What I've learned about scalable apps

UCSB CS 290 • November 20, 2014

Colin Kelley
CTO, Cofounder • Invoca, Inc.

colin@invoca.com

My background

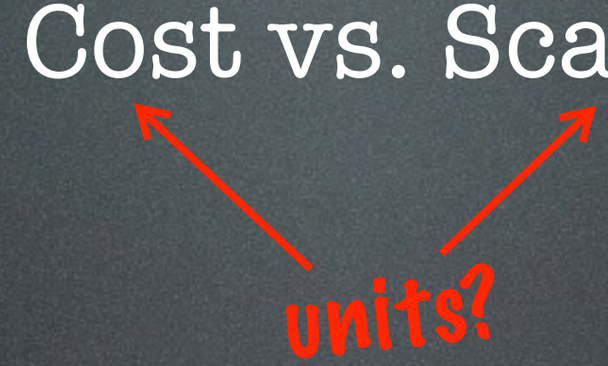
Digital Sound Corp / PulsePoint / Unisys
‘90s: Software Architect

CallWave, Inc ‘00s: CTO

RingRevenue, Inc ’08 +: Cofounder/
CTO

RingRevenue becomes Invoca, 2013

Scaling distribution



Complexity = O(n)

Cost = Density × O(n)

Digital Sound Corp

96 port voicemail system

- Intel 80386 133 MHz
- 8 MB RAM
- Custom Unix System V w/cooperative multi-threading

Challenge: cluster state-wide servers with distributed phone directory

Predecessor version

- Built for generality
- Disk intensive

  Tested for < 1,000 phone numbers. Fell over when asked to scale to 10,000.

  Why? $O(n) = n^2$!

Redesigned

- Network model database
- Clustered storage
- B+Tree indexed
- Most Recently Used disk cache

  Scaled to 1 million phone numbers

  Why? O(n) = log(n)

Generality

- ★ Generalize only if it simplifies.
- ★ Otherwise, trust agility.
  Generalize later!

- ★ Know what metrics determine your density & limit your scalability
- ★ Complexity O(n) matters most

Optimizing density

- Profiling discovered that ~20% of CPU spent calculating time(); ~15% of CPU spent copying static voice prompts
- Optimization: separate thread to get the time ~once / second
- Optimization: cache prompts
- 96 ports → 120 ports 1.25X cheaper

Optimization...

hinders evolution

Therefore:

~~Don't do it!~~

~~Do it later!~~

★ Do it only when you have the need and have the data

CallWave architecture

Clients:

- Windows EXE, web browser, phone

Servers:

- Session Managers: stateful non-persistent / partitioned
- Web Servers: stateless
- Call Managers: stateless
  + Media Store: stateful / partitioned
- Database: stateful persistent /
  partitioned Customer vs. Message
  primary + standby + reporting

Budget Stalemate Resolved

Legislature Raises Sales Tax 1%

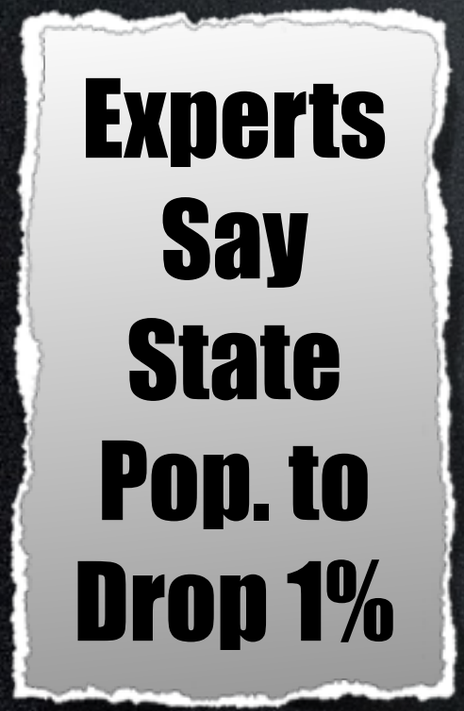
Experts Say State Pop. to Drop 1%

Scaling factor (X)

7.25% → 8.25%

Is that +1%? Or +14%?

