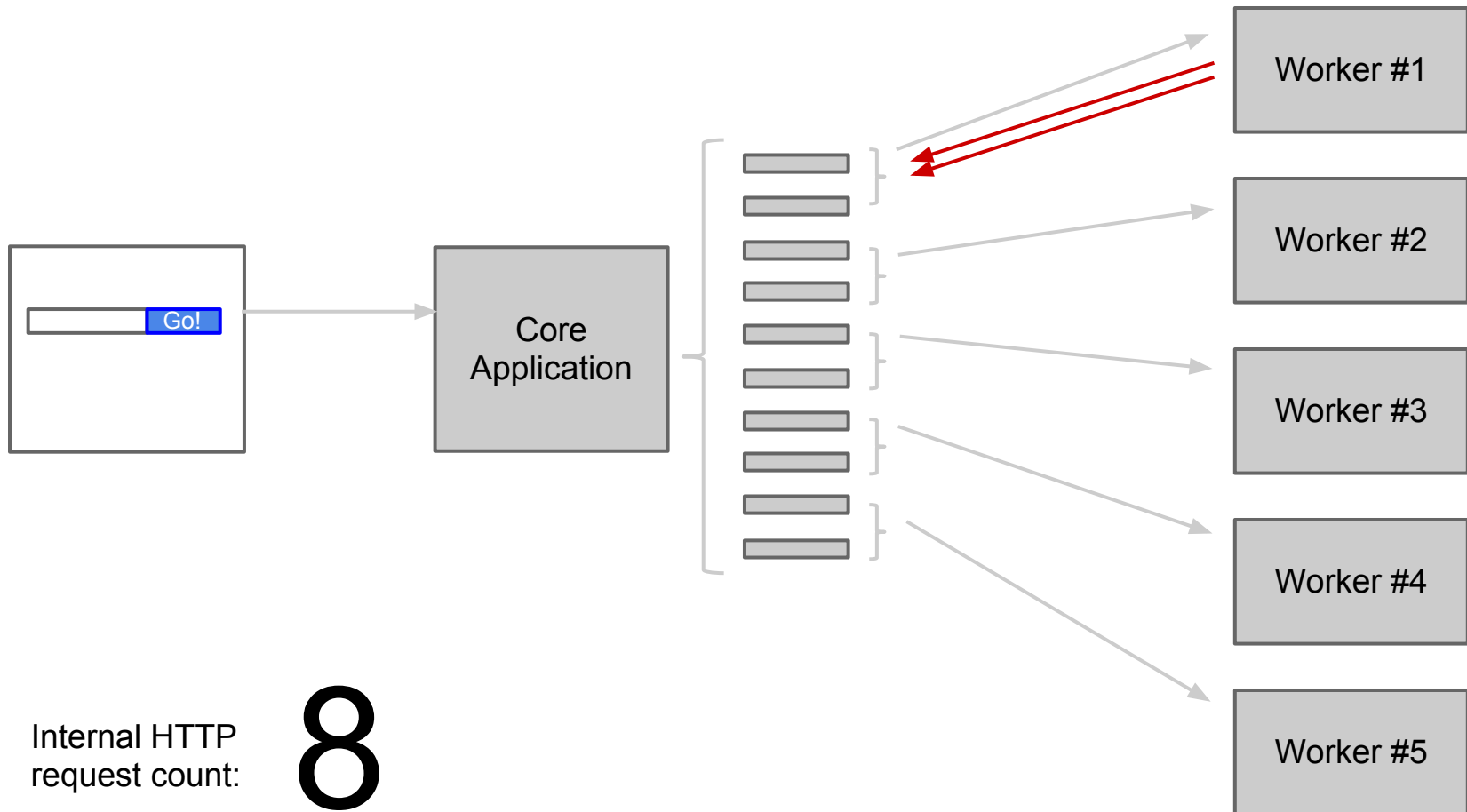
# SimplyTestable.com Test Lifecycle



