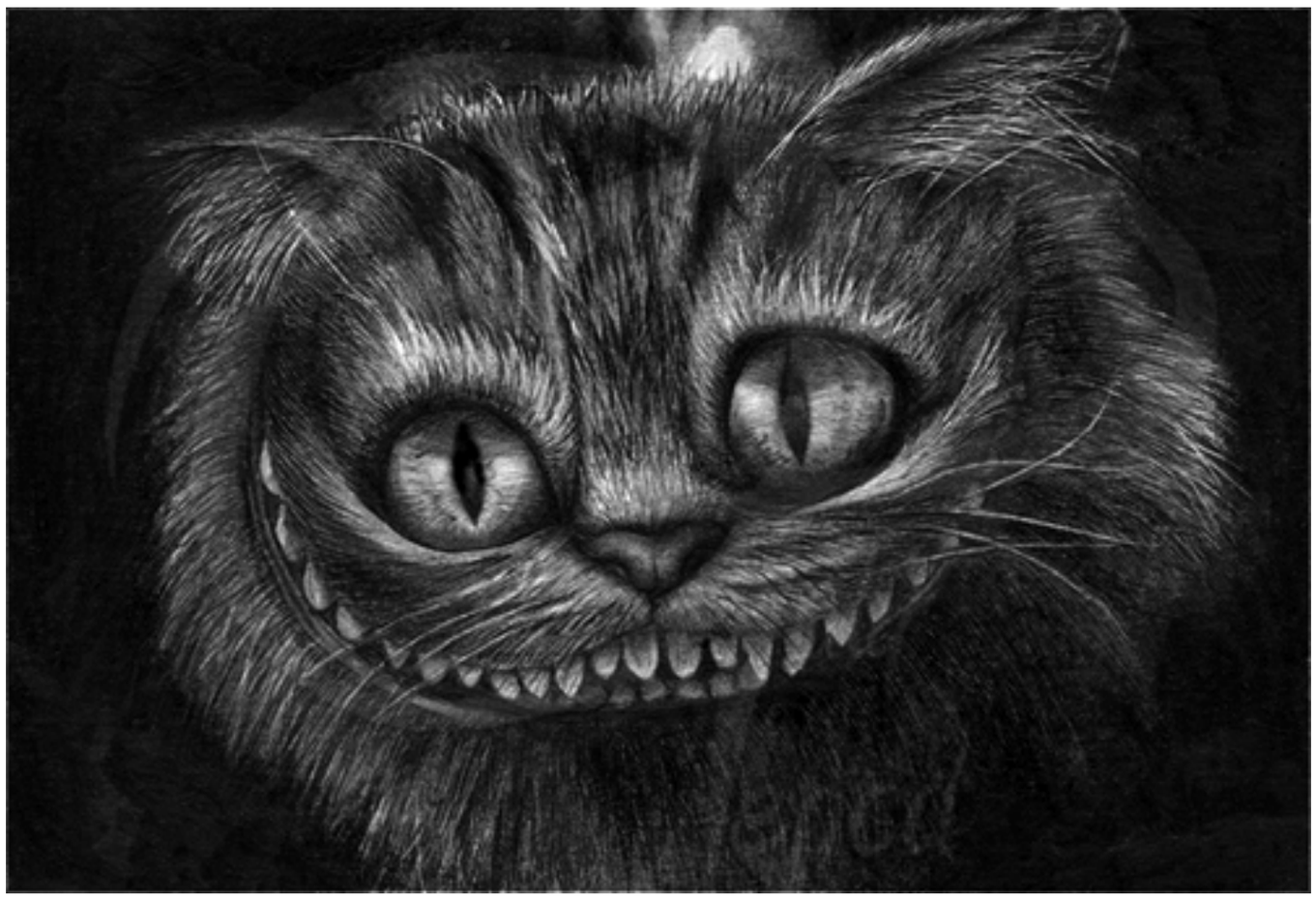

Alice in Performanceland
Down the Rabbit Hole with Frank Wiles

REVSYS revolution systems

www.revsys.com

This talk is about...

