Handling Flash Crowds from your Garage

Jeremy Elson, Jon Howell
Microsoft Research

presented by:

Dmitrijus Bugelskis
Vrije Universiteit Amsterdam
March 2, 2012
Overview

- Flash crowds
- Building blocks of a scalable system
- Scaling architectures
  - How do we actually scale?
  - What are the implications?
- Real life examples
- Conclusions
What is a Flash Crowd?¹

- Large masses of readers at your HTTPD

http://en.wikipedia.org/wiki/Slashdot_effect
What is a Flash Crowd?²

Reasons:

- Being mentioned on a popular site (/. effect)
- Advertisement campaign
- Other reasons
Why is this bad?

- Our goal!
Why is this bad?

- Our goal!
- Overloads our website:
  - Loss of readers (potential clients)
Why is this bad?

- Our goal!
- Overloads our website:
  - Loss of readers (potential clients)
- Unpredictable (mostly)
- Sudden
- Temporary (several hours)
Let’s make a scalable web service!
Essential properties

- Low overhead during the lean times
- Highly scalable
- Quickly scalable
- Cheap!
Introduction

Utility computing:

- Packaging of computing resources as a metered service
- Related to Cloud computing
- Relies on statistical multiplexing
SDN

- Storage Delivery Networks
- aka Cloud Storage
- Amazon S3, Nirvanix, Google Cloud Storage
- Similar, but more attractive than CDNs
Compute Clouds

- Amazon EC2
- Simple programmatic API
- Billed by the hour
- EC2 has a free tier!
A missing piece: relational databases

- Hardly scalable
- ... unless we abandon full transactionality
- Replace relation databases altogether:
  - NoSQL
  - Amazon S3
  - Amazon SimpleDB
  - MongoDB
  - Many others
But how do we actually scale?
Using bare SDN

- Unlikely failures (not our problem)
- Scaling up and down is not our problem
- The capacity is virtually infinite
- Static HTTP
HTTP redirection

- Redirects users to a specific machine
- Frontend only involved in new session establishment
- Single point of failure, but only for new sessions
- Scale up time: minutes
EC2 HTTP redirection experiment

- If CPU capacity < 50%, launch a new server
Introduction

Problem

Building blocks

Scaling architectures

Real life examples

Conclusions

L4/L7 Load balancing

- L4 aka DNAT
- L7 aka reverse HTTP proxy
- Bandwidth is limited by the frontend
- Single point of failure
- Scale up time: minutes
L4/L7 Load balancing

- L4 aka DNAT
- L7 aka reverse HTTP proxy
- Bandwidth is limited by the frontend
- Single point of failure
- Scale up time: minutes
- Need to worry about client-server affinity
  - L7: HTTP cookies
DNS load-balanced clusters

- Multiple A records (max. 25)
- No single point of failure
- Unpredictable client-server affinity
- Badly-behaved resolvers
- Slow scale-up time: tens of minutes
  - Server startup time
  - TTL, Cache
Hybrid approaches?!
What about real life?
MapCruncher¹

- AJAX-style interactive maps
- Output: a set of static content (images, js)
- Single powerful webserver
- Sample data: 25 GB of maps
MapCruncher²

- Press release from Microsoft (2006)
  - Unresponsive and failing service
MapCruncher\(^2\)

- Press release from Microsoft (2006)
  - Unresponsive and failing service
- Little locality of reference - fs cache is useless
- Disk-IO bound
- Solution: publish data to Amazon S3
  - Problem solved!
  - $4/month for 25GB storage
  - Flash crowd: one time charge of $200
  - Statistical multiplexing rules!
Assira

- CAPTCHA: requires image identification
- Many servers on Amazon EC2
- Uses DNS load balancing
- BerkeleyDB file is pushed to every server
- Servers store session data locally, on disk
  - client-server affinity issues
  - session request are forwarded to the right server
- Survives flash crowds
- Survives DoS attacks
Lessons learned from Assira

- Poor client affinity is not a big problem
- EC2 nodes fail:
  - And get different IP on a reboot
  - Already fixed!
InkblotPassword.com\textsuperscript{1}

- Similar to Assira
- \textbf{Laziness} as the new philosophy:
  - virtually no time spent on optimizing
  - it is cheaper to just rent a new server
- Store everything in S3, no local data
- Slow S3 writes are hidden (write-behind)
InkblotPassword.com^2

- An article on several popular tech sites
- No automatic expansion = unresponsive system
- Fixed by adding more servers (20 min.)
- Marginal cost < $150
- 4 different scaling strategies there discussed
- All of them are available to the garage innovator
No single scaling strategy dominates!

Questions?