

# CS 188

## Scalable Internet Services

Andrew Mutz  
September 26, 2019



# Today's Agenda

- Course Introduction
  - Motivation
  - Course Structure



# Course Introduction

Let's say...

...I want to find a home to live in.

...I am lost in a foreign city.

...I want to go on a date.

... what do I do?



# Course Introduction

Every day, billions of people use the same suite of technologies to solve these problems: internet services.

As these services get increasingly popular, they need to continue to function.

Scaling even relatively simple web applications (Twitter) can be very complex.



# Course Introduction

Suppose you've built something the world is excited about.

**What do you do when your popularity doubles?**

What do you do when your data set doubles in size?

- And doubles again. And again...

**These are good problems to have, and this course is about how you solve them.**



# Course Introduction

## What do we mean by an Internet Service?

- For the purposes of this class, we are referring to HTTP services.
- Yes, there are many other protocols on the internet, but HTTP is the 800 pound gorilla.
- Do we mean HTML?
  - Not necessarily. HTTP is much more than just HTML.



# Course Introduction

## Do we mean web browsers?

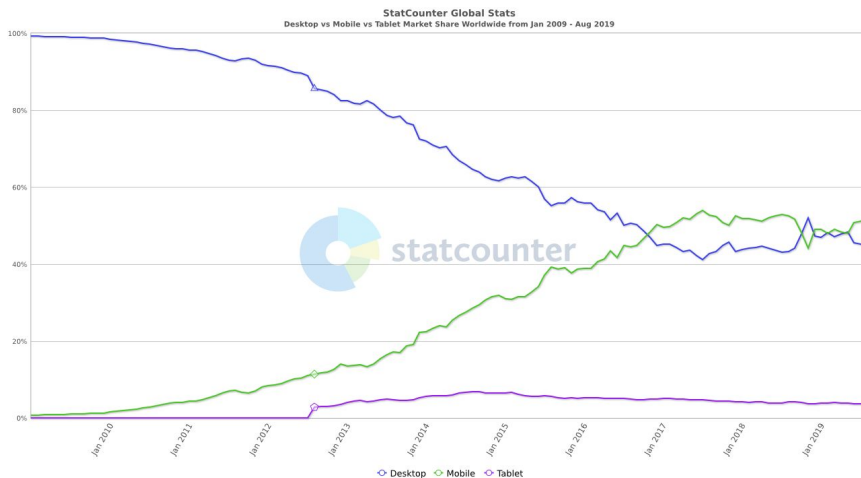
Yes, but so much more.

- HTTP is the foundation for most APIs on the internet today
- REST, GraphQL and SOAP are conventions on top of HTTP for machine to machine communication.
- If two companies, or two applications today want to exchange data, HTTP is the go-to solution



# Course Introduction

Do we mean mobile devices?



Yes!

- Mobile web isn't going away
  - We will cover optimizations.
- Most native mobile applications are backed by internet services.
  - We will cover REST





# Course Introduction

Ok, so we're clear on what we mean by an "internet service".  
**What do we mean by scalable?**



# Course Introduction

Ok, so we're clear on what we mean by an "internet service".

**What do we mean by scalable?**

An internet service is scalable if increasing demands can be effectively met with increasing capacity.

- By demands, we usually mean traffic
- Other demands apply as well, such as the size of data set



# Course Introduction

What do we mean by “effectively meet demands”?



# Course Introduction

What do we mean by “effectively meet demands”?

- Service stays available
- Response time does not excessively degrade

An example of an internet service that is not scalable:

- Run everything on one server.
- When traffic grows too high, buy a faster server
  - **Why is this not scalable?**



# Course Introduction

Scaling is the core of the course, but along the way we will learn about the tools and technologies surrounding scalable internet services today:

- Performance
- Agile software development
- Test driven development
- Security
- Intelligent applications



# Course Introduction

## In summary...

This course won't teach you how to build an application that gets worldwide attention and usage.

- That's your problem!

This course will teach you how to build an application **that can respond to** worldwide attention and usage.



# Course Structure

- Taught by
  - Andrew Mutz
    - Chief Technology Officer, Appfolio
    - Taught this course UCLA 2014-2016
  - John Rothfels
    - Director of Engineering, Appfolio/Dynasty
    - Taught at Stanford 2009-2012
- Teaching Assistants
  - Karl Marrett (kdmarrett@gmail.com)
  - Rishab Ketan Doti (rishabkdoshi@g.ucla.edu)



# Course Structure

**Lectures Tuesday and Thursday at 12PM in Boelter Hall 3400**

- Lectures will cover the concepts introduced in this class

**Lab Friday at 12pm in Dodd Hall 78 & 2pm in Rolfe Hall 3126.**

- The lab will be focused on the course project.
- Demo each week.
- Remember to charge your laptops.