- How to think about performance and scalability problems

REVSYS revolution systems

Introduction

This talk is about...

- How to think about performance and scalability problems
- Exposure to some stuff you probably don't normally think about

REVSYS revolution systems

This talk is about...

- How to think about performance and scalability problems
- Exposure to some stuff you probably don't normally think about
- Not really, "Do this in your config for WIN!"

REVSYS revolution systems

Thursday, September 9, 2010

I'm going to talk about how to think/approach these problems. This doesn't always necessary directly or accurately map to the underlying tech or CS concepts. Don't be concerned about this for now.

Thursday, September 9, 2010

Premature optimization can be the root of all evil. And I'm probably the worst sinner in the room.

But there is a big difference between doing things you know are inefficient (I know, I'll store each character in a username as separate rows in the database. Yay me!) and wasting tons of time optimizing away a millisecond on that one view that no one is going to use.

Everyone thinks they are going to be the next Twitter or Facebook, but I know for a fact that you're not going to be the next Twitter or Facebook. How do I know this? Because they already exist. And you’re probably not dumb enough to just completely copy them. Even if your project may be VERY similar to Twitter or Facebook, there will be differences. Different users, different amounts of data, different usage patterns, difference after difference after difference.

Blindly copying another’s tech choice without understanding the reason why it was chosen, the history of what they were using before, the tradeoffs between X and Y, etc. are not usually good ideas. Don't get me wrong, you should watch what others are using and figure out why they are using it and if it *would* be a benefit to the ACTUAL problems you are facing. Using "X because company Y is using it" is a good reason to investigate X, and maybe a great reason to evaluate X before or instead of Z, but it's a horrible reason to just "use X".

Same for blindly using a config option you read in some HOWTO without really understanding (or at least benchmarking the performance)

JFK said "What unites us is far greater than what divides us." It may apply to politics, but it does not apply to performance and scalability. It is the small differences that you need to be taking a close looking at.

Let's say you're building

- Micro blogging service
- 140 character limit
- Over 75 million users
- Similar APIs and a few dozen desktop and mobile clients
- Celebrities use it. It's a household name.

REVSYS revolution systems

Let's say you're building

- Micro blogging service
- 140 character limit
- Over 75 million users
- Similar APIs and a few dozen desktop and mobile clients
- Celebrities use it. It's a household name.

Except you only allow posting once per day.

REVSYS revolution systems

Let's say you're building

- Micro blogging service
- 140 character limit
- Over 75 million users
- Similar APIs and a few dozen desktop and mobile clients
- Celebrities use it. It's a household name.

Except you only allow posting once per day.

TOTALLY DIFFERENT

PROBLEM

REVSYS revolution systems

Setup as much logging and monitoring as you can. To paraphrase the quote, "If you don't know where you're going any tech will lead you there."

It's important to establish a baseline to measure against, to know what is "normal"

There is a lot going on in a web request...

DNS, is it cached locally?

TCP/IP, handshakes, etc.

Routing, switching, bandwith

There is a lot going on in a web request...

DNS, is it cached locally?

Dispatching from load balancer to web server and through Apache

Speed of light...

TCP/IP, handshakes, etc.

Routing, switching, bandwith

Parsing your GET/POST

There is a lot going on in a web request...

DNS, is it cached locally?

Dispatching from load balancer to web server and through Apache

Speed of light...

TCP/IP, handshakes, etc.

mod_wsgi

Your database library

Parsing your GET/POST

Routing, switching, bandwith

Your Django code

Parse, plan, and execute your SQL

REVSYS revolution systems

There is a lot going on in a web request...

DNS, is it cached locally?

Back into your app

Speed of light...

Dispatching from load balancer to web server and through Apache

...back out the load balancer

TCP/IP, handshakes, etc.

mod_wsgi

Your database library

Back to your webserver...

Routing, switching, bandwith

Parsing your GET/POST

Render your template

Your Django code

Parse, plan, and execute your SQL

REVSYS revolution systems

And I've glossed over a few hundred steps along the way...