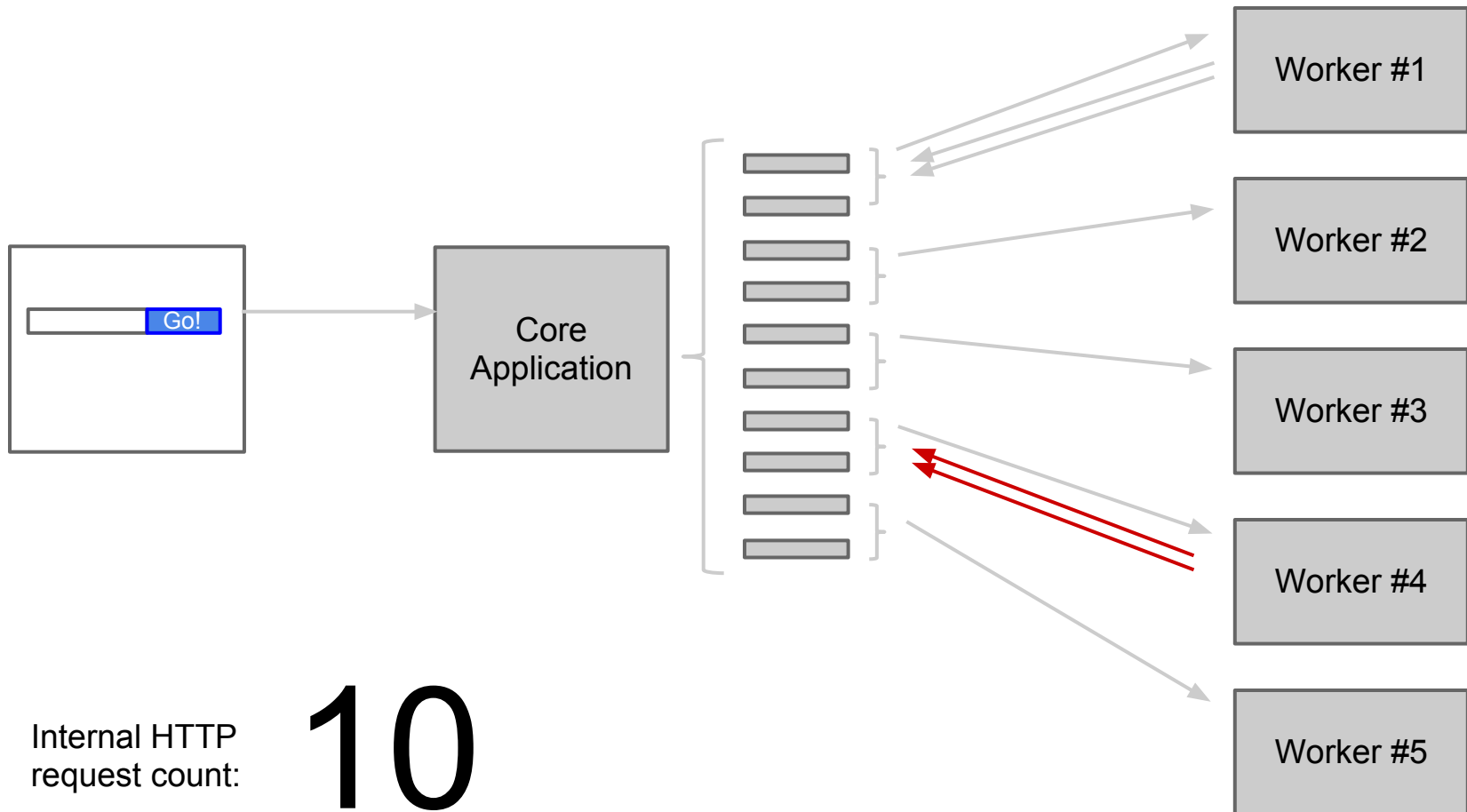
Internal HTTP  
request count:

6

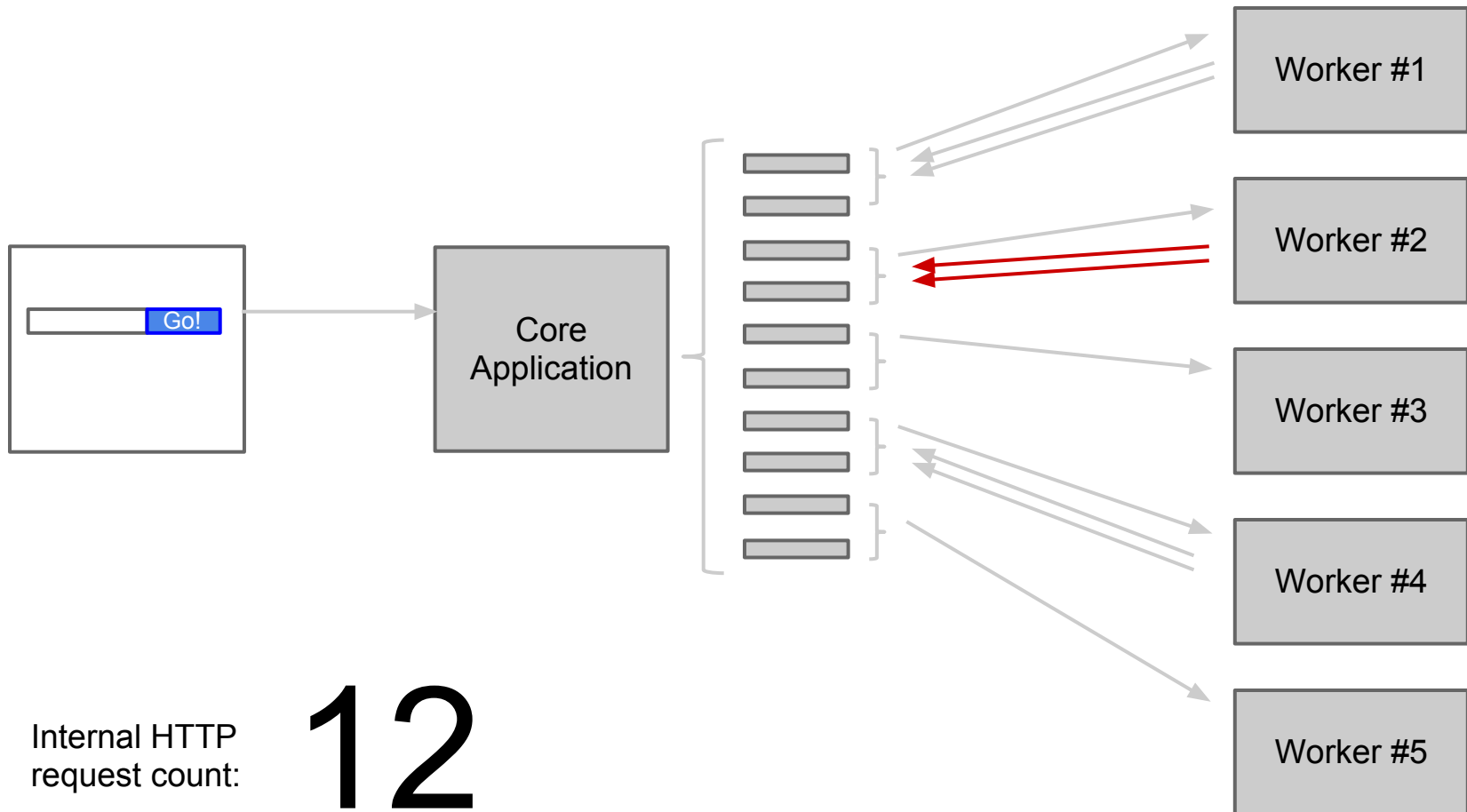
# SimplyTestable.com Test Lifecycle



# SimplyTestable.com Test Lifecycle

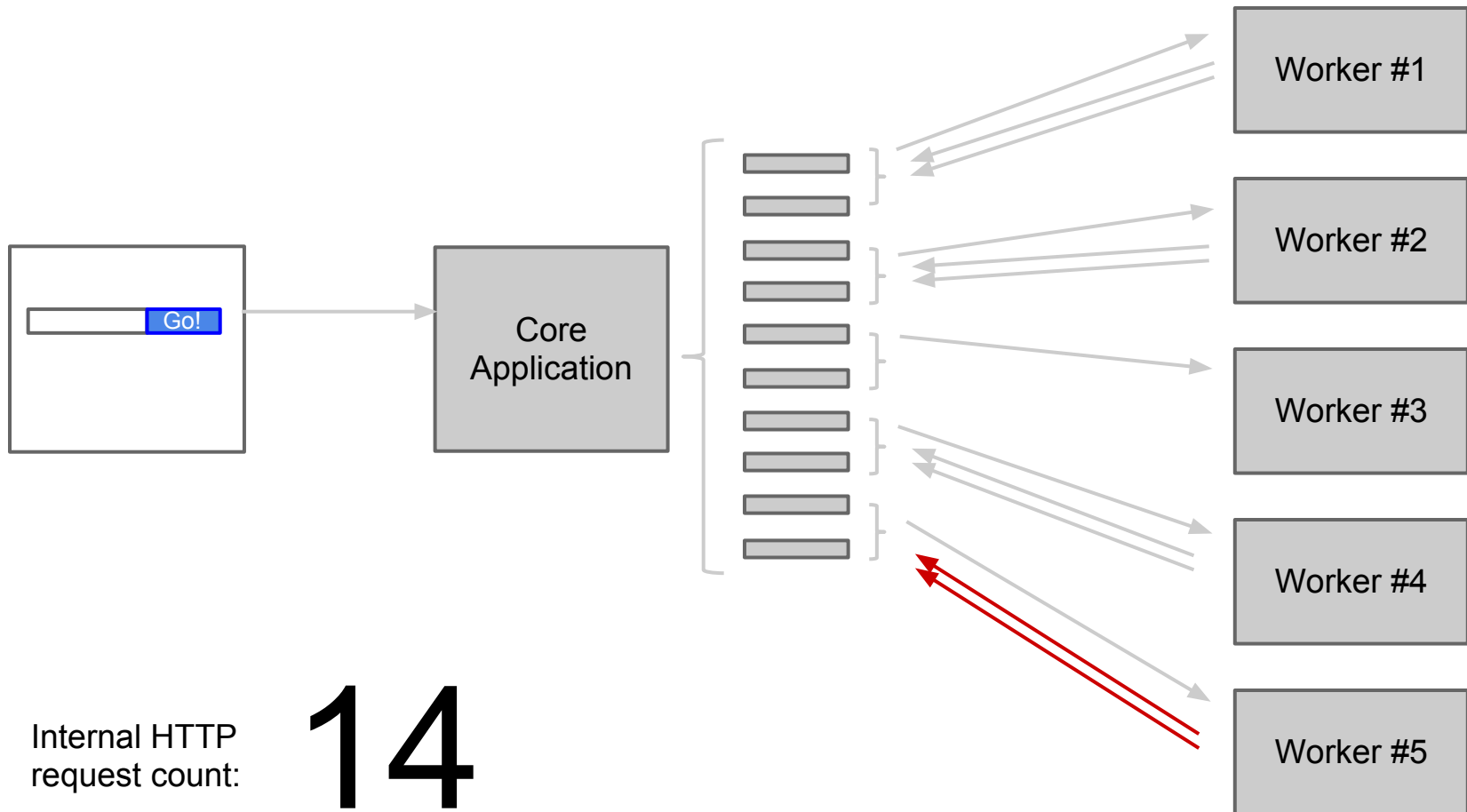


# SimplyTestable.com Test Lifecycle

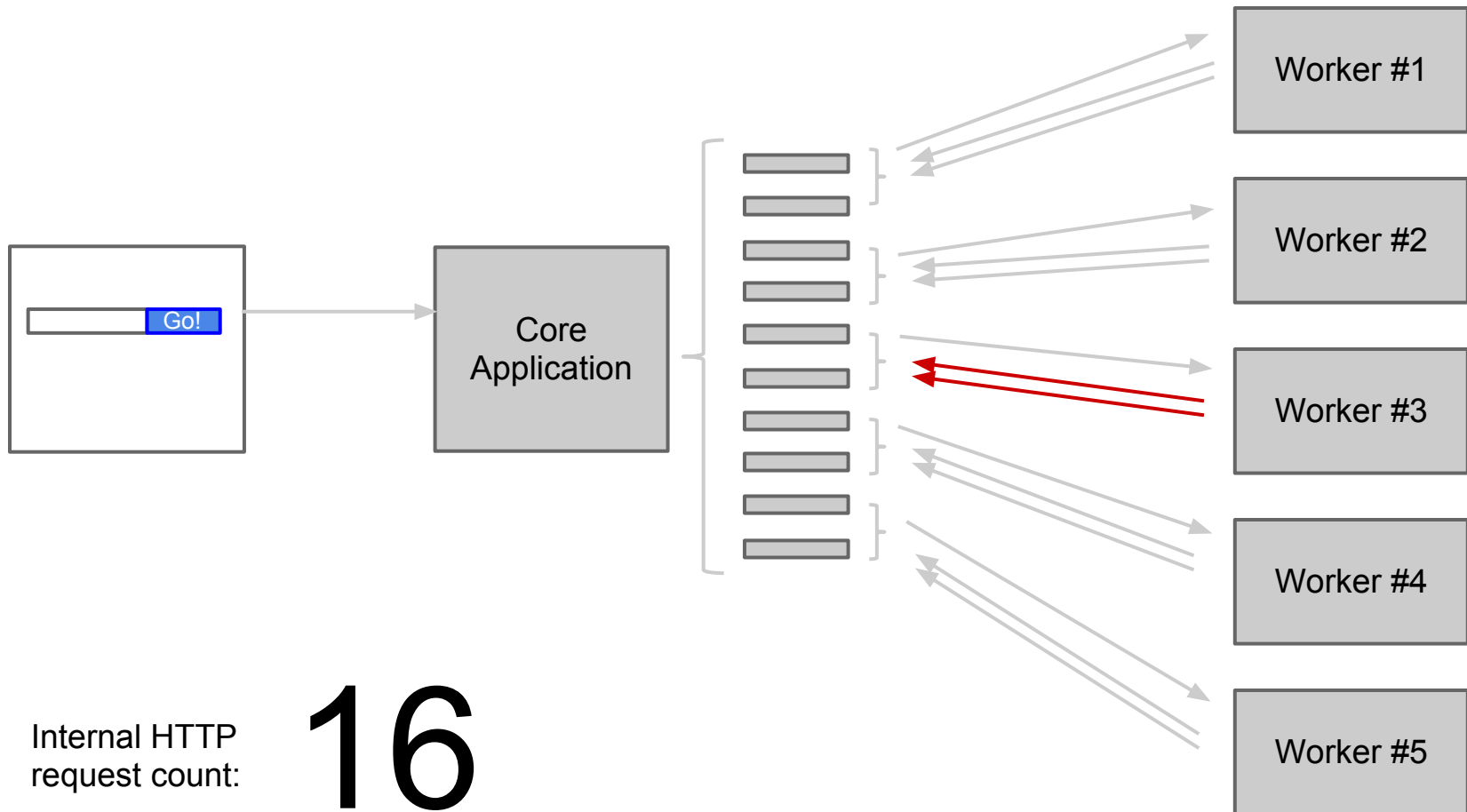




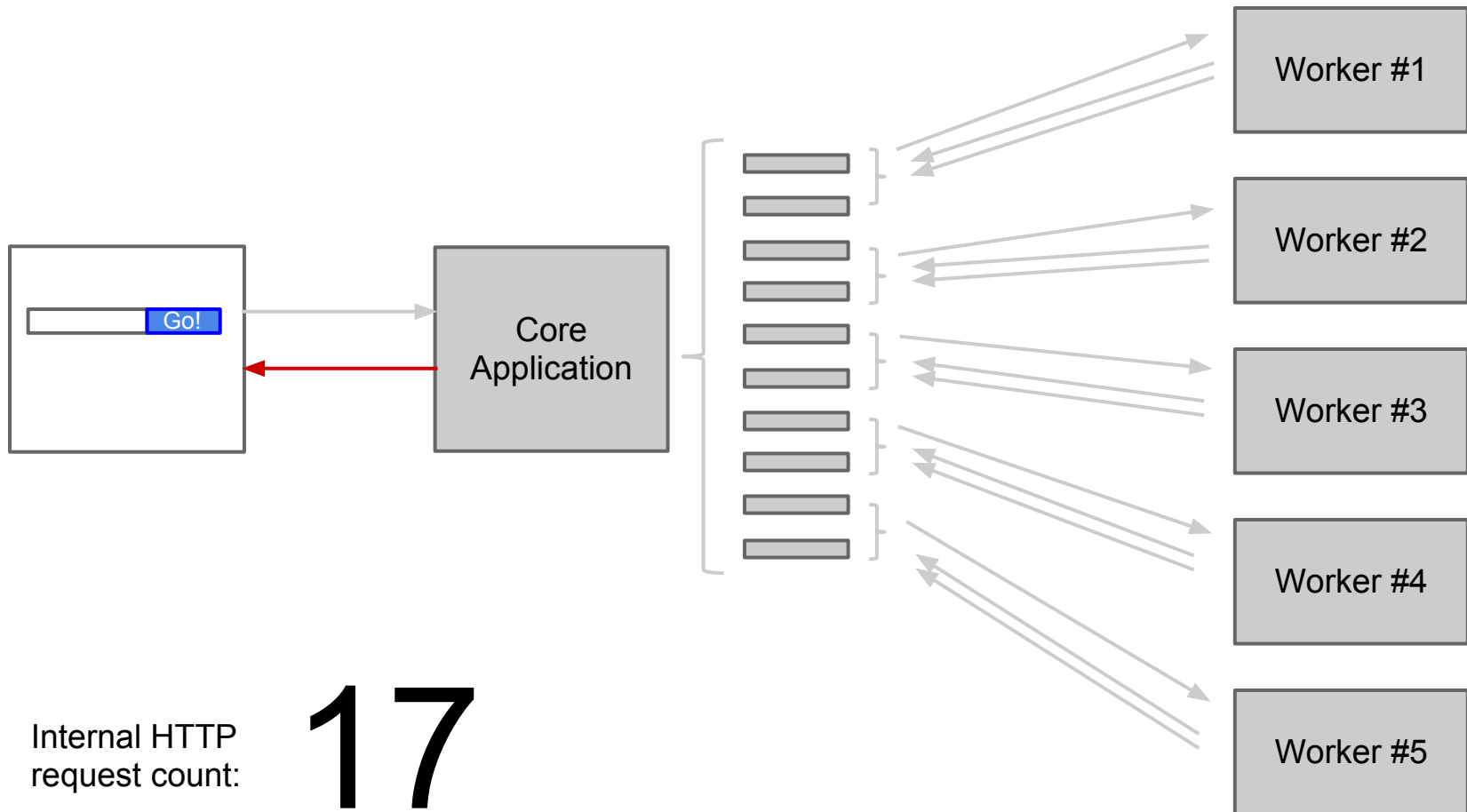
# SimplyTestable.com Test Lifecycle



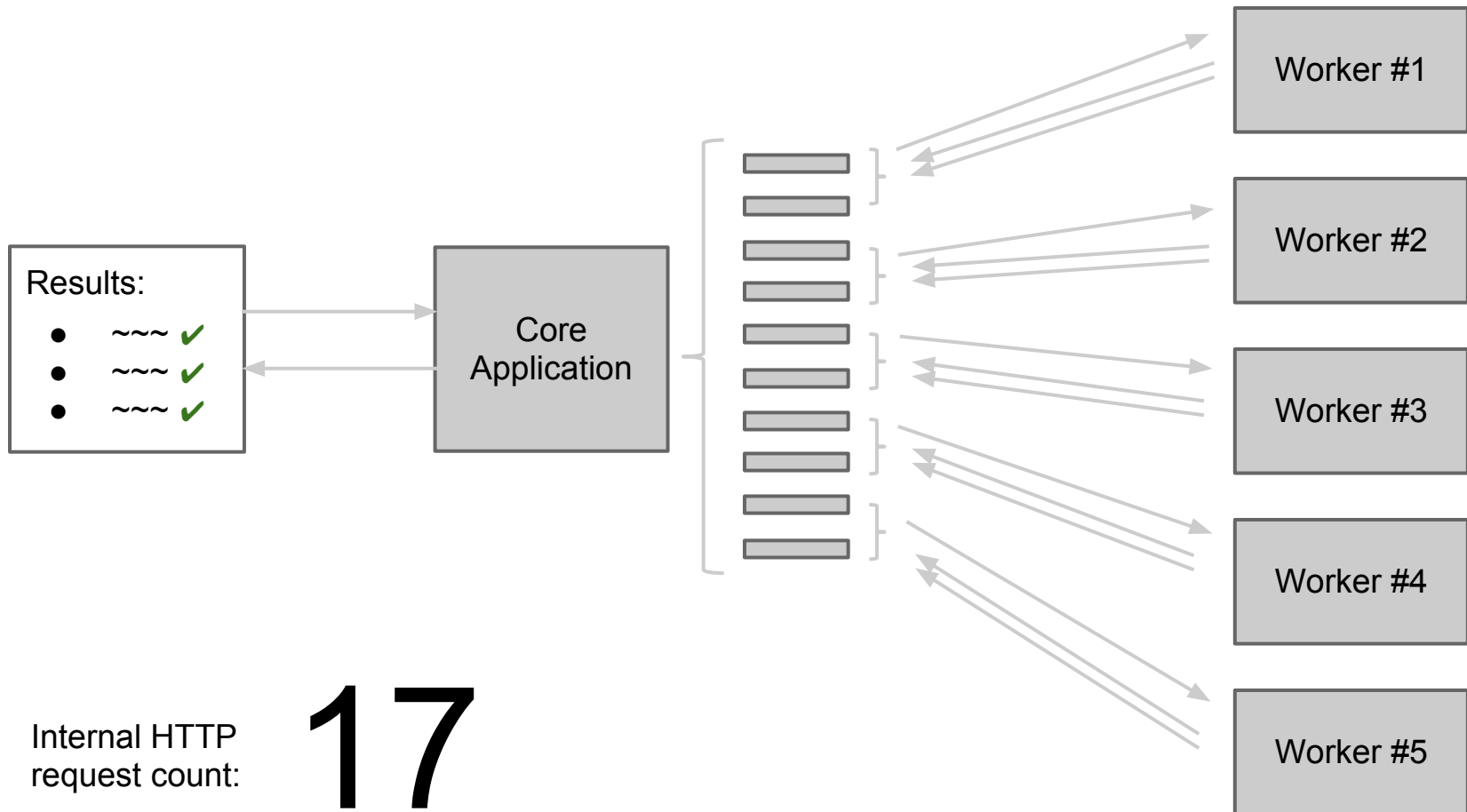