Scaling factor is 1.14X



100% → 99%

Scaling factor is .99X

Net change: 1.14 × .99 = 1.13X

Measure change

"CPU usage changed by 20%"

- Assuming it started at 50%, what did it change to?

Formula	=	X factor
50% - 20%	30%	1.67X
50% + 20%	70%	.71X
50% *80%	40%	1.25X
50% / 80%	62%	.80X
50% *120%	60%	1.20X
50% / 120%	42%	.83X

Sales funnels

Change lessons

- Express change as X factor
- If someone tells you change as a %, ask them restate as before → after values.

Scalability: Statelessness, like Ignorance, is Bliss

- Stateless servers are trivial to scale horizontally. Linear.

★ Caching still “stateless” (empty cache should be slow/harmless), but don’t forget:

  cache refresh / coherency

  priming...

Scalability 2: Statefulness is a necessary evil

- ★ Isolate state. Further isolate persistent state (database, media...) from non-persistent (session...)
- ★ Partition stateful servers when possible to scale horizontally
- Worst case: scale vertically... until you can't.

Availability

- No single point of failure: State partitioned or redundant
- ★Prefer Active/Active over Active/Passive
- ★Watch for correlated failure (e.g.: routers coordinated with BGP; network providers using same backbone; EBS using same S3)
- ★End-to-end failover is necessary **and sufficient**
- ★Without monitoring all systems lose their redundancy over time!

Lessons learned

- Invest in architecture. Use cheap commodity hardware.
- Don’t own / manage your own infrastructure
- Don’t prepay for scale based on Sales projections :-)
- Monitoring is **really** difficult

Moore's Law

- Computer power doubles every (1.5 -) 2 years; or
- For given power, cost drops in half every 2 years; or
- Annual Cost Of Goods Sold (COGS) improvement = 1.41X

