How we survived Hurricane Sandy

a look at what not to do

Dan White
@airplanedan

Mike Zaic
VP, Engineering, CafeMom
@focustrate
Introduction
Calm before the Storm
Into the Storm
Lessons Learned
CafeMom

- Founded: 2006
- Social Network for moms
- > 3 million members
- > 10 million monthly UVs
- ~ 130 employees
  - Tech: 13 people
  - Sales/Community: bulk of people
Introduction

Calm before the Storm

Into the Storm

Lessons Learned
Our Application(s)

- CafeMom
  - profiles
  - answers
  - advertising
  - sponsorships
  - advice
  - email

- MamasLatinas
  - stats
  - photos
  - admin tools
  - language switching
  - Que Mas
  - groups

- The Stir
  - video
  - games
DevOps Focus and Pace

• Sales Driven!
• Limited resources
• Product shifting
• Priorities – refactoring?

Photo: árticotropical (flickr)
Architecture

• Hosting
  – Single physical datacenter
  – Redundancy? pffff

• Cloud Presence
  – Limited to specific parts of the app
  – No database required
Infrastructure Growth

2006

minimal hardware

rapid growth

= infrastructure expands

2013

continued growth

= optimize for scale

stagnant infrastructure + more functionality

= tightly coupled birds nest

slower growth

= stagnant infrastructure
“If NYC has serious issues we’ll all have more important things to worry about...”
Introduction

Calm before the Storm

Into the Storm

Lessons Learned
Day 1: Bring on Sandy!
Day 2: waiting and hoping
Day 3: false hope

Photo: raneko (flickr)
Day 4: code with no data
Day 5: generator anticipation
Day 6: generator #fail
Day 7: everything... works?
Introduction

Calm before the Storm

Into the Storm

Lessons Learned
Technical Takeaways
what we did

• Redundancies
• Code without data
• DNS propagation
• Cloud replication
  — Codebase
  — Database
Technical Takeaways

next steps

• Architecting for degraded service?
• Simplify application?
• Second physical site?
• Geographic redundancy?
• Undoing "fixes" of fail-overs?
Overarching Problems

- Don’t trust vendor/provider assurances!
- Managing expectations during an outage
- Feelings of helplessness
- Scrambling doesn't necessarily get things up faster
- Must know downtime tolerance
- Cost of DR planning vs Lost opportunity of outage
- DR Planning is insurance
Know when to say when

• When is it appropriate to plan for DR?
• When is it appropriate to build for DR?
• When is it appropriate to retrofit DR for existing app?
• When is it appropriate to dictate product requirements based on ease of DR?
• During an outage - when is it appropriate to do nothing (and just sleep)?
When not a great time to think about DR?

During a disaster.

(unless you’re thinking about the next one)
Moving forward – inertia sucks!

- Change is tough
- Mature infrastructure/app resists drastic change
- Pace of development inertia against stepping back and revisiting
- Pace of business inertia against non-revenue projects
- No recent disasters inertia against DR necessity
<table>
<thead>
<tr>
<th>#</th>
<th>Priority</th>
<th>Description</th>
<th>Category</th>
<th>Type</th>
<th>Assignee</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>#8424</td>
<td>normal</td>
<td>DR site: improve launch procedures and ensure all core functionality is present</td>
<td>General</td>
<td>strategic</td>
<td>danw</td>
<td>2013-03-04</td>
</tr>
</tbody>
</table>

baby steps
thank you

http://www.slideshare.net/focustrate/how-we-survived-hurricane-sandy
thank you

http://www.slideshare.net/focustrate/how-we-survived-hurricane-sandy
thank you

http://www.slideshare.net/focustrate/how-we-survived-hurricane-sandy