# SimplyTestable.com Test Lifecycle



# SimplyTestable.com Test Lifecycle



# SimplyTestable.com Test Lifecycle



Internal HTTP  
request count:

17

# And Then It Was All Broken

- 12 concurrent full-site tests caused CPU bottlenecks
- DB queries were 2 orders of magnitude slower
- MySQL reading/writing caused disk I/O bottlenecks
- Full-site tests took 10x as long to complete
- Apache timeouts caused arbitrary failures
- FUBAR situations were frequent

# 1: Nginx + php-fpm instead of Apache + mod\_php

# 1: Nginx + php-fpm instead of Apache + mod\_php

Average continuous CPU utilisation:

Apache + mod\_php: 50%

Nginx + php-fpm: 4%

## 2: MySQL loves RAM, give it lots lots

Average continuous CPU utilisation:

MySQL default config: 20%

MySQL 'lots of ram' config: 4%


Tip: set `innodb_buffer_pool_size` to half your total RAM



# 3: Use /run/shm for impermanent file storage

- For Ubuntu and Debian-based distros
- Uses tmpfs filesystem
- Acts like a ram disk without the downsides
- Is available (mounted) out the box

# Improvement By Numbers

	Before	After	Improvement
<b>Concurrent tests</b>	12	40	3x
<b>Average CPU load</b>	30	4	7x
<b>Typical query time</b>	0.2 seconds	0.004 seconds	500x
<b>Complex query time</b>	1 second	0.02 seconds	50x
<b>FUBAR frequency</b>	daily	infrequent	

# Important bits to remember

- Do not use Apache in production
  - Nginx "native" reverse proxy is awesome
  - Nginx config is a delight to use
- Default MySQL configuration is atrocious
  - MySQL query cache is not worth crap
  - Give MySQL all of the RAM things and be done
- /run/shm makes impermanent disk I/O performance a non-issue