Do Less

- Less queries
- Less calculations
- Less parsing
- Less requests
- Less work

REVSYS revolution systems

I think Mike Malone originated this quote, “Performance isn’t about doing things faster, it’s about doing less”

The only easy wins...

- Turn off DEBUG
- Cache the easy stuff
- Gzip your content
- Set “Far Future” headers where possible
- Use something like YSlow to check yourself

REVSYS revolution systems

Thursday, September 9, 2010

First knock out the dead simple stuff that will be helpful to basically everyone.

“Far Future” -- Setting ETags and using the ConditionalGet Middleware are great, but the browser still ends up asking
you the question. Remove the question.

"Reality is what you can get away with" -- Robert Anton Wilson

REVSYS revolution systems

Performance is often really about the perception to the user. For example, you need to be at least looking at things like render time.

Everything is about trade offs. Time vs Money. Complexity vs Simplicity.
Evaluating the new hotness vs learning more about what you're already using.
Learning more yourself vs hiring someone who already knows.

In the near term throwing more hardware at it works and is maybe the right solution,
but over the long term you can save money and time by fixing your issues.

Prioritize your optimizations

Like Alice in Wonderland, look at the door in different ways. Start off looking at it in a macro sense.

Find the slowest bits. The most time consuming bits. The most frequently used portions of your site. Remove the overall "pressure" on your server.

Then look at it in the small/micro sense. Dig into that process. What can we remove? Does *this* bit here really need to be real time? Can we cache that? Or omit it entirely? Looking at how people really use your site can pay huge dividends.

Prioritize your optimizations

Some simple math. Removing a database query from a view that is visited 100x as often as another, removes 100x as many database queries from your system.

Reducing the time to process the same view by 100 milliseconds saves 10x seconds of "work" vs the view that is rarely hit.

REVSYS revolution systems

Thursday, September 9, 2010

I like to think about it as system or server "pressure". The overall pressure of doing the work. It helps to keep me from getting distracted by minutiae at this point.

Where to look first

- Remove database queries where you can
- Sessions
- Middleware
- ContextProcessors
- Utility functions and anything used throughout the app
- Any place X talks to Y

REVSYS revolution systems

Things to think about

REVSYS revolution systems

Does this bit right here need to be real time?

Can we live with it being cached for a little while?

Can it be 5 minutes out of date? An hour? A day?

Does it even need to be on this page at all?

REVSYS revolution systems

Thursday, September 9, 2010

I find the last question is VERY rarely asked from a performance perspective.
It's often talked about from a design perspective. While it looks fine on the page and is maybe useful, how useful? Do 50% of people use it? 5%? Or is it 5 people every 5 years...

How's the connectivity between my servers?

Is that maybe an issue?

What about between me and my users?

REVSYS revolution systems

When is the last time you checked any of this? Or are you just assuming you've got 100Mbps or GigE.
You might be surprised...

Am I swapping?

Why am I swapping?

Am I using my resources effectively?

Am I doing to much disk I/O here? Can I skip some of this?

REVSYS revolution systems

Is caching this slower?

REVSYS revolution systems

(FYI it can be for really simple operations due to seralization/desearlization or wasted resources if you cache things that will never be retrieved again)

Should we be using ESI?

REVSYS revolution systems

To make caching a bit easier in a multi-user system, should you be using this?

Can I **MOVE** this work somewhere else?

Should I be using Varnish in front of my app servers so they aren't bothered with the request or having to make the determination to cache at all?

REVSYS revolution systems

Can I move it out of the request? Can I move it to another server?

If I can move it somewhere else, especially horizontally out of band, that helps my ability to scale.

If you're having a problem today that you didn't have yesterday. What changed?

You'd be surprised how a small change can turn into a big problem quickly.

REVSYS revolution systems

Questions?

Or if you're shy, work for the NSA or in stealth mode...

frank@revsys.com @fwiles

REVSYS revolution systems

Photo Credits:

[1] http://www.flickr.com/photos/onlypencil/3768745143/sizes/m/

[2] http://www.flickr.com/photos/jm999uk/3513237937/sizes/l/