

# **CS290b - Lecture 1**

## **Ready, Set, Go!**

---

**Scalable Internet Services, Fall 2013**

**Jon Walker**

**Department of Computer Science**

**University of California at Santa Barbara**

# Course Goal

## ■ Scalable Internet Services

- How does one think about building these?
- What are the components involved, how do they work?
- What have others done? Read some papers.

## ■ Learn by doing, not listening

- Roll-up your sleeves and get working!
- 1 big project in multiple parts, lots of challenges, lots of work
- Lots to learn and use: Ruby, Rails, Apache, MySQL/NoSQL, EC2, Web Services, Load balancing, Parallelism, Fault tolerance, and more
- ... the lectures are 30% how-to, 70% background

# For today...

- **Course Intro – “Scalable Internet Services”**
- **Course project**
- **First Assignment**
- **Ruby on Rails – framework for this course**

# Your Info

 **Please write the following on a note card**

- Name
- Email
- MS/PhD and year
- Web technology experience?
- What do you hope to get out of this class?

# Jon Walker's Background

## ■ MS CS at Cal Poly San Luis Obispo

- Emphasis on network protocols

## ■ Adjunct Professor at UCSB and Westmont

- Taught Scalable Internet Services Fall 2011 & 2012
- Taught Software Engineering for 5 years
- Colloquia and talks on cloud computing and internet services

## ■ CTO at Miramar Systems

- Acquired by Computer Associates in 2004

## ■ Founder and CTO at Versora

- Acquired by Kaseya in 2006
- Web based systems management software

## ■ Founder and CTO at AppFolio

- Software-as-a-Service business solutions for vertical markets
- Using Ruby on Rails
- Agile Software Engineering processes

# Sarah Jones' Background

- **MS CS at UCSB**

- **Former student**

- Took Scalable Internet Services Fall of 2012

- **Works with Ruby on Rails**

- Software Engineer at AppFolio

# Define “Scalable Internet Services”

## ■ “Internet Services”

- Web sites
- Web services
- What about mobile?

## ■ Web sites – typically:

- Serve web pages in HTML
- Used/viewed by humans
- Database backed (usually relational but can be NoSQL)
  - ◆ Alternatively
    - Static files
    - Application-specific database
    - Live data, e.g. webcam
    - Live communication, e.g. chat, screen sharing, audio

# Define “Internet Services”

## ■ Web services

- Provide programmatic access to services / resources
- Typically RPC-style of interaction over HTTP
- Using XML or JSON to encode complex data structures
- Variety of protocols
  - ◆ REST, XML-RPC, SOAP, WSDL, WS-...
- Examples:
  - ◆ Your favorite Google, Twitter, Amazon, Facebook functionality



# Characteristics

## ■ A way to deliver applications and data to a large and mobile audience

- In contrast to selling software
- In contrast to selling data on CDs

*Service*

## ■ Have the potential to grow to a huge worldwide user base

- Scalability is always a consideration
- Authentication and authorization are always in play
- 24/7 operation and availability are concerns
- Contrast to locally installed single-user applications...

*Scale*

## ■ Enable communication among users

- Community, sharing, recommendations, etc.

## ■ Can be updated at any time

- Eternal “beta”, daily/weekly code upgrades
- Data feeds

# Define “scalable”

- From “Building Scalable Web Sites” by Cal Henderson, chapter 9



## ■ A scalable system has three characteristics:

- The system can accommodate increased usage
- The system can accommodate an increased dataset
- The system is maintainable

## ■ Scalability is one of the most abused terms by marketing

- What is the difference between scalability and performance?

# Demand, infrastructure, service levels

- ***“As demand grows, the system can be expanded to maintain service levels”***

- **Demand**

- Number of users
- Amount of data
- (frequency of use, geographic diversity, ...)

- **System**

- Servers (cpu, memory, ...)
- Storage (disks, storage systems, ...)
- Bandwidth (routers, network connections, ...)
- People (operators, programmers, sysadmins, ...)