# Course Structure

This course is **project intensive**. We will be learning about scalable internet services by building them. You will...

- Work in teams of four.
- Develop an interesting internet service
- Deploy it on Amazon EC2
- Measure its performance and scalability
- Apply the techniques presented in class to improve it
- Document these improvements and present them



# Course Structure

**This course demands a lot of work, but can be very rewarding.**

Skills you will learn in the next 10 weeks:

- A programming language (Ruby)
- An application development framework (Rails)
- Amazon Web Services: EC2, S3, ElasticBeanstalk
- How to load test an internet service using Tsung
- Test-Driven Development
- Agile/Scrum software development



# Course Structure

This is **not a deep-dive** in Relational Databases, Networking, Distributed Systems, Network Security, or Cloud Computing, but will touch on each of these.

- If you are looking for deep dives, please see
  - Networking: CS 118, 117, 211, 218
  - Distributed Systems: 133
  - Databases: 143
  - Network Security: 136



# Course Structure

This course has an **industrial focus**. We will use industrial software development techniques.

- The goal of the course is successful projects that demonstrate the concepts introduced in the course
- All project code will be open source
  - Learn from your classmates!
- Toolchain will be from industry:
  - Git(hub), Ruby/Rails, Travis CI, NewRelic



# Course Structure

## Why Ruby on Rails?

- Rails weaknesses: CPU usage, memory usage
- Rails strengths: building applications quickly

In order to teach advanced scaling topics, you need to get a project from zero to working quickly.



# Course Structure

What does this course have to offer you?

- If you're going into academia, knowledge of industrial software engineering is valuable
  - Sets the context for many important problems
- If you're headed for industry, there's never been a better time to understand this technology stack
  - It's less obvious than it was 5 years ago, but Rails is still the go-to choice for startups



# Course Structure

There are two course texts:

- **Concepts**

- “High Performance Browser Networking”
- by Ilya Grigorik
- Available for free online

- **Practice**

- Ruby on Rails Tutorial Book
- by Michael Hartl
- Available for free online



# Course Structure

Your grade will be based on three main factors

- **Your final project presentation**
- Your final project paper
- Your teammates' evaluation of your contributions

Also considered, to a lesser degree

- Weekly demo progress
- Assistance to other students on Piazza





# Course Structure

## Course Project

- Goal: Gain hands-on experience building and deploying a scalable web service
- Your project will...
  - Use Ruby/Rails/EC2
  - Have an interesting and large data set
  - Demonstrate scaling improvements with measurements
  - Ideally, relate to something that interests you



# Course Structure

## Course Project

- Process
  - Project teams of 4 students
  - Use weekly sprints to make progress
  - Use modern software engineering techniques:
    - Scrum, TDD, Pair programming
- Production deployed on Linux
- For your development machines, you have choice, but Unix is **highly** recommended (Mac OS, Linux, etc.)
  - Development on Windows is discouraged and not supported



# Course Structure

## Course Project

Want inspiration?

- Look to <http://scalableinternetservices.com/projects>
- Some fun ideas:
  - Uber for haircuts: hail a hairdresser/barber
  - Location-based game: compete for control of campus
  - AirBnB for pets: timeshare your dog



# Course Structure

- Course websites

- <http://scalableinternetservices.com>
- <https://github.com/scalableinternetservices>
- <http://piazza.com/ucla/fall2019/cs188>
  - Email notifications are a good idea
  - Students helping others is strongly encouraged



# Course Structure

## Lecture Structure

- The course lectures will cover material you need to build a successful project, and additional material that won't be directly applied.
- Necessary content for successful projects is front-loaded so you can apply these concepts early.
- The lectures in the second half of the quarter will be additional material and guest lectures.



# Course Structure

## First five weeks will cover:

- Intro to the basics: HTTP & HTML
- Industrial software engineering: Agile, TDD, CI, Pairing
- HTTP Application Server architectures
- High availability via load balancing: a share-nothing web stack
- Client-side and server-side caching
- Using relational databases in web applications: concurrency control and query analysis
- Scaling via sharding
- Scaling via SOA
- Scaling via read-slaves



# Course Structure

Subsequent lectures will be guest speakers and

- Scaling via non-relational data stores (NoSQL)
- Basics of web security: Firewalls, HTTPS, XSS, CSRF
- HTTP 2.0
- Client-side renaissance: Client-side MVC
- Thicker clients: Asm.js, Emscripten, Webruby
- Content-delivery networks
- How to build intelligent web systems



# For Next Time...

- No lab this week!
- Read Chapters 1 & 2 of HPBN
- Read <http://scalableinternetservices.com/projects/>
  - Start thinking about projects and groups
  - Post to Piazza with project ideas
- Start working on Ruby CodeAcademy
  - <http://www.codecademy.com/en/tracks/ruby/>