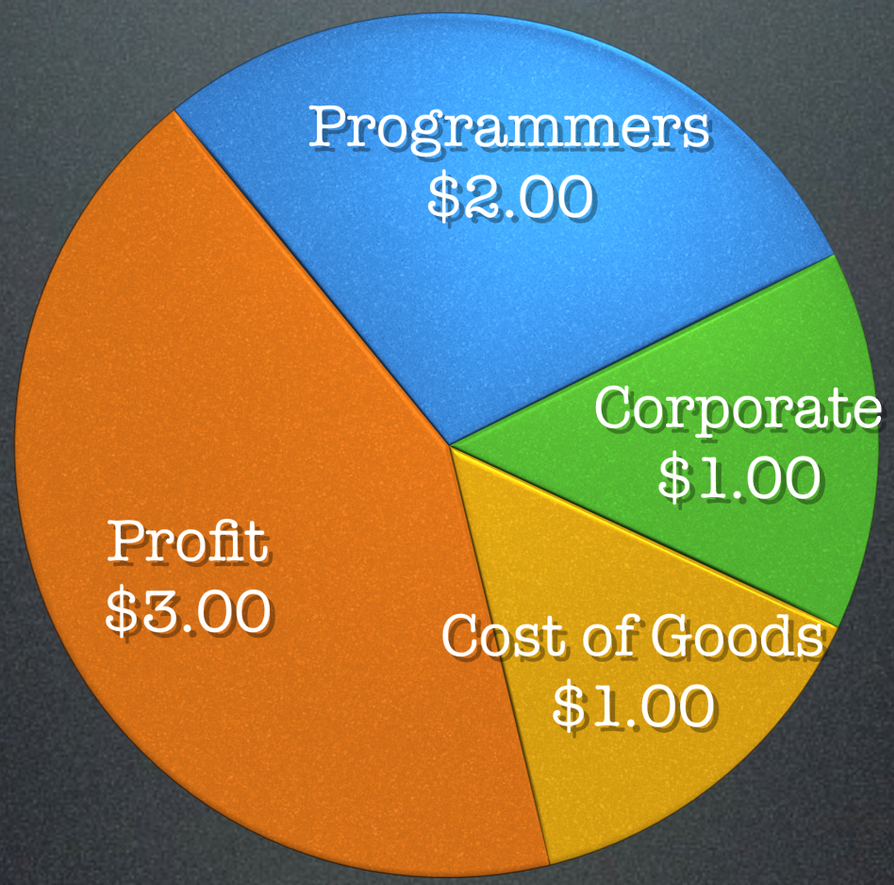
Moore's Law meets Software as a Service

Example: $7 monthly service

Year	Cost /user/month
2014	$1.00
2015	$0.71
2016	$0.50
2017	$0.36
2018	$0.25

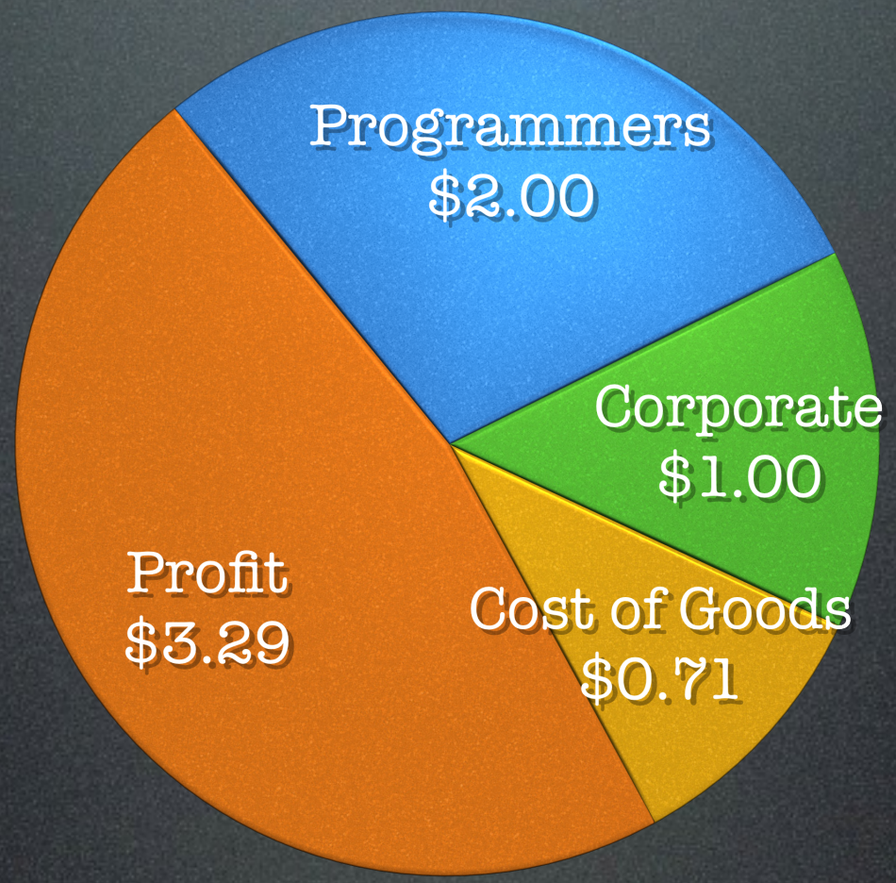
2014

$7 monthly service



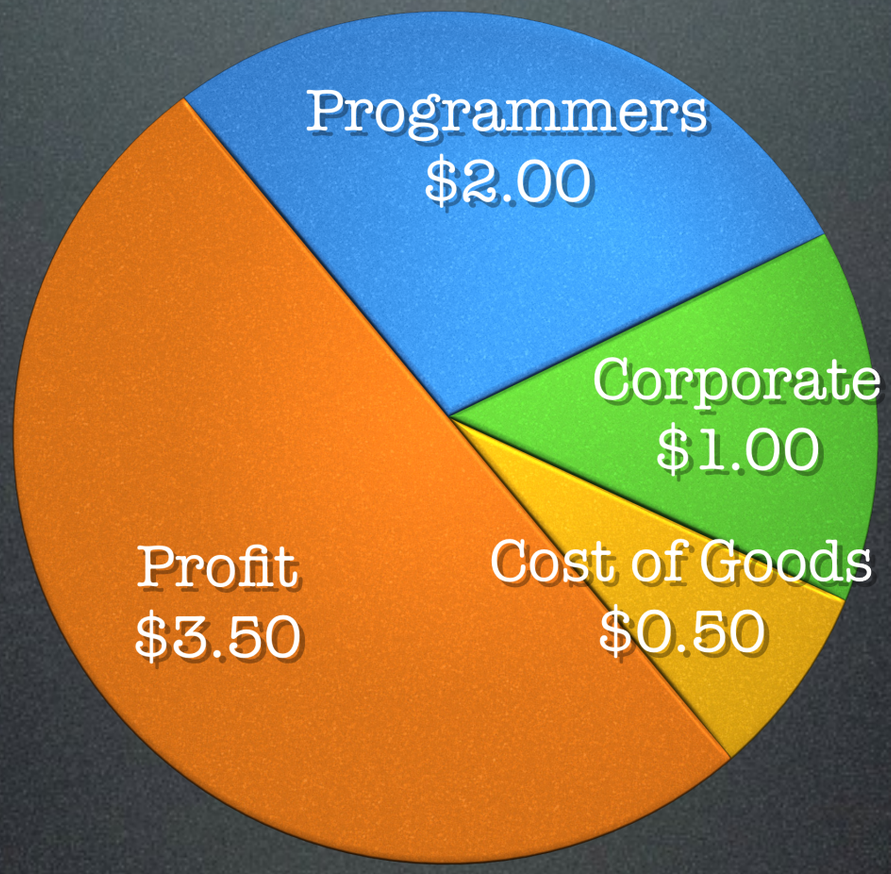
2015

$7 monthly service



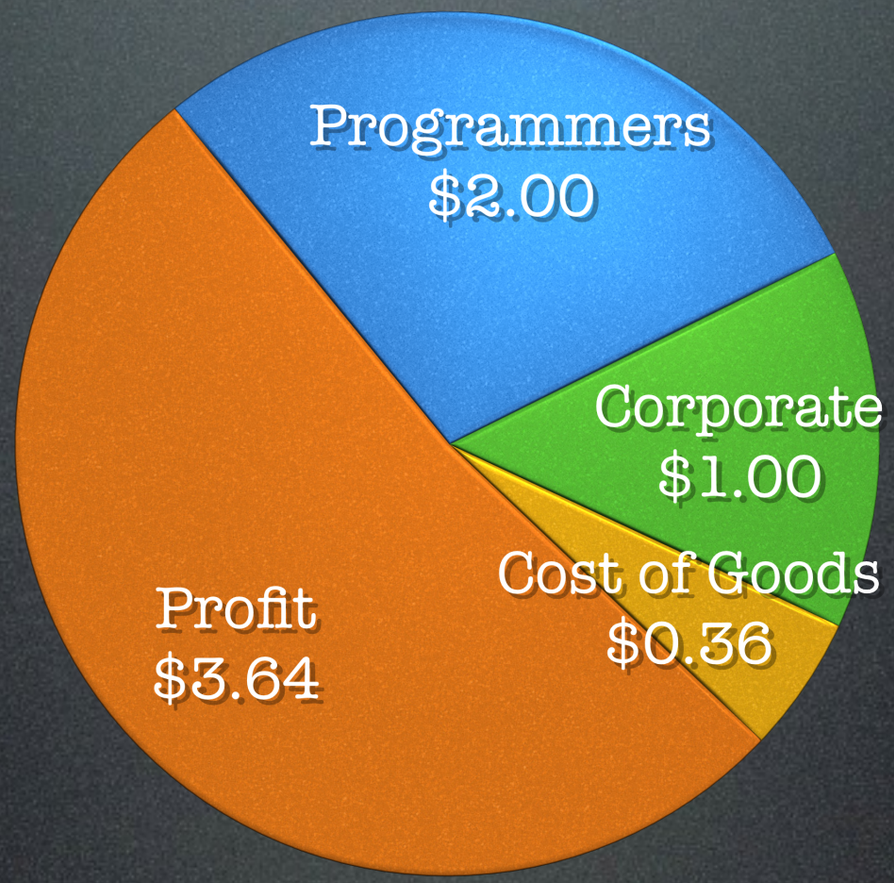
2016

$7 monthly service