- **Service levels**

- Response time / download speed
- Availability
- Functionality

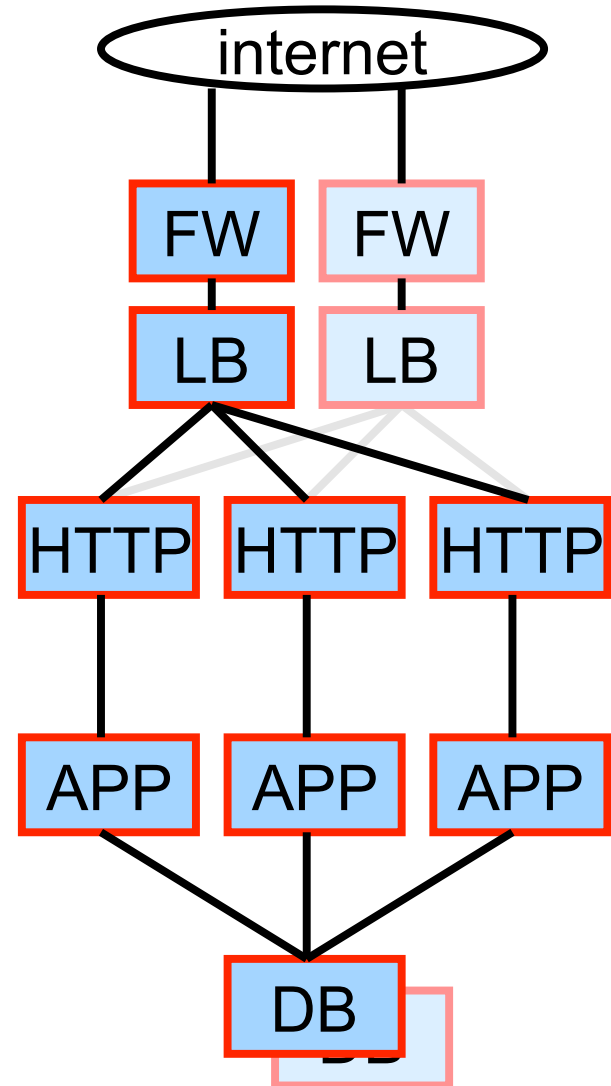
# “Scalable”

- **If users and data increase 10x  
can the servers, storage and bandwidth be  
increased to maintain response time?**
- **Second order questions:**
  - At what cost?
  - What is the demand -> cost function? [  $\text{cost} = f(\text{demand})$  ]
  - What is the headroom, what are limits?
    - ◆ E.g. what about another 10x increase, and another, and another?
- **Do not confuse with performance!**
  - With X+Y+Z as infrastructure, we can handle N users, M data, and provide response time R
  - Of course, good performance makes scalability easy

# How to build scalable web services in one slide

## Simple, share-nothing web stack

- Network devices – routers, firewall, load balancer, DNS, global LB, fail-over, ...
- HTTP server – thread/process model, HTTP protocol, caching, encryption (SSL)
- Single threaded app server – model-view-controller programming model, RoR, concurrency, atomicity, clustering, caching, threading
- Relational database – SQL, replication, redundancy, disaster recovery, partitioning




# Why Is It So Hard?

# Another Amazon Outage Exposes the Cloud's Dark Lining

By Brad Stone  | August 26, 2013



SEND TO  kindle

It was likely another eventful weekend for the engineers in Amazon's Web services division. On Sunday afternoon, a hardware failure at Amazon's U.S.-East data center in North Virginia led to spiraling problems at a host of well-trafficked online services, including Instagram, Vine, AirBnB, and the popular mobile magazine app Flipboard.

Amazon ([AMZN](#)) [blamed the outage](#) on glitches with a single networking device—what it called a “grey’ partial failure” that resulted in data loss—and said it was conducting a forensic analysis. The entire incident lasted all of 49 minutes, but like many recent cloud service outages, the resulting questions are likely to last considerably longer. Why are so many prominent Web companies overly dependent on a single cloud provider—and a single data center?


Amazon and other cloud providers preach the virtues of geographical redundancy: They say customers should spread out their services among multiple data centers, so if one goes down, another can pick up the slack. Sunday's outage, like so many other recent cloud service snafus, demonstrates that few cloud customers are properly following this orthodoxy and that true redundancy may be much more complicated than it sounds.

