3 Non-Code Production Performance Improvements For Developers

How I turned broken into working without changing my code

Created with love by
Jon Cram

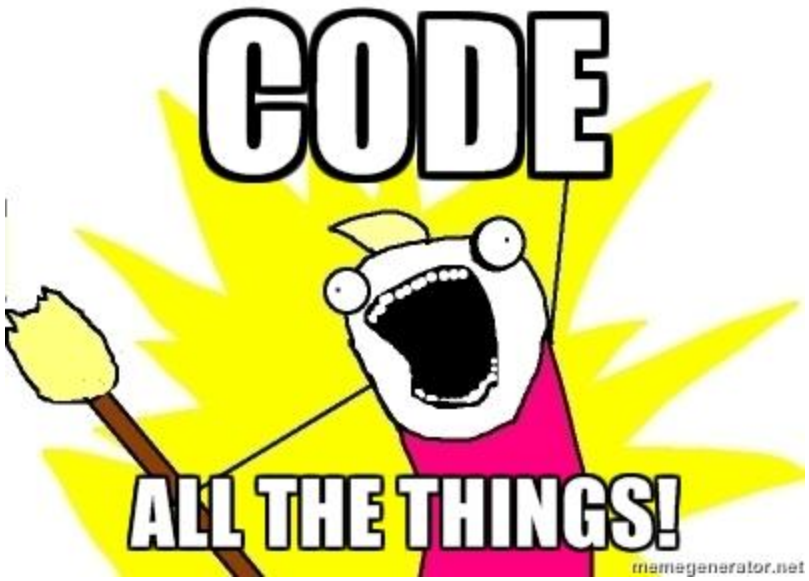
https://github.com/webignition/udiff-jan-2013

Jon Cram

- Software developer (Box UK, LGT)
- Got bored (made redundant)
- Built 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!

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

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.

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

SimplyTestable.com Overview

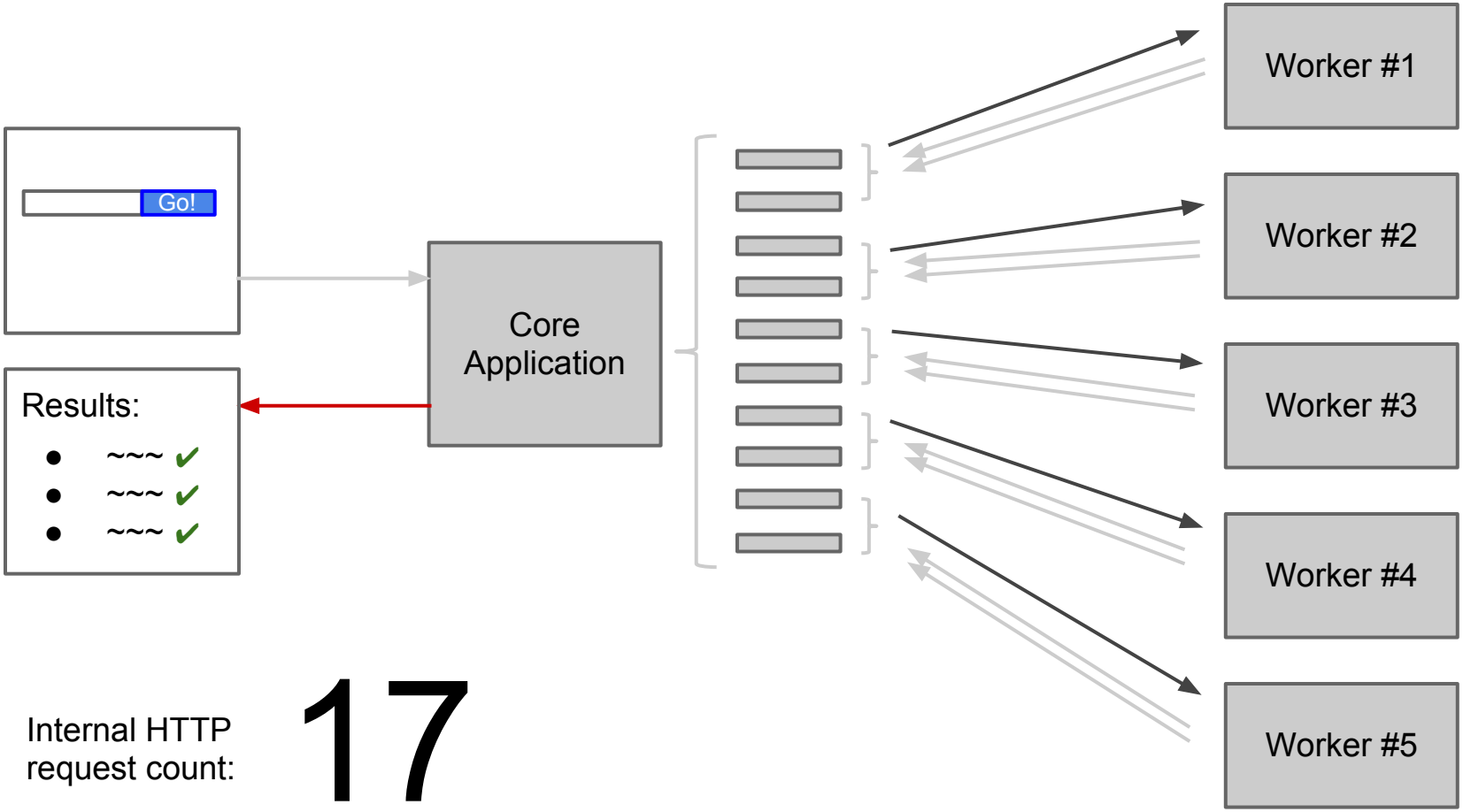
- full-site front-end testing
- performs a set of tests against every URL of a site
- co-ordinated effort between 7 applications
- all applications communicate over HTTP

Simply Testable Scale By Numbers

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

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

SimplyTestable.com Test Lifecycle



Internal HTTP request count: 17

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

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% ✔

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

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

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

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

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

Improvement By Numbers

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

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

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 attrocious
  - 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

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