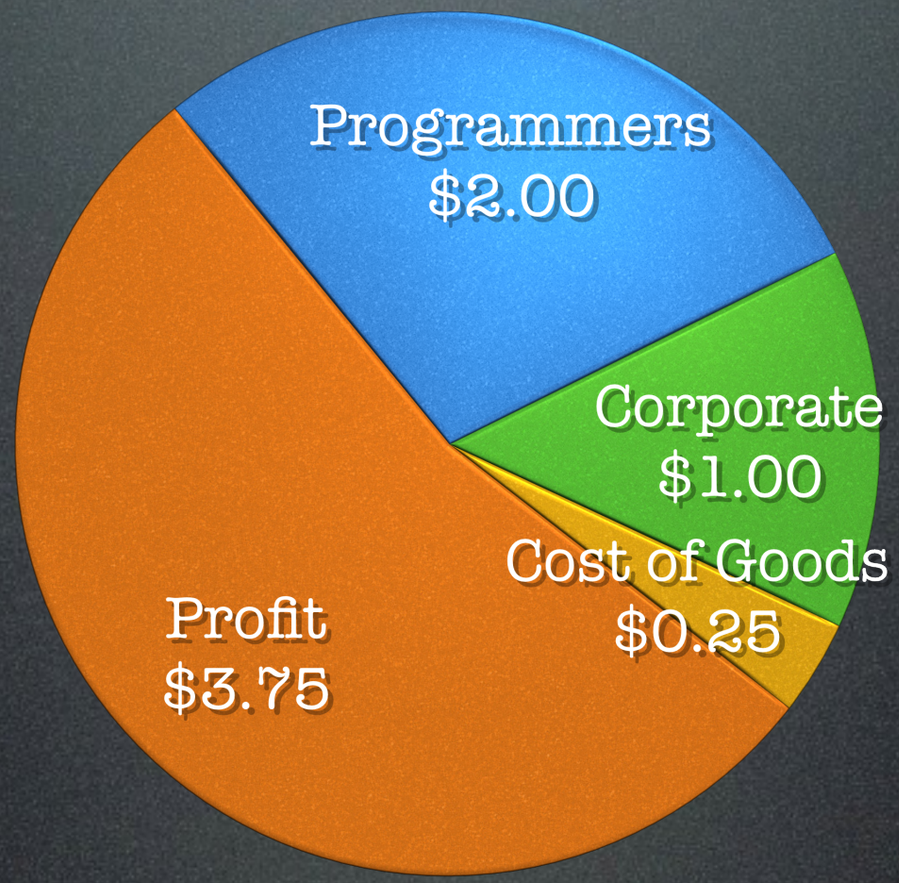
2017

$7 monthly service



2018

$7 monthly service



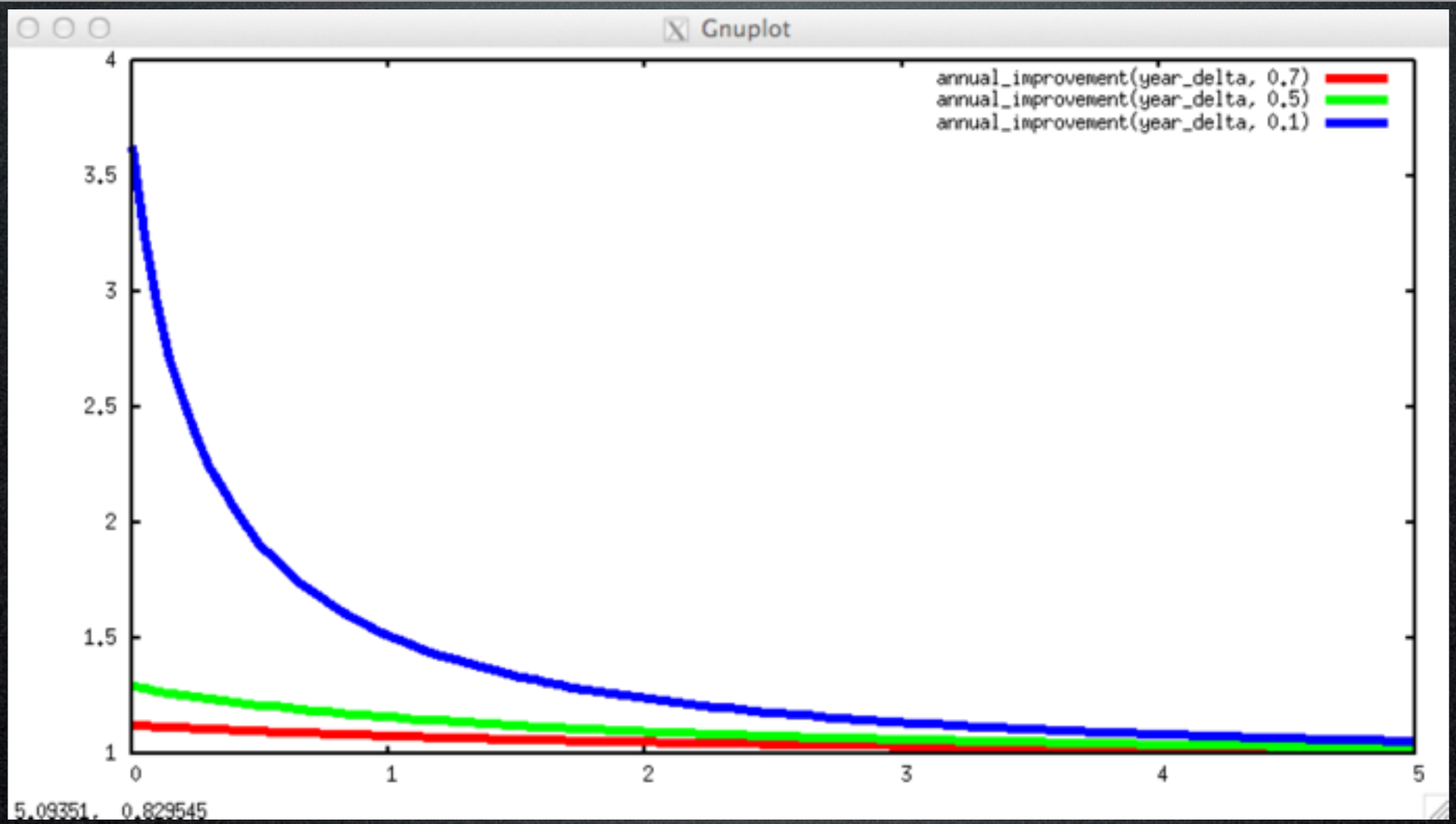
Moore's Law meets Software as a Service

Example: $7 monthly service

Year	Cost /user/month	Gross profit margin	Annual improvement
2014	$1.00	86%	
2015	$0.71	90%	1.05X
2016	$0.50	93%	1.03X
2017	$0.36	95%	1.02X
2018	$0.25	96%	1.01X

Annual Improvement

	Commodity (blue)	Innovation (red)
Typical gross margin	10%	70%



Invoca

- Web Servers (Rails)
- MySQL Database
    - transactional
    - tabular
        - + data warehouse
- Cassandra: transient, rapid, write-heavy
- Load Balancer
- Redis for caching
- Telephony Servers
- Cross-Region Proxy

Why Ruby on Rails at Invoca?

- Model / View / Controller architecture
- Agile/XP built in: automated unit, functional, integration testing
- Modern (e.g. Ajax and CSS not after-thoughts)
- SB: AppFolio, RightScale, RightSignature, BioIQ, InTouch Health, Procore...
- Fun!

Invoca lessons learned so far

- ★ Expressiveness = productivity
- ★ Meta-programming rocks
- ★ Ruby with its open (duck-punchable) classes is incredible for open source
- ★ Cloud computing deserves the hype!
- ★ Plan for scale but pay for it as you go

?