# Google: Dual network failure behind Monday's Gmail outage

**Summary:** *A failure of two separate, redundant network paths resulted in a service disruption to Gmail.*



By [Chris Duckett](#) | September 26, 2013 -- 00:54 GMT (17:54 PDT)

 [Follow @dobes](#)

For almost 12 hours on Monday, Google's Gmail service [suffered from delivery delays](#) and a backlog of messages due to the failure of two unrelated network paths.

The outage began at 5.54am PST, and the backlog of messages was cleared by 4.00pm PST, Google said [in a blog post](#). The company said that 71 percent of messages had no delay, and that the average delay on the 29 percent of affected messages was 2.6 seconds.

Significant delays of more than two hours did occur for 1.5 percent of messages, and Google said that "users who attempted to download large attachments on affected messages encountered errors".

The company said that although the failure of two separate, redundant network paths is a rare event, it is looking to improve network capacity and make message delivery more resilient to a loss in network capacity.

"We're updating our internal practices so that we can more quickly and effectively respond to network issues," wrote Sabrina Farmer, senior site reliability engineering manager for Gmail.

"We'll be working on all of these improvements and more over the next few weeks — even including this event, Gmail remains well above 99.9 percent available, and we intend to keep it that way!"

*Topics:* [Cloud](#), [Google](#), [Google Apps](#)





## More Details on Today's Outage

By Robert Johnson on Thursday, September 23, 2010 at 5:29pm

Early today Facebook was down or unreachable for many of you for approximately 2.5 hours. This is the worst outage we've had in over four years, and we wanted to first of all apologize for it. We also wanted to provide much more technical detail on what happened and share one big lesson learned.

The key flaw that caused this outage to be so severe was an unfortunate handling of an error condition. An automated system for verifying configuration values ended up causing much more damage than it fixed.

The intent of the automated system is to check for configuration values that are invalid in the cache and replace them with updated values from the persistent store. This works well for a transient problem with the cache, but it doesn't work when the persistent store is invalid.

Today we made a change to the persistent copy of a configuration value that was interpreted as invalid. This meant that every single client saw the invalid value and attempted to fix it. Because the fix involves making a query to a cluster of databases, that cluster was quickly overwhelmed by hundreds of thousands of queries a second.

To make matters worse, every time a client got an error attempting to query one of the databases it interpreted it as an invalid value, and deleted the corresponding cache key. This meant that even after the original problem had been fixed, the stream of queries continued. As long as the databases failed to service some of the requests, they were causing even more requests to themselves. We had entered a feedback loop that didn't allow the databases to recover.

The way to stop the feedback cycle was quite painful - we had to stop all traffic to this database cluster, which meant turning off the site. Once the databases had recovered and the root cause had been fixed, we slowly allowed more people back onto the site.

This got the site back up and running today, and for now we've turned off the system that attempts to correct configuration values. We're exploring new designs for this configuration system following design patterns of other systems at Facebook that deal more gracefully with feedback loops and transient spikes.

We apologize again for the site outage, and we want you to know that we take the performance and reliability of Facebook very seriously.

# 500 Internal Server Error

Sorry, something went wrong.

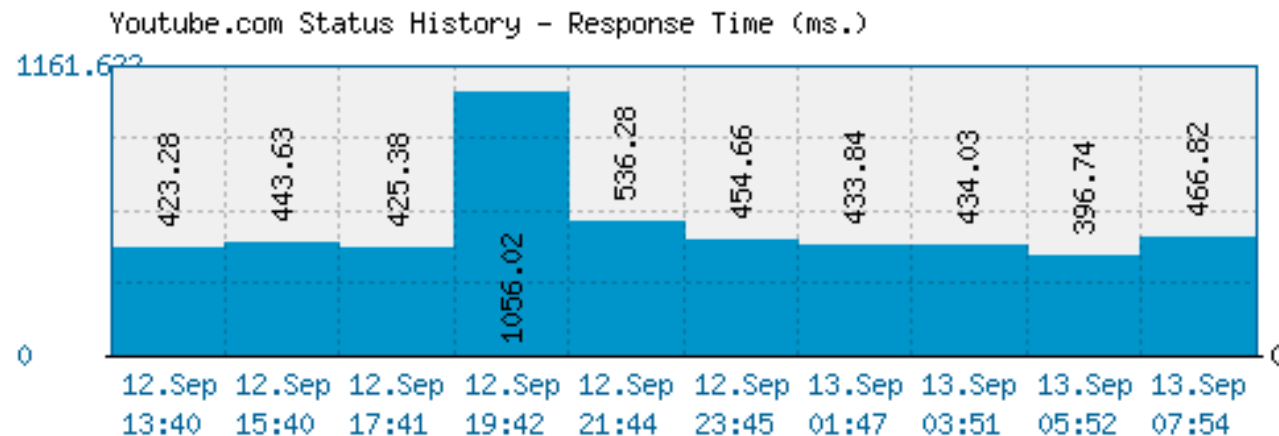
A team of highly trained monkeys has been dispatched to deal with this situation.

If you see them, show them this information:

WBFD TUTsEmNbgIfLqIvDbIonEUWGcL4fMg4DYRWanxvuE4l\_eS8HVAdIAjLq  
ljzNzQyrv1N0lyIcMCCxYPUuXeH4MpkwwNKtBA\_lkB0k82qpRcBvqgh3K4E9  
scHms6VB0dfBuF3fzeABI4CYXLALdGI0or9rOUTsFnaVWjl415RETaDqHIZ9  
O3jyfIL3Ex3jrau4YMRkqirCjqSdGvAYfTO1mimYb7IRInG-cOwoDhTYMlEv  
MIkp9RlELfHGu9KHZTSB\_EqGBrNd40xhl9xhTwyVRKLLoKHjhVKm06BadZHJ  
5uOyBLrY8j\_S54Fxrkr3vhE0BcDtXRR0NuiIb3cfZ5DSyIJo25AG0m5A3VzY  
kBAN\_vNqIUgbA2SKGxafPmw9kMKbQ5xcn3Dv5lTsIq\_8DNW9gKivYsiMvLfV  
YYpBvxkpgOXTtmCmx\_pCM6gTemaFDpyloI4bj2nZ5Lh0\_4fBgQd82MZqkuUk  
fXabM3axHm17oGm7cC\_d-L\_WEQBClg86SyaBSTXcjGjw-uvGZ5tyvEY-wHb7  
BrQB7Ksf00XWhrwwXK5Pq\_xYSCaOPybzRhjNysmdrzYKILlA7-g0CenLABts  
msPvMXKeE1Ak5LEMK1ZTIEiJaI\_kGzmBAKF63xik9N0O\_IScPlnftRyPFT9Y  
dg506aROJz6tlptiXA1DTBUcIkEuktQbKl5WU3h63ewuhovm7sTbNzI-RNiX

\*\*\*

## Youtube Website Status History



The above graph displays service status activity for Youtube.com over the last 10 automatic checks. The blue bar displays the response time, which is better when smaller. If no bar is displayed for a specific time it means that the service was down and the site was offline.

### Service Status History :

<u>Date</u>	<u>Time</u>	<u>Ping Time</u>	<u>Date</u>	<u>Time</u>	<u>Ping Time</u>
✓ 12.Sep.2013	15:40	443.63 ms.	✓ 12.Sep.2013	17:41	425.38 ms.
✓ 12.Sep.2013	19:42	1056.02 ms.	✓ 12.Sep.2013	21:44	536.28 ms.
✓ 12.Sep.2013	23:45	454.66 ms.	✓ 13.Sep.2013	01:47	433.84 ms.
✓ 13.Sep.2013	03:51	434.03 ms.	✓ 13.Sep.2013	05:52	396.74 ms.
✓ 13.Sep.2013	07:54	466.82 ms.	✓ 13.Sep.2013	09:55	546.39 ms.

\* Times displayed are PT, Pacific Time (UTC/GMT -7) | Current server time is 11:08

We have tried pinging Youtube website using our server and the website returned the above results. If youtube.com is down for us too there is nothing you can do except waiting. Probably the server is overloaded, down or unreachable because of a network problem, outage or a website maintenance is in progress...

# Pakistan causes YouTube outage for two-thirds of world

NEW YORK

By Peter Svensson



Most of the world's Internet users lost access to YouTube for several hours Sunday after an attempt by Pakistan's government to block access domestically affected other countries.

The outage highlighted yet another of the Internet's vulnerabilities, coming less than a month after broken fiber-optic cables in the Mediterranean took Egypt off line and caused communications problems from the Middle East to India.

An Internet expert explained that Sunday's problems arose when a Pakistani telecommunications company accidentally identified itself to Internet computers as the world's fastest route to YouTube. But instead of serving up videos of skateboarding dogs, it sent the traffic into oblivion.

On Friday, the Pakistan Telecommunication Authority ordered 70 Internet service providers to block access to YouTube.com, because of anti-Islamic movies on the video-sharing site, which is owned by Google.

The authority did not specify what the offensive material was, but a PTA official said the ban concerned a trailer for an upcoming film by Dutch lawmaker Geert Wilders, who has said he plans to release a movie portraying Islam as fascist and prone to inciting violence against women and homosexuals.

The block was intended to cover only Pakistan, but extended to about two-thirds of the global Internet population, starting at 1:47 p.m. ET Sunday, according to Renesys Corp., a Manchester, N.H., firm that keeps track of the pathways of the Internet for telecommunications companies and other clients.

The greatest effect was in Asia, where the outage lasted for up to two hours, Renesys said.

YouTube confirmed the outage on Monday, saying it was caused by a network in Pakistan.

# Thought Experiment

- Your web site runs on a small instance on Amazon EC2 and is becoming overloaded, what do you do?

# Course Project

## ■ Goal

- gain hands-on experience in building and deploying a scalable web service on the internet

## ■ Ideal Characteristics

- use Ruby + Rails + host your site on Amazon's Elastic Compute Cloud
- leverage data from a web service (Google maps, Amazon books, Yahoo local, Flickr, ...)
- build a transactional web site around the data
- implement user accounts, authentication, keep track of (http-) session data, perform some form of atomic transactions
- scale the application to multiple front-end servers
- load test it for many users and lots of data.
- Session data will need to be distributed, transactions will need to remain atomic.

## ■ Agile Process

- We will split into agile project team of 4 students
- Use weekly sprints to make progress
- Use Scrum, TDD, Pair programming

## ■ Project suggestions

- Please come up with your own suggestions of great, simple, but useful Internet services.
- Is there anything that you wish you had and can be done in 10 weeks?
- Review the suggestions on the class website
- Add your own suggestions

# Project examples (2011)

- Stab.rs - Voting plugin for browsers to down/up vote anything on the web
- bookSmart – Online book exchange for students
- Srvivr – Zombie apocalypse social network with heatmaps and escape routes
- Faceoff.co – Social gaming platform with player video chat



# Project examples (2012)

- Soundmap – Social sharing of location based sounds
- Carpoolers – Ride sharing site that allows users share cost of gas and reduce environmental impact of trips
- FriendAppMatcher – Receive recommendations for mobile apps from your friends
- Swapmeet – Trusted network barter system
- Ninja Challenge – Quiz and testing system



# Administrative Details

- Lab – M 5pm-7pm – Phelps 1401
- Class website at cs290.com
  - ◆ Filled out student info notecards so that we can give you edit access
- Buy book "Agile Development with Rails 3.2"
  - ◆ PDF book online for \$19 with coupon (normally \$27).
  - ◆ Discount code UCSBcs290b
- For class on Wed. Oct. 2<sup>nd</sup>
  - ◆ Install Ruby 2.0 and Rails 3.2 (follow description from Chapter 1 of book)
  - ◆ Work through Ruby Koans - <http://rubykoans.com/>
  - ◆ Add project ideas
- Class Schedule
  - ◆ Guest lectures: Amazing opportunity to learn from industry experts
- Grading policy

# Other Questions?